Questo testo raccoglie lezioni da me svolte presso l'Università degli Studi di Napoli Federico II, che non sono state riportate nella Parte prima. Gli argomenti trattati in questo testo (Parte prima e Parte seconda) rappresentano una base minima, ma indispensabile, per chi intende lavorare, in modo consapevole, nel settore del calcolo scientifico ("Scientific Computing").
D'altra parte, il progresso continuo di questa disciplina, fortemente influenzata dal quotidiano progresso della tecnologia, richiede sempre più la conoscenza degli strumenti e delle metodologie di base.

Indice

Capitolo 1

Risoluzione numerica di equazioni non lineari

1.1 Introduzione

Questo capitolo è dedicato all'analisi di alcuni algoritmi numerici per la risoluzione di equazioni non lineari. Assegnata una funzione non lineare $f : A \subseteq \Re \to \Re$, una equazione non lineare è un'equazione del tipo:

$$f(x) = 0 \, . \tag{1.1}$$

Risolvere l'equazione (1.1) equivale a calcolare i valori di $x \in [a, b] \subseteq A$ per i quali risulta $f(x) = 0$. Ogni valore di x per cui è soddisfatta la (1.1) è detto **radice dell'equazione** oppure **zero della funzione** f.

Spesso il modello matematico utilizzato per descrivere e risolvere problemi applicativi è costituito da equazioni non lineari e, generalmente, la difficoltà nel risolvere tali problemi si trasferisce direttamente alla difficoltà nella risoluzione dell'equazione non lineare associata. Si descrivono, nel seguito, alcuni esempi di problemi o applicazioni la cui risoluzione conduce ad una equazione del tipo (1.1).

♣ **Esempio 1.1.** Il problema della determinazione del minimo [massimo] di una funzione convessa [concava] $f \in C[a, b]$ è equivalente a quello della risoluzione di un'equazione non lineare:

$$f'(x) = 0. \tag{1.2}$$

Pertanto un capitolo fondamentale dei metodi numerici di ottimizzazione è costituito dalla risoluzione di sistemi di equazioni non lineari. ♣

♣ **Esempio 1.2.** La legge che regola la crescita di una popolazione è del tipo

$$p(t) = \frac{p_M}{1 + (p_M/p_0)e^{-kp_M t}} \tag{1.3}$$

dove p(t) fornisce il numero di individui presenti al tempo t, p_0 rappresenta la popolazione all'istante iniziale $p(t_0) = p_0$, p_M è il valore massimo raggiungibile dalla popolazione, mentre k è un prefissato fattore di crescita. Se si desidera conoscere in quale istante la popolazione raggiungerà un livello uguale a metà del massimo consentito, occorrerà risolvere l'equazione

$$\frac{p_M}{1 + (p_M/p_0)e^{-kp_M t}} - \frac{p_M}{2} = 0$$

nella variabile t. ♣

♣ **Esempio 1.3.** Si supponga di disporre di un capitale di 1000 euro da investire. Si assuma, inoltre, che il tipo di investimento prescelto frutti un interesse netto dell'otto per cento annuo e non sia consentito per frazioni di anno. Si vuol conoscere il tempo di investimento necessario (in anni) perché il capitale raddoppi.
Si osservi che al termine del primo anno il capitale sarà cresciuto dell'otto per cento e quindi sarà uguale a $1080 = 1000 \cdot 1.08$. Tale capitale, investito durante il secondo anno frutterà a sua volta un interesse dell'otto per cento; dunque, al termine del secondo anno, il capitale sarà uguale a $(1000 \cdot 1.08) \cdot 1.08 = 1000 \cdot (1.08)^2$. In generale, al termine dell' n-mo anno di investimento, il capitale ammonterà a $1000 \cdot (1.08)^n$. La risoluzione del problema si riconduce, quindi, a determinare il più piccolo intero $\overline{n}$ per il quale si verifichi:

$$1000 \cdot 1.08^{\overline{n}} \geq 2000.$$

ovvero a determinare la soluzione $\overline{x}$ dell'equazione

$$1000 \cdot 1.08^x - 2000 = 0$$

e calcolare successivamente $\overline{n}$ come minimo intero maggiore o uguale a $\overline{x}$. [1] ♣

♣ **Esempio 1.4.** In meccanica celeste, la posizione x di un pianeta nella sua orbita può essere determinata risolvendo un'equazione del tipo:

$$x - E \sin x - M = 0 \tag{1.4}$$

con E ed M costanti positive minori di 1. ♣

[1] Si osservi che da continuità, crescenza e codominio della funzione $f(x) = 1000 \cdot 1.08^x - 2000$ segue banalmente l'esistenza e unicità della soluzione del problema.

In tutti gli esempi precedenti l'equazione coinvolta è scalare. Nel caso generale, ovvero di una funzione di più variabili, cioè:

$$F: \ A \subseteq \Re^m \to \Re^n, \ \ m, n \geq 2,$$

l'equazione $F(x) = 0$ rappresenta un sistema di equazioni non lineari ed in tal caso la soluzione, quando esiste, è un vettore di m componenti.

♣ **Esempio 1.5.** Un esempio di sistema di equazioni non lineari è il seguente:

$$F(x_1, x_2) = \begin{pmatrix} x_1^2 + x_2^2 - 5 \\ x_1 + x_2 + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Una soluzione di tale sistema, di due equazioni non lineari, definita su tutto $\Re^2$, è $x^* = (1, -2)^T$. ♣

In seguito sarà considerato il caso scalare [2], con un breve cenno al caso vettoriale, nel §1.8.

L'esistenza ed il numero delle radici di equazioni non lineari scalari dipende non solo dalla funzione, ma anche dall'intervallo considerato. Ovvero, i dati del problema del calcolo delle radici di un'equazione non lineare sono la funzione f e l'intervallo di ricerca $[a, b]$.

♣ **Esempio 1.6.** Si consideri la seguente equazione:

$$x^4 - 12x^3 + 47x^2 - 60x = 0.$$

Le radici reali di tale equazione, definita su tutto $\Re$, sono 4 e precisamente $x^* = 0, 3, 4, 5$ (si osservi che la funzione considerata è un polinomio algebrico di quarto grado), tuttavia se si considera il problema di determinare le radici nell'intervallo $[-1, 1]$ l'equazione ha una sola soluzione ($x^* = 0$). Considerando invece l'intervallo $[-1, 3.5]$ l'equazione ha due radici ($x^* = 0, 3$). Mentre nell'intervallo $[1, 2]$, l'equazione non ha soluzioni. ♣

È importante osservare che anche quando si dispone dell'espressione analitica delle soluzioni, spesso non è possibile valutarla con un numero finito di operazioni.

[2] Tale scelta è di natura didattica ma è anche giustificata dal fatto che molti metodi per la risoluzione numerica di equazioni non lineari scalari sono estendibili al caso di sistemi.

♣ **Esempio 1.7.** L'equazione $\cos(\log(x)) = 0$ ha infinite radici nell'intervallo $(0, \infty)$, precisamente [3]:

$$x^* = e^{\pi/2+k\pi}$$

con k intero non negativo. Una valutazione delle stesse comporta delle approssimazioni. ♣

♣ **Esempio 1.8.** Si consideri il polinomio di quinto grado

$$f(x) = x^5 - 2x^4 - x^3.$$

Per il Teorema Fondamentale dell'Algebra, l'equazione (algebrica, in questo caso) non lineare $f(x) = 0$ ha 5 radici, e cioè i 5 zeri del polinomio. Per determinare tali soluzioni si può fattorizzare il polinomio nel prodotto di 2 polinomi, e precisamente $f(x) = x^3(x^2 - 2x - 1)$. L'equazione $f(x) = 0$ è, quindi, decomposta in due equazioni non lineari, $x^3 = 0$ e $x^2 - 2x - 1 = 0$, le cui soluzioni sono: $x_1 = 0$, con molteplicità 3, $x_4 = 1 - \sqrt{2}$, $x_5 = 1 + \sqrt{2}$. Nel caso del polinomio considerato si è riusciti a determinare gli zeri mediante una fattorizzazione in polinomi di grado più basso ma, in generale, non esistono formule per calcolare gli zeri di polinomi di grado maggiore o uguale a 5 (Teorema di Abel). ♣

♣ **Esempio 1.9.** Si analizzi la funzione seguente:

$$f(x) = xe^{3x} + x^3 + \log(x + 1) - 3.$$

Considerato l'intervallo $[0, 1.5]$, si ha che $f(0) = -3$ e $f(1.5) = 136.317$, e quindi la funzione assume agli estremi valori di segno opposto. Inoltre, si ha che $f'(x) = 3xe^{3x} + e^{3x} + 3x^2 + 1/(x+1) > 0$, $x \in [0, 1.5]$, cioè la funzione è strettamente crescente e, quindi, ha un unico zero nell'intervallo considerato. ♣

1.2 Il Metodo di Tabulazione

Un modo "naturale" per individuare i sottointervalli in cui possono trovarsi le radici di $f(x) = 0$ è indagare sui valori che la funzione f assume in un insieme finito di punti dell'intervallo. Scelti ad esempio n punti $t_1, \cdots, t_n$, analizzando i valori $f(t_i)$, si individuano i sottointervalli $[t_i, t_{i+1}]$ in cui la funzione assume agli estremi valori di segno opposto. Tali sottointervalli possono contenere uno o più zeri della funzione. Questa analisi consente, quindi, di delimitare le zone in cui possono cadere le soluzioni di $f(x) = 0$, eventualmente, poi, si può ripetere l'analisi in queste regioni fino a quando si riesce a individuare una delle soluzioni con una accuratezza soddisfacente.

[3] In seguito, dove non dichiarato diversamente, con *log* si farà riferimento al *logaritmo naturale*.

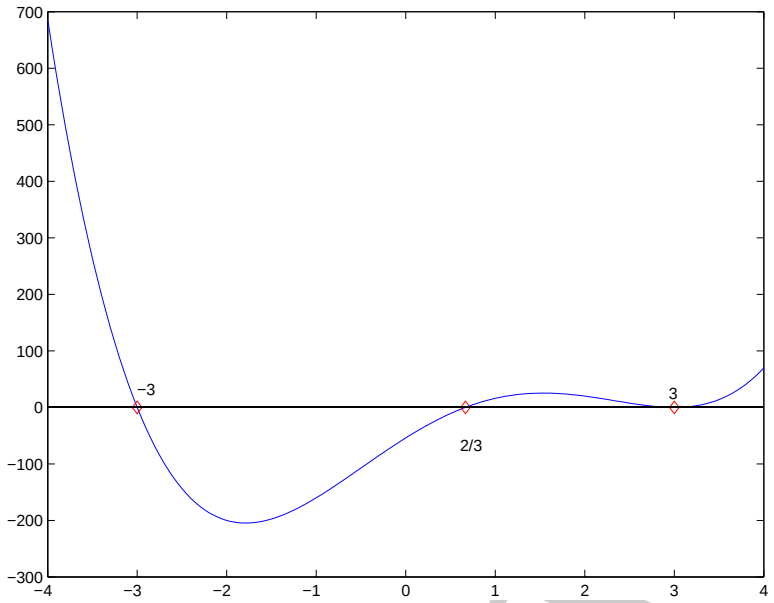

Figura 1.1: Grafico di $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$ in $[-4, 4]$. Con il simbolo '$\diamondsuit$' si indicano gli zeri della funzione.

♣ **Esempio 1.10.** Si consideri l'equazione seguente:

$$3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0. \tag{1.5}$$

In Fig. 1.1 è rappresentato il grafico di $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$.

Le radici sono $x_1 = -3$,$x_2 = 2/3$ e $x_3 = 3$ (quest'ultima è radice doppia). Si vogliono trovare le radici nell'intervallo $[-2, 2]$. Si considerano $n = 11$ punti equidistanziati, cioè i punti

$$t_i = -2 + (i-1) \cdot h, \quad i = 1, \ldots, 11, \quad con \quad h = 0.4,$$

e si valuta la funzione f in tali punti (Fig. 1.2).
In Tabella 1.1 sono riportati gli 11 punti considerati nell'intervallo $[-2, 2]$ ed i valori corrispondenti della funzione, approssimati alla quarta cifra significativa. Osservando tali valori possiamo dire che l'equazione (1.5) potrebbe avere una radice nell'intervallo $[0.4, 0.8]$ (qui la funzione assume valori di segno opposto agli estremi) (Fig. 1.3). Il punto medio dell'intervallo $[t_7, t_8]$, che è 0.6, potrebbe essere assunto come approssimazione della radice $x_2 = 0.666....$
Si può applicare di nuovo il procedimento appena descritto all'intervallo $[0.4, 0.8]$, considerando gli 11 punti

$$t_i = 0.4 + (i-1) \cdot h, \quad i = 1, \ldots, 11, \quad con \quad h = 0.04.$$

I valori che la funzione assume in tali punti, riportati in Tabella 1.2, mostrano che essa potrebbe avere uno zero nell'intervallo $[0.64, 0.68]$.
Se si arresta il procedimento a questo passo, si può assumere come approssimazione della radice dell'equazione (1.5) il punto medio dell'intervallo $[t_7, t_8]$, cioè 0.66, ottenendo, così un'accuratezza di 2 cifre significative. Se si desidera una accuratezza maggiore si può iterare il procedimento a partire dall'intervallo $[0.64, 0.68]$(vedi Tabella 1.3).

♣

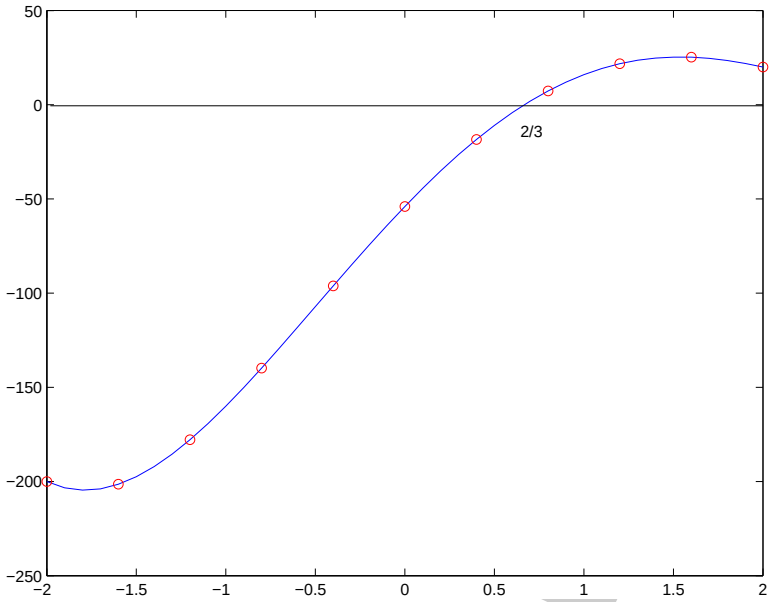

Figura 1.2: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico nell'intervallo $[-4, 4]$ è riportato in Fig. 1.1, in corrispondenza dei punti $t_i = -2 + (i-1)h,\ i = 1, \ldots, 11,\ h = 0.4$, appartenenti all'intervallo $[-2, 2]$.

t_i	$f(t_i)$
-.2000e+01	-2.000e+02
-.1600e+01	-2.014e+02
-.1200e+01	-1.778e+02
-.8000e+00	-1.397e+01
-.4000e+00	-9.617e+01
.0	-5.400e+01
.4000e+00	**-1.838e+00**
.8000e+00	**7.35e+01**
.1200e+01	2.177e+01
.1600e+01	2.524e+01
.2000e+01	2.000e+01

Tabella 1.1: Tabulazione di $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$ in corrispondenza dei punti $t_i = -2 + (i-1)h,\ i = 1, \ldots, 11,\ h = 0.4$, in $[-2, 2]$.

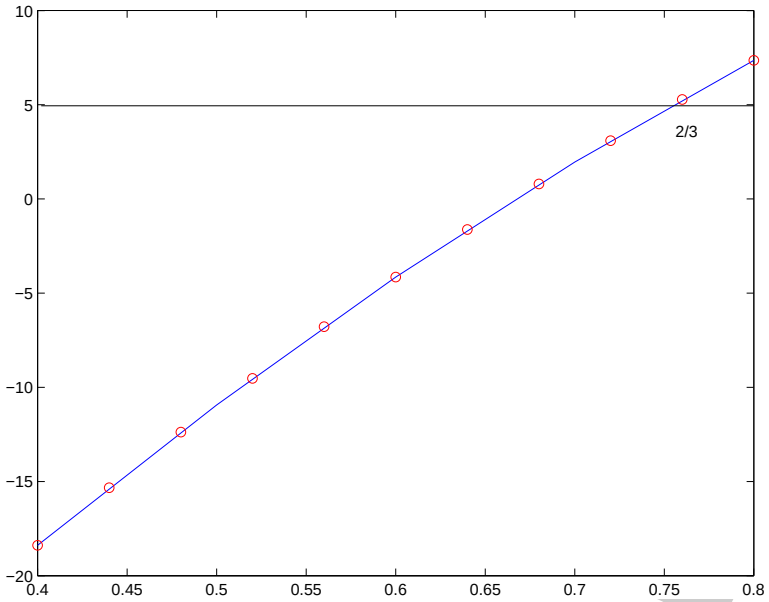

Figura 1.3: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico nell'intervallo $[-4, 4]$ è riportato in Fig. 1.1, in corrispondenza dei punti $t_i = 0.4 + (i - 1)h,\ i = 1, \ldots, 11,\ h = 0.04$, appartenenti all'intervallo $[0.4, 0.8]$.

t_i	$f(t_i)$
.4000e+00	-1.838e+01
.4400e+00	-1.5339e+01
.4800e+00	-1.237e+01
.5200e+00	-9.525e+00
.5600e+00	-6.782e+00
.6000e+00	-4.147e+00
.6400e+00	**-1.621e+00**
.6800e+00	**7.922e-01**
.7200e+00	3.094e+00
.7600e+00	5.282e+00
.8000e+00	7.356e+00

Tabella 1.2: Tabulazione di $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$ in corrispondenza dei punti $t_i = 0.4 + (i - 1)h,\ i = 1, \ldots, 11,\ h = 0.04$, in $[0.4, 0.8]$.

t_i	$f(t_i)$
0.6400e+01	-1.621e+00
0.6440e+01	-1.375e+00
0.6480e+01	-1.130e+00
0.6520e+00	-8.858e-01
0.6560e+00	-6.427e-01
0.6600e+01	-4.008e-01
0.6640e+01	**-1.599e-01**
0.6680e+00	**7.978e-02**
0.6720e+01	3.184e-01
0.6760e+01	5.559e-01
0.6800e+01	7.922e-01

Tabella 1.3: Tabulazione di $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$ in corrispondenza dei punti $t_i = 0.64 + (i-1)h,\ i = 1, \ldots, 11,\ h = 0.04$, in $[0.64, 0.68]$.

Tale procedimento è detto **Metodo di Tabulazione** e rappresenta il modo più semplice di determinare un intervallo contenente lo zero di una funzione definita in un intervallo $[a, b]$.

♣ **Esempio 1.11.** Si consideri l'equazione seguente:

$$x^2 - 0.09 = 0, \tag{1.6}$$

in Fig. 1.4 è rappresentato il grafico della funzione $f(x) = x^2 - 0.09$.
Le radici sono $x_1 = -0.3$ e $x_2 = 0.3$. Si vogliono trovare le radici nell'intervallo $[-4, 2]$. Si considerano $n = 5$ punti equidistanziati, cioè i punti

$$t_i = -4 + (i-1) \cdot h, \quad i = 1, ..., 5, \quad con \quad h = 1.5,$$

e si valuta la funzione f in tali punti (Fig. 1.5).
In Tabella 1.4 sono riportati i 5 punti considerati nell'intervallo $[-4, 2]$ ed i valori corrispondenti della funzione, approssimati alla quarta cifra significativa. In questo caso il Metodo di Tabulazione non rileva alcun intervallo in cui la funzione assume valori di segno opposto agli estremi. Si può, quindi, applicare di nuovo il Metodo di Tabulazione aumentando ad esempio il numero di punti di tabulazione, ovvero considerando gli $n = 9$ punti

$$t_i = -4 + (i-1) \cdot h, \quad i = 1, \ldots, 9, \quad con \quad h = 0.75.$$

I valori che la funzione assume in tali punti, riportati in Tabella 1.5, mostrano che il metodo individua due intervalli dove la funzione assume segno discorde agli estremi, ed in essi potrebbe avere uno zero (Fig. 1.6).

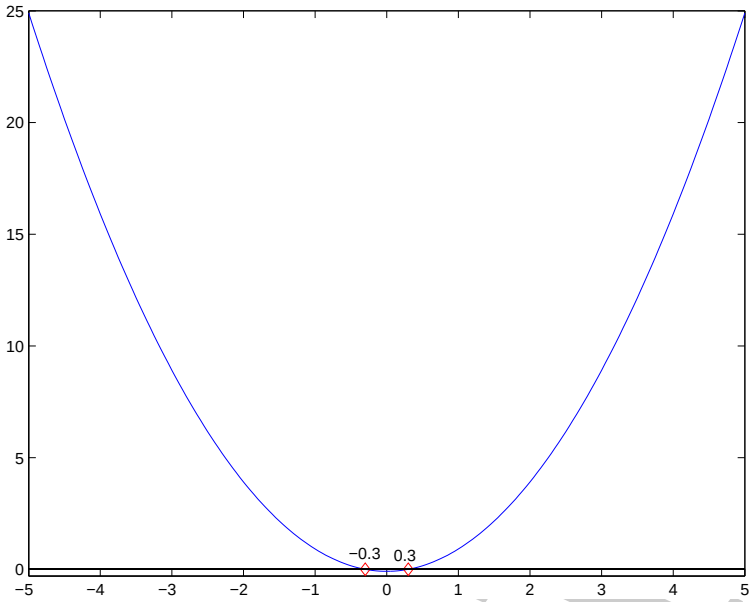

Figura 1.4: Grafico di $f(x) = x^2 - 0.09$ in $[-5, 5]$. Con il simbolo '$\diamondsuit$' si indicano gli zeri della funzione.

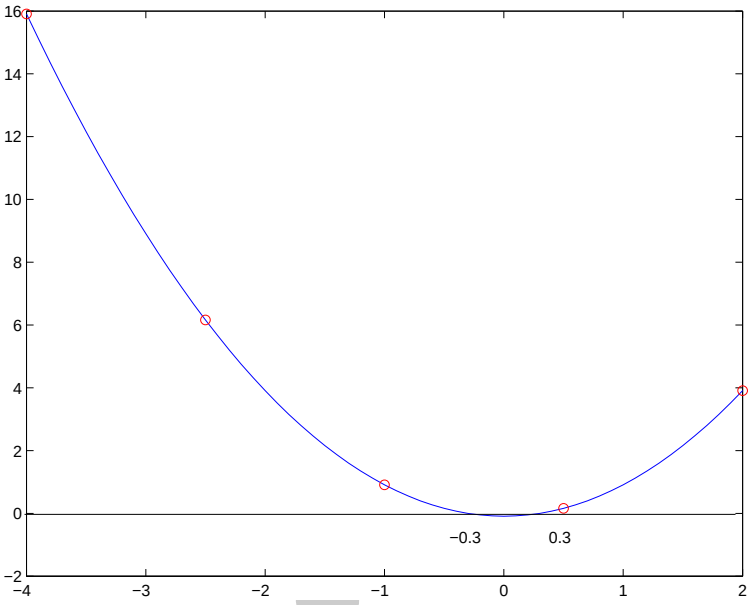

Figura 1.5: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico nell'intervallo $[-5, 5]$ è riportato in Fig. 1.4, in corrispondenza dei punti $t_i = -4 + (i - 1)h,\ i = 1, \ldots, 5,\ h = 1.5$, appartenenti all'intervallo $[-4, 2]$.

t_i	$f(t_i)$
-.4000e+01	.1591e+02
-.2500e+01	.6160e+01
-.1000e+01	.9100e+00
.5000e+00	.1600e+00
.2000e+01	.3910e+01

Tabella 1.4: Tabulazione di $f(x) = x^2 - 0.09$ in corrispondenza dei punti $t_i = -4 + (i-1)h,\ i = 1, \ldots, 5,\ h = 1.5$, in $[-4, 2]$.

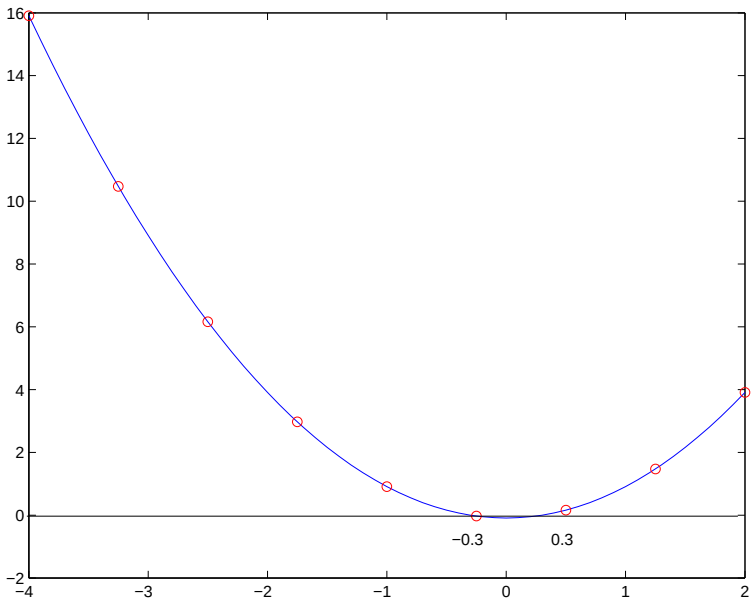

Figura 1.6: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico nell'intervallo $[-5, 5]$ è riportato in Fig. 1.4, in corrispondenza dei punti $t_i = -4 + (i-1)h,\ i = 1, \ldots, 9,\ h = 0.75$, appartenenti all'intervallo $[-4, 2]$.

t_i	$f(t_i)$
-.4000e+01	.1591e+02
-.3250e+01	.1047e+02
-.2500e+01	.6160e+01
-.1750e+01	.2972e+01
-.1000e+01	**.9100e+00**
-.2500e+00	**-0.2750e-01**
.5000e+00	**.1600e+00**
.1250e+01	.1472e+00
.2000e+01	.3910e+01

Tabella 1.5: Tabulazione di $f(x) = x^2 - 0.09$ in corrispondenza dei punti $t_i = -4 + (i-1)h,\ i = 1, \ldots, 9,\ h = 0.75$, in $[-4, 2]$.

Se si arresta il procedimento a questo passo, si può assumere, come approssimazione della radice dell'equazione (1.6), sia il punto medio dell'intervallo $[t_5, t_6]$, cioè -0.625, sia il punto medio dell'intervallo $[t_6, t_7]$, cioè 0.125. Se si desidera un'accuratezza maggiore, a partire da uno dei due intervalli, ad esempio l'intervallo più a sinistra $[t_5, t_6]$, si può iterare il procedimento. ♣

Il Metodo di Tabulazione non sempre è in grado di determinare al primo passo un intervallo che contiene zeri della funzione. In tal caso occorre diminuire opportunamente l'ampiezza dell'intervallo. Di seguito si riporta una sintesi dell'algoritmo che descrive il Metodo di Tabulazione e che tiene conto di questa possibilità.

Al passo k, indicato con $[a_k, b_k]$ l' intervallo corrente:

1. applicare il **processo di tabulazione**, ovvero fissare n punti distinti in $[a_k, b_k]$, in cui valutare f:
$$a_k = t_1 < ... < t_n = b_k\,;$$

2. scegliere, tra i sottointervalli $[t_i, t_{i+1}]$ (per $i = 1, \cdots, n-1$), l'intervallo dove si verifica una variazione di segno[4], cioè

   - se $s = \min\{i \in \{1, \cdots, n-1\}\ t.c.\ f(t_i) \cdot f(t_{i+1}) \leq 0\}$, allora
   $$[a_{k+1}, b_{k+1}] \equiv [t_s, t_{s+1}]$$

[4]Fra i vari sottointervalli in cui si verifica una variazione di segno, si può scegliere quello più a sinistra o stabilire un qualsiasi altro ordine.

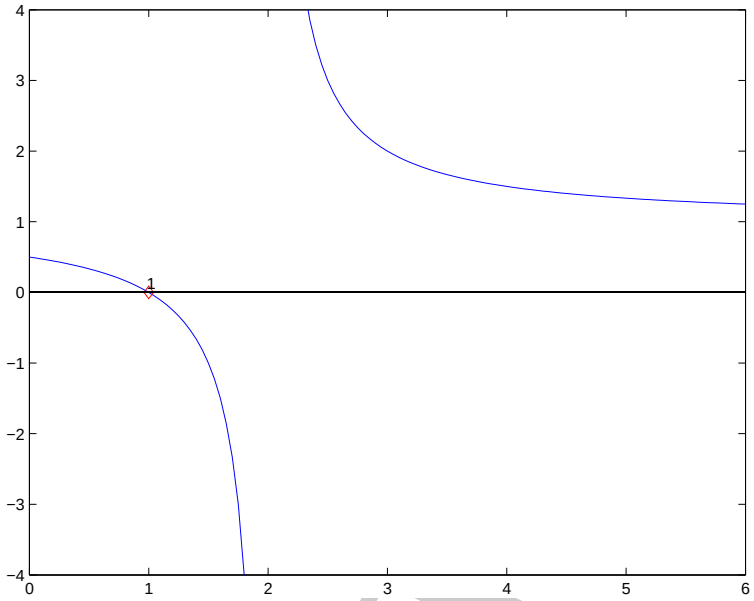

Figura 1.7: Grafico di $f(x) = \frac{x-1}{x-2}$ in $[0, 6]$. Con il simbolo '♢' si indica lo zero della funzione.

e si assume come approssimazione dello zero

$$x_k = (a_{k+1} + b_{k+1})/2 \tag{1.7}$$

- altrimenti, se la funzione non cambia segno in alcun intervallo, si torna al punto 1, ovvero si effettua una nuova tabulazione con $m > n$ punti.

1.2.1 Applicabilità

Il metodo in esame può essere sempre applicato a funzioni che siano valutabili su tutto l'intervallo di ricerca. Se tale condizione non è verificata, nulla si può dire sulla sua applicabilità.

♣ **Esempio 1.12.** Consideriamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = \frac{x-1}{x-2}$$

La funzione f ha uno zero in $x = 1$ e non è definita in $x = 2$. Applicando un passo del Metodo di Tabulazione in $[0, 6]$, con $n = 5$, si ottengono i risultati mostrati nella Tabella 1.6.
Da tali risultati si deduce che la funzione cambia segno in $[0, 1.5]$, ed una prima approssimazione della soluzione è data dal punto medio di tale intervallo, che è 0.75.
Tuttavia se vogliamo applicare il Metodo di Tabulazione alla stessa funzione f, cambiando da 5 a 7 il numero di punti di tabulazione, già alla prima iterazione il metodo diventa non applicabile. Infatti i punti della tabulazione sono $0, 1, 2, 3, 4, 5, 6$ e la funzione non è definita in uno di questi punti. ♣

t_i	$f(t_i)$
.0000e+00	**.5000e+00**
.1500e+00	**-.1000e+01**
.3000e+00	.2000e+01
.4500e+00	.1400e+01
.6000e+01	.1250e+01

Tabella 1.6: Tabulazione della funzione f, il cui grafico è riportato in Fig. 1.7, in corrispondenza dei punti $t_i = (i-1)h,\ i = 1, \ldots, 5,\ h = 0.15$, nell'intervallo $[0, 6]$.

1.2.2 Convergenza

Poichè il Metodo di Tabulazione è iterativo [5], ovvero a partire da valori iniziali, a e b, genera una successione di valori $\{x_k\}_{k\in\mathcal{N}}$, è importante studiare le condizioni che assicurano che tale successione si avvicini alla soluzione del problema x^*, ovvero si vuole studiare la **convergenza** del metodo[6].

♣ **Esempio 1.13.** Consideriamo la funzione:

$$f(x) = x^2 - \frac{1}{2};$$

essa ha due zeri nei punti $x_1 = -\sqrt{2}/2 \simeq -0.7071$ e $x_2 = \sqrt{2}/2 \simeq 0.7071$. Si vuole determinare uno zero di f in $[0, 1]$ utilizzando il Metodo di Tabulazione. Scegliendo 5 punti equidistanziati, cioè i punti $t_i = (i-1)\cdot h$, $i = 1, ..., 5$, con $h = 0.25$, si ottengono i valori riportati in Tabella 1.7.

[5] Da un punto di vista intuitivo, per **processo iterativo** si intende, in Matematica Numerica, una "regola" mediante la quale si calcola un valore, $x^{(k+1)}$, a partire da un valore $x^{(k)}$ già calcolato precedentemente con la stessa regola.

[6] Per un metodo iterativo, atto a risolvere la (1.1), sussiste la seguente definizione:

Definizione 1.2.1. *Considerata l'equazione*

$$f(x) = 0$$

e supposto che la funzione abbia almeno uno zero nell'intervallo $[a, b]$, sia $\{x_k\}_{k\in\mathcal{N}}$, con $\mathcal{N}$ insieme dei numeri naturali, la successione generata da un metodo iterativo. Si dice che il metodo converge se:

$$\lim_{k\to\infty} x_k = x^*, \tag{1.8}$$

dove x^* è uno zero della funzione nell'intervallo $[a, b]$.

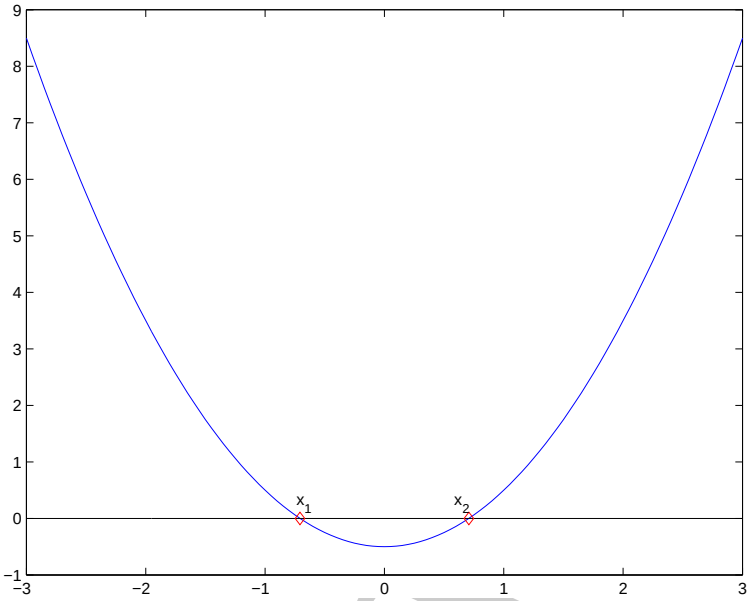

Figura 1.8: Grafico di $f(x) = x^2 - \frac{1}{2}$ in $[-3, 3]$. Con il simbolo '$\diamondsuit$' si indicano gli zeri della funzione: $x_1 = -\sqrt{2}/2$, $x_2 = \sqrt{2}/2$.

t_i	$f(t_i)$
.0000e+00	-.5000e+00
.2500e+00	-.4375e+00
.5000e+00	**-.2500e+00**
.7500e+00	**.6250e-01**
.1000e+01	.5000e+00

Tabella 1.7: Tabulazione della funzione f, il cui grafico è riportato in Fig. 1.8, in corrispondenza dei punti $t_i = (i-1)h,\ i = 1, \ldots, 5,\ h = 0.25$ nell'intervallo $[0, 1]$.

Si osserva che la funzione f assume segno discorde agli estremi dell'intervallo $[0.5, 0.75]$, il cui punto medio, 0.625, è la prima approssimazione dello zero di f. Procedendo con il Metodo di Tabulazione, si determinano via via nuovi sottointervalli di $[0, 1]$ dove la funzione cambia segno agli estremi. Indicato quindi con x_k il punto medio dell'intervallo corrente $[a_k, b_k]$ dopo 5 iterazioni si ottengono i risultati mostrati in Tabella 1.8.
Le approssimazioni determinate dal Metodo di Tabulazione sembrano avvicinarsi al valore $x_2 = \sqrt{2}/2 \simeq 0.7071$, ovvero il metodo sembra convergere. Una giustificazione di tale comportamento si può dare notando che se una funzione continua assume valori di segno opposto agli estremi di un intervallo, la funzione si annulla almeno in un punto interno a tale intervallo [7]. Da un punto di vista grafico tale

[7] Si ricorda, infatti, il seguente:

k	x_k	$f(x_k)$
1	.6250e+00	**-.1093e+00**
2	.7188e+00	**.1667e-01**
3	.7109e+00	**.5378e-02**
4	.7089e+00	**.2658e-02**
5	.7075e+01	**.5562e-03**

Tabella 1.8: Cinque iterazioni del Metodo di Tabulazione applicato alla funzione il cui grafico è riportato in Fig. 1.8. I valori di x_k indicano i punti medi degli intervalli $[a_k, b_k],\ k = 1, \ldots, 5$, sottointervalli di $[0, 1]$.

risultato può essere espresso osservando che se i due punti del grafico di $y = f(x)$, di ascissa a e b, si trovano rispettivamente al di sopra ed al di sotto (o viceversa) dell'asse delle x, allora il grafico della funzione interseca tale asse in almeno un punto.

♣

♣ **Esempio 1.14.** Consideriamo la funzione:

$$f(x) = \begin{cases} 3x^2 - 1, & x \leq 2.5 \\ 3x & x > 2.5 \end{cases}, \quad x \in [0, 4] \tag{1.9}$$

Essa ha uno zero nel punto $x^* = \sqrt{3}/3 \simeq 0.57735$ ed un punto di discontinuità in 2.5.
Si vuole determinare uno zero di f in $[0, 4]$ utilizzando il Metodo di Tabulazione. Scegliendo 5 punti equidistanziati, cioè i punti $t_i = i - 1$ con $i = 1, ..., 5$, si ottengono i valori riportati in Tabella 1.9.

t_i	$f(t_i)$
.0000e+00	**-.1000e+01**
.1000e+01	**.2000e+01**
.2000e+01	.1100e+02
.3000e+01	.9000e+01
.4000e+01	.1200e+02

Tabella 1.9: Tabulazione della funzione f, il cui grafico è riportato in Fig. 1.9, in corrispondenza dei punti $t_i = i - 1,\ i = 1, \ldots, 5$, nell'intervallo $[0, 4]$.

Teorema 1.2.1. [Teorema degli zeri]
Sia f una funzione continua in un intervallo $[a, b]$. Se $f(a) \cdot f(b) < 0$, allora esiste almeno un $x_0 \in (a, b)$ tale che $f(x_0) = 0$.

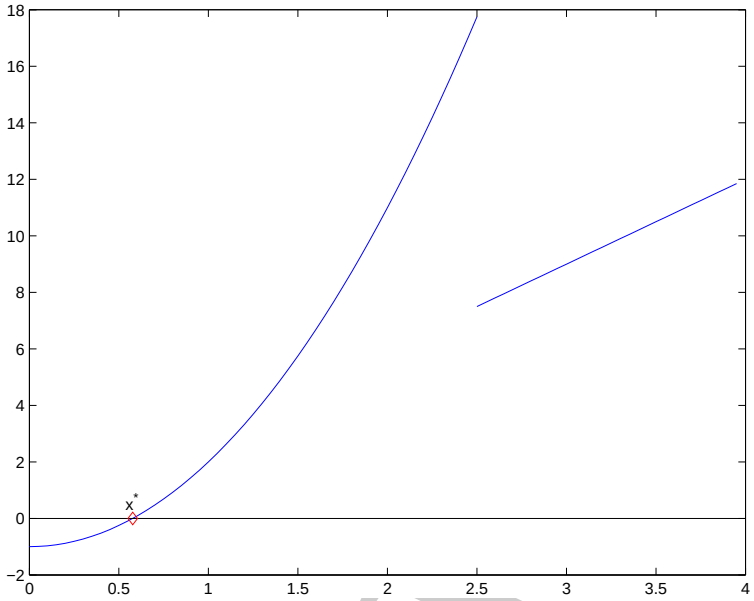

Figura 1.9: Grafico della funzione (1.9) in $[0, 4]$. Con il simbolo '♢' si indica lo zero della funzione: $x^* = \sqrt{3}/3$.

Si osserva che la funzione f assume segno discorde agli estremi dell'intervallo $[0, 1]$, il cui punto medio $x_1 = 0.5$ è la prima approssimazione dello zero di f. Dopo 5 iterazioni del Metodo di Tabulazione si ottengono i risultati riportati in Tabella 1.10.

k	x_k	$f(x_k)$
1	.5000e+00	**-.2500e+00**
2	.6250e+00	**.1718e+00**
3	.5937e+00	**.5761e-01**
4	.5703e+00	**-.2423e-01**
5	.5761e+00	**-.4078e-02**

Tabella 1.10: Cinque iterazioni del Metodo di Tabulazione applicato alla funzione il cui grafico è riportato in Fig. 1.9. I valori di x_k indicano i punti medi degli intervalli $[a_k, b_k]$, $k = 1, \ldots, 5$, sottointervalli di $[0, 4]$.

♣

Al fine di determinare condizioni sufficienti per la convergenza introduciamo l'**errore di troncamento analitico** alla k-ma iterazione, definito come:

$$e_k = x^* - x_k, \tag{1.10}$$

dove x^* è uno zero della funzione nell'intervallo $[a, b]$. Si osservi che la convergenza del metodo ad uno zero della funzione è equivalente alla convergenza a zero della successione dei valori assoluti degli errori, cioè:

$$\lim_{k\to\infty} x_k = x^* \qquad \Leftrightarrow \qquad \lim_{k\to\infty} |e_k| = 0 \tag{1.11}$$

Sussiste il seguente:

Teorema 1.2.2. *Sia f una funzione continua nell'intervallo chiuso e limitato $[a, b]$. Condizione sufficiente affinché il Metodo di Tabulazione con n punti equidistanziati* [8]*, converga a x^*, dove x^* è uno zero della funzione nell'intervallo $[a, b]$, è che*

$$f(a) \cdot f(b) < 0. \tag{1.12}$$

Dimostrazione Il Metodo di Tabulazione genera, a partire dall'intervallo $[a_0, b_0] = [a, b]$, ad ogni passo, $(n-1)$ sottointervalli. Essendo gli n punti della tabulazione equidistanziati, ognuno di questi intervalli ha ampiezza uguale a $1/(n-1)$ dell'ampiezza dell'intervallo precedente; si ha quindi:

$$b_k - a_k = \frac{1}{n-1}(b_{k-1} - a_{k-1}), \ \ \forall k \geq 1. \tag{1.13}$$

Applicando l'uguaglianza (1.13) ripetutamente, si ottiene che:

$$b_k - a_k = \frac{1}{(n-1)^k}(b_0 - a_0), \ \ \forall k \geq 1. \tag{1.14}$$

Detto $[a_k, b_k]$ l'intervallo contenente uno zero x^* della funzione, che esiste per l'ipotesi (1.12) e per il Teorema 1.2.1, avendo posto nella (1.7):

$$x_k = (a_{k+1} + b_{k+1})/2\,,$$

per la (1.10) si ha:

$$|e_k| \leq \frac{1}{2}(b_k - a_k) \tag{1.15}$$

da cui, in base alla (1.14) si ha:

$$|e_k| \leq \frac{1}{2(n-1)^k}(b_0 - a_0), \ \ \forall k \geq 0. \tag{1.16}$$

Quindi, la successione dei valori assoluti degli errori converge a zero perchè è limitata superiormente da una successione che converge a zero. Di conseguenza, la successione $\{x_k\}$ converge a x^*. ■

Dalla (1.15) discende la stima dell'errore di troncamento analitico, fornita dalla metà dell'ampiezza dell'intervallo corrente, che rappresenta il massimo errore, in valore assoluto, che si può avere in quella iterazione. Inoltre, dalla (1.16), si può dedurre la relazione che esiste tra il massimo errore alla $k+1$-ma iterazione e quello massimo commesso all'iterazione precedente:

$$|e_{k+1}| = \frac{1}{n-1}|e_k| \tag{1.17}$$

[8] Un risultato analogo continua a valere se i punti della tabulazione sono arbitrari.

La relazione (1.17) mostra che, ad ogni iterazione del Metodo di Tabulazione, l'errore diminuisce rispetto a quello all'iterazione precedente del fattore $n-1$. In tal caso, si dice che il metodo **converge in modo lineare**. In generale sussiste la seguente:

Definizione 1.2.2. *Si dice che una successione $\{x_k\}_{k\in\mathcal{N}}$ convergente a x^* ha* ordine di convergenza *p ($p \geq 1$) se esiste $\overline{n} \in \mathcal{N}$ ed un numero reale positivo C ($C < 1$ se $p = 1$) tale che, per ogni $n > \overline{n}$ si abbia*

$$|x_{n+1} - x^*| \leq C|x_n - x^*|^p$$

In particolare quando $p = 1$ si parla di **convergenza lineare**, per $p = 2$ di **convergenza quadratica** ed, in generale, per $p > 1$ di **convergenza superlineare**; la costante C prende il nome di **raggio di convergenza**.

Dalla (1.15) segue, inoltre, che un criterio di arresto naturale del processo di tabulazione si basa sull'ampiezza del generico intervallo $[a_k, b_k]$. Infatti, se $(b_k - a_k) < tol$ allora

$$|e_k| \leq \frac{1}{2}\, tol$$

Il Teorema 1.2.2 fornisce una condizione solo sufficiente per la convergenza. Se la funzione f assume valori di segno concorde agli estremi dell'intervallo, significa che potrebbe avere più di uno zero o non averne, in quell'intervallo. In tal caso sarà necessario effettuare alcuni passi di tabulazione dopodiché, nel caso in cui la funzione possegga più di uno zero, l'algoritmo individua l'intervallo che contiene uno zero e si arresta dopo aver raggiunto la tolleranza richiesta sull'ampiezza dell'intervallo $[a_k, b_k]$. Se la funzione non possiede alcuno zero, l'algoritmo comunque si arresta per aver raggiunto la tolleranza richiesta sull'ampiezza del generico intervallo di tabulazione.

Metodi iterativi per la risoluzione di equazioni non lineari, che godono della suddetta proprietà di convergenza, sono detti **globali** i quali si distinguono dai metodi **locali**, ovvero da quei metodi la cui convergenza ad uno zero della funzione dipende non solo dall'intervallo di ricerca, ma anche dalla scelta opportuna dei punti iniziali del procedimento iterativo.

1.3 Il Metodo di Bisezione

♣ **Esempio 1.15.** Consideriamo la funzione dell'esempio 1.13:

$$f(x) = x^2 - \frac{1}{2}\,;$$

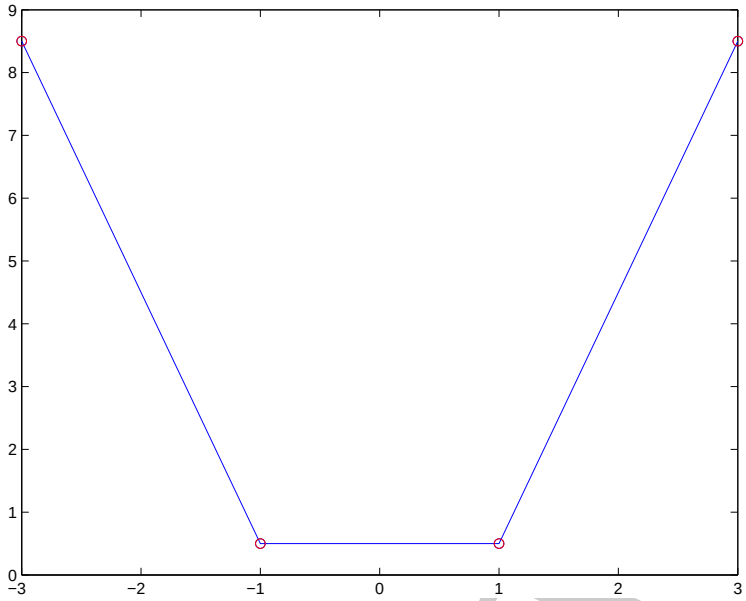

Figura 1.10: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico è riportato in Fig. 1.8, in corrispondenza dei punti $t_i = -3+(i-1)h,\ i = 1, \ldots, 4,\ h = 2$, nell'intervallo $[-3, 3]$.

essa ha due zeri nei punti $x_1 = -\sqrt{2}/2 \simeq -0.7071$ e $x_2 = \sqrt{2}/2 \simeq 0.7071$. Se tabuliamo la funzione in [-3,3] con 4 punti di tabulazione non troviamo alcun intervallo in cui la funzione cambia segno, come mostrato in Figura 1.10. Quindi, volendo applicare un altro passo, infittiamo la tabulazione, scegliendo $n = 7$ punti di tabulazione della funzione in [-3,3]. Dalla Figura 1.11 si può evincere la presenza di almeno due zeri della funzione, e precisamente il primo nell'intervallo $[-1, 0]$ ed il secondo nell'intervallo $[0, 1]$.

♣

Spesso l'individuazione di un intervallo che potrebbe contenere uno zero dipende dal numero di punti di tabulazione, nel senso che una tabulazione con pochi punti può non rivelare la presenza di uno o più zeri. D'altra parte per individuare intervalli contenenti delle soluzioni potrebbe essere necessario scegliere troppi punti di tabulazione, e quindi possono essere necessarie numerose valutazioni della funzione, la maggior parte delle quali, spesso, risultano superflue. Un metodo più efficiente per la risoluzione di equazioni non lineari è **il Metodo di Bisezione**.

♣ **Esempio 1.16.** Consideriamo l'equazione dell'esempio 1.10:

$$3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0; \tag{1.18}$$

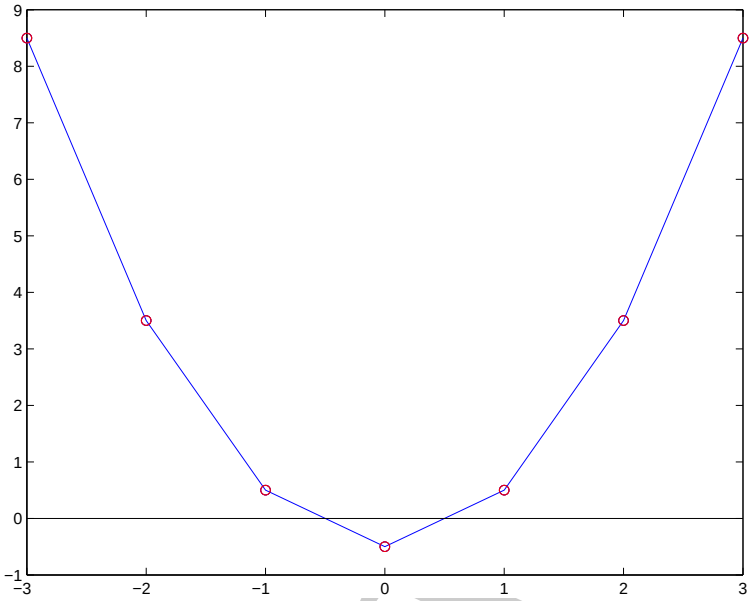

Figura 1.11: Con il simbolo 'o' si indica il valore assunto dalla funzione f, il cui grafico è riportato in Fig. 1.8, in corrispondenza dei punti $t_i = -3+(i-1)h,\ i = 1, \ldots, 7,\ h = 1$, nell'intervallo $[-3, 3]$.

si consideri l'intervallo $[0, 2]$, in cui la funzione ha un solo zero, $x = 2/3$ e valori estremi di segno opposto. Si consideri il punto medio dell'intervallo iniziale, cioè $x_1 = 1$. Si generano in tal modo due sottointervalli, $[0, 1]$ e $[1, 2]$, entrambi di ampiezza uguale alla metà di quello iniziale, e x_1 costituisce un'approssimazione dello zero della funzione. A questo punto, osservando che la funzione assume valori di segno opposto agli estremi dell'intervallo $[0, 1]$, si può applicare il procedimento appena descritto a tale intervallo, ottenendo una nuova approssimazione della soluzione, $x_2 = 0.5$, ed altri due sottointervalli, $[0, 0.5]$ e $[0.5, 1]$. Iterando il procedimento è possibile determinare una successione di intervalli, ciascuno di ampiezza uguali alla metà del precedente e contenente una soluzione di (1.18). Il procedimento descritto è noto come **Metodo di Bisezione**.
In Tabella 1.11 sono riportati i valori x_k, e i corrispondenti valori della funzione, per $k = 1, \ldots, 19$, ottenuti applicando il Metodo di Bisezione alla risoluzione di (1.18). Si osservi che, con 5 iterazioni ($k = 5$), si ottiene una approssimazione della soluzione migliore di quella che si ottiene tabulando la funzione in 11 punti equidistanziati nell'intervallo (vedi Tabella 1.1). Ciò conferma la maggiore efficienza del Metodo di Bisezione rispetto al Metodo di Tabulazione. In Figura 1.12 è rappresentata graficamente l'applicazione del metodo alla risoluzione di (1.18).

♣

k	x_k	$f(x_k)$
1	1.0000e+00	1.8768e+01
2	5.0000e-01	-1.0938e+01
3	7.5000e-01	4.7461e+00
4	6.2500e-01	-2.5559e+00
5	6.8750e-01	1.2325e+00
6	6.5625e-01	-6.2764e-01
7	6.7188e-01	3.1097e-01
8	6.6406e-01	-1.5620e-01
9	6.6797e-01	7.7921e-02
10	6.6602e-01	-3.9005e-02
11	6.6699e-01	1.9491e-02
12	6.6650e-01	-9.7485e-03
13	6.6675e-01	4.8735e-03
14	6.6663e-01	-2.4369e-03
15	6.6669e-01	1.2184e-03
16	6.6666e-01	-6.0922e-04
17	6.6667e-01	3.0461e-04
18	6.6666e-01	-1.5231e-04
19	6.6667e-01	7.6153e-05

Tabella 1.11: Valori numerici di x_k e $f(x_k)$ relativi al Metodo di Bisezione applicato all'equazione $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$ in $[0, 2]$ (19 iterazioni). x_k indica il punto medio del sottointervallo $[a_k, b_k]$, $k = 1, \ldots, 19$, di $[0, 2]$, determinato al passo k.

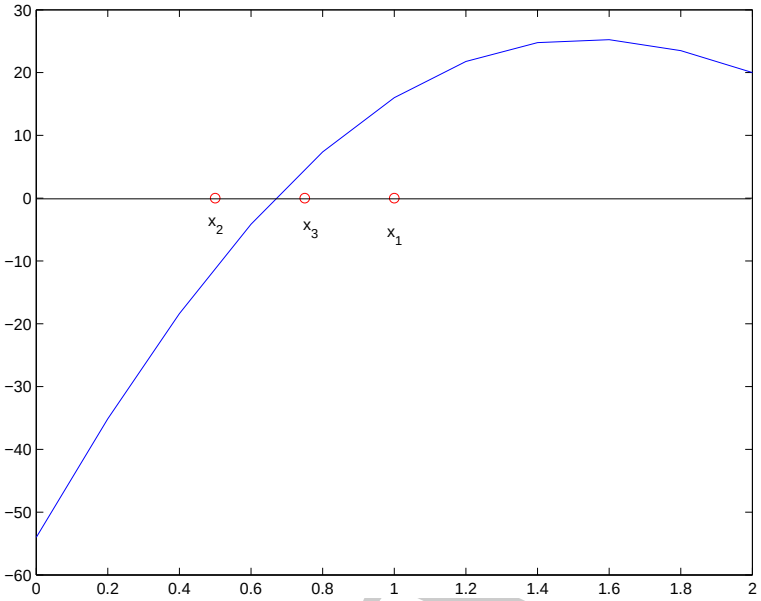

Figura 1.12: Illustrazione grafica del Metodo di Bisezione applicato all'equazione $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$ in $[0, 2]$ (3 iterazioni): $x_1 = 1$, $x_2 = 0.5$ e $x_3 = 0.75$.

♣ **Esempio 1.17.** Consideriamo la funzione:

$$f(x) = \begin{cases} x^3 - 1/3, & x \leq 2/3 \\ x/3 & x > 2/3 \end{cases} \tag{1.19}$$

il cui grafico è riportato in Figura 1.13.
Applicando il Metodo di Bisezione nell'intervallo $[0, 1[$ si ottengono i valori riportati in Tabella 1.12. ♣

Il Metodo di Bisezione è un particolare metodo iterativo che, a partire dall'intervallo di ricerca $[a, b]$, genera una successione di valori $\{x_k\}_{k=1,2,\ldots}$, come descritto di seguito: sia $[a_0, b_0] \equiv [a, b]$ e sia $[a_k, b_k]$ $(k > 0)$ il generico intervallo:

- calcolo del punto medio di $[a_k, b_k]$:

$$x_{k+1} = a_k + \frac{b_k - a_k}{2}$$

- scelta dell'intervallo successivo:
  - se $f(x_{k+1})f(a_k) \leq 0$, allora $[a_{k+1}, b_{k+1}] \equiv [a_k, x_{k+1}]$;
  - altrimenti $[a_{k+1}, b_{k+1}] \equiv [x_{k+1}, b_k]$.

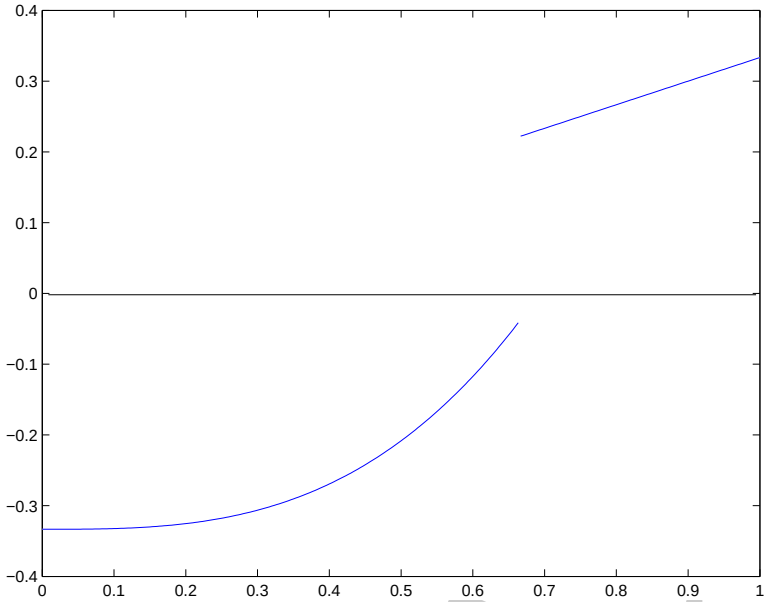

Figura 1.13: Grafico della funzione (1.19) in $[0, 1]$.

k	x_k	$f(x_k)$
1	1.0000e+00	1.8768e+01
2	5.0000e-01	-0.20833e+00
3	7.5000e-01	0.25000e+00
4	6.2500e-01	-0.89192e-01
5	6.8750e-01	0.22916e+00
6	6.5625e-01	-0.50710e-01
7	6.7188e-01	0.22396e+00
8	6.6406e-01	-0.40499e-01
9	6.6797e-01	0.22656e+00

Tabella 1.12: Valori numerici di x_k e $f(x_k)$ relativi al Metodo di Bisezione applicato alla (1.19) in $[0, 2]$ (9 iterazioni). x_k indica il punto medio del sottointervallo $[a_k, b_k]$, $k = 1, \ldots, 9$, di $[0, 2]$, determinato al passo k.

Di seguito è riportata una prima versione dell'algoritmo, scritto utilizzando il linguaggio Pascal-like, basato sul Metodo di Bisezione. Si osservi che, in tale versione, è fissato il numero di iterazioni da effettuare (rappresentato dalla variabile `itmax`).

```
procedure fxbis(input: a,b,f,itmax, out: c,fc)
/# SCOPO: calcolo di un'approssimazione dello zero
/#          di una funzione mediante il Metodo di Bisezione
/# SPECIFICHE PARAMETRI:
 var: a      : reale            {primo estremo dell'intervallo di ricerca}
 var: b      : reale            {secondo estremo dell'intervallo di ricerca}
 var: f      : funzione esterna {funzione di cui si cerca lo zero}
 var: itmax  : intero           {numero massimo di iterazioni}
 var: c      : reale            {approssimazione dello zero}
 var: fc     : reale            {valore di f in c}
/# INIZIO ISTRUZIONI:
    fa := f(a);
    fb := f(b);
    for i = 1, itmax
      c := a + (b − a)/2;        {calcolo del punto medio e }
      fc := f(c);                {del valore della funzione }
       if (fc ∗ fa ≤ 0) then          {test sul segno di f}
        b := c;
        fb := fc;
       else
        a := c;
        fa := fc;
       endif
    endfor
end fxbis
```

Procedura 1.1: Algoritmo del Metodo di Bisezione (prima versione)

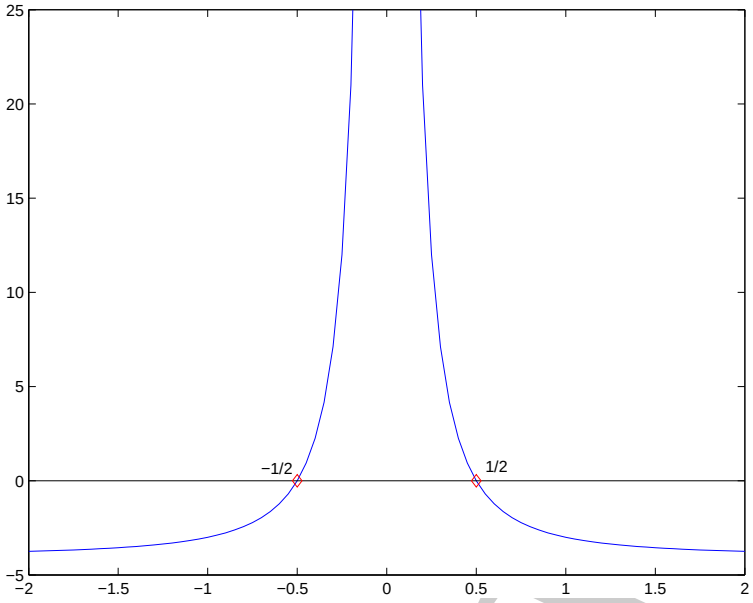

Figura 1.14: Grafico di $f(x) = \frac{1}{x^2} - 4$ in $[-2, 2]$. Con il simbolo '♢' si indicano gli zeri della funzione.

1.3.1 Applicabilità

Il Metodo di Bisezione è un procedimento di "tabulazione nel punto medio", più precisamente di tabulazione con $n = 3$ punti equidistanziati, ripetuto più volte. Di conseguenza il Metodo di Bisezione eredita le caratteristiche del Metodo di Tabulazione. Si deduce, così, che può essere sempre applicato a funzioni che sono valutabili su tutto l'intervallo di ricerca, mentre se tale condizione non è verificata, nulla si può dire sulla sua applicabilità.
L'esempio che segue mostra che, nel caso di funzioni non ovunque definite, l'applicabilità del Metodo di Bisezione dipende dall'intervallo di ricerca, ovvero dalla scelta dei punti di tabulazione.

♣ **Esempio 1.18.** Consideriamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = \frac{1}{x^2} - 4$$

La funzione f ha due zeri in $x_1 = -0.5$ e $x_2 = 0.5$. Applichiamo il Metodo di Bisezione in $[-2, 1.5]$. Indicato con c il punto medio dell'intervallo corrente $[a_k, b_k]$, dopo 5 iterazioni si ottengono i risultati mostrati nella Tabella 1.13.
Il metodo, in questo caso, sembra avvicinarsi ad una delle radici dell'equazione (in particolare alla $x_1 = -0.5$). Tuttavia, se vogliamo applicare il Metodo di Bisezione alla stessa funzione f, cambiando però l'intervallo in $[-2, 2]$, già alla prima iterazione il metodo diventa non applicabile. Infatti il valore del punto medio dell'intervallo al primo passo è $x = 0$, punto in cui la funzione non è definita.

k	x_k	$f(x_k)$
1	-0.2500	12.000
2	-1.1250	-3.2099
3	-0.6875	-3.4711
4	-0.4688	0.5511
5	**-0.5781**	**-1.0080**

Tabella 1.13: Cinque iterazioni del Metodo di Bisezione applicato alla funzione il cui grafico è riportato in Fig. 1.14. x_k indica il punto medio del sottointervallo $[a_k, b_k]$, $k = 1, \ldots, 5$, di $[-2, 1.5]$, determinato al passo k.

♣

1.3.2 Convergenza

Come per l'applicabilità, anche la convergenza può essere dedotta in analogia al Metodo di Tabulazione. Dal Teorema 1.2.2 segue, quindi:

Corollario 1.3.1. *Sia f una funzione continua nell'intervallo chiuso e limitato $[a, b]$. Condizione sufficiente affinché il Metodo di Bisezione converga a x^*, dove x^* è uno zero della funzione nell'intervallo $[a, b]$, è che $f(a) \cdot f(b) < 0$.*

Si osservi, infine, che analogamente al metodo di Tabulazione, anche per il metodo di Bisezione le ipotesi di continuità della funzione non sono necessarie per la convergenza della successione generata dall'algoritmo di bisezione [9]. Il raggio di convergenza del metodo di bisezione è $1/2$, come si evince dalla relazione (1.17). Infatti, ponendo $n = 3$ si ottiene

$$|e_{k+1}| = \frac{1}{2}|e_k| \tag{1.20}$$

♣ **Esempio 1.19.** La funzione dell'esempio 1.17 non ha zeri, tuttavia la successione determinata dal metodo di Bisezione nell'intervallo $[0, 1[$ sembra comunque convergere al valore $2/3$, che è un punto di discontinuità della funzione. ♣

Come per il Metodo di Tabulazione, anche per il Metodo di Bisezione un criterio di arresto naturale si basa sull'ampiezza dell'intervallo corrente. Come accade per il metodo

[9] Il metodo può convergere ma non avere come limite la soluzione.

di Tabulazione, se $f(a)f(b) > 0$, la funzione f potrebbe avere più di uno zero in $[a, b]$ oppure potrebbe non averne. In tal caso prima di applicare il Metodo di Bisezione occorre determinare l'intervallo contenente lo zero. A tal fine si può applicare un processo di tabulazione.

♣ **Esempio 1.20.** Si consideri una funzione f definita nell'intervallo $[a, b]$ e tale che $f(a)f(b) > 0$. Dividiamo l'intervallo $[a, b]$ in due sottointervalli $[a, c]$ e $[c, b]$. In ciascuno dei due sottointervalli verifichiamo se la funzione f possiede uno zero. Se $f(a)f(c) < 0$ o $f(c)f(b) < 0$, cioè è stato localizzato un sottointervallo che contiene uno zero, l'algoritmo termina. Altrimenti, si ripete la suddivisione di ciascun sottointervallo finché non si trova un intervallo in cui la funzione ha una variazione di segno agli estremi. In questo modo si effettua una tabulazione della funzione nell'intervallo $[a, b]$, utilizzando uno schema ad albero binario, la cui profondità dipende dall'ampiezza dell'intervallo e dalla tolleranza richiesta. L'algoritmo, infatti, termina quando si localizza un sottointervallo che contiene uno zero, oppure quando l'ampiezza del sottointervallo considerato è minore di una tolleranza richiesta. ♣

1.4 Il Metodo di Newton

Nel Metodo di Bisezione la successione di valori $\{x_k\}$ è generata tenendo conto solo delle variazioni di segno della funzione negli estremi degli intervalli $[a_k, b_k]$. Non è utilizzata alcuna informazione sull'andamento della funzione; invece potrebbe essere utile avere informazioni sulla "pendenza" della funzione nell'intervallo di ricerca, così come illustrato nell'esempio che segue.

♣ **Esempio 1.21.** Consideriamo l'equazione non lineare:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = 0.2x - log(x)$$

Se consideriamo la funzione f in [1, 7] essa ha uno zero in $x^* = 1.2958\ldots$, ovvero ha uno zero vicino all'estremo sinistro dell'intervallo. In questo caso il Metodo di Bisezione è abbastanza "lento". Osserviamo, infatti, la Figura 1.16, dove sono rappresentati i 3 passi del Metodo di Bisezione; x_3 è ancora "abbastanza" lontano dalla soluzione x^*, come si può osservare anche dai valori numerici riportati nella Tabella 1.14.

Se indichiamo con t la retta tangente a f nel punto $(1, f(1))$, detta z l'ascissa della sua intersezione con l'asse delle ascisse, tale punto risulta molto più vicino ad x^*, come si evince dalla Tabella 1.14.

♣

Un metodo che tiene conto dell'andamento della funzione è il **Metodo di Newton**.

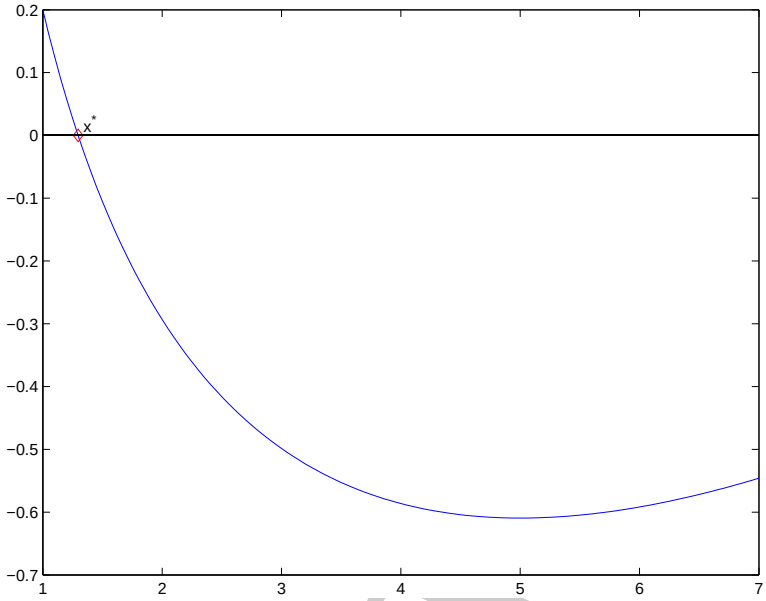

Figura 1.15: Grafico di $f(x) = 0.2x - log(x)$ in $[1, 7]$. Con il simbolo '$\diamondsuit$' si indica lo zero della funzione.

$x_1 = 4$	$f(x_1) = -0.5863...$
$x_2 = 2.5$	$f(x_2) = -0.4163...$
$x_3 = 1.75$	$f(x_3) = -0.2096...$
$z = 1.25$	$f(z) = 0.0269...$

Tabella 1.14: Tre iterazioni del Metodo di Bisezione applicato alla funzione in Fig. 1.16. x_1, x_2, x_3 sono le prime tre approssimazioni generate dal Metodo di Bisezione, z è l'ascissa del punto di intersezione con l'asse delle ascisse della tangente a $f(x) = 0.2x - log(x)$ nel punto $(1, f(1))$.

♣ **Esempio 1.22.** Considerata l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54, \tag{1.21}$$

si scelga il punto $x_0 = 1$ e si consideri l'equazione della retta tangente alla curva $y = f(x)$ in questo punto:

$$m_0(x) = f(x_0) + f'(x_0)(x - x_0) = 16 + 36(x - 1).$$

Il punto di intersezione di tale retta con l'asse delle ascisse (che si ottiene risolvendo $m_0(x) = 0$) è:

$$x_1 = 20/36 \simeq 0.55556,$$

Si può procedere in modo analogo a partire da x_1, ottenendo:

$$m_1(x) = f(x_1) + f'(x_1)(x - x_1) \simeq -7.0815 + 67.539(x - 0.55556);$$

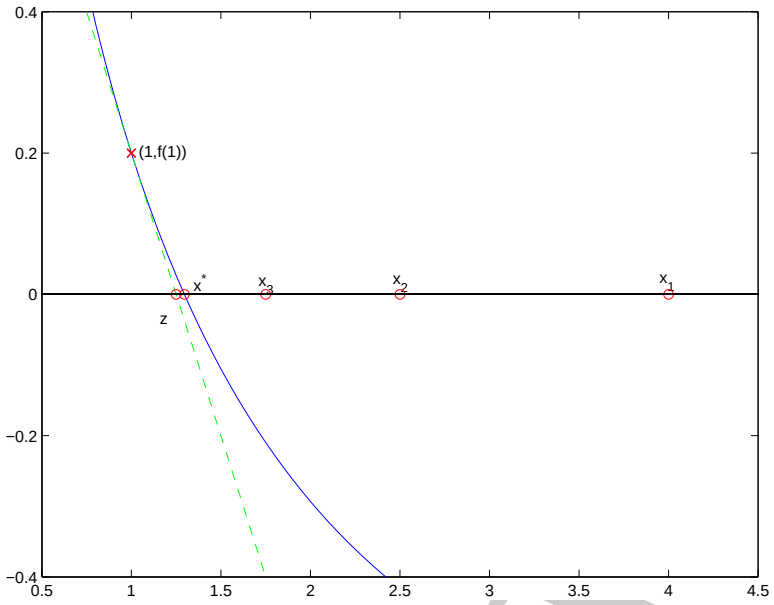

Figura 1.16: Illustrazione grafica del Metodo di Bisezione applicato all'equazione $0.2x - log(x) = 0$ in $[1, 7]$ (3 iterazioni): $x_1 = 4$, $x_2 = 2.5$ e $x_3 = 1.75$ sono le prime tre approssimazioni generate dal Metodo di Bisezione, z è l'ascissa del punto di intersezione con l'asse delle ascisse della tangente a $f(x) = 0.2x - log(x)$ nel punto $(1, f(1))$, x^* è uno zero di f in $[1, 7]$.

da cui, risolvendo $m_1(x) = 0$, si ottiene:

$$x_2 \simeq 0.66041.$$

In Tabella 1.15 sono riportati i valori ottenuti nelle prime 4 iterazioni del Metodo di Newton applicato all'equazione (1.21), con $x_0 = 1$.

k	x_k	$f(x_k)$
1	5.5556e-01	-7.0818e+00
2	6.6041e-01	-3.7601e-01
3	6.6664e-01	-1.3582e-03
4	6.6667e-01	-1.8001e-08

Tabella 1.15: Quattro iterazioni del Metodo di Newton applicato all'equazione $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$ con $x_0 = 1$. x_k indica l'approssimazione generata dal Metodo di Newton al passo k, $k = 1, \ldots, 4$.

Confrontando tali risultati con quelli ottenuti con il Metodo di Bisezione applicato in $[0, 2]$, riportati nella Tabella 1.11, si osserva che, per calcolare un'approssimazione $\tilde{x} = 0.66667$ con quattro cifre significative della soluzione $x = 0.666666\ldots$:

- il *Metodo di Bisezione* impiega 19 iterazioni calcolando:

$$\tilde{x} = x_{19} = 0.66667, \quad |f(x_{19})| = 7.6153 \times 10^{-5}$$

- il *Metodo di Newton* impiega 4 iterazioni calcolando:

$$\tilde{x} = x_4 = 0.66667, \quad |f(x_4)| = 1.8001 \times 10^{-8}$$

Per l'equazione considerata il Metodo di Newton fornisce valori vicini alla soluzione più velocemente rispetto al Metodo di Bisezione. ♣

♣ **Esempio 1.23.** Consideriamo l'equazione:

$$\tan x = 0$$

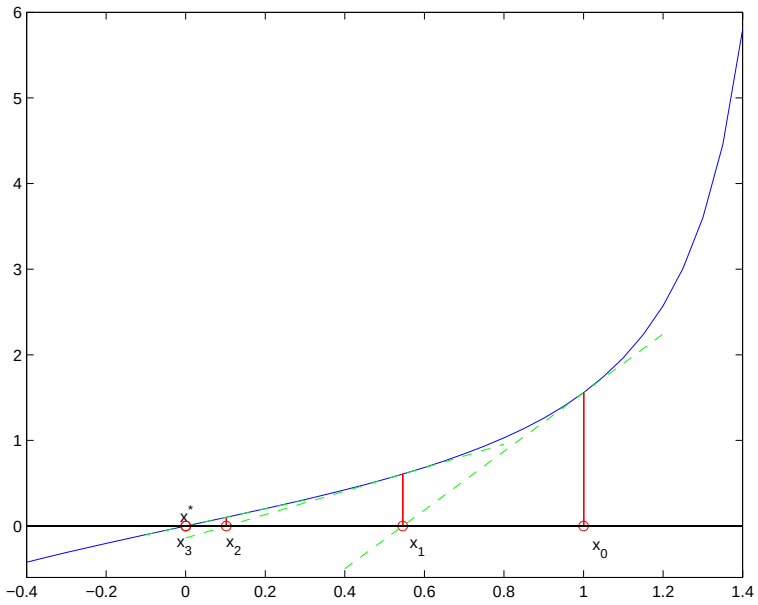

Figura 1.17: Illustrazione grafica del Metodo di Newton applicato all'equazione $\tan x = 0$ con $x_0 = 1$ (3 iterazioni): x_1 indica l'intersezione con l'asse delle ascisse della tangente a $f(x) = \tan(x)$ nel punto $(x_0, f(x_0))$; x_2 indica l'intersezione con l'asse delle ascisse della tangente a f nel punto $(x_1, f(x_1))$; x_3 indica l'intersezione con l'asse delle ascisse della tangente a f nel punto $(x_2, f(x_2))$; x^* è uno zero di f in $[-0.4, 1.4]$.

In Fig. 1.17 è illustrata graficamente l'applicazione di 3 iterazioni del Metodo di Newton alla risoluzione di tale equazione, con approssimazione iniziale della soluzione $x_0 = 1$. ♣

Il Metodo di Newton costruisce una successione di valori $\{x_k\}_{k=1,2,\ldots}$ nel modo seguente:

- $\forall k = 0, 1, 2, \ldots$, a partire da un valore iniziale x_0, si considera la tangente alla curva $y = f(x)$ nel punto $(x_k, f(x_k))$, e si assume come approssimazione dello zero della funzione il valore, x_{k+1}, ottenuto intersecando tale retta con l'asse delle ascisse, cioè:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{1.22}$$

Di seguito è riportata una prima versione dell'algoritmo di Newton per risolvere l'equazione $f(x) = 0$, $x \in [a, b] \subseteq A$, dove $f : A \subseteq \Re \to \Re$ è una funzione con derivata f' definita in $[a, b]$.

procedure fxnew(**input:** $x0$,f,f',$itmax$, **out:** $xnew$,$fxnew$)

/# SCOPO: calcolo di un'approssimazione dello zero
/# di una funzione mediante il Metodo di Newton
/# SPECIFICHE PARAMETRI:

var: $x0$	: **reale**	{*approssimazione iniziale dello zero*}
var: f	: **funzione esterna**	{*funzione di cui si cerca lo zero*}
var: f'	: **funzione esterna**	{*derivata di f*}
var: $itmax$	: **intero**	{*numero massimo di iterazioni*}
var: $xnew$	: **reale**	{*approssimazione dello zero*}
var: $fxnew$	: **reale**	{*valore di f in xnew*}

/# INIZIO ISTRUZIONI:

$k := 0$ {*generazione della successione*}
repeat {*delle approssimazioni*}
$x_{k+1} := x_k - f(x_k)/f'(x_k)$ {*calcolo*}
$k := k + 1$ {*dell'approssimazione*}
until (condizione di arresto verificata)
end fxnew

Procedura 1.2: Algoritmo del Metodo di Newton (prima versione)

1.4.1 Applicabilità

Abbiamo visto che, per applicare il Metodo di Bisezione, è sufficiente che la funzione f sia valutabile nell'intervallo di ricerca. Per il Metodo di Newton è necessaria anche la conoscenza della derivata prima della funzione. Inoltre, nel Metodo di Newton l'approssimazione x_{k+1} può essere calcolata solo se è $f'(x_k) \neq 0$.

♣ **Esempio 1.24.** Se si considera l'equazione:

$$sin(x) = 0$$

in $]0, 2\pi[$ e si vuole calcolare un'approssimazione dello zero π, con il Metodo di Newton, in tale intervallo, scegliendo come approssimazione iniziale $x_0 = \pi/2$, la retta tangente t non ha intersezione con l'asse delle x, essendo ad esso parallela. Infatti, in questo caso, $f'(x_0) = cos(\pi/2) = 0$
Purtoppo però, anche se come punto iniziale si sceglie un punto vicino a $\pi/2$, ad esempio, si pone $x_0 = 1.57$ per cui $f'(x_0) \neq 0$, la retta tangente s non ha intersezione con l'asse delle x *nell'intervallo di ricerca dello zero*, $]0, 2\pi[$.
È quindi intuitivo il fatto che il Metodo di Newton non sia applicabile non solo se il valore assoluto di $f'(x_k)$ è uguale a zero, ma anche se esso è molto piccolo (in questi casi si usa dire che x_k è uno zero numerico di f').

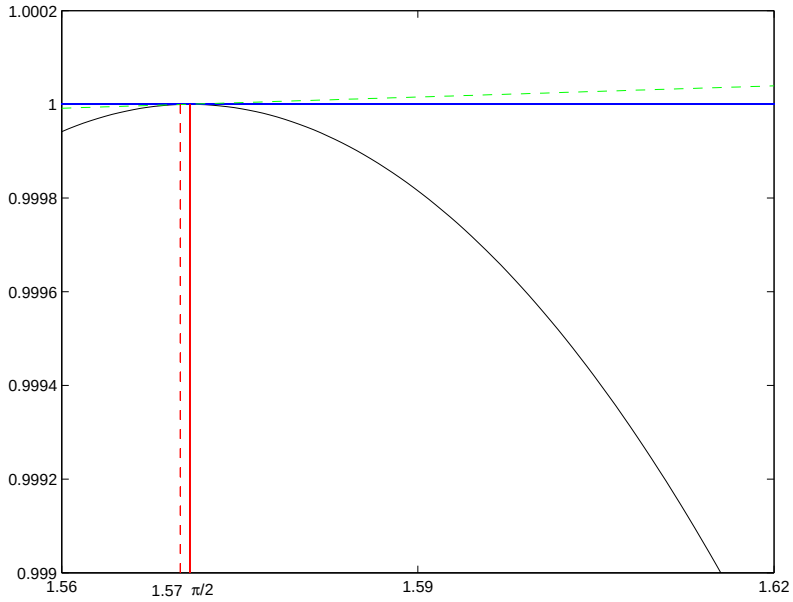

Figura 1.18: Illustrazione grafica del Metodo di Newton applicato all'equazione $sin(x) = 0$ con $x_0 = \pi/2$. Con il tratto '-' si descrive la tangente, t, alla curva in x_0; t non ha intersezione con l'asse delle x, essendo ad esso parallela. Con il tratto '$--$' si descrive la tangente, s, alla curva in $= 1.57$, "vicino" a $\pi/2$; s non è parallela all'asse delle x ma anch'essa non ha intersezione con l'asse delle x in $]0, 2\pi[$.

♣

1.4.2 Convergenza

Si vogliono analizzare le caratteristiche di convergenza del Metodo di Newton, ovvero si vuole studiare sotto quali condizioni il Metodo di Newton converge ad una radice di f.

♣ **Esempio 1.25.** Applichiamo il Metodo di Newton all'equazione:

$$\arctan(x) = 0$$

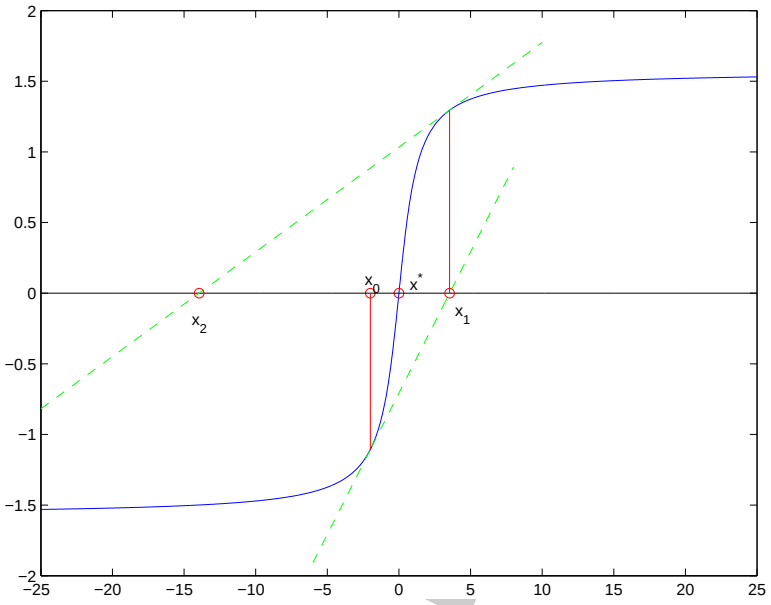

Figura 1.19: Illustrazione grafica del Metodo di Newton applicato all'equazione $\arctan(x) = 0$ con $x_0 = -2$ (2 iterazioni): x_1 indica l'intersezione con l'asse delle ascisse della tangente a $f(x) = \arctan(x)$ nel punto $(x_0, f(x_0))$; x_2 indica l'intersezione con l'asse delle ascisse della tangente a f nel punto $(x_1, f(x_1))$; x^* è uno zero di f in $[-25, 25]$.

In Fig. 1.19 sono illustrate le prime 2 iterazioni del Metodo di Newton, applicato nell'intervallo $[-25, 25]$. Dalla Tabella 1.16 si evince che, partendo da $x_0 = -2$, la successione di approssimazioni generata si allontana dallo zero $x^* = 0$ dell'equazione e pertanto il metodo diverge per tale scelta del punto iniziale.

k	x_k	$f(x_k)$
0	-2	-1.1071
1	3.5357	1.2952
2	-13.9510	-1.4992

Tabella 1.16: Valori numerici di x_k e $f(x_k)$ relativi all'applicazione del Metodo di Newton alla funzione $arctan(x)$, come rappresentato in Fig. 1.19.

Se applichiamo, invece, il Metodo di Newton scegliendo come punto iniziale $x_0 = 1$, come mostrato nella Tabella 1.17, dopo 3 iterazioni si ottiene una buona approssimazione dello zero:

$$x_3 = -0.001061012... \quad f(x_3) = -0.001061017...$$

ovvero con la scelta di tale punto iniziale il metodo converge.

k	x_k	$f(x_k)$
0	1	0.78
1	-0.5708...	-0.511...
2	0.1168...	0.1168...
3	-0.00106...	-0.00106..

Tabella 1.17: Valori numerici di x_k e $f(x_k)$ relativi all'applicazione del Metodo di Newton alla funzione $arctan(x)$, con $x_0 = 1$ (3 iterazioni).

♣

Per il Metodo di Newton la sola continuità della funzione e della sua derivata non è sufficiente per la convergenza, in quanto quest'ultima dipende dalla scelta del punto iniziale x_0: se tale punto è abbastanza vicino alla soluzione, allora il metodo genera una successione di valori convergente alla soluzione. Da ciò si deduce che, a differenza dei metodi di Tabulazione e Bisezione, il Metodo di Newton è un metodo "a convergenza locale". Condizioni sufficienti per la convergenza del Metodo di Newton sono fornite dal:

Teorema 1.4.1. *Sia $f \in C^2([a,b])$ con $x^* \in [a,b]$ zero di f. Si assuma che $|f'(x)| \geq m$, per ogni $x \in [a,b]$, con $m > 0$. Allora esiste un $\delta > 0$ tale che, se $x_0 \in [a,b]$ è scelto in modo che $|x_0 - x^*| < \delta$, la successione $\{x_k\}$ generata dal Metodo di Newton converge a x^*. Inoltre, si ha* [10]*:*

$$|x_{k+1} - x^*| = \frac{1}{2}\frac{|f''(\xi_k)|}{|f'(x_k)|}|x_k - x^*|^2, \quad \xi_k \in int[x_k, x^*], \quad k = 0, 1, ... \tag{1.23}$$

[10]Dati i numeri reali $\{a, b, \ldots, w\}$, indicheremo con $int[a, b, \ldots, w]$ l'intervallo $[min(a, b, \ldots, w), max(a, b, \ldots, w)]$.

Dimostrazione Posto $M = max|f''(x)|$, definiamo:

$$\delta = \tau(2m/M), \quad \text{con} \quad 0 < \tau < 1. \tag{1.24}$$

Considerata la formula di Taylor di f di punto iniziale x_0, si ha che:

$$f(x^*) = f(x_0) + f'(x_0)(x^* - x_0) + \frac{f''(\xi_0)}{2}(x^* - x_0)^2, \;\; \xi_0 \in int[x_0, x^*]. \tag{1.25}$$

Poiché x^* è uno zero di f, il primo membro di (1.25) è uguale a zero. Dividendo ambo i membri di (1.25) per $f'(x_0)$ (che è non nullo per ipotesi) si ha

$$0 = \frac{f(x_0)}{f'(x_0)} + (x^* - x_0) + \frac{f''(\xi_0)}{2f'(x_0)}(x^* - x_0)^2, \;\; \xi_0 \in int[x_0, x^*]. \tag{1.26}$$

Dalla (1.22) segue

$$|x_1 - x^*| = \frac{1}{2}\frac{|f''(\xi_0)|}{|f'(x_0)|}|x_0 - x^*|^2 \leq \frac{1}{2}\frac{M}{m}|x_0 - x^*|^2. \tag{1.27}$$

Poichè per ipotesi $|x_0 - x^*| < \delta = \tau(2m/M)$, dalla (1.27) segue che:

$$|x_1 - x^*| \leq \tau|x_0 - x^*| < |x_0 - x^*| < \delta. \tag{1.28}$$

Ciò dimostra che il nuovo punto x_1 è anch'esso distante dallo zero x^* a meno della quantità δ. Applicando il ragionamento precedente a partire da x_1 si ottiene:

$$|x_2 - x^*| = \frac{1}{2}\frac{|f''(\xi_1)|}{|f'(x_1)|}|x_1 - x^*|^2 \leq \frac{1}{2}\frac{M}{m}|x_1 - x^*|^2, \tag{1.29}$$

con $\xi_1 \in int[x_1, x^*]$. Inoltre:

$$|x_2 - x^*| \leq \tau|x_1 - x^*| < \tau^2|x_0 - x^*|. \tag{1.30}$$

In generale, abbiamo che:

$$|x_{k+1} - x^*| = \frac{1}{2}\frac{|f''(\xi_k)|}{|f'(x_k)|}|x_k - x^*|^2 \leq \frac{1}{2}\frac{M}{m}|x_k - x^*|^2 \tag{1.31}$$

e quindi la (1.23). Inoltre, dalla (1.30):

$$|x_k - x^*| \leq \tau^k|x_0 - x^*|. \tag{1.32}$$

Poichè $0 < \tau < 1$, si ha $\lim_{k\to\infty} \tau^k = 0$ e quindi $\lim_{k\to\infty} |x_k - x^*| = 0$, che è equivalente alla convergenza a x^* della successione generata da Metodo di Newton. ■

Il Teorema 1.4.1 conferma che la convergenza del Metodo di Newton è garantita se il punto iniziale è sufficientemente vicino alla soluzione.
Si osservi, dalla (1.23), che l'errore ad una certa iterazione è circa il quadrato dell'errore alla iterazione precedente, ovvero il Metodo di Newton converge quadraticamente. Informazioni sul raggio di convergenza si hanno dal seguente:

Teorema 1.4.2. [Raggio di convergenza del Metodo di Newton]
Nelle ipotesi del Teorema 1.4.1, il Metodo di Newton converge alla soluzione x^* se, posto

$$|x_0 - x^*| = \rho \qquad e \qquad K(\rho) = \frac{1}{2}\frac{\max\limits_{|x-x^*|<\rho}|f''(x)|}{\min\limits_{|x-x^*|<\rho}|f'(x)|}, \tag{1.33}$$

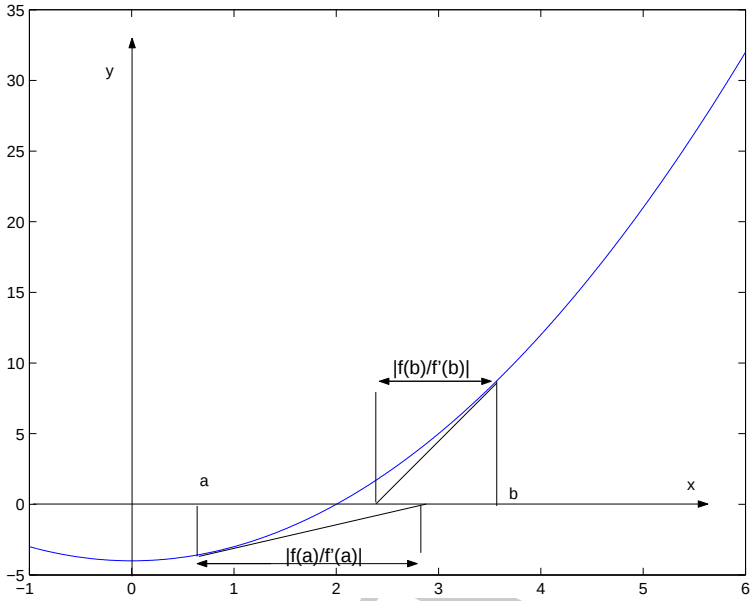

Figura 1.20: Illustrazione grafica del Teorema 1.4.3.

si ha

$$K(\rho)\rho < 1$$

Dimostrazione Dalla (1.23) del Teorema 1.4.1 si ha:

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{e_n^2} = C, \quad \text{con} \quad C = \frac{1}{2}\frac{|f''(x^*)|}{|f'(x^*)|}$$

Sia I un intorno di x^*, tale che

$$\frac{1}{2}\frac{|f''(y)|}{|f'(x)|} \leq m, \quad \forall x \in I, \quad y \in I.$$

se $x_n \in I$, dalla (1.23) è $|e_{n+1}| \leq me_n^2$, che si può riscrivere come $|me_{n+1}| \leq (me_n)^2$. Assumendo che sia $|me_0| < 1$ e che l'intervallo $[x^* - |e_0|, x^* + |e_0|]$ sia un sottoinsieme di I, per induzione si prova che $x_n \in I$ per ogni n e che

$$|e_n| \leq \frac{1}{m}(me_0)^{2^n}$$

Da cui segue che il metodo di Newton converge sempre (ad una radice semplice), assumendo che x_0 sia sufficientemente vicino a x^*, se

$$|me_0| = m|x_0 - x^*| < 1$$

Posto ρ e $K(\rho)$ come nella (1.33) si ha l'asserto. ■

Si può dimostrare che se x^* è una radice multipla, allora il Metodo di Newton converge (solo) linearmente. Si osservi che $K(\rho)\rho \to 0$ se $\rho \to 0$ e, quindi, i due teoremi

assicurano che, se x^* è radice semplice di $f \in C^2([a,b])$, allora il Metodo di Newton converge quadraticamente purché l'approssimazione iniziale sia scelta sufficientemente vicina a x^*. La convergenza quadratica implica che (almeno in vicinanza della soluzione) le cifre corrette della soluzione approssimata raddoppiano ad ogni iterazione.
Il teorema seguente, che ci limiteremo ad enunciare [3], fornisce una condizione sufficiente per la convergenza del Metodo di Newton, indipendentemente dalla scelta del punto iniziale. Chiaramente in questo caso le ipotesi sulla funzione f sono più forti.

Teorema 1.4.3. *Sia $f \in C^2([a,b])$; si supponga inoltre che sia $f(a)f(b) < 0$ e nell'intervallo $[a,b]$ sia $f' \neq 0$ e f'' di segno costante. Se*

$$\left|\frac{f(a)}{f'(a)}\right| < b-a, \quad \left|\frac{f(b)}{f'(b)}\right| < b-a, \tag{1.34}$$

allora il Metodo di Newton converge comunque si scelga il punto iniziale x_0 nell'intervallo $[a,b]$.

Le condizioni espresse dal Teorema 1.4.3 possono essere geometricamente (vedi Fig. 1.20) interpretate come segue: la funzione f è strettamente monotona nell'intervallo $[a,b]$, non cambia concavità in tale intervallo, assume valori di segno discorde agli estremi e le tangenti alla curva di equazione $y = f(x)$ nei punti di ascissa a e b intersecano l'asse x in punti interni all'intervallo $[a,b]$.

1.5 Il Metodo delle Secanti

♣ **Esempio 1.26.** Consideriamo la funzione:

$$f(x) = 0.2x - log(x) \ .$$

Assegnati i punti $x_1 = 0.5$ ed $x_2 = 2.2$ ed i valori corrispondenti $f(x_1)$ e $f(x_2)$, consideriamo la retta secante $y = r(x)$ passante per i punti $(x_1,\ f(x_1))$ ed $(x_2,\ f(x_2))$:

$$r(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1) \ . \tag{1.35}$$

La Figura 1.21 mostra che lo zero di f, $x^* = 1.2959...$, si può approssimare con x_3 ottenuto come intersezione di $y = r(x)$ con l'asse delle ascisse.
Tale idea è alla base del metodo detto **Metodo delle Secanti**. ♣

Il **Metodo delle Secanti** può essere interpretato come una variante del Metodo di Newton in cui, al passo $(k+1)-$mo anziché considerare la tangente alla curva di equazione $y = f(x)$ nel punto di ascissa x_k si costruisce la secante alla curva nei punti di ascissa x_k

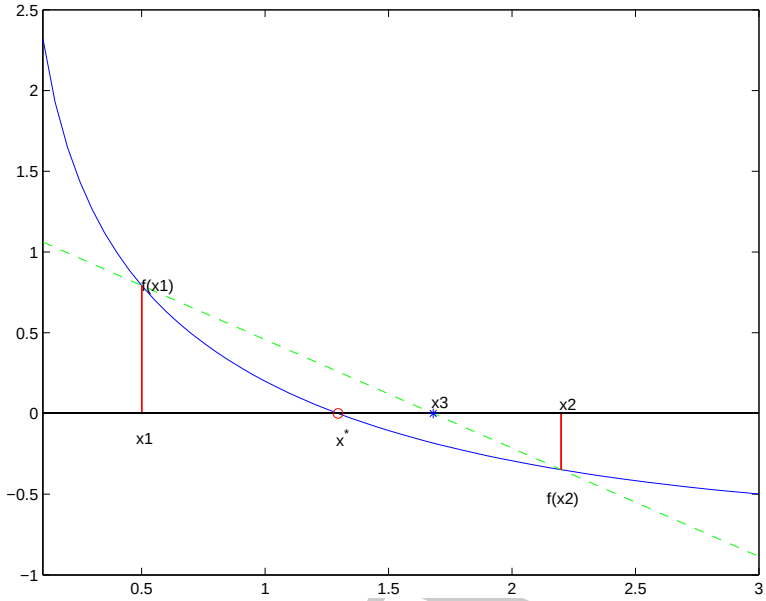

Figura 1.21: Illustrazione grafica del Metodo delle Secanti applicato alla funzione $f(x) = 0.2x - log(x)$, a partire dai punti $x_1 = 0.5$ e $x_2 = 2.2$; x_3 è il punto di intersezione con l'asse delle x della retta secante la curva, passante per i punti $(x_1, f(x_1))$ ed $(x_2, f(x_2))$.

e x_{k-1} rispettivamente; in altre parole, x_{k+1} è calcolata come intersezione di tale secante con l'asse delle ascisse.
Come il Metodo di Newton, anche il Metodo delle Secanti procede in modo iterativo: dati due valori iniziali x_0 e x_1 si calcola x_2, quindi noti x_1 e x_2 si calcola x_3 e così via. La generica iterazione k può essere sintetizzata come segue:

- si considera la retta $y = s_k(x)$, secante la curva $y = f(x)$ nei punti $(x_{k-1}, f(x_{k-1}))$ e $(x_k, f(x_k))$, con:

$$s_k(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k),$$

  e si assume come approssimazione dello zero il valore, x_{k+1}, ottenuto intersecando tale retta con l'asse delle ascisse, cioè:

$$s_k(x_{k+1}) = 0 \Leftrightarrow x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \tag{1.36}$$

In Fig. 1.22 è illustrata graficamente l'applicazione del Metodo delle Secanti.
Di seguito riportiamo l'algoritmo per il Metodo delle Secanti.

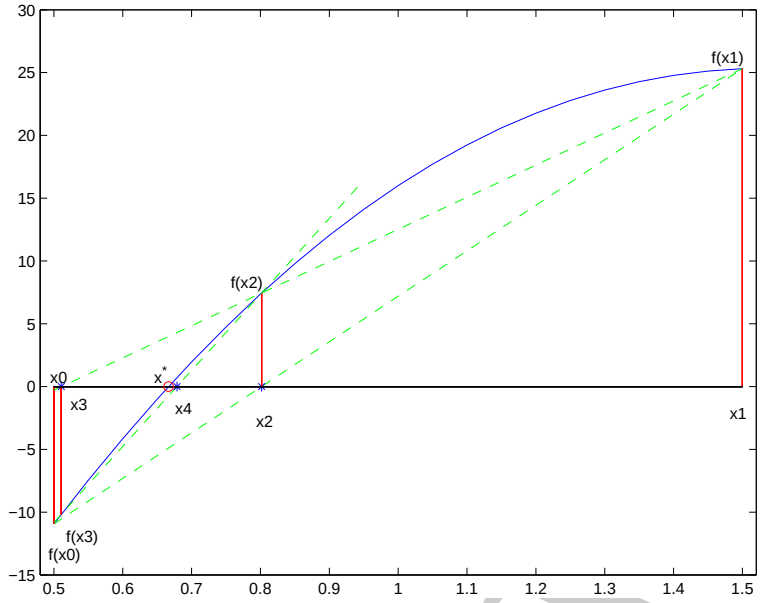

Figura 1.22: Illustrazione grafica del Metodo delle Secanti applicato alla risoluzione di $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$ con $x_0 = 0.5$ e $x_1 = 1.5$ (tre iterazioni): x_2 rappresenta il punto di intersezione con l'asse delle x della retta secante la curva, passante per i punti $(x_0,\ f(x_0))$ e $(x_1,f(x_1))$, x_3 il punto di intersezione con l'asse delle x della retta secante la curva, passante per i punti $(x_1,\ f(x_1))$ e $(x_2,f(x_2))$ e x_4 il punto di intersezione con l'asse delle x della retta secante la curva, passante per i punti $(x_2,\ f(x_2))$ e $(x_3,f(x_3))$.

procedure fxsec(**input:** $x0$,f,$itmax$, **out:** $xsec$,$fxsec$)

/# SCOPO: calcolo di un'approssimazione dello zero
/# di una funzione mediante il Metodo delle Secanti
/# SPECIFICHE PARAMETRI:

var: $x0$	: **reale**	{*approssimazione iniziale dello zero*}
var: f	: **funzione esterna**	{*funzione di cui si cerca lo zero*}
var: $itmax$	: **intero**	{*numero massimo di iterazioni*}

Procedura 1.3: Algoritmo del Metodo delle Secanti - continua

```
  var: xsec    : reale   {approssimazione dello zero}
  var: fxsec   : reale   {valore di f in xsec}
/# INIZIO ISTRUZIONI:
    k := 1
/# generazione della successione delle approssimazioni
    repeat
      x_{k+1} := x_k − f(x_k)(x_k − x_{k−1})/(f(x_k) − f(x_{k−1}))   {calcolo}
                                                  {dell'approssimazione}
      k := k + 1
    until (condizione di arresto verificata)
end fxsec
```

Procedura 1.3: Algoritmo del Metodo delle Secanti -fine

1.5.1 Applicabilità

Il Metodo delle Secanti offre, rispetto al Metodo di Newton il vantaggio di non richiedere la conoscenza dell'espressione della derivata della funzione. Tuttavia, alcuni problemi computazionali derivano dal calcolo di:

$$\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{1.37}$$

che, in vicinanza della soluzione, è un rapporto fra due quantità che possono diventare numericamente nulle.

1.5.2 Convergenza

♣ **Esempio 1.27.** Applichiamo il Metodo di Newton e quello delle Secanti per risolvere in $[-1, 1]$:

$$x^4(cosx - sinx) = 0$$

che ha uno zero in $x^* = \pi/4$.
In Fig. 1.23 è rappresentato il grafico di $f(x) = x^4(cosx - sinx)$.

I risultati in Tabella 1.18 mostrano che, per l'esempio in esame, almeno nelle vicinanze della soluzione, i due metodi hanno comportamenti simili; ciò non deve meravigliare visto che, se x_n e x_{n-1} sono vicini, allora la secante è una buona approssimazione della tangente. ♣

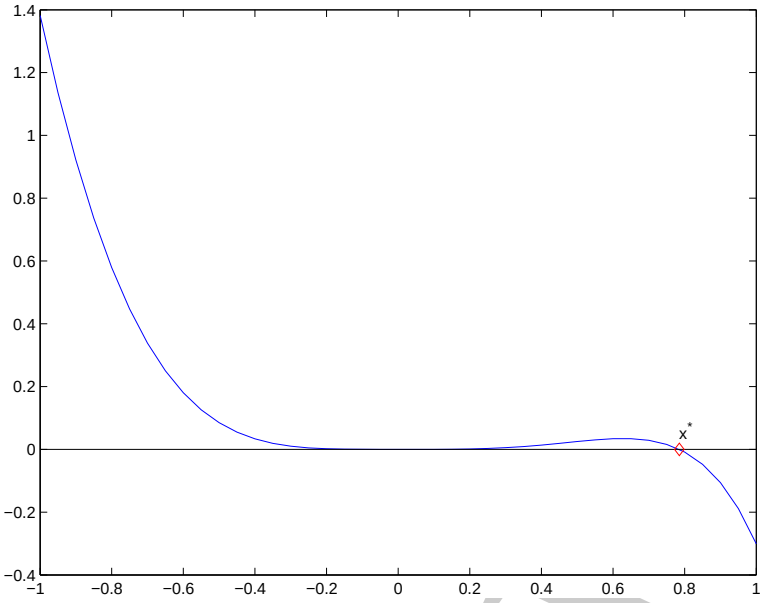

Figura 1.23: Grafico di $f(x) = x^4(cos\, x - sin\, x)$ in [-1,1]. Con il simbolo '$\diamondsuit$' si indica uno zero della funzione.

Il teorema che segue, ottenuto con argomenti del tutto analoghi a quelli utilizzati per la dimostrazione del Teorema 1.4.1, fornisce condizioni sufficienti per la convergenza del Metodo delle Secanti.

Teorema 1.5.1. [Convergenza del Metodo delle Secanti]
Si supponga $f \in C^2([a,b])$ e x^* sia soluzione di

$$f(x) = 0,\ x \in [a,b] \subseteq A,$$

dove $f : A \subseteq \Re \to \Re$ è una funzione con derivata f' definita in $[a,b]$, tale che $f'(x^*) \neq 0$. Se x_0 e x_1 sono scelti sufficientemente vicini a x^* la successione $\{x_n\}$ generata dal Metodo delle Secanti converge a x^* ed, inoltre

$$e_{n+1} = -\frac{1}{2}\frac{f''(\xi_n)}{f'(\zeta_n)} e_n e_{n-1}, \quad \xi_n, \zeta_n \in int[x^*, x_{n-1}, x_n] \tag{1.38}$$

dove $e_n = x_n - x^*$ è l'errore locale di troncamento.

Dimostrazione Consideriamo la retta secante la curva $y = f(x)$ nei punti $(x_{k-1}, f(x_{k-1}))$ e $(x_k, f(x_k))$,

$$m_{k+1}(x) = f(x_{k-1}) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_{k-1}).$$

n	x_n (Secanti)	$x_n - x^*$(Secanti)	x_n (Newton)	$x_n - x^*$(Newton)
1	1.8081e+00	1.0227e+00	1.8128e+00	1.0274e+00
2	1.6030e+00	8.1763e-01	1.4570e+00	6.7161e-01
3	1.3736e+00	5.8816e-01	1.2072e+00	4.2182e-01
4	1.2141e+00	4.2872e-01	1.0268e+00	2.4137e-01
5	1.0797e+00	2.9433e-01	9.0111e-01	1.1571e-01
6	9.7563e-01	1.9023e-01	8.2444e-01	3.9040e-02
7	8.9564e-01	1.1024e-01	7.9160e-01	6.2030e-03
8	8.3931e-01	5.3909e-02	7.8559e-01	1.8844e-04
9	8.0512e-01	1.9720e-02	7.8540e-01	1.8064e-07
10	7.8977e-01	4.3689e-03	7.8540e-01	1.6620e-13
11	7.8581e-01	4.0705e-04	7.8540e-01	0.0000e+00
12	7.8541e-01	8.9200e-06	7.8540e-01	1.1102e-16
13	7.8540e-01	1.8467e-08	7.8540e-01	0.0000e+00
14	7.8540e-01	8.3888e-13	7.8540e-01	1.1102e-16
15	7.8540e-01	1.1102e-16	7.8540e-01	0.0000e+00

Tabella 1.18: Metodo di Newton e Metodo delle Secanti applicati alla risoluzione di $x^4(cos\, x - sin\, x) = 0$, con $x_0 = 2.4$ e $x_1 = 2.6$ (Secanti), $x_0 = 2.5$ (Newton) e $x^* = \pi/4$.

Sia $x \in int[x_{k-1}, x_k]$, la formula di Newton per il polinomio di Lagrange interpolante f nei nodi x_{k-1}, x_k e x è:

$$\begin{aligned} f(x) &= m_{k+1}(x) + f[x_{k-1}, x_k, x](x - x_{k-1})(x - x_k) = \\ &= f(x_{k-1}) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_{k-1}) + f[x_{k-1}, x_k, x](x - x_{k-1})(x - x_k), \end{aligned} \tag{1.39}$$

dove

$$f[x_{k-1}, x_k, x] = \frac{f[x_k, x] - f[x_{k-1}, x_k]}{x - x_{k-1}} = \frac{\frac{f(x)-f(x_k)}{x-x_k} - \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}}{x - x_{k-1}}.$$

Scelto $x = x_{k+1}$ il punto di intersezione della secante m_{k+1} con l'asse delle ascisse; allora $m_{k+1}(x_{k+1}) = 0$, ovvero:

$$f(x_{k-1}) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x_{k+1} - x_{k-1}) = 0. \tag{1.40}$$

Ponendo $x = x^*$ nella (1.39) si ha

$$f(x^*) = 0 \quad \Rightarrow \quad f(x_{k-1}) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x^* - x_{k-1}) + f[x_{k-1}, x_k, x^*](x^* - x_{k-1})(x^* - x_k) = 0. \tag{1.41}$$

Sottraendo membro a membro, la (1.40) dalla (1.41), si ottiene

$$\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x^* - x_{k+1}) + f[x_{k-1}, x_k, x^*](x^* - x_{k-1})(x^* - x_k) = 0$$

per cui, essendo $e_{k+1} = x^* - x_{k+1}$, si ha:

$$e_{k+1} = -\frac{f[x_{k-1}, x_k, x^*]}{f[x_{k-1}, x_k]}(x^* - x_{k-1})(x^* - x_k),$$

Per ipotesi, $f \in C^2([a,b])$, da cui, se x_{k-1} e x_k sono sufficientemente vicini a x^*, cioè se:

$$\begin{aligned} x_{k-1} &\rightarrow x^*,\ k \rightarrow +\infty \\ x_k &\rightarrow x^*,\ k \rightarrow +\infty \end{aligned}$$

per la definizione di differenza divisa di ordine due su tre nodi coincidenti, si ha:

$$f[x_{k-1}, x_k, x^*] = \frac{f''(\xi_k)}{2} \quad \text{con} \quad \xi_k \in int[x^*, x_{k-1}, x_k].$$

Inoltre, essendo $f'(x^*) \neq 0$ per ipotesi, esiste un intorno di x^* in cui $|f'| > 0$ e, dunque, un punto ζ_k che vi appartiene, per il quale $f'(\zeta_k) \neq 0$; inoltre, se $\zeta_k \in int[x_{k-1}, x_k]$, per il Teorema del valor medio,

$$f[x_{k-1}, x_k] = f'(\zeta_k),$$

da cui

$$e_{k+1} = -\frac{1}{2}\frac{f''(\xi_k)}{f'(\zeta_k)} e_k e_{k-1}$$

e, dunque, l'asserto. ■

Calcoliamo l'ordine di convergenza del metodo delle Secanti.

Supponiamo che il Metodo delle Secanti sia convergente di ordine p, ovvero, posto $e_n = x^* - x_n$, supponiamo che:

$$|e_{n+1}| \leq C \cdot |e_n|^p$$

Dal Teorema 1.5.1 abbiamo ottenuto una stima dell'errore del tipo

$$|e_{n+1}| \leq K_n \cdot |e_n| \cdot |e_{n-1}|$$

con $K_n = \frac{1}{2}\frac{f''(\xi_n)}{f'(\zeta_n)}$. Quindi

$$C \cdot |e_n|^p \leq K_n \cdot |e_n| \cdot C^{-\frac{1}{p}} \cdot |e_n|^{\frac{1}{p}}$$

da cui segue la relazione

$$C \cdot |e_n|^p \approx K_n \cdot C^{-\frac{1}{p}} \cdot |e_n|^{\frac{1}{p}+1}$$

che sussiste, per ogni n, se

$$p = \frac{1}{p} + 1 \Rightarrow p^2 - p - 1 = 0 \Rightarrow p = \frac{1 \pm \sqrt{5}}{2} \tag{1.42}$$

e se

$$K_n \approx C^{1+\frac{1}{p}}$$

che, dalla (1.42), si può esprimere anche come

$$K_n \approx C^p.$$

Inoltre, dalla definizione di convergenza di ordine p, deve essere $p > 0$, da cui

$$p = \frac{1+\sqrt{5}}{2}$$

quindi, per $K >> 1$,

$$|e_{n+1}| \approx K_n^{\frac{1}{p}} \cdot |e_n|^p, \quad \text{con} \quad p = \frac{1+\sqrt{5}}{2} = 1.618\ldots.$$

Concludendo, il Metodo delle Secanti ha ordine di convergenza uguale a:

$$(1+\sqrt{5})/2 \simeq 1.62$$

e quindi converge piú lentamente del Metodo di Newton (che ha ordine di convergenza 2). Per un equo raffronto fra i due metodi si deve tenere conto del costo computazionale del Metodo di Newton, maggiore rispetto a quello delle Secanti; in particolare, se si assume che la complessità computazionale per il calcolo di f' è circa uguale a quella per il calcolo di f (assunzione ragionevole in molti casi), un'iterazione del primo metodo ha un costo computazionale di circa il doppio di una iterazione del secondo. Poiché per il Metodo delle Secanti si ha:

$$|e_{n+2}| \simeq |e_{n+1}|^{1.62} \simeq \left(|e_n|^{1.62}\right)^{1.62} \simeq |e_n|^{2.6244},$$

quest'ultimo, a parità di costo computazionale, può risultare più veloce del Metodo di Newton.

1.6 I metodi ibridi. Il Metodo di Dekker-Brent

I metodi ibridi si propongono di coniugare la robustezza (capacità di fornire sempre una soluzione) dei metodi globali con l'efficienza (veloce convergenza) dei metodi locali. Un esempio di tali metodi è il Metodo di Dekker-Brent che rappresenta una combinazione dei metodi di Bisezione e Secanti.
Nel Metodo di Dekker-Brent, a partire da un intervallo di ricerca $[a, b]$:

- la prima approssimazione si calcola utilizzando il Metodo delle Secanti.
- al fine di garantire la convergenza, non appena l'approssimazione calcolata sembra allontanarsi dalla soluzione, si utilizza il Metodo di Bisezione.
- al fine di garantire una velocitá di convergenza sufficientemente alta, si utilizza quante piú volte è possibile il Metodo delle Secanti.

Nel dettaglio, l'algoritmo che descrive il Metodo di Dekker-Brent è il seguente:

- **inizializzazione:** $x_0 = a,\ x_1 = b,\ y_1 = a,\ y_{-1} \equiv y_0 = a + 1$.

- $k+1$**-mo passo (** $k \geq 1$ **):** viene calcolato, con il Metodo delle Secanti, il valore:

$$x_s = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} \tag{1.43}$$

  Se x_s appartiene all'intervallo che ha per estremi x_k e y_k, e $y_k \neq y_{k-2}$, allora

$$x_{k+1} \equiv x_s\ ,$$

  altrimenti x_{k+1} è calcolato con il Metodo di Bisezione, cioè

$$x_{k+1} = \frac{x_k + y_k}{2}.$$

  Il valore di y_{k+1}, l'altro estremo dell'intervallo di ricerca, viene calcolato come:

$$y_{k+1} = \begin{cases} x_k & \text{se } f(x_k)f(x_{k+1}) < 0 \\ y_k & \text{altrimenti} \end{cases} \tag{1.44}$$

Osserviamo che se $y_k = y_{k-2}$ vuol dire che, per almeno 2 iterazioni, l'estremo dell'intervallo di ricerca $int[x_k, y_k]$ è rimasto fisso, ovvero la soluzione potrebbe trovarsi in prossimità di questo estremo. In tal caso si utilizza il Metodo di Bisezione. Infine, la (1.44) assicura che la funzione assume segno discorde nei punti x_{k+1} e y_{k+1}, pertanto l'intervallo che ha tali estremi (che quindi contiene una soluzione) ha ampiezza minore di quella dell'intervallo precedente.

♣ **Esempio 1.28.** Applichiamo il Metodo di Dekker-Brent alla risoluzione dell'equazione:

$$3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$$

In Figura 1.24 è illustrata l'applicazione del Metodo di Dekker-Brent alla risoluzione dell'equazione in $[0, 2]$. In Tabella 1.19 sono riportati i risultati ottenuti. Confrontando tali risultati con quelli riportati in Tabella 1.11, ottenuti applicando il Metodo di Bisezione, risulta evidente la superiorità del metodo ibrido.

♣

Il Metodo di Dekker-Brent eredita dal Metodo di Bisezione la proprietà di essere un metodo globalmente convergente; inoltre, in prossimità della soluzione, x_k viene calcolata con il Metodo delle Secanti (1.43), pertanto eredita dal Metodo delle Secanti l'ordine di convergenza.

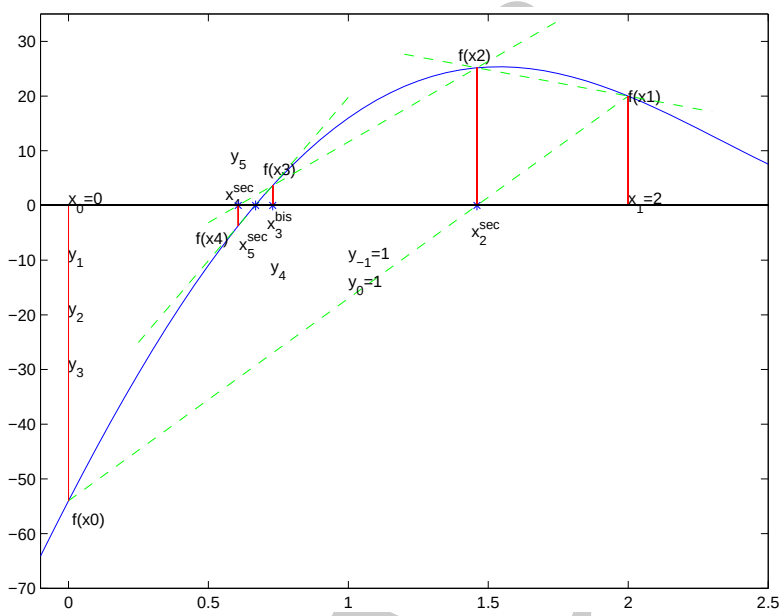

Figura 1.24: Illustrazione grafica del Metodo di Dekker-Brent applicato all'equazione $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$ con $x_0 = 0$ e $x_1 = 2$ (4 iterazioni).

k	x_k	$f(x_k)$
1	1.4595e+00	2.5172e+01
2	7.2973e-01	3.6369e+00
3	3.6486e-01	-2.1155e+01
4	6.7621e-01	5.6810e-01
5	6.6806e-01	8.3598e-02
6	6.6666e-01	-4.6986e-04
7	6.6667e-01	3.8397e-07

Tabella 1.19: Valori numerici di x_k e $f(x_k)$ relativi al Metodo di Dekker-Brent applicato all'equazione $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$ con $x_0 = 0$, $x_1 = 2$ (7 iterazioni).

1.7 Metodi *one point*. Il Metodo del punto fisso

Nel Metodo di Newton l'approssimazione x_{n+1} viene calcolata a partire da x_n come

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \ , \tag{1.45}$$

ovvero, indicata con $\Phi(x)$ la funzione di iterazione:

$$\Phi(x) = x - \frac{f(x)}{f'(x)}, \tag{1.46}$$

x_{n+1} è calcolata mediante l'iterazione funzionale:

$$x_{n+1} = \Phi(x_n) \tag{1.47}$$

Metodi del tipo (1.47) sono detti metodi iterativi **one-point** perché ciascuna iterata viene calcolata in funzione solo dell'iterata precedente.

Per risolvere un'equazione non lineare $f(x) = 0$, è possibile costruire metodi iterativi *one-point*, a partire da considerazioni di tipo geometrico (come per il Metodo di Newton) o analitico. Ad esempio, l'equazione $f(x) = 0$ può essere scritta come:

$$x = f(x) + x, \tag{1.48}$$

da cui

$$x_{n+1} = f(x_n) + x_n, \tag{1.49}$$

che è proprio un metodo iterativo *one-point* in cui la funzione di iterazione è data dall'espressione a secondo membro della (1.48).

Più in generale, se $x = g(x)$ è una equazione equivalente ad $f(x) = 0$ [11], un metodo iterativo del tipo (1.47) in cui la funzione g sia presa come funzione di iterazione, viene detto **del punto fisso** o di **iterazione funzionale**.
Data una funzione di iterazione Φ ed un punto iniziale x_0, è sempre possibile considerare la successione x_n generata mediante la (1.47); si osservi che, se Φ è continua e $x_n \to x^*$, allora

$$x^* = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \Phi(x_{n-1}) = \Phi(x^*),$$

cioè x^* è punto fisso per la funzione Φ. Graficamente un punto fisso è una intersezione del grafico della curva di equazione $y = \Phi(x)$ con la bisettrice del primo e terzo quadrante del piano cartesiano:

$$\begin{cases} y = \Phi(x) \\ y = x \end{cases} \Rightarrow \ \Phi(x) = x \tag{1.50}$$

[11] Un punto x^* è detto *punto fisso* della funzione g se è soluzione dell'equazione $x = g(x)$.

1.7.1 Convergenza

Il teorema seguente [12] può essere utilizzato per stabilire condizioni sufficienti alla convergenza del Metodo del punto fisso.

Teorema 1.7.1. [Teorema delle contrazioni]
Sia Φ una funzione continua e derivabile in $[a, b]$ a valori in $[a, b]$. Se esiste un numero $K < 1$ tale che $|\Phi'(x)| \leq K$ in $[a, b]$, allora

- *la funzione Φ ammette un unico punto fisso x^* in $[a, b]$;*
- *comunque si scelga x_0 in $[a, b]$, la successione generata mediante (1.47) converge a x^*.*

Una stessa equazione può ovviamente essere espressa nella forma $x = g(x)$ in infiniti modi, dando luogo a diverse funzioni di iterazione per il punto fisso che non sempre portano alla convergenza della successione generata mediante (1.47).

♣ **Esempio 1.29.** Si supponga di risolvere l'equazione:

$$x^3 - 3 = 0$$

che ovviamente ammette l'unica soluzione $\overline{x} = \sqrt[3]{3}$. Il Metodo del punto fisso può essere utilizzato con

$$g(x) = 3/x^2,$$

essendo l'equazione di partenza equivalente a:

$$x = 3/x^2$$

Si ottiene, quindi, la formula iterativa

$$x_{n+1} = \frac{3}{x_n^2}$$

che, scegliendo come punto iniziale $x_1 = 1$, genera una successione che non converge a $\overline{x}$ (addirittura non regolare). Si noti che, essendo $|g'(\overline{x})| = |-2| > 1$, non esiste alcun intervallo contenente la radice in cui valgono le ipotesi del Teorema 1.7.1. ♣

♣ **Esempio 1.30.** Si supponga di risolvere l'equazione dell'esempio precedente

$$x^3 - 3 = 0$$

Tale equazione ammette anche una formulazione equivalente:

$$x = \sqrt{3/x}$$

[12] La dimostrazione del **Teorema delle contrazioni** è nel **Capitolo 4**, **§4.10** (**Teorema 4.10.2**).

a partire dalla quale è possibile definire la formula iterativa

$$x_{n+1} = \sqrt{3/x_n}$$

Si osservi che per tale formulazione si verifica $g'(\overline{x}) = |-1/2| < 1$. Quindi è possibile trovare intervalli, di ampiezza sufficientemente piccola contenenti la radice $\overline{x}$, tali che le ipotesi del Teorema 1.7.1 siano verificate. ♣

♣ **Esempio 1.31.** Volendo risolvere l'equazione di Keplero (1.4), si osservi che questa è equivalente all'equazione:

$$x = E \sin x + M$$

e quindi:

$$x_{n+1} = E \sin x_n + M \tag{1.51}$$

Essendo $|g'(x)| = |E \cos x| \leq E$ e per ipotesi $E < 1$, si ha, in virtù del Teorema 1.7.1, che la successione generata mediante la (1.51) converge alla soluzione dell'equazione di Keplero, qualunque sia la scelta del punto iniziale. ♣

Per quanto riguarda il raggio e la velocità di convergenza, supposta la validità delle ipotesi del Teorema 1.7.1, si osserva che, dal Teorema di Lagrange (o del valor medio), per l'errore di troncamento risulta:

$$e_{n+1} = x_{n+1} - x^* = g(x_n) - g(x^*) = g'(\xi_n)(x_n - x^*) = g'(\xi_n)e_n$$

con ξ_n interno all'intervallo $int[x_{n+1}, \overline{x}]$ e, quindi,

$$\frac{e_{n+1}}{e_n} = g'(\xi_n)$$

Se g' è continua in $\overline{x}$ e $|g'(\overline{x})| < 1$, il Metodo del punto fisso ha convergenza lineare con raggio di convergenza $|g'(\overline{x})|$.
Nell'esempio 1.30, se si sceglie $g(x) = \sqrt{3/x}$ si ottiene che il raggio di convergenza è $1/2$.

1.8 Un cenno al metodo di Newton per sistemi non lineari

♣ **Esempio 1.32.** Consideriamo il sistema di $n = 2$ equazioni non lineari in $m = 2$ incognite:

$$F(\overline{x}) = F(x_1, x_2) = \begin{bmatrix} x_1 + x_2 - 3 \\ x_1^2 + x_2^2 - 9 \end{bmatrix} = 0 \tag{1.52}$$

che ammette come soluzione i punti $(3,0)$ e $(0,3)$.
Posto $\overline{x} = (x_1, x_2)$ e $x^0 = (x_1^0, x_2^0)$, introdotte le funzioni $F_1, F_2 : \Re^2 \to \Re$, dove

$$\begin{array}{l} F_1(\overline{x}) = x_1 + x_2 - 3 \\ F_2(\overline{x}) = x_1^2 + x_2^2 - 9 \end{array},$$

in analogia al Metodo di Newton per una equazione non lineare, consideriamo lo sviluppo in serie di Taylor di $F = (F_1, F_2)$ troncato al primo ordine, espresso in forma vettoriale:

$$F(\overline{x}) \simeq F(x^0) + J(x^0)(\overline{x} - x^0)$$

di punto iniziale x^0 [13]. Sostituendo a F lo sviluppo troncato, riformuliamo il problema del calcolo di uno zero di F nel modo seguente:

$$F(\overline{x}) = 0 \Leftrightarrow F(x^0) + J(x^0)(\overline{x} - x^0) = 0 \Leftrightarrow J(x^0)(\overline{x} - x^0) = -F(x^0)$$

Posto $\overline{x} - x^0 = s_0$, si ha quindi:

$$F(\overline{x}) = 0 \Leftrightarrow J(x^0)s_0 = -F(x^0)$$

Se $x^0 = (1,5)$, s_0 è soluzione del sistema lineare:

$$\begin{pmatrix} 1 & 1 \\ 2 & 10 \end{pmatrix} s_0 = - \begin{pmatrix} 3 \\ 17 \end{pmatrix}$$

Risolvendo tale sistema, si ottiene $s_0 = (-13/8, -11/8)$, cioè $\overline{x} = x^0 + s_0 = (-.625, 3.625)$. Tale valore fornisce una prima approssimazione di uno dei due zeri di F. Riapplicando lo stesso procedimento e assumendo, come punto iniziale per lo sviluppo in serie di Taylor di F, troncato al primo ordine, il punto $\overline{x}^1 = x^0 + s_0$, si ha:

$$F(\overline{x}_1) + J(\overline{x}_1)(\overline{x} - \overline{x}_1) = 0$$

dove:

$$J(\overline{x}_1) = \begin{pmatrix} 1 & 1 \\ -5/4 & 29/4 \end{pmatrix}$$

da cui:

$$F(\overline{x}_1) + J(\overline{x}_1)(\overline{x} - \overline{x}_1) = 0 \Leftrightarrow J(\overline{x}_1)(\overline{x} - \overline{x}_1) = -F(\overline{x}_1)$$

Posto $\overline{x} - \overline{x}_1 = s_1$, si ha quindi:

$$J(\overline{x}_1)s_1 = -F(\overline{x}_1)$$

cioè s_1 si ottiene come soluzione del sistema:

$$\begin{pmatrix} 1 & 1 \\ -5/4 & 29/4 \end{pmatrix} s_1 = - \begin{pmatrix} 0 \\ 145/32 \end{pmatrix}$$

Risolvendo tale sistema si ha $s_1 = (145/272, -145/272)$, da cui $\overline{x} = \overline{x}_1 + s_1 = (-.092, 3.092)$.
Dopo due iterazioni si ottiene una approssimazione dello zero $(0,3)$ corretta a due cifre significative.

♣

[13] Il simbolo $J(x^0)$ indica la matrice Jacobiana di F calcolata in x^0:

$$J(x^0) = \begin{pmatrix} 1 & 1 \\ 2x_1 & 2x_2 \end{pmatrix}$$

dove

$$J_{i,j}(x^0) = \frac{\partial F_i}{\partial x_j}(x^0)\,.$$

Nel caso in cui la funzione F sia una funzione vettoriale ed, in particolare,

$$F : \Re^n \rightarrow \Re^n$$

il **Metodo di Newton** si definisce in maniera del tutto analoga al caso scalare.

Considerato lo sviluppo in serie di Taylor di F, troncato al primo ordine [14], di punto iniziale $\overline{x}_k$:

$$F(\overline{x}) \simeq F(\overline{x}_k) + J(\overline{x}_k)(\overline{x} - \overline{x}_k)$$

dove il simbolo $J(\overline{x}_k)$ indica la matrice Jacobiana di F valutata in $\overline{x}_k$:

$$J(\overline{x}_k) = \begin{pmatrix} \frac{\partial F_1(\overline{x}_k)}{\partial x_1} & \dots & \frac{\partial F_1(\overline{x}_k)}{\partial x_n} \\ \dots & \dots & \\ \frac{\partial F_n(\overline{x}_k)}{\partial x_1} & \dots & \frac{\partial F_n(\overline{x}_k)}{\partial x_n} \end{pmatrix}$$

dove

$$F(x^1, \dots, x^n) = (F_1(x^1, \dots, x^n), \dots, F_n(x^1, \dots, x^n))$$

il **Metodo di Newton** per sistemi di equazioni non lineari costruisce una successione di approssimazioni di uno zero di F, $\{\overline{x}_k\}_{k=1,2,\dots}$, in cui ciascun valore $\overline{x}_k$ è ottenuto sostituendo a F lo sviluppo in serie di Taylor di F, troncato al primo ordine, di punto iniziale $\overline{x}_{k-1}$:

$$F(\overline{x}_{k-1}) + J(\overline{x}_{k-1})(\overline{x}_k - \overline{x}_{k-1}) = 0 \Leftrightarrow J(\overline{x}_{k-1})(\overline{x}_k - \overline{x}_{k-1}) = -F(\overline{x}_{k-1})$$

Posto $s_k = \overline{x}_k - \overline{x}_{k-1}$, al generico passo k il calcolo dell'approssimazione $\overline{x}_k$ richiede la risoluzione di un sistema di equazioni non lineari del tipo:

$$J(\overline{x}_{k-1})s_k = -F(\overline{x}_{k-1}), \qquad \overline{x}_k = \overline{x}_{k-1} + s_k$$

L'applicazione del metodo di Newton nel caso di una funzione vettoriale richiede, ad ogni passo, la costruzione della matrice Jacobiana, in analogia al caso scalare, e la risoluzione di un sistema di equazioni lineari. È evidente che entrambe le richieste aumentano in maniera significativa la complessità computazionale del metodo rispetto al caso scalare. A tal proposito, esistono svariate varianti del metodo di Newton, che si specializzano sia in base al tipo di approssimazione della matrice Jacobiana, sia in base al tipo di solutore del sistema lineare. Riguardo alla convergenza, in analogia al caso scalare, il teorema seguente garantisce, sotto opportune condizioni, che la velocità di convergenza del Metodo di Newton per sistemi non lineari sia quadratica [4].

[14] Ovviamente per la funzione F occorre assumere che sia dotata di derivate prime.

Teorema 1.8.1. *Sia $F : \Re^n \to \Re^n$, tale che $F \in C^1(D)$, con $D \subset \Re^n$ convesso. Sia inoltre $x^* \in D$ uno zero di F per cui la matrice Jacobiana $J(x)$ sia non singolare. Se $J(x)$ è Lipshitziana in un intorno di x^*, allora la successione generata dal metodo di Newton converge a x^* ed, inoltre, la velocità di convergenza di tale successione è quadratica.*

1.9 Condizionamento delle equazioni non lineari

Analizziamo il condizionamento del problema relativo al calcolo di uno zero di una funzione non lineare.

♣ **Esempio 1.33.** L'equazione:

$$f(x) \equiv x^8 - 10^{-4} = 0 \tag{1.53}$$

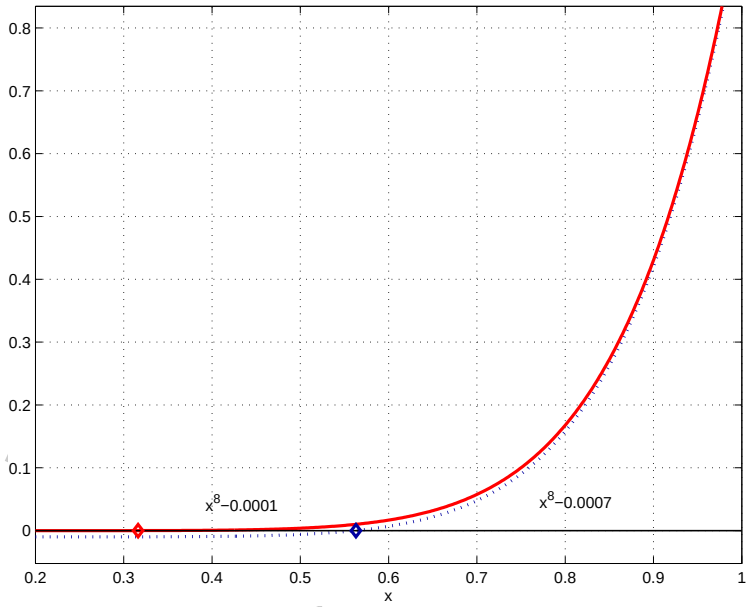

Figura 1.25: Grafico di $f(x) \equiv x^8 - 10^{-4}$ (linea continua) e di $\tilde{f}(x) \equiv x^8 - 7 \times 10^{-4}$ ('•') in $[0.2, 1]$. Con il simbolo '◇' si indicano gli zeri delle funzioni.

ha due zeri, di cui uno positivo $x = 10^{-1/2} = 0.316227\ldots$. L'equazione

$$\tilde{f}(x) \equiv x^8 - 7 \times 10^{-4} = 0 \tag{1.54}$$

che ha uno zero in $x = 0.403308\ldots$ differisce dall'equazione (1.53) della quantità

$$\delta = \max_{[0,1]} |f(x) - \tilde{f}(x)| = 0.6 \times 10^{-3}$$

L'equazione (1.54) è quindi ottenuta dall'equazione (1.53) a seguito di una perturbazione al più uguale alla quantità δ. Tale perturbazione ha condotto ad una soluzione dell'equazione $f(x) = 0$ completamente diversa. In tal caso

$$\sigma = |0.316227 - 0.403308| = 0.8708062\ldots \times 10^{-1}$$

quindi, il problema del calcolo dello zero x^* dell'equazione non lineare $f(x) = 0$ deve ritenersi **mal condizionato.** Osservando l'andamento della funzione f, si nota che, il condizionamento del calcolo di uno zero di una funzione non lineare dipende in maniera significativa dall'andamento della funzione in prossimità dello zero, e quindi dalla derivata prima della funzione. In questo esempio, la funzione cresce molto lentamente ovvero risulta $f'(x^*)$ "piccolo" in modulo.

♣

♣ **Esempio 1.34.** L'equazione:

$$f(x) \equiv 1/x - 100 = 0 \tag{1.55}$$

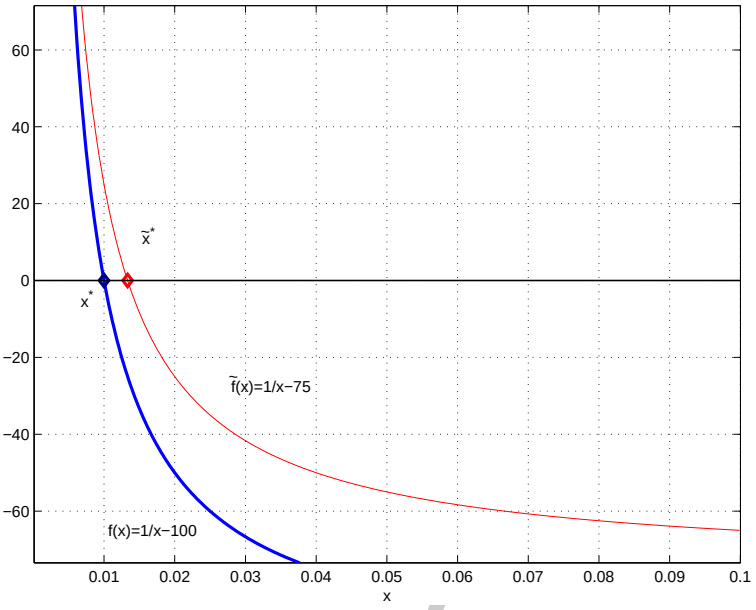

Figura 1.26: Grafico di $f(x) \equiv 1/x - 100$ e di $\tilde{f}(x) \equiv 1/x - 75$ in $[0.001, 0.1]$. Con il simbolo '$\diamondsuit$' si indicano gli zeri delle funzioni.

ha uno zero in $x^* = 0.1 \times 10^{-1}$. L'equazione

$$\tilde{f}(x) \equiv 1/x - 75 = 0 \tag{1.56}$$

ha uno zero in $x = 0.13333\ldots \times 10^{-1}$. Posto

$$\delta = \max_{(0,0.1]} |f(x) - \tilde{f}(x)| = \left|\frac{1}{x} - 100 - \frac{1}{x} + 75\right| = |-35| = 35$$

segue che

$$\tilde{f}(x) = f(x) + \epsilon, \quad |\epsilon| \leq \delta = 35$$

ovvero $\tilde{f}$ differisce da f di una quantità al più δ. Tale perturbazione sulla funzione f ha condotto ad una soluzione $\tilde{x}^*$ (lo zero di $\tilde{f}$) comunque vicina a x^* (lo zero di f): la perturbazione sulla soluzione è infatti

$$\sigma = |x^* - \tilde{x}^*| = 0.3333\ldots \times 10^{-2} \ .$$

In tal caso il problema del calcolo dello zero x^* dell'equazione non lineare $f(x) = 0$ può ritenersi **ben condizionato.** Osservando l'andamento della funzione f, in prossimità dello zero, si nota che essa decresce rapidamente, ovvero risulta $f'(x^*)$ "grande" in modulo. ♣

Consideriamo l' equazione non lineare

$$f(x) = 0 \ , \qquad x \in [a, b]$$

con f definita in $[a, b]$, continua e derivabile in un intorno $I \subset [a, b]$ dello zero. Osserviamo che in luogo dell'equazione $f(x) = 0$ si risolve l'equazione

$$\tilde{f}(x) = 0 \ , \qquad x \in [a, b]$$

dove

$$\tilde{f} = \tilde{f}(x) = f(x) + \Delta f(x)$$

indica una perturbazione della funzione f. Pertanto, in luogo dello zero x^* si determina $\tilde{x}^*$ quale zero di $\tilde{f}$. Indichiamo con

$$\sigma = |\tilde{x}^* - x^*|$$

l'errore assoluto sulla soluzione e con

$$\delta = |\Delta f(\tilde{x}^*)|$$

l'errore assoluto sul dato. Si vuole stimare σ in funzione di δ.

Teorema 1.9.1. *Il fattore di amplificazione dell'errore assoluto sul dato è:*

$$\mu_{abs}(f) = \frac{1}{|f'(x^*)|} \tag{1.57}$$

ovvero

$$\sigma = \mu_{abs}(f)\, \delta \ .$$

Dimostrazione Si consideri lo sviluppo in serie di Taylor della funzione f di punto iniziale x^* arrestato al primo ordine, nell'intorno I di x^*:

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + o(x - x^*) \ .$$

Supponiamo $f'(x^*) \neq 0$ (cioé x^* zero semplice) e valutiamo tale sviluppo in $\tilde{x}^*$:

$$f(\tilde{x}^*) = f(x^*) + f'(x^*)(\tilde{x}^* - x^*) + o(\tilde{x}^* - x^*) \ .$$

Trascurando il resto della serie $o(\tilde{x}^* - x^*)$, si ha:

$$\Delta f(\tilde{x}^*) = \tilde{f}(\tilde{x}^*) - f(\tilde{x}^*) = \tilde{f}(\tilde{x}^*) - f(x^*) - f'(x^*)(\tilde{x}^* - x^*)$$

da cui, poiché x^* è uno zero di f e $\tilde{x}^*$ è uno zero di $\tilde{f}$ ($f(x^*) = 0$, $\tilde{f}(\tilde{x}^*) = 0$), si ha:

$$\sigma = |\tilde{x}^* - x^*| = \frac{1}{|f'(x^*)|}|\Delta f(\tilde{x}^*)| = \frac{1}{|f'(x^*)|}\delta \ . \tag{1.58}$$

■

Definizione 1.9.1. *La quantità*

$$\mu_{abs}(f) = \frac{1}{|f'(x^*)|}$$

prende il nome di **indice di condizionamento assoluto dell'equazione non lineare** $f(x) = 0$.

Se $|f'(x^*)| \geq 1$ allora $\mu_{abs}(f) \leq 1$, pertanto funzioni "ripide" in prossimità dello zero x^* si possono ritenere ben condizionate. Viceversa funzioni con derivata piccola vicino a x^* sono mal condizionate.

Per quanto riguarda l'indice di condizionamento relativo, sussiste un teorema analogo al Teorema 1.9.1:

Teorema 1.9.2. *Il fattore di amplificazione dell'errore relativo sul dato è:*

$$\mu_{rel}(f) = \frac{|f(x^*)|}{|x^*\, f'(x^*)|} \tag{1.59}$$

ovvero

$$\sigma_{rel} = \mu_{rel}(f)\, \delta_{rel} \ .$$

Dimostrazione Dalla (1.58) si ha:

$$\frac{|\tilde{x}^* - x^*|}{|x^*|} = \frac{1}{|x^* f'(x^*)|}|\Delta f(\tilde{x}^*)| \tag{1.60}$$

Dividendo e moltiplicando per $|f(x^*)|$, segue che:

$$\frac{|\tilde{x}^* - x^*|}{|x^*|} = \frac{|f(x^*)|}{|x^* f'(x^*)|}\frac{|\Delta f(\tilde{x}^*)|}{|f(x^*)|} \ . \tag{1.61}$$

Posto

$$\sigma_{rel} = \frac{|\tilde{x}^* - x^*|}{|x^*|} \quad \text{e} \quad \delta_{rel} = \frac{|\Delta f(\tilde{x}^*)|}{|f(x^*)|}$$

rispettivamente l'errore relativo sulla soluzione e l'errore relativo sul dato, segue che:

$$\sigma_{rel} = \frac{|f(x^*)|}{|x^* f'(x^*)|}\delta_{rel}$$

■

Definizione 1.9.2. *La quantità*

$$\mu_{rel}(f) = \frac{|f(x^*)|}{|x^* f'(x^*)|}$$

è detta **indice di condizionamento relativo dell'equazione non lineare** $f(x) = 0$.

È bene notare che i risultati in (1.58) e (1.60) si sarebbero potuti anche ottenere osservando il legame tra il problema della ricerca di uno zero di f e il problema della valutazione della funzione inversa di f in 0.
Infatti se f è invertibile in un intorno I di x^* (questo accade certamente se f è derivabile in I e $f'(x^*) \neq 0$), il problema di determinare x^* tale che $f(x^*) = 0$ equivale al problema della valutazione di f^{-1} in 0. Pertanto l'indice di condizionamento assoluto della equazione non lineare $f(x) = 0$ coincide con l'indice di condizionamento assoluto della funzione f^{-1}. Quindi dal teorema di derivazione della funzione inversa risulta:

$$\mu_{abs}(f) = |[f^{-1}]'(0)| = \frac{1}{|[f'(x^*)]_{f(x^*)=0}|} \ .$$

Un risultato analogo si ha per l'indice di condizionamento relativo.

1.10 Criteri di arresto

I metodi numerici per la risoluzione di una equazione non lineare $f(x) = 0$ generano una successione $\{x_n\}$ di approssimazioni di uno zero. Un aspetto fondamentale legato all'utilizzo efficace di tali metodi è la scelta di un criterio opportuno in base al quale decidere quando arrestare il processo iterativo.
In generale, un criterio di arresto è ritenuto soddisfacente se il suo utilizzo conduce ad un risultato sufficientemente accurato, o meglio, consente di ottenere una accuratezza prefissata. In particolare, in questo contesto, le condizioni da utilizzare per stabilire quando terminare il processo iterativo, devono in generale:

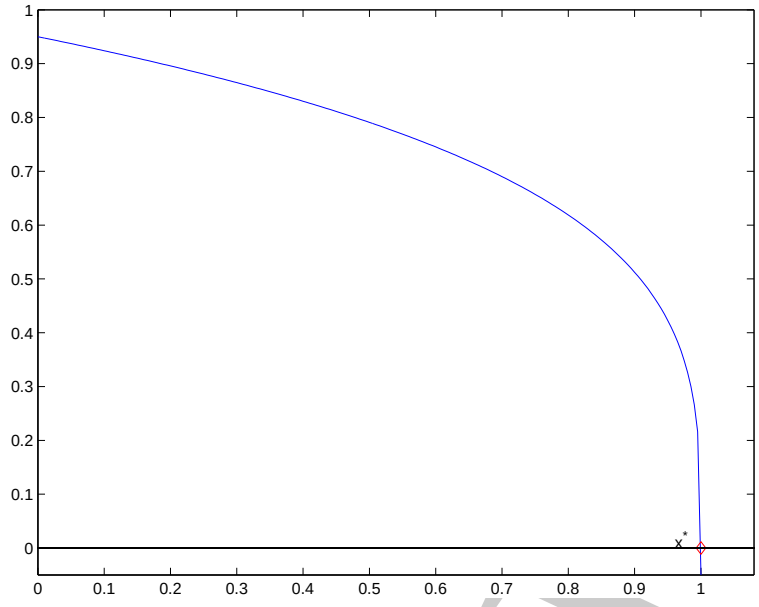

Figura 1.27: Grafico di $f(x) = \sqrt[4]{(1-x)} - 0.05$ in $[0,1]$. Con il simbolo '$\diamondsuit$' si indica lo zero della funzione.

1. verificare se il valore $f(x)$ è "sufficientemente" vicino a zero. A tal fine, si può utilizzare una condizione del tipo:

$$|f(x_k)| < \epsilon_f \tag{1.62}$$

con ϵ_f valore reale positivo prefissato. Quindi, si richiede di arrestare il procedimento quando il valore (assoluto) che la funzione assume nel valore x_k è minore di una quantità, detta **tolleranza**, la quale rappresenta l'accuratezza richiesta sulla soluzione dell'equazione.

♣ **Esempio 1.35.** Consideriamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = \sqrt[4]{(1-x)} - 0.05$$

Essa ha uno zero in $x = 0.99999375$. Arrestando il Metodo di Bisezione, applicato a $f(x)$ nell'intervallo $[0,1]$, dopo 7 iterazioni si ha:

$$x_7 = 0.9922, \quad f(x_7) = 0.2473$$

ovvero l'errore relativo su x:

$$E_r = \frac{0.9999 - 0.9922}{|0.9999|} = 0.76 \times 10^{-2}$$

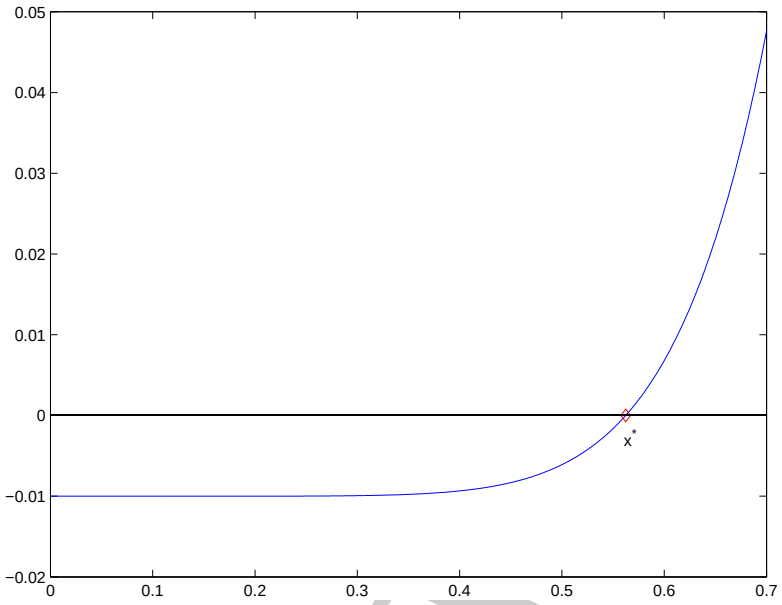

Figura 1.28: Grafico di $f(x) = x^8 - 0.01$ in $[0, 0.7]$. Con il simbolo '♢' si indica lo zero della funzione.

Dunque l'errore che si commette sull'approssimazione della radice è di 0.7% mentre:

$$E_a = |0 - f(x_7)| = 0.2473$$

ovvero l'errore che si commette sull'approssimazione di $f(0.99999375) = 0$ è di circa il 24%. ♣

2. verificare se x_k è "sufficientemente" vicino allo zero.

♣ **Esempio 1.36.** Consideriamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = x^8 - 0.01 = 0$$

Essa ha uno zero in $x^* = 0.562341...$ Arrestando al passo k il Metodo di Bisezione, applicato a $f(x)$ nell'intervallo $[0, 1]$, se:

$$|f(x_k)| < 0.01$$

dopo un'iterazione abbiamo:

$$x_1 = 0.5$$

e

$$E_a = |0 - f(x_1)| = 0.0061 < 0.01;$$

Considerando l'errore relativo su x^* si ottiene:

$$E_r = \frac{|x_1 - x^*|}{|x^*|} = \frac{0.562341 - 0.5}{|0.562341|} = 0.119$$

Dunque l'errore che si commette sull'approssimazione della radice è di circa il 12%. ♣

In generale, una condizione per l'arresto del metodo iterativo è richiedere che l'errore assoluto, $|x_k - x^*|$, sia minore di una quantità ϵ_x:

$$|x_{k+1} - x^*| \leq \epsilon_x. \tag{1.63}$$

Tuttavia, l'utilizzo della (1.63) come criterio di arresto presuppone la conoscenza della soluzione dell'equazione, che ovviamente non è nota.
Nel caso del Metodo di Bisezione, si è già osservato che una stima dell'errore è fornita dalla semiampiezza dell'intervallo corrente. Quindi, un criterio di arresto "effettivamente utilizzabile", perché basato su quantità calcolabili dall'algoritmo, è il seguente:

$$(b_k - a_k)/2 \leq \epsilon_x. \tag{1.64}$$

Se, in particolare, si desidera stimare l'errore relativo di troncamento analitico, e, quindi, controllare l'accuratezza della approssimazione in termini di cifre significative, basta osservare che dalla (1.15) segue che:

$$\frac{|x_{k+1} - x^*|}{|x^*|} \leq \frac{b_k - a_k}{2|a_k|},$$

ottenendo così il criterio di arresto basato sulla distanza relativa tra gli estremi dell'intervallo corrente:

$$\frac{b_k - a_k}{2|a_k|} \leq \epsilon_x. \tag{1.65}$$

♣ **Esempio 1.37.** Riprendiamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = x^8 - 0.01$$

La funzione ha uno zero in $x^* = 0.562341\ldots$. Assegniamo, come condizione di arresto, il verificarsi di una delle eventualità descritte nei punti 1. e 2., ovvero che al generico passo k:

1. $|f(x_k)| < \epsilon_f = 10^{-3}$;
2. $\frac{b_k - a_k}{2|a_k|} \leq \epsilon_x = 10^{-4}$.

k	a_k	b_k	x_k	$f(x_k)$
1	0	1	0.5	-0.60937
2	0.5	1	0.75	0.9×10^{-1}
3	0.5	0.75	0.625	0.0132
4	0.5	0.625	0.5625	0.259×10^{-4}
5	0.5	0.5625	0.5312	-0.366×10^{-2}
...	...	...	...	...
8	0.5546	0.5625	0.5546	-0.52×10^{-3}
9	0.5585	0.5625	0.5585	-0.25×10^{-3}
10	**0.56054**	**0.5625**	**0.560546**	-0.11×10^{-3}

Tabella 1.20: Valori numerici di x_k e $f(x_k)$ relativi all'applicazione del Metodo di Bisezione alla funzione il cui grafico è riportato in Fig. 1.28 (10 iterazioni). x_k indica il punto medio del sottointervallo $[a_k, b_k]$, $k = 1, \ldots, 10$, di $[0, 1]$, determinato al passo k.

Dai valori della Tabella 1.20, si osserva che, dopo 10 iterazioni del Metodo di Bisezione, si ha:

1. $|f(x_{10})| = |f(0.560546)| \approx 2.52 \times 10^{-4} < \epsilon_f = 10^{-3}$;
2. $\frac{b_{10}-a_{10}}{2|a_{10}|} = \frac{0.5625-0.56054}{2(0.56054)} \approx 10^{-3} > \epsilon_x = 10^{-4}$.

Il Metodo di Bisezione si arresta, in questo caso, per il verificarsi della condizione 1. ♣

♣ **Esempio 1.38.** Riprendiamo l'equazione:

$$f(x) = 0, \qquad \text{con} \qquad f(x) = \sqrt[4]{(1-x)} - 0.05$$

La funzione ha uno zero in $x^* = 0.9999....$ Assegniamo, come condizione di arresto, il verificarsi di una delle eventualità descritte nei punti 1. e 2., ovvero che al generico passo k:

1. $|f(x_k)| < \epsilon_f = 10^{-3}$;
2. $\frac{b_k-a_k}{2|a_k|} \leq \epsilon_x = 10^{-3}$.

Dai valori della Tabella 1.21, si osserva che dopo 10 iterazioni del Metodo di Bisezione, si ha:

1. $|f(x_{10})| = |f(0.9902)| \approx 1.26 \times 10^{-1} > \epsilon_f = 10^{-3}$;
2. $\frac{b_{10}-a_{10}}{2|a_{10}|} = \frac{1-0.99902}{2(0.9902)} \approx 4.88 \times 10^{-4} < \epsilon_x = 10^{-3}$.

Il Metodo di Bisezione si arresta, in questo caso, per il verificarsi della condizione 2. ♣

k	a_k	b_k	x_k	$f(x_k)$
1	0	1	0.5	7.9×10^{-1}
2	0.5	1	0.75	6.5×10^{-1}
3	0.75	1	0.875	5.4×10^{-1}
4	0.875	1	0.937	4.5×10^{-1}
5	0.937	1	0.968	3.7×10^{-1}
...	...	...	...	...
8	0.998	1	0.992	2.0×10^{-1}
9	0.992	1	0.996	1.6×10^{-1}
10	**0.9902**	**1**	**0.9902**	1.26×10^{-1}

Tabella 1.21: Valori numerici di x_k e $f(x_k)$ relativi all'applicazione del Metodo di Bisezione alla funzione il cui grafico è riportato in Fig. 1.27 (10 iterazioni). x_k indica il punto medio del sottointervallo $[a_k, b_k]$, $k = 1, \ldots, 10$, di $[0, 1]$, determinato al passo k.

Per il Metodo di Newton, invece, è possibile stimare la quantità $|x_n - x^*|$ a partire da considerazioni di tipo analitico. Si osservi, infatti, che, per il Teorema di Lagrange, si ha:

$$x^* - x_n = -\frac{f(x_n)}{f'(\xi_n)} \quad \xi_n \in int[x_n, x^*].$$

Poiché $f'(\xi_n) = f'(x_n) + \sigma$, con $\sigma = \mathcal{O}(x_n - \xi_n)$ (se la successione è convergente, x_n è in un intorno sufficientemente piccolo della soluzione), si ha:

$$(x^* - x_n)\,(f'(x_n) + \sigma) = -f(x_n)$$

e, dividendo ambo i membri per $f'(x_n)$:

$$\frac{(f'(x_n) + \sigma)(x^* - x_n)}{f'(x_n)} = \frac{-f(x_n)}{f'(x_n)} \tag{1.66}$$

Inoltre, per la (1.22):

$$\frac{-f(x_n)}{f'(x_n)} = x_{n+1} - x_n,$$

per cui, dalla (1.66), segue che:

$$(x^* - x_n)\frac{f'(x_n) + \sigma}{f'(x_n)} = x_{n+1} - x_n$$

e, quindi,

$$x^* - x_n = \frac{f'(x_n)}{f'(x_n) + \sigma}(x_{n+1} - x_n)$$

che si può scrivere come:

$$x^* - x_n = \frac{1}{1 + \frac{\sigma}{f'(x_n)}}(x_{n+1} - x_n)$$

da cui, passando ai valori assoluti:

$$|x^* - x_n| = \left| \frac{1}{1 + \frac{\sigma}{f'(x_n)}} \right| \cdot |x_{n+1} - x_n| \leq \frac{1}{\left| 1 - \left| \frac{\sigma}{f'(x_n)} \right| \right|} \cdot |x_{n+1} - x_n|$$

Posto

$$\theta = |\sigma| \,,$$

per la (1.57) si ha, infine,

$$|x^* - x_n| \leq \frac{1}{|1 - \theta \cdot \mu_{abs}|} \cdot |x_{n+1} - x_n| \tag{1.67}$$

dalla quale si può concludere che, quanto più piccolo è μ_{abs}, tanto più la quantità $|x_{n+1} - x_n|$ fornisce una stima calcolabile sufficientemente accurata dell'errore commesso al passo n, $|x^* - x_n|$; in effetti, se $\mu_{abs} \to 0$, allora

$$\frac{1}{|1 - \theta \cdot \mu_{abs}|} \to 1 \quad \Rightarrow \quad |x^* - x_n| = |x_{n+1} - x_n|$$

♣ **Esempio 1.39.** Applichiamo la stima calcolabile dell'errore per il Metodo di Newton alla risoluzione dell'equazione:

$$x^2 - 10^{-6} = 0$$

la Tabella 1.22 riporta il confronto tra l'errore e la stima dell'errore stesso. ♣

Infine, le condizioni da utilizzare per stabilire quando terminare il processo iterativo, devono:

3. considerare un controllo sul numero, k, di iterazioni eseguite; ciò può essere fatto ponendo un limite massimo di iterazioni, $itmax$:

$$k < ITMAX \tag{1.68}$$

n	$f(x_n)$	$\lvert x_n - x^* \rvert$	$\lvert x_n - x_{n+1} \rvert$
1	.250D+00	.499D+00	.500D+00
2	.625D-01	.249D+00	.250D+00
3	.156D-01	.124D+00	.125D+00
4	.391D-02	.615D-01	.625D-01
5	.976D-03	.303D-01	.312D-01
6	.244D-03	.146D-01	.156D-01
7	.607D-04	.686D-02	.779D-02
8	.149D-04	.299D-02	.386D-02
9	.350D-05	.112D-02	.187D-02
10	.680D-06	.296D-03	.825D-03

Tabella 1.22: Valori numerici relativi all'applicazione del Metodo di Newton alla risoluzione dell'equazione $x^2 - 10^{-6} = 0$ con $x_0 = 1$. ($x^* = 0.001$) (10 iterazioni).

È necessaria una corretta interpretazione e gestione delle condizioni di arresto. Ad esempio il verificarsi della condizione (1.68) può essere sintomo sia di non convergenza sia di convergenza lenta; discernere fra i due casi è di importanza fondamentale, ma non facile.
Si osserva, infine, che le condizioni di arresto ai punti 1. e 2. potrebbero essere, in generale, non soddisfacenti, per ogni metodo iterativo. Una condizione di arresto valida deve infatti consentire di gestire eventualità differenti:

(α) *Raggiungimento della soluzione*: Nella iterata corrente, x_n, il valore della funzione è piccolo. Ciò, ad esempio, può essere verificato con una condizione del tipo

$$|f(x_n)| < \epsilon_f \tag{1.69}$$

con tolleranza prefissata, ϵ_f. Si noti, però, che una condizione del tipo (1.69) potrebbe essere fuorviante essendo dipendente dall'ordine di grandezza della funzione.

Una condizione più soddisfacente della (1.69) è la seguente:

$$|f(x_n)| < \epsilon_f \|f\| \tag{1.70}$$

dove $\|\cdot\|$ è una qualsiasi norma; oppure una condizione del tipo

$$|f(x_n)| < \epsilon_f |f(x_0)| \tag{1.71}$$

Il vantaggio di (1.71) rispetto a (1.70) è quello di non richiedere il calcolo di alcuna norma; tuttavia, (1.71) può essere non praticabile in alcuni casi (ad es. quando

x_0 è già una buona stima della soluzione, per cui $f(x_0)$ è sufficientemente piccolo da ridurre l'ordine di grandezza di ϵ_f, alterando quindi, in maniera significativa, la tolleranza richiesta su $f(x_n)$.).

(β) *La successione x_n si è numericamente arrestata*: ovvero si verifica una condizione del tipo:

$$|x_n - x_{n-1}| < \epsilon_x \tag{1.72}$$

con ϵ_x tolleranza prefissata. Il criterio (1.72), essendo assoluto (indipendente cioè dall'ordine di grandezza delle quantità in gioco), è in genere sconsigliabile. Più affidabile è, ad esempio, il seguente:

$$|x_n - x_{n-1}| < \epsilon_x \ |x_{n-1}| \tag{1.73}$$

Di seguito, riportiamo l'algoritmo di Bisezione con il criterio di arresto descritto e l'algoritmo di Newton. Per quest'ultimo si osservi come si sia prevista l'eventualità che si verifichi $f'(x_n) = 0$.

procedure fxbis(**input:** a,b,f,ϵ_f,$itmax$, **out:** c,fc)

/# SCOPO: calcolo di un'approssimazione dello zero

/# di una funzione mediante il Metodo di Bisezione

/# SPECIFICHE PARAMETRI:

var: a : **reale** {*primo estremo dell'intervallo di ricerca*}

var: b : **reale** {*secondo estremo dell'intervallo di ricerca*}

var: f : **funzione esterna** {*funzione di cui si cerca lo zero*}

var: ϵ_f : **reale** {*accuratezza richiesta*}

var: $itmax$: **intero** {*numero massimo di iterazioni*}

var: c : **reale** {*approssimazione dello zero*}

var: fc : **reale** {*valore di f in c*}

/# INIZIO ISTRUZIONI:

```
    fa := f(a);
    fb := f(b);
    k := 0;
    repeat
/# calcolo del punto medio e del valore della funzione
      c := a + (b − a)/2;
      fc := f(c);
       if (fc ∗ fa ≤ 0) then     {test sul segno di f}
        b := c;
        fb := fc;
       else
```

Procedura 1.5: Algoritmo del Metodo di Bisezione (seconda versione) - continua

$a := c$;
$fa := fc$;
endif
$k := k + 1$;
until$(((b-a)/2 \leq \epsilon_{mac}|a|)$ or $(|fc| < \epsilon_f))$ or $k > itmax)$ {*criterio di arresto*}
end fxbis

Procedura 1.5: Algoritmo del Metodo di Bisezione (seconda versione) - fine

procedure newton(**input:** $x0$,f,f',$ftol$,$xtol$,$itmax$, **out:** $xnew$,$fxnew$)

/# SCOPO: calcolo di un'approssimazione dello zero

/# di una funzione mediante il Metodo di Newton

/# SPECIFICHE PARAMETRI:

var: $x0$	: **reale**	{*approssimazione iniziale dello zero*}
var: f	: **funzione esterna**	{*funzione di cui si cerca lo zero*}
var: f'	: **funzione esterna**	{*derivata di f*}
var: $ftol$	: **reale**	{*accuratezza valutazione*}
var: $xtol$	: **reale**	{*accuratezza soluzione*}
var: $itmax$	: **intero**	{*numero massimo di iterazioni*}
var: $xnew$	: **reale**	{*approssimazione dello zero*}
var: $fxnew$	: **reale**	{*valore di f in xnew*}

Procedura 1.6: Algoritmo del Metodo di Newton (seconda versione) - continua

```
/# INIZIO ISTRUZIONI:
    k := 0
    xold := x0
    fxnew := f(xold)
/# generazione della successione delle approssimazioni
    repeat
    fold := f'(xold)
        if fold = 0 then
          rapx := 0
        else
          rapx := fxnew/fold
        endif

        xnew := xold - rapx
        fxnew := f(xnew)
        err := |rapx|
        relerr := err/|xnew|
        xold := xnew
        k := k + 1
    until ( (|fxnew| ≥ ftol)  and  (relerr ≥ xtol)  and  (k < itmax) )
end newton
```

Procedura 1.6: Algoritmo del Metodo di Newton (seconda versione) - fine

```
procedure DB(input: x0,x1,f,itmax,xtol,ftol out: xdb,fxdb)
/# SCOPO: calcolo di un'approssimazione dello zero
/#          di una funzione mediante il Metodo di Dekker-Brent
/# SPECIFICHE PARAMETRI:
 var: x0     : reale            {estremo sinistro dell'intervallo di ricerca}
 var: x1     : reale            {estremo destro dell'intervallo di ricerca}
 var: f      : funzione esterna {funzione di cui si cerca lo zero}
 var: itmax  : intero           {numero massimo di iterazioni}
 var: ftol   : reale            {accuratezza valutazione}
 var: xtol   : reale            {accuratezza soluzione}
 var: xdb    : reale            {approssimazione dello zero}
 var: fxdb   : reale            {valore di f in xdb}
/# INIZIO ISTRUZIONI:
   y1 := x0
   y0 := y1 + 1
   y-1 := y0
   f0 := f(x0)
   f1 := f(x1)
   k := 1
   while ( ((|x1 - y1| > xtol|x1| or |f1| > ftol) and k < itmax)   {criterio
      if y1 ≠ y-1 then                                              d'arresto}
         d := f1(x1 - x0)/(f1 - f0)
         if (d(x1 - y1) < 0 or |d| > |x1 - y1|) then              {criterio di scelta
            d := (x1 - y1)/2                                        tra Metodo di
         endif                                                      Bisezione e
      else                                                          delle Secanti}
```

Procedura 1.7: Algoritmo del Metodo di Dekker-Brent - continua

$d := (x_1 - y_1)/2$
endif
$x_0 := x_1$; $f_0 := f_1$; $x_1 := x_1 - d$
$f_1 := f(x_1)$; $y_{-1} := y_0$; $y_0 := y_1$
if $f_0 f_1 < 0$ **then**
$y_1 := x_0$
endif
$k := k + 1$
endwhile
$xdb := x_1$;
$fxdb := f_1$;
end DB

Procedura 1.7: Algoritmo del Metodo di Dekker-Brent - fine

Nell'algoritmo del Metodo di Dekker-Brent, in relazione al *criterio di scelta tra Metodo di Bisezione e delle Secanti*, si osserva che, se si verifica la condizione $d(x_1 - y_1) < 0$, ovvero se $d = x_1 - x_s$ e $x_1 - y_1$ hanno segno discorde, l'algoritmo procede applicando il metodo di bisezione; infatti, se

$$d < 0 \quad \text{e} \quad x_1 - y_1 > 0$$

il punto x_s cade all'esterno dell'intervallo $[y_1, x_1]$; se

$$d > 0 \quad \text{e} \quad x_1 - y_1 < 0$$

x_s cade all'esterno dell'intervallo $[x_1, y_1]$. Analogamente, l'algoritmo procede applicando il metodo di bisezione anche se è soddisfatta la condizione $|d| > |x_1 - y_1|$; in tal caso, l'ampiezza dell'intervallo corrente è maggiore di quella dell'intervallo esaminato al passo precedente, ovvero la successione generata dal metodo delle secanti non sta convergendo.

1.11 Software matematico disponibile per la risoluzione numerica di equazioni non lineari

Si vuole descrivere il principale software matematico disponibile per la risoluzione numerica di equazioni non lineari del tipo $F(x) = 0$. Per tale problema, sono disponibili

in rete, nei principali package di software matematico, un gran numero di routine e funzioni. Tipicamente tali routine prevedono come parametri di input:

1. F, la funzione reale (o complessa) fornita dall'utente;
2. A, B, estremi dell'intervallo di ricerca, o in alternativa;
3. $X0$, approssimazione iniziale della radice o array contenente m approssimazioni iniziali di m radici della funzione;
4. $EPSABS$, $EPSREL$ tolleranze assoluta e relativa utilizzate nel criterio di arresto;
5. $MAXFN$, massimo numero di valutazioni di funzione;

e come parametri di output:

6. x, la stima della radice x^*
7. $IFAIL$, indicatore di errore che avverte sia se si è in presenza di errori di input, sia se si è ottenuta o meno convergenza.

Al fine di fornire una presentazione sintetica e sufficientemente esaustiva del vasto software disponibile, si fa riferimento alla classificazione introdotta da J.Rice in *Numerical methods, Software and Analysis*, Academic Press,1993 , e già descritta nel **§3.7**, **Capitolo 3**, **Parte prima**. Oltre a tale classificazione del software, adottiamo qui un'ulteriore classificazione riguardante il problema, proposta dal `GAMS - Guide to Available Mathematical Software`. Il `GAMS` rappresenta un deposito virtuale di tutte le componenti di software matematico e statistico di uso nel calcolo tecnico-scientifico. In tale classificazione il problema della ricerca degli zeri di una funzione è indicato con la lettera `F` e per tale classe sono individuati tre sottoproblemi:

- Classe `F1` - funzioni e routines per la ricerca delle soluzioni di un'equazione non lineare scalare: si distinguono il caso della ricerca delle radici di polinomi, classe `F1a`, e di zeri di funzioni trascendenti, classe `F1b`.
- Classe `F2` - funzioni e routines per la ricerca delle soluzioni di un sistema di equazioni non lineari $F(x) = 0$, ovvero gli zeri di funzioni vettoriali $F : \Re^n \to \Re^n$.
- Classe `F3` - funzioni e routines ausiliarie per i precedenti problemi: si tratta principalmente di routine che controllano la consistenza dei valori assunti dalla funzione F con le funzioni fornite dall'utente per il calcolo delle derivate parziali.

Ci si limita in questo contesto ad elencare il software disponibile per il problema della risoluzione di equazioni non lineari scalari, cioè la classe `F1`.
Tra i principali package e librerie di software matematico si menzionano il package di routine specifico **MINPACK** [8], le librerie *general-purpouse* **NAg** [10], **IMSL** [12] e **NAPACK** [9], e le collezioni di routine **SLATEC** [5] e **ACM-TOMS** [11].

La libreria **MINPACK**, sviluppata da Jorge Morè, Burt Garbow e Ken Hillstrom presso l'Argonne National Laboratory, rappresenta uno dei principali software per la risoluzione di sistemi non lineari e problemi di minimi quadrati non lineari. La libreria è costituita da 11 routines scritte in `FORTRAN` in doppia precisione, di cui 6 per la classe `F1`.

La libreria **NAg** (Mark 17 del 1996) del Numerical Algorithm Group di Oxford è una delle fonti più ampie di software per la ricerca di radici di funzioni non lineari. A tale problema nella libreria sono dedicati due capitoli: il capitolo `C02` per la ricerca degli radici di polinomi (corrispondente alla classe `F1a`) per cui sono disponibili 8 routine, e il capitolo `C05` per la ricerca di zeri di equazioni trascendenti (corrispondente alle classi `F1b`, `F2` e `F3`) per cui sono disponibili 6 routine per il caso scalare. Tali routine scritte in `FORTRAN` e in doppia precisione, sono basate principalmente sul metodo iterativo di Laguerre (Wilkinson 1965) per la ricerca delle radici di un polinomio (`F1a`), oppure sui metodi di bisezione, interpolazione lineare e estrapolazione (algoritmo di Bus e Dekker) (routine `C05ADF`) o su approcci che garantiscono convergenza globale come i metodi di omotopia e continuazione (routine `C05AJF` e `C05AXF`), per gli zeri di un'equazione non lineare.

La libreria **IMSL** della Visual Numerics Inc. comprende circa 700 routine e funzioni scritte in `FORTRAN`, in singola e doppia precisione, per risolvere problemi di matematica applicata, tra cui 6 routine per la classe `F1b`. Tra queste ricordiamo in particolare per `F1b` la routine `ZBREN` basata sul metodo di bisezione e su tecniche di interpolazione inversa della funzione, e le routine `ZREAL` e `ZANLY` (per funzioni analitiche) che ricercano la radice a partire da approssimazioni iniziali e utilizzano un raffinamento del metodo delle secanti noto come metodo di M*ü*ller.

Il package **NAPACK**, sviluppato da Bill Hager (del Dipartimento di Matematica, Università della Florida), è costituito da circa 150 subroutine trasportabili scritte in `FORTRAN` per problemi di algebra lineare ed ottimizzazione. Include in particolare una routine `ROOT` , in singola precisione, per la classe `F1b` che utilizza un metodo ibrido basato su interpolazione quadratica, metodo delle secanti e metodo di bisezione.

Il package **SLATEC** è una collezione molto vasta di subroutine (circa 800) scritte in `FORTRAN`, di pubblico dominio, per la risoluzione di una larga varietà di problemi della matematica. In **SLATEC** sono presenti 1 routine per il problema `F1b` (routine `FZERO` basata sul metodo di Dekker-Brent) e 3 routine per il problema `F2`.

La collezione **ACM-TOMS** (Collected Algorithms of the Association for Computing Machinery), costituita da piú di 650 elementi di software matematico, è distribuita dall'ACM Algorithms Distribution Service. In tale pacchetto sono presenti 3 moduli per la classe `F1`: il modulo **631**, routine `ZERO1` e i moduli **748** e **825**, basati su interpolazione razionale, interpolazione cubica inversa e metodi di punto fisso.

Si ricordano infine, tra le molte altre funzioni per i problemi della classe `F1b`,

- la routine `ZEROIN`, basata sul metodo di Dekker-Brent, del package **CMLIB**

(Core Math LIBrary) del National Institute of Standards and Technology (NIST), collezione di circa 750 subroutines e funzioni scritte in `FORTRAN`, che risolvono problemi standard in molte aree della matematica e della statistica;

- la funzione `FZERO` dell'ambiente di risoluzione di problemi `MATLAB` basata sul metodo di Dekker-Brent e sull'interpolazione quadratica inversa.

1.12 `MATLAB` e la risoluzione numerica di equazioni non lineari

`MATLAB` mette a disposizione, per la ricerca degli zeri di una funzione non lineare, la routine `fzero`, reperibile nella directory `funfun`. `fzero` implementa un algoritmo per la ricerca degli zeri di una funzione *continua, non lineare, in una sola variabile*.
Tale algoritmo si deve a T. Dekker e si basa sulla combinazione dei *metodi di bisezione e secanti*, in analogia al metodo di Dekker e Brent, con *tecniche di interpolazione quadratica inversa* per approssimare la funzione inversa di quella di cui si cerca lo zero. Una versione in *Algol 60*, si trova in [1]. Una versione *Fortran*, sulla quale si basa la function `fzero.m`, si trova, invece, in [6].

Dal *prompt* dei comandi è possibile richiamare la funzione con le relative opzioni:

```
>> x = fzero(fun,x0)
>> x = fzero(fun,x0,options)
>> [x,fval] = fzero(...)
>> [x,fval,exitflag] = fzero(...)
>> [x,fval,exitflag,output] = fzero(...)
```

In particolare, con

```
>> x = fzero(fun,x0)
```

dove `fun` è una funzione definita opportunamente[15] e `x0` uno scalare, si cerca uno zero[16] di una funzione, `fun`, che appartenga ad un intorno dell'estremo `x0`. La procedura individua, dapprima, un intervallo contenente `x0`, in cui la funzione cambia segno agli estremi, quindi, in tale intervallo, cerca uno zero della funzione. Il valore di output sarà un'approssimazione dello zero (se la funzione è continua) oppure sarà `NaN` se la ricerca fallisce, ovvero se, durante l'esecuzione, non si individua un intervallo, contenente `x0`, in cui la funzione cambia segno.
Se `x0` è un vettore di lunghezza due, `fzero` ricerca lo zero nell'intervallo di estremi le due componenti di `x0`, assumendo che, in tali estremi, la funzione cambi segno. In effetti

[15] Per la definizione della funzione si suggerisce di fare riferimento a `inline` o `function_handle` (@).
[16] Si osserva, in particolare, che per *zeri* si intendono ascisse di zeri semplici (molteplicità 1).

si riscontra un errore se questa ipotesi non è verificata. Anche quando `x0` è un vettore, il valore di output sarà vicino ad un punto in cui la funzione cambia segno.

♣ **Esempio 1.40.** Un'approssimazione di π si può realizzare attraverso la ricerca dello zero della funzione seno, in prossimità di 3:

```
>> x = fzero(@sin,3)
x =
    3.1416
```

♣

♣ **Esempio 1.41.** Trovare lo zero della funzione coseno compreso tra 1 e 2:

```
>> x = fzero(@cos,[1 2])
x =
    1.5708
```

Si osserva che $cos(1)$ e $cos(2)$ hanno segni opposti. ♣

Consideriamo:

```
>> fzero(@tan,1)
```

che fornisce, in output, il valore 1.5708, approssimazione numerica di un punto di discontinuità per la funzione `tan`. In tal caso la discontinuità della tangente in $\pi/2$ è interpretata, numericamente, dalla `fzero`, come un cambiamento di segno.
`fzero` restituisce, inoltre, `NaN` se, come nell'esempio seguente, la funzione non ammette alcuno zero appartenente all'asse reale.

♣ **Esempio 1.42.** Trovare lo zero della funzione $|x| + 1$ vicino a 1:

```
>>f=inline('abs(x)+1');
>>X = fzero(f, 1)
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -Inf is Inf.)
Check function or try again with a different starting value.

X =

   NaN
```

♣

È possibile aggiungere, in input, l'argomento `options`, creato attraverso la funzione `OPTIMSET` di `MATLAB`

```
>> x = fzero(fun,x0,options)
```

la funzione esegue la ricerca secondo i parametri specificati nella struttura `options`. Si supponga, ad esempio, di utilizzare l'opzione `Display`. Definendo quest'ultima come `'off'` non si visualizza alcun output sul display; al contrario si può scegliere di visualizzare l'output ad ogni iterazione, con `'iter'`, oppure solo il risultato finale, con `'final'`. Per default l'opzione `Display` è definita come `'notify'`, in modo da visualizzare l'output solo se la ricerca dello zero fallisce.

♣ **Esempio 1.43.** Con le opzioni:

```
>>  X = fzero(@sin,3,optimset('disp','iter'))

Search for an interval around 3 containing a sign change:
 Func-count   left       f(left)          right     f(right)     Procedure
    1             3       0.14112              3       0.14112   initial interval
    3       2.91515      0.224515        3.08485     0.0567094   search
    5          2.88      0.258619          3.12      0.021591   search
    7       2.83029      0.306295        3.16971    -0.0281093   search

Search for a zero in the interval [2.8303, 3.1697]:
 Func-count      x           f(x)         Procedure
    7         3.16971    -0.0281093        initial
    8         3.14118   0.000417192        interpolation
    9         3.14159 -5.41432e-008        interpolation
   10         3.14159  1.45473e-015        interpolation
   11         3.14159  1.22465e-016        interpolation
   12         3.14159  1.22465e-016        interpolation

Zero found in the interval [2.83029, 3.16971]

X =

    3.1416
```

l'esecuzione fornisce, in output, un'approssimazione di π, usa la tolleranza di default e visualizza informazioni relative a ciascuna delle iterazioni. ♣

Digitando:

```
>> [x,fval] = fzero(...)
```

la routine fornisce, in `fval`, la valutazione della funzione `fun` in corrispondenza dell'approssimazione calcolata, `fval=fun(x)`. Ulteriormente,

```
>> [x,fval,exitflag] = fzero(...)
```

la variabile `exitflag` fornisce informazioni sulla convergenza dell'algoritmo o, in caso contrario, descrive le condizioni che hanno determinato l'arresto dell'esecuzione.
Infine, con:

```
>> [x,fval,exitflag,output] = fzero(...)
```

sono memorizzate, nella struttura `output`, informazioni relative all'esecuzione dell'algoritmo, come il numero di valutazioni di funzione, il numero di iterazioni necessarie per individuare un intervallo, il numero di iterazioni necessarie per trovare uno zero ed, infine, un messaggio relativo alla fine dell'esecuzione.

♣ **Esempio 1.44.** Per individuare uno zero della funzione

$$f(x) = x^3 - 2x - 5$$

scrivere una funzione come:

```
>> f=inline('x.^3-2*x-5');
```

dunque cercare lo zero in un intorno di 2:

```
>> z = fzero(f,2)
z =
    2.0946
```

Poiché la funzione f è un polinomio, si può utilizzare, in alternativa, la funzione

```
>>  roots([1 0 -2 -5])
```

che individua lo stesso zero *reale* ed un coppia di zeri, complessi coniugati [17].

```
    2.0946
   -1.0473 + 1.1359i
   -1.0473 - 1.1359i
```

♣

[17] La funzione `roots` implementa un algoritmo basato sul calcolo degli autovalori di una *matrice associata al polinomio* (*companion matrix*), costruita a partire dal vettore dei suoi coefficienti; se, ad esempio, quest'ultimo è ordinato decondo le potenze decrescenti, tale matrice sarà quadrata, di dimensione uguale al grado del polinomio, avente tutti elementi uguali ad uno sotto la diagonale, mentre gli elementi della prima riga saranno gli opposti dei rapporti tra tutti i coefficienti del polinomio, a meno del coefficiente del termine di grado massimo, e quest'ultimo. Gli autovalori di tale matrice sono le radici del polinomio.

1.13 Esercizi

1.13.1 Esercizi numerici

Esercizio 1 Denotiamo con $[a_n, b_n]$ la successione di intervalli che deriva dal metodo di bisezione applicato ad una funzione continua f. Sia inoltre

$$[a_0, b_0] \equiv [a, b] \ , \quad x_{n+1} = \frac{a_n + b_n}{2} \ , \quad r = \lim_{n\to\infty} x_n \quad e \quad e_n = r - x_n \ .$$

Verificare che

(a) $|e_n| \leq 2^{-n}(b_0 - a_0)$;

(b) $e_n = \mathcal{O}(2^{-n})$ per $n \to \infty$;

(c) $|x_n - x_{n+1}| = 2^{-n-1}(b_0 - a_0)$;

(d) per ogni n, m, $a_m \leq b_n$;

(e) r è l'unico elemento dell'insieme $\bigcup_{n=0}^{\infty}[a_n, b_n]$;

(f) per ogni n, $[a_{n+1}, b_{n+1}] \subset [a_n, b_n]$.

Esercizio 2 Consideriamo il metodo di bisezione applicato all'intervallo $[1.5, 3.5]$.

(a) Qual è l'ampiezza dell'intervallo dopo n passi?

(b) Qual è la massima distanza possibile tra la radice r ed il punto medio di tale intervallo?

Esercizio 3 Utilizzando le notazioni introdotte nell'esercizio 1, trovare una formula per il numero di passi n del metodo di bisezione necessari per garantire

$$|r - x_n| < \epsilon$$

che coinvolga a_0, b_0 e ϵ.

Esercizio 4 Applicare il metodo di bisezione per trovare una radice positiva dell'equazione

$$x^2 - 4xsin(x) + (2sin(x))^2 = 0$$

con due cifre significative corrette. Utilizzare un sistema aritmetico a precisione finita, caratterizzato da base $\beta = 10$ e precisione $t = 11$.

Esercizio 5 Utilizzando le notazioni introdotte nell'esercizio 1, determinare una formula che coinvolga a_0, b_0, ϵ che fornisca il numero di passi necessario per ottenere un'approssimazione della radice con un errore relativo minore o uguale a ϵ. Assumere che $a_0 > 0$.

Esercizio 6 Se utilizziamo il metodo di bisezione sull'intervallo $[128, 129]$, utilizzando un un sistema aritmetico a precisione finita caratterizzato da un epsilon macchina dell'ordine di 10^{-6}, possiamo calcolare la radice con un errore assoluto minore di 10^{-6}?
Rispondere alla stessa domanda, riferendosi all'errore relativo.

Esercizio 7 Provare che il punto x_{n+1} calcolato nel metodo di bisezione, supponendo $f(a_n)f(b_n) < 0$ può essere pensato come il punto di intersezione della retta per i punti $(a_n, sign(f(a_n))$, $(b_n, sign(f(b_n))$, con l'asse delle ascisse.

Esercizio 8 Per calcolare il reciproco di un numero R senza effettuare la divisione, si può cercare la radice $r = 1/R$ di $f(x) = x^{-1} - R$. Scrivere un algoritmo per trovare $1/R$ utilizzando il metodo di Newton, evitando l'uso di divisioni ed esponenziali.

Esercizio 9 Se applichiamo il metodo di Newton a $f(x) = x^2 - 1$ con $x_0 = 10^{10}$, quanti passi sono necessari per ottenere una radice con un'accuratezza di 10^{-8}?

Esercizio 10 Supponiamo che r sia uno zero di molteplicità doppia di f. Allora, $f(r) = f'(r) = 0 \neq f''(r)$. Mostrare che se f'' è continua, allora il metodo di Newton avrà $e_{n+1} \approx 1/2 \cdot e_n$, cioè convergenza lineare.

Esercizio 11 Consideriamo una variazione del metodo di Newton, detto metodo della tangente fissa, in cui:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}.$$

Determinare C e s tali che

$$e_{n+1} = Ce_n^s.$$

Esercizio 12 Provare che l'iterazione del metodo di Newton non converge per le seguenti funzioni, per qualsiasi punto (reale) iniziale:

(a) $f(x) = x^2 + 1$

(b) $f(x) = 7x^4 + 3x^2 + \pi$

(c) $f(x) = 1 + e^{x^3}$

(d) $f(x) = 2(sin(1/x^2))^2 + 3(cos(1/x^2))^2$

Esercizio 13 Trovare le condizioni su α in modo da garantire che l'iterazione

$$x_{n+1} = x_n - \alpha f(x_n)$$

converga linearmente allo zero di f se il punto iniziale è sufficientemente vicino a zero.

Esercizio 14 Dimostrare che se r è uno zero di molteplicità k della funzione f, allora si ottiene una convergenza quadratica sostituendo al metodo di Newton l'iterazione seguente

$$x_{n+1} = x_n - k\frac{f(x_n)}{f'(x_n)}.$$

Esercizio 15 Consideriamo una funzione di iterazione del tipo

$$F(x) = x + f(x)g(x),$$

con $f(r) = 0$, $f'(r) \neq 0$. Trovare le condizioni su g affinché il metodo del punto fisso abbia una convergenza cubica alla radice r.

Esercizio 16 Avendo a disposizione una calcolatrice (o, equivalentemente, utilizzando un sistema aritmetico a precisione finita, caratterizzato da $\beta = 10, t = 11$), che numero si ottiene, premendo ripetutamente il bottone della funzione coseno, in corrispondenza di $\pi/4$? Motivare il risultato.

Esercizio 17 Se si utilizza il metodo di Newton per cercare il punto fisso della funzione F, cioè per risolvere l'equazione $F(x) - x = 0$, che formula iterativa si ottiene?

Esercizio 18 Sia p un numero positivo. Dire qual è il valore della seguente espressione:

$$x = \sqrt{p + \sqrt{p + \sqrt{p + \ldots}}}$$

Notare che tale espressione può essere interpretata come

$$x = \lim_{n\to\infty} x_n,$$

con $x_1 = \sqrt{p}$, $x_2 = \sqrt{p + \sqrt{p}}$ e così via.
(**Suggerimento:** osservare che $x_{n+1} = \sqrt{p + x_n}$)

Esercizio 19 Applicando il metodo di iterazione funzionale alla funzione

$$F(x) = 2 + (x - 2)^4,$$

partendo da $x_0 = 2.5$, il metodo converge? A quale valore? Con quale ordine di convergenza? Trovare l'intervallo a cui deve appartenere x_0 affinché il metodo converga. Notare che 2 è un punto fisso.

Esercizio 20 Se il metodo di iterazione funzionale è applicato alla funzione

$$f(x) = \frac{2}{(1 + x^2)}$$

a partire da $x_0 = 7$, la sequenza ottenuta converge? Se così, qual è il limite? Motivare la risposta.

Esercizio 21 Verificare che

$$f(x) = 2 + x - arctg(x)$$

è tale che $|f'(x)| < 1$. Provare, inoltre, che f non ha punti fissi e dire perché ciò non contraddice il Teorema delle contrazioni.

Esercizio 22 Scrivere due diverse procedure di punto fisso per trovare uno zero in $[0.5, 1]$ di

$$f(x) = 2x^2 + 6e^{-x} - 4.$$

Esercizio 23 Se il metodo di iterazione funzionale è applicato a

$$F(x) = x^2 + x - 2$$

e produce una successione convergente di numeri positivi, qual è il suo limite e qual è il valore di x_0?

Esercizio 24 Consideriamo una funzione del tipo $F(x) = x - f(x)f'(x)$, con $f(r) = 0$ e $f'(r) \neq 0$. Trovare le condizioni su f affinché il metodo dell'iterazione funzionale converga al più cubicamente a r se x_0 appartiene ad un suo intorno.

1.13.2 Problemi da risolvere con il calcolatore

Problema 1 Scrivere ed implementare una procedura per l'algoritmo di bisezione. Utilizzare le seguenti funzioni, cambiando l'accuratezza richiesta sulla soluzione ed il massimo numero di iterazioni:

(a) $x^{-1} - tan(x)$ in $[0.1, \pi/2]$

(b) $x^{-1} - 2^x$ in $[0.1, 1]$

(c) $e^x + 2cos(x) - 5.5$ in $[1, 3]$

(d) $(x^3 + 4x^2 + 3x + 5)/(2x^3 - 9x^2 + 18x - 2)$ in $[0, 4]$

Problema 2 Utilizzare la procedura implementata nel **Problema 1** per determinare una radice di $f(x) = x - tan(x)$ in $[1, 2]$.

Problema 3 Utilizzare l'implementazione del **Problema 1** per determinare una radice di

$$x^8 - 36x^7 + 546x^6 - 4536x^5 + 22449x^4 - 67284x^3 + 118124x^2 - 109584x + 40320 = 0,$$

nell'intervallo $[5.5, 6.5]$. Cambiare -36 con -36.001 e ripetere l'esecuzione.

Problema 4 Scrivere ed implementare una procedura per il metodo di Newton. Utilizzare l'implementazione per determinare le radici di $x - tan(x) = 0$ piú vicine a 4.5 e 7.7.

Problema 5 Trovare il minimo positivo della funzione

$$f(x) = x^{-2}tan(x)$$

calcolando lo zero di f' con la routine implementata nel **Problema 4**.

Problema 6 Attraverso la procedura implementata nel **Problema 4**, applicare 10 passi del metodo di Newton, partendo da $x_0 = 5$, per la ricerca delle radici dell'equazione

$$x^3 + 3x - 5x^2 - 7 = 0\,.$$

Problema 7 L'equazione

$$2x^4 + 24x^3 + 61x^2 - 16x + 1 = 0$$

ha due radici vicino a 0.1. Determinarle con la routine che implementa il metodo di Newton.

Problema 8 Utilizzare la procedura implementata nel **Problema 4**, per vedere come varia la radice negativa di

$$f(x) = e^x - 1.5 - arctg(x) + \epsilon = 0$$

al variare del termine costante ϵ tra i valori $0, 10^{-1}$ e 10^{-2}.

Problema 9 Utilizzare una delle due implementazioni, realizzate nel **Problema 1** e nel **Problema 4**, per determinare le prime 10 radici dell'equazione

$$tan(x) = x.$$

Problema 10 Scrivere ed implementare una procedura per il metodo delle secanti che, ricevendo, in input, due valori iniziali x_0 e x_1, una funzione f, una tolleranza $ftol$ sulla valutazione della funzione ed un massimo numero di iterazioni $itmax$, restituisca, in output, un'approssimazione dello zero di f. Eseguire una serie di test al fine di determinare uno zero delle funzioni:

(a) $sin(x/2) - 1$ con $x0 = 0, x1 = 1$ e $x0 = 6, x1 = 7$;

(b) $e^x - tan(x)$ con $x0 = 0, x1 = 1$ e $x0 = -7, x1 = -6$;

(c) $x^3 - 12x^2 + 3x + 1$ con $x0 = 0, 1, 2, 3$ e $x1 = x0 + 1$.

Problema 11 Scrivere ed implementare un algoritmo che risulti un raffinamento del metodo delle secanti, utilizzando l'approssimazione di f'

$$f'(x) \approx \frac{k^2 f(x+h) - h^2 f(x+k) + (h^2 - k^2) f(x)}{(k-h)kh}$$

nella formula di Newton. Osserviamo che sono necessari tre valori iniziali.

Problema 12 Risolvere le seguenti equazioni, utilizzando l'implementazione della procedura per il metodo delle secanti (**Problema 10**):

(a) $x^{20} - 1 = 0$, nell'intervallo $[x0, x1] = [0, 10]$;

(b) $tan(x) - 30x = 0$, nell'intervallo $[x0, x1] = [1, 1.57]$;

(c) $x^2 - (1 - x)^{10} = 0$, nell'intervallo $[x0, x1] = [0, 1]$;

(d) $x^3 + 10^{-4} = 0$, nell'intervallo $[x0, x1] = [-0.75, 0.5]$;

(e) $x^{19} + 10^{-4} = 0$, nell'intervallo $[x0, x1] = [-0.75, 0.5]$;

(f) $x^5 = 0$, nell'intervallo $[x0, x1] = [-1, 10]$;

(g) $x^9 = 0$, nell'intervallo $[x0, x1] = [-1, 10]$;

(h) $xe^{-x^2} = 0$, nell'intervallo $[x0, x1] = [-1, 4]$.

Problema 13 Testare il programma che implementa il metodo delle secanti sulla funzione

$$f(x) = x^3 - sinh(x) + 4x^2 + 6x + 9,$$

utilizzando come punti iniziali, dapprima 7 e 8 e poi 3 e 10. Commentare i risultati ottenuti.

Problema 14 Scrivere ed implementare una procedura per il metodo ibrido di Dekker-Brent. Testare la procedura implementata sulle funzioni dei **Problemi 10** e **12**.

Problema 15 Determinare una stima con 10 cifre significative della radice r di

$$x^3 + 3x - 5x^2 - 7 = 0$$

nell'intervallo $[0, 1]$, confrontando i risultati ottenuti attraverso le implementazioni del:

(a) metodo di bisezione con $[a, b] = [0, 1]$;

(b) metodo delle secanti con $[x0, x1] = [0, 1]$;

(c) metodo di Newton con $x0 = 1$;

(d) metodo di Dekker-Brent con $[x0, x1] = [0, 1]$.

Bibliografia

[1] Brent R. - *Algorithms for Minimization Without Derivatives* - Prentice-Hall, 1973.

[2] Cheney W., Kincaid D. R. - *Numerical Mathematics and Computing* - sixth ed., Pacific Grove, CA, Brooks/Cole, 2007.

[3] Dahlquist G., Björck A. - *Numerical Methods* - Prentice-Hall, 1974.

[4] Dennis J. E. Jr, Schnabel Robert B. - *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* - Classics in Applied Mathematics, 16, Soc. for Industrial & Applied Math., 1996.

[5] Fong K. W. , Jefferson T. H., Suyehiro T., Walton L. - *SLATEC Common Mathematical Library* - (1993), `http://www.netlib.org/slatec/`.

[6] Forsythe G., Malcom M. and Moler C. - *Computer methods for mathematical computations* - Prentice-Hall, 1977.

[7] Heath M. T. *Scientific Computing: An Introductory Survey* - McGraw-Hill, II ed. 2002

[8] Moré J., Garbow B., Hillstrom K. - *MINPACK* - `http://www.netlib.org/minpack/`

[9] *NAPACK* - `http://www.netlib.org/napack/`

[10] Numerical Algorithm Group - *NAg Library Manual, Mark 22* - Oxford (2009), `http://www.nag.com/`.

[11] *Transactions on Mathematical Software (TOMS)* - `http://www.netlib.org/toms`.

[12] Visual Numerics - *IMSL C Numerical Library version 7.0 (2008), IMSL Fortran Numerical Library Version 6.0 (2007)* - `http://www.vni.com`

Capitolo 2

La quadratura

2.1 Principali formule di quadratura

In questo capitolo sono ripresi e approfonditi i concetti alla base della costruzione e implementazione delle formule di quadratura, discussi nel **Capitolo 4** della **Parte prima**. In particolare, si descrivono le formule di quadratura esatte su uno spazio funzionale inquadrandole nell'ambito dei metodi di discretizzazione di un integrale definito, studiandone la convergenza, il condizionamento e la propagazione dell'errore di round off. Dopo un breve cenno alle problematiche relative alla costruzione di formule di quadratura multidimensionali, si discutono i concetti alla base dell'integrazione numerica di funzioni oscillanti periodiche, mostrando come un'attenta analisi del problema, consente di ottenere formule di quadratura efficienti.

2.1.1 Costruzione delle formule di quadratura

Dal **teorema fondamentale del calcolo integrale** è noto che

$$I[f] = \int_a^b f(x)\,dx = F(b) - F(a)$$

con $F(x)$ funzione tale che $F'(x) = f(x)$ (cioè F *funzione primitiva* di f).

♣ **Esempio 2.1.** Si calcoli:

$$I[f] = \int_1^3 \frac{dx}{x} = \log 3 - \log 1 = \log 3 = 1.098612288668... \simeq 0.109861 \times 10^1$$

In questo caso si ottiene un risultato approssimato dell'integrale, dovuto all'approssimazione della funzione $\log(x)$. ♣

♣ **Esempio 2.2.** Si calcoli:

$$I[f] = \int_0^1 e^{x^2}\, dx = \int_0^1 \sum_{i=0}^{\infty} \frac{x^{2i}}{i!}$$

Per calcolare tale integrale è necessario dapprima sviluppare in serie di Taylor la funzione integranda, quindi, troncare la serie ad un fissato termine L_{max}, cioè:

$$I[f] \simeq \sum_{i=0}^{L_{max}} \left[\frac{x^{2i+1}}{i!(2i+1)}\right]_0^1 \quad \text{ed} \quad E[f] = \sum_{i=L_{max}+1}^{\infty} \left[\frac{x^{2i+1}}{i!(2i+1)}\right]_0^1.$$

♣

Il calcolo dell'integrale negli esempi precedenti richiede un'approssimazione numerica che riguarda la primitiva della funzione integranda, o la stessa funzione f. L'idea alla base della quadratura è invece quella di sviluppare metodi che approssimino direttamente il valore di $I[f]$.
Al funzionale:

$$Q[f] = \sum_{i=1}^{n} A_i f(x_i) \tag{2.1}$$

si dà il nome di **formula di quadratura**, mentre al funzionale:

$$E[f] = I[f] - Q[f] = \int_a^b f(x)\, dx - \sum_{i=1}^{n} A_i f(x_i)$$

si dà il nome di **errore di discretizzazione**.

Si noti che la formula di quadratura (2.1) è un funzionale lineare così come $I[f]$ ed $E[f]$.
Il funzionale $Q[f]$ è determinato univocamente quando si fissa il valore di n ed i $2n$ parametri della formula di quadratura (n pesi A_i e n nodi x_i).
Uno degli obiettivi fondamentali della quadratura è far sì che l'errore $E[f]$ associato a $Q[f]$ sia "in qualche senso" il più piccolo possibile, in maniera da determinare una formula di quadratura "accurata".

A tal fine la ricerca dei pesi e dei nodi più opportuni può avvenire prevalentemente seguendo due strategie:

- si può richiedere di minimizzare il massimo errore su un generico spazio di funzioni $\mathcal{G}$, cioè:

$$\min_{A_i, x_i} \sup_{f \in \mathcal{G}} |E[f]| = \min_{A_i, x_i} \sup_{f \in \mathcal{G}} |I[f] - Q[f]| \tag{2.2}$$

  dove il minimo è scelto al variare dei parametri A_i e x_i della formula di quadratura $Q[f]$. Tali formule sono note come **formule ottimali rispetto allo spazio di funzioni** $\mathcal{G}$.

- si può imporre che l'errore sia nullo per uno spazio di funzioni $\mathcal{F}$ a dimensione finita, cioè

$$E[f] = I[f] - Q[f] = 0 \quad \forall f \in \mathcal{F}. \tag{2.3}$$

  Tali formule sono note come **formule di annullamento dell'errore** oppure come **formule esatte per lo spazio di funzioni $\mathcal{F}$**.

Per semplificare la costruzione delle formule di quadratura, in entrambi gli approcci è possibile fissare delle condizioni sui parametri; per esempio si può richiedere che i pesi siano uguali o, più comunemente, si possono fissare a priori i nodi.

Per la loro semplicità rispetto alle formule ottimali, le formule esatte per uno spazio di funzioni sono quelle più diffuse. Per tale ragione esse sono anche quelle maggiormente utilizzate per la realizzazione di software matematico per la quadratura.

2.1.2 Formule esatte per uno spazio di funzioni

Le formule esatte per uno spazio di funzioni $\mathcal{F}$ sono formule di quadratura che, applicate ad ogni funzione $f \in \mathcal{F}$, forniscono un risultato esatto, nel senso che annullano l'errore di discretizzazione $E[f]$.

Fissato uno spazio di funzioni $\mathcal{F}$ di dimensione finita q e scelta una sua base $\{\phi_r(x)\}, r = 1, ..., q$, $\forall f \in \mathcal{F}$ sussiste una relazione del tipo:

$$f(x) = \sum_{r=1}^{q} b_r \phi_r(x), \qquad \forall f \in \mathcal{F}$$

da cui, per la linearità del funzionale $E[f]$, si ha:

$$E[f] = E\left[\sum_{r=1}^{q} b_r \phi_r(x)\right] = b_1 E[\phi_1(x)] + \cdots + b_q E[\phi_q(x)].$$

Pertanto, costruire una formula di quadratura esatta per $\mathcal{F}$ equivale a determinare i nodi ed i pesi in maniera tale che risulti verificata la relazione:

$$E[\phi_r(x)] = 0, \qquad r = 1, ..., q. \tag{2.4}$$

La (2.4) rappresenta un sistema di equazioni non lineari nelle incognite x_i ed A_i. Se si fissano a priori gli n nodi distinti, il sistema (2.4) equivale ad un sistema lineare di q equazioni nelle n incognite A_i $(i = 1, ..., n)$. In tal caso il sistema è compatibile e determinato se e solo se $q = n$. Il calcolo dei pesi A_i mediante il sistema (2.4) è detto **metodo dei coefficienti indeterminati**.

Usualmente, lo spazio di funzioni $\mathcal{F}$ che si sceglie è quello dei polinomi al più di grado n (Π_n). [1]

♣ **Esempio 2.3.** Si determini una formula di quadratura nell'intervallo $[-1,1]$ tale che $E[f] = 0$ $\forall f \in \Pi_2$. Per la linearità del funzionale $E[f]$, la relazione precedente si verifica se e solo se

$$E[f] = E[b_2x^2 + b_1x + b_0] = \sum_{r=0}^{2} b_r E[x^r] = 0,$$

cioè se e solo se $E[x^r] = 0$ per $r = 0, 1, 2$. Si ha quindi:

$$\begin{cases} A_1 & +A_2 & \dots & +A_n & = \int_{-1}^{1} dx = 2 \\ A_1x_1 & +A_2x_2 & \dots & +A_nx_n & = \int_{-1}^{1} x\,dx = 0 \\ A_1x_1^2 & +A_2x_2^2 & \dots & +A_nx_n^2 & = \int_{-1}^{1} x^2\,dx = 2/3 \end{cases}$$

Il sistema precedente è un sistema non lineare di 3 equazioni nelle $2n$ incognite costituite dai nodi x_i e dai pesi A_i. Se i nodi x_i vengono fissati, ad esempio equidistanziati, si ottiene un sistema di equazioni lineare nelle n incognite A_i. Per garantire la compatibilità del sistema si deve porre $n = 3$ in modo da avere un sistema lineare di 3 equazioni in 3 incognite, la cui matrice dei coefficienti è

$$V = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Poichè V è non singolare il sistema ha l'unica soluzione:

$$A_1 = A_3 = \frac{1}{3} \qquad A_2 = \frac{4}{3}$$

da cui:

$$Q[f] = \frac{1}{3}(f(-1) + 4f(0) + f(1)).$$

Tale formula è nota come formula di Simpson, già ottenuta per altra via nel **§4.2** del **Capitolo 4**, **Parte prima**, nel generico intervallo $[a, b]$.

♣

Definizione 2.1.1. (Grado di precisione)
Una formula di quadratura ha **grado di precisione** *q se è esatta per uno spazio di funzioni di dimensione al più q. In particolare si parla di grado di precisione* **algebrico** *nel caso in cui $\mathcal{F}$ coincide con lo spazio dei polinomi Π_q o di grado di precisione* **trigonometrico** *se si considera lo spazio dei polinomi trigonometrici del tipo:*

$$S_q(x) = a_0 + \sum_{k=1}^{q} (a_k \cos kx + b_k \sin kx)$$

[1] Tale scelta è dovuta alle buone proprietà dei polinomi. Se $p(x)$ e $q(x)$ sono due polinomi, $p(x) \pm q(x)$, $p(x) \cdot q(x)$ e $p(q(x))$ sono anch' essi dei polinomi. Inoltre anche la derivata $p'(x)$ e l'integrale $\int p(x)\,dx$ sono dei polinomi. I polinomi sono anche strettamente collegati alle funzioni analitiche attraverso gli sviluppi in serie di potenze ed alle funzioni continue attraverso il teorema di Weierstrass. Un'ultima (ma non meno importante) proprietà dei polinomi è che essi sono rappresentati da espressioni per la cui valutazione sono noti algoritmi efficienti.

Data la formula di quadratura $Q[f] = \sum_{i=1}^{n} A_i f(x_i)$, se si impone che $Q[f]$ abbia grado di precisione algebrico q e si fissano gli n nodi $x_i,\ i = 1, ..., n$, si ottiene un sistema lineare di $q+1$ equazioni in n incognite, costituite dagli n pesi A_i:

$$\begin{cases} A_1 + A_2 + \cdots + A_n & = \int_a^b dx \\ A_1 x_1 + A_2 x_2 + \cdots + A_n x_n & = \int_a^b x\, dx \\ \cdots & \cdots \\ A_1 x_1^q + A_2 x_2^q + \cdots + A_n x_n^q & = \int_a^b x^q\, dx \end{cases} \tag{2.5}$$

la cui matrice dei coefficienti è:

$$V = \begin{pmatrix} 1 & ... & 1 \\ x_1 & & x_n \\ \vdots & & \vdots \\ x_1^q & ... & x_n^q \end{pmatrix} \tag{2.6}$$

Si osservi che tale sistema ha un'unica soluzione se e solo se $q = n-1$ **e** $x_i \neq x_j$**,** $\forall i \neq j$. Ciò vuol dire che la formula $Q[f]$, i cui pesi soddisfano il sistema precedente, ha grado di precisione algebrico $n-1$. Inoltre, dalla prima equazione si ha

$$A_1 + \cdots + A_n = \int_a^b dx = (b-a) \tag{2.7}$$

cioè la somma dei pesi di una formula di quadratura esatta per lo spazio dei polinomi è uguale all'ampiezza dell'intervallo di integrazione.

Si noti che la matrice (2.6) è una matrice di Vandermonde il cui indice di condizionamento cresce almeno esponenzialmente con la dimensione n (cfr. §6.2) e quindi il relativo sistema (2.5) è malcondizionato.

Un modo alternativo per il calcolo dei pesi si basa sull'osservazione che una formula esatta per lo spazio dei polinomi Π_{n-1} è esatta, in particolare, per i polinomi fondamentali di Lagrange $l_j(x)$ [2], ovvero

$$\int_a^b l_j(x)\, dx = \sum_{i=1}^{n} A_i l_j(x_i) = A_j \qquad j = 1, \ldots, n$$

Ricordando l'espressione dei polinomi fondamentali di Lagrange (equazione (3.17) del **§3.2.5**, **Capitolo 3**, **Parte prima**), ciascun l_i si può scrivere come:

$$l_i(x) = \frac{\omega(x)}{(x - x_i)\omega'(x_i)}, \quad \text{con} \quad \omega(x) = (x - x_1) \cdots (x - x_n)$$

[2] Si ricorda che

$$l_i(x_k) = \begin{cases} 1 & i = k, \\ 0 & i \neq k \end{cases} .$$

da cui si deduce che i pesi hanno espressione:

$$A_i = \int_a^b \frac{\omega(x)}{(x - x_i)\omega'(x_i)}\, dx, \quad \text{con} \quad \omega(x) = (x - x_1) \cdots (x - x_n) \tag{2.8}$$

Anche questo metodo, però, è poco efficace, per la necessità del calcolo dell'integrale (2.8).
Enunciamo infine alcune definizioni e risultati che verranno utilizzati nel seguito:

Definizione 2.1.2. *Sia $Q_n[A, f]$ una formula di quadratura in $[a, b]$ con $x_0 < x_1 < \ldots < x_n$. Se $x_0 = a$ e $x_n = b$ allora si dice che $Q_n[A, f]$ è **chiusa**, altrimenti si dice che $Q_n[A, f]$ è **aperta**.*

Proposizione 2.1.1. *[19]*
Sia s un intero non negativo, $[c, d]$ e $[a, b]$ due intervalli e

$$t = \phi(x) = x \cdot \frac{b - a}{d - c} + \frac{ad - bc}{d - c} \tag{2.9}$$

la trasformazione lineare da $[c, d]$ in $[a, b]$. Sia $w_1(x)$ una funzione peso su $[c, d]$, $w_2(t) = w_1(\phi^{-1}(t))$ la funzione peso su $[a, b]$ ottenuta da $w_1(x)$ e $Q[B, f] = \sum_{i=0}^{n} B_i f(x_i)$ una formula di quadratura in $[c, d]$, con grado di precisione algebrico s rispetto a $w_1(x)$, cioè:

$$\int_c^d w_1(x) q(x) dx = \sum_{i=0}^{n} B_i q(x_i) \qquad \forall q \in \Pi_s \tag{2.10}$$

Consideriamo la formula di quadratura su $[a, b]$, $Q[A, f] = \sum_{i=0}^{n} A_i f(t_i)$, i cui pesi A_i sono proporzionali ai pesi B_i di $Q[B, f]$ per mezzo del fattore $\frac{b-a}{d-c}$ ed i cui nodi t_i sono ottenuti dai nodi x_i mediante la $t = \phi(x)$.
Allora $Q[A, f]$ ha grado di precisione algebrico s, su $[a, b]$, rispetto a $w_2(x)$, cioè:

$$\int_a^b w_2(t) q(t) dt = \sum_{i=0}^{n} A_i q(t_i) \qquad \forall q \in \Pi_s \ .$$

2.1.3 Formule di Newton-Cotes

Un modo per ottenere formule esatte per lo spazio dei polinomi Π_{n-1}, è quello di fissare gli n nodi x_i con $x_i \neq x_j$ per $i \neq j$. Benché la scelta di tali nodi possa essere del tutto arbitraria, una scelta naturale è quella di fissarli equidistanziati nell'intervallo $[a, b]$, cioè:

$$x_1 = a < x_2 = a + h < ... < x_n = a + (n - 1)h = b \qquad h = \frac{(b - a)}{(n - 1)}$$

In tal caso, ad esempio, si ha:

$n = 2$ **Formula trapezoidale:**

$$T[f] = h\left[\frac{f(a)}{2} + \frac{f(b)}{2}\right] \qquad h = (b-a)$$

esatta per polinomi di grado 1.

$n = 3$ **Formula di Simpson:**

$$S[f] = h\left[\frac{f(a)}{3} + \frac{4}{3}f(a+h) + \frac{f(b)}{3}\right] \qquad h = \frac{(b-a)}{2}$$

esatta per polinomi di grado 2.

$n = 4$ **Formula dei '3/8':**

$$Q[f] = h\left[\frac{3}{8}f(a) + \frac{9}{8}f(a+h) + \frac{9}{8}f(a+2h) + \frac{3}{8}f(b)\right] \qquad h = \frac{(b-a)}{3}$$

esatta per polinomi di grado 3.

Dalle espressioni precedenti si deduce che le formule così ottenute si possono esprimere come:

$$Q[f] = h\left[B_1 f(x_1) + \cdots + B_n f(x_n)\right] \quad \text{con} \quad h = \frac{b-a}{n-1} \tag{2.11}$$

dove i valori B_i sono **indipendenti** dall'intervallo di integrazione $[a, b]$ e dalla funzione integranda, ma dipendono solo dal numero di nodi n. Inoltre i pesi relativi a nodi simmetrici rispetto al punto medio dell'intervallo $[a, b]$ sono uguali, cioè:

$$B_i = B_{n-i+1} \qquad i = 1, 2, ..., \left\lfloor \frac{n+1}{2} \right\rfloor$$

Per essi è quindi possibile una tabulazione opportuna (Tabella 2.1).

Definizione 2.1.3. (Formule di Newton-Cotes)
Le formule di quadratura con n nodi equidistanziati, esatte per lo spazio dei polinomi Π_{n-1}*, sono dette* **formule di Newton-Cotes**.

Dalla Tabella 2.1 si nota che per $n \leq 8$ e per $n = 10$ i pesi B_i sono tutti positivi[3], mentre per $n = 9$ (così come per $n \geq 11$), i pesi B_i sono di segno alterno e sono crescenti in modulo. La formula con $n = 10$ ha grado di precisione 9 e rappresenta la formula di Newton-Cotes con pesi tutti positivi, con grado di precisione algebrico più alto.
Se $Q[f]$ è la formula di Newton-Cotes in $[0, 1]$, abbiamo una rappresentazione asintotica dei suoi pesi B_i [4]:

$$B_0 = B_n = \frac{1}{n \log n}\left[1 + \mathcal{O}\left(\frac{1}{\log n}\right)\right] \tag{2.12}$$

[3]Come si vedrà nel paragrafo 2.3, tale caratteristica influenza fortemente il condizionamento di una formula di quadratura.

[4][5]

Formula	n	$B_1 = B_n$	$B_2 = B_{n-1}$	$B_3 = B_{n-2}$	$B_4 = B_{n-3}$	$B_5 = B_{n-4}$
2 nodi ($Trapezoidale$)	2	$\frac{1}{2}$				
3 nodi ($Simpson$)	3	$\frac{1}{3}$	$\frac{4}{3}$			
4 nodi (dei '3/8')	4	$\frac{3}{8}$	$\frac{9}{8}$			
5 nodi ($Boole$)	5	$\frac{14}{45}$	$\frac{64}{45}$	$\frac{24}{45}$		
6 nodi	6	$\frac{95}{288}$	$\frac{375}{288}$	$\frac{250}{288}$		
7 nodi	7	$\frac{41}{140}$	$\frac{216}{140}$	$\frac{27}{140}$	$\frac{272}{140}$	
8 nodi	8	$\frac{5257}{17280}$	$\frac{25039}{17280}$	$\frac{9261}{17280}$	$\frac{20923}{17280}$	
9 nodi	9	$\frac{3956}{14175}$	$\frac{23552}{14175}$	$\frac{-3712}{14175}$	$\frac{41984}{14175}$	$\frac{-18160}{14175}$
10 nodi	10	$\frac{25713}{89600}$	$\frac{141669}{89600}$	$\frac{9720}{89600}$	$\frac{174096}{89600}$	$\frac{52002}{89600}$

Tabella 2.1: Coefficienti delle formule di Newton-Cotes con $n \leq 10$.

e, per ciascun $i = 1, 2, \ldots, n-1$,

$$B_i = \frac{(-1)^{i-1}}{n \log^2 n} \binom{n}{i} \left[\frac{1}{i} + \frac{(-1)^n}{n-i} \right] \left[1 + \mathcal{O}\left(\frac{1}{\log n} \right) \right] \tag{2.13}$$

Si noti, infine, che se $Q^{(j)}[f]$ è una formula di Newton-Cotes, nel generico intervallo $[t_{j-1}, t_j]$ di ampiezza $\tau = (b-a)/m$, per la (2.11) essa avrà espressione

$$Q^{(j)}[f] = h \sum_{i=1}^{n} B_i f(x_i^{(j)}) \quad \text{con} \quad h = \frac{\tau}{n-1} = \frac{(b-a)}{m(n-1)}$$

con i pesi B_i opportunamente scelti nella Tabella 2.1. In tal caso la formula composita corrispondente è:

$$Q_m[f] = \sum_{j=1}^{m} Q^{(j)}[f] = h \sum_{j=1}^{m} \sum_{i=1}^{n} B_i f(x_i^{(j)})$$

con

$$h = \frac{(b-a)}{m(n-1)} \quad \text{e} \quad x_i^{(j)} = t_{j-1} + (i-1)h.$$

2.1.4 Formule di Gauss

Se non si impone alcun vincolo sui nodi, qual è il massimo grado di precisione algebrico di $Q[f]$?

♣ **Esempio 2.4.** Determinare una formula di quadratura nell'intervallo $[-1, 1]$ con il grado di precisione algebrico più alto possibile, utilizzando 2 nodi. Tale formula ha espressione:

$$Q[f] = A_1 f(x_1) + A_2 f(x_2).$$

Essendo disponibili 4 parametri (2 pesi e 2 nodi), quindi 4 incognite da calcolare, è ragionevole imporre di integrare esattamente un qualunque polinomio di terzo grado, ovvero di imporre 4 condizioni relative alle 4 funzioni della base di Π_3, cioè:

$$\mathcal{F} = \{1, x, x^2, x^3\}.$$

Si ha così il sistema:

$$\begin{cases} A_1 & +A_2 & = 2 \\ A_1 x_1 & +A_2 x_2 & = 0 \\ A_1 x_1^2 & +A_2 x_2^2 & = 2/3 \\ A_1 x_1^3 & +A_2 x_2^3 & = 0 \end{cases} \tag{2.14}$$

Il sistema (2.14) è non lineare. Se si osserva che l'intervallo $[-1, 1]$ è simmetrico rispetto allo 0, è plausibile che la soluzione conservi tale caratteristica, cioè $x_2 = -x_1$. Dalla seconda equazione di (2.14) segue allora che $A_1 = A_2$ e dalla prima equazione si ricava che $A_1 = A_2 = 1$. Dalla terza equazione di (2.14) segue infine che

$$x_1^2 = \frac{1}{3} \quad \Leftrightarrow \quad x_1 = \frac{-1}{\sqrt{3}} \ , \ \ x_2 = \frac{1}{\sqrt{3}}$$

Si ha quindi la formula:

$$Q[f] = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

La formula così ottenuta ha grado di precisione algebrico 3 ed è detta **formula di Gauss** con 2 nodi. ♣

Teorema 2.1.1. *Sia $G[f]$ una formula di quadratura del tipo*

$$G[f] = \sum_{i=1}^{n} A_i f(x_i) \tag{2.15}$$

Condizione necessaria e sufficiente affinché $G[f]$ abbia grado di precisione algebrico $2n-1$ è che sia esatta per polinomi $r(x)$ di grado al più $n-1$, cioè tale che:

$$\int_a^b r(x)\, dx = \sum_{i=1}^{n} A_i r(x_i) \quad \forall r \in \Pi_{n-1} \tag{2.16}$$

e che abbia come nodi $x_i \quad i = 1, ..., n$ *gli* n *zeri di un polinomio* $\omega(x) \in \Pi_n$ *ortogonale*[5] *a tutti i polinomi* $q(x)$ *di grado al più* $n-1$*, cioè tale che:*

$$\int_a^b \omega(x)q(x)\,dx = 0 \quad \forall q \in \Pi_{n-1} \tag{2.17}$$

Dimostrazione Se $G[f]$ ha grado di precisione $2n-1$ la (2.16) è banalmente soddisfatta. Siano $x_1, ..., x_n$ i nodi di $G[f]$ e sia $q(x)$ un generico polinomio di grado al più $n-1$. Posto allora $\omega(x) = (x-x_1)\cdots(x-x_n)$ e $p(x) = \omega(x)q(x)$ si ha che $p \in \Pi_{2n-1}$ e per tale polinomio, per ipotesi, si deve avere $E[p] = 0$, cioè:

$$0 = E[p] = \int_a^b p(x)\,dx - \sum_{i=1}^n A_i p(x_i) = \int_a^b \omega(x)q(x)\,dx - \sum_{i=1}^n A_i \omega(x_i)q(x_i)$$

ma poiché $\omega(x_i) = 0$, $i = 1, ..., n$, si deve avere

$$\int_a^b \omega(x)q(x)\,dx = 0$$

cioè la (2.17).

Viceversa, siano verificate le condizioni (2.16) e (2.17), e sia $p(x)$ un generico polinomio di grado al più $2n-1$. Detti $x_1, ..., x_n$ i nodi di $G[f]$ e posto $\omega(x) = (x-x_1)\cdots(x-x_n)$, dividendo $p(x)$ per $\omega(x)$ si ottiene:

$$p(x) = \omega(x)q(x) + r(x)$$

con $q(x)$ e $r(x)$ polinomi di grado al più $n-1$, rispettivamente quoziente e resto della divisione. Si ha quindi che

$$\begin{aligned} E[p] &= \int_a^b p(x)\,dx - \sum_{i=1}^n A_i p(x_i) \\ &= \int_a^b \omega(x)q(x)\,dx + \int_a^b r(x)\,dx - \sum_{i=1}^n A_i \omega(x_i)q(x_i) - \sum_{i=1}^n A_i r(x_i) \end{aligned}$$

da cui, per le condizioni (2.16) e (2.17) ed osservando che $\omega(x_i) = 0$, $i = 1, ..., n$, si ha $E[p] = 0 \quad \forall p \in \Pi_{2n-1}$.
Si ha poi che per il polinomio di grado $2n$:

$$p(x) = (x-x_1)^2 \cdots (x-x_n)^2$$

si avrà sempre:

$$\sum_{i=1}^n A_i p(x_i) = 0 \quad e \quad \int_a^b p(x)\,dx > 0$$

per ogni scelta dei pesi e dei nodi, per cui $2n-1$ è effettivamente il massimo grado di precisione possibile per una formula di quadratura di annullamento dell'errore con n nodi. ■

[5] Si veda l'**Appendice C**.

Definizione 2.1.4. (Formula di Gauss)
Una formula di quadratura su n nodi

$$G[f] = \sum_{i=1}^{n} A_i f(x_i) \tag{2.18}$$

avente grado di precisione algebrico $2n-1$ *è detta* **formula di Gauss**.

Come già detto per le formule di Newton-Cotes, anche per le formule di Gauss la matrice del sistema lineare proveniente dal metodo dei coefficienti indeterminati ha indice di condizionamento elevato al crescere di n. Pertanto per il calcolo dei pesi A_i delle formule di Gauss si utilizza un'altra rappresentazione.

A tal fine, fissato un sistema di polinomi ortogonali $\{p_n(x)\}$ nell'intervallo $[a,b]$, si consideri il sistema di polinomi ortonormali $\{p_n^*(x)\}$ corrispondente, ottenuto ponendo:

$$p_n^*(x) = \frac{p_n(x)}{||p_n(x)||_2}$$

con:

$$||p_n(x)||_2^2 = \int_a^b p_n^2(x)\,dx$$

Teorema 2.1.2. *Fissati i nodi secondo la (2.17), i pesi* A_i *delle formule di Gauss sono forniti dalla relazione seguente:*

$$A_i = -\frac{d_{n+1}}{d_n p_{n+1}^*(x_i) p_n^{*\prime}(x_i)} \qquad i = 1, ..., n \tag{2.19}$$

dove d_n *è il coefficiente del termine di grado massimo* (coefficiente direttore) *del polinomio* $p_n^*(x)$ *e* x_i *è l'i-mo nodo della formula di Gauss con n nodi.*

Dimostrazione La dimostrazione parte dall'uguaglianza nota come uguaglianza di Christoffel-Darboux:

$$(x-t)\sum_{k=0}^{n} p_k^*(x)p_k^*(t) = \frac{d_n}{d_{n+1}}\left[p_{n+1}^*(x)p_n^*(t) - p_n^*(x)p_{n+1}^*(t)\right]$$

Valutando tale uguaglianza per $t = x_i$ e dividendo ambo i membri per $x - x_i$, poiché $p_n^*(x_i) = 0$, si ottiene:

$$\sum_{k=0}^{n-1} p_k^*(x)p_k^*(x_i) = -\frac{d_n}{d_{n+1}}\frac{p_n^*(x)p_{n+1}^*(x_i)}{x - x_i}$$

Integrando entrambi i membri di tale uguaglianza nell'intervallo $[a,b]$ si ha:

$$\sum_{k=0}^{n-1}\int_a^b p_k^*(x)p_k^*(x_i)\,dx = -\frac{d_n}{d_{n+1}}\int_a^b \frac{p_n^*(x)p_{n+1}^*(x_i)}{x - x_i}\,dx \tag{2.20}$$

per l'ortonormalità del sistema di polinomi $p_0^*(x), ..., p_n^*(x)$ si ha:

$$p_k^*(x_i)\int_a^b p_k^*(x)\,dx = \begin{cases} 0 & \text{se } k \geq 1 \\ 1 & \text{se } k = 0 \end{cases}$$

per cui la (2.20) diventa:

$$1 = -\frac{d_n}{d_{n+1}} p^*_{n+1}(x_i) \int_a^b \frac{p^*_n(x)}{x - x_i}\, dx \tag{2.21}$$

Poichè le formule di Gauss sono formule esatte per lo spazio dei polinomi, per i pesi di tali formule vale la (2.8), o equivalentemente:

$$A_i = \int_a^b \frac{p^*_n(x)}{(x - x_i) p^{*'}_n(x_i)}\, dx,$$

per cui, moltiplicando e dividendo per $p^{*'}_n(x_i)$ il secondo membro della (2.21) si ha la tesi. ■

Utilizzando la (2.19) è possibile calcolare i pesi delle formule di Gauss per un numero qualunque di nodi. Su tali espressioni sono basate ad esempio le routine `D01BBF` e `D01BCF` della libreria NAg per il calcolo dei pesi di varie formule di Gauss.

Teorema 2.1.3. *Data una formula di Gauss*

$$G[f] = \sum_{i=1}^{n} A_i f(x_i)$$

si ha che $A_i > 0 \quad \forall i = 1, ..., n.$

Dimostrazione Sia $\omega(x)$ il polinomio ortogonale di grado n nell'intervallo $[a, b]$. Detti $x_1, ..., x_n$ gli n nodi di $G[f]$ si ha che $\omega(x_i) = 0$. Per ogni fissato indice j, sia quindi $q(x) = \omega(x)/(x - x_j)$. Poiché gli zeri di $\omega(x)$ sono tutti semplici si ha:

$$q(x_i) \begin{cases} = 0 \text{ se } i \neq j \\ \neq 0 \text{ se } i = j \end{cases}$$

Poiché inoltre $q^2(x)$ è un polinomio di grado $2n - 2$, per tale polinomio la formula $G[f]$ è esatta. Di conseguenza si ha:

$$0 < \int_a^b q^2(x)\, dx = \sum_{i=1}^{n} A_i q^2(x_i) = A_j q^2(x_j)$$

da cui la tesi. ■

È immediata l'estensione dei teoremi precedenti, al caso in cui si voglia calcolare l'integrale:

$$\int_a^b \psi(x) f(x)\, dx,$$

con ψ funzione peso ammissibile, cioè tale che

- $\psi(x) \geq 0$ e non quasi ovunque nulla;
- $\psi(x) f(x)$ integrabile in $[a, b]$;

$\psi(x)$	intervallo	tipo
1	$[-1,1]$	Gauss-Legendre
$(1-x^2)^{-1/2}$	$[-1,1]$	Gauss-Chebyshev (I)
$(1-x^2)^{1/2}$	$[-1,1]$	Gauss-Chebyshev (II)
$(1-x)^\alpha(1+x)^\beta \quad \alpha,\beta > -1$	$[-1,1]$	Gauss-Jacobi
e^{-x}	$[0,\infty[$	Gauss-Laguerre
$e^{-x^2/2}$	$]-\infty,\infty[$	Gauss-Hermite

Tabella 2.2: Funzioni peso ed intervalli relativi alle principali formule di Gauss.

- $|\int_a^b \psi(x)x^k\,dx| < \infty \quad \forall k = 0,1,\ldots$

In generale, solo se la funzione peso ψ è ammissibile è assicurata l'esistenza di un polinomio $\omega \in \Pi_n$ ortogonale a tutti i polinomi q al più di grado $n-1$ rispetto a tale funzione peso e tale che le sue radici siano reali, distinte e tutte interne ad $[a,b]$.

A seconda della funzione peso ψ e dell'intervallo $[a,b]$, si hanno formule di Gauss che prendono il nome dalla famiglia di polinomi ortogonali relativi alla funzione ψ.

Poichè i risultati esposti valgono per vari tipi di formule di Gauss, per semplicità in seguito, dove non dichiarato esplicitamente, si farà riferimento sempre alle formule di Gauss-Legendre.

Dalle proprietà degli zeri dei polinomi ortogonali si ricavano alcune caratteristiche delle formule di Gauss:

- le formule di Gauss sono di tipo *aperto*, cioè gli estremi dell'intervallo $[a,b]$ non appartengono all'insieme dei nodi;
- gli insiemi dei nodi di due distinte formule di Gauss sono disgiunti, ad eccezione del punto medio dell'intervallo $[a,b]$ presente in tutte le formule con un numero di nodi dispari.

Nella Tabella 2.3 sono riportati, ad esempio, i valori dei pesi e dei nodi delle formule di Gauss-Legendre nell'intervallo $[-1,1]$, con 8 cifre significative (a nodi di modulo uguale e di segno opposto corrispondono pesi uguali). [6]

♣ **Esempio 2.5.** Si calcoli l'integrale

$$I[f] = \int_1^2 f(t)\,dt$$

[6] È da notare, infine, che i polinomi ortogonali sono definiti generalmente nell'intervallo $[-1,1]$. Per tale motivo i nodi ed i pesi delle formule di Gauss vengono calcolati in $[-1,1]$ e poi trasformati per il generico intervallo $[a,b]$.

n	nodi	pesi
1	0.0	2.
2	$\pm$.57735027	1.
3	$\pm$.77459667	.55555556
	0.0	.88888889
4	$\pm$.86113631	.34785485
	$\pm$.33998104	.65214515
5	$\pm$.90617984	.23692689
	$\pm$.53846931	.47862867
	0.0	.56888889
6	$\pm$.93246951	.17132449
	$\pm$.66120939	.36076157
	$\pm$.23861919	.46791393
7	$\pm$.96028985	.10122853
	$\pm$.74153118	.27970539
	$\pm$.40584515	.38183005
	0.0	.41795918

Tabella 2.3: Nodi e pesi delle formule di Gauss-Legendre.

mediante la formula di Gauss-Legendre con tre nodi. In tal caso utilizzando la trasformazione lineare

$$t = (x+3)/2 \quad x \in [-1,1]$$

che trasforma l'intervallo $[-1,1]$ nell'intervallo $[1,2]$ si ha:

$$I[f] = \frac{1}{2}\int_{-1}^{1} f\left(\frac{x+3}{2}\right)\,dx \simeq \frac{1}{2}\sum_{i=1}^{3} A_i f\left(\frac{x_i+3}{2}\right)$$

con i pesi A_i ed i nodi $x_i \in [-1,1]$ riportati nella Tabella 2.3. ♣

La trasformazione dell'intervallo $[-1,1]$ nel generico intervallo $[a,b]$, è la seguente:

$$t = \left[\frac{(b-a)x + a + b}{2}\right], \qquad x \in [-1,1] \tag{2.22}$$

Con tale trasformazione si ha:

$$I[f] = \int_a^b f(t)\,dt = \frac{b-a}{2}\int_{-1}^{1} f\left[\frac{(b-a)x+a+b}{2}\right]\,dx$$

e quindi

$$I[f] \simeq Q[f] = \frac{(b-a)}{2}\sum_{i=1}^{n} A_i f\left[\frac{(b-a)x_i + a + b}{2}\right]$$

2.1.5 Formule di Gauss con nodi preassegnati

Come mostrato nel paragrafo 2.1.3, le formule di Gauss, per un numero fissato di nodi, hanno il grado di precisione algebrico più alto. Pertanto tali formule sono preferite nella costruzione di software matematico. In questo paragrafo si vuole descrivere un modo per stimare l'errore delle formule di Gauss con quantità calcolabili. Nel **§4.2** del **Capitolo 4**, **Parte prima**, si è osservato che una strategia per stimare l'errore di una formula composita è basata sulla (4.15) del **§4.2.1** del **Capitolo 4**, **Parte prima**. Tale tipo di stima ha carattere generale e può essere applicata a qualunque coppia di formule di quadratura[7]:

$$(Q^{(n_1)}[f]; Q^{(n_2)}[f])$$

rispettivamente con n_1 e n_2 nodi, in cui $n_1 > n_2$. In generale, il costo computazionale per valutare $E[f]$ mediante la relazione

$$|E[f]| \simeq \alpha |Q^{(n_1)}[f] - Q^{(n_2)}[f]|$$

è $n_1 + n_2$ valutazioni di funzione.

Come è noto però, le coppie di formule più adatte per calcolare una stima dell'errore sono le formule innestate (vedi **Definizione 4.6** del **Capitolo 4**, **Parte prima**), in cui, cioè, l'insieme dei nodi di una formula contiene l'insieme dei nodi dell'altra. In tal caso infatti la complessità computazionale per la stima di $E[f]$ si riduce a n_1 valutazioni di funzione.

Purtroppo però le formule di Gauss, benché convenienti in quanto hanno grado di precisione algebrico massimo, non godono della proprietà di essere innestate e la complessità computazionale per valutare l'errore $E[f]$ è di $n_1 + n_2$ valutazioni della funzione integranda.

Tale inconveniente può essere superato se parte dei nodi sono preassegnati.

♣ **Esempio 2.6.** Sia :

$$G[f] = f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

la formula di Gauss-Legendre con due nodi. Si vuole costruire una formula con cinque nodi del tipo:

$$K[f] = B_1 f\left(\frac{-1}{\sqrt{3}}\right) + B_2 f\left(\frac{1}{\sqrt{3}}\right) + \sum_{j=1}^{3} C_j f(\xi_j)$$

determinando i nodi ξ_j, $j = 1, ..., 3$ ed i pesi B_i, $i = 1, 2$ e C_j, $j = 1, ..., 3$ in maniera che $K[f]$ abbia grado di precisione algebrico massimo. Essendo disponibili 8 parametri (8 incognite) si può ragionevolmente imporre di integrare esattamente un qualunque polinomio di grado $d = 7$ (8 condizioni relative alle funzioni di base di Π_7). Analogamente all'esempio 2.4, è necessario quindi risolvere il

[7] In questo paragrafo apporremo l'apice n al simbolo $Q[f]$ per evidenziare la dipendenza della formula di quadratura dal numero dei nodi n.

sistema non lineare:

$$\begin{cases} B_1+B_2+C_1+C_2+C_3 & = 2 \\ \frac{1}{\sqrt{3}}(-B_1+B_2)+C_1\xi_1+C_2\xi_2+C_3\xi_3 & = 0 \\ \frac{1}{3}(B_1+B_2)+C_1\xi_1^2+C_2\xi_2^2+C_3\xi_3^2 & = \frac{2}{3} \\ \frac{1}{3\sqrt{3}}(-B_1+B_2)+C_1\xi_1^3+C_2\xi_2^3+C_3\xi_3^3 & = 0 \\ \frac{1}{9}(B_1+B_2)+C_1\xi_1^4+C_2\xi_2^4+C_3\xi_3^4 & = \frac{2}{5} \\ \frac{1}{9\sqrt{3}}(-B_1+B_2)+C_1\xi_1^5+C_2\xi_2^5+C_3\xi_3^5 & = 0 \\ \frac{1}{27}(B_1+B_2)+C_1\xi_1^6+C_2\xi_2^6+C_3\xi_3^6 & = \frac{2}{7} \\ \frac{1}{27\sqrt{3}}(-B_1+B_2)+C_1\xi_1^7+C_2\xi_2^7+C_3\xi_3^7 & = 0 \end{cases}$$

ottenuto imponendo che $K[f]$ sia esatta per i monomi x^r per $r = 0, 1, ..., 7$, base di Π_7.

Per risolvere tale sistema, si osservi che, poiché l'intervallo $[-1, 1]$ è simmetrico rispetto allo 0, è ragionevole porre $\xi_1 = -\xi_3$ e $\xi_2 = 0$. Pertanto dalla seconda e quarta equazione si ha $C_1 = C_3$ e quindi anche $B_1 = B_2$. In seguito, dalla terza, quinta e settima equazione si ricava prima $\xi_1 = \xi_3 = 0.92582009977255\ldots$ e quindi $B_1 = B_2 = 0.4\overline{90}$ e $C_1 = C_3 = 0.19\overline{79}$. Infine, dalla prima equazione si ricava $C_2 = 0.6\overline{2}$. ♣

L'idea è quella di partire da una formula di Gauss-Legendre con n nodi e di costruire una formula a $2n+1$ nodi che utilizzi gli n nodi della formula di Gauss-Legendre. In generale, assegnata una formula di Gauss-Legendre con n nodi:

$$G^{(n)}[f] = \sum_{i=1}^{n} A_i f(x_i),$$

si vuole costruire una formula di quadratura con $2n+1$ nodi del tipo

$$K^{(2n+1)}[f] = \sum_{i=1}^{n} B_i f(x_i) + \sum_{j=1}^{n+1} C_j f(\xi_j). \tag{2.23}$$

Tale formula viene detta *formula di Gauss con nodi preassegnati*[8]. Nella (2.23) i nodi x_i sono assegnati e bisogna calcolare i pesi B_i e C_j ed i nodi ξ_j in modo da ottenere una formula con il grado di precisione algebrico massimo possibile. La (2.23) ha $n + 2(n+1) = 3n+2$ parametri (3n+2 incognite), ed è quindi possibile aspettarsi una formula di quadratura con grado di precisione algebrico $3n+1$ (3n+2 condizioni). A tal proposito vale il teorema seguente:

Teorema 2.1.4. *Sia $K^{(2n+1)}[f]$ del tipo (2.23). Condizione necessaria e sufficiente affinché $K^{(2n+1)}[f]$ abbia grado di precisione algebrico $3n+1$ è che sia esatta per polinomi di grado al più $2n$, e che abbia come $n+1$ nodi gli zeri ξ_j, $j = 1, ..., n+1$ di un*

[8]Nonostante in letteratura il caso più trattato sia quello esposto, una forma più generale della precedente formula è:

$$K^{(n_1+n_2)}[f] = \sum_{i=1}^{n_1} B_i f(x_i) + \sum_{j=1}^{n_2} C_j f(\xi_j)$$

.

polinomio $\omega(x) \in \Pi_{n+1}$ ortogonale a tutti i polinomi $q(x) \in \Pi_n$ rispetto alla funzione peso $r(x) = (x - x_1) \cdots (x - x_n)$, cioè tale che:

$$\int_a^b r(x)q(x)\omega(x)\,dx = 0 \qquad \forall q \in \Pi_n \tag{2.24}$$

Dimostrazione Se la formula $K^{(2n+1)}[f]$ ha grado di precisione $3n+1$ la condizione

$$K^{(2n+1)}[v] = \int_a^b v(x)\,dx \qquad \forall v \in \Pi_{2n} \tag{2.25}$$

è banalmente soddisfatta, cioè è esatta per polinomi di grado al più $2n$. Siano quindi $x_1, ..., x_n$ e $\xi_1, ..., \xi_{n+1}$ i nodi di $K^{(2n+1)}[f]$ e sia $q(x)$ un polinomio di grado al più n. Posto allora $r(x) = (x - x_1) \cdots (x - x_n)$, $\omega(x) = (x - \xi_1) \cdots (x - \xi_{n+1})$ e $p(x) = r(x)q(x)\omega(x)$ si ha che $p \in \Pi_{3n+1}$ e per tale polinomio la formula $K^{(2n+1)}[f]$ è esatta, cioè:

$$0 = E[p] = \int_a^b p(x)\,dx - \sum_{i=1}^{n} B_i p(x_i) - \sum_{j=1}^{n+1} C_j p(\xi_j) =$$

$$= \int_a^b r(x)q(x)\omega(x)\,dx - \sum_{i=1}^{n} B_i r(x_i)q(x_i)\omega(x_i) - \sum_{j=1}^{n+1} C_j r(\xi_j)q(\xi_j)\omega(\xi_j)$$

Poiché $r(x_i) = 0$ e $\omega(\xi_j) = 0$ $\forall i$ e j la (2.24) è provata.

Viceversa si supponga che la (2.24) sia verificata e che la formula $K^{(2n+1)}[f]$ sia esatta per polinomi di grado al più $2n$; sia $p(x)$ un polinomio di grado al più $3n+1$. Dividendo $p(x)$ per il polinomio $r(x)\omega(x)$ di grado $2n+1$ si ha che $p(x) = r(x)\omega(x)q(x) + v(x)$ con $v(x)$ polinomio di grado al più $2n$. Per la (2.24) risulta allora:

$$\int_a^b p(x)\,dx = \int_a^b r(x)\omega(x)q(x)\,dx + \int_a^b v(x)\,dx = \int_a^b v(x)\,dx$$

Si noti inoltre che $p(x_i) = v(x_i)$ e $p(\xi_j) = v(\xi_j)$, poiché in tali punti il polinomio $r(x)\omega(x)$ si annulla. Per la (2.25) allora si ha:

$$\int_a^b v(x)\,dx = \sum_{i=1}^{n} B_i v(x_i) + \sum_{j=1}^{n+1} C_j v(\xi_j) = \sum_{i=1}^{n} B_i p(x_i) + \sum_{j=1}^{n+1} C_j p(\xi_j)$$

e per l'arbitrarietà di $p(x)$ il teorema è dimostrato. ■

Definizione 2.1.5. (Formula di Gauss-Kronrod)
Una formula del tipo

$$K^{(2n+1)}[f] = \sum_{i=1}^{n} B_i f(x_i) + \sum_{j=1}^{n+1} C_j f(\xi_j)$$

con grado di precisione algebrico $3n+1$, costruita in base al teorema precedente è detta **formula di Gauss-Kronrod**.

Si dimostra che i pesi delle formule di Gauss-Kronrod sono tutti positivi.

Il teorema precedente determina le condizioni per cui una formula del tipo (2.23) abbia grado di precisione algebrico $3n+1$, ma nulla dice circa l'esistenza del polinomio $\omega(x)$ e quindi dei nodi ξ_j. A tal proposito si osservi che i polinomi di Stieltjes [9] $s_n(x)$ godono della proprietà (2.24) del Teorema 2.1.4. Infatti si dimostra che

$$\int_{-1}^{1} l_n(x) s_{n+1}(x) x^k \, dx = 0 \quad k = 0, ..., n.$$

dove $l_n(x)$ è il polinomio di Legendre di grado n, $s_{n+1}(x)$ è il polinomio di Stieljes di grado $n+1$. Pertanto, nella (2.23) scegliendo $r(x) = l_n$, $\omega(x) = s_{n+1}(x)$ e $q(x) = x^k$, per il Teorema 2.1.4, gli zeri del polinomio di Stieltjes $s_{n+1}(x)$ di grado $n+1$ sono i nodi ξ_j cercati e gli zeri del polinomio di Legendre $l_n(x)$ di grado n sono i nodi della formula di Gauss $G^{(n)}[f]$.

Il costo computazionale della coppia di formule:

$$(G^{(n)}[f]; K^{(2n+1)}[f]) \tag{2.26}$$

è di $2n+1$ valutazioni di funzione. Tale costo è lo stesso della formula di Gauss con $2n+1$ nodi che non permette di ottenere una stima calcolabile dell'errore. Le coppie del tipo (2.26) hanno anche lo stesso costo computazionale delle coppie di formule di Gauss $G^{(n)}[f]$ e $G^{(n+1)}[f]$ ma la formula $K^{(2n+1)}[f]$ ha precisione molto maggiore di $G^{(n+1)}[f]$. Per tale motivo le coppie (2.26) sono le più utilizzate nello sviluppo di routine per il calcolo degli integrali. Numerose routine della libreria NAg e del package `QUADPACK` sono infatti basate su tali formule.

I risultati fin qui riportati valgono quando la formula $G[f]$ è una formula di Gauss-Legendre. Essi possono comunque essere estesi anche al caso in cui la $G[f]$ sia una formula di Gauss-Jacobi. In generale comunque l'esistenza dei nodi ξ_j non è assicurata per le altre formule di Gauss riportate nella Tabella 2.2.

Il metodo descritto, sviluppato inizialmente da Kronrod, è stato successivamente generalizzato da Patterson, che, a partire da una formula di Gauss-Legendre con n nodi, ha sviluppato una famiglia di formule innestate

$$\mathcal{K} = \{P^{(i)}[f]\}_{i=2^k(n+1)-1} \quad k = 1, 2, ...$$

con grado di precisione algebrico $3 \cdot 2^{k-1}(n+1) - 2$, ciascuna delle quali costruita in base al Teorema 2.1.4. Tale famiglia di formule è alla base della routine `QNG` del package `QUADPACK` e della routine `D01AHF` della libreria NAg.

[9] Si veda il §**C.3**.

2.2 Errori e criteri di convergenza per le formule di quadratura

2.2.1 Errore di discretizzazione

Ad ogni formula di quadratura $Q[f]$ è associato un errore $E[f]$, chiamato errore di discretizzazione, tale che

$$E[f] = I[f] - Q[f].$$

In questo paragrafo si vuole determinare un'espressione di $E[f]$. Uno strumento fondamentale in tale senso è il seguente:

Teorema 2.2.1. [Teorema di Peano]
Sia $E[f]$ il funzionale lineare del tipo [10]

$$E[f] = \int_a^b f(x)\,dx - \sum_{i=1}^{n} A_i f(x_i)$$

Se risulta $E[p] = 0 \quad \forall p(x) \in \Pi_q$ allora si ha che

$$E[f] = \int_a^b f^{(q+1)}(t) K(t)\,dt \qquad \forall f \in C^{q+1}([a,b]) \tag{2.27}$$

dove la funzione

$$K(t) = \frac{1}{q!} E_x[(x-t)_+^q] \quad con \quad (x-t)_+^q = \begin{cases} (x-t)^q & x \geq t \\ 0 & x < t \end{cases},$$

è detta kernel di Peano *e con il simbolo E_x si indica il funzionale E applicato alla funzione $(x-t)_+^q$ considerata come funzione della sola variabile x.*

Dimostrazione Sia $f \in C^{q+1}([a,b])$. Si consideri la formula di Taylor della funzione f di punto iniziale a:

$$f(x) = f(a) + f'(a)(x-a) + \ldots + \frac{f^{(q)}(a)(x-a)^q}{q!} + \frac{1}{q!}\int_a^x f^{(q+1)}(t)(x-t)^q dt. \tag{2.28}$$

L'ultimo addendo di tale formula si può anche esprimere come:

$$\frac{1}{q!}\int_a^b f^{(q+1)}(t)(x-t)_+^q dt.$$

[10] Si osservi che tale teorema è valido in generale per funzionali lineari del tipo

$$\Phi[f] = \int_a^b \left(\sum_{j=0}^{m} \alpha_j(x) f^{(j)}(x) \right) dx + \sum_{j_0=1}^{n_0} A_{j_0} f(a_{j_0}) + \ldots + \sum_{j_m=1}^{n_m} A_{j_m} f^{(m)}(a_{j_m})$$

con $\alpha_j(x)$ funzioni continue a tratti su $[a,b]$, $f \in C^{m+1}([a,b])$, $a_{j_i} \in [a,b]$, $A_{jk} \in \Re$ di cui il funzionale $E[f]$ è un caso particolare ottenuto ponendo $m = 0$ e $\alpha_0(x) \equiv 1$.

Applicando il funzionale $E[f]$ ad entrambi i membri della (2.28) e ricordando che esso è nullo per tutti i polinomi di grado al più q si ha:

$$E[f] = \frac{1}{q!} E_x \left[\int_a^b f^{(q+1)}(t)(x-t)_+^q dt \right]$$

Per la continuità della $f^{(q+1)}(t)$ e per la linearità del funzionale $E[f]$ è possibile invertire l'ordine dell'integrale e del funzionale, cioè:

$$E[f] = \frac{1}{q!} \int_a^b f^{(q+1)}(t) E_x[(x-t)_+^q] dt,$$

da cui la tesi. ■

Conseguenze del Teorema di Peano sono:

Corollario 2.2.1. *Nelle ipotesi del teorema di Peano risulta:*

$$|E[f]| \leq \max_{[a,b]} |f^{(q+1)}(t)| \int_a^b |K(t)| \, dt.$$

Dimostrazione Dalla (2.27) si ha:

$$|E[f]| = \left| \int_a^b f^{(q+1)}(t) K(t) \, dt \right| \leq \int_a^b |f^{(q+1)}(t)| \, |K(t)| \, dt$$

da cui segue la tesi. ■

Corollario 2.2.2. *Nelle ipotesi del Teorema di Peano, se $K(t)$ non cambia di segno in $[a, b]$ risulta:*

$$E[f] = \frac{f^{(q+1)}(\xi)}{(q+1)!} E[t^{q+1}] \qquad \xi \in [a, b].$$

Dimostrazione Per il *teorema del valor medio degli integrali* deve esistere un punto $\xi \in [a, b]$ tale che

$$E[f] = \int_a^b f^{(q+1)}(t) K(t) \, dt = f^{(q+1)}(\xi) \int_a^b K(t) \, dt \tag{2.29}$$

Ponendo $f(t) = t^{q+1}$ nella (2.27) si ha che

$$E[t^{q+1}] = (q+1)! \int_a^b K(t) \, dt \quad \text{cioè} \quad \int_a^b K(t) \, dt = \frac{E[t^{q+1}]}{(q+1)!}$$

da cui, sostituendo nella (2.29) segue la tesi. ■

♣ **Esempio 2.7.** Si calcoli l'errore di discretizzazione della formula trapezoidale

$$T[f] = \frac{b-a}{2}\left(f(a) + f(b)\right).$$

con $f \in C^2([a,b])$.
Cominciamo col verificare che il kernel $K(t)$ non cambia segno nell'intervallo $[a,b]$. Si ha quindi:

$$K(t) = \int_a^b (x-t)_+\,dx - \frac{(b-a)}{2}[(a-t)_+ + (b-t)_+] =$$

$$= \int_t^b (x-t)\,dx - \frac{(b-a)}{2}[(b-t)] = \frac{1}{2}(a-t)(b-t) \le 0 \qquad t \in [a,b].$$

È allora possibile applicare il Corollario 2.2.2 del Teorema di Peano:

$$E[f] = \frac{f''(\xi)}{2}E[x^2] \qquad \xi \in [a,b]. \tag{2.30}$$

Poichè

$$E[x^2] = \int_a^b x^2\,dx - \frac{b-a}{2}(a^2+b^2) = -\frac{1}{6}(b-a)^3,$$

da cui, sostituendo nella (2.30), si ricava

$$E[f] = -\frac{(b-a)^3}{12}f''(\xi) \qquad \xi \in [a,b].$$

♣

Si osservi che, per una formula di quadratura generica, non c'è garanzia che il kernel $K(t)$ sia di segno costante in $[a,b]$. Per le formule di Newton-Cotes si può dimostrare che risulta $K(t) \le 0 \quad \forall t \in [a,b]$. Per tali formule è quindi possibile applicare il Corollario 2.2.2 del teorema di Peano.

♣ **Esempio 2.8.** Si calcoli l'errore di discretizzazione della formula di Simpson

$$S[f] = \frac{(b-a)}{2}\left[\frac{f(a)}{3} + \frac{4}{3}f\left(\frac{(a+b)}{2}\right) + \frac{f(b)}{3}\right]$$

con $f \in C^4([a,b])$. Tale formula è esatta per tutti i polinomi $p(x) \in \Pi_2$, per cui dal Corollario 2.2.2 del Teorema di Peano (con $q = 2$) si ha:

$$E[f] = \frac{f'''(\xi)}{6}E[x^3] \qquad \xi \in [a,b].$$

Si osservi che

$$E[x^3] = \int_a^b x^3\,dx - \frac{(b-a)}{2}\left[\frac{a^3}{3} + \frac{4}{3}\left(\frac{a+b}{2}\right)^3 + \frac{b^3}{3}\right]$$

e con facili calcoli risulta:

$$E[x^3] = \frac{b-a}{4}\left[(b^2+a^2)(a+b) - (a^3+b^3+a^2b+ab^2)\right] = 0$$

Da ciò si ricava che la formula di Simpson è esatta anche per tutti i polinomi $p(x) \in \Pi_3$. Riapplicando quindi il Corollario 2.2.2 del Teorema di Peano (questa volta con $q = 3$), se la funzione è dotata anche di derivata quarta, si ha:

$$E[f] = \frac{f^{(4)}(\xi)}{24} E[x^4] \quad \xi \in [a, b] \tag{2.31}$$

e poiché risulta:

$$E[x^4] = -\frac{4}{15}\left(\frac{b-a}{2}\right)^5$$

sostituendo nella (2.31) si ha che l'errore di discretizzazione associato alla formula di Simpson è:

$$E[f] = -\frac{f^{(4)}(\xi)}{90}\left(\frac{b-a}{2}\right)^5 \qquad \xi \in [a, b].$$

♣

Risultati analoghi a quelli dell'esempio precedente si hanno per altre formule di Newton-Cotes con un numero di nodi n dispari e con $f \in C^{n+1}([a, b])$, cioè vale il:

Teorema 2.2.2. *Se $f \in C^{n+1}([a, b])$ e se il numero dei nodi è dispari, $n = 2k + 1$ $(k \geq 1)$, le formule di Newton-Cotes hanno grado di precisone algebrico n e l'espressione dell'errore associato risulta essere:*

$$E[f] = \frac{f^{(n+1)}(\xi)}{(n+1)!} E[x^{n+1}] \qquad \xi \in [a, b].$$

Dimostrazione Dal teorema di Peano e per la linearità del funzionale $E[f]$, integrando per parti si ha:

$$\begin{aligned} E[f] &= \frac{1}{(n-1)!} E_x\left[\int_a^b f^{(n)}(t)(x-t)_+^{n-1}\, dt\right] \\ &= \frac{1}{(n-1)!} E_x\left[-f^{(n)}(a)\frac{(x-a)^n}{n} + \int_a^b f^{(n+1)}(t)\frac{(x-t)_+^n}{n}\, dt\right] \\ &= -\frac{f^{(n)}(a)}{n!} E_x[(x-a)^n] + \left[\int_a^b f^{(n+1)}(t)K(t)\, dt\right]. \end{aligned}$$

Ma per la (2.8) si ha:

$$\begin{aligned} E_x[(x-a)^n] &= \int_a^b (x-a)^n\, dx - \sum_{i=1}^n (x_i - a)^n \int_a^b \frac{\omega(x)}{(x-x_i)\omega'(x_i)}\, dx \\ &= \int_a^b (x-a)^n dx - \int_a^b L_{n-1}(x)\, dx \\ &= \int_a^b \left[(x-a)^n - L_{n-1}(x)\right]\, dx \end{aligned}$$

dove, in questo caso, $L_{n-1}(x)$ è il polinomio di grado $n-1$ interpolante nel senso di Lagrange la funzione $(x-a)^n$ nei nodi x_i con $i = 1, ..., n$. Poiché il resto di ordine n del polinomio interpolante di Lagrange, della funzione $(x-a)^n$ è [11]:

$$(x-a)^n - L_{n-1}(x) = \frac{n!}{n!}\omega(x)$$

risulta:

$$E_x[(x-a)^n] = \int_a^b \omega(x)\, dx$$

Se il numero di nodi $n = 2k+1$ è dispari si può facilmente verificare che $\omega(x_k + t) = -\omega(x_k - t)$ cioè la funzione $\omega(x)$ è una funzione dispari rispetto al nodo x_k che coincide con il punto medio dell'intervallo $[a, b]$. Si ha quindi che

$$\int_a^b \omega(x)\, dx = 0,$$

da cui la tesi. ■

Si può, allora, affermare che una formula di quadratura di Newton-Cotes con un numero di nodi dispari, $n = 2k + 1$, ha lo stesso grado di precisione algebrico della successiva formula con un numero di nodi pari, $n = 2k + 2$. Nella Tabella 2.4 sono riportate le espressioni degli errori di alcune formule di Newton-Cotes. Si noti, ad esempio, come l'errore della formula di Simpson abbia le stesse caratteristiche dell'errore della formula dei '3/8'.
Nella stessa Tabella, dove nell'espressione dell'errore compare il termine $f^{(k)}(\xi)$, è tacitamente assunto che $f \in C^k([a, b])$. Ovviamente è possibile applicare il Teorema di Peano ed i suoi Corollari anche alle formule di Gauss.

♣ **Esempio 2.9.** Si calcoli l'errore della formula di Gauss-Legendre con $n = 2$ nodi. In tal caso la formula ha grado di precisione algebrico $q = 3$ ed il kernel di Peano è (vedi esempio 2.4):

$$K(t) = \frac{1}{3!}E_x\left[(x-t)_+^3\right] = \frac{1}{6}\left(\int_{-1}^{1}(x-t)_+^3\, dx - \left[\left(-\frac{1}{\sqrt{3}} - t\right)^3 + \left(\frac{1}{\sqrt{3}} - t\right)^3\right]\right)$$

da cui

$$K(t) = \frac{(1+t)^4}{24} \geq 0 \quad \forall t \in [-1, 1].$$

[11] Sussiste infatti il seguente:

Teorema 2.2.3. *Sia $f \in C^n([a, b])$ e $p \in \Pi_{n-1}$ il polinomio di Lagrange interpolante la funzione $f(x)$ in n nodi distinti x_i per $i = 1, ..., n$. Per ogni $x \in [a, b]$ esiste un punto $\xi \in (a, b)$ tale che*

$$f(x) - p(x) = \frac{1}{n!}f^{(n)}(\xi)\omega(x)\,.$$

La dimostrazione del teorema è nel **Capitolo 8**, **§8.3** (**Teorema 8.3.5**).

Formula	Errore
2 nodi (*Trapezoidale*)	$-\frac{f''(\xi)}{12}(b-a)^3$
3 nodi (*Simpson*)	$-\frac{f^{(4)}(\xi)}{90}\left(\frac{b-a}{2}\right)^5$
4 nodi (*dei '3/8'*)	$-\frac{3f^{(4)}(\xi)}{80}\left(\frac{b-a}{3}\right)^5$
5 nodi (*Boole*)	$-\frac{8f^{(6)}(\xi)}{945}\left(\frac{b-a}{4}\right)^7$
6 nodi	$-\frac{275f^{(6)}(\xi)}{12096}\left(\frac{b-a}{5}\right)^7$
7 nodi	$-\frac{9f^{(8)}(\xi)}{1400}\left(\frac{b-a}{6}\right)^9$
8 nodi	$-\frac{8183f^{(8)}(\xi)}{518400}\left(\frac{b-a}{7}\right)^9$
9 nodi	$-\frac{2368f^{(10)}(\xi)}{467775}\left(\frac{b-a}{8}\right)^{11}$
10 nodi	$-\frac{673175f^{(10)}(\xi)}{163459296}\left(\frac{b-a}{9}\right)^{11}$

Tabella 2.4: Errori delle formule di Newton-Cotes con $n \leq 10$.

Applicando il Corollario 2.2.2 del Teorema di Peano si ha allora

$$E[f] = \frac{f^{(4)}(\xi)}{24} E[x^4] \tag{2.32}$$

La quantità $E[x^4]$ è data da:

$$E[x^4] = \int_{-1}^{1} x^4\,dx - \left[\left(-\frac{1}{\sqrt{3}}\right)^4 + \left(\frac{1}{\sqrt{3}}\right)^4\right] = \frac{2}{5} - \frac{2}{9} = \frac{8}{45}$$

da cui, sostituendo nella (2.32), si ricava

$$E[f] = \frac{8}{45}\frac{f^{(4)}(\xi)}{24} = \frac{f^{(4)}(\xi)}{135} \quad \forall f \in C^4([-1,1])$$

♣

Nella Tabella 2.5 sono riportati gli errori di discretizzazione di alcune formule di Gauss-Legendre nell'intervallo $[-1,1]$.

Formula	Errore
$n=2$	$f^{(4)}(\xi)/135$
$n=3$	$f^{(6)}(\xi)/91854$
$n=4$	$f^{(8)}(\xi)/3472875$

Tabella 2.5: Errori delle formule di Gauss-Legendre in $[-1,1]$.

Più in generale, utilizzando la trasformazione (2.22), è possibile determinare l'errore delle formule di Gauss in un generico intervallo $[a,b]$ (Tabella 2.6).

Formula	Errore
$n=2$	$(b-a)^5 f^{(4)}(\xi)/4320$
$n=3$	$(b-a)^7 f^{(6)}(\xi)/11757312$
$n=4$	$(b-a)^9 f^{(8)}(\xi)/1778112000$

Tabella 2.6: Errori delle formule di Gauss-Legendre in $[a,b]$.

2.2.2 Errore delle formule composite e stima calcolabile dell'errore

Dalla Tabella 2.6 si può notare che l'errore dipende dall'ampiezza dell'intervallo di integrazione $[a,b]$; un modo per ridurre l'errore consiste quindi, come mostrato nel paragrafo **§4.2.1** del **Capitolo 4**, **Parte prima**, nel dividere l'intervallo di integrazione $[a,b]$ in

m sottointervalli di uguale ampiezza $\tau = (b-a)/m$ in ognuno dei quali si applica una fissata formula di quadratura (formule composite). Per valutare l'errore di una formula composita è sufficiente applicare il teorema di Peano in ogni sottointervallo $[t_{j-1}, t_j]$ dell'intervallo $[a, b]$.

♣ **Esempio 2.10.** Si calcoli l'errore della formula trapezoidale composita $T_m[f]$ per $f \in C^2([a,b])$. In tal caso applicando il Corollario 2.2.2 del Teorema di Peano al j-mo sottointervallo $[t_{j-1}, t_j]$ di ampiezza $\tau = t_j - t_{j-1} = (b-a)/m$ si ha che l'errore in tale sottointervallo è:

$$E^{(j)}[f] = -\frac{(b-a)^3}{12m^3} f''(\xi_j) \qquad \xi_j \in [t_{j-1}, t_j],$$

da cui, sommando sull'indice j dei sottointervalli, l'errore su tutto l'intervallo $[a, b]$ è:

$$E[f] = \sum_{j=1}^{m} E^{(j)}[f] = -\frac{(b-a)^3}{12m^3} \sum_{j=1}^{m} f''(\xi_j)$$

Poiché si ha

$$\min_{\xi\in[a,b]} f''(\xi) \le \min_j f''(\xi_j) \le \frac{1}{m}\sum_{j=1}^{m} f''(\xi_j) \le \max_j f''(\xi_j) \le \max_{\xi\in[a,b]} f''(\xi)$$

per la continuità di $f''(x)$ nell'intervallo $[a, b]$ esiste un punto $\xi \in [a, b]$ tale che

$$f''(\xi) = \frac{1}{m}\sum_{j=1}^{m} f''(\xi_j),$$

da cui si ha:

$$E_m[f] = -\frac{(b-a)^3}{12m^2} f''(\xi).$$

♣

Per ricavare gli errori delle formule di Newton-Cotes composite $Q_m[f]$, si noti che $Q[f]$ nell'intervallo $[t_{j-1}, t_j]$ di ampiezza $\tau = (b-a)/m$ ha errore

$$E^{(j)}[f] = \int_{t_{j-1}}^{t_j} f(x)\,dx - h\sum_{i=1}^{n} B_i f(x_i) \qquad h = \frac{(b-a)}{m(n-1)}$$

con $E^{(j)}[f]$ opportunamente scelto nella Tabella 2.4. Per ottenere l'errore di $E_m[f]$ su tutto $[a, b]$ è sufficiente sommare il contributo degli errori $E^{(j)}[f]$, cioè:

$$E_m[f] = \sum_{j=1}^{m} E^{(j)}[f]$$

Formula base	Errore
2 nodi (*Trapezoidale*)	$-\frac{f''(\xi)}{12}\frac{(b-a)^3}{m^2}$
3 nodi (*Simpson*)	$-\frac{f^{(4)}(\xi)}{180}\frac{(b-a)^5}{m^4}$
4 nodi (*dei '3/8'*)	$-\frac{f^{(4)}(\xi)}{80}\frac{(b-a)^5}{m^4}$
5 nodi (*Boole*)	$-\frac{2f^{(6)}(\xi)}{945}\frac{(b-a)^7}{m^6}$
6 nodi	$-\frac{55f^{(6)}(\xi)}{12096}\frac{(b-a)^7}{m^6}$
7 nodi	$-\frac{3f^{(8)}(\xi)}{2800}\frac{(b-a)^9}{m^8}$
8 nodi	$-\frac{1169f^{(8)}(\xi)}{518400}\frac{(b-a)^9}{m^8}$
9 nodi	$-\frac{296f^{(10)}(\xi)}{46775}\frac{(b-a)^{11}}{m^{10}}$
10 nodi	$-\frac{673175f^{(10)}(\xi)}{1471133664}\frac{(b-a)^{11}}{m^{10}}$

Tabella 2.7: Errori delle formule di Newton-Cotes composite per $n \leq 10$.

Procedendo in maniera analoga al precedente esempio 2.10 si hanno le espressioni degli errori delle formule di Newton-Cotes composite riportate nella Tabella 2.7.

Nello sviluppo di software matematico per la quadratura è rilevante la determinazione di una stima calcolabile dell'errore di discretizzazione, basata cioè solo su quantità calcolabili. Il Corollario 2.2.2 del Teorema di Peano fornisce uno strumento generale.

♣ **Esempio 2.11.** Si determini una stima calcolabile dell'errore di discretizzazione per la formula di Simpson composita per una funzione $f \in C^4([a,b])$. A tale scopo, posto $h = (b-a)/2$, siano:

$$Q[f] = \frac{(b-a)}{6}[f(a) + 4f(c) + f(b)]$$

$$Q_2[f] = \frac{(b-a)}{12}\left[f(a) + 4f\left(a + \frac{h}{2}\right) + 2f(c) + 4f\left(a + 3\frac{h}{2}\right) + f(b)\right]$$

rispettivamente la formula di Simpson sull'intervallo $[a,b]$ e la sua composita negli intervalli $[a,c]$ e $[c,b]$ con c punto medio di $[a,b]$. Al fine di determinare una stima per

$$|E_2[f]| = |I[f] - Q_2[f]|$$

si noti che, dal Corollario 2.2.2 del Teorema di Peano si ricava:

$$E[f] = -\frac{f^{(4)}(\xi)}{90}h^5, \qquad E_2[f] = -\frac{f^{(4)}(\xi')}{90}\left(\frac{h}{2}\right)^5 - \frac{f^{(4)}(\xi'')}{90}\left(\frac{h}{2}\right)^5 \tag{2.33}$$

e dalle formule di Taylor di punto iniziale ξ della funzione $f^{(4)}(x)$:

$$f^{(4)}(\xi') = f^{(4)}(\xi) + \mathcal{O}(h), \qquad f^{(4)}(\xi'') = f^{(4)}(\xi) + \mathcal{O}(h); \tag{2.34}$$

sostituendo, quindi, le (2.34) nella (2.33), si ha:

$$E_2[f] = -2\frac{\left[f^{(4)}(\xi) + \mathcal{O}(h)\right]}{90}\left(\frac{h}{2}\right)^5 = -2\frac{f^{(4)}(\xi)}{90 \cdot 32}h^5 + \mathcal{O}\left(h^6\right) =$$

$$= \frac{E[f]}{16} + \mathcal{O}\left(h^6\right).$$

Procedendo poi come mostrato nell'**esempio 4.6** del **§4.2.1**, **Capitolo 4**, **Parte prima**, si perviene alla stima calcolabile dell'errore $E_2[f]$:

$$|E_2[f]| \simeq \frac{|Q[f] - Q_2[f]|}{15} \tag{2.35}$$

♣

Per una coppia di formule di quadratura $(Q'[f], Q''[f])$ dove $Q''[f]$ è "in qualche senso" più accurata di $Q'[f]$, è possibile utilizzare come stima dell'errore un' opportuna funzione Ψ della differenza tra $Q'[f]$ e $Q''[f]$, cioè:

$$E[f] \simeq \Psi\left[\ |Q'[f] - Q''[f]|\ \right]$$

con Ψ funzione crescente che dipende dalle particolari formule di quadratura scelte e da considerazioni di carattere sperimentale [12].

2.2.3 Criteri di convergenza delle formule di quadratura

Abbiamo già dimostrato, mediante il **Teorema 4.1** del **§4.2.1**, **Capitolo 4**, **Parte prima**, la convergenza della formula trapezoidale composita costruita su m sottointervalli di uguale ampiezza $\tau = (b-a)/m$. Più in generale, sussiste il seguente:

Teorema 2.2.4. *Sia f una funzione integrabile nell'intervallo $[a,b]$ e sia*

$$Q[f] = \sum_{i=1}^{n} A_i f(x_i) \qquad x_i \in [-1,1] \tag{2.36}$$

una formula di quadratura tale che:

$$Q[1] = \sum_{i=1}^{n} A_i = \int_{-1}^{1} dx = 2 \tag{2.37}$$

Partizionato l'intervallo $[a,b]$ in m sottointervalli $[t_{j-1}, t_j]$ $j = 1, ..., m$, sia $Q_m[f]$ la formula composita ottenuta applicando la (2.36) negli m sottointervalli. Per ogni fissato m, si indichi con τ_m la massima ampiezza dei sottointervalli, cioè

$$\tau_m = \max_{j=1,\dots,m} (t_j - t_{j-1})\,.$$

Se

$$\lim_{m\to\infty} \tau_m = 0 \tag{2.38}$$

allora si ha che[13]

$$\lim_{m\to\infty} Q_m[f] = I[f] \Longleftrightarrow \lim_{m\to\infty} E_m[f] = 0$$

Dimostrazione Data $Q[f]$ del tipo (2.36), dividendo $[a,b]$ in m sottointervalli $[t_{j-1}, t_j]$ per la proprietà additiva degli integrali, si ha:

$$I[f] = \int_a^b f(x)\,dx = \sum_{j=1}^{m} \int_{t_{j-1}}^{t_j} f(x)\,dx \tag{2.39}$$

[12] Ad esempio alcune routine di `QUADPACK`, relativamente alla coppia di formule di Gauss-Kronrod $(G^{(7)}[f], K^{(15)}[f])$ rispettivamente con 7 e 15 nodi, utilizzano come stima calcolabile dell'errore la quantità:

$$E[f] = \left(200|G^{(7)}[f] - K^{(15)}[f]|\right)^{1.5}.$$

[13] In particolare l'ipotesi (2.38) è banalmente soddisfatta nel caso di intervalli di uguale ampiezza $\tau = (b-a)/m$.

Mediante la trasformazione

$$x = \left[\frac{(t_j - t_{j-1})s + t_j + t_{j-1}}{2}\right]$$

dell'intervallo $[-1, 1]$ nell'intervallo $[t_{j-1}, t_j]$ si ha

$$\int_{t_{j-1}}^{t_j} f(x)\,dx = \frac{(t_j - t_{j-1})}{2} \int_{-1}^{1} f\left[\frac{(t_j - t_{j-1})s + t_j + t_{j-1}}{2}\right] ds \tag{2.40}$$

Posto allora

$$x_i^{(j)} = \frac{(t_j - t_{j-1})s_i + t_j + t_{j-1}}{2}$$

applicando la (2.36) all'integrale al secondo membro della (2.40) si ha:

$$\int_{-1}^{1} f\left[\frac{(t_j - t_{j-1})s + t_j + t_{j-1}}{2}\right] ds = \sum_{i=1}^{n} A_i f(x_i^{(j)}) + E[f] \quad \text{con} \quad x_i^{(j)} \in [t_{j-1}, t_j]$$

Sostituendo tale espressione nella (2.40), si ha

$$\int_{t_{j-1}}^{t_j} f(x)\,dx = \frac{(t_j - t_{j-1})}{2}\left[\sum_{i=1}^{n} A_i f(x_i^{(j)}) + E[f]\right]$$

e quindi la formula di quadratura composita $Q_m[f]$ sull'intero intervallo $[a, b]$ è

$$Q_m[f] = \sum_{j=1}^{m} \frac{(t_j - t_{j-1})}{2} \sum_{i=1}^{n} A_i f(x_i^{(j)}) = \frac{1}{2} \sum_{i=1}^{n} A_i \sum_{j=1}^{m} (t_j - t_{j-1}) f(x_i^{(j)})$$

Nell'ultima espressione, per ogni fissato indice i, la somma più interna è una somma di Riemann per cui, per l'ipotesi (2.38), si ha

$$\lim_{m\to\infty} E_m[f] = \lim_{m\to\infty} [I[f] - Q_m[f]] = I[f] - \frac{1}{2} I[f] \sum_{i=1}^{n} A_i$$

Per l'ipotesi (2.37), si ha quindi la tesi. ■

♣ **Esempio 2.12.** La Figura 2.1 mostra l'andamento degli errori associati alle successioni $T_m[f]$ e $S_m[f]$, rispettivamente formule trapezoidali e di Simpson composite, al crescere del numero di sottointervalli m, per l'integrale:

$$\int_0^{\pi} e^x\, cos(x)\,dx = -\frac{(e^{\pi} + 1)}{2} = -12.0703463... \tag{2.41}$$

ottenuti utilizzando la doppia precisione di un calcolatore con aritmetica standard IEEE.
Dalla figura si può notare come l'errore si riduca quando il numero dei sottointervalli m aumenta.
Nella Tabella 2.8 è invece riportato il rapporto degli errori delle formule trapezoidali e di Simpson composite per il problema (2.41) quando il numero dei sottointervalli m raddoppia. Si può osservare che per la formula trapezoidale composita l'errore $E_m[f]$ viene diviso approssimativamente per 4 quando m raddoppia (cioè quando si dimezza h), mentre per la formula di Simpson composita l'errore $E_m[f]$ viene diviso approssimativamente per 16 raddoppiando il numero di nodi. Tale fenomeno trova giustificazione dalle espressioni degli errori riportati nella Tabella 2.7 , dalla quale si ricava il fattore di riduzione dell'errore delle varie formule composite, quando si raddoppia m ($1/m^2$ per la formula trapezoidale composita, $1/m^4$ per la formula di Simpson composita.)

♣

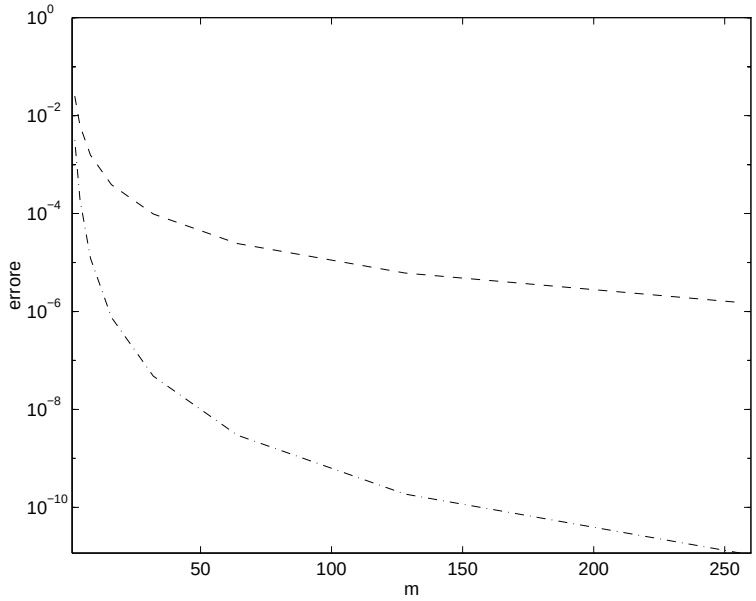

Figura 2.1: Confronto tra l'errore della formula composita di Simpson (–) e la formula trapezoidale (-.).

Trapezoidale composita		Simpson composita	
m	$\lvert E_m[f]/E_{m/2}[f]\rvert$	m	$\lvert E_m[f]/E_{m/2}[f]\rvert$
2	4.2693258D+0	2	13.915154D+0
4	4.2024268D+0	4	15.537889D+0
8	4.0590479D+0	8	15.888486D+0
16	4.0152590D+0	16	15.972376D+0
32	4.0038452D+0	32	15.993104D+0
64	4.0009632D+0	64	15.998199D+0
128	4.0002409D+0	128	15.998005D+0
256	4.0000602D+0	256	15.986632D+0

Tabella 2.8: Fattore di riduzione dell'errore della formula trapezoidale composita e della formula di Simpson composita.

Diamo ora la seguente:

Definizione 2.2.1. (Ordine di convergenza)
Una famiglia di formule di Newton-Cotes composite $\{Q_m[f]\}$ *ha* **ordine di convergenza** q *se e solo se*

$$E_m[f] = \mathcal{O}\left(\frac{1}{m^q}\right) \quad \Leftrightarrow \quad E_m[f] = \mathcal{O}(h^q) \qquad h = \frac{b-a}{m(n-1)}$$

Da tale definizione e dalla Tabella 2.7 è possibile dire che la formula trapezoidale composita ha ordine di convergenza 2 mentre la formula di Simpson composita ha ordine di convergenza 4.

Il Teorema 2.2.4 garantisce la convergenza delle formule composite al crescere del numero di sottointervalli m, facendo tendere a zero la distanza fra i nodi fissato l'intervallo di integrazione $[a, b]$. Il teorema che segue fornisce, invece, un risultato relativo alle proprietà di convergenza al crescere del numero di nodi n, delle formule di quadratura esatte per lo spazio dei polinomi.

Teorema 2.2.5. *Siano assegnate una funzione* $f \in C([a, b])$ *ed una famiglia di formule del tipo:*

$$Q_n[f] = \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) \qquad n = 1, 2, ...$$

ognuna delle quali esatta per lo spazio dei polinomi Π_{n-1}*. Se la somma dei moduli dei pesi è limitata da una costante* M *indipendente dal numero di nodi* n*, cioè:*

$$\sum_{i=1}^{n} |A_i^{(n)}| \leq M \qquad \forall n = 1, 2, ... \tag{2.42}$$

si ha:

$$\lim_{n\to\infty} E[f] = \lim_{n\to\infty} \left[\int_a^b f(x)\, dx - \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) \right] = 0$$

Dimostrazione Dimostriamo che per ogni ε esiste un indice ν tale che $\forall n \geq \nu$ l'errore $E[f] \leq K\varepsilon$ con K costante indipendente da m. Fissato quindi un $\varepsilon > 0$, poichè la funzione f è continua, per il teorema di Weierstrass è possibile determinare un ν ed un polinomio $p(x) \in \Pi_\nu$ tale che

$$|f(x) - p(x)| < \varepsilon \quad \forall x \in [a, b] \tag{2.43}$$

L'errore $E[f]$ è quindi

$$\begin{aligned} E[f] &= \int_a^b f(x)\, dx - \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) \\ &= \int_a^b f(x)\, dx - \int_a^b p(x)\, dx + \int_a^b p(x)\, dx - \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) \end{aligned} \tag{2.44}$$

Per ipotesi si ha che $\forall n \geq \nu$

$$\int_a^b p(x)\,dx = \sum_{i=1}^{n} A_i^{(n)} p(x_i^{(n)}) \qquad p \in \Pi_n$$

da cui, sostituendo nella (2.44), si ha:

$$E[f] = \int_a^b f(x)\,dx - \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) = \int_a^b [f(x) - p(x)]\,dx + \sum_{i=1}^{n} A_i^{(n)} \left[p(x_i^{(n)}) - f(x_i^{(n)})\right]$$

e quindi per la (2.43):

$$|E[f]| = \left| \int_a^b f(x)\,dx - \sum_{i=1}^{n} A_i^{(n)} f(x_i^{(n)}) \right| \leq \varepsilon(b-a) + \varepsilon \sum_{i=1}^{n} |A_i^{(n)}| = \varepsilon \left((b-a) + \sum_{i=1}^{n} |A_i^{(n)}| \right)$$

Per l'ipotesi (2.42), posto $K = [(b-a) + M]$, si ha quindi la tesi. ■

Si può affermare quindi che, se la funzione integranda è continua, per il Teorema 2.2.5 e per la (2.7), le famiglie delle formule di Gauss sono convergenti. Nulla si può dire, invece, circa la convergenza delle formule di Newton-Cotes nell'ipotesi di sola continuità della f, in quanto i pesi di tali formule sono di segno alterno per $n \geq 11$ e non verificano l'ipotesi (2.42) del Teorema 2.2.5.

♣ **Esempio 2.13.** Se si calcola il valore dell'integrale:

$$\int_{-\sqrt{3}}^{\sqrt{3}} \frac{4}{1+x^2} dx = \frac{8}{3}\pi$$

mediante la successione di formule di Newton-Cotes ottenute al variare di $n = 2, 3, 4, ...$, tale successione è divergente, mentre nel caso dell'integrale:

$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$

si ottengono i valori in Tabella 2.9.

In tal caso infatti, al crescere del numero di nodi n, l'errore delle formule di Newton-Cotes tende a zero, mentre ciò non accade nel primo caso.

Si noti che i due problemi hanno la stessa funzione integranda che è di classe C^∞ e differiscono solo per l'intervallo di integrazione. Per comprendere il perché di tale fenomeno si osservi che, essendo formule esatte per lo spazio dei polinomi, le formule di Newton-Cotes sono in particolare esatte anche per il polinomio di Lagrange $L_{n-1}(f, x)$ interpolante la funzione $f(x)$ nei nodi equidistanziati x_i, $i = 1, \ldots, n$.

È possibile verificare che tali formule sono convergenti per tutte le funzioni per cui valgono i criteri di convergenza uniforme delle famiglie dei polinomi interpolanti di Lagrange costruiti su nodi equidistanziati[14]. In tal caso è infatti possibile passare al limite sotto il segno di integrale, cioè:

$$\lim_{n\to\infty} \sum_{i=1}^{n} A_i f(x_i) = \lim_{n\to\infty} \sum_{i=1}^{n} A_i L_n(f, x_i) =$$

$$\lim_{n\to\infty} \int_a^b L_n(f, x)\,dx = \int_a^b \lim_{n\to\infty} L_n(f, x)\,dx = \int_a^b f(x)\,dx$$

[14]Sussiste infatti il seguente

n	$Q[f]$
2	$.300000 \times 10^1$
3	$.313333 \times 10^1$
4	$.313846 \times 10^1$
5	$.314211 \times 10^1$
6	$.314187 \times 10^1$
7	$.314157 \times 10^1$
8	$.314157 \times 10^1$
9	$.314159 \times 10^1$
10	$.314159 \times 10^1$

Tabella 2.9: Successione dei valori ottenuti dalle formule di Newton-Cotes per $n = 2, 3, ..., 10$.

Nell'esempio in esame, la funzione definita nel campo complesso $f(z) = 4/(1+z^2)$ (di cui la $f(x)$ è la restrizione al campo reale), ha singolarità in corrispondenza dell'unità immaginaria e del suo opposto $\pm i$. Tali singolarità, nel secondo caso cadono all'esterno dell'ellisse di centro 1/2 e semiassi 5/8 e 3/8 (Figura 2.2), mentre nel primo caso cadono all'interno dell'ellisse di centro 0 e con semiassi $5\sqrt{3}/4$ e $3\sqrt{3}/4 = 1.299...$.

♣

2.3 Condizionamento di una formula di quadratura

Poiché i dati di una formula di quadratura sono i pesi A_i ed i valori della funzione integranda f nei nodi x_i, in questo paragrafo e nel successivo useremo il simbolo $Q[A, f]$

Teorema 2.2.6. *Condizione necessaria e sufficiente affinchè la successione $\{L_{n-1}(x)\}$ converga uniformemente in $[-1, 1]$ alla funzione $f(x)$ definita in $[-1, 1]$ è che la funzione di variabile complessa $f(z)$ con $z = x + iy$ sia olomorfa in tutti i punti della regione del piano limitata dalla curva di equazione:*

$$\frac{1}{2}(1+x)log[(1+x)^2+y^2] + \frac{1}{2}(1-x)log[(1-x)^2+y^2] +$$
$$+\pi|y| - y\left(arctg\frac{y}{1+x} + arctg\frac{y}{1-x}\right) = 2log2 \tag{2.45}$$

A tale risultato si può pervenire facendo ricorso al *teorema della lemniscata* ([4], pag.90), che determina, appunto, le condizioni necessarie e sufficienti di convergenza uniforme di una successione di polinomi interpolanti una funzione data. A causa dell'espressione della curva, in generale è abbastanza difficile verificare se la funzione $f(z)$ sia olomorfa nella regione limitata dalla (2.45). In generale, per un intervallo $[a, b]$, è possibile ottenere una condizione sufficiente alla convergenza verificando che la $f(z)$ sia olomorfa all'interno dell'ellisse $\mathcal{E}$ di centro $(a+b)/2$ e semiassi $5(b-a)/8$ e $3(b-a)/8$ che è contenuta nella figura in esame ([22], pp 356-357).
La dimostrazione del teorema è nel **Capitolo 8**, **§8.3** (**Teorema 8.3.9**).

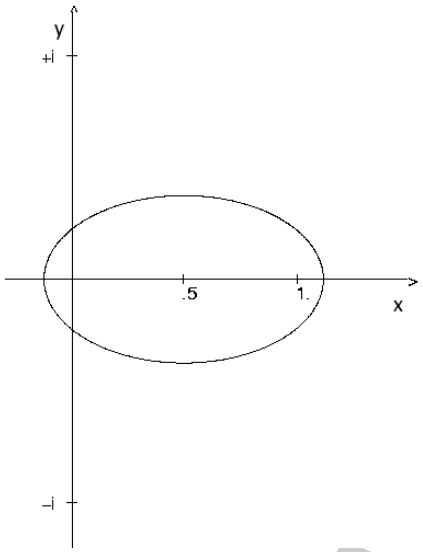

Figura 2.2: Rappresentazione nel piano complesso (x, iy) della regione di convergenza delle formule di Newton-Cotes nell'intervallo $[0, 1]$.

al posto di $Q[f]$ in modo da evidenziare la dipendenza del problema Q dai dati A_i e f. Siano ΔA_i e $\Delta f(x_i)$ rispettivamente le perturbazioni effettuate sui pesi A_i e sulle valutazioni $f(x_i)$, cioè:

$$\tilde{A}_i = A_i + \Delta A_i \quad \widetilde{f(x_i)} = f(x_i) + \Delta f(x_i)$$

Sussiste il seguente:

Teorema 2.3.1. *Se $Q[A, f] \neq 0$, l'errore relativo della soluzione del problema con dati perturbati $Q[\tilde{A}, \tilde{f}]$ rispetto alla soluzione del problema non perturbato $Q[A, f]$ è:*

$$\frac{|Q[\tilde{A}, \tilde{f}] - Q[A, f]|}{|Q[A, f]|} \leq \frac{\sum_{i=1}^{n} |A_i||f(x_i)|}{|\sum_{i=1}^{n} A_i f(x_i)|} \left(\max_i \left| \frac{\Delta A_i}{A_i} \right| + \max_i \left| \frac{\Delta f(x_i)}{f(x_i)} \right| \right) \tag{2.46}$$

Dimostrazione Dalle posizioni fatte si ha:

$$\begin{aligned} Q[\tilde{A}, \tilde{f}] &= \sum_{i=1}^{n} (A_i + \Delta A_i)(f(x_i) + \Delta f(x_i)) \\ &= \sum_{i=1}^{n} A_i f(x_i) + \sum_{i=1}^{n} A_i \Delta f(x_i) + \sum_{i=1}^{n} \Delta A_i f(x_i) + \sum_{i=1}^{n} \Delta A_i \Delta f(x_i) \end{aligned}$$

da cui, trascurando i termini $\Delta A_i \Delta f(x_i)$, si ottiene:

$$\begin{aligned} |Q[\tilde{A}, \tilde{f}] - Q[A, f]| &\leq \sum_{i=1}^{n} (|A_i||\Delta f(x_i)| + |\Delta A_i||f(x_i)|) \\ &= \sum_{i=1}^{n} |A_i||f(x_i)| \frac{|\Delta f(x_i)|}{|f(x_i)|} + \sum_{i=1}^{n} |A_i||f(x_i)| \frac{|\Delta A_i|}{|A_i|} \end{aligned}$$

da cui:

$$\frac{|Q[\tilde{A},\tilde{f}]-Q[A,f]|}{|Q[A,f]|} \leq \frac{\sum_{i=1}^{n}|A_i||f(x_i)|}{|\sum_{i=1}^{n}A_i f(x_i)|}\left(\max_i\left|\frac{\Delta A_i}{A_i}\right|+\max_i\left|\frac{\Delta f(x_i)}{f(x_i)}\right|\right)$$

quindi la tesi. ■

Dalla (2.46) si nota che la quantità:

$$\mu(Q[A,f]) = \frac{\sum_{i=1}^{n}|A_i||f(x_i)|}{|\sum_{i=1}^{n}A_i f(x_i)|} \tag{2.47}$$

rappresenta il fattore di amplificazione delle perturbazioni

$$\frac{|\Delta A_i|}{|A_i|} \quad \text{e} \quad \frac{|\Delta f(x_i)|}{|f(x_i)|},$$

ed è quindi l'**indice di condizionamento relativo** del problema in esame[15]. Si noti infine che $\mu(Q[A,f]) \geq 1$, $\forall Q[A,f]$.

♣ **Esempio 2.14.** Se $Q[A,f]$ è una formula di quadratura con pesi non negativi, cioè $A_i \geq 0$, l'indice di condizionamento è:

$$\mu(Q[A,f]) = \frac{\sum_{i=1}^{n}A_i|f(x_i)|}{|\sum_{i=1}^{n}A_i f(x_i)|}$$

Se inoltre la funzione f non cambia segno nell'intervallo di integrazione, i prodotti $A_i f(x_i)$ hanno tutti lo stesso segno e quindi $\sum_{i=1}^{n}A_i|f(x_i)| = |\sum_{i=1}^{n}A_i f(x_i)|$. In tal caso l'indice di condizionamento assume il valore ottimale $\mu(Q[A,f]) = 1$.
Esempi di formule con le caratteristiche citate sono le formule di Gauss e le formule composite basate su formule con pesi non negativi (ad esempio le formule composite trapezoidale e di Simpson). ♣

[15] Ricordando (**Capitolo 1**, **Parte prima**) che, per una funzione vettoriale ad m componenti, $\underline{f}$, in n variabili,

$$\begin{aligned}\underline{f} &= (f_1,\ldots,f_m)\\ f_i &= f_i(\underline{x}) \qquad i=1,\ldots,m\\ \underline{x} &= (x_1,\ldots,x_n)\end{aligned}$$

l'indice di condizionamento relativo in un punto $\underline{x}$ è:

$$C(\underline{f},\underline{x}) = \left\|\left(\frac{\delta f_i}{\delta x_j}\cdot\frac{x_j}{f_i}\right)\right\|_\infty$$

segue che, considerata la formula di quadratura $Q[A,f]$ in funzione delle sue $2n$ variabili, A_i, $i=1,\ldots,n$ e $f(x_i)$, $i=1,\ldots,n$, il suo indice di condizionamento relativo può esprimersi come:

$$\mu(Q[A,f]) = 2\cdot\frac{\sum_{i=1}^{n}|A_i||f(x_i)|}{|\sum_{i=1}^{n}A_i f(x_i)|} \tag{2.48}$$

Tuttavia tale risultato non è in disaccordo con la (2.47), se si considera che la quantità (2.48) rappresenta il fattore di amplificazione del *massimo* errore relativo presente sulla totalità dei dati, siano essi pesi o valutazioni della funzione integranda nei nodi.

Se $f \in C([a,b])$, posto $M = \max_{[a,b]} |f(x)|$, per la (2.47) si ha:

$$\mu(Q[A,f]) \leq \frac{M \sum_{i=1}^{n} |A_i|}{|Q[A,f]|} \tag{2.49}$$

Pertanto $\mu(Q[A,f])$ dipende strettamente dalla somma dei valori assoluti dei pesi $|A_i|$ della formula di quadratura. Le formule di Newton-Cotes, per $n \geq 9$ hanno pesi di segno alterno e non limitati in modulo[16], per cui il condizionamento di tali formule peggiora al crescere di n. Per tale motivo le formule di Newton-Cotes vengono utilizzate con un numero di nodi molto basso, in genere $n \leq 8$. I pesi delle formule di Gauss, invece, sono tutti positivi, e per la (2.7) si ha

$$\sum_{i=1}^{n} |A_i| = \sum_{i=1}^{n} A_i = (b-a)$$

e quindi $\mu(Q[A,f])$ è limitato per ogni valore di n. Le formule di Gauss, quindi, possono essere considerate ben condizionate.

Osservazione 2.1. *Per una funzione integrabile secondo Riemann in un intervallo $[a,b]$, assegnati $n+1$ nodi $x_i^{(n)}$ in $[a,b]$ $(i = 0,\ldots,n)$ e $n+1$ pesi $A_i^{(n)}$ $(i = 0,\ldots,n)$, una formula di quadratura può essere rappresentata come*

$$Q_n[A,f] = A_0^{(n)} f(x_0^{(n)}) + \cdots + A_n^{(n)} f(x_n^{(n)})$$

oppure, se ciò non genera ambiguità, semplicemente, come

$$Q_n[A,f] = A_0 f(x_0) + \cdots + A_n f(x_n)$$

o, direttamente, come $Q[A,f]$.
Con questa notazione si può riformulare la definizione di formula di quadratura composita:

Definizione 2.3.1. *[19]*
Sia $Q[A,f]$ una formula di quadratura in $[a,b]$. Partizionando questo intervallo in m $(m \geq 2)$ sottointervalli $[t_{j-1}, t_j]$, $(j = 1,\ldots,m)$ ciascuno di ampiezza $\tau_j = t_j - t_{j-1}$, la ***formula composita*** *di $Q[A,f]$ è:*

$$Q_{m,n}[f] = \sum_{j=1}^{m} Q^{(j)}[f] = \sum_{j=1}^{m} \sum_{i=0}^{n} A_{i,j} f(x_{i,j})$$

[16] Ad eccezione, come si è detto, del caso $n = 10$ i cui pesi sono tutti positivi e la formula, quindi, risulta ben condizionata.

dove $Q^{(j)}[f] = \sum_{i=0}^{n} A_{i,j} f(x_{i,j})$ *si ottiene applicando* $Q[A, f]$ *ai sottointervalli* $[t_{j-1}, t_j]$, *cioè*

$$A_{i,j} = A_i \frac{t_j - t_{j-1}}{b-a} \qquad x_{i,j} = t_{j-1} + i\frac{t_j - t_{j-1}}{n} \quad i = 1, \ldots, n \ .$$

Si può, dunque, dare la seguente

Definizione 2.3.2. *Sia* f *una funzione integrabile secondo Riemann su un intervallo reale* $[a, b]$ *e* $Q_n[A, f]$ *una formula di quadratura in* $[a, b]$. *Se* $Q_n[A, f]$ *è diversa da zero, l'**indice di condizionamento relativo*** $\mu(Q)$ *di* $Q_n[A, f] = \sum_{i=0}^{n} A_i f(x_i)$, *è definito come segue:*

$$\mu(Q) = \mu(Q_n[A, f]) \stackrel{\text{def}}{=} \frac{\sum_{i=0}^{n} |A_i||f(x_i)|}{|\sum_{i=0}^{n} A_i f(x_i)|} \tag{2.50}$$

Analogamente è possibile dimostrate che, se $\Delta A_i = 0$, $\forall i = 1, \ldots, n$ allora si ha:

$$\sum_{i=1}^{n} \widetilde{A_i \Delta f(x_i)} = \sum_{i=1}^{n} A_i f(x_i) + \sum_{i=1}^{n} A_i \Delta f(x_i)$$

da cui:

$$|Q[\tilde{A}, \tilde{f}] - Q[A, f]| \leq \max_i |\Delta f(x_i)| \sum_{i=1}^{n} |A_i|$$

Si definisce quindi **indice di condizionamento assoluto** la quantità:

$$\nu(Q[A, f]) = \sum_{i=1}^{n} |A_i|$$

Tale definizione mette nuovamente in risalto la dipendenza del condizionamento di una formula di quadratura dalla somma dei valori assoluti dei pesi.

2.4 Alcuni risultati sull'indice di condizionamento assoluto

In questo paragrafo si analizza l'indice di condizionamento assoluto per le seguenti famiglie di formule di quadratura: le formule di Gauss e di Gauss-Kronrod, le formule di Newton-Cotes e le formule interpolatorie composite.

Proposizione 2.4.1. *Sia* s *un intero non negativo,* $[c, d]$ *e* $[a, b]$ *due intervalli,* w_1 *e* w_2 *come nella **Proposizione 2.1.1** e* $Q[B, f] = \sum_{i=0}^{n} B_i f(x_i)$ *una formula di quadratura nell'intervallo* $[c, d]$, *con grado di precisione algebrico* s *rispetto a* w_1. *Sia* $Q[A, f]$ *la formula di quadratura in* $[a, b]$, *con grado di precisione algebrico* s *rispetto a* w_2, *ottenuta da* $Q[B, f]$ *attraverso la trasformazione lineare (2.9), allora:*

$$\nu(Q[A, f]) = \frac{b-a}{d-c} \nu(Q[B, f]) \ .$$

Dimostrazione Per A_i e B_i sussiste $A_i = \frac{b-a}{d-c} \cdot B_i$ $(i = 0, \ldots, n)$, quindi:

$$\nu(Q[A, f]) = \sum_{i=0}^{n} |A_i| = \sum_{i=0}^{n} \frac{b-a}{d-c} \cdot |B_i| = \frac{b-a}{d-c} \nu(Q[B, f]) \ .$$

■

Le formule di Newton-Cotes con $n+1$ nodi sono interpolatorie e hanno grado di precisione algebrico almeno n, quindi dalla **Proposizione 2.1.1** e dalla **2.4.1** segue che:

Corollario 2.4.1. *Sia $Q_n[B, f]$ la formula di Newton-Cotes nell'intervallo $[0, 1]$ e $Q_n[A, f]$ la formula di Newton-Cotes in $[a, b]$, allora:*

$$\nu(Q_n[A, f]) = (b - a) \cdot \nu(Q_n[B, f]) \ . \tag{2.51}$$

2.4.1 Formule di Gauss e Gauss-Kronrod

Per quanto riguarda le formule di Gauss e Gauss-Kronrod osserviamo, preliminarmente, che:

Osservazione 2.2. *Se $Q[A, f] = \sum_{i=0}^{n} A_i f(x_i)$ ha pesi positivi ed è esatta per funzioni costanti, rispetto alla funzione peso w, allora:*

$$\nu(Q) = \int_a^b w(x)dx \ .$$

Le formule di Gauss hanno pesi positivi e sono esatte per funzioni costanti, allora [5]:

Corollario 2.4.2. *L'indice di condizionamento assoluto di una formula di Gauss nell'intervallo $[a, b]$, rispetto alla funzione peso w, è $\int_a^b w(x)dx$.*

Le formule di Gauss-Kronrod hanno pesi positivi sono esatte per funzioni costanti, allora [5]:

Corollario 2.4.3. *L'indice di condizionamento assoluto di una formula di Gauss-Kronrod in $[a, b]$ è $b - a$.*

Dai corollari precedenti, segue che l'indice di condizionamento assoluto delle formule di Gauss e Gauss-Kronrod è indipendente dai nodi, ovvero esse sono formule ben condizionate.

2.4.2 Formule di Newton-Cotes

Per l'indice di condizionamento assoluto di una formula di Newton-Cotes, si ha [19]:

Teorema 2.4.1. *Sia $Q_n[A,f]$ la formula di Newton-Cotes nell'intervallo $[a,b]$ con $n+1$ nodi, allora*

$$\lim_{n\to+\infty} \nu(Q_n[A,f]) = +\infty \tag{2.52}$$

Determiniamo un'espressione asintotica dell'indice di condizionamento assoluto delle formule di Newton-Cotes. Dimostriamo, a tal fine, il seguente:

Teorema 2.4.2. *Sia $Q_n[A,f] = \sum_{i=0}^{n} A_i f(x_i)$ la formula di Newton-Cotes nell'intervallo $[a,b]$, allora, asintoticamente, sussiste il risultato:*

$$\psi_1(n)\frac{2^{n+2}}{(n^3-n)\log^2 n}(b-a) < \nu(Q_n[A,f]) < \psi_2(n)\frac{2^n}{(n-1)\log^2 n}(b-a) \tag{2.53}$$

dove

$$\psi_1(n) = \left[1+\left(\frac{2(n^2-1)\log n - 8}{2^{n+2}}\right)\right]K_n \quad , \quad \psi_2(n) = \left[1+\left(\frac{2\log n - 2}{2^n}\right)\right]K_n \quad ,$$

$$K_n = \left[1+\mathcal{O}\left(\frac{1}{\log n}\right)\right] \qquad e \qquad \lim_{n\to\infty}\psi_1(n) = \lim_{n\to\infty}\psi_2(n) = 1$$

Dimostrazione Consideriamo la formula di Newton-Cotes in $[0,1]$, allora $\nu(Q_n[B,f]) = \sum_{i=0}^{n}|B_i|$. Dalla (2.13), per $i = 1,2,\ldots,n-1$:

$$|B_i| = \frac{1}{n\log^2 n}\binom{n}{i}\left|\frac{n-i+(-1)^n i}{(n-i)i}\right|K_n \tag{2.54}$$

Usando la (2.12), per B_0 e B_n:

$$\frac{1}{n\log n}K_n = |B_0| = |B_n| < \frac{1}{(n-1)\log n}K_n \tag{2.55}$$

Dalla (2.54) e dal **Lemma 2.4.1** che segue, per $i = 1,2,\ldots,n-1$:

$$\frac{4}{(n^3-n)\log^2 n}\binom{n}{i}K_n \le |B_i| \le \frac{1}{(n-1)\log^2 n}\binom{n}{i}K_n \tag{2.56}$$

Infine, sommando le disuguaglianze nella (2.55) e nella (2.56), per $i = 0,\ldots,n$:

$$|B_o| + |B_n| + \sum_{i=1}^{n-1}\frac{4\binom{n}{i}K_n}{(n^3-n)\log^2 n} < |B_o| + |B_n| + \sum_{i=1}^{n-1}|B_i| < \frac{2K_n}{(n-1)\log n} + \sum_{i=1}^{n-1}\frac{\binom{n}{i}K_n}{(n-1)\log^2 n}$$

$$\frac{2K_n}{n\log n} + \frac{4K_n}{(n^3-n)\log^2 n}\sum_{i=1}^{n-1}\binom{n}{i} < \nu(Q_n[B,f]) < \frac{2K_n}{(n-1)\log n} + \frac{K_n}{(n-1)\log^2 n}\sum_{i=1}^{n-1}\binom{n}{i}$$

$$\frac{[4(2^n-2)+2(n^2-1)\log n]K_n}{(n^3-n)\log^2 n} < \nu(Q_n[B,f]) < \frac{[(2^n-2)+2\log n]K_n}{(n-1)\log^2 n}$$

$$\frac{2^{n+2}\left[1+\left(\frac{2(n^2-1)\log n-8}{2^{n+2}}\right)\right]K_n}{(n^3-n)\log^2 n} < \nu(Q_n[B,f]) < \frac{2^n\left[1+\left(\frac{2\log n-2}{2^n}\right)\right]K_n}{(n-1)\log^2 n} \tag{2.57}$$

Sostituendo ψ_1 e ψ_2 nella (2.57) e moltiplicando per l'ampiezza dell'intervallo $[a,b]$, si ottiene la tesi. ■

Il Teorema 2.4.2 permette di stimare il comportamento asintotico dell'indice di condizionamento assoluto di una formula di Newton-Cotes, consentendo di dedurne, per una classe di funzioni, il comportamento della convergenza.

Lemma 2.4.1. *Per ogni $n \in \mathcal{N} - \{1\}$ e per ciascun $i = 1, \ldots, n-1$, sussiste:*

$$\frac{4}{n^2-1} \leq \left|\frac{1}{i} + \frac{(-1)^n}{(n-i)}\right| \leq \frac{n}{n-1} \tag{2.58}$$

Dimostrazione Per un fissato $n \in \mathcal{N} - \{1\}$, consideriamo la funzione reale:

$$\varphi(x) = \left|\frac{1}{x} + \frac{(-1)^n}{n-x}\right| = \left|\frac{n-x+(-1)^n x}{(n-x)x}\right|.$$

Vogliamo determinare il massimo ed il minimo valore di φ assunto in corrispondenza di valori interi nell'intervallo $[1, n-1]$. Osserviamo che $\varphi(x)$ è simmetrica rispetto a $x = n/2$. In effetti,

$$\varphi(n-x) = \left|\frac{n-(n-x)+(-1)^n(n-x)}{x(n-x)}\right| = \left|\frac{x+(-1)^n(n-x)}{(n-x)x}\cdot(-1)^n\right| =$$

$$= \left|\frac{(-1)^n x+(n-x)(-1)^{2n}}{(n-x)x}\right| = \left|\frac{n-x+(-1)^n x}{(n-x)x}\right| = \varphi(x).$$

Quindi possiamo limitarci a trovare il massimo ed il minimo valore solo in $[1, n/2]$. Distinguiamo i due casi: n pari e n dispari.
Se n è pari, allora:

$$\varphi(x) = \left|\frac{n-x+x}{(n-x)x}\right| = \frac{n}{(n-x)x}\ .$$

Derivando

$$\varphi'(x) = \frac{n(2x-n)}{(n-x)^2x^2}.$$

Se $x \in [1, n/2]$, φ' è negativa e si annulla se e solo se $x = n/2$. Quindi φ è una funzione decrescente ed assume il massimo ed il minimo valore nei punti estremi dell'intervallo: $x = 1$ e $x = n/2$ che sono interi (n è pari e $n/2$ è un intero), così:

$$\varphi\left(\frac{n}{2}\right) = \frac{4}{n} \leq \left|\frac{n-i+(-1)^n i}{(n-i)i}\right| \leq \frac{n}{n-1} = \varphi(1) \qquad (i = 1, \ldots, n-1) \tag{2.59}$$

Se n è dispari, allora:

$$\varphi(x) = \left|\frac{n-x-x}{(n-x)x}\right| = \frac{n-2x}{(n-x)x}\ .$$

Derivando

$$\varphi'(x) = -\frac{[(n-2x)^2+2x(n-x)]}{x^2(n-x)^2}.$$

In $[1, n/2]$, φ' è negativa. Quindi φ è una funzione decrescente ed, in corrispondenza di valori interi, assume il massimo valore nell'estremo dell'intervallo $x = 1$ ed il minimo valore in corrispondenza dell'intero $x = (n-1)/2$ (n dispari), così:

$$\varphi\left(\frac{n-1}{2}\right) = \frac{4}{n^2-1} \leq \left|\frac{n-i+(-1)^n i}{(n-i)i}\right| \leq \frac{n-2}{n-1} = \varphi(1) \qquad (i = 1, \ldots, n-1) \tag{2.60}$$

Combinando la (2.59) e la (2.60) si prova la tesi. ■

2.4.3 Formule di quadratura composite basate sulle formule interpolatorie

Analizziamo l'indice di condizionamento assoluto per la classe di formule interpolatorie composite. In accordo con la notazione nella **Definizione 2.3.1** dimostriamo il seguente:

Teorema 2.4.3. *Sia $Q[A, f] = \sum_{i=0}^{n} A_i f(x_i)$ una formula di quadratura interpolatoria. Sia $Q_{m,n}[f]$ una formula composita di $Q[A, f]$ su m ($m \geq 2$) sottointervalli. Allora:*

$$\nu(Q_{m,n}[f]) \leq \nu(Q) \ .$$

dove la disuguaglianza stretta sussiste se e solo se $Q[A, f]$ è una formula chiusa e se ha A_0 ed A_n di segno discorde.

Per dimostrare il **Teorema 2.4.3** facciamo una distinzione tra formule base chiuse e formule base non chiuse. Se una formula base non è chiusa, allora:

Lemma 2.4.2. *Sia $Q[A, f] = \sum_{i=0}^{n} A_i f(x_i)$ una formula di quadratura interpolatoria non chiusa, $Q_{m,n}[f]$ una formula composita di $Q[A, f]$ su m ($m \geq 2$) sottointervalli e $Q^{(j)}[f] = \sum_{i=0}^{n} A_{i,j} f(x_{i,j})$ la formula ottenuta applicando $Q[A, f]$ in $[t_{j-1}, t_j]$. Allora*

$$\nu(Q_{m,n}[f]) = \sum_{j=1}^{m} \nu(Q^{(j)}[f]) \ . \tag{2.61}$$

Dimostrazione $Q[A, f]$ è una formula non chiusa quindi, se denotiamo con W_k i pesi di $Q_{m,n}[f]$, segue che:

$$\nu(Q_{m,n}[f]) = \sum_{k=0}^{m(n+1)-1} |W_k| = \sum_{j=1}^{m} \sum_{i=0}^{n} |A_{i,j}| = \sum_{j=1}^{m} \nu(Q^{(j)}[f])$$

■

Lemma 2.4.3. *Sia* $Q[A,f] = \sum_{i=0}^{n} A_i f(x_i)$ *una formula di quadratura interpolatoria in* $[a,b]$ *e* $Q^{(j)}[f] = \sum_{i=0}^{n} A_{i,j} f(x_{i,j})$ *la formula ottenuta applicando* $Q[A,f]$ *in* $[t_{j-1}, t_j]$. *Allora:*

$$\nu(Q^{(j)}[f]) = \frac{t_j - t_{j-1}}{b-a}\nu(Q) \ . \tag{2.62}$$

Dimostrazione Applicando $Q[A,f]$ in $[t_{j-1}, t_j]$, costruiamo $Q^{(j)}[f]$ mediante una trasformazione lineare. I pesi $A_{i,j}$ sono proporzionali ai pesi A_i, mediante il fattore $\frac{t_j - t_{j-1}}{b-a}$. La **Proposizione 2.4.1** completa la dimostrazione. ■

Proposizione 2.4.2. *Sia* $Q[A,f] = \sum_{i=0}^{n} A_i f(x_i)$ *una formula di quadratura interpolatoria non chiusa. Sia* $Q_{m,n}[f]$ $(m \geq 2)$ *una formula composita di* $Q[A,f]$ *su* m *sottointervalli. Allora:*

$$\nu(Q_{m,n}[f]) = \nu(Q) \ .$$

Dimostrazione Dalle ipotesi, applicando il **Lemma 2.4.2**:

$$\nu(Q_{m,n}[f]) = \sum_{j=1}^{m} \nu(Q^{(j)}[f]) \ .$$

Poiché $Q[A,f]$ è interpolatoria e, per il **Lemma 2.4.3**, si ha:

$$\nu(Q^{(j)}[f]) = \frac{t_j - t_{j-1}}{b-a}\nu(Q)$$

allora

$$\nu(Q_{m,n}[f]) = \sum_{j=1}^{m} \nu(Q)\left(\frac{t_j - t_{j-1}}{b-a}\right) = \frac{\nu(Q)}{b-a}\sum_{j=1}^{m}(t_j - t_{j-1}) = \nu(Q)$$

■

Se la formula base è chiusa, allora:

Lemma 2.4.4. *Sia* $Q[A,f] = \sum_{i=0}^{n} A_i f(x_i)$ *una formula di quadratura interpolatoria chiusa e* $Q_{m,n}[f]$ *una formula composita di* $Q[A,f]$ *su* m $(m \geq 2)$ *sottointervalli. Allora:*

$$\nu(Q_{m,n}[f]) \leq \sum_{j=1}^{m} \nu(Q^{(j)}[f]) \tag{2.63}$$

dove l'uguaglianza sussiste se e solo se A_0 *e* A_n *hanno lo stesso segno.*

Dimostrazione $Q[A, f]$ è una formula chiusa, allora se denotiamo con W_k i pesi di $Q_{m,n}[f]$, segue che:

$$W_{tn} = A_{n,t} + A_{0,t+1} \qquad t = 1, \ldots, m-1$$

e quindi

$$|W_{tn}| \leq |A_{n,t}| + |A_{0,t+1}|$$

dove il segno dell'uguaglianza sussiste se e solo se $A_{n,t}$ e $A_{0,t+1}$ hanno lo stesso segno o, equivalentemente, se A_0 e A_n di $Q[A, f]$ hanno lo stesso segno. Questo dimostra la tesi:

$$\nu(Q_{m,n}[f]) = \sum_{k=0}^{mn} |W_k| \leq \sum_{j=1}^{m} \sum_{i=0}^{n} |A_{i,j}| = \sum_{j=1}^{m} \nu(Q^{(j)}[f]) \ .$$

■

Proposizione 2.4.3. *Sia* $Q[A, f] = \sum_{i=0}^{n} A_i f(x_i)$ *una formula di quadratura interpolatoria e chiusa. Sia* $Q_{m,n}[f]$ $(m \geq 2)$ *una formula composita di* $Q[A, f]$ *su* m *sottointervalli. Allora:*

$$\nu(Q_{m,n}[f]) \leq \nu(Q)$$

dove l'uguaglianza sussiste se e solo se A_0 *e* A_n *hanno lo stesso segno.*

Dimostrazione Per il **Lemma 2.4.4**:

$$\nu(Q_{m,n}[f]) \leq \sum_{j=1}^{m} \nu(Q^{(j)}[f]) \ .$$

Poiché $Q[A, f]$ è interpolatoria e, per il **Lemma 2.4.3**, si ha:

$$\nu(Q^{(j)}[f]) = \frac{t_j - t_{j-1}}{b-a} \nu(Q)$$

allora:

$$\nu(Q_{m,n}[f]) \leq \sum_{j=1}^{m} \nu(Q) \left(\frac{t_j - t_{j-1}}{b-a} \right) = \frac{\nu(Q)}{b-a} \sum_{j=1}^{m} (t_j - t_{j-1}) = \nu(Q)$$

■

Le **Proposizioni 2.4.2** e **2.4.3** dimostrano il **Teorema 2.4.3**.

Abbiamo due semplici corollari del **Teorema 2.4.3**. Le formule di Gauss e Gauss-Kronrod sono interpolatorie aperte, quindi:

Corollario 2.4.4. *Le formule composite di Gauss e di Gauss-Kronrod hanno lo stesso indice di condizionamento assoluto delle corrispondenti formule base.*

Lo stesso risultato resta vero per una formula di Newton-Cotes. Infatti esse sono formule interpolatorie e chiuse e hanno $A_0 = A_n$.

Corollario 2.4.5. *Le formule composite di Newton-Cotes hanno lo stesso indice di condizionamento assoluto delle corrispondenti formule base.*

2.5 Errore di round-off nella valutazione di una formula di quadratura

L'approssimazione di un integrale $I[f]$ mediante una formula di quadratura è affetta, oltre che dall'errore di discretizzazione $E[f]$:

$$E[f] = I[f] - Q[A, f],$$

anche dall'errore di round-off R, generato dal calcolo della somma $Q[A, f]$ in un sistema aritmetico a precisione finita. Cioè:

$$R = Q^*[A, f] - Q[A, f]$$

dove $Q^*[A, f]$ è il risultato dell'algoritmo utilizzato per la somma che definisce la formula di quadratura e che diremo *risultato computazionale*[17]. Quindi:

$$|I[f] - Q^*[A, f]| \leq |E[f]| + |R|$$

♣ **Esempio 2.15.** Si calcoli:

$$I[f] = \int_0^{\pi/2} (3x + 4)\, dx = 9.9843,$$

mediante la formula trapezoidale composita $T_m[f]$ al variare del numero di sottointervalli $m = 2^k$, $k = 1, ..., 16$ in un sistema aritmetico a precisione finita. Nella Tabella 2.10 sono riportati gli errori $|T_m^*[f] - I[f]|$ calcolati utilizzando un calcolatore con aritmetica standard IEEE in singola precisione. Si noti che la formula trapezoidale composita ha grado di precisione algebrico 1, e quindi dovrebbe fornire, in un'aritmetica a precisione infinita, un errore nullo qualunque sia il valore del numero di sottointervalli m.
Dalla Tabella si può notare che fino a $m = 1024$ l'errore $|T_m^*[f] - I[f]|$ è nullo, ma per $m > 1024$ l'errore è diverso da zero. Il degrado dell'accuratezza dopo un numero elevato di passi è dovuto ovviamente all'accumulo degli errori di round-off nella valutazione della somma della formula di quadratura. ♣

Proposizione 2.5.1. *L'errore di round-off nel calcolo di una formula di quadratura è:*

$$|Q^*[A, f] - Q[A, f]| \leq \mu(Q[A, f]) \left(\frac{nu}{1 - nu}\right) 2 \cdot |Q[A, f]|$$

[17]Spesso, per esprimere il risultato computazionale relativo ad un valore f, cioè l'output dell'algoritmo utilizzato per valutare f, si usa anche la notazione *alg(f)*.

m	$\lvert T_m^*[f] - I[f]\rvert$	m	$\lvert T_m^*[f] - I[f]\rvert$
2	.0000000000E+00	1024	.0000000000E+00
4	.0000000000E+00	2048	.5435943604E-04
8	.0000000000E+00	4096	.9346008301E-04
16	.0000000000E+00	8192	.5722045898E-04
32	.0000000000E+00	16384	.3242492676E-04
64	.0000000000E+00	32768	.1525878906E-04
128	.0000000000E+00	65536	.3433227539E-04
256	.0000000000E+00	131072	.1621246338E-04
512	.0000000000E+00	262144	.4596710205E-03

Tabella 2.10: Errore di round-off per la formula trapezoidale composita.

Dimostrazione Allo scopo di fornire una stima per $|R|$ si considerino le somme:

$$S_k = \sum_{i=1}^{k} A_i f(x_i) = S_{k-1} + A_k f(x_k) \quad k = 1, ..., n \quad \text{con } S_0 = 0 \tag{2.64}$$

Posto $|\sigma_i| \leq u$ e $|\tau_i| \leq u$ rispettivamente gli errori di round-off associati alle operazioni di prodotto e di somma e $u = \frac{1}{2} \times \beta^{1-t}$ la massima accuratezza relativa del sistema aritmetico floating point $\mathcal{F}$ a precisione t, la sequenza calcolata da un algoritmo che valuta le (2.64) in $\mathcal{F}$, è la seguente:

$$\begin{array}{ll} S_1^* & = A_1 f(x_1)(1+\sigma_1) \\ S_2^* & = [S_1^* + A_2 f(x_2)(1+\sigma_2)](1+\tau_2) = \\ & = [A_1 f(x_1)(1+\sigma_1) + A_2 f(x_2)(1+\sigma_2)](1+\tau_2) = \\ & = A_1 f(x_1)(1+\sigma_1)(1+\tau_2) + A_2 f(x_2)(1+\sigma_2)(1+\tau_2) \\ \vdots & \vdots \\ Q^*[A,f] & = S_n^* = \\ & = A_1 f(x_1)(1+\tau_2)\cdots(1+\tau_n)(1+\sigma_1) + \\ & + A_2 f(x_2)(1+\tau_2)\cdots(1+\tau_n)(1+\sigma_2) + \\ & + A_n f(x_n)(1+\tau_n)(1+\sigma_n) \end{array} \tag{2.65}$$

Per semplificare le espressioni precedenti è possibile non fare distinzione tra gli errori di round-off nelle somme e nei prodotti (rispettivamente τ_i e σ_i), identificandoli tutti con δ_i. In tal modo, ricordando che se $|\delta_i| \leq u$ e $nu < 1$, si ha:

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n \qquad |\theta_n| \leq \frac{nu}{1-nu} \tag{2.66}$$

per cui, applicando la (2.66) all'ultima uguaglianza della (2.65), si ha:

$$\begin{array}{rcl} Q^*[A,f] & = & A_1 f(x_1)(1+\theta_n) + A_2 f(x_2)(1+\theta_n) \\ & + & A_3 f(x_3)(1+\theta_{n-1}) + \cdots + A_n f(x_n)(1+\theta_2) \end{array} \tag{2.67}$$

Dalla (2.67) si ricava che $Q^*[A,f]$ può essere considerata, dal punto di vista della backward error analysis, come la formula di quadratura ottenuta in un sistema aritmetico a precisione infinita a partire dai pesi perturbati $\widetilde{A_i}$:

$$Q^*[A,f] = Q[\tilde{A},f] \quad \text{con} \quad \frac{|A_i - \widetilde{A_i}|}{|A_i|} \leq |\theta_i| \leq \frac{nu}{1-nu} \quad i = 1, ..., n$$

oppure come la formula di quadratura ottenuta in un sistema aritmetico a precisione infinita a partire dalle valutazioni di funzione perturbate $\widetilde{f(x_i)}$:

$$Q^*[A,f] = Q[A,\tilde{f}] \quad \text{con} \quad \frac{|f(x_i) - \widetilde{f(x_i)}|}{|f(x_i)|} \leq |\theta_i| \leq \frac{nu}{1-nu} \quad i = 1, ..., n$$

Dal Teorema 2.3.1, facendo uso della (2.47), si ottiene l'asserto. ■

Si noti infine che le formule di quadratura con pesi uguali del tipo:

$$Q[f] = A\sum_{i=1}^{n} f(x_i)$$

riducono ulteriormente l'errore di round-off, in quanto gli errori σ_i nella (2.65) risultano nulli.

2.6 Proprietà notevoli della formula trapezoidale composita

Si vuole approfondire un risultato relativo alla convergenza della formula trapezoidale composita per funzioni integrande sviluppabili in serie di Fourier nell'intervallo $[0,1]$.

♣ **Esempio 2.16.** Si calcoli l'errore della formula trapezoidale composita relativamente all'integrale:

$$\int_0^1 (1 + .5sin(2\pi x))^{-1} dx = 1.154700534\ldots$$

utilizzando un calcolatore con aritmetica standard IEEE. Si noti che la funzione integranda è una funzione periodica nell'intervallo $[0,1]$. Nella Tabella 2.11 è mostrato l'andamento dell'errore $E_m[f]$ al crescere del numero di sottointervalli m.

L'esempio mostra che al crescere del numero di nodi la velocità di convergenza è molto maggiore di quella prevista, che come è noto, è dell'ordine $\mathcal{O}(h^2)$. Sette cifre decimali di accuratezza sono ottenute già con $m = 16$. ♣

Teorema 2.6.1. *Sia f una funzione di classe almeno $C^1([0,1])$ tale che $f(0) = f(1)$ e sia $T_m[f] = \frac{1}{m}\sum_{j=1}^{m} f\left(\frac{j}{m}\right)$ la formula trapezoidale composita su m intervalli di ampiezza $h = 1/m$:*

$$[t_{j-1}, t_j], \quad t_j = jh, \quad j = 1, ..., m, \quad t_0 = 0$$

m	$E_m[f]$
2	$0.1547005177 \times 10^{0}$
4	$0.1196610928 \times 10^{-01}$
8	$0.6139278412 \times 10^{-04}$
16	$0.1192092896 \times 10^{-06}$
32	$0.1192092896 \times 10^{-06}$
64	$0.1192092896 \times 10^{-06}$

Tabella 2.11: Errore della formula trapezoidale composita su m sottointervalli per la funzione periodica $f(x) = (1 + .5\, sin(2\pi x))^{-1}$.

Allora si ha che:

$$E_m[f] = T_m[f] - I[f] = \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} a_{km}(f) \tag{2.68}$$

con a_{km} coefficienti di Fourier di f i cui indici sono multipli di m.

Dimostrazione Si consideri lo sviluppo in serie di Fourier della funzione f:

$$f(t) = \sum_{r=-\infty}^{\infty} a_r(f) e^{2\pi i r t} \tag{2.69}$$

dove con $a_r(f)$ sono stati indicati i coefficienti di Fourier:

$$a_r(f) = \int_0^1 f(t) e^{-2\pi i r t}\, dt. \tag{2.70}$$

Si nota che $a_0(f) = I[f]$. Sostituendo la (2.69) nella formula trapezoidale si ottiene:

$$T_m[f] = \frac{1}{m} \sum_{j=1}^{m} \sum_{r=-\infty}^{\infty} a_r(f) e^{2\pi i r j/m} = I[f] + \sum_{\substack{r=-\infty \\ r\neq 0}}^{\infty} a_r(f) T_m[e^{2\pi i r x}],$$

da cui si ricava:

$$E_m[f] = T_m[f] - I[f] = \sum_{\substack{r=-\infty \\ r\neq 0}}^{\infty} a_r(f) T_m[e^{2\pi i r x}]. \tag{2.71}$$

La formula precedente (detta formula della sommatoria di Poisson) esprime l'errore $E_m[f] = T_m[f] - I[f]$ mediante una serie coinvolgente i coefficienti di Fourier della funzione integranda.
In generale, risulta

$$T_m[e^{2\pi i r x}] = \begin{cases} 1 & \text{se } r/m \text{ intero} \\ 0 & \text{se } r/m \text{ non intero} \end{cases}$$

e, quindi:

$$E_m[f] = T_m[f] - I[f] = \sum_{\substack{k=-\infty \\ k\neq 0}}^{\infty} a_{km}(f). \tag{2.72}$$

La sommatoria di Poisson si riduce quindi alla somma dei soli coefficienti di Fourier i cui indici sono multipli di m. ■

♣ **Esempio 2.17.** Calcolare l'errore della formula trapezoidale composita nel caso in cui $m = 3$. Dalla (2.71) si ha allora che

$$T_3[e^{2\pi i 1x}] = \frac{1}{3}\sum_{j=1}^{3} e^{2\pi ij/3} = \frac{1}{3}\sum_{j=1}^{3} \cos(2\pi\tfrac{j}{3}) + i\sin(2\pi\tfrac{j}{3}) = 0$$
$$T_3[e^{2\pi i 2x}] = \frac{1}{3}\sum_{j=1}^{3} e^{4\pi ij/3} = \frac{1}{3}\sum_{j=1}^{3} \cos(4\pi\tfrac{j}{3}) + i\sin(4\pi\tfrac{j}{3}) = 0$$
$$T_3[e^{2\pi i 3x}] = \frac{1}{3}\sum_{j=1}^{3} e^{6\pi ij/3} = \frac{1}{3}\sum_{j=1}^{3} \cos(6\pi\tfrac{j}{3}) + i\sin(6\pi\tfrac{j}{3}) = 1$$

Da cui si ha che:

$$E_3[f] = a_{\pm 3}(f) + a_{\pm 6}(f) + a_{\pm 9}(f) + \ldots$$

♣

Ł'utilità del risultato del **Teorema 2.6.1** sta nel fatto che esso consente di legare l'andamento asintotico dell'errore a quello dei coefficienti di Fourier della funzione f. La formula trapezoidale composita è quindi particolarmente adatta per le funzioni i cui coefficienti di Fourier $a_r(f)$ decadono rapidamente al crescere di r. La velocità con cui gli $a_r(f)$ decadono a 0 è strettamente legata al grado di regolarità della funzione integranda. Infatti vale il seguente:

Teorema 2.6.2. *Se $f \in C^p([0,1])$ si ha che*

$$a_r(f) = -\sum_{q=1}^{p} \frac{f^{(q-1)}(1) - f^{(q-1)}(0)}{(2\pi ir)^q} + \frac{1}{(2\pi ir)^p}\int_0^1 f^{(p)}(t)e^{-2\pi irt}\,dt.$$

Dimostrazione Integrando ripetutamente per parti il secondo membro della (2.70) si ha:

$$a_r(f) = -\frac{f(1) - f(0)}{2\pi ir} + \frac{1}{2\pi ir}\int_0^1 f'(t)e^{-2\pi irt}\,dt =$$

$$= -\frac{f(1) - f(0)}{2\pi ir} - \frac{f'(1) - f'(0)}{(2\pi ir)^2} + \frac{1}{(2\pi ir)^2}\int_0^1 f''(t)e^{-2\pi irt}\,dt = \cdots$$

per cui la tesi si ottiene dopo p integrazioni per parti. ■

Corollario 2.6.1. *Se la funzione integranda f è di classe $C^p([0,1])$ con $p > 1$ e periodica con le sue $p-1$ derivate, allora si ha:*

$$E_m[f] = \mathcal{O}(m^{-p}) = \mathcal{O}(h^p), \tag{2.73}$$

cioè l'ordine di convergenza dell' errore di discretizzazione è p

Dimostrazione Poichè $f \in C^p([0,1])$ con $p > 1$ i coefficienti di Fourier soddisfano la disuguaglianza:

$$|a_r(f)| \leq \frac{C}{r^p},$$

con C costante che dipende da $f^{(p)}(t)$ e, dalla formula (2.72), si ha

$$\lim_{m\to\infty} |T_m[f] - I[f]| = 0$$

Inoltre, poiché il termine più significativo è $a_m(f)$, si può affermare che

$$E_m[f] = \mathcal{O}(m^{-p}) = \mathcal{O}(h^p),$$

e che l'ordine di convergenza è p. ■

La funzione integranda dell'esempio 2.16 è periodica ed è di classe $C^\infty([0,1])$, per cui l'ordine di convergenza della formula trapezoidale è, in questo caso, superiore a $\mathcal{O}(h^p)$ per ogni intero p.
Se la funzione integranda non presenta le caratteristiche di periodicità, necessarie per ottenere la rapida convergenza della formula trapezoidale composita, si può cercare di modificare opportunamente tale funzione mediante una trasformazione del tipo:

$$x = \psi(t)$$

facendo in modo che quest'ultima abbia le caratteristiche richieste. Per scelte opportune di $\psi(t)$, si ha:

$$\int_0^1 f(x)\,dx = \int_0^1 f(\psi(t))\psi'(t)\,dt = \int_0^1 g(t)\,dt$$

con $g(t)$ soddisfacente le proprietà richieste. Una possibile scelta per $x = \psi(t)$ è, ad esempio, $\psi(t) = 3t^2 - 2t^3$. Tale funzione ha come derivata $\psi'(t) = 6t(1-t)$ e quindi la funzione $g(t)$ risulta periodica nell'intervallo $[0,1]$, poiché $\psi'(0) = \psi'(1) = 0$.

Corollario 2.6.2. *Se la funzione integranda f è sufficientemente regolare, dal Teorema 2.6.2, si ha che l'errore di discretizzazione $E_m[f]$ è dato da:*

$$E_m[f] = C_2h^2 + C_4h^4 + C_6h^6 + \cdots \tag{2.74}$$

con

$$C_{2k} = \frac{(-1)^{(k+1)}B_{2k}(f^{(2k-1)}(1) - f^{(2k-1)}(0))}{2k!} \quad k = 1, 2, ...,$$

dove i B_{2k} sono detti coefficienti di Bernoulli:

$$B_{2k} = \frac{(2k)!}{(2\pi)^{2k}} 2 \sum_{j=1}^{\infty} j^{-2k}, \qquad k = 1, 2, \ldots$$

Dimostrazione È utile osservare che, se la funzione è sufficientemente regolare, dal Teorema 2.6.2, si ha che l'errore $E[f] = T_m[f] - I[f]$ è dato dalla somma delle quantità:

$$a_m(f) = -\frac{f'(1)-f'(0)}{(2\pi im)^2} - \frac{f''(1)-f''(0)}{(2\pi im)^3} - \frac{f'''(1)-f'''(0)}{(2\pi im)^4} - ...$$

$$a_{-m}(f) = -\frac{f'(1)-f'(0)}{(2\pi im)^2} + \frac{f''(1)-f''(0)}{(2\pi im)^3} - \frac{f'''(1)-f'''(0)}{(2\pi im)^4} - ...$$

$$a_{2m}(f) = -\frac{f'(1)-f'(0)}{2^2(2\pi im)^2} - \frac{f''(1)-f''(0)}{2^3(2\pi im)^3} - \frac{f'''(1)-f'''(0)}{2^4(2\pi im)^4} - ...$$

$$a_{-2m}(f) = -\frac{f'(1)-f'(0)}{2^2(2\pi im)^2} + \frac{f''(1)-f''(0)}{2^3(2\pi im)^3} - \frac{f'''(1)-f'''(0)}{2^4(2\pi im)^4} - ...$$

da cui, sommando i termini relativi alle stesse derivate, si ha:

$$E[f] = \frac{f'(1)-f'(0)}{m^2(2\pi)^2} 2\sum_{j=1}^{\infty} j^{-2} + 0 - \frac{f'''(1)-f'''(0)}{m^4(2\pi)^4} 2\sum_{j=1}^{\infty} j^{-4} + 0 + \cdots$$

Posto quindi:

$$B_2 = \frac{2!}{(2\pi)^2} 2\sum_{j=1}^{\infty} j^{-2} \quad B_4 = \frac{4!}{(2\pi)^4} 2\sum_{j=1}^{\infty} j^{-4} \quad B_{2k} = \frac{(2k)!}{(2\pi)^{2k}} 2\sum_{j=1}^{\infty} j^{-2k}, \qquad k = 1, 2, \ldots \tag{2.75}$$

si ottiene:

$$E[f] = T_m[f] - I[f] = \frac{B_2(f'(1)-f'(0))}{2!m^2} - \frac{B_4(f'''(1)-f'''(0))}{4!m^4} + ... \tag{2.76}$$

Ricordando che $h = 1/m$, dalla (2.76) si ottiene:

$$E[f] = T_m[f] - I[f] = \frac{B_2h^2(f'(1)-f'(0))}{2!} - \frac{B_4h^4(f'''(1)-f'''(0))}{4!} + ... \tag{2.77}$$

Posto $C_{2j} = (-1)^{(j+1)}B_{2j}(f^{(2j-1)}(1) - f^{(2j-1)}(0))/2j! \quad j = 1, 2, ...$, si ha:

$$E_m[f] = T_m[f] - I[f] = C_2h^2 + C_4h^4 + C_6h^6 + \cdots$$

da cui si ritrova la (2.73) nel caso in cui le derivate di f assumano lo stesso valore agli estremi dell'intervallo $[a, b]$. I coefficienti B_{2k} della (2.75) sono detti coefficienti di Bernoulli e non dipendono da m né dalla funzione integranda, mentre la (2.76) è nota come formula di Eulero-McLaurin. ■

Si osservi che il primo termine della (2.74) è di ordine $\mathcal{O}(h^2)$ ovvero l'ordine di convergenza della formula trapezoidale composita è $p = 2$. La formula di Eulero-McLaurin fornisce, dunque, un'espressione asintotica dell' errore di discretizzazione $E_m[f] = T_m[f] - I[f]$, in termini del passo di discretizzazione h. Tale formula rappresenta lo strumento matematico alla base di uno schema per la costruzione di una classe di formule di quadratura con ordine di convergenza che raddoppia ad ogni passo. Sussiste il risultato seguente:

Proposizione 2.6.1. *Sia $T_m[f]$ la formula trapezoidale composita su m sottointervalli. Posto $T_m^{(1)}[f] = (4T_{2m}[f] - T_m[f])/3$ si ha:*

$$T_m^{(1)}[f] - I[f] = -C_4h^4/4 - 5C_6h^6/16 + \cdots$$

Dimostrazione Dalla (2.74) si ha:

$$\begin{array}{lll} T_m[f] - I[f] & = C_2 h^2 & +C_4 h^4 \quad +C_6 h^6 + \cdots \\ T_{2m}[f] - I[f] & = C_2 (h/2)^2 & +C_4 (h/2)^4 \quad +C_6 (h/2)^6 + \cdots \end{array}$$

da cui, moltiplicando ambo i membri della seconda espressione per 4 e sottraendo a quest'ultima la prima equazione, si ottiene:

$$4T_{2m}[f] - T_m[f] - 3I[f] = -3C_4 h^4/4 - 15C_6 h^6/16 + \cdots$$

da cui, dividendo ambo i membri per 3 si ha:

$$\frac{4T_{2m}[f] - T_m[f]}{3} - I[f] = T_m^{(1)}[f] - I[f] = -C_4 h^4/4 - 5C_6 h^6/16 + \cdots \tag{2.78}$$

■

La (2.78) definisce, in effetti, la formula $T_m^{(1)}[f]$ come una formula di quadratura il cui errore di discretizzazione è dato dall'ultimo membro della (2.78). Dalla (2.78) si osserva, inoltre, che $T_m^{(1)}[f]$ ha ordine di convergeza $p = 4$, cioè il doppio rispetto a $T_m[f]$. In altre parole, a partire da $T_m[f]$ abbiamo ricavato un'approssimazione di $I[f]$ che, utilizzando approssimazioni di $I[f]$ meno accurate, fornisce una formula di quadratura molto più accurata. Tale tecnica è nota, in generale, come *procedimento di estrapolazione di Romberg*. Il procedimento di estrapolazione di Romberg rientra tra gli schemi di "accelerazione della convergenza" di una successione[18]. Applicando il procedimento di estrapolazione di Romberg alla coppia di formule $T_{2m}[f]$ e $T_{4m}[f]$, si ottiene:

$$T_{2m}^{(1)}[f] - I[f] = -C_4 h^4/2^6 - 5C_6 h^6/2^{10} + \cdots = \mathcal{O}(h^4).$$

Così proseguendo si costruisce la successione $\{T_m^{(1)}[f]\}_{m=2k} \quad k = 1, 2, \ldots$ che converge ad $I[f]$ con ordine $p = 4$. Lo stesso procedimento può essere applicato alla sequenza

$$T_m^{(1)}[f],\ T_{2m}^{(1)}[f],\ T_{4m}^{(1)}[f],\ T_{8m}^{(1)}[f],\ \ldots.$$

In particolare:

Proposizione 2.6.2.

$$\begin{array}{lll} T_m^{(1)}[f] - I[f] & = D_4 h^4 & +D_6 h^6 + \cdots \\ T_{2m}^{(1)}[f] - I[f] & = D_4 (h/2)^4 & +D_6 (h/2)^6 + \cdots \end{array}$$

avendo definito in maniera opportuna i coefficienti D_i.

[18] Assegnata una successione $\{s_n\}_{n\in\mathcal{N}}$ convergente a s, uno schema di accelerazione della convergenza di $\{s_n\}_{n\in\mathcal{N}}$ è un procedimento che, a partire da $\{s_n\}_{n\in\mathcal{N}}$, costruisce una successione $\{s'_n\}_{n\in\mathcal{N}}$ tale che $\{s'_n\}_{n\in\mathcal{N}}$ converga a s e $lim_{n\to\infty} \frac{s'_n}{s_n} = 0$.

Dimostrazione Moltiplicando ambo i membri della seconda espressione per 16 e sottraendo a quest'ultima la prima equazione, si ottiene:

$$16T_{2m}^{(1)}[f] - T_m^{(1)}[f] - 15I[f] = 3D_6h^6/4 + \cdots$$

da cui, dividendo ambo i membri per 15 si ha:

$$\frac{16T_{2m}^{(1)}[f] - T_m^{(1)}[f]}{15} - I[f] = T_m^{(2)}[f] - I[f] = 1D_6h^6/20 + \cdots \tag{2.79}$$

ottenendo quindi una formula con ordine di convergenza $p = 6$. ■

In definitiva si ottiene la Tabella 2.12. In essa, la prima colonna ha ordine di convergenza

$T_1[f]$			
$T_2[f]$	$T_1^{(1)}[f]$		
$T_4[f]$	$T_2^{(1)}[f]$	$T_1^{(2)}[f]$	
$T_8[f]$	$T_4^{(1)}[f]$	$T_2^{(2)}[f]$	$T_1^{(3)}[f]$
$\vdots$			$\ddots$

Tabella 2.12: Procedimento di estrapolazione di Romberg.

$p = 2$, la seconda colonna ha ordine di convergenza $p = 4$, la terza $p = 6$ e così via, a patto, comunque, che la funzione sia sufficientemete regolare in quanto i coefficienti C_{2j} sono definiti a partire dalle derivate.

2.7 La quadratura multidimensionale

Il problema della quadratura multidimensionale è lo sviluppo e l'analisi di metodi numerici per il calcolo di:

$$I[f] = \int \ldots \int_\Omega f(x_1, \ldots, x_d)\, dx_1 \ldots dx_d, \tag{2.80}$$

dove $\Omega \subseteq \Re^d$ con $d > 1$ è una regione dello spazio euclideo d-dimensionale e $f(x_1, \ldots, x_d)$ è una funzione integrabile in Ω. Analogamente al caso monodimensionale ($d = 1$) le formule di quadratura utilizzate (talvolta per $d > 1$ chiamate *formule di cubatura*) hanno espressione:

$$Q[f] = \sum_{i=1}^{n} A_i f(\underline{x}_i)$$

dove $\underline{x}_i$ è un punto dello spazio euclideo d-dimensionale, mentre l'errore di discretizzazione ha espressione:

$$E[f] = \int \ldots \int_\Omega f(x_1, \ldots, x_d)\, dx_1 \ldots dx_d - \sum_{i=1}^{n} A_i f(\underline{x}_i)$$

Benché molte idee di base della quadratura multidimensionale siano dirette estensioni di quelle in una dimensione, essa pone problemi (sia computazionali sia teorici) diversi. Uno di questi è legato alle caratteristiche dei domini di integrazione Ω su cui è possibile definire $I[f]$. Infatti mentre per il caso monodimensionale è possibile definire solo tre tipi di domini (intervalli chiusi e limitati, intervalli limitati solo superiormente o inferiormente ed intervalli illimitati), in più dimensioni è possibile integrare su sottospazi di $\Re^d$ come un ipercubo, una sfera, un simplesso o su una qualunque varietà algebrica come curve e superfici. La strategia che viene seguita consiste nello sviluppare formule per i domini più semplici e, quando possibile, effettuare per regioni diverse opportune trasformazioni per ricondurle a domini per cui le formule sono note.

Un altro problema caratteristico della quadratura multidimensionale è il numero di nodi necessario per ottenere l'accuratezza richiesta. Tale numero, infatti, in alcuni casi cresce esponenzialmente con il numero di dimensioni e ciò rende la scelta della formula di quadratura da utilizzare particolarmente delicata, poiché, per dimensioni elevate, l'algoritmo di quadratura può risultare impraticabile essendo, la sua complessità, non polinomiale. In questi casi, inoltre, l'errore di round-off può avere influenze notevoli sul risultato.

In questo paragrafo si presentano alcuni esempi significativi di formule di quadratura multidimensionale nel caso in cui $\Omega = [-1, 1]^d$.

2.7.1 Formule prodotto

Uno dei metodi più semplici ed immediati per la costruzione di formule per la quadratura multidimensionale è quello di comporre una o più formule di quadratura monodimensionale mediante prodotto cartesiano. Sia, ad esempio, $d = 2$ e siano $R[f]$ e $S[f]$ due formule di quadratura in una dimensione:

$$R[f] = \sum_{i=1}^{n_1} A_i f(x_i), \qquad S[f] = \sum_{j=1}^{n_2} B_j f(y_j).$$

Definizione 2.7.1. (Prodotto cartesiano)
Se si indicano con $X = \{x_1, \ldots, x_{n_1}\}$ *e* $Y = \{y_1, \ldots, y_{n_2}\}$ *rispettivamente gli insiemi dei nodi delle due formule di quadratura, si definisce* **prodotto cartesiano** *di* X *e* Y *l'insieme delle coppie:*

$$X \times Y = \{(x_i, y_j), i = 1, \ldots, n_1, \;\; j = 1, \ldots, n_2\}.$$

Definizione 2.7.2. (Formula prodotto)
Si definisce **formula prodotto** *la formula di quadratura definita sui nodi di* $X \times Y$*:*

$$(R \times S)[f] = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} A_i B_j f(x_i, y_j).$$

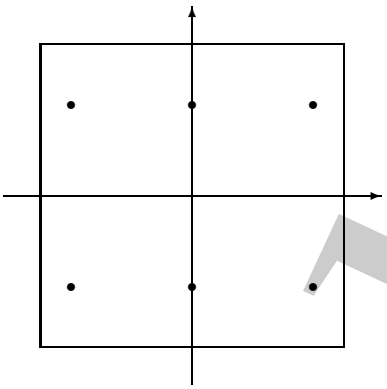

Figura 2.3: Nodi della formula prodotto di Gauss con sei nodi nella regione $[-1,1] \times [-1,1]$.

Tale formula richiede $n_1 \cdot n_2$ valutazioni della funzione integranda.

Se le due formule di quadratura $R[f]$ e $S[f]$ hanno grado di precisione algebrico rispettivamente p_1 e p_2 allora si ha:

$$E[f] = \int_{-1}^{1}\int_{-1}^{1} x^r y^s \, dx \, dy - \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} A_i B_j x_i^r y_j^s =$$

$$\int_{-1}^{1} x^r \, dx \int_{-1}^{1} y^s \, dy - \sum_{i=1}^{n_1} A_i x_i^r \sum_{j=1}^{n_2} B_j y_j^s = 0$$

$\forall r \leq p_1$ e $\forall s \leq p_2$. Si ha quindi che la formula prodotto è esatta per tutti i monomi $x^r y^s$ con $r \leq p_1$ e $s \leq p_2$ cioè essa ha grado di precisione algebrico $p = \min(p_1, p_2)$ anche se avrà errore nullo per qualche monomio di grado $p_1 + p_2$[19]. In Figura 2.3 è mostrata la disposizione dei nodi nel quadrato unitario $[-1,1] \times [-1,1]$ nel caso in cui $R[f]$ è una formula di Gauss con tre nodi e $S[f]$ è una formula di Gauss con due nodi.

Il metodo esposto è facilmente generalizzabile ad un numero di dimensioni qualunque, per cui le formule prodotto sono molto utilizzate. Esse hanno però lo svantaggio che il numero di nodi cresce esponenzialmente con la dimensione d. Ad esempio, se si utilizza una formula in $d = 20$ dimensioni, in ognuna delle quali si utilizzano $n = 2$ nodi, si hanno complessivamente $2^{20} \simeq 10^6$ valutazioni di funzione della funzione integranda. Per tale motivo le formule prodotto sono generalmente basate su formule particolarmente precise (ad esempio le formule di Gauss) ed utilizzate per un numero di dimensioni molto limitato ($d \leq 5$). Ad esempio la routine automatica `D01AJF` della libreria NAg basata sulla formula trapezoidale prodotto è in grado di integrare funzioni fino ad un massimo di quattro variabili.

[19]La definizione di grado di precisione algebrico per una formula di quadratura multidimensionale è del tutto analoga al caso monodimensionale: si dice che $Q[f]$ ha grado di precisione p se è esatta per i monomi algebrici multidimensionali di grado al più p.

2.7.2 Formule monomiali

Un approccio diverso per lo sviluppo di formule per la quadratura multidimensionale è quello di costruire formule esatte per un insieme di monomi di grado fissato, cioè tale che

$$E[f] = 0 \qquad \forall f(x_1, \ldots, x_d) = x_1^{i1} \cdots x_d^{id} \in \Pi_p, \qquad i1 + \ldots + id = p$$

dove Π_p è l'insieme di polinomi di grado al più p in d variabili. Le formule che si ottengono in questo modo sono dette **formule monomiali**; tra le più note vi è la famiglia di formule di Genz e Malik [14]. Sia $P^{(m,d)}$ l'insieme di tutte le distinte d-partizioni degli interi $0, 1, \ldots, m$ con componenti non crescenti, cioè

$$P^{(m,d)} = \{(p_1, \ldots, p_d) : m \geq p_1 \geq \ldots \geq p_d \geq 0, |\mathbf{p}| \leq m\},$$

dove $|\mathbf{p}| = \sum_{i=1}^{d} p_i$. Ad esempio per m=3 e d=4 si ha:

$$\begin{aligned} P^{(3,4)} \;=\; & \{(0,0,0,0), (1,0,0,0), (2,0,0,0), (1,1,0,0), (3,0,0,0), (2,1,0,0), \\ & (1,1,1,0)\} \end{aligned}$$

Fissati quindi m+1 valori reali distinti $\lambda_0 = 0, \lambda_1, \ldots, \lambda_m$ (generatori) si pone $\boldsymbol{\lambda}_p = \{\lambda_{p_1}, \ldots, \lambda_{p_d}\}$ e si definisce *formula base pienamente simmetrica* la somma:

$$f[\boldsymbol{\lambda}_p] = \sum_{q \in \Pi_p} \sum_{\mathbf{s}} f(s_1 \lambda_{q_1}, \ldots, s_d \lambda_{q_d}),$$

dove Π_p è l'insieme di tutte le permutazioni del generico $\mathbf{p} \in P^{(m,d)}$ e la somma interna è eseguita su tutte le combinazioni di segno delle componenti dei nodi che si ottengono per $s_i = \pm 1$. Tale somma è quindi effettuata mediante tutte le permutazioni e cambio di segno per i valori $\lambda_{q_i} \neq 0$. Per esempio con $d = 4$, $m = 2$ e $\mathbf{p} = (2, 0, 0, 0)$ si ha:

$$\begin{aligned} f[\boldsymbol{\lambda}_p] \;=\; & f(\lambda_2, 0, 0, 0) + f(-\lambda_2, 0, 0, 0) + f(0, \lambda_2, 0, 0) + f(0, -\lambda_2, 0, 0) + \\ & + f(0, 0, \lambda_2, 0) + f(0, 0, -\lambda_2, 0) + f(0, 0, 0, \lambda_2) + f(0, 0, 0, -\lambda_2). \end{aligned}$$

Infine posto $\underline{\delta} = \{\delta, \ldots, \delta\}$, con δ positivo, la formula di quadratura $R^{(m,d)}[f]$ è definita mediante:

$$R^{(m,d)}[f] = \sum_{p \in P^{(m-1,d)}} w_p^{(m,d)} f[\boldsymbol{\lambda}_p] + W^{(m,d)} f[\underline{\delta}].$$

Si può notare che i generatori λ_i sono indipendenti da m, per cui la formula $R^{(m,d)}$ utilizza tutti i nodi di $R^{(m-1,d)}$, che utilizza tutti i nodi di $R^{(m-2,d)}$ e così via. Tali formule costituiscono quindi una famiglia di formule innestate e per tale motivo esse sono tra le più utilizzate nella realizzazione di routine di quadratura multidimensionale efficienti. Il calcolo dei parametri che caratterizzano la formula richiede la risoluzione di sistemi di equazioni non lineari. Mediante un'opportuna scelta di W, λ_i e δ è possibile

determinare i pesi w_p in modo che la formula $R^{(m,d)}$ abbia grado di precisione $2m+1$ [14]. Le formule con grado di precisione al più 15 sono utilizzate per $6 \leq d \leq 20$.

Tra le routine disponibili per la quadratura multidimensionale, si ricordano la routine `DCUHRE` della collezione ACM *Transactions on Mathematical Software* e le routine `D01EAF` e `D01FCF` della libreria del NAg.

2.7.3 Metodi Monte Carlo

Quando la dimensione d aumenta ($d \geq 20$) i metodi precedentemente descritti sono computazionalmente impraticabili e si fa ricorso a tecniche di tipo numerico-statistico. Ad esempio, nel dominio $B = [0,1]^d$ l'integrale:

$$I[f] = \int_B f(t)dt = \int_{R^d} \chi_B(t) f(t)dt = \mu(f)$$

viene interpretato come il valore atteso $\mu(f)$ della funzione f, vista come variabile casuale, con densità di probabilità $\chi_B(t)$, dove $\chi_B(t)$ è la funzione caratteristica della regione B. Per simulare l'andamento della variabile casuale f, si scelgono gli n nodi in maniera casuale su $[0,1]^d$. In tal modo si hanno i cosiddetti *metodi Monte Carlo* per la quadratura multidimensionale:

$$Q[f] = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \tag{2.81}$$

Per la legge forte dei grandi numeri, se le n variabili casuali sono uniformemente distribuite si ha che

$$\mathcal{P}\left(\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(x_k) = I[f] \right) = 1.$$

dove con $\mathcal{P}(\mathcal{E})$ si è indicata la probabilità dell'evento $\mathcal{E}$. Si ha quindi che la probabilità che la (2.81) sia una buona approssimazione di $I[f]$ può essere resa arbitrariamente prossima a 1 se si sceglie n sufficientemente grande. Inoltre se esiste anche $I[f^2]$ si ha, per il teorema centrale del limite, che

$$\mathcal{P}\left(|Q[f] - I[f]| \leq \frac{\lambda\sigma}{\sqrt{n}} \right) = \frac{1}{\sqrt{2\pi}} \int_{-\lambda}^{\lambda} e^{-x^2/2}\, dx + \mathcal{O}\left(\frac{1}{\sqrt{n}} \right).$$

dove $\sigma = [I[f^2] - (I[f])^2]^{1/2}$ è la deviazione standard.

In genere per ogni fissato λ tale espressione è presa come una misura dell'errore $E[f]$ prodotto da un metodo Monte Carlo. Per $\lambda \geq 2$ il primo addendo al secondo membro è circa uguale a 1, per cui è possibile porre

$$|E[f]| = |I[f] - Q[f]| \simeq \mathcal{O}(n^{-1/2})$$

Questo errore è tipico dei metodi Monte Carlo e ha lo svantaggio di decrescere lentamente al crescere di n. Un altro aspetto negativo dei metodi Monte Carlo è che sequenze realmente casuali non sono computazionalmente disponibili.

Osservazione 2.3. *Nello sviluppo di software basato sui metodi Monte Carlo, per la determinazione dei nodi si utilizzano sequenze di numeri così dette pseudo casuali, calcolate, deterministicamente, mediante algoritmi chiamati impropriamente "generatori di numeri casuali".*

È da notare che l'errore prodotto da un metodo Monte Carlo, è indipendente dalla dimensione e dalla regolarità della funzione integranda. Per tale motivo i metodi Monte Carlo sono particolarmente utili nel caso di integrali in un gran numero di dimensioni o di funzioni integrande estremamente irregolari. Un esempio di routine basata su un metodo Monte Carlo è la routine `D01GBF` della libreria del NAg.

2.7.4 Le Lattice Rule

Le Lattice Rule possono essere considerate una particolare generalizzazione della formula trapezoidale composita nello spazio a più dimensioni. Esse hanno un' elevata velocità di convergenza nel caso in cui la funzione integranda sia sufficientemente regolare e periodica in ogni sua componente nel cubo unitario $\Omega = [0,1]^d$ (vedi paragrafo 2.6). Inoltre, come la formula trapezoidale composita, utilizzano come nodi di integrazione un insieme estremamente regolare di punti, le cosiddette *Lattice* di integrazione:

Definizione 2.7.3. (Lattice)
Si definisce **Lattice** *di integrazione nello spazio a d dimensioni un sottoinsieme discreto dello spazio euclideo $\Re^d$, che sia chiuso rispetto all'addizione (cioè tale che la somma di due elementi della lattice è ancora un elemento della lattice) e che contenga al suo interno l'insieme dei vettori d-dimensionali a componenti intere $\mathcal{Z}^d = \{(z_1, ..., z_d) : z_i \in \mathcal{Z}\}$.*

Definizione 2.7.4. (Lattice Rule)
Si definisce **Lattice Rule** *una formula di quadratura del tipo (2.81) che utilizzi come nodi i punti di una Lattice che appartengono al cubo semiaperto $[0,1[^d$.*

♣ **Esempio 2.18.** Nello spazio a $d = 2$ dimensioni la formula:

$$Q[f] = \frac{1}{5}\left[f(0,0) + f\left(\frac{1}{5},\frac{2}{5}\right) + f\left(\frac{2}{5},\frac{4}{5}\right) + f\left(\frac{3}{5},\frac{1}{5}\right) + f\left(\frac{4}{5},\frac{3}{5}\right)\right]$$

è una Lattice Rule. Si noti infatti che l'insieme dei punti:

$$(x,y) = \alpha\left(\frac{1}{5},\frac{2}{5}\right) + \beta\left(\frac{3}{5},\frac{1}{5}\right) \quad \alpha,\beta \text{ interi}$$

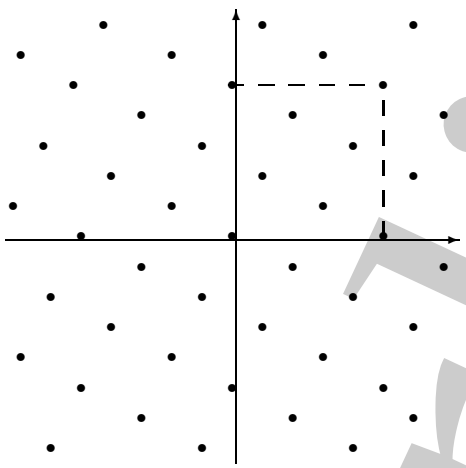

Figura 2.4: Esempio di Lattice in $d = 2$ dimensioni generata da $(\frac{1}{5}, \frac{2}{5})$ e $(\frac{3}{5}, \frac{1}{5})$.

è un insieme chiuso rispetto all'addizione ed inoltre contiene al suo interno $\mathcal{Z}^2$ (Figura 2.4). I punti $(\frac{1}{5}, \frac{2}{5})$ e $(\frac{3}{5}, \frac{1}{5})$ sono chiamati **generatori** della Lattice Rule.

Nella figura sono rappresentati i nodi della Lattice ed è messo in evidenza il quadrato unitario $[0, 1[^2$. ♣

Un altro legame delle Lattice Rule con la formula trapezoidale composita consiste nel fatto che è possibile esprimere l'errore in termini dei coefficienti di Fourier della funzione integranda (per la trapezoidale composita si veda il paragrafo 2.6). Se si suppone infatti che la funzione $f(\underline{x}) = f(x_1, ..., x_d)$ sia sviluppabile in serie di Fourier, cioè:

$$f(\underline{x}) = \sum_{\underline{h} \in \mathcal{Z}^d} a_h(f) e^{2\pi i \underline{h} \cdot \underline{x}} \tag{2.82}$$

dove $\underline{h} \cdot \underline{x}$ è l'usuale prodotto scalare in $\Re^d$ e gli $a_h(f)$ sono i coefficienti di Fourier della funzione f, applicando la formula (2.81) alla (2.82) si ha:

$$Q[f] = \sum_{\underline{h} \in \mathcal{Z}^d} a_h(f) Q[e^{2\pi i \underline{h} \cdot \underline{x}}]$$

Poiché è possibile verificare, analogamente a come mostrato nel paragrafo 2.6, che

$$Q[e^{2\pi i \underline{h} \cdot \underline{x}}] = \begin{cases} 1 & \text{se } \underline{h} \cdot \underline{x} \in \mathcal{Z} \\ 0 & \text{altrimenti} \end{cases}$$

l'errore $E[f]$ è:

$$E[f] = I[f] - Q[f] = \sum_{\underline{h} \cdot \underline{x} \in \mathcal{Z}, \underline{h} \neq 0} a_h(f)$$

Si noti infine che la formula trapezoidale composita prodotto è una particolare Lattice Rule, in quanto, ad esempio, la formula prodotto $(T_m \times T_m)[f]$ può essere generata dai nodi $(0, \frac{1}{m})$ e $(\frac{1}{m}, 0)$. Di tali metodi esistono alcune implementazioni riportate sulla rivista specializzata di software matematico ACM *Transactions on Mathematical Software*.

2.8 Le formule di quadratura ottimali

Come accennato nel paragrafo 2.1.1, le formule ottimali minimizzano il massimo errore di discretizzazione in uno spazio di funzioni $\mathcal{G}$, prefissato. Considerata la formula di quadratura:

$$Q[f] = \sum_{i=1}^{n} A_i f(x_i),$$

esatta per i poinomi di grado al piú $r-1$, ovvero tale che

$$E[f] = I[f] - Q[f] = 0 \quad \forall f \in \Pi_{r-1}$$

calcoliamone i pesi A_k ed i nodi x_k in modo che:

$$\min_{A_i, x_i} \sup_{f \in \mathcal{G}} |E[f]| = \min_{A_i, x_i} \sup_{f \in \mathcal{G}} |I[f] - Q[f]|$$

sull'insieme $\mathcal{G} \equiv C^r([a,b])$.
Il teorema di Peano fornisce una maggiorazione per $E[f]$, con $f \in C^r([a,b])$. Infatti, dall'espressione fornita dal Teorema di Peano:

$$E[f] = \int_a^b f^{(r)}(t) K(t)\, dt$$

dove $f \in C^r([a,b])$ e

$$K(t) = \frac{1}{r!} E_x[(x-t)_+^r],$$

applicando la disuguaglianza di Schwarz, si ha:

$$|E[f]| \leq \left(\int_a^b [f^{(r)}(t)]^2\, dt \right)^{1/2} \left(\int_a^b K^2(t)\, dt \right)^{1/2}.$$

Supponendo che:

$$\left(\int_a^b [f^{(r)}(t)]^2\, dt \right)^{1/2} \leq M$$

si ha $|E[f]| < MJ$, dove

$$J = \left[\int_a^b K^2(t)\, dt \right]^{1/2} = \sigma(A_k, x_k) \tag{2.83}$$

La maggiorazione ottenuta per $E[f]$ dipende dallo spazio di funzioni $C^r([a,b])$ attraverso M e dalla formula di quadratura attraverso $J= \sigma(A_k, x_k)$ che è funzione delle $2n$ variabili A_k e x_k. Minimizzare $E[f]$ equivale quindi a minimizzare la funzione $\sigma(A_k, x_k)$.

Una delle famiglie più note tra le formule ottimali si ottiene fissando l'insieme dei nodi $X = \{x_1, \ldots, x_n\}$.

Definizione 2.8.1. (Formula ottimale nel senso di Sard)
La formula di quadratura

$$Q[f] = \sum_{k=1}^{n} A_k^* f(x_k) \tag{2.84}$$

tale che

$$\sigma(A_k^*, x_k) = \min_{A_k} \sigma(A_k, x_k)$$

con il minimo calcolato sull'insieme

$$\mathcal{Q} = \{A_k \, : Q[f] = I[f], \quad \forall f \in \Pi_{r-1}\} \tag{2.85}$$

è detta **formula ottimale nel senso di Sard di grado** $r-1$.

Tali formule sono caratterizzate dal:

Teorema 2.8.1. [Teorema di Schoenberg]
Sia $X = \{x_1 < \ldots < x_n\}$. È possibile dimostrare che

1) esiste uno ed un solo funzionale

$$Q[s] = \sum_{k=1}^{n} A_k s(x_k)$$

con i pesi soddisfacenti la (2.85) e tale che

$$I[s] = Q[s] \quad \forall s \in \mathcal{S}_{2r-1}(X)$$

dove s è la spline naturale di grado $2r-1$ sull'insieme dei nodi X.

2) Tale funzionale è l'unica formula ottimale di Sard di grado $r-1$ relativa all'insieme dei nodi X.

Dimostrazione

1) Si può innanzitutto notare che, per le ipotesi fatte, il funzionale

$$Q[f] = \sum_{k=1}^{n} A_k f(x_k)$$

è univocamente determinato da:

$$\begin{array}{ll} a) & Q[f] = I[f] \quad \forall f \in \Pi_{r-1} \\ b) & \text{esiste uno e un solo } q \in \Pi_{r-1} \text{ tale che } E[(x-x_i)_+^{2r-1}] = q(x_i) \end{array}$$

Posto quindi:

$$\begin{aligned} I(x^j) &= \alpha_j \quad j = 0, 1, \ldots, r-1 \\ I[(x-x_i)_+^{2r-1}] &= \beta_i \quad i = 1, 2, \ldots, n \end{aligned}$$

dalle condizioni a) e b) si ottengono rispettivamente

$$\begin{array}{ll} a') & \sum_{k=1}^n A_k x_k^j = \alpha_j \quad j = 0, 1, \ldots, r-1 \\ b') & \beta_i - \sum_{k=1}^n A_k (x_k - x_i)_+^{2r-1} = q(x_i) \quad i = 1, 2, \ldots, n \end{array}$$

che costituiscono un sistema lineare di ordine $n+r$, le cui incognite sono i coefficienti A_k, $k = 1, \ldots, n$ e gli r coefficienti di $q(x)$. La matrice dei coefficienti risulta essere di tipo Vandermonde, per cui si prova facilmente che il sistema ammette un'unica soluzione. Dimostriamo che questo funzionale è tale che $Q[f] = I[f]$, $\forall f \in \mathcal{S}_{2r-1}(X)$.
Sia infatti $f(x)$ la generica spline di grado $2r-1$. Si ha, per le proprietà delle spline naturali,

$$\begin{array}{ll} I) & f(x) = p(x) + \sum_{i=1}^n \alpha_i (x-x_i)_+^{2r-1} \quad p \in \Pi_{r-1} \\ II) & \sum_{i=1}^n \alpha_i x_i^j = 0 \quad j = 0, \ldots, r-1 \end{array}$$

Applicando il funzionale $I[f]$ ad $f \in \mathcal{S}_{2r-1}(X)$, per la I) si ha

$$I[f] = I[p] + \sum_{i=1}^n \alpha_i I[(x-x_i)_+^{2r-1}]$$

ed applicando la condizione a) si ha

$$I[f] = Q[p] + \sum_{i=1}^n \alpha_i \beta_i$$

Dalla condizione b'), inoltre:

$$E[(x-x_i)_+^{2r-1}] = \beta_i = Q[(x-x_i^{2r-1})_+] + q(x_i)$$

e quindi

$$I[f] = Q[p] + \sum_{i=1}^n \alpha_i Q[(x-x_i)_+^{2r-1}] + \sum_{i=1}^n \alpha_i q(x_i)$$

e per la II) si ha

$$I[f] = Q[p] + \sum_{i=1}^n \alpha_i Q[(x-x_i)_+^{2r-1}] = Q[f]$$

cioè la 1).

2) Sia $Q[f]$ tale che $Q[f] = I[f] \quad \forall f \in \mathcal{S}_{2r-1}$ e sia $Q'[f]$ tale che $Q'[f] = I[f] \quad \forall f \in \Pi_{r-1}$. Bisogna dimostrare che

i) $\int_a^b K(t)^2\,dt \leq \int_a^b K'(t)^2\,dt$

ii) l'uguaglianza sussiste se e solo se $Q = Q'$

Dalle posizioni fatte si ha che

$$\begin{array}{ll} I[f] - Q[f] = E[f] = 0 & \forall f \in \Pi_{r-1} \\ I[f] - Q'[f] = E'[f] = 0 & \forall f \in \Pi_{r-1} \end{array}$$

e per il teorema di Peano si ha

$$\begin{array}{ll} E[f] = \int_a^b f^{(r)}(t) K(t)\, dt & \forall f \in C^r([a,b]) \\ E'[f] = \int_a^b f^{(r)}(t) K'(t)\, dt & \forall f \in C^r([a,b]) \end{array}$$

Sia ora $\sigma(t)$ la funzione:

$$\sigma(t) = K(t) - K'(t) = \frac{1}{(r-1)!}(E - E')_x[(x-t)_+^{r-1}] =$$

$$= \frac{1}{(r-1)!} \sum_{k=1}^{n} (A'_k - A_k)(x_k - t)_+^{r-1}$$

Si può notare che per $t \geq x_n$ le potenze troncate sono nulle, per cui $\sigma(t) = 0$ e per $t \leq x_1$ le potenze troncate sono potenze ordinarie e si ha

$$(E - E')_x[(x-t)_+^{r-1}] = (E - E')_x[p] \quad \text{con} \quad p \in \Pi_{r-1}$$

cioè $E - E'$ è applicato ad un polinomio di grado $r-1$ per cui $\sigma(t) = 0$. Inoltre è possibile porre

$$x^j = x_+^j + (-1)^j (-x)^j,$$

per cui:

$$\sigma(t) = p(t) + \sum_{k=1}^{n} c_k (t - x_k)_+^{j-1} \quad p \in \Pi_{r-1}$$

Per le proprietà viste, si ha che la funzione $\sigma(t)$ può essere vista come la derivata r-ma di una spline $s(t)$ di grado $2r-1$, cioè

$$\sigma(t) = s^{(r)}(t), \quad s \in \mathcal{S}_{2r-1}(X)$$

Poiché

$$K'(t)^2 = K(t) + \sigma(t)^2 - 2K(t)\sigma(t)$$

per dimostrare la **i)** basta dimostrare che

$$\int_a^b K(t)\sigma(t)\, dt = 0$$

Ciò è immediato, infatti per la 1) si ha

$$I[s] = Q[s], \quad s^{(r)}(x) = \sigma(t)$$

da cui

$$I[s] - Q[s] = E[s] = 0$$

e per il teorema di Peano

$$0 = E[s] = \int_a^b s^{(r)}(t) K(t)\, dt = \int_a^b \sigma(t) K(t)\, dt$$

Dimostriamo infine che

$$\int_a^b K(t)\, dt = \int_a^b K'(t)\, dt \iff Q[f] = Q'[f], \quad \forall f \in C^r([a,b])$$

Se $Q[f] = Q'[f], \quad \forall f \in C^r([a,b])$, allora $E[f] = E'[f]$ e, quindi, $K(t) = K'(t)$. Viceversa, se $\int K(t)^2 = \int K'(t)^2$ si ha $\int \sigma(t)^2 = 0$ e quindi $\sigma(t) = 0$. Cioè $Q[f] = Q'[f], \quad \forall f \in C^r([a,b])$. Ciò completa la dimostrazione.

■

Da tale teorema discende che

i) fissati i nodi, esiste una sola formula ottimale di Sard di grado $r-1$;

ii) le formule di Sard sono esatte di grado $r-1$ anche su tutto lo spazio $\mathcal{S}_{2r-1}$ (si ricordi che $\Pi_{r-1} \subseteq \mathcal{S}_{2r-1}$);

iii) tali formule minimizzano l'errore sull'insieme delle funzioni di classe $C^r([a,b])$.

I pesi A_k^* delle formule di Sard sono la soluzione di un sistema lineare che è in genere malcondizionato. È possibile ottenere però il valore che il funzionale $Q[f]$ assume applicato ad una funzione $f(x) \in C^{r-1}([a,b])$. Infatti, sussiste il seguente:

Corollario 2.8.1. *Fissato $X = \{x_1 < \ldots < x_n\}$ il funzionale $Q[s]$ tale che*

$$I[s] = Q[s], \quad \forall s(t) \in \mathcal{S}_{2r-1}(X)$$

è l'unico funzionale che gode della proprietà

$$Q[f] = I[s_f] \quad \forall f(x) \in C^{r-1}([a,b])$$

dove $s_f \in \mathcal{S}_{2r-1}(X)$ è la spline naturale interpolante $f(x)$ nei nodi dell'insieme X, cioè

$$s_f(x_i) = f(x_i) \quad i = 1, \ldots, n$$

Sulla base del corollario precedente, si può dire che per calcolare un'approssimazione $Q[f]$ di $I[f]$ mediante una formula di Sard, è sufficiente calcolare il valore $I[s_f]$.

2.9 Le formule Filon-like

Consideriamo il problema del calcolo di integrali trigonometrici del tipo [27]:

$$I_c[f] = \int_0^b f(x)\cos(\omega x)\,dx \qquad I_s[f] = \int_0^b f(x)\sin(\omega x)\,dx \tag{2.86}$$

♣ **Esempio 2.19.** Si consideri il problema del calcolo dell'integrale:

$$I_c[f] = \int_0^{2\pi} f_\omega(x)\,dx \qquad \text{con} \qquad \omega = 1, 2, \ldots, 100 \tag{2.87}$$

dove

$$f_\omega(x) = x^2\cos(\omega x)\ . \tag{2.88}$$

Consideriamo l'errore

$$E_c[f] = I_c[f] - \widetilde{G[f]} \ .$$

$E_c[f]$ è l'errore che si commette, sostituendo all'integrale $I_c[f]$, il risultato computazionale $\widetilde{G[f]}$ ottenuto mediante la routine `DQNG.f` di `QUADPACK` [31], utilizzando la doppia precisione dell'aritmetica standard IEEE. Tale routine è basata sulla formula di Gauss-Kronrod $G[f]$ a 87 nodi.
Osservando la Figura 2.19 e la Tabella 2.13, che riportano il comportamento dell'errore $E_c[f]$ al variare di ω, si nota che:

- per $\omega \leq 40$ la stima dell'integrale (2.87) puó ritenersi attendibile (fino a cinque cifre significative);
- per $40 < \omega \leq 53$ l'errore cresce rapidamente fino a 10^1 (perdita di ogni cifra significativa);
- infine, per $\omega > 53$, l'errore è dell'ordine di 10^1, ovvero la stima dell'integrale ottenuta è completamente inaccurata.

In altre parole le formule di Gauss, pur se con alto grado di precisione algebrico, non risultano adatte per il calcolo dell'integrale (2.86) al crescere di ω, ovvero quando la funzione integranda (2.88) diventa fortemente oscillante.

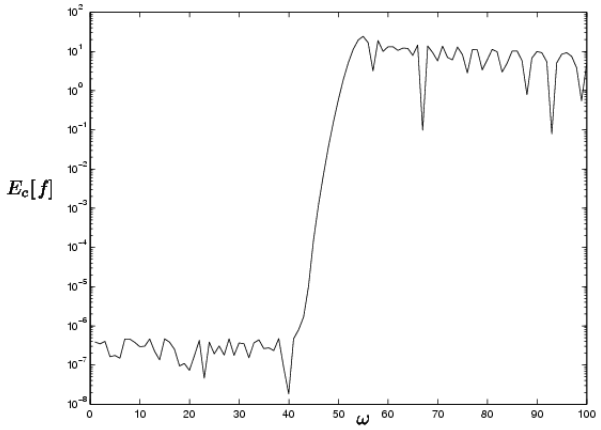

Figura 2.5: Errore $E_c[f] = I_c[f] - \widetilde{G[f]}$ per $\omega = 1, 2, \ldots, 100$.

Tale comportamento si giustifica osservando che, applicando la routine `DQNG.f`, la massima distanza tra due nodi consecutivi è

$$h_{max} = 0.117345\ldots \ .$$

Pertanto, volendo che la funzione $f_\omega(x)$ sia valutata almeno una volta nel periodo $T = \frac{2\pi}{\omega}$, deve risultare

$$h_{max} < T \ . \tag{2.89}$$

In effetti se $\omega \leq 53$ si ha

$$T \geq 0.11855\ldots$$

e la condizione (2.89) è verificata, mentre se $\omega > 53$ essa non è verificata.
L'esempio mostra quindi che, in questi casi, è opportuno utilizzare formule di quadratura per le quali il passo di discretizzazione h_{max} si possa scegliere indipendente dal parametro ω. In tale senso operano le formule Filon-Like.

♣

ω	$I_c[f]$		$\widetilde{G[f]}$		$E_c[f] = I_c[f] - \widetilde{G[f]}$
5	5.026548	E-01	5.026550	E-01	1.754256 E-07
10	1.256637	E-01	1.256640	E-01	2.938564 E-07
15	5.585053	E-02	5.585100	E-02	4.639362 E-07
20	3.141592	E-02	3.141600	E-02	7.346411 E-08
25	2.010619	E-02	2.010600	E-02	1.929829 E-07
30	1.396263	E-02	1.396300	E-02	3.659840 E-07
35	1.025826	E-02	1.025800	E-02	2.617259 E-07
40	7.853981	E-03	7.854000	E-03	1.836603 E-08
45	6.205615	E-03	6.051000	E-03	1.546151 E-04
50	5.026548	E-03	5.941700	E-01	5.891434 E-01
55	4.154172	E-03	-2.426848	E+01	2.427264 E+01
60	3.490658	E-03	1.311877	E+01	1.311528 E+01
65	2.974288	E-03	-7.952352	E+00	7.955326 E+00
70	2.564565	E-03	-5.771141	E+00	5.773705 E+00
75	2.234021	E-03	8.331598	E+00	8.329363 E+00
80	1.963495	E-03	6.078054	E+00	6.076090 E+00
85	1.739290	E-03	-1.015912	E+01	1.016086 E+01
90	1.551403	E-03	9.936548	E+00	9.934997 E+00
95	1.392395	E-03	-8.505070	E+00	8.506463 E+00
100	1.256637	E-03	4.693296	E+00	4.692039 E+00

Tabella 2.13: Errore $E_c[f] = I_c[f] - \widetilde{G[f]}$ per $\omega = 5, 10, \ldots, 100$.

L'idea alla base delle formule di Filon-like è quella di assumere $\cos(\omega x)$ e $\sin(\omega x)$ come funzioni peso rispetto alle quali integrare la funzione f e determinare gli A_i, imponendo che l'errore di discretizzazione $E_c[f]$ sia nullo per tutti i polinomi al più di grado q rispetto alla funzione peso $\cos(\omega x)$.

♣ **Esempio 2.20.** Si determini la formula $Q_c[f]$ esatta per i polinomi appartenenti a Π_2, rispetto alla funzione peso $\cos(x)$ nell'intervallo $[0, 2\pi]$. Per garantire la compatibilità del sistema si deve avere $n = 3$, cioè:

$$I_c[f] = \int_0^{2\pi} f(x)\cos(x)\,dx \simeq \sum_{i=1}^{3} A_i f(x_i)$$

Per semplicità scelti i 3 nodi equidistanziati, con distanza $x_i - x_{i-1} = \pi$, cioè:

$$x_1 = 0 \quad x_2 = \pi \quad x_3 = 2\pi$$

imponendo $E_c[f] = 0$, si ottiene il sistema lineare di tre equazioni in tre incognite:

$$Q_c[x^r\cos(x)] = I_c[x^r\cos(x)] \quad r = 0, 1, 2$$

cioè

$$\begin{cases} A_1 & +A_2 & +A_3 & = \int_0^{2\pi}\cos(x)\,dx = 0 \\ & \pi A_2 & +2\pi A_3 & = \int_0^{2\pi} x\cos(x)\,dx = 0 \\ & \pi^2 A_2 & +4\pi^2 A_3 & = \int_0^{2\pi} x^2\cos(x)\,dx = 4\pi \end{cases} \tag{2.90}$$

I pesi della formula $Q_c[f]$ costituiscono la soluzione del sistema (2.90), cioè:

$$A_1 = \frac{2}{\pi} \quad A_2 = \frac{-4}{\pi} \quad A_3 = \frac{2}{\pi}$$

♣

Proposizione 2.9.1. *Sia $n = 3$. Siano fissati i nodi x_i, $i = 1, \ldots, 3$ equidistanziati, cioè:*

$$x_1 = 0 \quad x_2 = h \quad x_3 = 2h = b, \quad \textit{con} \quad h = \frac{b}{2}, \tag{2.91}$$

e sia posto

$$\omega = \frac{\pi\alpha}{h} \tag{2.92}$$

con α numero reale opportuno. Allora la formula di quadratura esatta per i polinomi appartenenti a Π_2, rispetto alla funzione peso $\cos(\omega\, x)$, in un intervallo $[0, b]$, è:

$$\int_0^b f(x)\cos(\omega x)\,dx \;=\; Q_c[f] = b\,[L_1 f(0) + L_2 f(h) + L_3 f(b)] \tag{2.93}$$

con i coefficienti $L_i,\ i = 1, \ldots, 3$ *definiti nel seguente modo:*

$$\begin{aligned} L_1 &= \tfrac{2\pi\alpha\,\cos(2\pi\alpha) - 4\,\sin(2\pi\alpha) + 6\pi\alpha}{8\pi^3\alpha^3} \\ L_2 &= \tfrac{8\,\sin(2\pi\alpha) - 8\pi\alpha\,\cos(2\pi\alpha) - 8\pi\alpha}{8\pi^3\alpha^3} \\ L_3 &= \tfrac{2\pi\alpha - 4\,\sin(2\pi\alpha) + 6\pi\alpha\,\cos(2\pi\alpha) + 4\pi^2\alpha^2\,\sin(2\pi\alpha)}{8\pi^3\alpha^3} \end{aligned}$$

Dimostrazione Considerato il sisitema lineare ottenuto da $E[x^r cos(\omega\, x)] = 0,\ r = 0, 1, 2$, la matrice V ed il vettore dei termini noti del sistema sono, rispettivamente, uguali a:

$$V = \begin{pmatrix} 1 & 1 & 1 \\ 0 & h & 2h \\ 0 & h^2 & 4h^2 \end{pmatrix} \qquad \begin{aligned} \textstyle\int_0^b \cos(\omega x)\,dx &= b\,\tfrac{\sin(2\pi\alpha)}{2\pi\alpha} \\ \textstyle\int_0^b x\cos(\omega x)\,dx &= b^2\,\tfrac{\cos(2\pi\alpha) + 2\pi\alpha\,\sin(2\pi\alpha) - 1}{4\pi^2\alpha^2} \\ \textstyle\int_0^b x^2\cos(\omega x)\,dx &= b^3\,\tfrac{4\pi^2\alpha^2\,\sin(2\pi\alpha) + 4\pi\alpha\,\cos(2\pi\alpha) - 2\,\sin(2\pi\alpha)}{8\pi^3\alpha^3} \end{aligned}$$

La soluzione del sistema è quindi:

$$\begin{aligned} A_1 &= b\tfrac{2\pi\alpha\,\cos(2\pi\alpha) - 4\,\sin(2\pi\alpha) + 6\pi\alpha}{8\pi^3\alpha^3} &&= bL_1 \\ A_2 &= b\tfrac{8\,\sin(2\pi\alpha) - 8\pi\alpha\,\cos(2\pi\alpha) - 8\pi\alpha}{8\pi^3\alpha^3} &&= bL_2 \\ A_3 &= b\tfrac{2\pi\alpha - 4\,\sin(2\pi\alpha) + 6\pi\alpha\,\cos(2\pi\alpha) + 4\pi^2\alpha^2\,\sin(2\pi\alpha)}{8\pi^3\alpha^3} &&= bL_3 \end{aligned} \tag{2.94}$$

da cui segue la tesi. ■

Si noti che la formula determinata nell'esempio 2.20 si ottiene ponendo nella (2.94) $b = 2\pi$ (e quindi $h = \pi$) e $\alpha = 1$.
Analogamente a come si è determinata la formula di quadratura $Q_c[f]$ che approssima l'integrale $I_c[f]$, è possibile determinare la formula $Q_s[f]$ che approssima l'integrale $I_s[f]$. Si dimostra, infatti, la seguente:

Proposizione 2.9.2. *Sia* $n = 3$. *Siano fissati i nodi* $x_i,\ i = 1, \ldots, 3$ *equidistanziati, cioè:*

$$x_1 = 0 \quad x_2 = h \quad x_3 = 2h = b, \quad con \quad h = \frac{b}{2},$$

e sia posto

$$\omega = \frac{\pi\alpha}{h}$$

con α *numero reale opportuno. Allora la formula di quadratura* $Q_s[f]$ *esatta per i polinomi appartenenti a* Π_2*, rispetto alla funzione peso* $sin(\omega\, x)$*, in un intervallo* $[0, b]$*, è:*

$$\int_0^b f(x)\sin(\omega x)\,dx \simeq Q_s[f] = b\,[M_1 f(0) + M_2 f(h) + M_3 f(b)] \tag{2.95}$$

con i coefficienti M_i dati da:

$$\begin{aligned} M_1 &= \tfrac{4\pi^2\alpha^2+2\pi\alpha\,\sin(2\pi\alpha)+4\,\cos(2\pi\alpha)-4}{8\pi^3\alpha^3} \\ M_2 &= \tfrac{8-8\pi\alpha\,\sin(2\pi\alpha)-8\,\cos(2\pi\alpha)}{8\pi^3\alpha^3} \\ M_3 &= \tfrac{6\pi\alpha\,\sin(2\pi\alpha)+4\,\cos(2\pi\alpha)-4-4\pi^2\alpha^2\,\cos(2\pi\alpha)}{8\pi^3\alpha^3} \end{aligned} \tag{2.96}$$

Si noti che i coefficienti L_i e M_i, definiti dalle (2.94) e (2.96), dipendono dal parametro α (e quindi da ω).
Si vogliono determinare delle espressioni generali, valide per ogni valore del parametro ω. A tal fine sussiste il seguente:

Teorema 2.9.1. *Sia $n = 3$. Siano fissati i nodi $x_i,\ i = 1, \ldots, 3$ equidistanziati, cioè:*

$$x_1 = 0 \quad x_2 = h \quad x_3 = 2h = b, \quad con \quad h = \frac{b}{2}, \tag{2.97}$$

e sia posto

$$\omega = \frac{\pi\alpha}{h} \tag{2.98}$$

con α numero reale opportuno. È possibile determinare opportune costanti λ_k^i con $k = 1, \ldots, 3$ tali che i coefficienti L_i ed M_i delle (2.93) e (2.95) rispettivamente, si possano esprimere come:

$$L_i = \sum_{k=1}^{3} \lambda_k^i V_k^\alpha \qquad M_i = \sum_{k=1}^{3} \lambda_k^i W_k^\alpha \tag{2.99}$$

dove:

$$V_k^\alpha = \int_0^1 t^k \cos(2\pi\alpha t)\, dt$$

e W_k^α:

$$W_k^\alpha = \int_0^1 t^k \sin(2\pi\alpha t)\, dt$$

Dimostrazione Per come è stata costruita, la formula $Q_c[f]$ è esatta per tutti i polinomi $p(x) \in \Pi_2$ rispetto alla funzione peso $\cos(\omega x)$. In particolare essa sarà esatta per il polinomio interpolante di Lagrange $p(x)$ che nei nodi x_i, dati dalla (2.97), assume valori $f(x_i)$. Allora dall'uguaglianza:

$$\int_0^b p(x)\cos(\omega x)\, dx = \sum_{i=1}^{3} A_i p(x_i)$$

e ricordando che un'espressione del polinomio $p(x)$ è data da:

$$p(x) = \frac{(x-h)(x-2h)}{(-h)(-2h)} f(0) + \frac{x(x-2h)}{h(-h)} f(h) + \frac{x(x-h)}{(2h)(h)} f(b)$$

è facile verificare che i pesi A_i hanno espressione:

$$A_1 = \int_0^b \frac{(x-h)(x-2h)}{(-h)(-2h)} \cos(\omega x)\, dx \qquad A_2 = \int_0^b \frac{x(x-2h)}{h(-h)} \cos(\omega x)\, dx$$

i	k=1	k=2	k=3
1	1	-3	2
2	0	4	-4
3	0	-1	2

Tabella 2.14: Tabella delle λ_k^i per $n = 3$

$$A_3 = \int_0^b \frac{x(x-h)}{(2h)(h)} \cos(\omega x)\, dx \tag{2.100}$$

Prendendo in considerazione il primo degli integrali della (2.100), e utilizzando il cambio di variabile $x = 2ht$ si ha:

$$A_1 = 2h \int_0^1 \frac{(2ht-h)(2ht-2h)}{2h^2} \cos\left(\frac{\pi\alpha}{h} 2ht\right) dt = 2h \int_0^1 (2t-1)(t-1)\cos(2\pi\alpha t)\, dt$$

Ponendo quindi:

$$V_k^\alpha = \int_0^1 t^k \cos(2\pi\alpha t)\, dt$$

si ha:

$$L_1 = \frac{A_1}{2h} = \int_0^1 (2t^2 - 3t + 1)\cos(2\pi\alpha t)\, dt = 1\, V_1^\alpha - 3\, V_2^\alpha + 2\, V_3^\alpha$$

Procedendo in modo del tutto analogo, per gli altri coefficienti L_i ed M_i, è possibile determinare delle opportune costanti λ_k^i con $k = 1, .., 3$ tali che

$$L_i = \sum_{k=1}^{3} \lambda_k^i V_k^\alpha \qquad M_i = \sum_{k=1}^{3} \lambda_k^i W_k^\alpha$$

dove si è indicato con W_k^α la quantità:

$$W_k^\alpha = \int_0^1 t^k \sin(2\pi\alpha t)\, dt$$

da cui l'asserto. ■

Le costanti λ_k^i non dipendono da α e possono quindi calcolarsi una volta per tutte e tabulate (Tabella 2.14).

Le quantità V_k^α e W_k^α dipendono, invece, solo dal parametro α e sono facilmente calcolabili mediante le formule ricorrenti:

$$\begin{array}{ll} V_k^\alpha = \frac{1}{2\pi\alpha}(\sin(2\pi\alpha) - kW_{k-1}^\alpha) & V_0^\alpha = \frac{\sin(2\pi\alpha)}{2\pi\alpha} \\ W_k^\alpha = \frac{1}{2\pi\alpha}(kV_{k-1}^\alpha - \cos(2\pi\alpha)) & W_0^\alpha = \frac{1-cos(2\pi\alpha)}{2\pi\alpha} \end{array} \tag{2.101}$$

	i	$k=1$	$k=2$	$k=3$	$k=4$
$n=4$	1	1.000	−5.500	9.000	−4.500
	2	0.000	9.000	−22.50	13.50
	3	0.000	−4.500	18.00	−13.50
	4	0.000	1.000	−4.500	4.500
$n=3$	1	1.000	−3.000	2.000	
	2	0.000	4.000	−4.000	
	3	0.000	−1.000	2.000	
$n=2$	1	1.000	−1.000		
	2	0.000	1.000		

Tabella 2.15: Tabella delle λ_k^i per vari valori di n

Il loro calcolo, per mezzo delle (2.101), è semplice e rapido, e, inoltre, come è facile verificare, al crescere di k l'errore di arrotondamento si mantiene limitato, se risulta $\frac{k}{2\pi\alpha} \leq 1$[20].

Le formule $Q_c[f]$ e $Q_s[f]$ definite dalle (2.93) e (2.95) con i pesi L_i e M_i dati dalla (2.99) e le λ_k^i in Tabella 2.14, sono note come **formule di Filon** e sono state le prime storicamente determinate in tale modo nel 1928.

Ovviamente è possibile generalizzare tale procedimento ad un numero di nodi n qualunque ottenendo le cosiddette **formule Filon-like**. Come si è detto, per un fissato n, le quantità caratteristiche del metodo sono solo le costanti λ_k^i, che sono quindi riportate in Tabella 2.15 per vari valori di n. In particolare le formule con $n=2$ sono note come **formule di Clendenin**.

♣ **Esempio 2.21.** Si consideri il calcolo dell'integrale (2.87). Indichiamo con

$$E_c[f] = I_c[f] - \widetilde{C[f]} \ ,$$

l'errore assoluto che si commette sostituendo all'integrale $I_c[f]$, il risultato computazionale $\widetilde{C[f]}$, ottenuto mediante la routine `DQAWO.f` di `QUADPACK` [31], utilizzando la doppia precisione dell'aritmetica standard IEEE. Tale routine è basata sulla *procedura di Clenshaw-Curtis* la quale riprende l'idea comune alle formule Filon-Like.

[20] Se $\frac{k}{2\pi\alpha} > 1$ conviene servirsi degli sviluppi in serie ([27]):

$$V_k^\alpha = \sum_{n=0}^{\infty} \frac{(-1)^n (2\pi\alpha)^{2n+1}}{(2n+1)!(k+2n+2)}$$

$$W_k^\alpha = \sum_{n=0}^{\infty} \frac{(-1)^n (2\pi\alpha)^{2n}}{(2n)!(k+2n+1)}$$

In Figura 2.6 ed in Tabella 2.16, è riportato il comportamento dell'errore $E_c[f]$ al variare di ω. Si nota che per ogni valore di ω considerato, $(1 \leq \omega \leq 100)$, l'errore $E_c(f)$ è dell'ordine di 10^{-12} e, pertanto, la stima dell'integrale (2.86) puó ritenersi attendibile (fino a dodici cifre significative).
Contrariamente a quanto accade utilizzando le formule di Gauss-Kronrod, la stima dell'integrale ottenuta si puó ritenere accurata, anche per valori grandi di ω.

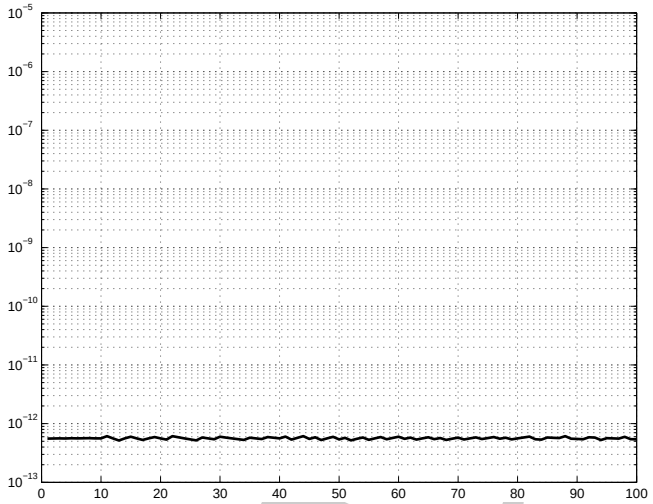

Figura 2.6: Errore $E_c[f] = I_c[f] - \widetilde{C[f]}$ per $\omega = 1, 2, \ldots, 100$.

♣

ω	$I_c[f]$		$\widetilde{C[f]}$		$E_c[f] = I_c[f] - \widetilde{C[f]}$
5	5.026548	E-01	5.026548	E-01	5.642153 E-13
10	1.256637	E-01	1.256637	E-01	5.600242 E-13
15	5.585053	E-02	5.585053	E-02	5.977926 E-13
20	3.141592	E-02	3.141592	E-02	5.604267 E-13
25	2.010619	E-02	2.010619	E-02	5.381944 E-13
30	1.396263	E-02	1.396263	E-02	5.981916 E-13
35	1.025826	E-02	1.025826	E-02	5.769256 E-13
40	7.853981	E-03	7.853981	E-03	5.609540 E-13
45	6.205615	E-03	6.205615	E-03	5.485204 E-13
50	5.026548	E-03	5.026548	E-03	5.385665 E-13
55	4.154172	E-03	4.154172	E-03	5.304177 E-13
60	3.490658	E-03	3.490658	E-03	5.984236 E-13
65	2.974288	E-03	2.974288	E-03	5.869198 E-13
70	2.564565	E-03	2.564565	E-03	5.770592 E-13
75	2.234021	E-03	2.234021	E-03	5.685105 E-13
80	1.963495	E-03	1.963495	E-03	5.610303 E-13
85	1.739290	E-03	1.739290	E-03	5.808309 E-13
90	1.551403	E-03	1.551403	E-03	5.485637 E-13
95	1.392395	E-03	1.392395	E-03	5.669356 E-13
100	1.256637	E-03	1.256637	E-03	5.385897 E-13

Tabella 2.16: Errore $E_c[f] = I_c[f] - \widetilde{C[f]}$ per $\omega = 5, 10, \ldots, 100$.

2.10 Software matematico disponibile per la quadratura

Nella costruzione di software per la quadratura sono molti gli obiettivi da realizzare. I principali sono rilevare e ridurre l'errore di round-off, fornire un risultato accurato, la facilità di uso e l'efficienza computazionale.

La disponibilità di routine di buona qualità per la quadratura è comunque molto ampia. Un gran numero di queste è raccolto nelle principali librerie di software matematico commerciale come le librerie del NAg e dell' IMSL, nonché nel package specifico per la quadratura denominato `QUADPACK`. Quasi tutte le routine di queste librerie sono basate su schemi adattativi.

La libreria NAg [30] del Numerical Algorithm Group di Oxford è sicuramente la fonte più ampia di software per il calcolo degli integrali. Sono presenti infatti circa 50 routine nel capitolo D01 dedicato alla quadratura per vari tipi di problemi, più varie routine ausiliarie che calcolano i pesi ed i nodi di particolari formule di Gauss. In particolare, oltre a numerose routine di tipo generale, sono disponibili routine per funzioni integrande particolari quali, ad esempio, quelle oscillanti o dotate di singolarità, etc. Esistono anche routine per l' integrazione di funzioni note su un insieme discreto di punti e per integrali definiti su intervalli non limitati. Sono inoltre disponibili delle routine per la quadratura multidimensionale fino ad un numero di dimensioni $d = 20$ su domini di tipo ipercubo, ipersfera e simplesso. Per buona parte delle routine esistono versioni in FORTRAN e C, in singola e doppia precisione.

Nella libreria IMSL [37] della Visual Numeric, sono disponibili routine sviluppate in linguaggio FORTRAN e C per la quadratura, più alcune routine ausiliarie per il calcolo dei pesi e dei nodi delle formule di Gauss. In particolare sono presenti anche routine per problemi in cui la funzione integranda è dotata di singolarità e per funzioni oscillanti. Altre routine infine sono dedicate al calcolo di integrali multidimensionali su iperrettangoli.

Oltre alle librerie commerciali, esistono moltissime routine di dominio pubblico reperibili per via elettronica.

`QUADPACK` [31], ad esempio, è un package costituito da numerose routine specificamente progettate per il caso monodimensionale. Esso è stato sviluppato in linguaggio FORTRAN negli anni 80 da alcuni ricercatori dell'Università di Leuven, dell'Università di Vienna e del National Bureau of Standard americano. La maggior parte delle routine sono basate su algoritmi adattativi che usano le formule di Gauss-Kronrod nonché particolari tecniche per l'accelerazione della convergenza. In particolare sono presenti routine per integrali con diverse funzioni peso, per funzioni con singolarità e per intervalli non limitati. Alcune routine di tale package sono presenti, con diverso nome, anche nella libreria NAg.

SLATEC [12] è una collezione di routine FORTRAN portabili e di dominio pubblico sviluppata da ricercatori di diversi centri di ricerca americani tra cui il Lawrence Livermore National Laboratory, l'Oak Ridge National Laboratory ed il National Institute of Standard. La versione 4.1 del 1993 contiene oltre 1400 routine di cui circa 50 dedicate alla quadratura.

Alcune riviste, infine, sono specializzate nella pubblicazione di software portabile. La più autorevole in proposito è *Transactions on Mathematical Software (TOMS)* dell'Association of Computing Machinery (ACM) [36]. Nel corso degli anni sono state pubblicate oltre 800 routine scritte in FORTRAN, di cui circa 20 sono dedicate al calcolo degli integrali di vari tipi di problemi.

Analogamente il *Journal of Computational and Applied Mathematics* [17] ha pubblicato numerose routine di software matematico di cui tre dedicate alla quadratura (due di tipo generale in una dimensione per integrali limitati e non limitati ed una per il calcolo di un integrale doppio definito su un triangolo).

2.10.1 Esempio d'uso

Di seguito si mostra un esempio di utilizzo di una routine per la quadratura. La routine in esame è D01AJF presente nella versione FORTRAN della libreria NAg. Tale routine calcola l'integrale di una funzione f nell'intervallo $[a, b]$.

La routine D01AJF (peraltro presente anche in QUADPACK con il nome di QAGS) è una routine adattativa globale basata sulla coppia di formule di Gauss-Kronrod

$$(G^{(7)}, K^{(15)})$$

I parametri di input sono:

F funzione reale;

A, B estremi dell'intervallo di integrazione;

EPSABS, EPSREL tolleranze assoluta e relativa.

I parametri di output sono:

RESULT contiene l'approssimazione di $I[f]$;

ABSERR contiene una stima dell'errore $E[f]$;

IFAIL indicatore di errore.

La routine D01AJF utilizza inoltre le aree di lavoro

W array reale di dimensione LW;

LW intero;

IW array intero di dimensione LIW;

LIW intero.

Volendo utilizzare la routine `D01AJF` per calcolare il seguente integrale

$$I[f] = \int_0^{2\pi} \frac{x \sin(30x)}{\sqrt{1 - x^2/4\pi^2}} dx = -2.54326831...$$

con una tolleranza relativa $\varepsilon = 10^{-4}$, il programma chiamante può essere il seguente[21]:

```
      SUBROUTINE D01AJF(F, A, B, EPSABS, EPSREL, RESULT,
      ABSERR, W, LW, IW, LIW, IFAIL)
      INTEGER LW, IW(LIW), LIW, IFAIL
      REAL F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
      EXTERNAL F
C
      INTEGER KOUNT
      REAL PI
      PARAMETER (LW=800, LIW=200)
      EXTERNAL FST, X01AAF
      COMMON /TELNUM/ PI, KOUNT
C
C inizializzazione variabili
C
      PI=X01AAF(PI)
      EPSABS=0.
      EPSREL=1.0E-4
C
      A=0.
      B=2.*PI
      KOUNT=0
      IFAIL=-1
C
C chiamata di D01AJF
C
      CALL D01AJF(FST,A,B,EPSABS,EPSREL,RESULT,ABSERR,W,LW,IW,LIW,IFAIL)
C
```

Esempio di programma chiamante FORTRAN per la subroutine D01AJF - continua

[21] La routine `X01AAF` è disponibile nella libreria NAg per il calcolo di π. Si noti la variabile `KOUNT` utilizzata come contatore delle chiamate alla function `FST`.

```
      IF (IFAIL.EQ.0) THEN
           WRITE(*,10) RESULT
           WRITE(*,11) ABSERR
           WRITE(*,12) KOUNT
      ELSE
           WRITE(*,20) ,IFAIL
      ENDIF
C
      STOP
C
C
10    FORMAT(' RISULTATO =',F9.5)
11    FORMAT(' ERRORE STIMATO =',E9.2)
12    FORMAT(' NO. VALUTAZ. FUNZIONE =',I4)
20    FORMAT(' ERRORE. IFAIL = ',I2)
C
      END
C
C
      REAL FUNCTION FST(X)
C
      INTEGER KOUNT
      REAL PI
      COMMON /TELNUM/ PI, KOUNT
C
      KOUNT=KOUNT+1
      FST=X*SIN(30.*X)/SQRT(1.-X**2/(4.*PI**2))
C
      RETURN
C
      END
```

Esempio di programma chiamante e function FORTRAN per la subroutine D01AJF - fine

Il risultato che si ottiene chiamando D01AJF con i dati forniti è:

```
RISULTATO             =-2.54326
ERRORE STIMATO        =0.13E-4
NO. VALUTAZ. FUNZIONE  =777
```

2.11 MATLAB e la quadratura

Oltre alle funzioni per la quadratura già citate nel **§4.3.4** della **Parte prima**, MATLAB (Release 7.7) mette a disposizione, per l'integrazione numerica: `quadgk`, `dblquad` e `triplequad` reperibili nella directory `funfun`.
La function `quadgk` implementa un algoritmo automatico di tipo adattativo basato sulla strategia globale. Come nucleo computazionale `quadgk` utilizza la coppia di formule di Gauss-Kronrod

$$(G^{(7)}, K^{(15)})$$

e consente il calcolo di un integrale definito o *improprio* di una funzione scalare f su di un intervallo $I \subseteq \Re$. Come confermato dall'esempio che segue, se la funzione integranda è oscillante `quadgk` fornisce un'accuratezza maggiore rispetto a `quad`, che implementa un algoritmo automatico di tipo adattativo con strategia locale, basato sulla formula di Simpson; inoltre `quadgk` consente l'integrazione di funzioni discontinue, quando le discontinuità cadono nell'intervallo di integrazione.

♣ **Esempio 2.22.** Il valore dell'integrale $I[f] = \int_0^\pi e^x sin(x)$, calcolato in aritmetica a precisione infinita, è:

$$I[f] = \int_0^\pi e^x sin(x) = \frac{1}{2}(e^\pi + 1) = 12.07034631638964...$$

Si calcoli un'approssimazione di $I[f]$ mediante una coppia di formule di quadratura di Gauss-Kronrod:

```
>> f=inline('exp(x).*sin(x)')
>> Q = quadgk(f,0,pi)
 Q=12.0703463163896
```

Lo stesso integrale può essere calcolato con la function `quad`:

```
>> Q=quad(f,0,pi)

Q =

  12.07034630813520
```

I risultati forniti dalle due funzioni confermano che `quadgk` fornisce un'approssimazione più accurata dell'integrale $\int_0^\pi e^x sin(x)$ rispetto a `quad`. ♣

Dal *prompt* dei comandi è possibile richiamare la funzione `quadgk` aggiungendo, opzionalmente, parametri legati ad efficienza ed accuratezza:

```
>>[q,errbnd] = quadgk(f,a,b,param1,val1,param2,val2,...)
```

I parametri `param1,param2,...` sono scelti tra i seguenti:

- `'AbsTol'`, attraverso il quale l'utente può assegnare una tolleranza sull'errore assoluto. In tal caso la funzione fornisce l'approssimazione dell'integrale `q` a meno di un errore `errbnd` tale che `errbnd` $\leq$ `AbsTol`. Il valore di default di `AbsTol` è 10^{-11} (per la doppia precisione) e 10^{-6} (per la singola).
- `'ReTol'`, attraverso il quale l'utente può assegnare una tolleranza sull'errore relativo. In tal caso la funzione fornisce l'approssimazione dell'integrale `q` a meno di un errore `errbnd` tale che `errbnd` $\leq$ `ReTol` $*$ `q`. Il valore di default di `ReTol` è 10^{-7} (per la doppia precisione) e 10^{-5} (per la singola).
- `'Waypoints'`, vettore in cui memorizzare eventuali singolarità della funzione integranda, che cadono nell'intervallo di integrazione $[a, b]$; i loro valori devono essere inseriti nel vettore `'Waypoints'` in ordine crescente o decrescente. Tali discontinuità saranno utilizzate dalla function come estremi di sottointervalli in cui suddividere l'intervallo di integrazione.
- `'MaxIntervalCount'` con cui l'utente stabilisce un massimo numero di sottointervalli; il valore di default è 650.

♣ **Esempio 2.23.** Si calcoli un'approssimazione dell'integrale $\int_0^\infty x^5 e^{-x} sin(x) dx$, con un errore relativo di ordine al più 10^{-8}:

```
>>f=inline('x.^5.*exp(-x).*sin(x)')
[q,errbnd] = quadgk(f,0,inf,'RelTol',1.0e-8)

q =

  -15.0000


errbnd =

  9.4386e-009
```

♣

La funzione `dblquad` utilizza la function `quad` per il calcolo dell'integrale doppio di una funzione $f(x, y)$, esteso ad un rettangolo:

$$\int_{x_{min}}^{x_{max}} \int_{y_{min}}^{y_{max}} f(x, y) dx dy$$

La stima fornita da `dblquad` è accurata a meno di una tolleranza `tol` che di default è 10^{-6}.

♣ **Esempio 2.24.** Si calcoli l'integrale doppio

$$I = \int_{\pi}^{2\pi} \int_{0}^{\pi} (ysin(x) + xcos(y))dxdy$$

con una tolleranza sull'errore dell'ordine di 10^{-8}:

```
>> f=inline('y*sin(x)+x*cos(y)');
>> Q = dblquad(f,pi,2*pi,0,pi,1.0e-8)
Q =

   -9.86960440090704
```

♣

Anche `triplequad` utilizza la function `quad` per il calcolo dell'integrale triplo di una funzione $f(x, y, z)$ esteso ad un parallelepipedo:

$$\int_{x_{min}}^{x_{max}} \int_{y_{min}}^{y_{max}} \int_{z_{min}}^{z_{max}} f(x, y, z)dxdydz$$

La stima fornita da `triplequad` risulta accurata a meno di una tolleranza `tol` che di default è 10^{-6}.

♣ **Esempio 2.25.** Si calcoli l'integrale triplo

$$I = \int_{0}^{\pi} \int_{0}^{1} \int_{-1}^{1} (ysinx + zcosx)dxdydz$$

a meno di una tolleranza sull'errore dell'ordine di 10^{-8}:

```
>> f=inline('y*sin(x)+z*cos(x)');
>> Q = triplequad(f,0,pi,0,1,-1,1,1.0e-8);
Q =

    1.99999999995544
```

♣

Si illustrano, infine, due esempi di utilizzo di `MATLAB` per l'implementazione del Metodo Monte Carlo per la quadratura multidimensionale. Il calcolo dell'integrale definito $\int_{\Omega} f dV$ si realizza mediante la formula di quadratura:

$$Q[f] = \frac{\int_{\Omega} dV}{N} \sum_{i=1}^{N} f(\underline{x}_i) \qquad \underline{x}_i \in \Re^d, \quad i = 1, ..., N$$

oppure, se non sono note formule di quadratura per integrali estesi a Ω, con la formula di quadratura:

$$Q[f]=\frac{\int_W dV}{N}\sum_{i=1}^{N} f(\underline{x}_i)\chi_\Omega(\underline{x}_i)$$

dove W è una regione di $\Re^d$ contenente Ω e χ_Ω è la funzione caratteristica associata a Ω:

$$\chi_\Omega(\underline{x}) = \begin{cases} 1 & se\ \underline{x}\in\Omega \\ 0 & se\ \underline{x}\notin\Omega \end{cases}$$

Negli esempi che seguono l'integrale esteso ad un sottospazio $\Omega\subseteq\Re^d$, $d = 2$, è ricondotto ad un dominio di forma nota.

♣ **Esempio 2.26.** Si calcoli l'integrale doppio di una funzione $f(x, y)$ esteso ad una regione $\Omega\subset\Re^2$ contenuta nel rettangolo $W = [a, b]\times[c, d]$:

```
>> x=a+(b-a)*rand(N,1);    % vettore casuale delle ascisse
>> y=c+(d-c)*rand(N,1);    % vettore casuale delle ordinate
>> i=find(Indicator(x,y)); % coordinate dei punti appartenenti al dominio di integrazione
>> x=x(i),y=y(i);
>> area=(b-a)*(d-c);
>> Q=(area/n)*sum(f(x,y))
```

avendo indicato con `Indicator(x,y)` la funzione caratteristica di Ω e con `Q` l'approssimazione dell'integrale doppio di $f(x, y)$ su Ω. ♣

♣ **Esempio 2.27.** Sia $\Omega\subset\Re^2$ il dominio definito da:

$$cos(2\sqrt{x^2+y^2})x\leq y, \qquad \text{tale che} \qquad x^2+y^2\leq 4.$$

Il baricentro C di Ω si calcola attraverso gli integrali: $M = \int_\Omega dV$, $M_x = \int_\Omega x dV$ e $M_y = \int_\Omega y dV$, essendo C la coppia $(\frac{M_x}{M}, \frac{M_y}{M})$. Si osserva che Ω è contenuto nel quadrato $W = [-2, 2] \times [-2, 2]$, per cui il calcolo di C si può ricondurre al calcolo degli stessi integrali, M, M_x e M_y, ma estesi a W.

```
>> N=100;
>>x=4*rand(100,1)-2;
>> y=4*rand(100,1)-2;
>>i=find(cos(2*sqrt(x.^2+y.^2)).*x<=y & x.^2+y.^2<=4);
>>area=4*4;
>> x=x(i); y=y(i);
>>M=(area/N)*length(x);
>>M_x=(area/N)*sum(x);
>>M_y=(area/N)*sum(y);
```

♣

2.12 Esercizi

2.12.1 Esercizi numerici

Esercizio 1 Utilizzare il metodo dei coefficienti indeterminati per ottenere i valori dei pesi delle seguenti Formule di Newton-Cotes:

(a) $\int_2^4 f(x)\,dx \simeq A_0 f(2) + A_1 f(3) + A_2 f(4)$, (f. di Cavalieri-Simpson in [2,4]);

(b) $\int_0^4 f(x)\,dx \simeq A_0 f(0) + A_1 f(2) + A_2 f(4)$, (f. di Cavalieri-Simpson in [0,4]);

(c) $\int_0^3 f(x)\,dx \simeq A_0 f(0) + A_1 f(1) + A_2 f(2) + A_3 f(3)$, (formula dei '3/8' in [0,3]);

(d) $\int_1^4 f(x)\,dx \simeq A_0 f(1) + A_1 f(2) + A_2 f(3) + A_3 f(4)$, (formula dei '3/8' in [1,4]).

Esercizio 2 Mostrare che l'errore di discretizzazione è nullo nei seguenti casi:

(a) Formula di Cavalieri-Simpson applicata in [2,3] a $f(x) = x^3 + x^2 + 2$;

(b) Formula di Cavalieri-Simpson applicata in [0,1] a $f(x) = x - \pi$;

(c) Formula dei '3/8' applicata in [2,3] a $f(x) = x^3 + x^2 + 2$;

(d) Formula dei '3/8' applicata in [0,1] a $f(x) = x - \pi$.

Esercizio 3 Fornire una stima dell'errore di discretizzazione, utilizzando l'espressione ottenuta come conseguenza del teorema di Peano, nei seguenti casi:

(a) Formula trapezoidale applicata in [0,1] a $f(x) = x^2$;

(b) Formula trapezoidale applicata in [2,3] a $f(x) = e^x sin(x)$;

(c) Formula di Cavalieri-Simpson applicata in [2,3] a $f(x) = x^4 + x^2 + 2$;

(d) Formula di Cavalieri-Simpson applicata in [0,0.5] a $f(x) = 3^x$;

(e) Formula dei '3/8' applicata in [2,2.5] a $f(x) = x^4 + 1$;

(f) Formula dei '3/8' applicata in [0,1] ad $f(x) = e^x(x + 1)$.

Esercizio 4 Verificare che la formula di Newton Cotes con $n = 5$:

$$\int_0^1 f(x)dx \simeq \frac{1}{90}\left(7f(0) + 32f\left(\frac{1}{4}\right) + 12f\left(\frac{1}{2}\right) + 32f\left(\frac{3}{4}\right) + 7f(1)\right)$$

ha grado di precisione algebrico 4.

Esercizio 5 A partire dai valori di nodi e pesi della formula dell'esercizio 4, dedurre l'espressione della formula di Newton-Cotes con $n = 5$, nell'intervallo $[-2, 2]$

$$\int_{-2}^2 f(x)dx \simeq A_0 f(x_0) + A_1 f(x_1) + A_2 f(x_2) + A_3 f(x_3) + A_4 f(x_4)\ .$$

Esercizio 6 Cosa accade se si applica una formula di Newton-Cotes di ordine n elevato per integrare la funzione di Runge

$$\int_{-1}^{1} \frac{1}{1+25x^2} dx \ ?$$

Giustificare la risposta. Il risultato cambia se si considera la funzione

$$\frac{1}{1+25(x-3)^2} \ ?$$

Esercizio 7 Quale delle seguenti affermazioni è vera? Motivare le risposte.

(a) La distanza tra due nodi consecutivi in una formula di Newton-Cotes è costante e non dipende dal numero di nodi;

(b) La distanza tra due nodi consecutivi in una formula di Newton-Cotes è costante e dipendente solo dal numero di nodi n e dall'ampiezza dell'intervallo di integrazione $b-a$;

(c) La posizione dei nodi nell'intervallo $[a,b]$, in una formula di Newton-Cotes, dipende solo dal numero di nodi n e dall'ampiezza dell'intervallo di integrazione $b-a$.

(d) I pesi delle formule di Newton-Cotes sono tutti positivi indipendentemente da n

(e) Una formula di Newton-Cotes con n nodi ha sempre grado di precisione algebrico $n-1$.

(f) La distanza tra due nodi consecutivi in una formula di Gauss è costante e dipende solo dal numero di nodi n e dalla funzione peso.

(g) La distanza tra due nodi consecutivi in una formula di Gauss non è costante e dipende solo dal numero di nodi n.

(h) Le formule di Gauss hanno sempre grado di precisione algebrico dispari.

(i) Le formule di Gauss sono formule di tipo aperto.

(l) Formule di Gauss con un diverso numero di nodi non hanno in alcun caso nodi in comune.

(m) I pesi delle formule di Gauss sono tutti positivi indipendentemente da n e dalla funzione peso.

(n) A parità di numero di nodi, nell'intervallo $[-1,1]$, le formule di Newton-Cotes sono meno accurate delle formule di Gauss-Legendre, ma piú accurate delle formule di Gauss-Chebyshev.

Esercizio 8 Qual è il grado di precisione algebrico delle seguenti formule di quadratura?

(a) Formula di Newton-Cotes con un numero di nodi n dispari;

(b) Formula di Newton-Cotes con un numero di nodi n pari;

(c) Formula di Gauss-Legendre con un numero di nodi n dispari;

(d) Formula di Gauss-Legendre con un numero di nodi n pari;

(e) Formula di Gauss-Laguerre con n nodi;

(f) Formula di Gauss-Jacobi con n nodi;

(h) Formula di Gauss-Hermite con n nodi.

Esercizio 9 Qual è il più piccolo valore del numero di nodi n affinchè la formula di Gauss abbia grado di precisione 3? Verificare la risposta con l'integrale $\int_0^2 x^3\,dx$.

Esercizio 10 Mostrare che se si impone che la formula di quadratura con un nodo

$$\int_a^b f(x)\,dx \simeq A_0 f(x_0)$$

sia esatta per le funzioni $f(x) = 1$ e $f(x) = x$ (formula di Gauss-Legendre con 1 nodo) si ottiene la formula del punto medio (ciò rappresenta una spiegazione al fatto che la formula del punto medio ha grado di precisione 1).

Esercizio 11 Ottenere, imponendo l'esattezza per lo spazio dei polinomi di grado al piú 1, il valore del nodo x_0 e del peso A_0 delle formule di Gauss-Chebyshev

(I) $\int_{-1}^1 (1-x^2)^{-1/2} f(x)\,dx \simeq A_0 f(x_0)$;

(II) $\int_{-1}^1 (1-x^2)^{1/2} f(x)\,dx \simeq A_0 f(x_0)$.

Esercizio 12 Ottenere, imponendo l'esattezza per lo spazio dei polinomi di grado al piú 1, il valore del nodo x_0 e del peso A_0 per le formule di Gauss-Jacobi con un nodo ($n = 1$), caratterizzate dalle seguenti scelte dei parametri α e β nella funzione peso

$$w(x) = (1-x)^\alpha (1+x)^\beta \ .$$

(a) $\int_{-1}^1 (1-x)^{-1/2} f(x)\,dx \simeq A_0 f(x_0), \quad \alpha = -1/2,\ \beta = 0;$

(b) $\int_{-1}^1 (1-x)^{1/2} f(x)\,dx \simeq A_0 f(x_0), \quad \alpha = 1/2, \quad \beta = 0;$

(c) $\int_{-1}^1 (1+x)^{-1/2} f(x)\,dx \simeq A_0 f(x_0), \quad \alpha = 0, \quad \beta = -1/2;$

(d) $\int_{-1}^1 (1+x)^{1/2} f(x)\,dx \simeq A_0 f(x_0), \quad \alpha = 0, \quad \beta = 1/2;$

Esercizio 13 Che formula si ottiene ponendo $\alpha = \beta = -1/2$ nella funzione peso che caratterizza le formule di Gauss-Jacobi? e ponendo $\alpha = \beta = 1/2$?

Esercizio 14 Ottenere, imponendo l'esattezza per lo spazio dei polinomi di grado al piú 1, il valore del nodo x_0 e del peso A_0 della formula di Gauss-Laguerre con un nodo ($n = 1$):

$$\int_0^\infty e^{-x} f(x)\,dx \simeq A_0 f(x_0) \ .$$

Esercizio 15 Ottenere, imponendo l'esattezza per lo spazio dei polinomi di grado al piú 1, il valore del nodo x_0 e del peso A_0 della formula di Gauss-Hermite con un nodo ($n = 1$):

$$\int_{-\infty}^{\infty} e^{-x^2/2} f(x)\, dx \simeq A_0 f(x_0) \ .$$

Esercizio 16 Che relazione esiste tra le famiglie di formule di quadratura di Gauss e le famiglie di polinomi ortogonali?

Esercizio 17 Verificare la bontà risultati ottenuti, per i valori dei nodi x_0, negli esercizi 10, 11, 12, 14 e 15 rispettivamente con le radici dei polinomi di primo grado appartenenti alle famiglie di polinomi ortogonali di Chebyshev, di Laguerre, di Jacobi e di Hermite.

Esercizio 18 Determinare i nodi x_0 e x_1 della formula di Gauss-Chebyshev

$$\int_{-1}^{1} (1 - x^2)^{-1/2} f(x)\, dx \simeq A_0 f(x_0) + A_1 f(x_1) \ , \qquad (I)$$

utilizzando l'espressione del corrispondente polinomio ortogonale di secondo grado.

Esercizio 19 A partire dai valori determinati nell'esercizio 9, determinare il valore dei pesi A_0 e A_1 della formula di Gauss-Chebyshev impondendo che l'errore di discretizzazione sia nullo per $f(x) = 1$ e $f(x) = x$.

Esercizio 20 Ripetere gli esercizi 18 e 19 per le formule con 2 nodi di:
Gauss-Jacobi

$$\int_{-1}^{1} (1 - x)^{-1/2} f(x)\, dx \simeq A_0 f(x_0) + A_1 f(x_1), \quad (\alpha = -1/2,\ \beta = 0) \ ;$$

di Gauss-Laguerre

$$\int_{0}^{\infty} e^{-x} f(x)\, dx \simeq A_0 f(x_0) + A_1 f(x_1) \ ;$$

e di Gauss-Hermite

$$\int_{-\infty}^{\infty} e^{-x^2/2} f(x)\, dx \simeq A_0 f(x_0) + A_1 f(x_1) \ .$$

Esercizio 21 Dire quale delle seguenti formule di quadratura ha, a parità del numero di nodi, il grado precisione algebrico piú elevato:

(a) Formule di Newton-Cotes;

(b) Formule di Gauss-Legendre;

(c) Formule di Gauss-Kronrod.

Esercizio 22 Determinare i pesi della formula di quadratura con due nodi equispaziati, del tipo:

$$\int_0^1 \sqrt{x} f(x)\, dx \simeq A_0 f(0) + A_1 f(1)$$

esatta per polinomi di primo grado.

Esercizio 23 Determinare i nodi della formula di quadratura a coefficienti costanti $A_0 = A_1 = 1/2$:

$$\int_0^1 \sqrt{x} f(x)\, dx \simeq \frac{1}{2} f(x_0) + \frac{1}{2} f(x_1)$$

in modo da avere il massimo grado di precisione algebrico. Calcolare il valore dell'errore di discretizzazione

$$E[x^2] = \int_0^1 \sqrt{x} x^2\, dx - \frac{1}{2} x_0^2 + \frac{1}{2} x_1^2\,,$$

e confrontarlo con l'errore di discretizzazione che si commette applicando la formula trapezoidale alla funzione $f(x) = x^{5/2}$.

Esercizio 24 Determinare i pesi della formula di quadratura con tre nodi equispaziati,

$$\int_0^1 f(x)\, dx \simeq A_0 f\left(\frac{1}{3}\right) + A_1 \left(\frac{1}{2}\right) + A_2 f\left(\frac{2}{3}\right)$$

esatta per polinomi di secondo grado col metodo dei coefficienti indeterminati (Formule di Newton Cotes aperte). Qual è il grado di precisione algebrico della formula ottenuta?
Confrontare i risultati ottenuti da tale formula con la formula di Simpson sui seguenti integrali:

$$\int_0^1 \frac{dx}{1+x} = \log 2 \qquad \int_0^1 \frac{4}{1+x^2} = \pi \qquad \int_0^1 e^{-x}\, dx = 1 - \frac{1}{e}$$

Esercizio 25 Determinare i pesi della formula di quadratura con due nodi equispaziati, del tipo:

$$\int_0^{\pi/2} cos(x) f(x)\, dx \simeq A_0 f(0) + A_1 f(\pi/2)$$

esatta per polinomi di primo grado. Qual è il grado di precisione algebrico della formula ottenuta? Determinare il valore dell'errore di discretizzazione che si commette applicando la formula ottenuta per il calcolo dell'integrale

$$\int_0^{\pi/2} cos(x) x^3\, dx$$

Esercizio 26 Determinare i nodi della formula di quadratura a coefficienti costanti $A_0 = A_1 = 1/2$:

$$\int_0^1 cos(x) f(x)\, dx \simeq \frac{1}{2} f(x_0) + \frac{1}{2} f(x_1)$$

in modo da avere il massimo grado di precisione algebrico. Calcolare il valore dell'errore di discretizzazione

$$E[x^2] = \int_0^1 cos(x) x^2\, dx - \frac{1}{2} x_0^2 + \frac{1}{2} x_1^2\,,$$

e confrontarlo con l'errore di discretizzazione che si commette applicando la formula trapezoidale alla funzione $f(x) = x^2 cos(x)$. Quale delle due formule ottiene una approssimazione piú accurata dell'integrale?

Esercizio 27 Utilizzando l'espressione delle formule base, determinare pesi e nodi delle seguenti formule composite su m sottointervalli:

(a) formula trapezoidale composita $T_m[f]$ in [0,1] con $m = 3$;

(b) formula trapezoidale composita $T_m[f]$ in [0,5] con $m = 5$;

(c) formula di Cavalieri-Simpson composita $S_m[f]$ in $[0, 4]$ con $m = 4$;

(d) formula di Cavalieri-Simpson composita $S_m[f]$ in $[-3, 3]$ con $m = 3$;

(e) formula dei '3/8' composita $Q_m[f]$ in $[0, 4]$ con $m = 2$;

(e) formula dei '3/8' composita $Q_m[f]$ in $[-6, 0]$ con $m = 3$.

Esercizio 28 Mostrare che l'errore di discretizzazione è nullo nei seguenti casi:

(a) Formula del punto medio composita su 2 sottointervalli applicata in [0,2] a $f(x) = 3x$;

(a) Formula trapezoidale composita su 3 sottointervalli applicata in [-3,3] a $f(x) = \pi x + 2$;

(b) Formula di trapezoidale composita su 4 sottointervalli applicata in [0,4] a $f(x) = -2x + 1$;

(a) Formula di Cavalieri-Simpson composita su 3 sottointervalli applicata in [0,3] a $f(x) = x^3$;

(b) Formula di Cavalieri-Simpson composita su 2 sottointervalli applicata in [-1,1] a $f(x) = 3x + \pi$;

(c) Formula dei '3/8' composita su 2 sottointervalli applicata in [0,2] a $f(x) = x^3 + x^2 + 2$.

Esercizio 29 Fornire una stima dell'errore di discretizzazione, utilizzando l'espressione ottenuta come conseguenza del teorema di Peano, nei seguenti casi:

(a) Formula trapezoidale composita su 5 sottointervalli applicata in [0,1] a $f(x) = x^2 - 1$;

(b) Formula trapezoidale composita su 10^2 sottointervalli applicata in [2,3] a $f(x) = e^x cos(x)$;

(c) Formula di Cavalieri-Simpson composita su 10 sottointervalli applicata in [2,3] a $f(x) = 2x^4 + 3$;

(d) Formula di Cavalieri-Simpson composita su 10^3 sottointervalli applicata in [0,1] a $f(x) = 2^x$;

(e) Formula dei '3/8' composita su 10 sottointervalli applicata in [0,1] a $f(x) = x^4 + 1$;

(f) Formula dei '3/8' composita su 10^5 sottointervalli applicata in [0,1] a $f(x) = e^x(x + 1)$.

Esercizio 30 Cosa si intende con ordine di convergenza di una formula composita? Per ognuna delle seguenti formule specificare l'ordine di convergenza.

(a) Formula trapezoidale composita;

(b) Formula di Cavalieri-Simpson composita;

(c) Formula dei '3/8' composita.

Esercizio 31 Se applichiamo ad una funzione integrabile in $[a, b]$, una formula composita con ordine di convergenza 3, quale riduzione dell'errore di discretizzazione è lecito attendersi passando da $m = 100$ a $m = 400$ sottointervalli?

2.12.2 Problemi da risolvere con il calcolatore

Problema 1 Scrivere una procedura basata sulla formula trapezoidale composita. Tale procedura deve ricevere in input:

1. gli estremi dell'intervallo di integrazione a, b;
2. la funzione esterna f;
3. il numero di sottointervalli m (oppure il passo di discretizzazione $h = \frac{b-a}{m}$);

e fornire in output

4. una stima *integ* dell'integrale tra a e b di f.

Problema 2 Testare la procedura implementata nel Problema 1 per calcolare il seguente integrale:

$$(a) \quad \int_0^1 (2 - x)dx = 1.5$$

ponendo $m = 100, 700, 5000$. Quanto vale l'errore di discretizzazione nei casi considerati? Cosa si osserva circa l'errore effettivamente commesso? Spiegare quali tipologie di errore sono presenti sulla soluzione.

Problema 3 Testare la procedura implementata nel Problema 1 per calcolare i seguenti integrali:

$$(b) \quad \int_{0.9}^{3} x^2 dx = 8.757 \quad con\ m = 100, 200, 400$$
$$(c) \quad \int_{0}^{1} x^{3/2} dx = 0.4 \quad con\ m = 300, 600, 1200$$

Cosa si osserva circa l'errore commesso, al raddoppiare del numero di sottointervalli? Commentare analogie o differenze di comportamento dei risultati tra i due casi considerati.

Problema 4 Scrivere una procedura basata sulla formula di Cavalieri-Simpson composita su m sottointervalli. Tale procedura deve ricevere in input:

1. gli estremi dell'intervallo di integrazione a, b;
2. la funzione esterna f;
3. il numero di sottointervalli m (oppure il passo di discretizzazione $h = \frac{b-a}{m}$);

e fornire in output

4. una stima *integ* dell'integrale tra a e b di f.

Problema 5 Testare la procedura implementata nel Problema 4 per calcolare il seguente integrale:

$$(a) \quad \int_{0}^{1} (2 - x) dx = 1.5$$

ponendo $m = 100, 700, 5000$. Quanto vale l'errore di discretizzazione nei casi considerati? Cosa si osserva circa l'errore effettivamente commesso? Spiegare bene quali tipologie di errore sono presenti sulla soluzione. Confrontare inoltre i risultati con quelli ottenuti attraverso la formula trapezoidale.

Problema 6 Testare la procedura implementata nel Problema 4 per calcolare i seguenti integrali:

$$(b) \quad \int_{0}^{2} x^3 dx = 4 \quad con\ m = 100, 200, 400$$
$$(c) \quad \int_{0}^{1} x^{5/2} dx = 2/7 \quad con\ m = 300, 600, 1200$$

Cosa si osserva circa l'errore commesso, al raddoppiare del numero di sottointervalli? Commentare analogie o differenze di comportamento dei risultati tra i due casi considerati. Quali caratteristiche del problema influenzano la velocità di convergenza della formula composita di Cavalieri-Simpson composita?

Problema 7 Scrivere una procedura che calcoli le formule di quadratura

$$T_m^{(1)}[f] = \frac{4T_{2m}[f] - T_m[f]}{3} \qquad m = 1, 2, \ldots$$

ottenute al primo passo del procedimento di estrapolazione di Romberg. Tale procedura, che puó fare uso della procedura scritta per la formula trapezoidale composita, deve ricevere in input:

1. gli estremi dell'intervallo di integrazione a, b;
2. la funzione esterna f;
3. il numero di sottointervalli m (oppure il passo di discretizzazione $h = \frac{b-a}{m}$);

e fornire in output

4. una stima *integ* dell'integrale tra a e b di f.

Problema 8 Testare la procedura implementata nel Problema 7 per calcolare i seguenti integrali:

$$(a) \quad \int_0^3 e^x dx = e^3 - 1 = 19.08553692318767... \quad con\ m = 2, 4, 8, 16$$
$$(b) \quad \int_0^1 x^{5/2} dx = 2/7 = 0.285714285714... \quad con\ m = 1, 2, 4, 8, 16, 32$$

Cosa si osserva circa l'errore commesso, al raddoppiare del numero di sottointervalli? Verificare numericamente che le formule di Romberg ottenute al primo passo hanno ordine di convergenza 4. Qual è l'ordine di convergenza effettivo nel secondo caso?

Problema 9 Scrivere una procedura che calcoli le formule di quadratura

$$T_m^{(2)}[f] = \frac{16T_{2m}^{(1)}[f] - T_m^{(1)}[f]}{15} \qquad m = 1, 2, \ldots$$

ottenute al secondo passo del procedimento di estrapolazione di Romberg. Tale procedura puó fare uso della procedura scritta per il Problema 8.

Problema 10 Ripetere i test del Problema 8 utilizzando la procedura implementata nel Problema 9.

Problema 11 Posto $\alpha = \frac{\omega b}{2\pi}$, le formule di Filon

$$\int_0^b f(x)cos(\omega x)dx \simeq Q_c[f] = b[L_1 f(0) + L_2 f(b/2) + L_3 f(b)]$$

$$\int_0^b f(x)sin(\omega x)dx \simeq Q_s[f] = b[M_1 f(0) + M_2 f(b/2) + M_3 f(b)]$$

hanno pesi L_i e M_i (i=1,2,3) dati da

$$L_i = \sum_{k=0}^{2} \lambda_k^i V_k^\alpha \qquad M_i = \sum_{k=0}^{2} \lambda_k^i W_k^\alpha$$

con λ_k^i opportune costanti e

$$V_k^\alpha = \int_0^1 t^k cos(2\pi\alpha t)dt, \quad W_k^\alpha = \int_0^1 t^k sin(2\pi\alpha t)dt\ .$$

(a) Scrivere una procedura per il calcolo dei V_k^α e W_k^α con $k = 0, 1, \ldots, n$, basata sulle formule ricorrenti

$$\begin{array}{ll} V_k^\alpha = \frac{1}{2\pi\alpha}(\sin(2\pi\alpha) - kW_{k-1}^\alpha) & V_0^\alpha = \frac{\sin(2\pi\alpha)}{2\pi\alpha} \\ W_k^\alpha = \frac{1}{2\pi\alpha}(kV_{k-1}^\alpha - \cos(2\pi\alpha)) & W_0^\alpha = \frac{1-cos(2\pi\alpha)}{2\pi\alpha} \end{array} \tag{2.102}$$

che deve ricevere in input:

(i) il valore α;

(ii) il numero di integrali da calcolare n;

e fornire in output

(iii) due array V, W che forniscono le stime degli integrali voluti.

(b) posto $b = 0.3$, $\omega = 2\pi$ e di conseguenza $\alpha = b = 0.3$ testare tale procedura per calcolare i primi 3 integrali V_k^α e W_k^α con $k = 0, 1, 2$.

(c) calcolare gli stessi integrali utilizzando una routine di quadratura adattativa standard (ad esempio una tra le routine DQAG.f e DQAWO.f di QUADPACK) e confrontare i risultati ottenuti con quelli precedenti;

(d) a partire dai risultati ottenuti, e utilizzando i valori

$$\lambda_0^1 = 1,\ \lambda_1^1 = -3,\ \lambda_2^1 = 2, \quad \lambda_0^2 = 0,\ \lambda_1^2 = 4,\ \lambda_2^2 = -4, \quad \lambda_0^3 = 0,\ \lambda_1^1 = -1\ ,\lambda_2^3 = 2$$

calcolare i coefficienti L_i e M_i delle formule di Filon.

(e) utilizzare le formule ottenute per calcolare i seguenti integrali

(e1) $\int_0^{0.3} e^x cos(2\pi x)dx$

(e2) $\int_0^{0.3} (x^3 + 1)cos(2\pi x)dx$

(e3) $\int_0^{0.3} e^{-x} sin(2\pi x)dx$

(e4) $\int_0^{0.3} (x^3 + x)sin(2\pi x)dx$

Problema 12 Ripetere il Problema 11 per diversi valori di α, ω e b.

Bibliografia

[1] Chawla M. - *Convergence of Newton-Cotes quadratures for analytic functions* - BIT, vol. 11, pp. 159-167 (1971).

[2] Cools R. - *Advances in multidimensional integration* - Journal of Computational and Applied Mathematics, vol. 149, Issue 1, pp. 1-12 (2002).

[3] Cools R., Kuo Frances Y., Nuyens D. - *Constructing Embedded Lattice Rules for Multivariate Integration* - SIAM Journal on Scientific Computing archive, vol. 28, Issue 6, pp. 2162-2188 (2006).

[4] Davis P. J. - *Interpolation and approximation* - Blaisdell, New York, 1975.

[5] Davis P., Rabinowitz P. - *Methods of numerical integration* - Dover Publications, Inc. Mineola, NY, 2nd ed., 2007.

[6] De Angelis P. L., Murli A., Pirozzi M. A. - *Sulla valutazione dell'errore nel calcolo numerico degli integrali trigonometrici* - Calcolo Vol. 13 (1976).

[7] Delves L. and Mohamed J. - *Computational methods for integral equations* - Cambridge University Press, 1985.

[8] Dick J. - *Explicit constructions of quasi-Monte Carlo rules for the numerical integration of high-dimensional periodic functions* - SIAM Journal on Numerical Analysis, vol. 45, Issue 5, pp. 2141-2176 (2007).

[9] Engels P. - *Numerical quadrature and cubature* - Academic Press, Inc., 1980.

[10] Espelid T. O. - *Algorithm 868: Globally doubly adaptive quadrature - reliable Matlab codes* - ACM Transactions on Mathematical Software, vol. 33, Issue 3, Article No. 21 (2007).

[11] Evans G. - *Practical Numerical Integration* - Wiley, Chichester, U.K., 1993.

[12] Fong K. W., Jefferson T. H., Suyehiro T., Walton L. - *SLATEC Common Mathematical Library* - (1993), `http://www.netlib.org/slatec`.

[13] Gentle James E. - *Random number generation and Monte Carlo methods* - Springer, New York, 2nd edition, 2003.

[14] Genz A., Malik A. - *An imbedded family of fully symmetric numerical integration rules* - SIAM Journal on Numerical Analysis, vol. 20, No. 3, pp. 580-588 (1983).

[15] Giunta G., Murli A. - *Algorithm 649. A Package for Computing Trigonometric Fourier Coefficients Based on Lyness Algorithm* - ACM Transactions on Mathematical Software, n. 13, n^o 1, pp. 97-107 (1987).

[16] Hammersley J., Handscomb D. - *Monte Carlo methods* - Methuen, London, 1964.

[17] *Journal of Computational and Applied Mathematics (JCAM)* - `http://www.elsevier.com/locate/cam`.

[18] Krommer A., Ueberhuber C. - *Numerical Integration on advanced computer systems* - Springer Verlag, 1994.

[19] Krommer A. R., Ueberhuber Ch. W. - *Computational integration* - SIAM, 1998.

[20] Kronrod A. - *Nodes and weights of quadrature formulas* - Consultants bureau, 1965.

[21] Krylov V. - *Approximate calculation of integrals* - Mc Millan Comp., 1962.

[22] Lanczos C. - *Applied Analysis* - Prentice Hall, Inc., 1956.

[23] Malcom L., Simpson R. - *Local versus Global strategies for adaptive quadrature* - ACM Transactions on Mathematical Software, vol. 1, pp. 129-146 (1975).

[24] `http://www.mathworks.com/`.

[25] Monegato G. - *Stieltjes polynomials and related quadrature rules* - SIAM Review, vol. 24, pp. 137-158 (1982).

[26] Monegato G. - *Positivity of weights of extended Gauss-Legendre quadrature rules* - Mathematics of Computation, vol. 32, No. 141, pp. 243-245 (1978).

[27] Murli A. - *Il calcolo numerico degli integrali trigonometrici* - Calcolo, Supplemento n.1, vol. V (1968).

[28] Murli A. - *Sull'impiego di un metodo numerico per il calcolo dell'antitrasformata di Laplace* - Rendiconti dell'Accademia di Scienze Fisiche e Matematiche della Società Nazionale di Scienze, Lettere ed Arti in Napoli, Serie 4, Vol. XXXVII (1970).

[29] Murli A. - *Alcune formule per il calcolo numerico della Trasformata di Fourier* - Calcolo, vol. 4, fasc. 4 (1967).

[30] Numerical Algorithm Group - *NAg Library Manual, Mark 22* - Oxford (2009), `http://www.nag.com/`.

[31] Piessens R. et al - *QUADPACK, a subroutine package for automatic integration* - Springer Verlag (1983), `http://www.netlib.org/quadpack`.

[32] Shampine L. F. - *Vectorized adaptive quadrature in MATLAB* - Journal of Computational and Applied Mathematics, vol. 211, Issue 2, pp. 131-140 (2008).

[33] Sloan I., Joe S. - *Lattice Methods for Multiple Integration* - Clarendon Press, 1994.

[34] Stroud A. H. - *Approximate calculation of multiple integrals* - Prentice Hall, 1971.

[35] Szegö G. - *Ortogonal polynomials* - American Mathematical Society Colloquium Publications, Volume XXIII, 1939.

[36] *Transactions on Mathematical Software (TOMS)* - `http://www.netlib.org/toms`.

[37] Visual Numerics - *IMSL C Numerical Library version 7.0 (2008), IMSL Fortran Numerical Library Version 6.0 (2007)* - `http://www.vni.com/`.

Capitolo 3

Risoluzione numerica di ODE: problemi a valori iniziali

3.1 Alcuni esempi di problemi a valori iniziali

In generale, un'equazione differenziale è un'equazione che coinvolge una funzione incognita e le sue derivate. In particolare, si parla di *equazioni differenziali ordinarie* (ODE)[1] se l'incognita è funzione di una sola variabile indipendente.

♣ **Esempio 3.1. (Moto rettilineo uniforme)** Si consideri un punto che si muove di moto rettilineo uniforme con velocità v. La posizione $x(t)$ del punto ad un generico istante t, lungo la retta che costituisce la traiettoria del moto, soddisfa la seguente ODE:

$$x'(t) = v \tag{3.1}$$

♣

♣ **Esempio 3.2. (Decadimento radioattivo)** In natura esistono alcuni elementi chimici che decadono spontaneamente in loro isotopi o in altri elementi, mediante emissione di particelle α, particelle β o fotoni. Tali elementi sono detti radioattivi. Ad esempio il carbonio ^{14}C decade nel suo isotopo ^{12}C. Se si considera un numero elevato di atomi radioattivi, cioè dell'ordine del numero di Avogadro ($\sim 10^{23}$), la velocità con cui tale numero varia è direttamente proporzionale al numero stesso di atomi, ovvero il numero $N(t)$ di atomi radioattivi all'istante t soddisfa la ODE:

$$N'(t) = -kN(t) \tag{3.2}$$

dove k è una costante positiva, generalmente determinata in maniera sperimentale.[2] ♣

[1] Dall'inglese *Ordinary Differential Equations*.

[2] Si noti che il numero di atomi è trattato come una grandezza continua e non discreta, quale è in realtà. Tale approssimazione si può ritenere valida perché si considera un numero elevato di atomi.

Le equazioni (3.1) e (3.2) costituiscono un esempio di ODE del *primo ordine*. In generale, una ODE del primo ordine è un'equazione del tipo:

$$y'(t) = f(t, y(t)), \quad t \in I \tag{3.3}$$

dove I è un intervallo dell'asse reale, y, l'incognita, è una funzione reale di variabile reale, derivabile in I, ed f è una funzione reale nota, definita in $I \times \Re$. L'equazione si dice del primo ordine in quanto la derivata di ordine massimo della funzione incognita che compare in essa è la derivata prima. L'equazione (3.3) è inoltre in *forma esplicita*, cioè è esplicitata rispetto alla derivata di ordine massimo di y.

Spesso un'equazione di questo tipo rappresenta l'evoluzione temporale di qualche fenomeno e la variabile indipendente t rappresenta il tempo. Nel seguito si considera $I = [t_0, T]$, con $t_0 < T < +\infty$ oppure $I = [t_0, +\infty)$.

Una soluzione di (3.3) è denominata anche *integrale* di (3.3); da (3.3) discende infatti che

$$y(t) = \int_{t_0}^{t} f(\tau, y(\tau))d\tau + c, \quad t \in I,$$

con c costante arbitraria. Le eventuali soluzioni di (3.3) costituiscono quindi una famiglia di funzioni, dipendenti da un parametro.

♣ **Esempio 3.3. (Moto rettilineo uniforme - continuazione)** La generica soluzione dell'equazione (3.1) ha l'espressione

$$x(t) = v\,t + c\,,$$

dove c è una costante arbitraria. Tale soluzione è rappresentata, al variare di c in Figura 3.1 (sinistra). È chiaro che la posizione del punto all'istante t dipende dalla posizione del punto ad un istante precedente t_0, che si può assumere come istante iniziale, cioè dipende dalla condizione iniziale

$$x(t_0) = x_0\,.$$

Imponendo tale condizione si determina un'unica soluzione

$$x(t) = v\,(t - t_0) + x_0\,,$$

che fornisce la posizione del punto nel caso particolare considerato. ♣

♣ **Esempio 3.4. (Decadimento radioattivo - continuazione)** La generica soluzione dell'equazione (3.2) è del tipo

$$N(t) = c\,e^{-kt}\,,$$

dove c è una costante arbitraria. Tale soluzione è rappresentata, al variare di c e per $k = 1$, in Figura 3.1 (destra). Analogamente all'esempio precedente, il numero di atomi ad un certo istante t dipende dal numero iniziale di atomi, cioè dipende dalla condizione iniziale

$$N(t_0) = N_0\,,$$

dove t_0 indica l'istante iniziale. Imponendo tale condizione si determina la soluzione particolare

$$N(t) = N_0 e^{-k(t-t_0)}.$$

♣

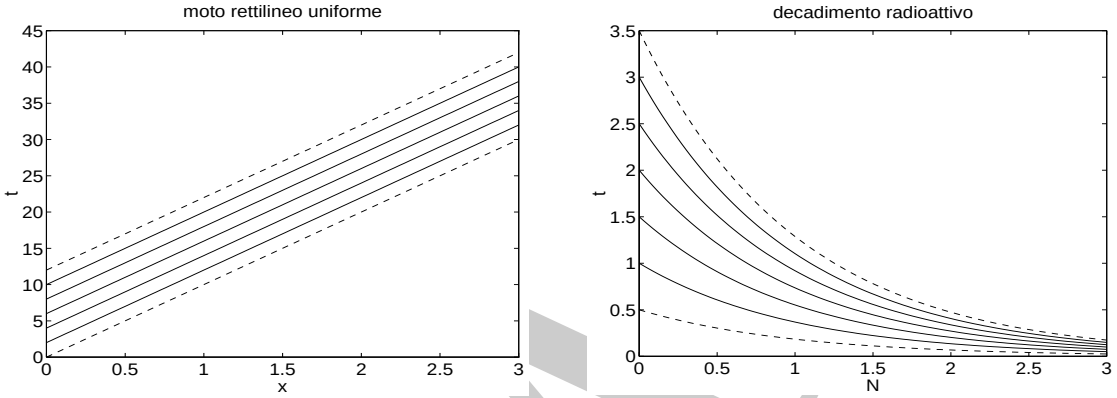

Figura 3.1: Famiglia di soluzioni dell'equazione del moto rettilineo uniforme e dell'equazione del decadimento radioattivo.

In generale, sotto ipotesi opportune su f, l'assegnazione di una condizione aggiuntiva sull'incognita y, quale ad esempio la condizione

$$y(t_0) = y_0, \tag{3.4}$$

consente di determinare una ed una sola soluzione dell'equazione. La condizione (3.4) si dice *condizione iniziale* ed il problema

$$M(P) \equiv \begin{cases} y'(t) &= f(t, y(t)), \quad t \in I \\ y(t_0) &= y_0 \end{cases} \tag{3.5}$$

si dice *problema di Cauchy* o *problema a valori iniziali (IVP)*[3]. Nel seguito si suppone che f verifichi le ipotesi che garantiscono l'esistenza e l'unicità della soluzione, ovvero le ipotesi del teorema:

Teorema 3.1.1. [Esistenza e unicità]
Sia $f(t, y)$ continua nella striscia $S = I \times \Re = \{(t, y) : t \in I,\ y \in \Re\}$, con $I = [a, b]$. Si

[3]Dall'inglese *Initial Value Problem*.

supponga che f soddisfi in S una condizione di Lipschitz rispetto ad y, uniformemente rispetto a t, cioè che esista $L > 0$ tale che:

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \tag{3.6}$$

per ogni $t \in I$ e $y_1, y_2 \in \Re$.
Allora, per ogni $t_0 \in I$ e $y_0 \in \Re$, esiste una ed una sola soluzione del problema (3.5), cioè esiste una ed una sola funzione $y(t)$, derivabile in I, tale che

$$\begin{cases} y'(t) &= f(t, y(t)) \qquad t \in I \\ y(t_0) &= y_0 \end{cases}$$

Più in generale, nella costruzione di modelli matematici di problemi reali, si possono ottenere problemi a valori iniziali relativi a *sistemi di ODE* del primo ordine.

♣ **Esempio 3.5. (Problema predatore-preda)** Si considerino due popolazioni di individui appartenenti a specie che interagiscono secondo una relazione predatore-preda, quali ad esempio squali e pesci più piccoli nell'oceano, oppure lupi e lepri. Si supponga che la popolazione di prede sia in grado di procurarsi cibo a sufficienza e che quindi il numero di prede $x(t)$, in assenza di predatori, cresca con una velocità $x'(t)$ direttamente proporzionale al numero stesso. Analogamente, si supponga che il numero di predatori $y(t)$ diminuisca, in assenza di prede, ovvero per mancanza di cibo, con una velocità $y'(t)$ direttamente proporzionale ad $y(t)$ stesso. Si supponga inoltre che il numero di prede uccise dai predatori dipenda dalla possibilità che predatore e preda si incontrino, più precisamente che $x(t)$ diminuisca con una velocità direttamente proporzionale al prodotto $x(t)y(t)$ del numero di prede per il numero di predatori, e che $y(t)$ cresca in maniera analoga. In tali ipotesi, la dinamica delle due popolazioni è descritta dal seguente sistema di due ODE, note come equazioni di *Lotka-Volterra*:[4]

$$\begin{cases} x'(t) &= ax(t) - bx(t)y(t) \\ y'(t) &= -cy(t) + dx(t)y(t) \end{cases}$$

dove a, b, c, d sono costanti positive. A tali equazioni vanno aggiunte le condizioni iniziali:

$$x(t_0) = x_0, \quad y(t_0) = y_0,$$

che specificano il numero di individui ad un certo istante t_0, assunto come istante iniziale.

♣

In generale un problema a valori iniziali per un sistema di ODE del primo ordine, in forma esplicita, è del tipo:

$$\begin{cases} y_i'(t) &= f_i(t, y_1(t), y_2(t), \ldots, y_n(t)), \quad t \in I \\ y_i(t_0) &= y_{i,0} \\ & \qquad\qquad\qquad\qquad i = 1, \ldots, n \end{cases} \tag{3.7}$$

[4]Come nell'esempio 3.2, il modello in esame presuppone che $x(t)$ e $y(t)$ siano grandezze continue, mentre esse sono discrete.

Posto

$$Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix}, \quad F(t, Y) = \begin{bmatrix} f_1(t, y_1, y_2, \ldots, y_n) \\ f_2(t, y_1, y_2, \ldots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \ldots, y_n) \end{bmatrix},$$

il problema (3.7) si può riscrivere nella forma:

$$\begin{cases} Y'(t) &= F(t, Y(t)) \quad t \in I \\ Y(t_0) &= Y_0 \end{cases},$$

con $Y_0 = [y_{1,0}, y_{2,0}, \ldots, y_{n,0}]^T$, cioè si può rappresentare come problema a valori iniziali per una singola ODE vettoriale. Se l'ordine massimo delle derivate della funzione incognita è m, si parla di ODE di ordine m.

♣ **Esempio 3.6. (Moto del pendolo semplice)** Si consideri un pendolo semplice, cioè una particella di massa m sospesa ad un punto O mediante una fune inestendibile di lunghezza l e di massa trascurabile. La variabile s misura la lunghezza dell'arco descritto dal pendolo a partire dalla posizione di equilibrio, ovvero dal punto più basso della circonferenza di centro O e raggio l (Fig. 3.2).

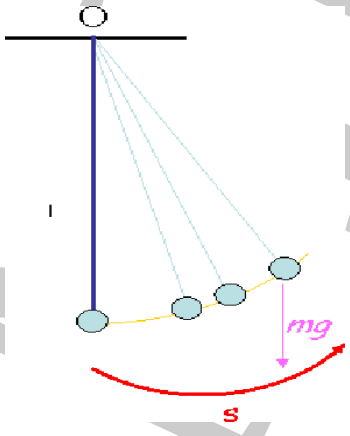

Figura 3.2: Pendolo semplice.

Applicando la legge di Newton, il moto della particella è regolato dalla ODE del secondo ordine:

$$s''(t) = -\frac{\lambda}{ml} s'(t) - \frac{g}{l} \sin s(t), \tag{3.8}$$

dove $s(t)$ è l'arco che la fune descrive con la verticale per O all'istante t, g è l'accelerazione di gravità e λ è un coefficiente dipendente dalla viscosità dell'aria. All'equazione (3.8) sono associate le condizioni iniziali

$$s(t_0) = \hat{s}, \quad s'(t_0) = 0,$$

che indicano posizione e velocità della particella all'istante iniziale t_0. La soluzione

$$s(t) = \epsilon a_0 e^{-\frac{\lambda}{2ml}t} cos \left[\omega_0 t - \frac{\epsilon^2 a_0^2}{\omega_0 \frac{\lambda}{2ml} 32}^{-\frac{\lambda}{ml}t} + b_0 \right] + \mathcal{O}(\epsilon^3)$$

con ϵ, a_0 e b_0 costanti e con $\omega_0 = \sqrt{\frac{g}{l}}$, descrive il moto del pendolo nell'intervallo di tempo $[t_0, T]$ in cui esso percorre, in senso orario, l'arco di lunghezza $s_0 = \epsilon a_0$, partendo dalla posizione iniziale, occupata all'istante t_0.
Successivamente, a partire dalla posizione $s_0 = \epsilon a_0$, per $t \geq T$ il moto cambia verso e prosegue regolato dalla legge

$$s(t) = s_0 e^{-\frac{\lambda}{2ml}t} cos\left[\omega_0 t - \frac{s_0^2}{\omega_0 \frac{\lambda}{2ml} 32}\left(e^{-\frac{\lambda}{ml}t} - 1\right)\right] + \mathcal{O}(\epsilon^2).$$

Per effetto della resistenza dell'aria il pendolo oscilla descrivendo archi la cui lunghezza diminuisce gradualmente fino a quando il pendolo si ferma. Aggiungendo l'ipotesi "s piccolo" rispetto a l, ovvero

$$sin\frac{s}{l} \approx \frac{s}{l},$$

il modello nella (3.8) risulta semplificato, ovvero descritto dalla ODE:

$$s''(t) = -\omega^2 s(t),$$

con $\omega^2 = \frac{g}{l}$, la cui soluzione è

$$s = s_0 + cos(\omega t + \delta),$$

con δ spostamento angolare all'istante $t = 0$ e s_0 arco di lunghezza massima, descritto dal pendolo. ♣

In generale, una ODE di ordine m, in forma esplicita, ha un'espressione del tipo:

$$y^{(m)}(t) = f(t, y(t), y'(t), \ldots, y^{(m-1)}(t)) \tag{3.9}$$

e per ottenere un'unica soluzione di tale equazione è necessario assegnare m condizioni in un punto t_0, su y e tutte le sue derivate fino a quella di ordine $m-1$. Si osservi che una ODE di ordine superiore al primo si può ricondurre ad un sistema di ODE del primo ordine. Infatti, posto

$$\begin{cases} w_1(t) &= y(t) \\ w_2(t) &= y'(t) \\ & \vdots \\ w_m(t) &= y^{(m-1)}(t) \end{cases}$$

l'equazione (3.9) si può riscrivere nel modo seguente:

$$\begin{cases} w_1'(t) &= w_2(t) \\ w_2'(t) &= w_3(t) \\ & \vdots \\ w_m'(t) &= f(t, w_1(t), w_2(t), \ldots, w_m(t)) \end{cases}$$

Per la ODE dell'esempio 3.6 si ha:

$$\begin{cases} s_1'(t) &= s_2(t) \\ s_2'(t) &= -\frac{\lambda}{ml} s_2(t) - \frac{g}{l} \sin s_1(t) \end{cases}$$

Nel seguito si rivolge quindi l'attenzione alle ODE del primo ordine.

3.2 Un metodo numerico di risoluzione: il metodo di Eulero

La necessità di avere a disposizione metodi numerici e software per la risoluzione di problemi a valori iniziali per equazioni differenziali ordinarie (ODE) nasce non solo dal fatto che in generale non è possibile ottenere le soluzioni in forma analitica ma soprattutto dal fatto che, in numerose applicazioni, è sentita l'esigenza di avere strumenti in grado di calcolare la soluzione numerica in maniera automatica e ripetitiva oltre che, chiaramente, affidabile ed efficiente ed il software numerico gioca quindi un ruolo fondamentale.

Si consideri il problema a valori iniziali (3.5), dove $I = [t_0, T]$ è un intervallo finito. Il più semplice dei metodi numerici per la risoluzione di tale problema si può ottenere applicando la formula di Taylor alla funzione incognita y. Si consideri lo sviluppo in serie di Taylor della funzione y di punto iniziale x_0 arrestato al termine del primo ordine:

$$y(x) = y(x_0) + (x - x_0)y'(x_0) + \tau(x_0, x - x_0)\,, \tag{3.10}$$

dove $\tau(x_0, x - x_0)$ è il resto della serie. Ponendo nella (3.10)

$$x_0 = t\,, \qquad x - x_0 = h$$

e quindi $x = t + h$, si ottiene:

$$y(t + h) = y(t) + hy'(t) + \tau(t, h) = y(t) + hf(t, y(t)) + \tau(t, h)\,. \tag{3.11}$$

Si consideri l'espressione di $y(t + h)$ ottenuta trascurando il resto $\tau(t, h)$:

$$y(t + h) \simeq y(t) + hf(t, y(t))\,. \tag{3.12}$$

Posto

$$h = \frac{T - t_0}{n}\,, \tag{3.13}$$

con n intero positivo, si considerino i punti dell'asse t, $t_0, t_1, \ldots, t_n = T$, con

$$t_i = t_0 + ih\,,$$

cioè si *discretizzi* l'intervallo $[t_0, T]$ utilizzando un *passo di discretizzazione* h. Sostituendo t_i a t e t_{i+1} a $t + h$ nella (3.12) si ha la relazione:

$$y(t_{i+1}) \simeq y(t_i) + hf(t_i, y(t_i))\,,$$

che consente di calcolare un'approssimazione y_{i+1} di y nel punto t_{i+1}, noto il valore iniziale $y(t_0) = y_0$:

$$y_{i+1} = y_i + hf(t_i, y_i)\,, \qquad i = 0, 1, 2, \ldots, n - 1\,. \tag{3.14}$$

La (3.14), unitamente al valore noto $y(t_0) = y_0$, descrive un metodo numerico per la risoluzione del problema (3.5), noto come *metodo di Eulero*.

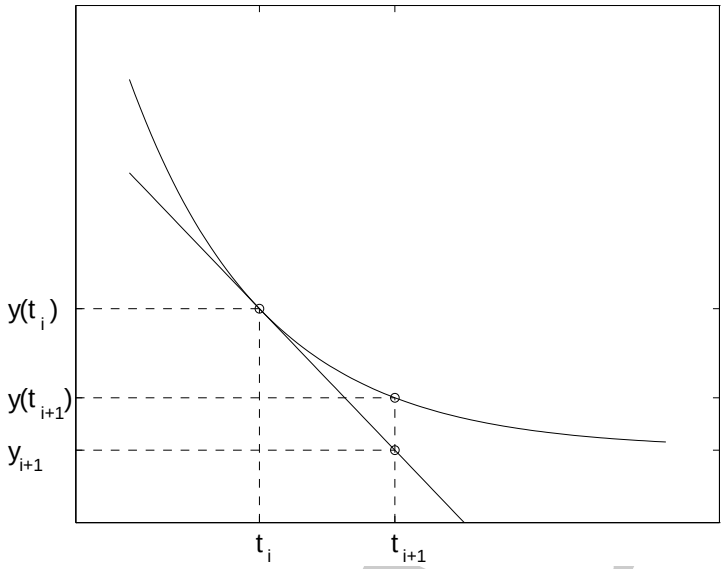

Figura 3.3: Interpretazione geometrica del metodo di Eulero.

Da un punto di vista geometrico si ha la seguente interpretazione del metodo di Eulero: nel punto $(t_i, y(t_i))$ si sostituisce la curva che rappresenta la soluzione $y(t)$ con la retta ad essa tangente e si calcola y_{i+1} come ordinata del punto di ascissa t_{i+1}. Tale situazione è illustrata graficamente in Figura 3.3.

La relazione (3.14) mostra che y_{i+1} dipende da y_i, ma non da y_j con $j < i$; per tale motivo il metodo di Eulero è detto *ad un passo*. Inoltre, il secondo membro della (3.14) non dipende da y_{i+1} e quindi il metodo è detto *esplicito* [5]. A partire dalla relazione (3.14) si ottiene immediatamente un algoritmo per la risoluzione numerica del problema (3.5) (Procedura 3.1).

[5] In generale, sussistono le definizioni seguenti.

Definizione 3.2.1. (Metodo ad un passo)
Un metodo numerico per la risoluzione del problema (3.5) si dice ad un passo, o one-step, se ha un'espressione del tipo

$$y_{i+1} = y_i + h\varphi(t_i, y_i, y_{i+1}, h, f), \tag{3.15}$$

cioè se y_{i+1} dipende solo da y_i.

Definizione 3.2.2. (Metodo ad un passo esplicito o implicito)
Un metodo numerico ad un passo per la risoluzione del problema (3.5) si dice esplicito se ha un'espressione del tipo

$$y_{i+1} = y_i + h\varphi(t_i, y_i, h, f),$$

cioè se φ non dipende da y_{i+1}. Altrimenti il metodo si dice implicito.

Si osservi che per il metodo di Eulero si ha $\varphi(t_i, y_i, h, f) = f(t_i, y_i)$.

```
procedure eulero(input: t0,T,y0,f,n,out: y)
  /# SCOPO: Risoluzione numerica di equazioni differenziali
  /#          ordinarie del primo ordine in forma esplicita
  /# SPECIFICHE PARAMETRI:
    var: t0  : reale              {estremo sinistro dell'intervallo}
    var: T   : reale              {estremo destro dell'intervallo}
    var: y0  : reale              {condizione iniziale}
    var: f   : funzione esterna   {funzione f(t,y)}
    var: n   : intero             {numero di sottointervalli}
    var: y   : array di reali     {restituisce la soluzione calcolata.}
  /# INIZIO ISTRUZIONI:
      h := (T - t0)/n;
      y(0) := y0;
      for i = 1,n do
        t := t0 + (i - 1) * h;
        y(i) := y(i - 1) + h * f(t, y(i - 1));
      endfor
  end procedure eulero
```

Procedura 3.1: Algoritmo per la risoluzione numerica del problema (3.5) con il metodo di Eulero

La *complessità di tempo* della Procedura 3.1 dipende essenzialmente dal numero di valutazioni della funzione $f(t, y)$, in quanto il numero di operazioni ad essa relative è in generale predominante rispetto alle altre operazioni eseguite dall'algoritmo. In questo caso il numero di valutazioni di f è n, cioè è uguale al numero di intervalli in cui è stato suddiviso l'intervallo di definizione del problema continuo. L'algoritmo ha dunque una complessità di tempo

$$T_{Eulero}(n) = \mathcal{O}(n) \text{ valutazioni di funzione.}$$

La *complessità di spazio* è

$$S_{Eulero}(n) = \mathcal{O}(n) \text{ variabili reali,}$$

i	t_i	$\tilde{u}(t_i)$	$y(t_i)$
0	0.0	1.000	1.000
1	0.1	0.900	0.909
2	0.2	0.820	0.835
3	0.3	0.756	0.774
4	0.4	0.705	0.725
5	0.5	0.664	0.684
6	0.6	0.631	0.651
7	0.7	0.605	0.623
8	0.8	0.584	0.601
9	0.9	0.567	0.583
10	1.0	0.554	0.568

Tabella 3.1: Applicazione del metodo di Eulero al problema (3.16). Valori della soluzione calcolata $\tilde{u}(t_i)$ e della soluzione effettiva $y(t_i)$.

in quanto l'algoritmo utilizza un array di reali, y, di dimensione $n+1$.

♣ **Esempio 3.7.** Si consideri il problema a valori iniziali:

$$\begin{cases} y'(t) &= -2y(t)+1, \quad t \in [0,1] \\ y(0) &= 1 \end{cases} \tag{3.16}$$

la cui soluzione è:

$$y(t) = \frac{1}{2}\left(e^{-2t}+1\right) \ .$$

Applicando ad esso la Procedura 3.1 con $n = 10$, ovvero $h = 0.1$, si ottengono i valori $\tilde{u}(t_i)$ riportati in Tabella 3.1.

♣

3.3 Analisi degli errori introdotti dal metodo di Eulero

Nell'esempio 3.7 i valori della soluzione effettiva e di quella calcolata con il metodo di Eulero sono diversi. Ciò è in accordo col fatto che la curva soluzione è approssimata ad ogni passo con la retta tangente ad essa in un determinato punto. In altri termini, l'espressione (3.12), dalla quale si ottiene il metodo di Eulero, corrisponde alla sostituzione,

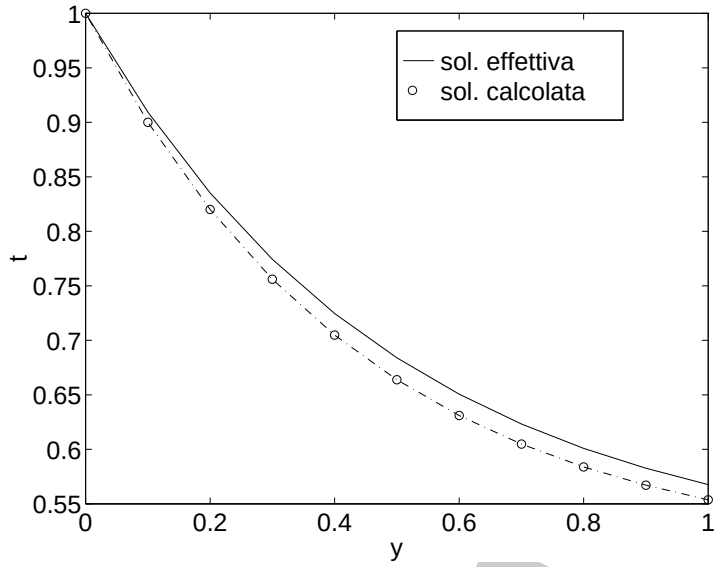

Figura 3.4: Applicazione del metodo di Eulero al problema (3.16): soluzione calcolata e soluzione effettiva. I grafici sono relativi ai dati in Tabella 3.1.

ad ogni passo, della derivata prima della funzione incognita con il rapporto incrementale:

$$y'(t) \simeq \frac{y(t+h) - y(t)}{h}\,.$$

Tale approssimazione non è però l'unica fonte di errore nel calcolo della soluzione.

♣ **Esempio 3.8.** Applicando la Procedura 3.1 al problema:

$$\begin{cases} y'(t) &= 0.01, \quad t \in [0,1] \\ y(0) &= 1 \end{cases}\,, \tag{3.17}$$

con $n = 10^5$, ovvero $h = 10^{-5}$, si ottengono i risultati riportati in Tabella 3.2 (i valori della soluzione effettiva $y(t_i)$ e quelli della soluzione calcolata $\tilde{u}(t_i)$ sono rappresentati con sette cifre significative). Si noti che, anche se la soluzione del problema (3.17) è $y(t) = 0.01t + 1$ e quindi la retta tangente alla curva soluzione in un qualsiasi punto coincide con la curva stessa, cioè la soluzione del problema discreto $u(t_i)$ coincide con la soluzione del problema continuo $y(t_i)$, la soluzione calcolata $\tilde{u}(t_i)$ è diversa. Tale situazione è illustrata anche graficamente, in Figura 3.5. Ciò è dovuto al fatto che l'algoritmo è stato eseguito in un sistema aritmetico a precisione finita e quindi i valori calcolati $\tilde{u}(t_i)$ sono affetti dall'errore di roundoff. ♣

Gli errori che si verificano nel calcolo della soluzione numerica di un problema a valori iniziali mediante il metodo di Eulero sono da addebitarsi a due cause fondamentali:

i	t_i	$\tilde{u}(t_i)$	$y(t_i)$
91000	0.91	1.010848	1.009100
92000	0.92	1.010967	1.009200
93000	0.93	1.011086	1.009300
94000	0.94	1.011206	1.009400
95000	0.95	1.011325	1.009500
96000	0.96	1.011444	1.009600
97000	0.97	1.011563	1.009700
98000	0.98	1.011683	1.009800
99000	0.99	1.011802	1.009900
100000	1.00	1.011921	1.010000

Tabella 3.2: Applicazione del metodo di Eulero al problema (3.17). Valori della soluzione calcolata e della soluzione effettiva .

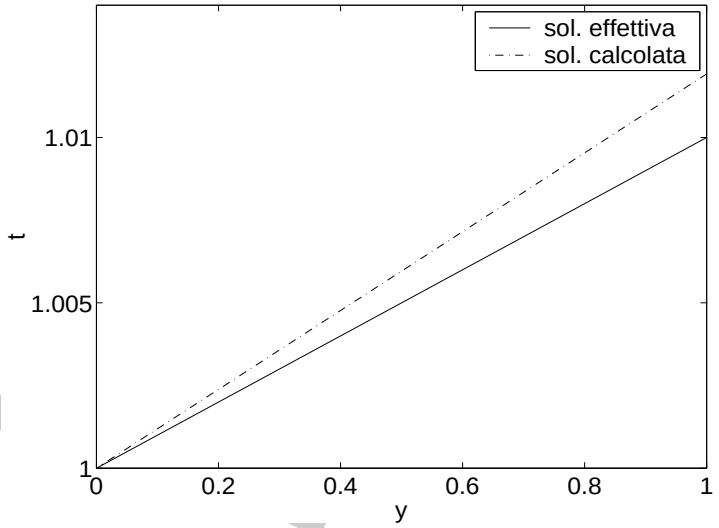

Figura 3.5: Applicazione del metodo di Eulero al problema (3.17): soluzione calcolata e soluzione effettiva. I grafici sono relativi ai dati in Tabella 3.2.

(1) errore di roundoff, che dipende dal sistema aritmetico;

(2) errore di discretizzazione, che dipende dal metodo numerico;

In questo paragrafo l'attenzione è dedicata allo studio di tali errori e della loro propagazione.

3.3.1 Il problema discreto come approssimazione del problema continuo

Si supponga di eseguire un solo passo, di ampiezza h, del metodo di Eulero. Introdotta la funzione $u(t)$, di I in $\Re$, il metodo di Eulero fornisce la soluzione del **problema discreto**

$$M_h(P) \equiv \begin{cases} \frac{u(t+h)-u(t)}{h} = f(t, y(t)), & \forall t \in I \\ u(t) = y(t) \end{cases} . \tag{3.18}$$

dove $y(t)$ è la soluzione del problema continuo $M(P)$ [6] (3.5). Dalla (3.18) si ha:

$$u(t+h) = y(t) + hf(t, y(t))$$

Per la formula di Taylor, si ha:

$$y(t+h) = y(t) + hf(t, y(t)) + \tau(t, h)$$

dove $\tau(t, h)$ esprime il resto di Taylor della funzione $y(t)$ nel punto t. Da cui:

$$y(t+h) - u(t+h) = \tau(t, h)$$

L'errore $\tau(t, h)$, detto *errore locale di troncamento del metodo di Eulero* nel punto t, si commette ad **ogni passo** del metodo di Eulero e influenza il valore ottenuto al passo successivo. Sussiste inoltre la seguente:

Definizione 3.3.2. *Si dice errore di discretizzazione di un metodo ad un passo del tipo (3.15) la quantità:*

$$\frac{y(t+h) - y(t)}{h} - \frac{u(t+h) - u(t)}{h} \equiv T(t, h) = \frac{\tau(t, h)}{h} . \tag{3.19}$$

L'errore di discretizzazione T misura la differenza tra il problema continuo $M(P)$ e il problema discreto $M_h(P)$.

[6] L'equazione in (3.18) è detta anche *equazione alle differenze*, in quanto la derivata $y'(t)$ è sostituita con la *differenza finita* $(u(t+h) - u(t))/h$. A tal proposito, si noti che esiste una classe di metodi detti metodi alle differenze finite per la risoluzione di equazioni differenziali (ordinarie o alle derivate parziali), i quali si basano proprio sull'idea di approssimare il problema continuo con un problema discreto sostituendo l'equazione differenziale con una opportuna equazione alle differenze. Sussiste la seguente:

Definizione 3.3.1. (Errore di discretizzazione - metodo ad un passo)
Si dice errore di discretizzazione di un metodo ad un passo del tipo (3.15) la quantità:

$$T(t, h, \varphi) = \frac{y(t+h) - y(t)}{h} - \varphi(t, y(t), y(t+h), h, f) .$$

Si utilizzerà la notazione $T(t, h)$ in luogo di $T(t, h, \varphi)$, cioè non si espliciterà la dipendenza da φ, sia per il metodo di Eulero sia per un qualsiasi metodo, ogni qual volta dal contesto si evincerà chiaramente di quale metodo si sta parlando.

Consistenza

Si vuole dare una stima dell'errore di discretizzazione del metodo di Eulero. Si ha:

Lemma 3.3.1. *Se $y(t) \in C^2([t_0, T])$ è la soluzione del problema continuo $M(P)$ (3.5), per l'errore di discretizzazione del metodo di Eulero si ha:*

$$T(t,h) = \mathcal{O}(h)\,.$$

Dimostrazione Dalla continuità della derivata seconda si ha per l'errore locale di troncamento:

$$\tau(t,h) = h^2 \frac{y''(\xi)}{2} \qquad (t < \xi < t+h).$$

Posto

$$C = \max_{[t_0,T]} |y''|\,,$$

risulta

$$|\tau(t,h)| \leq \frac{C}{2} h^2 \tag{3.20}$$

Quindi da (3.19) e (3.20) si ha per l'errore di discretizzazione:

$$|T(t,h)| = \frac{|\tau(t,h)|}{h} \leq \frac{C}{2} h$$

da cui la tesi. ■

Dal Lemma 3.3.1 si evince che per ogni punto t fissato, al tendere del passo h a 0 l'errore di discretizzazione tende a 0 con ordine di infinitesimo uguale a 1. A tal proposito si dice che *il metodo di Eulero è consistente di ordine 1* [7]. La consistenza esprime il fatto che il problema discreto (3.18) è un'approssimazione del problema continuo (3.5) e l'ordine esprime quanto tale approssimazione sia accurata. La consistenza costituisce una condizione necessaria per ottenere un'approssimazione della soluzione.

[7] In generale si ha la seguente definizione:

Definizione 3.3.3. (Consistenza-metodo ad un passo)
Un metodo del tipo (3.15) si dice consistente se, per t fissato, risulta

$$\lim_{h\to 0} T(t,h) = 0.$$

Se, inoltre, si ha

$$|T(t,h)| \leq D h^p\,,$$

con $D > 0$ costante, il metodo si dice consistente di ordine p. In generale, p è detto ordine del metodo.

Convergenza

Anche se il metodo di Eulero è consistente, ovvero il problema discreto (3.18) tende al problema continuo (3.5), $\forall t \in I$, non è detto che la soluzione $u(t_i) = y_i$ del problema discreto:

$$M_h(P) \equiv \begin{cases} \frac{u(t_{i+1})-u(ti)}{h} & = & f(t_i, u(t_i)) \quad i = 0, 1, \ldots \\ u(t_0) & = & y(t_0) \end{cases} \tag{3.21}$$

in un punto t_i, ottenuta a partire dalla condizione iniziale $u(t_0) = y(t_0)$, sia una buona approssimazione della soluzione $y(t_i)$ del problema continuo (3.5). È naturale allora chiedersi quale sia la differenza tra tali soluzioni. In particolare, per il metodo di Eulero, si noti che calcolare y_1 a partire da y_0 equivale ad applicare un solo passo del metodo di Eulero al seguente problema continuo

$$\begin{cases} y'(t) & = & f(t, y(t)) \\ y(t_0) & = & u(t_0) = y_0 \end{cases}.$$

Analogamente, all'i-mo passo, calcolare y_i a partire dal valore y_{i-1} equivale ad applicare solo un passo del metodo di Eulero al problema continuo

$$\begin{cases} y'(t) & = & f(t, y(t)) \\ y(t_{i-1}) & = & u(t_{i-1}) = y_{i-1} \end{cases}.$$

Il metodo di Eulero comporta quindi, ad ogni passo, il passaggio da una soluzione ad un'altra della famiglia di soluzioni dell'equazione differenziale (3.3), come illustrato nelle Figure 3.6 e 3.7, che si riferiscono, rispettivamente, al caso in cui l'equazione differenziale è $y' = y$ e $y' = -y$.

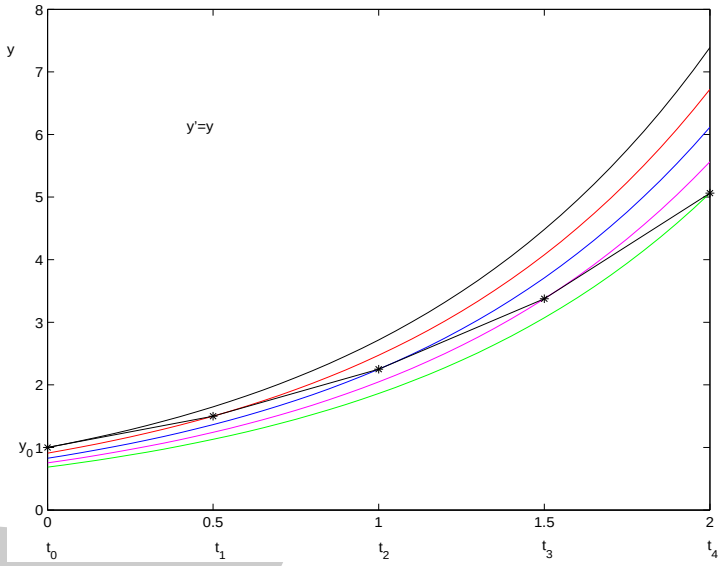

Figura 3.6: Esecuzione del metodo di Eulero: passaggio da una soluzione all'altra della famiglia di soluzioni dell'equazione $y' = y$.

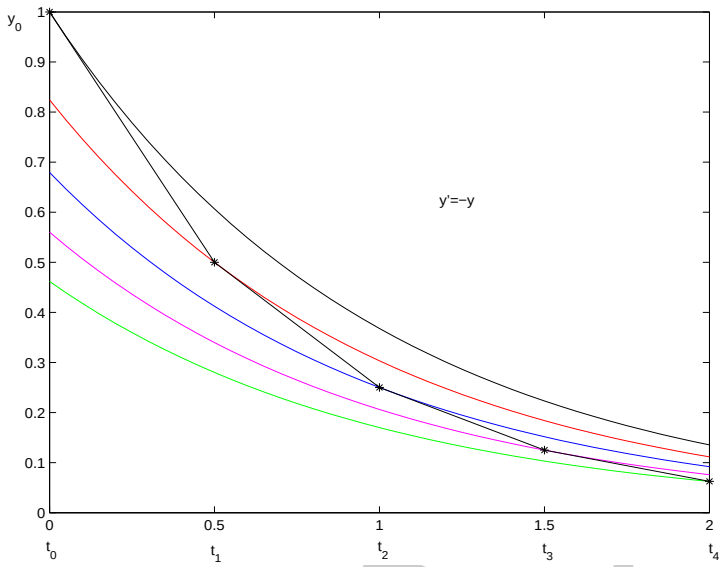

Figura 3.7: Esecuzione del metodo di Eulero: passaggio da una soluzione all'altra della famiglia di soluzioni dell'equazione $y' = -y$.

La soluzione ottenuta dal metodo di Eulero, in un generico punto t_i, risente così di tutti gli errori commessi nei punti precedenti $(t_{i-1}, t_{i-2}, \ldots, t_2, t_1)$. Si dà quindi la seguente:

Definizione 3.3.4. *Si dice errore globale di troncamento del metodo di Eulero, nel punto t_i, la quantità*

$$E_i(h) = y(t_i) - u(t_i) \ , i = 0, 1, 2, \ldots \tag{3.22}$$

cioè la differenza tra la soluzione del problema continuo $M(P)$ (3.5) e la soluzione del problema discreto $M_h(P)$ (3.21).

♣ **Esempio 3.9.** Si consideri nuovamente il problema

$$\begin{cases} y'(t) &= -2y(t) + 1, \quad t \in [0, 1] \\ y(0) &= 1 \end{cases} \tag{3.23}$$

dell'esempio 3.7 e si applichi ad esso la Procedura 3.1 con differenti valori del passo h, precisamente con $h = 0.2, 0.1, 0.05$. Si considerino ad esempio i valori calcolati della soluzione per $t = 0.4, 1$. I risultati riportati in Tabella 3.3 mostrano che l'errore globale di troncamento decresce al decrescere di h e, in particolare, al dimezzarsi di h si riduce di circa la metà.

Tale situazione si verifica in generale in ogni punto t_i della discretizzazione, come mostrato dai grafici in Figura 3.8. ♣

Al fine di studiare l'errore globale di troncamento si premette il lemma:

t	y(t)	h	$\tilde{u}(t)$	errore
0.4	0.725	0.20	0.680	0.447×10^{-1}
		0.10	0.705	0.199×10^{-1}
		0.05	0.715	0.943×10^{-2}
1.0	0.568	0.20	0.539	0.288×10^{-1}
		0.10	0.554	0.140×10^{-1}
		0.05	0.561	0.688×10^{-2}

Tabella 3.3: Applicazione del metodo di Eulero al problema (3.23). Soluzione calcolata ed errore globale di troncamento in $t = 0.4, 1$, per $h = 0.2, 0.1, 0.05$.

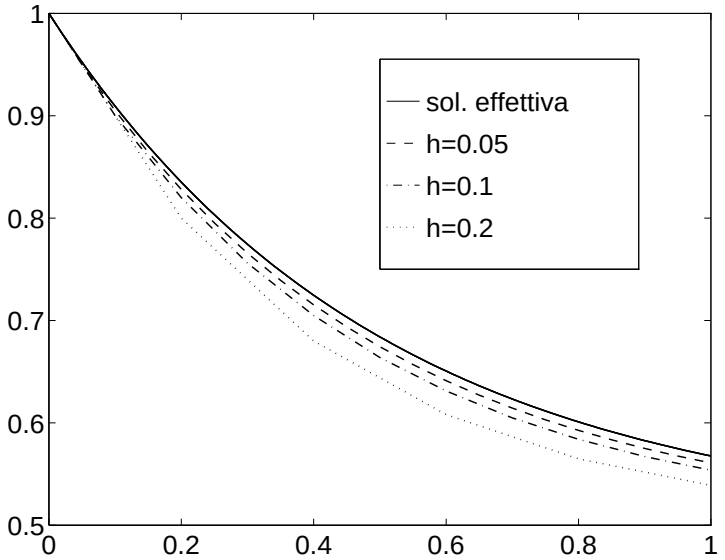

Figura 3.8: Applicazione del metodo di Eulero al problema (3.23). Soluzioni calcolate con $h = 0.2, 0.1, 0.05$ e soluzione effettiva.

Lemma 3.3.2. *Siano dati un intero positivo n e le costanti $K > 0$ e $H \geq 0$. Se, per le costanti M_i con $i = 0, 1, 2, \ldots, n$, si ha:*

$$|M_i| \leq (1 + K)|M_{i-1}| + H, \qquad \text{per } i = 1, 2, \ldots, n$$

allora:

$$|M_i| \leq e^{nK}|M_0| + \frac{e^{nK} - 1}{K} H \qquad \text{per } i = 1, 2, \ldots, n \ . \tag{3.24}$$

Dimostrazione Mostriamo preliminarmente che

$$|M_i| \leq (1 + K)^i |M_0| + \frac{(1 + K)^i - 1}{K} H \qquad \text{per } i = 1, 2, \ldots, n \ . \tag{3.25}$$

Procediamo per induzione. Per $i = 1$, si ha la base d'induzione. Infatti:

$$|M_1| \leq (1 + K)|M_0| + \frac{(1 + K) - 1}{K} H = (1 + K)|M_0| + H \ .$$

Supponiamo la (3.25) valida per un dato $i < n$. Si ha:

$$\begin{aligned} |M_{i+1}| &\leq (1+K)|M_i| + H \leq (1+K)\left((1+K)^i|M_0| + \frac{(1+K)^i - 1}{K}H\right) + H = \\ &= (1+K)^{i+1}|M_0| + \left\{(1+K)\frac{(1+K)^i - 1}{K} + 1\right\}H = \\ &= (1+K)^{i+1}|M_0| + \frac{(1+K)^{i+1} - 1}{K}H \quad . \end{aligned}$$

Quindi la (3.25) vale anche per $i+1$ e si ha la tesi. Dallo sviluppo in serie di MacLaurin di e^K, si ha:

$$e^K = 1 + K + \cdots + \frac{K^r}{r!} + \cdots > 1 + K \ ,$$

da cui

$$(1+K)^i < e^{iK} \leq \ e^{nK}. \tag{3.26}$$

Sostituendo la (3.26) nella (3.25) si ottiene la (3.24). ■

Per l'errore globale di troncamento del metodo di Eulero si ha:

Teorema 3.3.1. [Errore globale di troncamento del metodo di Eulero] *Si supponga che $f(t,y)$ soddisfi le ipotesi del Teorema 3.1.1 e sia $y \in C^2([t_0,T])$ la soluzione del problema continuo $M(P)$ (3.5). Per l'errore globale di troncamento $E_i(h)$ del metodo di Eulero, definito in (3.22), si ha:*

$$|E_i(h)| \leq e^{L(T-t_0)}|E_0| + \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\frac{C}{2}h \quad , \quad i = 1, 2, \ldots, n \tag{3.27}$$

dove E_0 indica l'errore nel dato iniziale e $C = max_{[t_0,T]}|y''|$.

Dimostrazione Dalla (3.11) e dalla (3.18), si ha:

$$\begin{aligned} E_i(h) &= y(t_i) - u(t_i) = \\ &= y(t_{i-1}) - u(t_{i-1}) + h\left[f(t_{i-1}, y(t_{i-1})) - f(t_{i-1}, u(t_{i-1}))\right] + \tau(t_{i-1}, h)\,. \end{aligned}$$

Passando ai valori assoluti e usando la (3.6) si ottiene:

$$|E_i(h)| \leq (1+hL)|y(t_{i-1}) - u(t_{i-1})| + |\tau(t_{i-1}, h)| = (1+hL)|E_{i-1}(h)| + |\tau(t_{i-1}, h)|\,,$$

ricordando la (3.20), si ha:

$$|E_i(h)| \leq (1+hL)|E_{i-1}(h)| + \frac{C}{2}h^2.$$

Applicando il Lemma 3.3.2, con $K = hL$ e $H = \frac{C}{2}h^2$, si ricava:

$$|E_i(h)| \leq e^{nhL}|E_0| + \frac{\left(e^{nhL} - 1\right)}{Lh}\frac{C}{2}h^2 \ .$$

Dalla (3.13) segue la tesi. ■

Dal Teorema 3.3.1 si ricava inoltre:

Corollario 3.3.1. *Nelle ipotesi del Teorema 3.3.1, supposto $E_0 = 0$, per l'errore globale di troncamento del metodo di Eulero si ha:*

$$E_i(h) = \mathcal{O}(h) \ .$$

Dimostrazione Ponendo $E_0 = 0$ nella (3.27), si ha la tesi. ■

Dal Corollario 3.3.1 si evince che se si fa tendere h a 0, $E_i(h)$ tende a zero con ordine di infinitesimo 1. A tal proposito si dice che *il metodo di Eulero è convergente di ordine 1*[8].

La convergenza di un metodo esprime il fatto che la soluzione del problema discreto descritto dal metodo stesso è un'approssimazione della soluzione del problema continuo. L'ordine di convergenza esprime l'accuratezza di tale approssimazione. Si osservi che tale ordine coincide con l'ordine di consistenza.

3.3.2 La risoluzione del problema discreto

La convergenza di un metodo assicura che, per $h \to 0$, la soluzione del problema discreto converge alla soluzione del problema continuo, ma non assicura che, per un valore di h fissato, pur se scelto sufficientemente piccolo, il metodo sia in grado di calcolare una soluzione "accettabile". In questo paragrafo l'attenzione è quindi posta sulla risoluzione del problema discreto definito dal metodo e sulla scelta dell'opportuno valore di h.

Stabilità

♣ **Esempio 3.10.** Si consideri il problema a valori iniziali:

$$\begin{cases} y'(t) &= -15y(t) \quad t \in [0, 0.8] \\ y(0) &= 0.6 \end{cases} \ . \tag{3.28}$$

[8] In generale sussiste la seguente definizione:

Definizione 3.3.5. (Convergenza - metodo ad un passo)
Un metodo del tipo (3.15) si dice convergente se, posto $E_0 = 0$, per $t_i = t_0 + ih$ fissato risulta

$$\lim_{h \to 0} E_i(h) = 0.$$

Se, inoltre, si ha

$$|E_i(h)| \leq Kh^p \, ,$$

con $K > 0$ costante, il metodo si dice convergente di ordine p.

Come nel caso dell'errore globale di troncamento, questa definizione non si applica solo ad i metodi ad un passo, ma ha una validità più generale.

Utilizzando la Procedura 3.1 con $h = 0.16, 0.08, 0.04$ si ottengono i risultati riportati in Figura 3.9. Si osserva che per $h = 0.16$ l'errore si amplifica in modo tale da condurre ad una soluzione che è completamente inaccettabile.

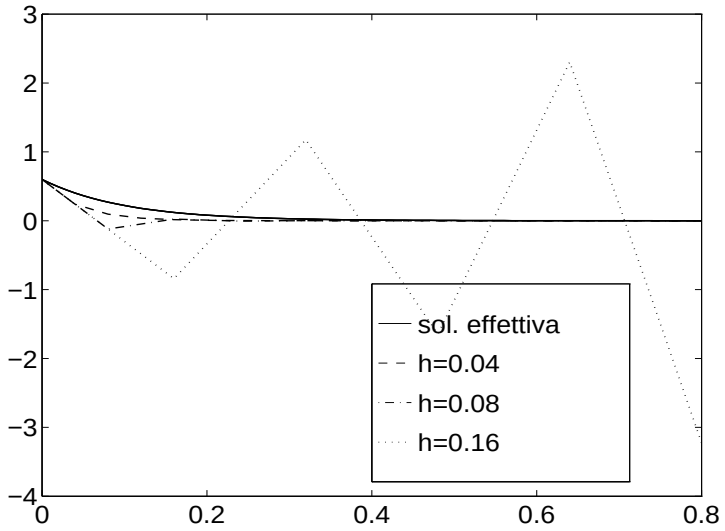

Figura 3.9: Applicazione del metodo di Eulero al problema (3.28). Soluzione calcolata per $h = 0.16, 0.08, 0.04$ e soluzione effettiva.

♣

La relazione (3.27) mostra che l'errore globale di troncamento non è la somma degli errori locali di troncamento. Ciò è illustrato graficamente nelle Figure 3.10 e 3.11, che si riferiscono, rispettivamente, al caso in cui l'equazione differenziale è $y' = y$ e $y' = -y$. La stima dell'errore globale di troncamento mostra che l'errore è somma di due componenti: una dovuta "all'accumulo" degli errori locali di troncamento $\tau(t_i, h)$, l'altra dovuta alla propagazione dell'errore presente nel dato iniziale. Siamo ora interessati ad indagare sulla propagazione degli errori durante l'esecuzione del metodo di Eulero. In pratica si è interessati all'analisi della stabilità del metodo.
Siano $u(t_i) = y_i$ $(i = 0, 1, 2, \ldots)$ la soluzione del problema discreto (3.18) e $v(t_i) = v_i$ $(i = 0, 1, 2, \ldots)$ la soluzione del problema discreto perturbato:

$$\begin{cases} \frac{v(t_{i+1}) - v(t_i)}{h} = f(t_i, v(t_i)), & i = 0, 1, \ldots \\ v(t_0) = y(t_0) + \epsilon_0 \end{cases} \tag{3.29}$$

dove ϵ_0 esprime la perturbazione introdotta sul dato iniziale $y(t_0)$. La quantità

$$\epsilon_i(h) = v(t_i) - u(t_i) = v_i - y_i$$

rappresenta quindi l'errore dovuto alla propagazione di ϵ_0 ed è detta *errore di propagazione*. Lo studio della propagazione di tale errore equivale ad indagare sulle caratteristiche di stabilità del metodo di Eulero e, a tal fine, si pongono due problemi:

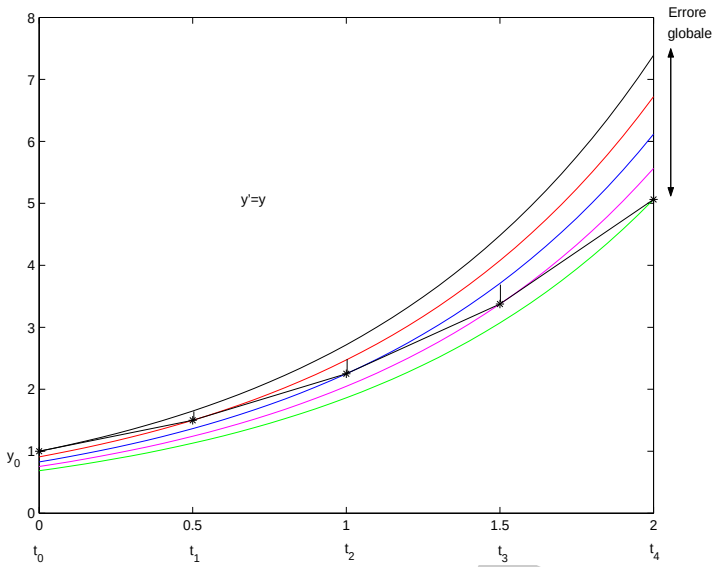

Figura 3.10: Errore globale ed errore locale di troncamento nel metodo di Eulero per l'equazione $y' = y$.

(a) Sia $t \in [t_0, \infty[$, posto $t = t_i = t_0 + ih$, come si comporta l'errore di propagazione $\epsilon_i(h)$ per h fissato e per $i \to \infty$ (cioè per $t \to \infty$) ?

(b) Sia $t \in [t_0, T]$, posto $t = t_i = t_0 + ih$, come si comporta l'errore di propagazione $\epsilon_i(h)$ per t fissato (e quindi ih fissato) e per $h \to 0$ (cioè $i \to \infty$) ?

Si osservi che, sebbene i due problemi abbiano una formulazione simile, il primo, dove il valore di h è fissato, è legato alla sola soluzione del problema discreto, mentre il secondo, dove $h \to 0$, è ovviamente rilevante ai fini della convergenza.

(a) Stabilità assoluta

Analizziamo in particolare come si propaga $\epsilon_i(h)$ nell'intervallo $[t_0, \infty[$, nel caso in cui il metodo di Eulero sia applicato al seguente problema a valori iniziali, detto *problema test*:[9]

[9]La scelta del problema test come problema di riferimento è giustificata dal fatto che ci si può ricondurre localmente a tale problema, mediante una linearizzazione opportuna della funzione f in (3.5). Infatti in un intorno di un generico punto y si ha:

$$f(t, v) \simeq f(t, y) + \frac{\partial f}{\partial y}(t, y)(v - y)$$

e quindi, posto $\epsilon(t) = v(t) - y(t)$, con $v(t)$ soluzione della ODE in (3.5) corrispondente ad un dato v_0 "sufficientemente vicino" a y_0, risulta

$$\epsilon'(t) = v'(t) - y'(t) = f(t, v(t)) - f(t, y(t)) \simeq \frac{\partial f}{\partial y}(t, y(t))\epsilon(t).$$

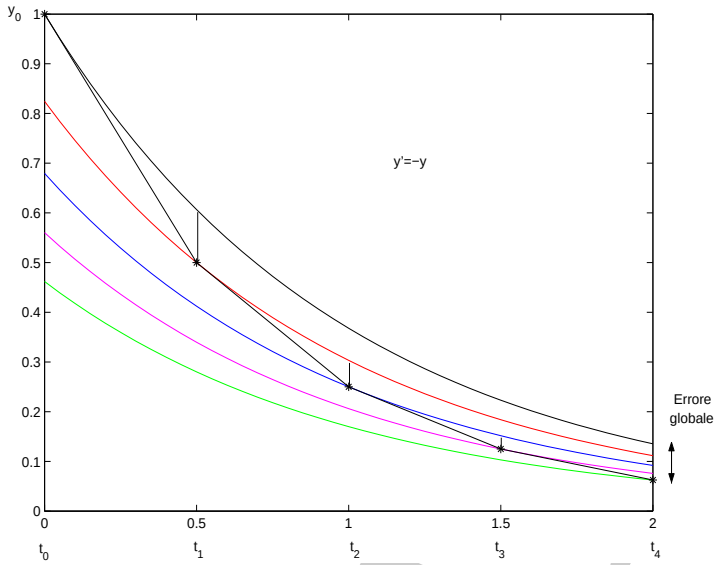

Figura 3.11: Errore globale ed errore locale di troncamento nel metodo di Eulero per l'equazione $y' = -y$.

$$\begin{cases} y'(t) &= \lambda y(t), \quad t \in [t_0, +\infty[\quad \lambda \in R \\ y(t_0) &= y_0 \end{cases} . \tag{3.30}$$

Sussiste la seguente definizione:

Definizione 3.3.6. (Stabilità assoluta - metodo ad un passo)
Siano y_i e v_i le soluzioni, determinate con un metodo del tipo (3.15), dei problemi test corrispondenti, rispettivamente, ai dati iniziali y_0 e $v_0 \neq y_0$. Il metodo considerato si dice assolutamente stabile, per un dato valore di h e per un dato valore di λ, se

$$|\epsilon_0| = |y_0 - v_0| < \delta \implies |\epsilon_i(h)| = |y_i - v_i| < \delta \quad i = 1, 2, 3, \ldots .$$

Si ha:

Proposizione 3.3.1. *Il metodo di Eulero è assolutamente stabile se e solo se*

$$(\lambda < 0\,, \quad h \leq -2/\lambda) \quad \text{oppure} \quad \lambda = 0.$$

Supposto, per semplicità,

$$\frac{\partial f}{\partial y}(t, y) = \lambda = costante,$$

si ha, a meno delle approssimazioni effettuate,

$$\epsilon'(t) = \lambda \epsilon(t),$$

ovvero si ottiene l'equazione test, della quale, come osservato precedentemente, $\epsilon_i(h)$ è la soluzione ottenuta con il metodo di Eulero.

Dimostrazione Essendo $f(t,y) = \lambda y$, i problemi discreti (3.18) e (3.29) assumono la forma equivalente

$$\begin{cases} y_i = (1+h\lambda)y_{i-1}, & i = 1,2,3,\dots \\ y_0 = y(t_0) \end{cases} . \tag{3.31}$$

e

$$\begin{cases} v_i = (1+h\lambda)v_{i-1}, & i = 1,2,3,\dots \\ v_0 = y(t_0) + \epsilon_0 \end{cases} .$$

Quindi per l'errore di propagazione $\epsilon_i(h)$ si ha:

$$\epsilon_i(h) = v_i - y_i = (1+h\lambda)v_{i-1} - (1+h\lambda)y_{i-1} = (1+h\lambda)\epsilon_{i-1}(h)\,. \tag{3.32}$$

Applicando ripetutamente la (3.32), si ricava:

$$\epsilon_i(h) = (1+h\lambda)^i\epsilon_0\,. \tag{3.33}$$

Dalla (3.33) si deduce:

$$\lim_{i\to\infty} |\epsilon_i(h)| = \begin{cases} 0 & \text{se } |1+h\lambda| < 1 \\ |\epsilon_0| & \text{se } |1+h\lambda| = 1 \\ +\infty & \text{se } |1+h\lambda| > 1 \end{cases} .$$

Quindi l'errore di propagazione non si amplifica, e il metodo di Eulero è assolutamente stabile, se e solo se

$$|1+h\lambda| \le 1 \iff -1 \le 1+h\lambda \le 1 \iff -2 \le h\lambda \le 0. \tag{3.34}$$

Dalla (3.34) segue la tesi. ■

La stabilità assoluta di un metodo dipende dunque dal problema attraverso λ, e dal metodo attraverso il passo di discretizzazione h, ed esprime il fatto che, fissati tali parametri, l'errore nel dato iniziale si propaga senza amplificarsi in ogni valore della soluzione calcolata. Nell'esempio 3.10 si ha $\lambda = -15$ e $-2/\lambda \simeq 0.13$; il metodo di Eulero non è quindi assolutamente stabile per $h = 0.16$ e ciò spiega perché la soluzione calcolata con tale valore di h sia inaccettabile. Si noti infine che il metodo di Eulero non è mai assolutamente stabile se $\lambda > 0$, ovvero l'errore di propagazione cresce illimitatamente[10].

(b) Zero-stabilità e teorema di Dahlquist

Analizziamo come si comporta l'errore di propagazione nel caso in cui si risolvono i problemi discreti (3.18) e (3.29) nell'intervallo finito $[t_0, T]$. Sussiste la seguente definizione:

[10]Tuttavia se si considera *l'errore relativo di propagazione*, $\overline{\epsilon}_i(h) = \epsilon_i(h)/y_i(h)$, per il metodo di Eulero applicato al problema test (3.30), si ha:

$$|\overline{\epsilon}_i(h)| = \frac{|(1+h\lambda)^i\epsilon_0|}{|(1+h\lambda)^i y_0|} = \frac{|e_0|}{|y_0|},$$

Cioè l'errore relativo resta costante qualunque sia il passo h. È bene però osservare che esiste una classe di problemi, di particolare interesse per le applicazioni, noti come *problemi stiff*, per cui il concetto di errore relativo non fornisce informazioni sulla bontà delle soluzioni e l'errore assoluto gioca un ruolo fondamentale.

Definizione 3.3.7. (Zero-stabilità - metodo ad un passo)
Un metodo del tipo (3.15) si dice zero-stabile se esistono $h_0 > 0$ e $M > 0$ tali che

$$|\epsilon_i(h)| \leq M|\epsilon_o|, \text{ per } h < h_0 \text{ e } i = 1, 2, \ldots, n \text{ con } n = \frac{T - t_0}{h} .$$

La zero-stabilità esprime il fatto che, se h è sufficientemente piccolo ($h < h_0$), l'errore di propagazione $|\epsilon_i(h)|$ si mantiene limitato, in tutto l'intervallo $[t_0, T]$.
Si ha:

Proposizione 3.3.2. *Il metodo di Eulero è zero-stabile.*

Dimostrazione Ricordando che $u(t_i) = y_i$ e $v(t_i) = v_i$, si ha:

$$\begin{aligned} \epsilon_i(h) &= v_{i-1} + hf(t_{i-1}, v_{i-1}) - y_{i-1} - hf(t_{i-1}, y_{i-1}) = \\ &= v_{i-1} - y_{i-1} + h(f(t_{i-1}, v_{i-1}) - f(t_{i-1}, y_{i-1})) . \end{aligned}$$

Passando ai valori assoluti e usando la (3.6) si ottiene:

$$|\epsilon_i(h)| \leq |v_{i-1} - y_{i-1}| + hL|v_{i-1} - y_{i-1}| = (1 + hL)|\epsilon_{i-1}(h)|$$

Applicando il Lemma 3.3.2, con $K = hL$ e $H = 0$, si ottiene:

$$|\epsilon_i(h)| \leq e^{nhL}|\epsilon_0|,$$

e ricordando la (3.13):

$$|\epsilon_i(h)| \leq e^{L(T-t_0)}|\epsilon_0|. \tag{3.35}$$

Dalla (3.35) si deduce che esiste $M = e^{(T-t_0)L} > 0$ tale che

$$|\epsilon_i(h)| \leq M|\epsilon_0|, \text{ per } i = 1, 2, \ldots, n, \text{ con } n = \frac{T - t_0}{h} . \tag{3.36}$$

Dalla (3.36) si ha che se ϵ_0 tende a zero anche $\epsilon_i(h)$ tende a zero, da cui se $|\epsilon_0|$ è "sufficientemente piccolo", $|\epsilon_i(h)|$ si può mantenere al di sotto di un valore fissato ovvero si ha la zero-stabilità

■

La zero-stabilità è detta anche *stabilità* oppure *D-stabilità*, in onore di G.Dahlquist, che formalizzò tale concetto e lo analizzò per una classe di metodi alle differenze finite, a cui il metodo di Eulero appartiene[11]. Tra l'altro a Dahlquist va il merito di aver provato che la zero-stabilità è fortemente connessa alle proprietà di consistenza e di convergenza, enunciando il teorema:

Teorema 3.3.2. [di Dahlquist o di equivalenza di Lax-Richtmeyer] [12]
Un metodo numerico del tipo [13] *(3.15) consistente è convergente se e solo se è zero-stabile.*

[11] Per un approfondimento si vedano [4] ed i riferimenti bibliografici ivi citati.

[12] Il teorema è noto come *teorema di Dahlquist*, in quanto dimostrato da Dahlquist nel 1956 [3] per una particolare classe di metodi alle differenze finite, dei quali il metodo di Eulero fa parte, e come *teorema di Lax-Richtmeyer*, in quanto pubblicato da Lax e Ritchmeyer nello stesso anno [9], con equazioni alle differenze finite lineari derivanti dalla discretizzazione di problemi ai valori iniziali descritti da equazioni differenziali alle derivate parziali.

[13] Vedere note 5, 7 e 8.

Il Teorema 3.3.2 ha una validità generale nell'ambito dei metodi numerici per la risoluzione di problemi ai valori iniziali[14] e costituisce uno strumento di importanza fondamentale nello studio di tali metodi. Infatti, è spesso "più facile" verificare la consistenza e la zero-stabilità di un metodo anzichè direttamente la sua convergenza.

Condizionamento del problema discreto

Analizziamo come si comporta l'indice di condizionamento nel metodo di Eulero applicato al problema test (3.30). Si ha:

Proposizione 3.3.3. *Il metodo di Eulero è ben condizionato*[15] *se e solo se*

$$(\lambda < 0\,,\ \ h \leq -2/\lambda) \quad \text{oppure} \quad \lambda = 0.$$

Dimostrazione Indicata con F_i la funzione che al dato iniziale y_0 associa il valore della soluzione y_i del problema discreto (3.31), ottenuto applicando il metodo di Eulero al problema test (3.30), si ha:

$$F_i(y_0) = y_i = (1 + h\lambda)^i y_0$$

quindi l'indice di condizionamento del problema (3.31), nel punto t_i, indicato don μ_i, è dato da

$$\mu_i(h, i, \lambda) = |F_i'(y_0)| = |1 + h\lambda|^i. \tag{3.37}$$

Dalla (3.37) si deduce che il metodo di Eulero è ben condizionato se e solo se

$$|\mu_i(h, i, \lambda)| \leq 1 \iff |1 + h\lambda|^i \leq 1 \iff |1 + h\lambda| \leq 1 \iff -2 \leq h\lambda \leq 0. \tag{3.38}$$

Dalla (3.38) segue la tesi. ■

Si osservi che il risultato della Proposizione 3.3.3 coincide con quello della Proposizione 3.3.1. D'altra parte, in questo caso, lo studio della stabilità assoluta del metodo di Eulero coincide con lo studio del condizionamento del problema discreto individuato dal metodo stesso. Per quanto riguarda la scelta del problema test, o meglio del problema discreto ad esso corrispondente, nell'analisi del condizionamento, valgono le stesse motivazioni addotte nel §3.3.2.
Si osservi infine che, se si considera il problema discreto (3.18) ottenuto applicando il metodo di Eulero al problema differenziale (3.5), i dati del problema sono il valore iniziale y_0 e la funzione f e quindi un'analisi completa del condizionamento deve tener conto degli effetti generati da perturbazioni su entrambi i dati.

Effetti dell'errore di roundoff

L'analisi effettuata nel §3.3.1 mostra che il metodo di Eulero può fornire un'approssimazione della soluzione del problema (3.5) con un grado di accuratezza arbitrario, a

[14] Per una sua formulazione e dimostrazione in casi più generali si vedano, ad esempio, [2, 3, 7].
[15] In tale analisi si fa riferimento al condizionamento assoluto del problema.

patto che il parametro h possa essere scelto "sufficientemente piccolo". Tale analisi non tiene però conto degli errori introdotti nell'esecuzione delle operazioni aritmetiche. In effetti, si calcola, con un numero finito di passi del metodo, una soluzione numerica del problema discreto (3.18), usando un opportuno valore di h in un sistema aritmetico a precisione finita.

Si consideri il problema test (3.30). L'analisi effettuata nel paragrafo precedente mostra che, se $\lambda < 0$, per $h \leq -2/\lambda$ il metodo di Eulero non amplifica l'errore introdotto sul dato iniziale. Inoltre la soluzione calcolata dal metodo, in generale, costituisce un'approssimazione della soluzione del problema continuo con un errore $\mathcal{O}(h)$, che decresce linearmente al decrescere di h. Tali considerazioni possono indurre a scegliere il valore del parametro h piccolo in modo arbitrario. Tuttavia, se si eseguono le operazioni in un sistema aritmetico a precisione finita si osserva che riducendo il valore di h non sempre si ha una riduzione analoga dell'errore sulla soluzione calcolata.

♣ **Esempio 3.11. (Errore di roundoff nel metodo di Eulero)** Si consideri il problema differenziale

$$\begin{cases} y'(t) &= -2y(t), \quad t \in [0,1] \\ y(0) &= 1 \end{cases} \tag{3.39}$$

e si applichi ad esso la Procedura 3.1 con passo $h = 10^{-4}, h = 10^{-5}, h = 10^{-6}$ (cioè con $n = 10^4, 10^5, 10^6$). In Tabella 3.4 sono riportati i valori in $t = 0.2, 0.4, 0.6, 0.8, 1$ delle soluzioni calcolate con i diversi passi di discretizzazione. Si noti che per $h = 10^{-4}$ e $h = 10^{-5}$ l'errore nella soluzione calcolata è dello stesso ordine di grandezza di h, mentre per $h = 10^{-6}$ tale errore è dell'ordine di 10^{-3} o 10^{-4} ed è maggiore degli errori corrispondenti agli altri valori di h. In altri termini, al decrescere di h l'accuratezza del metodo di Eulero diminuisce.

♣

Indichiamo con $\tilde{u}(t_i)$ il valore calcolato, in un sistema aritmetico floating-point a precisione finita, della soluzione $u(t_i)$ del problema discreto (3.18) nel punto t_i. Sussiste la seguente:

Definizione 3.3.8. *Si dice errore globale di roundoff del metodo di Eulero, nel punto t_i, la quantità*

$$R_i(h) = u(t_i) - \tilde{u}(t_i) \tag{3.40}$$

Applicando un solo passo del metodo di Eulero e indicando con ρ_i l' *errore locale di roundoff del metodo di Eulero* nel punto t_i, ovvero l'errore di roundoff commesso nella valutazione di f e nell'esecuzione delle operazioni aritmetiche, si ha:

$$\tilde{u}(t_i) = \tilde{u}(t_{i-1}) + hf(t_{i-1}, \tilde{u}(t_{i-1})) + \rho_i \ . \tag{3.41}$$

L'errore globale di roundoff $R_i(h)$ risente pertanto di tutti gli errori locali di roundoff $(\rho_{i-1}, \rho_{i-2}, \ldots, \rho_2, \rho_1)$, commessi nei punti precedenti $(t_{i-1}, t_{i-2}, \ldots, t_2, t_1)$.
Per l'errore globale di roundoff del metodo di Eulero si ha:

t	$y(t)$	h	$\tilde{u}(t)$	errore
0.2	0.6703200	10^{-4}	0.6702936	0.265×10^{-4}
		10^{-5}	0.6703188	0.125×10^{-5}
		10^{-6}	0.6702805	0.395×10^{-4}
0.4	0.4493290	10^{-4}	0.4492932	0.357×10^{-4}
		10^{-5}	0.4493268	0.224×10^{-5}
		10^{-6}	0.4494673	0.138×10^{-3}
0.6	0.3011942	10^{-4}	0.3011584	0.358×10^{-4}
		10^{-5}	0.3011920	0.221×10^{-5}
		10^{-6}	0.3013079	0.114×10^{-3}
0.8	0.2018965	10^{-4}	0.2018644	0.322×10^{-4}
		10^{-5}	0.2018946	0.189×10^{-5}
		10^{-6}	0.2020078	0.111×10^{-3}
1.0	0.1353353	10^{-4}	0.1353082	0.271×10^{-4}
		10^{-5}	0.1353334	0.186×10^{-5}
		10^{-6}	0.1354239	0.886×10^{-4}

Tabella 3.4: Applicazione del metodo di Eulero al problema (3.39). Soluzione calcolata $\tilde{u}(t)$ ed errore in $t = 0.2, 0.4, 0.6, 0.8, 1$ per $h = 10^{-4}, h = 10^{-5}, h = 10^{-6}$.

Teorema 3.3.3. [Errore globale di roundoff del metodo di Eulero]
Si supponga che $f(t, y)$ soddisfi le ipotesi del Teorema 3.1.1 e sia $u(t_i)$ $(i = 1, 2, \ldots, n)$ la soluzione del problema discreto $M_h(P)$ (3.18). Per l'errore globale di roundoff $R_i(h)$ del metodo di Eulero, definito in (3.40), si ha:

$$|R_i(h)| \leq e^{L(T-t_0)}|R_0| + \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\frac{\rho}{h} \,, \tag{3.42}$$

dove R_0 indica l'errore di roundoff nel dato iniziale, $\rho = max_{i=1,2,\ldots,n}|\rho_i|$ e ρ_i è l'errore locale di roundoff nel punto t_i.

Dimostrazione Dalla (3.18) e dalla (3.41) segue

$$\begin{aligned} R_i(h) &= u(t_i) - \tilde{u}(t_i) = u(t_{i-1}) + hf(t_{i-1}, u(t_{i-1})) - [\tilde{u}(t_{i-1}) + hf(t_{i-1}, \tilde{u}(t_{i-1})) + \rho_i] = \\ &= u(t_{i-1}) - \tilde{u}(t_{i-1}) + h[f(t_{i-1}, u(t_{i-1})) - f(t_{i-1}, \tilde{u}(t_{i-1}))] - \rho_i \end{aligned} \,.$$

Passando ai valori assoluti e usando la (3.6), si ha:

$$|R_i(h)| \leq |u(t_{i-1}) - \tilde{u}(t_{i-1})| + hL|u(t_{i-1}) - \tilde{u}(t_{i-1})| + |\rho_i| = (1 + hL)|R_{i-1}(h)| + |\rho_i| \,,$$

da cui, posto $\rho = \max_{i=1,2,\ldots,n} |\rho_i|$:

$$|R_i(h)| \leq (1 + hL)|R_{i-1}(h)| + \rho \,.$$

Applicando il Lemma 3.3.2, con $K = hL$ e $H = \rho$, si ricava:

$$|R_i(h)| \leq e^{nhL}|R_0| + \frac{\left(e^{nhL} - 1\right)}{Lh}\rho \,.$$

Dalla (3.13) segue la tesi. ■

La stima (3.42) mostra che l'errore di roundoff cresce, al decrescere del passo h. Si introduce a questo punto la seguente definizione:

Definizione 3.3.9. *Si dice errore totale del metodo di Eulero, nel punto t_i, la quantità*

$$E_i^{tot}(h) = |y(t_i) - \tilde{u}(t_i)| \ . \tag{3.43}$$

$E_i^{tot}(h)$ esprime l'errore totale commesso nel calcolare la soluzione del problema (3.5) con il metodo di Eulero. Si osserva che

$$E_i^{tot}(h) = |y(t_i) - u(t_i) + u(t_i) - \tilde{u}(t_i)| = |E_i(h) + R_i(h)| \leq |E_i(h)| + |R_i(h)| \ . \tag{3.44}$$

Per l'errore totale del metodo di Eulero si ha:

Corollario 3.3.2. *Nelle ipotesi del Teorema 3.3.1 e del Teorema 3.3.3, per l'errore totale del metodo di Eulero, definito in (3.43), si ha:*

$$E_i^{tot}(h) \leq e^{L(T-t_0)}(|E_0| + |R_0|) + \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\left(\frac{C}{2}h + \frac{\rho}{h}\right). \tag{3.45}$$

Dimostrazione Sostituendo le stime (3.27) e (3.42) in (3.44), si ha la tesi. ■

In Figura 3.12 è mostrato l'andamento delle stime degli errori globali di roundoff, di troncamento, e dell'errore totale in funzione del passo di discretizzazione h. Dalla figura si evince che esiste un valore ottimale del passo, h_{opt}, che rende minima la stima (3.45). Per determinare tale valore è sufficiente minimizzare, in $]0, +\infty[$, la funzione di h:

$$\psi(h) = e^{L(T-t_0)}(|E_0| + |R_0|) + \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\left(\frac{C}{2}h + \frac{\rho}{h}\right) \ .$$

Dall'espressione

$$\psi'(h) = \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\left(\frac{C}{2} - \frac{\rho}{h^2}\right) = 0$$

si ricava che l'unico zero di ψ', in $]0, +\infty[$, è $h_* = \sqrt{\frac{2\rho}{C}}$. Inoltre si ha:

$$\psi''(h) = \frac{\left(e^{L(T-t_0)} - 1\right)}{L}\frac{2\rho}{h^3} > 0 \qquad\qquad (0 < h < +\infty),$$

per cui ψ è convessa in $]0, +\infty[$ ed h_* è minimo assoluto. Pertanto il valore di h_{opt} cercato è h_*. Si osserva che se $h < h_{opt}$, l'errore di roundoff diventa dominante rispetto all'errore

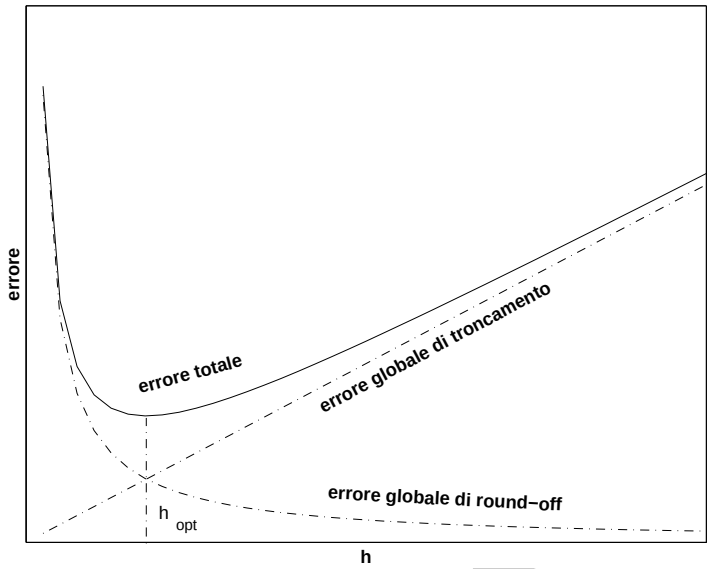

Figura 3.12: Stima dell'andamento dell'errore globale di troncamento, dell'errore globale di roundoff e dell'errore totale in funzione del passo di discretizzazione h.

globale di troncamento. Tuttavia il calcolo effettivo di h_{opt} richiede la conoscenza di C e di ρ, i quali non sono facilmente determinabili. In generale, la determinazione di h_{opt} non è affatto un problema di semplice risoluzione. Nella pratica sono realizzati algoritmi che scelgono dinamicamente h, passo dopo passo, tenendo conto in qualche modo dell'errore totale.

Si osservi infine che la stima suddetta è pessimistica in quanto in generale gli errori ρ_i non raggiungono il valore massimo ρ e possono compensarsi assumendo a volta valore positivo a volte valore negativo. Una stima più stringente si può ottenere con un'analisi statistica dell'errore (si vedano, ad esempio, [6, 7]).

3.4 Un metodo implicito: il metodo di Eulero all'indietro

Si faccia riferimento ancora al problema continuo $M(P)$ (3.5) e sia y la funzione incognita soluzione di tale problema. Considerato lo sviluppo in serie di Taylor ottenuto in (3.10), vi si ponga

$$x = t \qquad x_0 - x = h$$

e quindi $x_0 = t + h$. Si ottiene:

$$y(t) = y(t+h) - hy'(t+h) + \tau(t,h) = y(t+h) - hf(t+h, y(t+h)) + \tau(t,h)\,. \quad (3.46)$$

Da cui, trascurando il resto ed esplicitando la (3.46) rispetto ad $y(t+h)$, si ha:

$$y(t+h) \simeq y(t) + hf(t+h, y(t+h))\,. \quad (3.47)$$

Discretizzando l'intervallo $I = [t_0, T]$ come per il metodo di Eulero esplicito, utilizzando un passo di discretizzazione h, sostituendo t_i a t, t_{i+1} a $t + h$, indicando con y_i un approssimazione di y nel punto t_i, si ottiene la relazione:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) , \qquad i = 0, 1, \ldots, n-1 . \tag{3.48}$$

La (3.48), unitamente al valore noto $y(t_0) = y_0$, costituisce il *metodo di Eulero all'indietro* o, più brevemente, *metodo EI*. Il metodo EI è un metodo implicito ad un passo, pertanto, ad ogni passo, la sua applicazione a partire dal punto t_i, richiede la valutazione di f in y_{i+1} che non è noto. La (3.48) equivale al calcolo di un valore di y_{i+1} che sia soluzione dell'equazione:

$$\Phi(x) = hf(t_{i+1}, x) - x + y_i = 0 \tag{3.49}$$

L'equazione (3.49) è, in generale, non lineare[16] e le tecniche per "risolverla" sono molteplici, la preferenza di una rispetto ad un'altra dipende dal problema.
Scelta dunque una tecnica di risoluzione per (3.49), a partire dalla relazione (3.48) si ottiene immediatamente un algoritmo per la risoluzione numerica del problema (3.5) (Procedura 3.2).

[16] Tale equazione è lineare se e solo se lo è la funzione f rispetto alla variabile x.

```
procedure eulerоI(input: t0,T,y0,f,n, out: y)
  /# SCOPO: Risoluzione numerica di equazioni differenziali
  /#          ordinarie del primo ordine in forma esplicita
  /# SPECIFICHE PARAMETRI:
    var: t0  : reale              {estremo sinistro dell'intervallo}
    var: T   : reale              {estremo destro dell'intervallo}
    var: y0  : reale              {condizione iniziale}
    var: f   : funzione esterna   {funzione f(t,y)}
    var: n   : intero             {numero di punti di sottointervalli}
    var: y   : array di reali     {restituisce la soluzione calcolata.}
  /# INIZIO ISTRUZIONI:
       h := (T - t0)/n;
       y(0) := y0;
       for i = 0, n - 1 do
          t := t0 + (i + 1) * h
          call NonLinSolve(f,t,y(i),h,y(i + 1))   {Routine per la risoluzione}
                                                  {dell'equazione non lineare.}
                                                  {Richiede in input f, t, y(i), h}
                                                  {Restituisce in output y(i + 1)}
       endfor
  end procedure eulerоI
```

Procedura 3.2: Algoritmo per la risoluzione numerica del problema (3.5) con il metodo EI

Come per la Procedura 3.1, la *complessità di tempo* della Procedura 3.2 dipende essenzialmente dal numero di valutazioni della funzione $f(t, y)$. In questo caso però, il numero di valutazioni dipende oltre che dal numero di passi n, anche da quante valutazioni sono richieste, ad ogni passo, per la risoluzione numerica dell'equazione (3.49). Se indichiamo con m tale quantità, allora la procedura ha una complessità di tempo

$$T_{EI}(n) = \mathcal{O}(n \cdot m) \text{ valutazioni di funzione.}$$

La *complessità di spazio* è ancora

$$S_{EI}(n) = \mathcal{O}(n) \text{ variabili reali,}$$

in quanto l'algoritmo utilizza un array di reali, y, di dimensione $n+1$.

3.5 Analisi degli errori introdotti dal metodo di Eulero all'indietro

L'espressione (3.47), dalla quale si deriva il metodo EI, corrisponde alla sostituzione della derivata con il rapporto incrementale:

$$y'(t+h) \simeq \frac{y(t+h) - y(t)}{h}\,.$$

Sostanzialmente si sostituisce il problema continuo (3.5) con il problema discreto

$$M'_h(P) \equiv \begin{cases} \frac{u(t_{i+1})-u(t_i)}{h} = f(t_{i+1}, u(t_{i+1})), & i = 0, 1, \ldots, n-1 \\ u(t_0) = y(t_0) \end{cases} . \tag{3.50}$$

Inoltre la soluzione del problema discreto (3.50) è calcolata con un sistema aritmetico floating-point a precisione finita. Quindi la risoluzione numerica del problema continuo (3.5) con il metodo EI è, in analogia al metodo di Eulero esplicito, affetta da errori dovuti alle stesse cause fondamentali:

(1) la sostituzione del problema continuo $M(P)$ (3.5) con il problema discreto $M'_h(P)$ (3.50) (errore di discretizzazione);

(2) la risoluzione del problema discreto $M'_h(P)$ (3.50) in un sistema aritmetico floating-point a precisione finita (errore di roundoff) [17].

3.5.1 Il problema discreto come approssimazione del problema continuo

Come per il metodo di Eulero esplicito, l'errore $\tau(t_i, h)$, che si commette in un solo passo del metodo EI, è detto *errore locale di troncamento del metodo EI*. Analogamente sussistono le seguenti definizioni:

Definizione 3.5.1. *Si dice errore di discretizzazione del metodo EI la quantità:*

$$T(t,h) = \frac{\tau(t,h)}{h}\,, \tag{3.51}$$

cioè la differenza tra operatore differenziale del problema continuo (3.5) (derivata prima di $y(t)$) e operatore alle differenze del problema discreto (3.50) (rapporto incrementale).

[17] Nel computo totale degli errori si dovrebbe tener conto, oltre che di (1) e (2), anche dell'errore dovuto alla risoluzione dell'equazione (3.49). Tuttavia tale errore è assimilabile, ad ogni passo, all'*errore locale di troncamento* e pertanto non viene considerato.

Definizione 3.5.2. *Si dice errore globale di troncamento del metodo EI, nel punto t_i, la quantità*

$$E_i(h) = y(t_i) - u(t_i) \ , \tag{3.52}$$

cioè la differenza tra la soluzione del problema continuo $M(P)$ (3.5) e la soluzione del problema discreto $M_h(P)$ (3.50).

Consistenza

In perfetta analogia al §3.3.1, si ha:

Lemma 3.5.1. *Se $y(t) \in C^2([t_0, T])$ è la soluzione del problema continuo $M(P)$ (3.5), per l'errore di discretizzazione del metodo EI si ha:*

$$T(t, h) = \mathcal{O}(h) \, .$$

Dal Lemma 3.5.1 si evince che per ogni punto t fissato, al tendere del passo h a 0 l'errore di discretizzazione tende a 0 con ordine di infinitesimo uguale a 1. Pertanto si dice che *il metodo EI è consistente di ordine 1.*

Convergenza

Al fine di studiare l'errore globale di troncamento si premette il lemma:

Lemma 3.5.2. *Se h e L sono costanti positive tali che*

$$h \leq \frac{1}{2L} \ , \tag{3.53}$$

allora:

$$1 + 2hL \leq 2 \tag{3.54}$$

$$1 - hL > 0 \tag{3.55}$$

$$\tfrac{1}{1-hL} \leq 1 + 2hL \tag{3.56}$$

Dimostrazione Da (3.53) si ottiene

$$hL \leq \frac{1}{2} \ , \tag{3.57}$$

da cui, si ricava

$$1 + 2hL \leq 1 + 2\frac{1}{2} \ , \tag{3.58}$$

ovvero la (3.54). Da (3.57) si ha

$$1 - hL \geq 1 - \frac{1}{2} = \frac{1}{2} \ , \tag{3.59}$$

da cui segue la (3.55). Invertendo la (3.59) si ricava

$$\frac{1}{1 - hL} \leq 2 \ . \tag{3.60}$$

Quindi da (3.57) e (3.60):

$$\frac{hL}{1-hL} \leq \frac{1}{2}2 = 1 \; . \tag{3.61}$$

Applicando la (3.61) all'identità

$$\frac{1}{1-hL} = 1 + hL\left(1 + \frac{hL}{1-hL}\right)$$

si ha la (3.56). ■

Per l'errore globale di troncamento del metodo EI si ha:

Teorema 3.5.1. [Errore globale di troncamento del metodo EI]
Si supponga che $f(t,y)$ soddisfi le ipotesi del Teorema 3.1.1 e sia $y \in C^2([t_0,T])$ la soluzione del problema continuo $M(P)$ (3.5). Se per il passo di discretizzazione vale [18]

$$h \leq \frac{1}{2L} \, ,$$

con L definita da (3.6), per l'errore globale di troncamento $E_i(h)$ del metodo EI, definito in (3.52), si ha:

$$|E_i(h)| \leq e^{2L(T-t_0)}|E_0| + \frac{\left(e^{2L(T-t_0)}-1\right)}{L}\frac{C}{2}h \; , \tag{3.62}$$

dove E_0 indica l'errore nel dato iniziale e $C = max_{[t_0,T]}|y''|$.

Dimostrazione Dalla (3.46) e dalla (3.48), si ha:

$$E_i(h) = y(t_i) - u(t_i) = y(t_{i-1}) - u(t_{i-1}) + h\left[f(t_i, y(t_i)) - f(t_i, u(t_i))\right] + \tau(t_{i-1}, h) \, .$$

Passando ai valori assoluti e usando la (3.6) si ottiene:

$$\begin{aligned} |E_i(h)| &\leq |y(t_{i-1}) - u(t_{i-1})| + hL|y(t_i) - u(t_i)| + |\tau(t_{i-1}, h)| = \\ &= |E_{i-1}(h)| + hL|E_i(h)| + |\tau(t_{i-1}, h)| \, , \end{aligned}$$

Dalla continuità della derivata seconda si ha per l'errore locale di troncamento:

$$|\tau(t_{i-1}, h)| \leq \frac{C}{2}h^2$$

da cui si ha:

$$|E_i(h)|(1-hL) \leq |E_{i-1}(h)| + \frac{C}{2}h^2 \, .$$

Per la (3.55) si ha:

$$|E_i(h)| \leq \frac{1}{1-hL}|E_{i-1}(h)| + \frac{1}{1-hL}\frac{C}{2}h^2 \, .$$

[18] Analoghi risultati si possono ottenere sotto la condizione più generale:

$$h < \frac{1}{L}.$$

Inoltre dalla (3.56) segue:

$$|E_i(h)| \leq (1+2hL)|E_{i-1}(h)| + (1+2hL)\frac{C}{2}h^2 .$$

Applicando il Lemma 3.3.2, con $K = 2hL$ e $H = (1+2hL)\frac{C}{2}h^2$, si ricava:

$$|E_i(h)| \leq e^{2nhL}|E_0| + \frac{\left(e^{2nhL}-1\right)}{2hL}(1+2hL)\frac{C}{2}h^2 .$$

Ricordando la (3.13):

$$|E_i(h)| \leq e^{2L(T-t_0)}|E_0| + \frac{\left(e^{2L(T-t_0)}-1\right)}{L}(1+2hL)\frac{C}{4}h.$$

Dalla (3.54) segue infine la tesi. ■

Dal Teorema 3.5.1 si ricava inoltre:

Corollario 3.5.1. *Nelle ipotesi del Teorema 3.5.1, supposto $E_0 = 0$, per l'errore globale di troncamento del metodo EI si ha:*

$$E_i(h) = \mathcal{O}(h) . \tag{3.63}$$

Dimostrazione Ponendo $E_0 = 0$ nella (3.62), si ha la tesi. ■

Dal Corollario 3.5.1 si evince che se si fa tendere h a 0, $E_i(h)$ tende a zero con ordine di infinitesimo 1. Pertanto *il metodo EI è convergente di ordine 1*

3.5.2 La risoluzione del problema discreto

Come per il metodo di Eulero esplicito si analizzano stabilità, condizionamento ed effetti dell'errore di roundoff.

Stabilità

Al fine di indagare sulla stabilità del metodo EI, in analogia al §3.3.2, indichiamo con $u(t_i) = y_i$ $(i = 1, 2, \ldots)$ la soluzione del problema discreto (3.50), e con $v(t_i) = v_i$ $(i = 1, 2, \ldots)$ la soluzione del problema discreto perturbato

$$\begin{cases} \frac{v(t_{i+1})-v(t_i)}{h} = f(t_{i+1}, v(t_{i+1})), \quad i = 0, 1, \ldots \\ v(t_0) = y(t_0) + \epsilon_0 \end{cases} \tag{3.64}$$

dove ϵ_0 è la perturbazione sul dato iniziale $y(t_0)$. Siamo interessati allo studio dell'errore di propagazione

$$\epsilon_i(h) = v(t_i) - u(t_i) = v_i - y_i .$$

Analogamente al metodo di Eulero esplicito, si pongono due problemi:

(a') Sia $t \in [t_0, \infty[$, posto $t = t_i = t_0 + ih$, come si comporta l'errore di propagazione $\epsilon_i(h)$ per h fissato e per $i \to \infty$ (cioè per $t \to \infty$) ?
(Studio della *stabilità assoluta*)

(b') Sia $t \in [t_0, T]$, posto $t = t_i = t_0 + ih$, come si comporta l'errore di propagazione $\epsilon_i(h)$ per t fissato (e quindi ih fissato) e per $h \to 0$ (cioè $i \to \infty$) ?
(Studio della *zero-stabilità*)

(a') Stabilità assoluta

Analizziamo come si propaga $\epsilon_i(h)$ nell'intervallo $[t_0, \infty[$, nel caso in cui il metodo EI è applicato al problema test (3.30). Si ha:

Proposizione 3.5.1. *Il metodo di EI è assolutamente stabile se e soltanto se*

$$\lambda \leq 0 \quad \text{oppure} \quad (\lambda > 0, \;\; h \geq 2/\lambda) \;\; .$$

Dimostrazione Essendo $f(t, y) = \lambda y$, i problemi discreti (3.50) e (3.64) assumono la forma equivalente

$$\begin{cases} y_i = y_{i-1} + h\lambda y_i, & i = 1, 2, 3, \dots \\ y_0 = y(t_0) \end{cases} \tag{3.65}$$

e

$$\begin{cases} v_i = v_{i-1} + h\lambda v_i, & i = 1, 2, 3, \dots \\ v_0 = y(t_0) + \epsilon_0 \end{cases} .$$

Quindi per l'errore di propagazione $\epsilon_i(h)$ si ha:

$$\epsilon_i(h) = v_i - y_i = (v_{i-1} - y_{i-1}) + h\lambda(v_i - y_i) = \epsilon_{i-1}(h) + h\lambda\epsilon_i(h) \, ,$$

da cui, supposto $1 - h\lambda \neq 0$,

$$\epsilon_i(h) = \epsilon_{i-1}(h)\frac{1}{1 - h\lambda} \, . \tag{3.66}$$

Applicando ripetutamente la (3.66), si ricava:

$$\epsilon_i(h) = \epsilon_0 \left(\frac{1}{1 - h\lambda}\right)^i \, . \tag{3.67}$$

Dalla (3.67) si deduce:

$$\lim_{i \to \infty} |\epsilon_i(h)| = \begin{cases} 0 & \text{se} \quad |1 - h\lambda| > 1 \\ |\epsilon_0| & \text{se} \quad |1 - h\lambda| = 1 \\ +\infty & \text{se} \;\; 0 < |1 - h\lambda| < 1 \end{cases} .$$

Quindi l'errore di propagazione non si amplifica e il metodo EI è assolutamente stabile, se e solo se

$$|1 - h\lambda| \geq 1 \implies h\lambda \leq 0 \quad \text{oppure} \quad h\lambda \geq 2 \tag{3.68}$$

Dalla (3.68) segue la tesi. ■

Si osserva che se $\lambda < 0$ il metodo EI è assolutamente stabile comunque si scelga h. Se invece $\lambda > 0$, affinché ci sia assoluta stabilità, è necessario che $h > 2/\lambda$, cioè non si può scegliere h piccolo quanto si vuole [19]. In questo caso, quindi il metodo puó non raggiungere l'accuratezza desiderata.

[19] Si ricorda che per $\lambda > 0$ il metodo di Eulero esplicito non è assolutamente stabile per alcun valore di h.

(b') Zero-stabilità

Analizziamo come si comporta l'errore di propagazione nel caso in cui si risolvono i problemi discreti (3.50) e (3.64) nell'intervallo finito $[t_0, T]$. Si ha:

Proposizione 3.5.2. *Il metodo di Eulero all'indietro è zero-stabile se*

$$h \leq h_0 = 1/(2L) \ .$$

Dimostrazione Ricordando che $u(t_i) = y_i$ e $v(t_i) = v_i$, si ha:

$$\begin{aligned} \epsilon_i(h) &= v_{i-1} + hf(t_i, v_i) - y_{i-1} - hf(t_i, y_i) = \\ &= v_{i-1} - y_{i-1} + h(f(t_i, v_i) - f(t_i, y_i)) \ . \end{aligned}$$

Passando ai valori assoluti e usando la (3.6) si ottiene:

$$\begin{aligned} |\epsilon_i(h)| &\leq |y_{i-1} - v_{i-1}| + hL|v_i - y_i| = \\ &= |\epsilon_{i-1}(h)| + hL|\epsilon_i(h)| \ , \end{aligned}$$

da cui si ha

$$|\epsilon_i(h)|(1 - hL) \leq |\epsilon_{i-1}(h)|$$

Essendo $h \leq \frac{1}{2L}$, da (3.55) e (3.56) segue:

$$|\epsilon_i(h)| \leq |\epsilon_{i-i}(h)|(1 + 2hL)$$

Applicando il Lemma 3.3.2, con $K = 2hL$ e $H = 0$, si ottiene:

$$|\epsilon_i(h)| \leq e^{2nhL}|\epsilon_0|,$$

e ricordando la (3.13):

$$|\epsilon_i(h)| \leq e^{2(T-t_0)L}|\epsilon_0|. \tag{3.69}$$

Dalla (3.69) si deduce che esistono $M = e^{2(T-t_0)L} > 0$ e $h_0 = 1/(2L)$ tali che

$$|\epsilon_i(h)| \leq M|\epsilon_0|, \ \text{ per } h \leq h_0 \ \text{ e } \ i = 1, 2, \ldots, n, \ \text{ con } \ n = \frac{T - t_0}{h} \ . \tag{3.70}$$

Dalla (3.70) si ha che se ϵ_0 tende a zero anche $\epsilon_i(h)$ tende a zero, da cui se $|\epsilon_0|$ è "sufficientemente piccolo", $|\epsilon_i(h)|$ si può mantenere al di sotto di un valore fissato ovvero si ha la zero-stabilità. ■

Si ricorda che la stima ottenuta per l'errore globale di troncamento richiede la stessa condizione sul passo h. Questa analogia è in accordo con il Teorema 3.3.2. Infatti essendo il metodo EI consistente, esso è zero-stabile quando e solo quando risulta convergente (cioè per $h \leq 1/(2L)$).

Condizionamento del problema discreto

Analizziamo ora come si comporta l'indice di condizionamento nel metodo EI. In analogia al §3.3.2, si ha:

Proposizione 3.5.3. *Il metodo EI è ben condizionato se e solo se*

$$\lambda \leq 0 \qquad \text{oppure} \quad (\lambda > 0, \;\; h \geq 2/\lambda) \;\; .$$

Dimostrazione Indicata con F_i la funzione che al dato iniziale y_0 associa il valore della soluzione del problema discreto (3.65), ottenuto applicando il metodo EI al problema test (3.30), si ha:

$$F_i(y_0) = y_i = \left(\frac{1}{1-h\lambda}\right)^i y_0 \;\; ,$$

quindi l'indice di condizionamento del problema (3.65), nel punto t_i, è dato da

$$\mu_i(h,i,\lambda) = |F_i'(y_0)| = \left|\frac{1}{1-h\lambda}\right|^i . \tag{3.71}$$

Dalla (3.71) si deduce che il metodo EI è ben condizionato se e solo se

$$|\mu_i(h,i,\lambda)| \leq 1 \Longleftrightarrow |1-h\lambda|^i \geq 1 \Longleftrightarrow |1-h\lambda| \geq 1 \Longleftrightarrow h\lambda \leq 0 \text{ oppure } h\lambda \geq 2. \tag{3.72}$$

Dalla (3.72) segue la tesi. ■

Dalla Proposizione 3.5.3 si evince che il problema discreto (3.65) è ben condizionato se $h\lambda \leq 0$ oppure $h\lambda \geq 2$ e mal condizionato altrimenti (in particolare per $h\lambda \simeq 1$). Si osservi infine che $\mu_i(h,i,\lambda)$ coincide con il fattore di amplificazione dell'errore di propagazione ottenuto in (3.67). Quindi, analogamente al metodo di Eulero esplicito, lo studio del condizionamento del problema discreto (3.50) può essere ricondotto a quello dell'assoluta stabilità.

Effetti dell'errore di roundoff

Si vogliono analizzare gli effetti dell'errore di roundoff nell'applicazione del metodo di Eulero all'indietro in un sistema aritmetico floating-point a precisione finita.
In analogia al §3.3.2, indichiamo con $\tilde{u}(t_i)$ il valore calcolato[20], in un sistema aritmetico floating-point a precisione finita, della soluzione $u(t_i)$ del problema discreto (3.50) nel punto t_i. Sussiste la seguente:

Definizione 3.5.3. *Si dice errore globale di roundoff del metodo EI, nel punto t_i, la quantità*

$$R_i(h) = u(t_i) - \tilde{u}(t_i) \tag{3.73}$$

[20]In tale analisi si assume per semplicità di non commettere errori (di troncamento analitico) nella risoluzione dell'equazione non lineare (3.49).

Indicando con ρ_i l'*errore locale di roundoff del metodo EI* nel punto t_i, ovvero l'errore di roundoff commesso, applicando un solo passo del metodo EI, nella valutazione di f e nell'esecuzione delle operazioni aritmetiche, si ha:

$$\tilde{u}(t_i) = \tilde{u}(t_{i-1}) + hf(t_i, \tilde{u}(t_i)) + \rho_i \ . \tag{3.74}$$

Per l'errore globale di roundoff del metodo EI si ha:

Teorema 3.5.2. [Errore globale di roundoff del metodo EI]
Si supponga che $f(t, y)$ soddisfi le ipotesi del Teorema 3.1.1 e sia $u(t_i)$ $(i = 1, 2, \ldots, n)$ la soluzione del problema discreto $M'_h(P)$ (3.50). Se per il passo di discretizzazione vale

$$h \leq \frac{1}{2L} \ ,$$

con L definita da (3.6), per l'errore globale di roundoff $R_i(h)$ del metodo EI, definito in (3.73), si ha:

$$|R_i(h)| \leq e^{2L(T-t_0)}|R_0| + \frac{\left(e^{2L(T-t_0)} - 1\right)}{L}\frac{\rho}{h} \ , \tag{3.75}$$

dove R_0 indica l'errore di roundoff nel dato iniziale, $\rho = max_{i=1,2,\ldots,n}|\rho_i|$ e ρ_i è l'errore locale di roundoff nel punto t_i.

Dimostrazione Dalla (3.50) e dalla (3.74) segue

$$\begin{aligned} R_i(h) &= u(t_i) - \tilde{u}(t_i) = u(t_{i-1}) + hf(t_i, u(t_i)) - [\tilde{u}(t_{i-1}) + hf(t_i, \tilde{u}(t_i)) + \rho_i] = \\ &= u(t_{i-1}) - \tilde{u}(t_{i-1}) + h[f(t_i, u(t_i)) - f(t_i, \tilde{u}(t_i))] - \rho_i \, . \end{aligned}$$

Passando ai valori assoluti e usando la (3.6), si ha:

$$|R_i(h)| \leq |u(t_{i-1}) - \tilde{u}(t_{i-1})| + hL|u(t_i) - \tilde{u}(t_i)| + |\rho_i| = |R_{i-1}(h)| + hL|R_i(h)| + |\rho_i| \, ,$$

da cui, posto $\rho = \max_{i=1,2,\ldots,n} |\rho_i|$:

$$|R_i(h)|(1 - hL) \leq |R_{i-1}(h)| + \rho \, .$$

Per la (3.55) si ha:

$$|R_i(h)| \leq \frac{1}{1 - hL}|R_{i-1}(h)| + \frac{1}{1 - hL}\rho \, .$$

Inoltre dalla (3.56) segue:

$$|R_i(h)| \leq (1 + 2hL)|R_{i-1}(h)| + (1 + 2hL)\rho \, .$$

Applicando il Lemma 3.3.2, con $K = 2hL$ e $H = (1 + 2hL)\rho$, si ricava:

$$|R_i(h)| \leq e^{2nhL}|R_0| + \frac{\left(e^{2nhL} - 1\right)}{2hL}(1 + 2hL)\rho \ .$$

Ricordando la (3.13):

$$|R_i(h)| \leq e^{2L(T-t_0)}|R_0| + \frac{\left(e^{2L(T-t_0)} - 1\right)}{L}(1 + 2hL)\frac{\rho}{2h}.$$

Dalla (3.54) segue infine la tesi. ■

La stima (3.75) mostra che l'errore di roundoff cresce, al decrescere del passo h.

Si introduce a questo punto la seguente definizione:

Definizione 3.5.4. *Si dice errore totale del metodo di Eulero, nel punto t_i, la quantità*

$$E_i^{tot}(h) = |y(t_i) - \tilde{u}(t_i)| \ . \tag{3.76}$$

$E_i^{tot}(h)$ esprime l'errore totale commesso nel calcolare la soluzione del problema (3.5) con il metodo EI. Si osserva che

$$E_i^{tot}(h) = |y(t_i) - u(t_i) + u(t_i) - \tilde{u}(t_i)| = |E_i(h) + R_i(h)| \leq |E_i(h)| + |R_i(h)| \ . \tag{3.77}$$

Per l'errore totale del metodo EI si ha:

Corollario 3.5.2. *Nelle ipotesi del Teorema 3.5.1 e del Teorema 3.5.2, per l'errore totale del metodo EI, definito in (3.76), si ha:*

$$E_i^{tot}(h) \leq e^{2L(T-t_0)}(|E_0| + |R_0|) + \frac{\left(e^{2L(T-t_0)} - 1\right)}{L}\left(\frac{C}{2}h + \frac{\rho}{h}\right). \tag{3.78}$$

Dimostrazione Sostituendo le stime (3.62) e (3.75) in (3.77) , si ha la tesi. ■

La stima (3.78), valida per $h \leq 1/(2L)$, è analoga alla (3.45). Pertanto, come per il metodo di Eulero esplicito, è possibile determinare un valore ottimale del passo di discretizzazione, h_{opt}, che minimizza nell'intervallo $]0, 1/(2L)]$, la seguente funzione

$$\psi(h) = e^{2L(T-t_0)}(|E_0| + |R_0|) + \frac{\left(e^{2L(T-t_0)} - 1\right)}{L}\left(\frac{C}{2}h + \frac{\rho}{h}\right).$$

In perfetta analogia al §3.3.2 si ricava che l'unico zero di ψ', in $]0, +\infty[$, è $h_* = \sqrt{\frac{2\rho}{C}}$. Quindi, se $h_* \leq 1/(2L)$, allora $h_{opt} = h_*$. Altrimenti, se $h_* > 1/(2L)$, essendo ψ decrescente in $]0, 1/(2L)]$, il minimo assoluto è $h_{opt} = 1/(2L)$.

In ogni caso, indipendentemente dal valore di h_{opt}, la stima (3.78) conduce alla stessa conclusione raggiunta per il metodo di Eulero esplicito. Cioè non è possibile scegliere h piccolo in modo arbitrario, in quanto per $h < h_{opt}$ si ha una corrispondente crescita della stima dell'errore totale.

3.6 Software matematico disponibile per ODE

Descriveremo il software matematico disponibile per la risoluzione di ODE classificandolo secondo la convenzione introdotta da *J. Rice* in *Numerical methods, Software and Analysis*, Academic Press, 1993 (cfr. **Parte prima, Cap. 3**). Parliamo, dunque, di:

1. routine individuali, o eventualmente raccolte in collezioni;
2. packages specializzati;
3. librerie di software *general - purpouse*;
4. Ambienti di risoluzione di problemi (PSE).

Analizziamo, brevemente, le singole classi.

1. Tra le routine individuali citiamo quelle dell'ACM-TOMS [13], come la routine `GERK`, per la risoluzione di sistemi non lineari di equazioni differenziali ordinarie, a valori inziali, con metodi Runge Kutta del quarto e quinto ordine. È disponibile, inoltre il software `ODESSA`, e il codice `rksuite-90`, scritto in Fortran 90, per problemi a valori iniziali.

2. Sul sito di Netlib sono disponibili packages specializzati, finalizzati alla risoluzione di ODE, tra cui `ODE` [11] e `ODEPACK` [12]. `ODEPACK` consiste di 9 solutori per problemi a valori iniziali, sia di tipo stiff che non. I solutori si differenziano in base alla forma del sistema di ODE. Precisamente, 6 solutori risolvono sistemi di ODE in forma esplicita, di questi, 4 usano metodi diretti per la risoluzione del sistema lineare e 2 usano invece metodi iterativi basati su sottospazi di Krylov. Gli alltri 3 solutori, risolvono sistemi lineari di ODE in forma implicita.

3. Tra le liberie *general - purpose*, citiamo quella del NAg [10] e la IMSL [14]. La libreria NAg del *Numerical Algorithm Group* di Oxford è una fonte molto ampia di software matematico. Per buona parte delle routine esistono versioni in FORTRAN e C, in singola e doppia precisione, nonché una versione parallela (in Fortran 77). Il Capitolo D02 della NAg's Fortran 90 Library, è dedicato alla risoluzione numerica di problemi a valori iniziali e problemi al contorno per equazioni differenziali ordinarie.
   La libreria dell'IMSL (*International Mathematics and Statistics Library*) è una collezione di software matematico, di tipo commerciale. Il Capitolo 5 contiene routine per la risoluzione di problemi a valori iniziali per ODE, problemi al contorno, sistemi differenziali-algebrici.

4. Tra gli ambienti di risoluzione di problemi (PSE) il più utilizzato è `MATLAB` [8]. Nel §3.7 si riportano alcuni esempi risolti attraverso l'uso di funzioni `MATLAB` .

3.7 MATLAB e le ODE

Il MATLAB possiede svariate funzioni per la risoluzione numerica delle equazioni differenziali ordinarie, la maggior parte delle quali si basa su raffinamenti e/o varianti del metodo di Eulero, descritto nel §3.2.
Tutte le funzioni per **ODE** di MATLAB fanno parte del *toolbox:* **funfun**[21] e sono di tipo adattativo, nel senso che il passo di discretizzazione h e, dunque, il numero di passi n, sono calcolati in modo da assicurare che l'errore locale sia dell'ordine di una tolleranza prefissata.
Tra le funzioni disponibili le piú note sono `ode45` e `ode23`, basate su metodi iterativi espliciti ad un passo. Per comprenderne l'uso e la sintassi consideriamo alcuni esempi.

♣ **Esempio 3.12.** Si risolva numericamente il seguente problema di Cauchy:

$$\begin{cases} y'(t) = -y - 5e^{-t} sin(5t) \quad t \in [0,5] \\ y(0) = 1 \end{cases} \tag{3.79}$$

la cui soluzione effettiva è data dalla funzione: $y(t) = e^t cos(5t)$.
Inanzitutto è necessario creare il file **funz.m** che contiene le operazioni che definiscono la funzione in (3.79):

```
function f=funz(t,y)
        f=-y-5*exp(-t).*sin(5*t);
```

Quindi, bisogna digitare i seguenti comandi:

```
>> t0=0;
>> T=5;
>> y0=1;
>> [t,y]=ode45('funz',[t0 T],y0);
```

La soluzione numerica del problema può essere confrontata graficamente (Figura 3.13) con la soluzione effettiva come segue:

```
>> exa=inline('exp(-t).*cos(5*t)')
exa=
    Inline function:
    exa(t)=exp(-t).*cos(5*t)
>> ye=exa(t);
>> plot(t,ye,'b')
>> hold on
>> plot(t,y,'r*')
```

♣

[21] In questo paragrafo si fa riferimento alla versione **6** di MATLAB

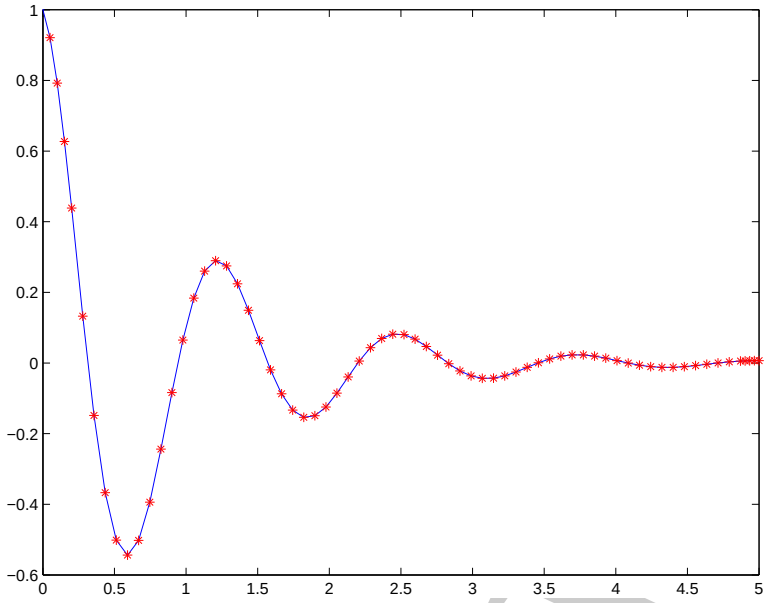

Figura 3.13: **Esempio 3.12:** Grafico di $y(t) = e^t cos(5t)$. Con il simbolo '*' si indicano i valori della soluzione numerica calcolata con `ode45`.

In generale, la funzione `ode45` determina un'approssimazione numerica dei valori della funzione $y(t)$ in modo tale che, ad ogni passo i, per $i = 1, n$, l'errore soddisfi la relazione:

$$e_i \leq max\left(RelTol \cdot |y(t_i)|, AbsTol\right)$$

dove: $RelTol = 10^{-3}$ e $AbsTol = 10^{-6}$, sono rispettivamente i valori di *default*, relativi alla tolleranza relativa ed alla tolleranza assoluta.
È possibile modificare tali valori mediante la funzione specifica `odeset`, descritta nell'esempio seguente.

♣ **Esempio 3.13.** Si risolva numericamente il problema dell'esempio 3.12, con tolleranza relativa $RelTol = 10^{-2}$ e tolleranza assoluta $AbsTol = 10^{-3}$.
Per risolvere numericamente il problema (3.79) con le tolleranze assegnate è necessario digitare i seguenti comandi:

```
>> options=odeset('AbsTol',1e-3,'RelTol',1e-2)
>> t0=0;
>> T=5;
>> y0=1;
```

```
>> [t,y]=ode45('funz',[t0 T],y0,options);
```

i	t_i	$y(t_i)$	i	t_i	$y(t_i)$	i	t_i	$y(t_i)$
0	0	1.0000	15	1.6716	-0.0901	30	3.5430	0.0122
1	0.0796	0.8505	16	1.7930	-0.1492	31	3.6680	0.0223
2	0.1592	0.5964	17	1.9180	-0.1440	32	3.7930	0.0224
3	0.2389	0.2902	18	2.0430	-0.0913	33	3.9180	0.0146
4	0.3185	-0.0157	19	2.1680	-0.0187	34	4.0430	0.0036
5	0.4435	-0.3846	20	2.2930	0.0456	35	4.1680	-0.0062
6	0.5685	-0.5416	21	2.4180	0.0787	36	4.2930	-0.0118
7	0.6935	-0.4767	22	2.5430	0.0778	37	4.4180	-0.0119
8	0.8185	-0.2565	23	2.6680	0.0502	38	4.5430	-0.0080
9	0.9407	-0.0033	24	2.7930	0.0105	39	4.6680	-0.0021
10	1.0629	0.1957	25	2.9180	-0.0235	40	4.7930	0.0032
11	1.1851	0.2866	26	3.0430	-0.0421	41	4.8448	0.0048
12	1.3072	0.2619	27	3.1680	-0.0419	42	4.8965	0.0059
13	1.4287	0.1552	28	3.2930	-0.0270	43	4.9483	0.0066
14	1.5501	0.0218	29	3.4180	-0.0061	44	5.0000	0.0067

Tabella 3.5: Esempio 3.13: soluzione numerica con `ode45`**;** $RelTol = 10^{-2}$**,** $AbsTol = 10^{-3}$**.**

♣

Tuttavia, a parità di tolleranza richiesta, in particolare se non si desidera un'accuratezza troppo elevata, la funzione `ode23` ha un costo computazionale minore, rispetto a `ode45` e, dunque, risulta più efficiente.
L'esempio seguente ne mostra un'applicazione.

♣ **Esempio 3.14.** Il problema dell'esempio 3.12 può essere risolto utilizzando la funzione `ode23`, con i seguenti comandi:

```
>> options=odeset('AbsTol',1e-3,'RelTol',1e-2)
>> t0=0;
>> T=5;
>> y0=1;
>> [t,y]=ode23('funz',[t0 T],y0,options);
```

Confrontando i risultati, riportati in tabella 3.6, con quelli in tabella 3.5 si osserva che, per il problema (3.79), con $RelTol = 10^{-2}$ e $AbsTol = 10^{-3}$, la funzione `ode23` risulta essere più efficiente.
Le soluzioni numeriche sono confrontate in Figura 3.14.

♣

i	t_i	$y(t_i)$	i	t_i	$y(t_i)$	i	t_i	$y(t_i)$
0	0	1.0000	11	1.3137	0.2580	22	2.8123	0.0045
1	0.1341	0.6851	12	1.4549	0.1278	23	2.9554	-0.0310
2	0.2682	0.1744	13	1.5820	-0.0115	24	3.1300	-0.0436
3	0.3600	-0.1584	14	1.6872	-0.1016	25	3.3075	-0.0246
4	0.4518	-0.4042	15	1.8044	-0.1513	26	3.4850	0.0045
5	0.5841	-0.5437	16	1.9442	-0.1367	27	3.6663	0.0222
6	0.7459	-0.3942	17	2.0840	-0.0676	28	3.9331	0.0134
7	0.8655	-0.1577	18	2.2107	0.0065	29	4.1672	-0.0063
8	0.9605	0.0348	19	2.3303	0.0594	30	4.3980	-0.0123
9	1.0556	0.1867	20	2.4772	0.0826	31	4.6898	-0.0010
10	1.1724	0.2827	21	2.6448	0.0562	32	5.0000	0.0066

Tabella 3.6: Esempio 3.14: soluzione numerica con `ode23`; $RelTol = 10^{-2}$**,** $AbsTol = 10^{-3}$**.**

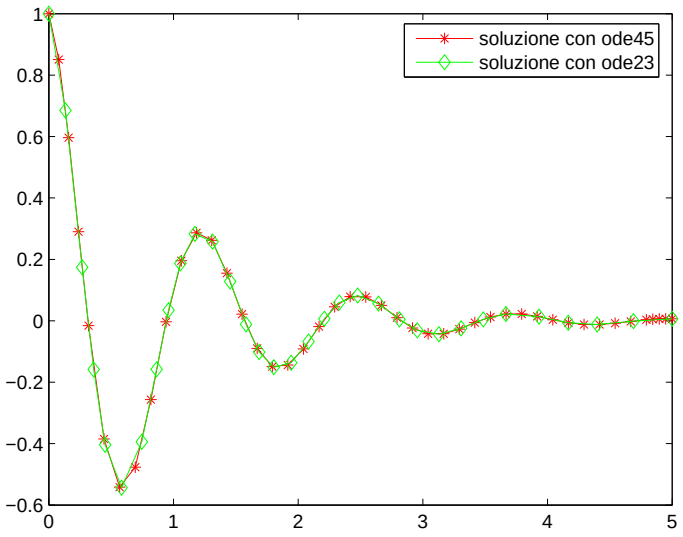

Figura 3.14: Soluzioni numeriche ottenute mediante le funzioni `ode45` e `ode23`. Con il tratto '$-*$' si descrive la soluzione numerica calcolata con `ode45`, con il tratto '$-\diamond$' si descrive la soluzione numerica calcolata con `ode23`.

3.8 Esercizi

3.8.1 Esercizi numerici

Esercizio 1 Si vuole utilizzare il metodo di Eulero per risolvere il problema a valori iniziali:

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0 \end{cases}$$

Quale delle seguenti affermazioni è vera? Motivare le risposte.

(a) L'errore locale di troncamento dipende da h e da t ma non dalla funzione f.

(b) Se dimezziamo il valore del passo di discretizzazione, anche il valore dell'errore locale di troncamento sarà circa la metà del precedente.

(c) L'errore di discretizzazione dipende da h, da t e dalla funzione f.

(d) Il metodo di Eulero è consistente di ordine 2 in quanto per l'errore locale di troncamento si ha $\tau(t, h) = \mathcal{O}(h^2)$ per $h \to 0$.

(e) Il metodo di Eulero è consistente ma non convergente.

(f) Se $f(t, y) = 3t$ l'errore locale di troncamento e l'errore globale di troncamento sono nulli.

(g) Considerato il problema test

$$\begin{cases} y'(t) = -3y(t) & t \in [t_0, +\infty[\\ y(t_0) = y_0 \end{cases},$$

il metodo di Eulero è assolutamente stabile per $0 \leq h \leq 0.7$.

(h) Il metodo di Eulero è assolutamente stabile per qualunque h se $\lambda > 0$.

(i) Il metodo di Eulero è zero-stabile, indipendentemente da h e da f.

(l) Un metodo numerico ad un passo, consistente ma non convergente, puó essere zero-stabile.

(m) L'errore locale di roundoff ρ_i dipende dal passo di discretizzazione h e dalla precisione del sistema aritmetico floating-point utilizzato.

(n) L'errore globale di roundoff $R_i(h)$ dipende dal passo di discretizzazione h, ma non dalla precisione del sistema aritmetico floating-point utilizzato.

(o) L'errore globale di roundoff $R_i(h)$ cresce al crescere del passo di discretizzazione h e della precisione del sistema aritmetico floating-point utilizzato.

(p) L'errore totale del metodo di Eulero $E_i^{tot}(h)$ puó essere stimato attraverso una funzione convessa $\psi(h)$ del passo di discretizzazione. Il punto di minimo di tale funzione dipende dalla precisione del sistema aritmetico floating-point utilizzato ma non dalla funzione f.

Esercizio 2 Supponendo $E_0 = 0$, ottenere una stima dell'errore globale di troncamento $E_i(h)$, in funzione di h, per i seguenti problemi a valori iniziali:

$$(a) \quad \begin{cases} y' = y + t & t \in [0,1] \\ y(0) = 0 \end{cases} \quad \text{(soluzione } y = e^t - t - 1)$$

$$(b) \quad \begin{cases} y' = y - t & t \in [0,1] \\ y(0) = 2 \end{cases} \quad \text{(soluzione } y = e^t + t + 1)$$

$$(c) \quad \begin{cases} y' = t - y & t \in [0,1] \\ y(0) = 0 \end{cases} \quad \text{(soluzione } y = e^{-t} + t - 1)$$

(*Suggerimento: Determinare i valori delle costanti C ed L, a partire dalla soluzione y e dalla funzione f.*)

Esercizio 3 Trovare una formula per il passo di discretizzazione h, necessario per garantire

$$E_i(h) \leq \epsilon$$

che coinvolga L, t_0, T e ϵ.

Esercizio 4 Utilizzare la formula precedente, per ognuno dei problemi a valori iniziali dell'Esercizio 2, in modo da stabilire il massimo valore di h per cui risulta

$$E_i(h) \leq 10^{-3} \ .$$

Esercizio 5 Supponendo $R_0 = 0$ e di eseguire i calcoli in un sistema aritmetico floating-point a precisione finita, caratterizzato da una massima accuratezza relativa pari a $u = 10^{-8}$, ottenere una stima dell'errore globale di roundoff $R_i(h)$, in funzione di h, per i problemi a valori iniziali dell'Esercizio 2. (*Suggerimento: porre $\rho = u$.*)

Esercizio 6 Utilizzando i risultati degli Esercizi 2 e 5, fornire, per ognuno dei problemi a valori iniziali (a),(b),(c), una stima dell'errore totale del metodo di Eulero in funzione del passo di discretizzazione h. Determinare inoltre il valore h_{opt} che rende minima tale stima.

Esercizio 7 Ripetere ognuno degli Esercizi 1-6 considerando, il luogo del metodo di Eulero, il metodo di Eulero all'indietro.

3.8.2 Problemi da risolvere con il calcolatore

Problema 1 Scrivere una procedura che implementi il metodo di Eulero. Tale procedura deve ricevere in input:

1. gli estremi dell'intervallo di ricerca t_0, T;
2. la condizione iniziale y_0;
3. la funzione esterna f;

4. e il numero di sottointervalli n (oppure il passo di discretizzazione h);

e fornire in output

5. un array di reali $\mathbf{y}=(y(1),\ldots,y(n))$ contenente la soluzione calcolata.

Problema 2 Testare la procedura implementata per risolvere i seguenti problemi di Cauchy:

$$(a)\quad \begin{cases} y' = te^{3t} - 2y \quad t \in [0,1] \\ y(0) = 0 \end{cases} \qquad \text{con} \qquad h = 0.05$$

$$(b)\quad \begin{cases} y' = 1 + (t-y)^2 \quad t \in [2,3] \\ y(2) = 1 \end{cases} \qquad \text{con} \qquad h = 0.1$$

$$(c)\quad \begin{cases} y' = 1 + \frac{y}{t} \quad t \in [1,2] \\ y(1) = 2 \end{cases} \qquad \text{con} \qquad h = 0.01$$

$$(d)\quad \begin{cases} y' = cos(2t) + sin(3t) \quad t \in [0,1] \\ y(0) = 1 \end{cases} \qquad \text{con} \qquad h = 0.05$$

Problema 3 Utilizzare la procedura implementata per risolvere il seguente problema di Cauchy:

$$\begin{cases} y'(t) = -5y(t) \quad t_0 = 0, T = 1 \\ y(0) = 1 \end{cases} .$$

(a) Calcolare la soluzione approssimata $\tilde{u}(t_n)$ in $t_n = 1$, per $h = 10^{-2}$ (n=100).

(b) Osservato che la soluzione esatta è

$$y(t_n) = y(1) = e^{-5} ,$$

determinare il valore dell'errore totale in $t_n = 1$.

Ripetere i punti (a) e (b) cambiano h in 10^{-3} e 10^{-4} (n=1000,10000) e commentare i risultati.

Problema 4 Scrivere una procedura che implementi il metodo di Eulero all'indietro. Tale procedura deve ricevere in input:

1. gli estremi dell'intervallo di ricerca t_0, T;
2. la condizione iniziale y_0;
3. la funzione esterna f;
4. e il numero di sottointervalli n (oppure il passo di discretizzazione h);

deve fornire in output:

5. un array di reali $\mathbf{y}=(y(1),\ldots,y(n))$ contenente la soluzione calcolata.

e deve utilizzare, ad ogni passo, per la risoluzione dell'equazione non lineare:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) \,, \qquad i = 0, 1, \ldots, n-1 \,.$$

una routine che sia basata sul metodo di iterazione funzionale: a tal scopo si puó utilizzare come punto iniziale, il valore $y_{i+1}^{(0)}$ ottenuto con il metodo di Eulero:

$$y_{i+1}^{(0)} = y_i + hf(t_i, y_i)$$

e porre poi:

$$y_{i+1}^{(k+1)} = y_i + hf(t_{i+1}, y_{i+1}^{(k)}) \qquad k = 0, 1, \ldots$$

Problema 5 Testare la procedura implementata, eseguendo ad ogni passo solo $k = 3$ passi del metodo di iterazione funzionale, per risolvere i problemi di Cauchy del Problema 2. Confrontare i risultati con quelli ottenuti attraverso il metodo di Eulero.

Problema 6 Si calcoli la soluzione per $t = 1$, del problema di Cauchy:

$$\begin{cases} y' = -y \\ y(0) = 1 \end{cases}$$

Utilizzando le procedure implementate per il metodo di Eulero in avanti e per il metodo di Eulero all'indietro. Utilizzare i seguenti valori per il passo di discretizzazione: $h = 0.1, 0.01, 0.001$. Utilizzare $k = 5$ iterazioni del metodo di iterazione funzionale per la risoluzione, nel metodo di Eulero all'indietro, dell'equazione non lineare.

Problema 7 Si consideri il problema di Cauchy:

$$\begin{cases} y' = -te^{-y} \quad t \in [0, 1] \\ y(0) = 0 \end{cases}$$

Osservato che la soluzione è data da

$$y(t) = \log\left(1 - \frac{t^2}{2}\right) \,,$$

determinare il valore dell'errore totale che si commette in $t = 0.2$, $t = 0.5$ e $t = 1$, utilizzando le procedure implementate per il metodo di Eulero in avanti e per il metodo di Eulero all'indietro. Assegnare al passo h i seguenti valori $h = 10^{-2}, 10^{-3}, 10^{-4}$. Che relazione c'è tra l'errore totale ed il passo di discretizzazione?

Bibliografia

[1] Ascher U. M., Petzold L. R. - *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations* - SIAM, 1998.

[2] Butcher J. C. - *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods* - John Wiley & Sons, 1987.

[3] Dahlquist G. - *Convergence and Stability in the Numerical Integration of Ordinary Differential Equations* - Math. Scand. 4, pp.33-53 (1956).

[4] Dahlquist G. - *33 Years of Numerical Instability, Part I* - BIT 25, pp. 188-204 (1985).

[5] Dahlquist G., Björck A. - *Numerical Methods* - Prentice-Hall, 1974.

[6] Gear C. W. - *Numerical initial Value problems in Ordinary Differential Equations* - Prentice-Hall, 1971.

[7] Henrici P. - *Discrete Variable Methods in Ordinary Differential Equations* - John Wiley & Sons, 1962.

[8] `http://www.mathworks.com/`.

[9] Lax P. D., Richtmeyer R. D. - *Survey of the Stability of Linear Finite Difference Equations* - Communications on Pure and Applied Mathematics, vol. IX, pp. 267-293 (1956).

[10] Numerical Algorithm Group - *NAg Library Manual, Mark 22* - Oxford (2009), `http://www.nag.com/`.

[11] *ODE* - `http://www.netlib.org/ode/`.

[12] *ODEPACK* - `http://www.netlib.org/odepack/`.

[13] *Transactions on Mathematical Software (TOMS)* - `http://www.netlib.org/toms`.

[14] Visual Numerics - *IMSL C Numerical Library version 7.0 (2008), IMSL Fortran Numerical Library Version 6.0 (2007)* - `http://www.vni.com`.

Capitolo 4

Calcolo matriciale: metodi iterativi

4.1 Introduzione

Nella risoluzione di sistemi di equazioni lineari la scelta dell'algoritmo risolutivo più efficace richiede un'analisi delle caratteristiche del sistema. In molte applicazioni i sistemi sono di **grandi dimensioni** (di ordine elevato), cioè con un grande numero di equazioni e quindi di incognite, e **sparsi**, cioè i coefficienti di molte incognite sono uguali a zero (questo significa che ogni equazione coinvolge solo un piccolo numero di incognite).

♣ **Esempio 4.1.** Si consideri un oggetto bidimensionale. Si assuma per semplicità che l'oggetto sia quadrato e sia suddiviso in $n = d^2$ quadrati (vedi Figura 4.1, in cui $d = 8$), detti *pixel*, numerati da 1 a n. Una sorgente di raggi-X emette un fascio di raggi che, dopo aver attraversato l'oggetto, è intercettato da un rilevatore che ne misura l'assorbimento totale da parte dell'oggetto. Si vuole calcolare il coefficiente di assorbimento di ogni pixel in cui è stato suddiviso l'oggetto.
Tale tecnica, detta *tomografia*, è utilizzata in medicina diagnostica; l'oggetto è una sezione piana del corpo umano e viene irradiata con raggi-X al fine di scoprire eventuali anomalie organiche (radiografia, TAC, ...).
Per ogni raggio che attraversa l'oggetto l'assorbimento totale del raggio è dato dalla somma degli assorbimenti nei pixel. Indicato con b_i l'assorbimento totale del raggio $i-$mo, misurato dal rilevatore, con x_j, $j = 1, ..., n$, il coefficiente di assorbimento del j-mo pixel (incognite del problema) e con a_{ij} il peso del contributo del j-mo pixel all'assorbimento totale dell'i-mo raggio, si ha:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i \ . \tag{4.1}$$

Se n è il numero di raggi proiettati, si ottiene un sistema di n equazioni in n incognite, che nel seguito si chiamerà sistema `PIXEL`, la cui soluzione costituisce l'insieme dei coefficienti di assorbimento dei pixel in cui è stato suddiviso l'oggetto.
Supponendo che i raggi siano proiettati perpendicolarmente (vedi Figura 4.1), ogni raggio attraversa solo d pixel. Di conseguenza, la generica equazione (4.1) del sistema `PIXEL` ha almeno $n - d$ coefficienti a_{ij} uguali a zero, cioè quelli relativi ai pixel non attraversati dall'i-mo raggio e che, quindi, non danno contributo al suo assorbimento. Il numero totale di coefficienti nulli del sistema `PIXEL` è quindi $n\times(n-d)$.
Una misura di "quanto è sparso" il sistema `PIXEL` si ha considerando il rapporto tra il numero di coefficienti nulli e il numero complessivo dei coefficienti ($n \times n$), cioè:

$$SP = \frac{n(n-d)}{n^2} = 1 - \frac{d}{n}.$$

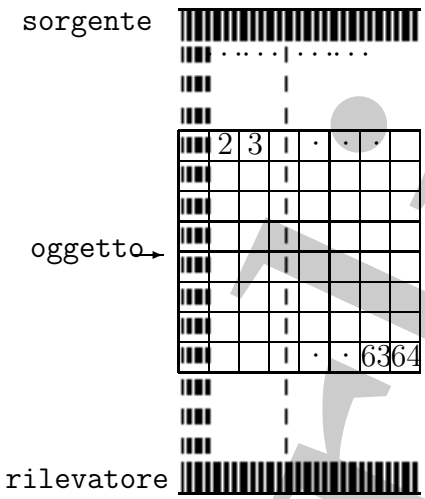

Figura 4.1: Analisi della struttura interna di un oggetto mediante proiezione di raggi-X.

d	n^2	$n(n-d)$	SP
16	65536	61440	0.937
32	1048576	1015808	0.968
64	16777216	16515072	0.984
128	268435456	266338304	0.992

Tabella 4.1: Grado di sparsità del sistema `PIXEL` al variare di d.

Nel caso considerato in Figura 4.1 si ha:

$$
\begin{aligned}
&d = 8\\
&n = d^2 = 64\\
&n \times (n-d) = 3584\\
&n \times n = 4096
\end{aligned}
$$

e, quindi, $SP = 0.875$. In altri termini, il numero di coefficienti nulli del sistema `PIXEL` è l'87.5% del numero complessivo di coefficienti. In Tabella 4.1 sono riportati i valori di SP al variare di d tra 16 e 128. Come si aspettava, la "sparsità" del sistema `PIXEL` aumenta al crescere di d e quindi delle sue dimensioni. Si osserva, infine, che, nella pratica, per effettuare un'analisi sufficientemente accurata della struttura interna dell'oggetto, è necessario considerare sia un numero di pixel elevato sia un numero di raggi elevato. In generale $n > 10^5$ e, di conseguenza, il sistema di equazioni risultante è da considerarsi di grandi dimensioni e con sparsità elevata.

Definizione 4.1.1. (Grado di sparsità)
Dato un sistema lineare di n equazioni in n incognite, sia p il numero dei coefficienti diversi da zero. Si definisce **grado di sparsità** *del sistema la quantità:*

$$SP = \frac{n^2 - p}{n^2} = 1 - \frac{p}{n^2}.$$

Si ha $0 \leq SP \leq 1$[1].

Se il sistema è di grandi dimensioni e ha anche un grado di sparsità elevato è importante utilizzare metodi *ad hoc* che sfruttino, da un punto di vista della complessità computazionale, il fatto che la maggior parte dei coefficienti sono nulli.

♣ **Esempio 4.2.** Si consideri il seguente sistema lineare di 3 equazioni in 3 incognite:

$$\begin{cases} 3x_1 & +4x_2 & + x_3 & = 7 \\ 2x_1 & & +4x_3 & = 6 \\ x_1 & +2x_2 & & = 3. \end{cases} \tag{4.2}$$

Dopo il primo passo del metodo di eliminazione di Gauss tale sistema è trasformato[2] nel sistema equivalente:

$$\begin{cases} 3x_1 & + & 4x_2 & + & x_3 & = 7 \\ & & -8/3x_2 & +10/3x_3 & = 4/3 \\ & & 2/3x_2 & - & 1/3x_3 & = 2/3. \end{cases}$$

[1]Data una matrice A di dimensione $n \times m$, se p è il numero degli elementi non nulli, si ha:

$$SP = \frac{nm - p}{nm} = 1 - \frac{p}{nm}.$$

Se tutti gli elementi di A sono non nulli, allora $p = n \times m$ e si ha $SP = 0$. Se, al contrario, tutti gli elementi di A sono nulli (in tal caso A è la matrice nulla), allora $p = 0$ e $SP = 1$. Quindi, si ha $0 \leq SP \leq 1$.
In particolare, considerando le matrici quadrate ($n = m$) non singolari (cioè con determinante non nullo), una matrice che ha il minor numero di elementi non nulli è quella diagonale. Tale matrice ha, in questa classe di matrici, il massimo grado di sparsità, che è dato da:

$$SP = \frac{n(n-1)}{n^2} = 1 - \frac{1}{n}.$$

Si può affermare che una matrice (o un sistema lineare) ha un elevato grado di sparsità se SP è molto vicino a 1.
Si osservi, infine, che, data una matrice A quadrata di ordine n, se d è il numero di elementi diversi da zero su ciascuna riga (come nella matrice dei coefficienti del sistema PIXEL dell'esempio 4.1), si ha:

$$SP = \frac{n(n-d)}{n^2} = 1 - \frac{d}{n}.$$

[2]A causa di tali modifiche può accadere che coefficienti uguali a zero diventino non nulli. Tale fenomeno è detto **fill-in**.

Si osserva che i coefficienti a_{22} e a_{33}, inizialmente nulli, sono divenuti non nulli (rispettivamente $-8/3$ e $-1/3$) dopo l'applicazione del primo passo del metodo.
Il metodo di eliminazione di Gauss, almeno nella sua forma classica[3], non "sfrutta" il fatto che il sistema ha molti coefficienti nulli. Tale inefficienza è ancora più evidente per sistemi sparsi di dimensioni maggiori rispetto a quelle del sistema (4.2).
Per chiarire questa considerazione si supponga di risolvere un sistema di 10 equazioni in 10 incognite. Il metodo di eliminazione di Gauss applicato a tale sistema richiede circa 400 operazioni floating-point per ottenere la soluzione. Se il sistema ha solo 2 coefficienti non nulli per ciascuna equazione, il numero totale di coefficenti diversi da zero è 20, cioè la quinta parte del numero complessivo di coefficienti del sistema. Sarebbe, quindi, desiderabile ottenere la soluzione con un numero di operazioni uguale a 400/5. Sfortunatamente, la complessità del metodo di eliminazione di Gauss è, in generale, sempre la stessa, anche se il sistema presenta molti coefficienti uguali a zero. Ciò è dovuto al fatto che il metodo di eliminazione di Gauss modifica i coefficienti del sistema, durante il processo di trasformazione del sistema dato in un sistema triangolare equivalente. ♣

Una valida alternativa è costituita dai **metodi iterativi**. Essi consentono di sfruttare l'eventuale sparsità del sistema perché non modificano i suoi coefficienti. Si analizzano, di seguito, le caratteristiche fondamentali dei metodi iterativi e gli aspetti di base per un loro utilizzo efficiente. In particolare, tale analisi è svolta, inizialmente, attraverso l'esame di due tra i più noti ed antichi metodi iterativi, il **metodo di Jacobi** ed il **metodo di Gauss-Seidel**.

4.2 I metodi di Jacobi e Gauss-Seidel

♣ **Esempio 4.3.** Si consideri il sistema lineare di 3 equazioni in 3 incognite:

$$\begin{cases} 8x_1 - x_2 + 2x_3 = 56 \\ x_1 - 4x_2 + x_3 = -1 \\ x_1 + 2x_2 - 5x_3 = -37. \end{cases} \tag{4.3}$$

Per risolvere tale sistema si procede nel modo seguente. Si riscrive il sistema (4.3) in modo che a primo membro della i-ma equazione, $i = 1, 2, 3$, compaia solo l'incognita x_i. Questa semplice "riscrittura" può sempre essere fatta[4], eventualmente, modificando l'ordine delle equazioni:

$$\begin{cases} 8x_1 = x_2 - 2x_3 + 56 \\ -4x_2 = -x_1 - x_3 - 1 \\ -5x_3 = -x_1 - 2x_2 - 37. \end{cases} \tag{4.4}$$

Si dividono ambo i membri della i-ma equazione del sistema (4.4), $i = 1, 2, 3$, per il coefficiente di x_i. Il sistema (4.3) è quindi trasformato nel sistema equivalente:

[3] È importante ricordare che per alcune classi di sistemi di equazioni lineari sparsi, in cui i coefficienti non nulli sono disposti secondo un preciso schema (in tal caso si parla di **sistemi sparsi strutturati**), esistono opportune versioni del metodo di eliminazione di Gauss o, in generale, opportuni metodi diretti (ad esempio il metodo di Cholesky per sistemi lineari a banda simmetrici definiti positivi), che consentono di minimizzare il fenomeno del fill-in e, di conseguenza, di ridurre la complessità computazionale.

[4] Se il sistema è non singolare.

$$\begin{cases} x_1 = & \frac{1}{8}(& x_2 & -2x_3 & +56) \\ x_2 = & -\frac{1}{4}(-x_1 & - & x_3 & - 1) \\ x_3 = & -\frac{1}{5}(-x_1 & -2x_2 & & -37). \end{cases} \tag{4.5}$$

Questa rappresentazione del sistema è il punto di partenza per costruire un processo iterativo: infatti, se si scelgono ad arbitrio dei valori per le incognite x_1, x_2 e x_3 utilizzandoli nel secondo membro delle equazioni (4.5), si ottengono nuovi valori per x_1, x_2 e x_3. Indicati con $x_1^{(0)}$, $x_2^{(0)}$ e $x_3^{(0)}$ i valori iniziali scelti si ha:

$$\begin{cases} x_1^{(1)} = & \frac{1}{8}(& x_2^{(0)} & -2x_3^{(0)} & +56) \\ x_2^{(1)} = & -\frac{1}{4}(-x_1^{(0)} & - & x_3^{(0)} & - 1) \\ x_3^{(1)} = & -\frac{1}{5}(-x_1^{(0)} & -2x_2^{(0)} & & -37). \end{cases}$$

Il passo successivo consiste nell'utilizzare nel secondo membro delle equazioni (4.5) i nuovi valori $x_1^{(1)}$, $x_2^{(1)}$ e $x_3^{(1)}$ al posto dei valori iniziali corrispondenti, cioè:

$$\begin{cases} x_1^{(2)} = & \frac{1}{8}(& x_2^{(1)} & -2x_3^{(1)} & +56) \\ x_2^{(2)} = & -\frac{1}{4}(-x_1^{(1)} & - & x_3^{(1)} & - 1) \\ x_3^{(2)} = & -\frac{1}{5}(-x_1^{(1)} & -2x_2^{(1)} & & -37). \end{cases}$$

In generale, se $x_1^{(k)}$, $x_2^{(k)}$ e $x_3^{(k)}$ sono i valori delle incognite x_1, x_2 e x_3 ottenuti dopo k passi, si calcolano nuovi valori $x_1^{(k+1)}$, $x_2^{(k+1)}$ e $x_3^{(k+1)}$ mediante le equazioni:

$$\begin{cases} x_1^{(k+1)} = & \frac{1}{8}(& x_2^{(k)} & -2x_3^{(k)} & +56) \\ x_2^{(k+1)} = & -\frac{1}{4}(-x_1^{(k)} & - & x_3^{(k)} & - 1) \\ x_3^{(k+1)} = & -\frac{1}{5}(-x_1^{(k)} & -2x_2^{(k)} & & -37). \end{cases} \tag{4.6}$$

Il procedimento descritto è, quindi, di tipo iterativo; a partire da valori iniziali arbitrari delle incognite $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$, si esegue una sequenza di passi, detti **iterazioni**, in ognuno dei quali si calcolano nuovi valori per le incognite "sostituendo" i valori calcolati al passo precedente. Si noti inoltre che ad ogni iterazione i coefficienti del sistema (4.5) non cambiano e sono semplicemente legati in maniera opportuna ai coefficienti del sistema iniziale (4.3).
Le (4.6) definiscono la generica iterazione del procedimento descritto, noto con il nome di **metodo di Jacobi**. Nella Tabella 4.2 sono riportati i valori[5], arrotondati alla sesta cifra decimale, ottenuti nelle prime 9 iterazioni del metodo di Jacobi, avendo scelto come valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$. La soluzione del sistema (4.3) è $x_1 = 5.000$, $x_2 = 4.000$ e $x_3 = 10.000$. Si osserva che già dopo 5 iterazioni si hanno valori che sono sufficientemente vicini alla soluzione. ♣

Si consideri il sistema lineare di n equazioni in n incognite:

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 & + \ldots + & a_{1n}x_n & = b_1 \\ a_{21}x_1 + & a_{22}x_2 & + \ldots + & a_{2n}x_n & = b_2 \\ \ldots & \ldots & \ldots & \ldots & = \ldots \\ a_{n1}x_1 + & a_{n2}x_2 & + \ldots + & a_{nn}x_n & = b_n. \end{cases} \tag{4.7}$$

[5] Tali valori e tutti i risultati sperimentali riportati in seguito si riferiscono ad elaborazioni effettuate utilizzando il sistema aritmetico standard IEEE.

Per risolvere tale sistema si procede nel modo seguente. Si riscrive il sistema (4.7) in modo che a primo membro della i-ma equazione, $i = 1, \ldots, n$, compaia solo l'incognita x_i. Questa semplice "riscrittura" è sempre possibile [6], eventualmente modificando l'ordine delle equazioni:

$$\begin{cases} a_{11}x_1 = & -a_{12}x_2 - & a_{13}x_3 & -\ldots + & b_1 \\ a_{22}x_2 = & -a_{21}x_1 - & a_{23}x_3 & -\ldots + & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{kk}x_k = & -a_{k1}x_1 - & a_{k2}x_2 & -\ldots + & b_k \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{nn}x_n = & -a_{n1}x_1 - & a_{n2}x_2 & -\ldots + & b_n. \end{cases} \tag{4.8}$$

Si dividono ambo i membri della i-ma equazione del sistema (4.8), $i = 1, \ldots, n$, per il coefficiente di x_i. Il sistema (4.7) è quindi trasformato nel sistema equivalente:

$$\begin{cases} x_1 = & -\frac{1}{a_{11}}(a_{12}x_2 + & a_{13}x_3 & +\ldots - & b_1) \\ x_2 = & -\frac{1}{a_{22}}(a_{21}x_1 + & a_{23}x_3 & +\ldots - & b_2) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_k = & -\frac{1}{a_{kk}}(a_{k1}x_1 + & a_{k2}x_2 & +\ldots - & b_k) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_n = & -\frac{1}{a_{nn}}(a_{n1}x_1 + & a_{n2}x_2 & +\ldots - & b_n). \end{cases} \tag{4.9}$$

Questa rappresentazione del sistema è il punto di partenza per costruire un processo iterativo: infatti, se si scelgono ad arbitrio dei valori per le incognite x_1, x_2, ..., x_n, utilizzandoli nel secondo membro delle equazioni (4.9), si ottengono nuovi valori per x_1, x_2, ..., x_n. Indicati con $x_1^{(0)}$, $x_2^{(0)}$, ..., $x_n^{(0)}$ i valori iniziali scelti si ha:

$$\begin{cases} x_1^{(1)} = & -\frac{1}{a_{11}}(a_{12}x_2^{(0)} + & a_{13}x_3^{(0)} & +\ldots - & b_1) \\ x_2^{(1)} = & -\frac{1}{a_{22}}(a_{21}x_1^{(0)} + & a_{23}x_3^{(0)} & +\ldots - & b_2) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_k^{(1)} = & -\frac{1}{a_{kk}}(a_{k1}x_1^{(0)} + & a_{k2}x_2^{(0)} & +\ldots - & b_k) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_n^{(1)} = & -\frac{1}{a_{nn}}(a_{n1}x_1^{(0)} + & a_{n2}x_2^{(0)} & +\ldots - & b_n) \end{cases}$$

Il passo successivo consiste nell'utilizzare, nel secondo membro delle equazioni (4.9), i

[6] Se il sistema è non singolare.

nuovi valori $x_1^{(1)}$, $x_2^{(1)}$, ..., $x_n^{(1)}$ al posto dei valori iniziali corrispondenti, cioè:

$$\begin{cases} x_1^{(2)} = -\frac{1}{a_{11}}(a_{12}x_2^{(1)} + a_{13}x_3^{(1)} + \ldots - b_1) \\ x_2^{(2)} = -\frac{1}{a_{22}}(a_{21}x_1^{(1)} + a_{23}x_3^{(1)} + \ldots - b_2) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ x_k^{(2)} = -\frac{1}{a_{kk}}(a_{k1}x_1^{(1)} + a_{k2}x_2^{(1)} + \ldots - b_k) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ x_n^{(2)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(1)} + a_{n2}x_2^{(1)} + \ldots - b_n). \end{cases}$$

In generale, se $x_1^{(k)}$, $x_2^{(k)}$, ..., $x_n^{(k)}$ sono i valori delle incognite x_1, x_2, ..., x_n ottenuti dopo k passi, si calcolano i nuovi valori $x_1^{(k+1)}$, $x_2^{(k+1)}$, ..., $x_n^{(k+1)}$ mediante le equazioni:

$$\begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \ldots - b_1) \\ x_2^{(k+1)} = -\frac{1}{a_{22}}(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \ldots - b_2) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ x_k^{(k+1)} = -\frac{1}{a_{kk}}(a_{k1}x_1^{(k)} + a_{k2}x_2^{(k)} + \ldots - b_k) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\ x_n^{(k+1)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \ldots - b_n). \end{cases} \tag{4.10}$$

Il procedimento descritto è, quindi, di tipo iterativo; a partire da valori iniziali arbitrari, delle incognite ($x_1^{(0)}$, $x_2^{(0)}$, ..., $x_n^{(0)}$), si esegue una sequenza di passi, detti **iterazioni**, in ognuno dei quali si calcolano nuovi valori per le incognite "sostituendo" i valori calcolati al passo precedente. Si noti inoltre che ad ogni iterazione i coefficienti del sistema (4.9) non cambiano e sono semplicemente legati in maniera opportuna ai coefficienti del sistema iniziale (4.7).
Le (4.10) definiscono la generica iterazione del procedimento descritto, noto con il nome di **metodo di Jacobi**.
Nella notazione utilizzata, gli indici in alto si riferiscono al numero di iterazioni; in altre parole, $x_1^{(k+1)}$, $x_2^{(k+1)}$, $x_n^{(k+1)}$ sono i valori ottenuti alla $(k+1)$-ma iterazione utilizzando i valori $x_1^{(k)}$, $x_2^{(k)}$ e $x_n^{(k)}$ calcolati nell'iterazione precedente.

♣ **Esempio 4.4.** Riprendiamo l'esempio 4.3. Considerando le (4.6), si osserva che quando $x_1^{(k+1)}$ è stato calcolato, è possibile utilizzare tale valore al posto di $x_1^{(k)}$, per determinare il valore $x_2^{(k+1)}$. Analogamente, per calcolare $x_3^{(k+1)}$ è possibile utilizzare i valori di x_1 e x_2 già calcolati alla $(k+1)$-ma

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	7.000000	0.250000	7.400000
2	5.181250	3.850000	8.900000
3	5.256250	3.770312	9.976250
4	4.977227	4.058125	9.959375
5	5.017422	3.984150	10.018700
6	4.993345	4.009029	9.997145
7	5.001842	3.997622	10.002280
8	4.999133	4.001031	9.999417
9	5.000275	3.999638	10.000240
⋮	⋮	⋮	⋮

Tabella 4.2: Valori ottenuti nelle prime 9 iterazioni del metodo di Jacobi applicato al sistema (4.3).

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	7.000000	2.000000	9.600000
2	4.850000	3.862500	9.915000
3	5.004063	3.979766	9.992719
4	4.999291	3.998003	9.999060
5	4.999986	3.999761	9.999902
6	4.999995	3.999974	9.999989
⋮	⋮	⋮	⋮

Tabella 4.3: Valori ottenuti nelle prime 6 iterazioni del metodo di G-S applicato al sistema (4.3).

iterazione. Quanto detto conduce al seguente metodo iterativo:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{8}(\quad x_2^{(k)} \quad -2x_3^{(k)} \quad +56) \\ x_2^{(k+1)} = -\frac{1}{4}(-x_1^{(k+1)} \quad - \quad x_3^{(k)} \quad - \quad 1) \\ x_3^{(k+1)} = -\frac{1}{5}(-x_1^{(k+1)} \quad -2x_2^{(k+1)} \quad -37), \end{cases} \tag{4.11}$$

noto con il nome di **metodo di Gauss-Seidel (G-S)**.
Nella Tabella 4.3 sono riportati i valori ottenuti nelle prime 6 iterazioni del metodo di G-S, scegliendo come valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$.

Si osserva che i valori ottenuti con il metodo di G-S si avvicinano alla soluzione più rapidamente, cioè con un minore numero di iterazioni, di quelli ottenuti con il metodo di Jacobi. Tale fenomeno si verifica spesso, ma non sempre: infatti, per alcuni sistemi di equazioni lineari, il metodo di Jacobi genera valori che si avvicinano alla soluzione del sistema, mentre ciò non accade per il metodo di G-S. ♣

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	-1.000000	-3.000000
2	5.000000	-3.000000	-3.000000
3	1.000000	1.000000	1.000000
4	1.000000	1.000000	1.000000
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Tabella 4.4: Valori ottenuti nelle prime 4 iterazioni del metodo di Jacobi applicato al sistema (4.13).

Riprendiamo il sistema (4.10). Si osserva che quando $x_1^{(k+1)}$ è stato calcolato, è possibile utilizzare tale valore al posto di $x_1^{(k)}$, per determinare il valore $x_2^{(k+1)}$. Analogamente, per calcolare $x_n^{(k+1)}$ è possibile utilizzare i valori di $x_1^{(k+1)}$, $x_2^{(k+1)}$, ..., $x_{n-1}^{(k+1)}$ già calcolati alla $(k+1)$-ma iterazione. Quanto detto conduce al seguente metodo iterativo:

$$\begin{cases} x_1^{(k+1)} = -\frac{1}{a_{11}}(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \ldots - b_1) \\ x_2^{(k+1)} = -\frac{1}{a_{22}}(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \ldots - b_2) \\ \quad \ldots \qquad\qquad \ldots \qquad\qquad \ldots \\ x_n^{(k+1)} = -\frac{1}{a_{nn}}(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + \ldots - b_n), \end{cases} \tag{4.12}$$

noto con il nome di **metodo di Gauss-Seidel (G-S)**.

♣ **Esempio 4.5.** Si consideri il sistema lineare:

$$\begin{cases} x_1 - 2x_2 + 2x_3 = 1 \\ -x_1 + x_2 - x_3 = -1 \\ -2x_1 - 2x_2 + x_3 = -3, \end{cases} \tag{4.13}$$

la cui soluzione è $x_i = 1$, $i = 1, 2, 3$.
Nelle Tabelle 4.4 e 4.5 sono riportati alcuni valori ottenuti risolvendo tale sistema con i metodi di Jacobi e di G-S. Si nota che i valori relativi al metodo di Jacobi si avvicinano alla soluzione (nel caso particolare i valori sono proprio uguali alla soluzione) già dopo 3 iterazioni, mentre i valori generati dal metodo di G-S si allontanano sempre di più ad ogni iterazione.

♣

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	0.000000	-1.000000
2	3.000000	1.000000	5.000000
3	-7.000000	-3.000000	-23.00000
4	41.00000	17.00000	113.0000
5	-191.0000	-79.00000	-543.0000
6	929.0000	385.0000	2625.000
⋮	⋮	⋮	⋮

Tabella 4.5: Valori ottenuti nelle prime 6 iterazioni del metodo di G-S applicato al sistema (4.13).

Si consideri ora il sistema lineare di n equazioni in n incognite (4.7), che può essere scritto anche nelle due forme equivalenti:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, \ldots, n, \tag{4.14}$$

$$Ax = b,$$

con $A = [a_{ij}]$[7] matrice di dimensione $n \times n$, $b = [b_i]$ e $x = [x_i]$ vettori di dimensione n. In relazione a tale sistema, le (4.6) si generalizzano come segue:

METODO DI JACOBI

$$x_i^{(k+1)} = \left(b_i - a_{i1}x_1^{(k)} - \cdots - a_{i,i-1}x_{i-1}^{(k)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{in}x_n^{(k)}\right)/a_{ii} = \\ = \left(b_i - \sum_{j=1,j\neq i}^{n} a_{ij}x_j^{(k)}\right)/a_{ii}, \quad i = 1, \ldots, n; \quad k \geq 0. \tag{4.15}$$

Ovviamente, affinchè le (4.15) siano definite è necessario che gli elementi diagonali di A, a_{ii}, $i = 1, \ldots, n$, siano diversi da zero[8].
In relazione al sistema (4.14), il metodo di G-S è definito da:

[7] Si assume qui e nel seguito che la matrice dei coefficienti $A \in \Re^{n\times n}$, del sistema (4.14) sia non singolare, cioè con determinante diverso da zero, e, quindi, che il sistema (4.14) ammetta una ed una sola soluzione.

[8] Se il sistema (4.14) è non singolare, ciò può essere sempre ottenuto scambiando in modo opportuno le equazioni.

METODO DI GAUSS-SEIDEL

$$x_i^{(k+1)} = \left(b_i - a_{i1}x_1^{(k+1)} - \cdots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \cdots - a_{in}x_n^{(k)}\right)/a_{ii} =$$
$$= \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)/a_{ii}, \quad i = 1, \ldots, n; \quad k \geq 0. \tag{4.16}$$

Come per il metodo di Jacobi, si assume che gli elementi diagonali di A siano diversi da zero.

4.2.1 Primi esempi di algoritmi

Un primo esempio di algoritmo per la risoluzione di un sistema di equazioni lineari con il metodo iterativo di Jacobi è illustrato nella **Procedura 4.1**. Essa è scritta nel linguaggio *pascal-like* e prevede come dati di input la dimensione del sistema `n`, la matrice dei coefficienti del sistema, rappresentata dall'array `A`, il vettore dei termini noti, rappresentato dall'array `b`, i valori iniziali contenuti nell'array `xold` ed il numero di iterazioni da effettuare, `MaxIt`. I soli dati di output sono i valori ottenuti dopo l'esecuzione di `MaxIt` iterazioni del metodo di Jacobi, immagazzinati nell'array `x`. Si osservi che, nella procedura in esame, l'array `xold` contiene i valori calcolati all'iterazione precedente, mentre l'array `x` contiene i nuovi valori da calcolare.
Una prima versione dell'algoritmo basato sul metodo di G-S è invece illustrato nella **Procedura 4.2**. Si osservi che in tale procedura, a differenza di quella relativa al metodo di Jacobi, non è necessario utilizzare l'array `xold`, poiché, per calcolare i nuovi valori `x(i)` alla generica iterazione, si utilizzano i valori `x(j)`, `j` $<$ `i`, già calcolati in quella stessa iterazione.

```
procedure Jacobi1(in: n, A, b, xold, MaxIt ; out: x)
 /# SCOPO: risoluzione di Ax=b con il metodo di Jacobi
 /# SPECIFICHE PARAMETRI:
 /# PARAMETRI DI INPUT:
   var: n        : intero          {dimensione del sistema}
   var: A(n,n)   : array di reali  {coefficienti del sistema}
   var: b(n)     : array di reali  {termini noti del sistema}
   var: xold(n)  : array di reali  {valori iniziali}
   var: MaxIt    : intero          {numero di iterazioni da effettuare}
 /# PARAMETRI DI OUTPUT:
   var: x(n)     : array di reali  {soluzione calcolata}
 /# VARIABILI LOCALI:
   var: i,j,k    : interi          {contatori}
 /# INIZIO ISTRUZIONI:
      for k = 1, MaxIt do                    {ciclo sul numero di iterazioni}
        for i = 1, n do                      {ciclo per il calcolo di x^(k)}
          x(i) := b(i);
          for j = 1, n do                    {ciclo per il calcolo di x_i^(k)}
            if i ≠ j then
              x(i) := x(i) - A(i,j) * xold(j);
            endif
          endfor
          x(i) := x(i)/A(i,i);
        endfor
        for i = 1, n do                      {aggiornamento di xold}
          xold(i) := x(i);
        endfor
      endfor
      return x
    end Jacobi1
```

Procedura 4.1: Algoritmo di Jacobi (prima versione)

```
procedure G-S1(in: n, A, b, x, MaxIt ; out: x)
 /# SCOPO: risoluzione di Ax=b con il metodo di Gauss-Seidel
 /# SPECIFICHE PARAMETRI:
 /# PARAMETRI DI INPUT:
   var: n       : intero           {dimensione del sistema}
   var: A(n,n)  : array di reali   {coefficienti del sistema}
   var: b(n)    : array di reali   {termini noti del sistema}
   var: x(n)    : array di reali   {valori iniziali}
   var: MaxIt   : intero           {numero di iterazioni da effettuare}
 /# PARAMETRI DI OUTPUT:
   var: x(n)    : array di reali   {soluzione calcolata}
 /# VARIABILI LOCALI:
   var: i,j,k   : interi           {contatori}
 /# INIZIO ISTRUZIONI:
     for k = 1, MaxIt do                    {ciclo sul numero di iterazioni}
       for i = 1, n do                      {ciclo per il calcolo di x^(k)}
         x(i) := b(i);
         for j = 1, n do                    {ciclo per il calcolo di x_i^(k)}
           if i ≠ j then
             x(i) := x(i) − A(i,j) * x(j);
           endif
         endfor
         x(i) := x(i)/A(i,i);
       endfor
     endfor
     return x
   end G-S1
```

Procedura 4.2: Algoritmo di G-S (prima versione)

4.2.2 Complessità computazionale

Considerando la generica iterazione del metodo, si osserva che il calcolo di ciascuna componente di $x^{(k+1)}$ richiede $n-1$ moltiplicazioni, $n-2$ addizioni, 1 sottrazione e 1 divisione. Contando una divisione come una moltiplicazione ed una sottrazione come un'addizione, il numero totale di operazioni richieste da un'iterazione del metodo di Jacobi è:

$$n^2\mathcal{M} + n(n-1)\mathcal{A}, \tag{4.17}$$

dove $\mathcal{A}$ denota un'addizione e $\mathcal{M}$ denota una moltiplicazione e, dunque, una stima del numero di flops, T_{jac}, richieste da una iterazione è

$$T_{jac} = \mathcal{O}(n^2). \tag{4.18}$$

Dalla (4.18) si rileva che il costo di ciascuna iterazione del metodo è equivalente a quello del calcolo del prodotto tra una matrice ed un vettore.
Nel derivare la stima (4.18) si è supposto che, per ciascuna componente di $x^{(k+1)}$, tutte le operazioni in (4.15) siano effettivamente eseguite. Se invece il sistema lineare da risolvere ha dei coefficienti a_{ij} nulli, le operazioni nelle quali si utilizzano tali coefficienti possono non essere eseguite e, quindi, non devono essere considerate nel calcolo del numero di flops richieste dal metodo di Jacobi. Per chiarire questa considerazione si assuma che il sistema abbia solo p coefficienti non nulli, con $p < n^2$. Di conseguenza, il calcolo di $x^{(k+1)}$ richiede un numero di flops circa uguale a p. In tal caso, la (4.18) si generalizza nella:

$$T_{jac} = \mathcal{O}(p) = \mathcal{O}(n^2(1-(1-p/n^2))) = \mathcal{O}(n^2(1-SP)), \tag{4.19}$$

dove SP è il grado di sparsità del sistema. Si osservi che, quando $SP = 0$ (cioè, tutti i coefficienti sono non nulli), dalla (4.19) si ottiene la (4.18). Inoltre, la (4.19) mostra che se il sistema ha un grado di sparsità elevato (che significa $p << n^2$ e, quindi, SP molto vicino a 1), il costo di un'iterazione del metodo di Jacobi si riduce notevolmente. Come già detto ciò rappresenta uno dei vantaggi dell'utilizzo dei metodi iterativi nella risoluzione di sistemi di dimensioni elevate.
Dalla (4.16), si verifica che tutte le considerazioni appena fatte sulla complessità computazionale del metodo di Jacobi si possono applicare anche al metodo di G-S giungendo alle stesse conclusioni, e quindi:

$$T_{gs} = \mathcal{O}(n^2(1-SP)).$$

4.2.3 Interpretazione geometrica

Si consideri il sistema lineare di due equazioni:

$$\begin{cases} 2x_1 - x_2 = 0 \\ x_1 - 3x_2 = 0, \end{cases} \tag{4.20}$$

ciascuna della quali, in un sistema di riferimento di assi cartesiani in cui x_1 è l'asse delle ascisse e x_2 è l'asse delle ordinate, rappresenta una retta del piano; il loro punto di

intersezione, l'origine, è la soluzione del sistema (vedi Figura 4.2). Il metodo di Jacobi applicato a tale sistema ha la forma:

$$\begin{cases} x_1^{(k+1)} &= \frac{1}{2}x_2^{(k)} \\ x_2^{(k+1)} &= \frac{1}{3}x_1^{(k)}. \end{cases}$$

Quindi, se $x^{(0)}$ è il punto iniziale, la prima equazione è risolta rispetto alla variabile x_1 con la variabile x_2 fissata. Dalla Figura 4.2 si evince che geometricamente ciò significa determinare sulla retta definita da tale equazione il punto P che ha la stessa ordinata di $x^{(0)}$. Procedendo allo stesso modo per la seconda equazione, si determina sulla retta ad essa relativa il punto Q che ha la stessa ascissa di $x^{(0)}$. Il nuovo punto $x^{(1)}$ si ottiene geometricamente come estremo del vettore somma dei vettori definiti dai punti

$$(x^{(0)}, P) \quad \text{e} \quad (x^{(0)}, Q)$$

così determinati, puntati in $x^{(0)}$. Nelle iterazioni successive si procede in maniera analoga e si osserva che ci si avvicina sempre di più alla soluzione.

Infine, in Figura 4.3 è illustrata l'interpretazione geometrica del metodo di G-S applicato al sistema (4.20):

$$\begin{cases} x_1^{(k+1)} &= \frac{1}{2}x_2^{(k)} \\ x_2^{(k+1)} &= \frac{1}{3}x_1^{(k+1)}. \end{cases}$$

A partire da $x^{(0)}$, il primo passo è identico a quello di Jacobi. Il punto $x^{(1)}$, invece, si ottiene a partire dal punto determinato sulla prima retta e considerando il punto sulla seconda retta che ha la stessa ascissa. Confrontando le Figure 4.2 e 4.3 si può osservare graficamente che per il sistema in esame il metodo di G-S si avvicina alla soluzione più velocemente del metodo di Jacobi.

4.3 Convergenza

I metodi di Jacobi e di G-S generano n successioni di valori $\{x_i^{(k)}\}$, $i = 1, \ldots, n$, mediante le equazioni (4.15) o (4.16). Quindi, il primo problema relativo all'utilizzo di tali metodi è la determinazione di condizioni che assicurino che tali valori, al crescere di n, si avvicinino sempre di più alla soluzione del sistema. In altre parole, è necessario analizzare il problema della **convergenza** dei metodi di Jacobi e G-S.

Indicata con $x^* = [x_i^*]$ la soluzione del sistema (4.14), si vogliono determinare condizioni opportune per le quali si abbia:

$$\lim_{k\to\infty} x_i^{(k)} = x_i^*, \quad i = 1, \ldots, n.$$

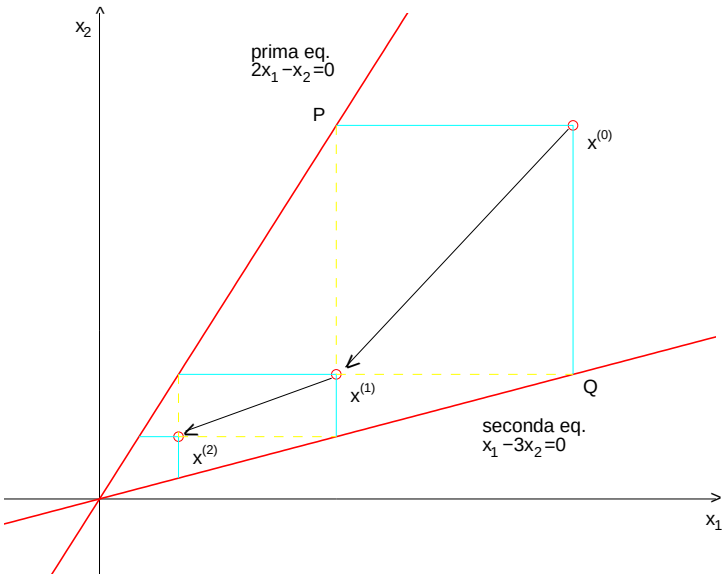

Figura 4.2: Interpretazione geometrica del metodo di Jacobi applicato al sistema (4.20).

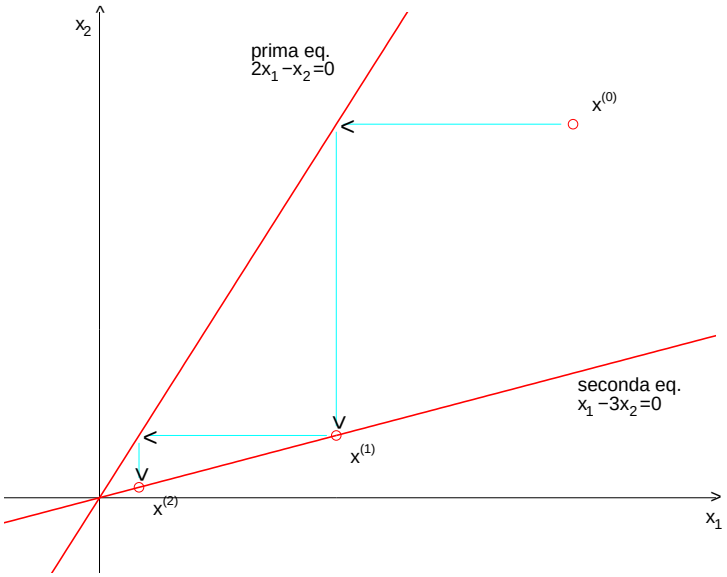

Figura 4.3: Interpretazione geometrica del metodo di G-S applicato al sistema (4.20).

♣ **Esempio 4.6.** Si consideri il sistema lineare:

$$\begin{cases} x_1 & +2x_2 & & = 1 \\ x_1 & + \ x_2 & +x_3 & = 0 \\ & 2x_2 & +x_3 & = 1, \end{cases} \tag{4.21}$$

la cui soluzione è $x_1^* = -1/3$, $x_2^* = 2/3$, $x_3^* = -1/3$. Risolvendo tale sistema con i metodi di Jacobi e di G-S, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$, si hanno i risultati riportati nelle Tabelle 4.6

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	0.000000	1.000000
2	1.000000	-2.000000	1.000000
3	5.000000	-2.000000	5.000000
4	5.000000	-10.00000	5.000000
5	21.00000	-10.00000	21.00000
6	21.00000	-42.00000	21.00000
⋮	⋮	⋮	⋮
15	21845.00	-10922.00	21845.00
⋮	⋮	⋮	⋮

Tabella 4.6: Valori ottenuti nelle prime 15 iterazioni del metodo di Jacobi applicato al sistema (4.21).

e 4.7: entrambi i metodi generano valori che non si avvicinano alla soluzione.

♣

♣ **Esempio 4.7.** Si consideri il sistema lineare:

$$\begin{cases} x_1 & -3x_2 & = 0 \\ 2x_1 & - \ x_2 & = 0, \end{cases} \tag{4.22}$$

ottenuto dal sistema (4.20) semplicemente scambiando le equazioni e la cui soluzione è $x_1^* = 0,\ x_2^* = 0$. Per tale sistema, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = 0.5$, sia il metodo di Jacobi sia il metodo di G-S non convergono. Dopo 10 iterazioni si ha infatti:

$$\begin{aligned} &(Jacobi) \qquad x_1 = 3888;\ \ x_2 = 3888, \\ &(G-S) \qquad x_1 = 1.51165 \times 10^7;\ \ x_2 = 3.02331 \times 10^7. \end{aligned}$$

Tale fenomeno si evidenzia anche nelle Figure 4.4 e 4.5, dove è illustrata l'interpretazione geometrica dei metodi di Jacobi e G-S applicati al sistema in esame. Ricordando che per il sistema (4.20) entrambi i metodi convergono, si deduce che l'ordine delle equazioni può influire sulla convergenza dei metodi in esame. (Si veda l'esempio 4.23.)

♣

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
1	1.000000	-1.000000	3.000000
2	3.000000	-6.000000	13.00000
3	13.00000	-26.00000	53.00000
4	53.00000	-106.0000	213.0000
5	213.0000	-426.0000	853.0000
6	853.0000	-1706.000	3413.000
⋮	⋮	⋮	⋮
15	$2.2 \cdot 10^8$	$-4. \cdot 10^8$	$8.9 \cdot 10^8$
⋮	⋮	⋮	⋮

Tabella 4.7: Valori ottenuti nelle prime 15 iterazioni del metodo di G-S applicato al sistema (4.21).

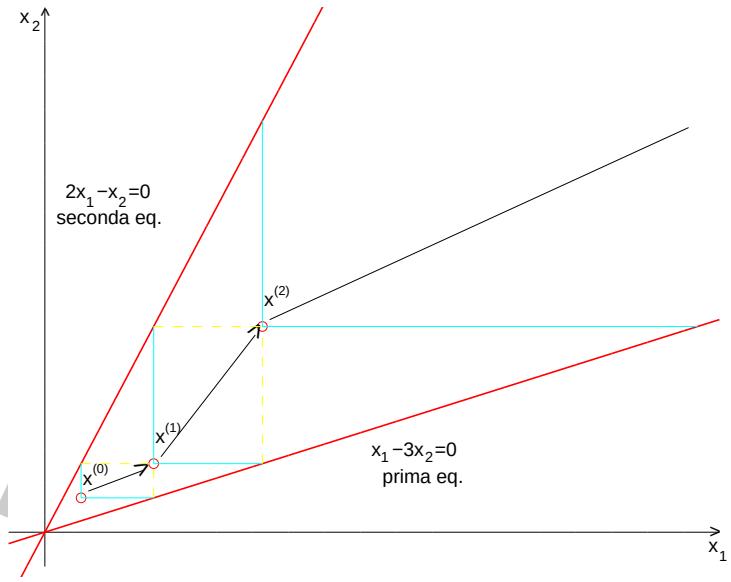

Figura 4.4: Interpretazione geometrica del metodo di Jacobi applicato al sistema (4.22).

Allo scopo di illustrare le idee di base relative allo studio della convergenza dei metodi di Jacobi e di G-S si osservi che entrambi i metodi possono essere espressi nella forma:

$$x_i^{(k+1)} = c_{i1}x_1^{(k)} + \cdots + c_{in}x_n^{(k)} + d_i \; = \\ = \sum_{j=1}^{n} c_{ij}x_j^{(k)} + d_i, \quad i = 1, \ldots, n, \quad k \geq 0, \tag{4.23}$$

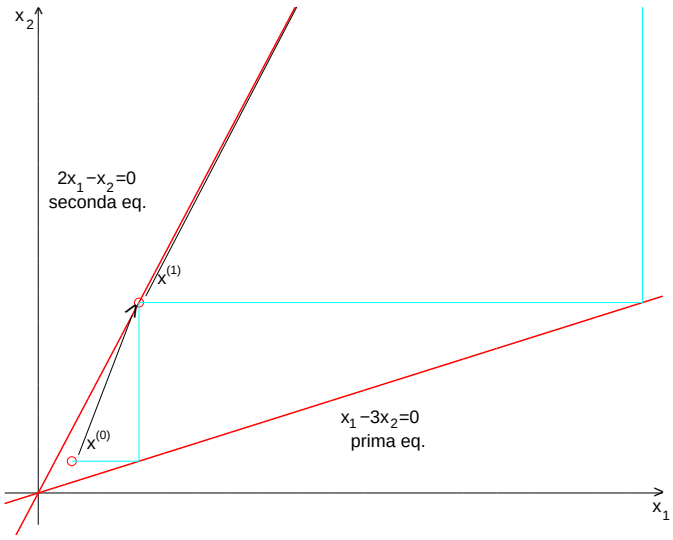

Figura 4.5: Interpretazione geometrica del metodo di G-S applicato al sistema (4.22).

o equivalentemente:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0,$$

dove $C = [c_{ij}]$ è una matrice $n \times n$ e $d = [d_i]$ è un vettore di dimensione n.
È immediato verificare, dalla (4.15), che per il metodo di Jacobi si ha:

$$c_{ij} = \begin{cases} 0 & i = j \\ -a_{ij}/a_{ii} & i \neq j \end{cases}, \quad d_i = b_i/a_{ii}. \tag{4.24}$$

Per il metodo di G-S scritto in forma (4.23) [9] si ha:

$$c_{ij} = \begin{cases} -(\sum_{l=1}^{i-1} a_{il}c_{lj})/a_{ii}, & j = 1, 2, \ldots, i \\ -(\sum_{l=1}^{i-1} a_{il}c_{lj} + a_{ij})/a_{ii}, & j = i+1, \ldots, n \end{cases} \tag{4.25}$$

$$d_i = (b_i - \sum_{l=1}^{i-1} a_{il}d_l)/a_{ii}, \quad i = 1, \ldots, n.$$

Definizione 4.3.1. (Metodo convergente)
Si dice che un metodo del tipo (4.23) è **convergente** *se, qualunque siano i valori iniziali,*

[9] Non è altrettanto immediato scrivere il metodo di G-S nella forma (4.23). Tuttavia ciò può essere fatto con un procedimento per induzione. Si osservi innanzitutto che la (4.16) per $i = 1$ è già in forma (4.23) con $c_{11} = 0$ e $c_{1j} = -a_{1j}/a_{11}$, $j = 2, ..., n$ e $d_1 = b_1/a_{11}$. Si assuma che $x_2^{(k+1)}, ..., x_{i-1}^{(k+1)}$ dati dalla (4.16) abbiano la forma (4.23). Si dimostra ora che anche $x_i^{(k+1)}$ può essere scritto in forma (4.23).

le successioni $\{x_i^{(k)}\}$, $i = 1, ..., n$, *convergono, cioè se*

$$\lim_{k\to\infty} x_i^{(k)} = \bar{x}_i, \quad i = 1, \ldots, n, \quad \forall x_i^{(0)} \ , \tag{4.26}$$

oppure, in modo completamente equivalente, se:

$$\lim_{k\to\infty} x^{(k)} = \bar{x}, \quad \forall x^{(0)},$$

dove $x^{(k)}$ *e* $\bar{x}$ *sono i vettori di componenti, rispettivamente,* $(x_1^{(k)}, ..., x_n^{(k)})$ *e* $(\bar{x_1}, ..., \bar{x_n})$.

Si osservi che, se il metodo (4.23) converge, passando al limite in entrambi i membri delle (4.23), si ha:

$$\bar{x_i} = \sum_{j=1}^{n} c_{ij}\bar{x_j} + d_i \ , \quad i = 1, \ldots, n \ . \tag{4.27}$$

che, in forma compatta, può scriversi come:

$$\bar{x} = C\bar{x} + d \ , \quad o \ \ equivalentemente \ \ [(I - C)\bar{x} = d]$$

Se ora si considera il metodo di Jacobi e si assume che sia convergente, indicato con $\bar{x}$ il limite, dalle (4.24) si ha:

$$\bar{x_i} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}\bar{x_j}\right) = \bar{x_i} + \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{n} a_{ij}\bar{x_j}\right), \quad i = 1, \ldots, n,$$

Infatti, considerando l'espressione di $x_i^{(k+1)}$ in (4.16), si ha:

$$\begin{aligned}
x_i^{(k+1)} &= \frac{1}{a_{ii}}\left[b_i - \sum_{l=1}^{i-1} a_{il}x_l^{(k+1)} - \sum_{l=i+1}^{n} a_{il}x_l^{(k)}\right] \\
&= \frac{1}{a_{ii}}\left[b_i - \sum_{l=1}^{i-1} a_{il}\left(\sum_{j=1}^{n} c_{lj}x_j^{(k)} + d_l\right) - \sum_{l=i+1}^{n} a_{il}x_l^{(k)}\right] \\
&= \sum_{j=1}^{i}\left(\sum_{l=1}^{i-1}\frac{-a_{il}c_{lj}}{a_{ii}}\right)x_j^{(k)} + \sum_{j=i+1}^{n}\left(\sum_{l=1}^{i-1}\frac{-a_{il}c_{lj}}{a_{ii}} - \frac{a_{ij}}{a_{ii}}\right)x_j^{(k)} \\
&\quad + \frac{1}{a_{ii}}\left(b_i - \sum_{l=1}^{i-1} a_{il}d_l\right)
\end{aligned}$$

da cui segue la (4.25). Si osservi che per il metodo di G-S non è conveniente utilizzare l'equazione (4.23) con i coefficienti c_{ij} dati dalle (4.25). Le equazioni (4.16) risultano molto più "semplici". La sola ragione per cui si è introdotta la forma (4.23) è quella di facilitare una discussione elementare sulla convergenza dei metodi iterativi in esame.

e, di conseguenza:

$$\sum_{j=1}^{n} a_{ij}\bar{x_j} = b_i, \quad i = 1, \ldots, n,$$

cioè, il limite è soluzione del sistema $Ax = b$ e, quindi, $\bar{x_i} = x_i^*$, $i = 1, \ldots, n$.
Si può perciò affermare:

Proprietà 4.3.1. *Se il metodo di Jacobi converge, allora il limite è la soluzione del sistema (4.14).*

Con un ragionamento analogo, si può verificare che anche il metodo di G-S gode della proprietà suddetta, che è detta **consistenza** del metodo con il sistema (4.14). È evidente che tale proprietà è essenziale per un qualsiasi metodo iterativo del tipo (4.23).

Definizione 4.3.2. (Metodo consistente)
Un metodo del tipo (4.23) è **consistente**[10] *con il sistema (4.14) se, quando converge, il limite è la soluzione del sistema (4.14).*

Ad ogni iterazione di un metodo iterativo del tipo (4.23), il vettore $x^{(k)}$ rappresenta un'approssimazione della soluzione del sistema (4.14).

Definizione 4.3.3. (Errore di troncamento analitico)
Sia $\{x^k\}$ la successione generata dal metodo iterativo:

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij}x_j^{(k)} + d_i, \quad i = 1, \ldots, n, \quad k \geq 0.$$

Si definisce **errore di troncamento analitico alla k-ma iterazione** *del metodo, il vettore:*

$$e^{(k)} = x^{(k)} - x^*. \tag{4.28}$$

In base alla definizione precedente è evidente che la convergenza di un metodo del tipo (4.23) alla soluzione del sistema (4.14) è equivalente alla convergenza della successione degli errori a zero, cioè alla condizione:

$$\lim_{k\to\infty} |e_i^{(k)}| = 0, \quad i = 1, \ldots, n, \tag{4.29}$$

dove $e_i^{(k)} = (x_i^{(k)} - x_i^*)$ è l'i-ma componente di $e^{(k)}$.

[10] Nei prossimi paragrafi il concetto di consistenza verrà ripreso con maggiore approfondimento.

♣ **Esempio 4.8.** Si consideri il sistema

$$\begin{cases} 8x_1 & - & x_2 & +2x_3 & = & 56 \\ x_1 & -4x_2 & + & x_3 & = & -1 \\ x_1 & +2x_2 & -5x_3 & & = & -37. \end{cases} \tag{4.30}$$

Nella Tabella che segue sono riportati i valori assoluti delle componenti dell'errore relativo alle prime 9 iterazioni (vedi Tabella 4.2) risolvendo il sistema con il metodo di Jacobi. Si osserva che tali quantità si avvicinano a zero al crescere del numero di iterazioni, come ci si aspetta sempre quando il metodo è convergente alla soluzione.

k	$\lvert e_1^{(k)}\rvert$	$\lvert e_2^{(k)}\rvert$	$\lvert e_3^{(k)}\rvert$
1	2.000000	3.750000	2.600000
2	0.181250	0.150000	1.100000
3	0.256250	0.229688	0.023750
4	0.022773	0.058125	0.040625
5	0.017422	0.015850	0.018700
6	0.006655	0.009029	0.002855
7	0.001842	0.002378	0.002280
8	0.000867	0.001031	0.000583
9	0.000275	0.000362	0.000239

♣

♣ **Esempio 4.9.** Si consideri il sistema (4.30). La matrice C relativa al metodo di Jacobi espresso nella forma (4.23) applicato a tale sistema è:

$$C = \begin{bmatrix} 0 & 0.125 & -0.25 \\ 0.25 & 0 & 0.25 \\ 0.2 & 0.4 & 0 \end{bmatrix}.$$

Si ha quindi $\|C\| = 0.6$. ♣

Una semplice condizione sufficiente per la convergenza dei metodi iterativi della forma (4.23) è data dal seguente Teorema:

Teorema 4.3.1. *Il metodo iterativo:*

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i, \quad i = 1, \ldots, n, \quad k \geq 0,$$

è convergente se $\|C\| < 1$.

Dimostrazione Se $\|C\| < 1$, la matrice $I - C$ è non singolare (vedi **Teoremi B.5** e **B.12** dell'**Appendice B** della **Parte prima**) e, quindi, il sistema lineare $x = Cx + d$ ha un'unica soluzione, $\bar{x}$. Posto:

$$e^{(k)} = x^{(k)} - \bar{x}, \ \ \forall k \geq 0,$$

dalle (4.23) e dalle (4.27) si ha:

$$e_i^{(k+1)} = x_i^{(k+1)} - \bar{x}_i = \sum_{j=1}^{n} c_{ij}(x_j^{(k)} - \bar{x}_j) = \sum_{j=1}^{n} c_{ij} e_j^{(k)} \ , \quad i = 1, \ldots, n \ , \tag{4.31}$$

da cui:

$$|e_i^{(k+1)}| \leq \sum_{j=1}^{n} |c_{ij}||e_j^{(k)}| \ , \quad i = 1, \ldots, n. \tag{4.32}$$

Utilizzando le norma infinito per vettori e matrici, dalla (4.32) si ha:

$$|e_i^{(k+1)}| \leq \sum_{j=1}^{n} |c_{ij}|\|e^{(k)}\| \ , \quad i = 1, \ldots, n \ , \tag{4.33}$$

da cui si ottiene:

$$|e_i^{(k+1)}| \leq \|C\|\|e^{(k)}\| \ , \quad i = 1, \ldots, n \ ,$$

e quindi:

$$\|e^{(k+1)}\| \leq \|C\|\|e^{(k)}\| \ .$$

Allo stesso modo si può dimostrare che:

$$\|e^{(k)}\| \leq \|C\|\|e^{(k-1)}\| \ ,$$

da cui segue:

$$\|e^{(k+1)}\| \leq \|C\|\|e^{(k)}\| \leq \|C\|^2\|e^{(k-1)}\| \ .$$

Ripetendo altre $k - 1$ volte tale procedimento, si ottiene:

$$\|e^{(k+1)}\| \leq \|C\|^{k+1}\|e^{(0)}\| \ . \tag{4.34}$$

Se $\|C\| < 1$, il secondo membro della (4.34) converge a zero per $k \to \infty$, qualunque sia $e^{(0)}$. Si ha quindi:

$$\lim_{k\to\infty} \|e^{(k+1)}\| = 0, \ \ \forall e^{(0)},$$

che equivale a:

$$\lim_{k\to\infty} |e_i^{(k+1)}| = 0, \ \ i = 1, ..., n, \ \ \forall e_i^{(0)},$$

da cui:

$$\lim_{k\to\infty} x_i^{(k)} = \bar{x}_i, \ \ i = 1, ..., n, \ \ \forall x_i^{(0)},$$

cioè, il metodo è convergente a $\bar{x}$. ■

Osservazione 4.1. *Il teorema precedente stabilisce una condizione solo sufficiente per la convergenza basata sull'ordine di grandezza dei coefficienti c_{ij} in (4.23). Come già detto esistono altre misure della grandezza di tali coefficienti e per tali misure vale il risultato stabilito dal Teorema 4.3.1*[11].

[11] Ad esempio, se si considera la **norma del valore assoluto** (o **norma 1**), si può verificare che la relazione (4.34) vale ancora; il Teorema 4.3.1 è, quindi, vero anche se si considera tale norma.

Si consideri il metodo di Jacobi espresso nella forma (4.23), dove i coefficienti c_{ij} e d_i sono dati dalle (4.24). Poiché:

$$\|C\| = \max_{1\le i\le n}\left(\sum_{j=1}^{n}|c_{ij}|\right) = \max_{1\le i\le n}\left(\sum_{j=1,j\neq i}^{n}|a_{ij}|/|a_{ii}|\right).$$

Quindi, se:

$$\sum_{j=1,j\neq i}^{n}|a_{ij}|/|a_{ii}| < 1,\quad i=1,\ldots,n,$$

che è equivalente alla condizione:

$$\sum_{j=1,j\neq i}^{n}|a_{ij}| < |a_{ii}|,\quad i=1,\ldots,n, \tag{4.35}$$

si ha $\|C\| < 1$. Un sistema di equazioni lineari in cui i coefficienti a_{ij} soddisfano le (4.35) è detto a **diagonale strettamente dominante**.
Si può quindi enunciare il seguente risultato:

Corollario 4.3.1. *Se il sistema lineare (4.14) è a* **diagonale strettamente dominante***, il metodo di Jacobi è convergente.*

Poichè i risultati dimostrati in questo paragrafo forniscono condizioni solo sufficienti per la convergenza, un metodo iterativo del tipo (4.23) può convergere anche se il sistema non è a diagonale strettamente dominante o, in generale, non soddisfa l'ipotesi del Teorema 4.3.1.

♣ **Esempio 4.10.** Considerato il sistema lineare:

$$\begin{cases} x_1 & -2x_2 & +2x_3 & = 1 \\ 2x_1 & + x_2 & +2x_3 & = 1 \\ x_1 & + x_2 & + x_3 & = 1, \end{cases}$$

la cui soluzione è $x_1^* = -3$, $x_2^* = 1$, $x_3^* = 3$, è immediato verificare che esso non è a diagonale strettamente dominante. Inoltre, considerato il metodo di Jacobi applicato a tale sistema:

$$\begin{cases} x_1^{(k+1)} = +2x_2^{(k)} - 2x_3^{(k)} + 1 \\ x_2^{(k+1)} = -2x_1^{(k)} - 2x_3^{(k)} + 1 \\ x_3^{(k+1)} = -\ x_1^{(k)} - \ x_2^{(k)} + 1\ , \end{cases}$$

si verifica che i coefficienti c_{ij} del metodo non soddisfano l'ipotesi del Teorema 4.3.1. Tuttavia, il metodo di Jacobi converge alla soluzione del sistema, come mostrano i valori riportati nella seguente Tabella.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	1.000000	1.000000	1.000000
1	1.000000	-3.000000	-1.000000
2	-3.000000	1.000000	3.000000
3	-3.000000	1.000000	3.000000
⋮	⋮	⋮	⋮

♣

4.4 Un semplice criterio di arresto

Un aspetto fondamentale legato all'utilizzo efficace dei metodi iterativi in esame è la scelta di un opportuno criterio per decidere quando arrestare il processo iterativo. In generale, un criterio di arresto soddisfacente deve essere tale che il suo utilizzo conduca ad un risultato sufficientemente accurato, o meglio, consenta di ottenere l'accuratezza desiderata. Appare evidente, quindi, che il criterio più naturale è richiedere che la "distanza" tra $x^{(k)}$ e la soluzione del sistema, e cioè l'errore di troncamento analitico, sia abbastanza piccola. Se si utilizza la norma del massimo per misurare la grandezza dell'errore, una possibile condizione per l'arresto del metodo iterativo è richiedere che l'errore assoluto, $\|x^{(k+1)} - x^*\|$, sia minore di una tolleranza `Tol`:

$$\|x^{(k+1)} - x^*\| \leq \texttt{Tol}, \tag{4.36}$$

che rappresenta l'accuratezza richiesta.

♣ **Esempio 4.11.** Si consideri il sistema lineare:

$$\begin{cases} 4x_1 + x_2 - x_3 = -64 \\ 3x_1 + 5x_2 + x_3 = 721 \\ x_1 - 4x_2 + 6x_3 = 807, \end{cases}$$

la cui soluzione è $x_1^* = 9$, $x_2^* = 99$, $x_3^* = 199$. Nella Tabella che segue sono riportati i valori ottenuti risolvendo tale sistema con il metodo di Jacobi, partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$. Inoltre, per ciascuna iterazione, è indicato l'errore assoluto. Si osserva che se si utilizzasse come criterio di arresto la condizione (4.36) con `Tol`$= 10^{-3}$, il metodo si fermerebbe dopo 17 iterazioni e si otterrebbe un'approssimazione della soluzione corretta a 3 cifre decimali. Analogamente, con `Tol`$= 10^{-4}$ si otterrebbe, dopo 21 iterazioni, un'accuratezza fino a 4 cifre decimali.

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|x^{(k)} - x^*\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	9.05555	98.91611	199.04701	$.83887\times10^{-1}$
11	9.03272	98.95727	198.93482	$.65183\times10^{-1}$
12	8.99439	98.99340	198.96606	$.33941\times10^{-1}$
13	8.99316	99.01016	198.99654	$.10156\times10^{-1}$
14	8.99660	99.00479	199.00791	$.79100\times10^{-2}$
15	9.00078	99.00046	199.00376	$.37636\times10^{-2}$
16	9.00083	98.99878	199.00018	$.12201\times10^{-2}$
17	9.00035	98.99947	198.99905	$.95100\times10^{-3}$
18	8.99989	98.99998	198.99959	$.41216\times10^{-3}$
19	8.99990	99.00015	199.00000	$.14545\times10^{-3}$
20	8.99996	99.00006	199.00011	$.11333\times10^{-3}$
21	9.00001	99.00000	199.00004	$.44536\times10^{-4}$
22	9.00001	98.99998	199.00000	$.17207\times10^{-4}$
23	9.00000	98.99999	198.99999	$.13391\times10^{-4}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

♣

Come mostra l'esempio 4.11, se nella relazione (4.36) si pone `Tol`$= 10^{-m}$, si richiede che il metodo iterativo si arresti quando è stata ottenuta un'approssimazione della soluzione corretta a m cifre decimali. Tuttavia, l'utilizzo della (4.36) presuppone, come criterio di arresto, la conoscenza della soluzione del sistema, che non è nota a priori. È necessario, allora, determinare stime calcolabili dell'errore di troncamento analitico ad ogni iterazione.

Teorema 4.4.1. *Dato il metodo iterativo:*

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i, \quad i = 1, \ldots, n, \;\; k \geq 0,$$

si assuma che sia consistente con il sistema lineare $Ax = b$. Se $\|C\| < 1$ si ha:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|} \|x^{(k+1)} - x^{(k)}\|, \;\; k \geq 0. \tag{4.37}$$

Dimostrazione Poichè il metodo è consistente, se $\|C\| < 1$ esso converge alla soluzione del sistema, x^*. Valgono quindi le (4.31), con x_i^* al posto di $\bar{x}_i$, che possono essere riscritte nel modo seguente:

$$x_i^{(k+1)} - x_i^* = \sum_{j=1}^{n} c_{ij}(x_j^{(k+1)} - x_j^*) - \sum_{j=1}^{n} c_{ij}(x_j^{(k+1)} - x_j^{(k)}) \,, \;\; i = 1, \ldots, n,$$

da cui

$$|x_i^{(k+1)} - x_i^*| \leq \sum_{j=1}^{n} |c_{ij}||x_j^{(k+1)} - x_j^*| + \sum_{j=1}^{n} |c_{ij}||x_j^{(k+1)} - x_j^{(k)}| \,, \;\; i = 1, \ldots, n,$$

e quindi:

$$\|x^{(k+1)} - x^*\| \leq \|C\|\|x^{(k+1)} - x^*\| + \|C\|\|x^{(k+1)} - x^{(k)}\|.$$

Poichè $\|C\| < 1$ per ipotesi, si ottiene:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|}\|x^{(k+1)} - x^{(k)}\|.$$

■

♣ **Esempio 4.12.** Relativamente alla risoluzione del sistema dell'esempio 4.11, nella Tabella che segue è riportato, per ciascuna iterazione, sia l'errore assoluto, sia la norma del massimo della differenza tra il vettore calcolato in quella iterazione ed il vettore ottenuto nell'iterazione precedente. Si osserva che tale valore costituisce una buona stima dell'errore assoluto.

k	$\|x^{(k)} - x^*\|$	$\|x^{(k)} - x^{(k-1)}\|$
⋮	⋮	⋮
10	$.83887\times10^{-1}$	$.25544\times10^{0}$
11	$.65183\times10^{-1}$	$.11220\times10^{0}$
12	$.33941\times10^{-1}$	$.38338\times10^{-1}$
13	$.10156\times10^{-1}$	$.30478\times10^{-1}$
14	$.79100\times10^{-2}$	$.11373\times10^{-1}$
15	$.37636\times10^{-2}$	$.43332\times10^{-2}$
16	$.12201\times10^{-2}$	$.35861\times10^{-2}$
17	$.95100\times10^{-3}$	$.11285\times10^{-2}$
18	$.41216\times10^{-3}$	$.53885\times10^{-3}$
19	$.14545\times10^{-3}$	$.41671\times10^{-3}$
20	$.11333\times10^{-3}$	$.10878\times10^{-3}$
21	$.44536\times10^{-4}$	$.68793\times10^{-4}$
22	$.17207\times10^{-4}$	$.47863\times10^{-4}$
23	$.13391\times10^{-4}$	$.10962\times10^{-4}$
⋮	⋮	⋮

♣

Il Teorema e l'esempio precedenti mostrano, quindi, che una stima dell'errore di troncamento analitico è data dalla differenza tra i vettori ottenuti in due iterazioni successive del metodo iterativo. Di conseguenza un possibile criterio di arresto "effettivamente utilizzabile", perchè basato su quantità calcolate, è il seguente:

$$\|x^{(k+1)} - x^{(k)}\| \leq \texttt{Tol}. \tag{4.38}$$

Inoltre, in base al risultato stabilito dal Teorema 4.4.1, se si arresta il metodo iterativo quando è verificata la condizione (4.38), per l'errore assoluto di troncamento analitico si ha la maggiorazione:

$$\|x^{(k+1)} - x^*\| \leq \frac{\|C\|}{1 - \|C\|}\texttt{Tol}. \tag{4.39}$$

Se, in particolare, si desidera stimare l'errore relativo di troncamento analitico:

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^*\|},$$

basta osservare che, dalla (4.37), segue:

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^*\|} \leq \frac{\|C\|}{1 - \|C\|}\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^*\|}.$$

Poichè la soluzione x^* non è nota, nella pratica si sostituisce ad essa il valore corrente $x^{(k+1)}$, ottenendo così il seguente criterio di arresto basato sulla distanza relativa tra i valori ottenuti in due iterazioni successive:

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|} \leq \texttt{Tol}. \tag{4.40}$$

♣ **Esempio 4.13.** Si consideri ancora la risoluzione del sistema dell'esempio 4.11. Nella Tabella seguente sono riportati i valori dell'errore relativo e della distanza relativa tra i valori ottenuti in due iterazioni successive del metodo di Jacobi. Si osserva, inanzitutto, che l'errore relativo fornisce informazioni sull'accuratezza in termini di cifre significative. Inoltre la quantità $\|x^{(k)} - x^{(k-1)}\|/\|x^{(k)}\|$ costituisce una stima affidabile dell'errore relativo.

k	$\frac{\|x^{(k)}-x^*\|}{\|x^*\|}$	$\frac{\|x^{(k)}-x^{(k-1)}\|}{\|x^{(k)}\|}$
⋮	⋮	⋮
10	$.42154\times10^{-3}$	$.12833\times10^{-2}$
11	$.32755\times10^{-3}$	$.56398\times10^{-3}$
12	$.17056\times10^{-3}$	$.19269\times10^{-3}$
13	$.51035\times10^{-4}$	$.15316\times10^{-3}$
14	$.39749\times10^{-4}$	$.57151\times10^{-4}$
15	$.18912\times10^{-4}$	$.21774\times10^{-4}$
16	$.61311\times10^{-5}$	$.18021\times10^{-4}$
17	$.47789\times10^{-5}$	$.56707\times10^{-5}$
18	$.20711\times10^{-5}$	$.27078\times10^{-5}$
19	$.73090\times10^{-6}$	$.20940\times10^{-5}$
20	$.56949\times10^{-6}$	$.54663\times10^{-6}$
21	$.22380\times10^{-6}$	$.34569\times10^{-6}$
22	$.86467\times10^{-7}$	$.24052\times10^{-6}$
23	$.67290\times10^{-7}$	$.55086\times10^{-7}$
⋮	⋮	⋮

♣

Il costo computazionale aggiuntivo, relativo all'utilizzo dei criteri di arresto (4.38) e (4.40), è solo quello del calcolo delle quantità $\|x^{(k+1)} - x^{(k)}\|$ e $\|x^{(k+1)}\|$, in quanto i vettori che si considerano sono già calcolati durante il processo iterativo.

Nella prima versione degli algoritmi relativi ai metodi iterativi di Jacobi e G-S (**Procedure 4.1** e **4.2**) l'unica condizione per l'arresto del processo iterativo è il valore di `MaxIt`, che rappresenta il numero di iterazioni da eseguire. Una seconda versione di tali algoritmi, che include sia un criterio di arresto basato sulla relazione (4.40) sia una condizione di arresto basata su un numero massimo di iterazioni (`MaxIt`), è riportata nelle **Procedure 4.3 e 4.4**. Tali procedure forniscono, in output, anche una stima (`Err`) dell'errore relativo, cioè la quantità:

$$\frac{\|x^{(k+1)} - x^{(k)}\|}{\|x^{(k+1)}\|},$$

dove $x^{(k+1)}$ è l'ultimo valore calcolato prima dell'arresto del processo iterativo. Inoltre, altri due parametri di output delle procedure sono: la variabile `Iflag`, che segnala se è stata raggiunta l'accuratezza desiderata e la variabile `It`, che indica il numero di iterazioni che sono state eseguite.

procedure Jacobi2(**in:** $n, A, b, xold, MaxIt, Tol$; **out:** $x, Err, Iflag, It$)

/# SCOPO: risoluzione di un sistema di equazioni lineari
con il metodo iterativo di Jacobi

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: n	: **intero**	{*dimensione del sistema*}
var: $A(n,n)$	: **array di reali**	{*coefficienti del sistema*}
var: $b(n)$	: **array di reali**	{*termini noti del sistema*}
var: $xold(n)$	: **array di reali**	{*valori iniziali*}
var: $MaxIt$	: **intero**	{*numero massimo di iterazioni*}
var: Tol	: **reale**	{*tolleranza usata nel criterio di arresto*}

/# PARAMETRI DI OUTPUT:

var: $x(n)$	: **array di reali**	{*soluzione calcolata*}
var: Err	: **reale**	{*stima dell'errore*}
var: $Iflag$	: **intero**	{*= 0 se la tolleranza non è soddisfatta*}
		{*= 1 se la tolleranza è soddisfatta*}
var: It	: **intero**	{*numero di iterazioni eseguite*}

/# VARIABILI LOCALI:

var: i, j	: **interi**	{*contatori*}
var: $xnorm$	: **reale**	{*norma del vettore all'ultimo passo*}
var: $dnorm$	: **reale**	{*norma della diff. tra i vettori ottenuti*}
		{*in due iterazioni successive*}

/# INIZIO ISTRUZIONI:

$It := 0;$ {*inizializzazione di* `It`}
$Iflag := 0;$ {*inizializzazione di* `Iflag`}
repeat {*ciclo principale con*}
  $xnorm := 0.;$ {*condizioni di arresto*}
  $dnorm := 0.;$

Procedura 4.3: Algoritmo di Jacobi (seconda versione) - continua

```
        for i = 1, n do                                  {ciclo per il calcolo di x^(k)}
          x(i) := b(i);
          for j = 1, n do                                {ciclo per il calcolo di x_i^(k)}
            if i ≠ j then
              x(i) := x(i) − A(i, j) ∗ xold(j);
            endif
          endfor
          x(i) := x(i)/A(i, i);
          if |x(i)| > xnorm then                         {calcolo di ||x^(k+1)||}
            xnorm := |x(i)|;
          endif
          if |x(i) − xold(i)| > dnorm then               {calcolo di ||x^(k+1) − x^(k)||}
            dnorm := |x(i) − xold(i)|;
          endif
        endfor
        Err := dnorm/xnorm;                              {stima dell'errore}
        if Err ≤ Tol then                                {controllo condizione di arresto}
          Iflag := 1;
        endif
        It := It + 1;                                    {aggiornamento del contatore It }
        for i = 1, n do                                  {aggiornamento di xold}
          xold(i) := x(i);
        endfor
      until (It = MaxIt or Iflag = 1)                    {verifica condizioni di arresto}
      return x, Err, Iflag, It
end Jacobi2
```

Procedura 4.3: Algoritmo di Jacobi (seconda versione) - fine

procedure G-S2(**in:** $n, A, b, xold, MaxIt, Tol$; **out:** $x, Err, Iflag, It$)

/# SCOPO: risoluzione di un sistema di equazioni lineari
con il metodo iterativo di Gauss-Seidel

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: n : **intero** {*dimensione del sistema*}
var: $A(n,n)$: **array di reali** {*coefficienti del sistema*}
var: $b(n)$: **array di reali** {*termini noti del sistema*}
var: $xold(n)$: **array di reali** {*valori iniziali*}
var: $MaxIt$: **intero** {*numero massimo di iterazioni*}
var: Tol : **reale** {*tolleranza usata nel criterio di arresto*}

/# PARAMETRI DI OUTPUT:

var: $x(n)$: **array di reali** {*soluzione calcolata*}
var: Err : **reale** {*stima dell'errore*}
var: $Iflag$: **intero** {*= 0 se la tolleranza non è soddisfatta*}
{*= 1 se la tolleranza è soddisfatta*}
var: It : **intero** {*numero di iterazioni eseguite*}

/# VARIABILI LOCALI:

var: i, j : **interi** {*contatori*}
var: $xnorm$: **reale** {*norma del vettore all'ultimo passo*}
var: $dnorm$: **reale** {*norma della diff. tra i vettori ottenuti*}
{*in due iterazioni successive*}

/# INIZIO ISTRUZIONI:

$It := 0;$ {*inizializzazione di* `It`}
$Iflag := 0;$ {*inizializzazione di* `Iflag`}
for $i = 1, n$ **do** {*inizializzazione di* x}
$x(i) := xold(i);$
endfor

Procedura 4.4: Algoritmo di Gauss-Seidel (seconda versione) - continua

```
    repeat                                          {ciclo principale con}
      xnorm := 0.;                                  {condizioni di arresto}
      dnorm := 0.;
      for i = 1, n do                               {ciclo per il calcolo di x^(k)}
        x(i) := b(i);
        for j = 1, n do                             {ciclo per il calcolo di x_i^(k)}
          if i ≠ j then
            x(i) := x(i) − A(i, j) ∗ x(j);
          endif
        endfor
        x(i) := x(i)/A(i, i);
        if |x(i)| > xnorm then                      {calcolo di ||x^(k+1)||}
          xnorm := |x(i)|;
        endif
        if |x(i) − xold(i)| > dnorm then            {calcolo di ||x^(k+1) − x^(k)||}
          dnorm := |x(i) − xold(i)|;
        endif
      endfor
      Err := dnorm/xnorm;                           {stima dell'errore}
      if Err ≤ Tol then                             {controllo condizione di arresto}
        Iflag := 1;
      endif
      It := It + 1;                                 {aggiornamento del contatore It }
      for i = 1, n do                               {aggiornamento di xold}
        xold(i) := x(i);
      endfor
    until (It = MaxIt or Iflag = 1)                 {verifica condizioni di arresto}
    return x, Err, Iflag, It
end G-S2
```

Procedura 4.4: Algoritmo di Gauss-Seidel (seconda versione) - fine

4.5 Un esempio di software matematico per i metodi iterativi

A partire dalle procedure descritte nel precedente paragrafo, si possono costruire elementi di software matematico, per la risoluzione di sistemi di equazioni lineari mediante i metodi di Jacobi e Gauss-Seidel. Uno degli scopi di tale software è rendere trasparente, a chi le utilizza, i dettagli dell'algoritmo su cui si basano e di fornire la soluzione a partire da un insieme minimo di dati di input relativi al sistema da risolvere. Se, ad esempio, si utilizza il linguaggio di programmazione FORTRAN, le testate delle subroutine relative alle procedure `Jacobi2` e `G-S2` possono essere le seguenti:

```
SUBROUTINE JACOBI2(N,A,LDA,B,XOLD,MAXIT,TOL,X,ERR,IFLAG,IT)

SUBROUTINE G-S2(N,A,LDA,B,XOLD,MAXIT,TOL,X,ERR,IFLAG,IT)
```

Di seguito è illustrato un esempio di programma chiamante la subroutine `JACOBI2`, per risolvere il sistema lineare:

$$\begin{cases} 3x_1 & +0.5x_2 & +0.3x_3 & = 3.8 \\ 0.5x_1 & + \ 2x_2 & +0.9x_3 & = 3.4 \\ -0.1x_1 & +0.6x_2 & + \ x_3 & = 1.5, \end{cases}$$

la cui soluzione è $x^* = (1, 1, 1)$, con `MAXIT`=50, `TOL`=0.000001 e $x_i^{(0)} = 0.$, $i = 1, 2, 3$. In tale esempio si è supposto che i dati relativi alla matrice dei coefficienti ed al termine noto del sistema da risolvere, siano forniti da una subroutine (`SISTEMA`), che deve essere sviluppata dall'utente. Se si vuole risolvere un sistema lineare diverso è sufficiente modificare solo tale subroutine.

```
      PROGRAM JACOBI
C
C dichiarazione delle variabili
C
      REAL A(10,10), B(50), XOLD(10), X(10), TOL, ERR
      INTEGER N, LDA, MAXIT, CONV, K, IT, IFLAG, I
C
C inizializzazione delle variabili di input
C
      N=3
      LDA=10
      MAXIT=50
      TOL=0.000001
      DO 10 I=1,N
           XOLD(I)=0.0
 10    CONTINUE
C
C chiamata di SISTEMA per generare A e b
C
      CALL SISTEMA(N,A,LDA,B)
C
C chiamata di JACOBI2
C
      CALL JACOBI2(N,A,LDA,B,XOLD,MAXIT,TOL,
     * X,ERR,IFLAG,IT)
C
C stampa dei risultati
C
      IF (IFLAG .EQ. 1) THEN
           WRITE(*,*) Soluzione        =,(X(I),I=1,N)
           WRITE(*,*) Errore           =,ERR
           WRITE(*,*) Numero Iter.     =,IT
           WRITE(*,*) Iflag            =,IFLAG
      ELSE
```

Esempio di programma chiamante FORTRAN per la subroutine JACOBI2 - continua

```
            WRITE(*,*) Tolleranza non
           * raggiunta
            ENDIF
            STOP
            END



C
C subroutine che genera la matrice
C ed il vettore dei termini noti
C
            SUBROUTINE SISTEMA(N,A,LDA,B)
C
C dichiarazione dei parametri
C
            REAL A(LDA,N), B(N)
            INTEGER N, LDA
C
C definizione di A e di B
C
            A(1,1)=3.
            A(1,2)=0.5
            A(1,3)=0.3
            A(2,1)=0.5
            A(2,2)=2.
            A(2,3)=0.9
            A(3,1)=-0.1
            A(3,2)=0.6
            A(3,3)=1.

            B(1)=3.8
            B(2)=3.4
            B(3)=1.5

            RETURN
            END
```

Esempio di programma chiamante FORTRAN per la subroutine JACOBI2 - fine

Sistema (a)	IT	x_1	x_2	x_3	ERR
Jacobi	48	5.199987	6.799983	3.399991	$0.98{\cdot}10^{-6}$
G-S	26	5.199993	6.799994	3.399997	$0.70{\cdot}10^{-6}$

Tabella 4.8: Schema riassuntivo relativo alla risoluzione del sistema (a) con i metodi di Jacobi e G-S.

L'esecuzione del programma che implementa la versione in singola precisione della subroutine `JACOBI2`, con l'esempio di programma chiamante appena illustrato, fornisce i risultati seguenti:

```
Soluzione          = .9999998 .9999997 .9999995
Errore             = 9.53675E-07
Numero iter.       = 26
Iflag              = 1
```

♣ **Esempio 4.14.** Si risolvano i seguenti sistemi lineari:

$$(a)\ \begin{cases} 3x_1 - 2x_2 = 2 \\ -x_1 + 2x_2 - x_3 = 5 \\ -x_2 + 2x_3 = 0 \end{cases} \qquad (b)\ \begin{cases} 4x_1 + 2x_2 + x_3 = 1 \\ x_1 + 3x_2 + x_3 = 0 \\ 2x_2 - 3x_3 = 1 \end{cases}$$

$x_1^* = 5.2,\ x_2^* = 6.8,\ x_3^* = 3.4$ $\qquad$ $x_1^* = 1/3,\ x_2^* = 0,\ x_3^* = -1/3$

$$(c)\ \begin{cases} 4x_1 - 2x_2 = 0 \\ -2x_1 + 4x_2 - 2x_3 = 2 \\ -2x_2 + 4x_3 = 0 \end{cases}$$

$x_1^* = 0.5,\ x_2^* = 1,\ x_3^* = 0.5$

con i metodi di Jacobi e di Gauss-Seidel.
Nelle Tabelle 4.8, 4.9, 4.10 sono riportati i risultati ottenuti utilizzando `MAXIT`=50, `TOL`=10^{-6} e come valori iniziali $x_i^{(0)} = 0$, $i = 1, 2, 3$.
Si osservi che `IT` indica il numero di iterazioni eseguite per soddisfare il criterio di arresto, x_1, x_2, x_3 sono gli ultimi valori calcolati prima dell'arresto del processo iterativo, `ERR` rappresenta la stima finale dell'errore relativo; la soluzione ottenuta è un'approssimazione della soluzione accurata a 7 cifre significative. Infine, dai valori di `IT` si evince che, per i sistemi in esame, il metodo di Gauss-Seidel converge alla soluzione più velocemente rispetto al metodo di Jacobi.

♣

Sistema (b)	IT	x_1	x_2	x_3	ERR
Jacobi	16	0.333333	0.000002	-0.333333	$0.60 \cdot 10^{-6}$
G-S	4	0.333333	0.000000	-0.333333	$0.10 \cdot 10^{-6}$

Tabella 4.9: Schema riassuntivo relativo alla risoluzione del sistema (b) con i metodi di Jacobi e G-S.

Sistema (c)	IT	x_1	x_2	x_3	ERR
Jacobi	38	0.499999	0.999998	0.499999	$0.95 \cdot 10^{-6}$
G-S	20	0.499999	0.999999	0.499999	$0.95 \cdot 10^{-6}$

Tabella 4.10: Schema riassuntivo relativo alla risoluzione del sistema (c) con i metodi di Jacobi e G-S.

4.6 Efficienza

Nel paragrafo 4.2 si è analizzata la complessità di tempo di una iterazione dei metodi di Jacobi e di G-S e si è visto che il numero di flops è circa uguale al numero di coefficienti non nulli del sistema. Il risultato di tale analisi si estende evidentemente ad un generico metodo iterativo della forma:

$$x_i^{(k+1)} = \sum_{j=1}^{n} c_{ij} x_j^{(k)} + d_i, \quad i = 1, \ldots, n, \quad k \geq 0, \tag{4.41}$$

per cui, indicato con $T_{it}(n)$ il relativo numero di flops, si ha:

$$T_{it}(n) = \mathcal{O}(n^2(1 - SP)),$$

con SP grado di sparsità del sistema. Da ciò discende che la complessità di tempo totale è data da:

$$T(n) = k \cdot T_{it}(n) = \mathcal{O}(kn^2(1 - SP)), \tag{4.42}$$

dove k è il numero di iterazioni che sono eseguite. Dalla (4.42) si evince, pertanto, che uno dei fattori da cui dipende l'efficienza di un metodo iterativo è il numero di iterazioni che sono eseguite e, quindi, la **velocità** con la quale la successione, generata dal metodo, converge alla soluzione del sistema.
Confrontando tale quantità con la complessità di tempo del metodo di eliminazione di Gauss, $T_{gauss} = \mathcal{O}(n^3/3)$, si evince che risulta più efficiente utilizzare il metodo di Jacobi o il metodo di G-S, e cioè $T_{it}^{tot} < T_{gauss}$, se si effettua un numero di iterazioni k tale che

$$k < \frac{n}{3(1 - SP)}. \tag{4.43}$$

4.6.1 Velocità di convergenza

Se si assume che il metodo iterativo (4.41) sia consistente e che sia $\|C\| < 1$, esso è convergente a x^*, dove x^* è la soluzione del sistema $Ax = b$.
Assunto, $e^{(k)} = x^{(k)} - x^*$, l'errore alla k-ma iterazione, il punto di partenza per ottenere una stima della velocità di convergenza del metodo iterativo è considerare la disuguaglianza:

$$\|e^{(k)}\| \leq \|C\|^k \|e^{(0)}\|, \tag{4.44}$$

che si ottiene dalla (4.34), dimostrata nel Teorema 4.3.1. La (4.44) mette in relazione l'errore al passo k con l'errore iniziale e mostra che quanto più piccola è la quantità $\|C\|$ tanto più rapidamente converge a zero $\|e^{(k)}\|$ e quindi tanto più basso sarà il numero di iterazioni necessario per rendere la norma dell'errore minore di una prefissata tolleranza.

♣ **Esempio 4.15.** Se $\|C\| = \frac{1}{2}$ e $\|e^{(0)}\| = 1$, dalla (4.44) si ha $\|e^{(k)}\| \leq (\frac{1}{2})^k$; in tal caso, l'errore sarà minore di 10^{-7} dopo 24 iterazioni (infatti $k = 24$ è il più piccolo numero intero positivo per cui $(\frac{1}{2})^k \leq 10^{-7}$). D'altra parte, se $\|C\| = 0.8$, sono necessarie 73 iterazioni per avere $\|e^{(k)}\| \leq 10^{-7}$. ♣

Se le due quantità, $\|e^{(0)}\|$ e $\|C\|$, sono note (o possono essere facilmente calcolate), utilizzando la (4.44) è possibile ottenere una stima del minimo numero di iterazioni necessarie per rendere la norma dell'errore minore di una quantità `Tol` fissata. Infatti, si deve avere che:

$$\|C\|^k \|e^{(0)}\| \leq \texttt{Tol}. \tag{4.45}$$

Applicando la funzione logaritmo naturale ad entrambi i membri della disuguaglianza (4.45) si ha:

$$k \cdot \log(\|C\|) + \log(\|e^{(0)}\|) \leq \log(\texttt{Tol}). \tag{4.46}$$

Poichè $\|C\| < 1$ per ipotesi, si ha $\log(\|C\|) < 0$ e, quindi, dalla (4.46) si ricava che una stima del numero minimo di iterazioni per ottenere la stabilita riduzione dell'errore è:

$$k \geq \frac{\log(\texttt{Tol}) - \log(\|e^{(0)}\|)}{\log(\|C\|)}. \tag{4.47}$$

Osservazione 4.2. *Esistono vari problemi legati all'utilizzo della stima (4.47). Il primo è che, non essendo nota la soluzione del sistema, la quantità* $\log(\|e^{(0)}\|)$ *non è calcolabile. Un altro problema è dovuto al fatto che la stima (4.47) dipende dalla norma che si utilizza, come mostra l'esempio 4.15. A conferma di ciò si consideri il sistema lineare:*

$$\begin{cases} x_1 + 0.4x_2 - 0.1x_3 = 1 \\ 0.3x_1 + x_2 - 0.001x_3 = -2.5 \\ 0.1x_1 + 0.4x_2 + x_3 = 3.0, \end{cases}$$

la cui soluzione è $x_1^* = 2.7315$, $x_2^* = -3.3154$, $x_3^* = 4.0530$. *Si verifica facilmente che, applicando il metodo di Jacobi a tale sistema, partendo da* $x_i^{(0)} = 0$, $i = 1, 2, 3$, *si ha* $\|e^{(0)}\|_\infty = 4.0530$ *e* $\|C\|_\infty = 0.5$, *mentre* $\|e^{(0)}\|_1 = 10.0999$ *e* $\|C\|_1 = 0.8$. *Nel primo caso la (4.47) mostra che sono necessarie circa 25 iterazioni affinchè l'errore sia minore di* 10^{-7}, *mentre nel secondo caso ne sono necessarie circa 82. Anche da tale esempio si evince, quindi, che il numero di iterazioni, che si stima essere necessario per rendere l'errore minore di* `Tol`, *dipende dalla norma che si utilizza per misurare l'ordine di grandezza della matrice* C [12].

4.6.2 Un algoritmo per la memorizzazione dei coefficienti di una matrice ad elevato grado di sparsità

La risoluzione di un sistema lineare di grandi dimensioni comporta, in generale, un'elevata complessità di spazio, oltre che di tempo. Se il sistema è sparso è possibile utilizzare opportune tecniche per memorizzare in forma compatta solo i coefficienti diversi da zero e ciò può consentire una risoluzione più efficiente.

Uno dei più noti schemi per la memorizzazione di una matrice sparsa è il cosiddetto **schema dei tre vettori**. La matrice $A \in \Re^{n\times n}$ è rappresentata utilizzando tre vettori, R, C, J. Il vettore R contiene gli elementi della matrice diversi da zero, riga per riga. L'indice di colonna in A dell'elemento $R(k)$ è dato da $C(k)$, mentre $J(i)$, $i = 1, \ldots, n$, fornisce la posizione, nell'array R, del primo elemento diverso da zero della i-ma riga.

♣ **Esempio 4.16.** Utilizzando lo schema dei tre vettori, la matrice

$$A = \begin{pmatrix} 7 & 0 & -3 & 0 & -1 & 0 \\ 2 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -2 & 0 & 6 \end{pmatrix}$$

è memorizzata nel modo seguente:

$$R = (7, -3, -1, 2, 8, 1, -3, 5, -1, 4, -2, 6);$$

$$C = (1, 3, 5, 1, 2, 3, 1, 4, 2, 5, 4, 6);$$

$$J = (1, 4, 6, 7, 9, 11, 13).$$

Si noti che l'ultimo elemento di J è uguale al numero totale degli elementi di R più uno.

♣

[12] Nei paragrafi successivi si vedrà che esiste un'altra stima della velocità di convergenza, che a differenza della (4.47) non dipende dalla norma utilizzata e quindi risulta ovviamente più affidabile.

Nel caso in cui la matrice è simmetrica, è possibile utilizzare uno schema di memorizzazione più efficiente di quello appena illustrato, lo **schema del profilo**. Si fa uso di due array, R e M: il primo contiene, per ogni riga della matrice A, gli elementi compresi tra il primo non nullo su quella riga e l'elemento diagonale. Il secondo array, $M(i)$, $i = 1, \ldots, n$, contiene la posizione, in R, dell'elemento diagonale della i-ma riga.

♣ **Esempio 4.17.** Secondo lo schema del profilo, la matrice

$$A = \begin{pmatrix} 25 & 3 & 0 & 0 & 0 \\ 3 & 21 & 2 & 4 & 0 \\ 0 & 2 & 23 & 0 & 0 \\ 0 & 4 & 0 & 22 & 1 \\ 0 & 0 & 0 & 1 & 20 \end{pmatrix}$$

è memorizzata mediante

$$R = (25, 3, 21, 2, 23, 4, 0, 22, 1, 20);$$
$$M = (1, 3, 5, 8, 10).$$

♣

Utilizzando tale schema, si osserva che il generico elemento a_{ij} di A è memorizzato in R se $i - j < M(i) - M(i-1)$. Inoltre $a_{ij} = R(p)$, con $p = M(i) - i + j$.

Negli esempi precedenti, date le dimensioni della matrice A, non appare evidente il vantaggio nell'utilizzo degli schemi descritti rispetto alle tecniche di memorizzazione usuali (**array bidimensionale**); il vantaggio è invece evidente per matrici di grandi dimensioni.

♣ **Esempio 4.18.** Si consideri una matrice quadrata A di dimensione $n = 100$ con $s = 400$ elementi diversi da zero. Se si usa per la sua memorizzazione l'usuale array bidimensionale, la complessità di spazio è $n \times n = 10000$, mentre se si sceglie lo schema dei tre vettori la complessità si riduce a $s + s + n + 1 = 901$, con un risparmio, quindi, di circa il 91%. ♣

Se k è il numero di iterazioni eseguite, la complessità di tempo totale è:

$$T_{it}^{tot}(n) = \mathcal{O}(k \cdot n^2(1 - SP)).$$

Nella **Procedura 4.5** è illustrata una nuova versione dell'algorimo relativo al metodo di Jacobi, in cui si assume che la matrice dei coefficienti del sistema sia memorizzata con lo schema dei tre vettori. Nella notazione utilizzata, i tre vettori R, C, J, sono rappresentati rispettivamente dagli array `r`, `c`, `ia`.

procedure Jacobi3(**in:** $n, s, r, c, ia, b, xold, MaxIt, Tol$; **out:** $x, Err, Iflag, It$)

/# SCOPO: risoluzione di un sistema di equazioni lineari
con il metodo iterativo di Jacobi;
la matrice è memorizzata con lo schema dei tre vettori

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: n	: **intero**	{*dimensione del sistema*}
var: s	: **intero**	{*numero coefficienti non nulli*}
var: $r(s)$	: **array di reali**	{*coefficienti non nulli riga per riga*}
var: $c(s)$	: **array di interi**	{*indici di colonna degli elementi di r*}
var: $ia(n)$	: **array di interi**	{*ia(i) = posizione in r del primo*} {*elemento non nullo della i-ma riga*}
var: $b(n)$	: **array di reali**	{*termini noti del sistema*}
var: $xold(n)$	: **array di reali**	{*valori iniziali*}
var: $MaxIt$	: **intero**	{*numero massimo di iterazioni*}
var: Tol	: **reale**	{*tolleranza usata nel criterio di arresto*}

Procedura 4.5: Algoritmo di Jacobi (terza versione) - continua

```
/# PARAMETRI DI OUTPUT:
  var: x(n)     : array di reali  {soluzione calcolata}
  var: Err      : reale           {stima dell'errore}
  var: Iflag    : intero          {= 0 se la tolleranza non è soddisfatta}
                                  {= 1 se la tolleranza è soddisfatta}
  var: It       : intero          {numero di iterazioni eseguite}
/# VARIABILI LOCALI:
  var: i, j, l  : interi          {contatori}
  var: diff     : intero          {numero degli elementi non nulli di}
                                  {ciascuna riga della matrice dei coefficienti}
  var: xnorm    : reale           {norma del vettore calcolato all'ultimo passo}
  var: dnorm    : reale           {norma della differenza tra i vettori ottenuti}
                                  {in due iterazioni successive}
  var: d        : reale           {elemento diagonale, riga per riga}
/# INIZIO ISTRUZIONI:
    It := 0;                      {inizializzazione di It ed Iflag}
    Iflag := 0;
    repeat                                  {ciclo principale con}
      l := 0;                               {condizioni di arresto}
      xnorm := 0.;
      dnorm := 0.;
      for i = 1, n do                       {ciclo per il calcolo di x^(k)}
        x(i) := b(i);
        diff := ia(i + 1) − ia(i);
        for j = 1, diff do                  {ciclo per il calcolo di x_i^(k)}
```

Procedura 4.5: Algoritmo di Jacobi (terza versione) - continua

```
            l := l + 1;
            if c(l) ≠ i then
               x(i) := x(i) − r(l) ∗ xold(c(l));
            else
               d := r(l);
            endif
         endfor
         x(i) := x(i)/d;
         if |x(i)| > xnorm then                    {calcolo di ‖x^(k+1)‖}
            xnorm := |x(i)|;
         endif
         if |x(i) − xold(i)| > dnorm then          {calcolo di ‖x^(k+1) − x^(k)‖}
            dnorm := |x(i) − xold(i)|;
         endif
      endfor
      Err := dnorm/xnorm;                          {calcolo della stima dell'errore}
      if Err ≤ Tol then                            {controllo condizione di arresto}
         Iflag := 1;
      endif
      It := It + 1;                                {aggiornamento del contatore It}
      for i = 1, n do                              {aggiornamento di xold}
         xold(i) := x(i);
      endfor
   until (It = MaxItorIflag = 1)                   {verifica condizioni di arresto}
   return x, Err, Iflag, It
end Jacobi3
```

Procedura 4.5: Algoritmo di Jacobi (terza versione) - fine

4.7 Un esempio di programma MATLAB per i metodi iterativi

In questo paragrafo si descrive l'utilizzo del software MATLAB per lo sviluppo di un programma che implementi la **Procedura 4.3** relativa al metodo di Jacobi.

Allo stato attuale non sono presenti funzioni per la risoluzione di sistemi di equazioni lineari con metodi iterativi. Tuttavia, MATLAB fornisce la possibilità di utilizzare un insieme di istruzioni, come un linguaggio di programmazione, per lo sviluppo di funzioni che risolvono problemi specifici.

Un esempio di programma MATLAB per la risoluzione di sistemi lineari con il metodo di Jacobi è riportato nella Funzione 1. Tale funzione ha 4 parametri di output, cioè quelli racchiusi tra parentesi quadre dopo la parola chiave **function**. La funzione ha 5 parametri di input, quelli specificati nelle parentesi tonde dopo il nome della funzione. Il simbolo % si utilizza per inserire i commenti. La funzione **norm(x,inf)** calcola la norma del massimo del vettore `x`. La funzione **length(b)** restituisce, in output, la lunghezza del vettore `b`. Segue un esempio di utilizzo della funzione in esame.

♣ **Esempio 4.19.** Si risolva il sistema lineare:

$$\begin{cases} 3x_1 & +0.5x_2 & +0.3x_3 & = 3.8 \\ 0.5x_1 & + 2x_2 & +0.9x_3 & = 3.4 \\ -0.1x_1 & +0.6x_2 & + x_3 & = 1.5, \end{cases}$$

la cui soluzione è $x^* = (1, 1, 1)$, uilizzando la Funzione 1.
Le seguenti istruzioni definiscono le variabili che servono come parametri di input per la funzione.

```
>> A=[3 0.5 0.3; 0.5 2 0.9; -0.1 0.6 1];    b=[3.8; 3.4; 1.5];    x=[0; 0; 0;];
>> Tol=0.000001;    MaxIt=50;
```

L'istruzione che segue chiama la funzione.

```
>> [x,ERR,IFLAG,IT] = Jacobi(A,b,x,Tol,MaxIt)
```

L'output è:

```
x =
      0.99999984417416
      0.99999968961009
      0.99999966015567

ERR =
      9.221026131790661e-07

IFLAG =
      1

IT =
      26
```

♣

```
function [x,ERR,IFLAG,IT] = Jacobi(A,b,x,Tol,MaxIt);
%
% ---------------------------------------------------------------------
%
%       SCOPO
%       =====
%       Calcola la soluzione di un sistema di equazioni lineari
%       mediante il metodo iterativo di Jacobi.
%
%
%       PARAMETRI DI INPUT
%       ==================
%
%       A - Matrice dei coefficienti del sistema.
%
%       b - Vettore dei termini noti.
%
%       x - Valori iniziali.
%
%       Tol - Tolleranza per il criterio di arresto.
%
%       MaxIt - Numero massimo di iterazioni.
%
%
%       PARAMETRI DI OUTPUT
%       ===================
%
%       x - Soluzione.
%
%       ERR - Stima dell'errore relativo.
%
%       IFLAG - Vale 1 se si è raggiunta la tolleranza,
%                    0 se si è raggiunto il numero massimo di iterazioni.
%
%       IT - Numero di iterazioni eseguite.
%
%
% ---------------------------------------------------------------------
%
%
```

Funzione 1:
Funzione MATLAB per la risoluzione di un sistema di equazioni lineari con il metodo di Jacobi - continua

```
n=length(b);                                % lunghezza di b =
                                            % = ordine del sistema
xold=x;                                     % vettore di appoggio
ERR=1.0;                                    % inizializzazione della stima dell'errore
IFLAG=0;                                    % inizializzazione di IFLAG
IT=0;                                       % inizializzazione del numero di iterazioni

while ( ERR>Tol & IT<MaxIt)                 % k-ma iterazione
    for i=1:n
      sum=b(i)-(A(i,[1:i-1,i+1:n]))*...
                        (xold([1:i-1,i+1:n]));
      x(i)=sum/A(i,i);                      % i-ma componente della
                                            % k-ma approssimazione
    end
    ERR=norm(x-xold, inf)/norm(x,inf);      % stima dell'errore relativo
    xold=x;
    IT=IT+1;
end

if (ERR ≤ Tol)                              % controllo sulla stima errore
    IFLAG=1;
end
```

Funzione 1:
Funzione MATLAB per la risoluzione di un sistema di equazioni lineari con il metodo di Jacobi - fine

4.8 Metodi iterativi stazionari

In questo paragrafo, attraverso una riformulazione in termini matriciali, sono ripresi ed approfonditi i concetti di base discussi nel paragrafo precedente. In particolare l'attenzione è rivolta alla classe di metodi noti con il nome di **metodi iterativi stazionari**. Inoltre, sono illustrate alcune tecniche classiche per accelerare la convergenza di metodi iterativi.

Si consideri il sistema di equazioni lineari:

$$Ax = b, \tag{4.48}$$

con $A = [a_{ij}] \in \Re^{n\times n}$ **non singolare**, $x = [x_i]$ e $b = [b_i] \in \Re^n$. Sia $x^* = A^{-1}b$ la sua soluzione.
Si possono dare le seguenti definizioni generali:

Definizione 4.8.1. (Metodo iterativo)
Una famiglia di funzioni $\{G_k\}$*:*

$$G_k \ : \ D_k \ \subseteq \ (\Re^n)^{k+p} \ = \ \underbrace{\Re^n \times ... \times \Re^n}_{k+p \ volte} \rightarrow \ \Re^n, \quad k = 0, 1, ...$$

definisce un **processo iterativo** *(***metodo iterativo***)* **con** p **punti iniziali** *e con dominio* $D^* \subseteq D_0$*, se* D^* *è non vuoto e se per ogni punto* $(x^{(0)}, ..., x^{(-p+1)}) \in D^*$*, la successione* $\{x^{(k)}\}$ *generata mediante le:*

$$x^{(k+1)} = G_k(x^{(k)}, ..., x^{(-p+1)}), \quad k = 0, 1, ... \tag{4.49}$$

è tale che $(x^{(k)}, ..., x^{(-p+1)}) \in D_k$*,* $\forall k \geq 0$*. Un punto* x^* *tale che* $\lim_{k\rightarrow\infty} x^{(k)} = x^*$ *è detto un* **limite** *del processo iterativo.*

Le seguenti definizioni caratterizzano un processo iterativo del tipo (4.49).

Definizione 4.8.2. (Metodo iterativo a s passi)
Detto p *il numero dei punti iniziali, un processo iterativo (4.49) è detto un* **metodo a** s **passi***, se* $p = s - k$ *e se*

$$G_k \ : \ D_k \ \subseteq \ (\Re^n)^s \rightarrow \ \Re^n, \quad k = 0, 1, ...$$

e quindi

$$x^{(k+1)} = G_k(x^{(k)}, ..., x^{(k-s+1)}), \quad k = 0, 1, ...$$

Definizione 4.8.3. (Metodo iterativo stazionario)
Un metodo iterativo del tipo (4.49) a s *passi è detto* **stazionario** *se* $G_k = G$ *e* $D_k = D$*,* $\forall k \geq 0$*. Quindi, in tal caso si ha:*

$$x^{(k+1)} = G(x^{(k)}, ..., x^{(k-s+1)}), \quad k = 0, 1, ...$$

La funzione G *è detta la* **funzione di iterazione del metodo.**

I metodi di Jacobi e di Gauss-Seidel possono essere scritti nella forma:

$$x^{(k+1)} = G(x^{(k)}) = Cx^{(k)} + d, \quad k = 0, 1, 2, ... , \tag{4.50}$$

dove $C = [c_{ij}] \in \Re^{n\times n}$ è detta **matrice di iterazione** e $d = [d_i] \in \Re^n$. La (4.50) definisce la generica iterazione di un metodo iterativo **stazionario ad un passo**. Il

termine *stazionario* si riferisce al fatto che nel metodo, a parte ovviamente la sequenza $\{x^{(k)}\}$, non ci sono quantità che variano al variare delle iterazioni (la matrice C ed il vettore d sono costanti). L'espressione *ad un passo* sta ad indicare che il vettore alla generica iterazione si ottiene utilizzando solo il vettore calcolato alla iterazione precedente.

♣ **Esempio 4.20.** Altri esempi di metodi iterativi sono i seguenti:

- **metodo non stazionario ad un passo**
$$x^{(k+1)} = C_{k+1}x^{(k)} + d^{(k+1)}, \quad k = 0, 1, \dots$$
- **metodo stazionario a due passi**
$$x^{(k+1)} = Cx^{(k)} + Bx^{(k-1)} + d, \quad k = 0, 1, \dots$$
- **metodo non stazionario a due passi**
$$x^{(k+1)} = C_{k+1}x^{(k)} + B_{k+1}x^{(k-1)} + d^{(k+1)}, \quad k = 0, 1, \dots$$
- **metodo stazionario a s passi**
$$x^{(k+1)} = \sum_{i=1}^{s} Cx^{(k+1-i)} + d, \quad k = 0, 1, \dots$$

♣

I metodi iterativi concettualmente più semplici, ed in molti casi più utilizzati, sono quelli **stazionari ad un passo** (p=1), cioè quelli del tipo:

$$x^{(k+1)} = G(x^{(k)}), \qquad k = 0, 1, \dots . \tag{4.51}$$

Affinchè un metodo iterativo della forma (4.50) costituisca una procedura effettivamente utilizzabile per risolvere il sistema (4.48), è indispensabile che esso sia consistente.
Se la successione generata dal metodo iterativo (4.50) converge, indicato con $\bar{x}$ il limite, risulta

$$\bar{x} = \lim_{k\to\infty} x^{(k+1)} = C\left(\lim_{k\to\infty} x^{(k)}\right) + d = C\bar{x} + d,$$

cioè il limite $\bar{x}$ è soluzione del sistema lineare:

$$x = Cx + d\ , \quad o\ \ equivalentemente\ \ (I - C)x = d, \tag{4.52}$$

il quale è detto il **sistema associato** al sistema (4.48). Allora, per i metodi iterativi del tipo (4.50) la definizione di consistenza può essere così riformulata:

Definizione 4.8.4. (Metodo consistente)
Un metodo iterativo del tipo (4.50) si dice consistente con il sistema di equazioni lineari (4.48), se la matrice $(I - C)$ è non singolare e, detta $\bar{x}$ l'unica soluzione del sistema di equazioni lineari $(I - C)x = d$, si ha che $\bar{x}$ è anche la soluzione del sistema $Ax = b$.

4.9 Metodi basati sulla decomposizione della matrice (splitting)

La tecnica più comune per sviluppare un metodo iterativo del tipo (4.50) che sia consistente è quella di considerare una decomposizione (**splitting**) della matrice A della forma:

$$A = M - R, \tag{4.53}$$

dove $M \in \Re^{n\times n}$ è una matrice non singolare, detta **matrice di splitting**. Utilizzando tale decomposizione, il sistema lineare (4.48) è trasformato nel sistema:

$$Mx = Rx + b,$$

da cui deriva il metodo iterativo:

$$Mx^{(k+1)} = Rx^{(k)} + b, \quad k = 0, 1, 2, ...$$

che, essendo M non singolare, può essere anche scritto come:

$$x^{(k+1)} = M^{-1}Rx^{(k)} + M^{-1}b, \quad k = 0, 1, 2, \dots . \tag{4.54}$$

Il metodo (4.54) è un metodo del tipo (4.50), dove in particolare:

$$C = M^{-1}R = I - M^{-1}A, \qquad d = M^{-1}b. \tag{4.55}$$

Inoltre si ha:

Teorema 4.9.1. *Posto $A = M - R$, con M matrice non singolare, il metodo iterativo:*

$$x^{(k+1)} = M^{-1}Rx^{(k)} + M^{-1}b, \quad k = 0, 1, 2, \dots .$$

è consistente con il sistema $Ax = b$.

Dimostrazione Poichè A e M sono matrici non singolari, in base alla (4.55) anche la matrice $I - C = M^{-1}A$ è non singolare. Sia $\bar{x}$ l'unica soluzione del sistema $x = Cx + d$. Si ha:

$$\bar{x} = C\bar{x} + d = (I - M^{-1}A)\bar{x} + M^{-1}b = \bar{x} - M^{-1}(A\bar{x} - b),$$

da cui discende che $A\bar{x} = b$ e, quindi, $\bar{x}$ è la soluzione del sistema $Ax = b$. In base alla definizione 4.8.4, il metodo è quindi consistente. ■

È importante osservare che la scelta della matrice di splitting M in (4.53), per ragioni di efficienza, non può essere completamente arbitraria. Dalla (4.54) si osserva infatti che, ad ogni iterazione del metodo, è richiesto il calcolo di un vettore del tipo $y = M^{-1}z$.

Di conseguenza, è importante scegliere la matrice M in modo che la risoluzione di sistemi lineari della forma $My = z$ comporti un "ragionevole" costo computazionale. In generale, la scelta della matrice M cade su matrici per le quali il calcolo di $y = M^{-1}z$ richieda un costo computazionale equivalente a quello di un prodotto matrice-vettore (circa n^2 flops).

♣ **Esempio 4.21.** Si consideri la seguente decomposizione della matrice $A = [a_{ij}] \in \Re^{n\times n}$:

$$A = D + L + U, \tag{4.56}$$

dove:

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1,n-1} & 0 \\ 0 & \cdots & \cdots & 0 & a_{nn} \end{pmatrix}, \quad a_{ii} \neq 0, \quad i = 1, \ldots, n$$

$$L = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ a_{21} & 0 & \cdots & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1,n-1} & a_{1n} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{n-2,n-1} & a_{n-2,n} \\ 0 & \cdots & \cdots & 0 & a_{n-1,n} \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}.$$

Utilizzando tale decomposizione e denotando con il simbolo $[Px]_i$ la i-ma componente del prodotto tra una matrice $P \in \Re^{n\times n}$ ed un vettore $x \in \Re^n$, il metodo di Jacobi:

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right) / a_{ii}, \quad i = 1, \ldots, n,$$

può essere riscritto nel modo seguente:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \frac{1}{a_{ii}} [(L+U)x^{(k)})]_i, \quad i = 1, \ldots, n,$$

da cui:

$$x_i^{(k+1)} = [D^{-1}b]_i - [D^{-1}(L+U)x^{(k)})]_i, \quad i = 1, \ldots, n,$$

e, quindi, in forma matriciale:

$$x^{(k+1)} = -D^{-1}(L+U)x^{(k)} + D^{-1}b. \tag{4.57}$$

Dalla (4.57) si evince che il metodo iterativo di Jacobi deriva da uno splitting di A del tipo (4.53), dove in particolare:

$$M_J = D; \quad R_J = -(L+U).$$

Si osservi che la matrice di splitting del metodo di Jacobi è la matrice diagonale i cui elementi sono gli elementi diagonali della matrice A e quindi, poichè tali elementi sono diversi da zero per ipotesi, è non singolare. Di conseguenza, il metodo di Jacobi è consistente (Teorema 4.9.1).
In conclusione, in forma (4.50) il metodo di Jacobi assume la seguente espressione:

METODO DI JACOBI

$$x^{(k+1)} = C_J x^{(k)} + d_J$$
$$C_J = -D^{-1}(L+U) = I - D^{-1}A$$
$$d_J = D^{-1}b.$$

♣

♣ **Esempio 4.22.** Il metodo di G-S :

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, \ldots, n,$$

può essere riscritto come:

$$\frac{1}{a_{ii}}\sum_{j=1}^{i} a_{ij}x_j^{(k+1)} = \frac{b_i}{a_{ii}} - \frac{1}{a_{ii}}\sum_{j=i+1}^{n} a_{ij}x_j^{(k)}, \quad i = 1, \ldots, n,$$

da cui, in base della decomposizione (4.56), si ha:

$$[D^{-1}(D+L)x^{(k+1)}]_i = [D^{-1}b]_i - [D^{-1}Ux^{(k)}]_i, \quad i = 1, \ldots, n,$$

e quindi, in forma matriciale:

$$x^{(k+1)} = -(D+L)^{-1}Ux^{(k)} + (D+L)^{-1}b. \tag{4.58}$$

Dalla (4.58) si osserva che il metodo di G-S deriva da uno splitting del tipo (4.53), dove, in particolare:

$$M_{GS} = D + L; \quad R_{GS} = -U.$$

La matrice di splitting del metodo di G-S è quindi triangolare inferiore e, poichè gli elementi diagonali di A sono diversi da zero, è non singolare, da cui discende che il metodo di G-S è consistente.
In conclusione, in forma (4.50) esso assume la seguente espressione:

METODO DI GAUSS-SEIDEL

$$x^{(k+1)} = C_G x^{(k)} + d_G$$
$$C_G = -(D+L)^{-1}U = I - (D+L)^{-1}A$$
$$d_G = (D+L)^{-1}b$$

♣

4.10 Studio della convergenza

In questo paragrafo si studiano alcune condizioni sotto le quali la successione $\{x^{(k)}\}$ generata da un metodo iterativo del tipo (4.50) converge, qualunque sia la scelta di $x^{(0)}$.

Si analizzano ora le proprietà di convergenza dei metodi iterativi *stazionari ad un passo*. A tal fine si premettono alcuni risultati [13].

Teorema 4.10.1. *Sia $G : D \subseteq \Re^n \to \Re^n$ una contrazione*[14] *su $D_0 \subseteq D$. Allora G può avere al più un punto fisso in D_0.*

Dimostrazione Siano x e $y \in D_0$, $x \neq y$, due punti fissi di G. Si ha:

$$\|x - y\| = \|G(x) - G(y)\| < \|x - y\|,$$

che è una contraddizione. Quindi $x = y$. ■

Il Teorema che segue [15] (noto come **Teorema delle Contrazioni**) assicura che, nelle ipotesi in cui G sia una contrazione, la successione $\{x^{(k)}\}$ generata da un metodo stazionario ad un passo converge ad un punto fisso di G, qualunque sia il punto iniziale $x^{(0)}$. Ciò mostra anche che una contrazione ha almeno un punto fisso. Per il Teorema 4.10.1 esso è unico.

Teorema 4.10.2. *Sia $G : D \subseteq \Re^n \to \Re^n$ una contrazione su un insieme chiuso $D_0 \subseteq D$ e tale che $G(D_0) \subseteq D_0$. Allora la successione $\{x^{(k)}\}$ generata dal metodo iterativo (4.51) converge ad un punto fisso di G, qualunque sia il punto iniziale $x^{(0)}$.*

Dimostrazione Sia $x^{(0)}$ un arbitrario punto di D_0. Poichè $G(D_0) \subseteq D_0$, la successione $\{x^{(k)}\}$ generata da $x^{(k+1)} = G(x^{(k)})$, $k = 0, 1, ...$, è ben definita ed appartiene a D_0. Inoltre, dall'ipotesi che G sia una contrazione su D_0 si ha:

$$\|x^{(k+1)} - x^{(k)}\| = \|G(x^{(k)}) - G(x^{(k-1)})\| \leq \alpha \|x^{(k)} - x^{(k-1)}\|.$$

[13] Tali risultati si possono estendere al caso più generale degli spazi metrici. In particolare, il **Teorema delle Contrazioni** è noto anche come **Teorema di punto fisso di Banach-Caccioppoli** ed è un importante strumento nella teoria degli spazi metrici.

[14]

Definizione 4.10.1. (Contrazione)
Una funzione $G : D \subseteq \Re^n \to \Re^n$ è una **contrazione** *sull'insieme $D_0 \subseteq D$, se esiste un $0 < \alpha < 1$ tale che:*

$$\|G(x) - G(y)\| \leq \alpha \|x - y\|, \quad \forall x, y \in D_0.$$

Si verifica immediatamente che se una funzione $G : D \subseteq \Re^n \to \Re^n$ è una contrazione su $D_0 \subseteq D$, allora essa è continua in D_0.

[15] Il **Teorema delle contrazioni** è già stato enunciato nel **Capitolo 1**, **§1.7.1** (**Teorema 1.7.1**).

Applicando lo stesso ragionamento a $\|x^{(k)} - x^{(k-1)}\|$ si ottiene:

$$\|x^{(k+1)} - x^{(k)}\| \leq \alpha^2 \|x^{(k-1)} - x^{(k-2)}\|$$

e, quindi, iterando il procedimento, si ha:

$$\|x^{(k+1)} - x^{(k)}\| \leq \alpha^k \|x^{(1)} - x^{(0)}\|. \tag{4.59}$$

La (4.59) è facilmente generalizzabile nella:

$$\|x^{(k+i)} - x^{(k+i-1)}\| \leq \alpha^{i-1} \|x^{(k+1)} - x^{(k)}\|, \;\; i \geq 2, \; \forall k \geq 0. \tag{4.60}$$

Dalle (4.59) e (4.60) segue che:

$$\|x^{(k+p)} - x^{(k)}\| \leq \sum_{i=1}^{p} \|x^{(k+i)} - x^{(k+i-1)}\| \leq (\alpha^{p-1} + \alpha^{p-2} + ... + 1)\|x^{(k+1)} - x^{(k)}\|$$

$$\leq (\alpha^{p-1} + \alpha^{p-2} + ... + 1)\alpha^k \|x^{(1)} - x^{(0)}\| \leq [\alpha^k/(1-\alpha)]\|x^{(1)} - x^{(0)}\|,$$

dall' essere

$$\sum_{j=0}^{p-1} \alpha^j = \frac{1 - \alpha^{p-1}}{1 - \alpha},$$

e poiché $0 < \alpha < 1 \Rightarrow 0 < \alpha^{p-1} < 1$ con $p \geq 1$. Quindi, $\{x^{(k)}\}$ è una successione di Cauchy e converge ad un limite $x^* \in D_0$. Inoltre, dalla continuità di G si ha:

$$x^* = \lim_{k \to \infty} x^{(k+1)} = \lim_{k \to \infty} G(x^{(k)}) = G(x^*),$$

per cui x^* è un punto fisso di G. ■

Il Teorema precedente può essere applicato anche a processi iterativi non stazionari e si generalizza nel seguente:

Teorema 4.10.3. *Sia $G : D \subseteq \Re^n \to \Re^n$ una contrazione su un insieme chiuso $D_0 \subseteq D$ e tale che $G(D_0) \subseteq D_0$. Si assuma che $G_k : D_0 \subseteq \Re^n \to \Re^n$, $k = 0, 1, \ldots$, siano funzioni tali che $G_k(D_0) \subseteq D_0$, $\forall k \geq 0$, e che:*

$$\lim_{k \to \infty} \|G_k(x) - G(x)\| = 0, \quad \textit{uniformemente rispetto a } x \in D_0 \; . \tag{4.61}$$

Allora la successione $\{x^{(k)}\}$ generata dal metodo non stazionario:

$$x^{(k+1)} = G_k(x^{(k)}), \quad k = 0, 1, ... \tag{4.62}$$

converge all'unico punto fisso x^* di G in D_0, qualunque sia il punto iniziale $x^{(0)}$.

Dimostrazione Sia $x^{(0)}$ un arbitrario punto di D_0. Poichè $G_k(D_0) \subseteq D_0$, $\forall k \geq 0$, la successione $\{x^{(k)}\}$ generata dal metodo (4.62) è ben definita ed appartiene a D_0. Inoltre, dall'ipotesi che G sia una contrazione su D_0 si ha:

$$\|x^{(k+1)} - x^*\| = \|G_k(x^{(k)}) - G(x^*)\| \leq \|G_k(x^{(k)}) - G(x^{(k)})\| + \|G(x^{(k)}) - G(x^*)\| \leq$$

$$\leq \epsilon_k + \alpha\|x^{(k)} - x^*\| \leq \cdots \leq \sum_{j=0}^{k} \alpha^{k-j}\epsilon_j + \alpha^{k+1}\|x^{(0)} - x^*\|, \tag{4.63}$$

con $\epsilon_k = \|G_k(x^{(k)}) - G(x^{(k)})\|$. Per l'ipotesi (4.61) la successione $\{\epsilon_k\}$ converge a zero e, quindi, dato un $\epsilon > 0$, esiste un k_0 tale che $\epsilon_k \leq \epsilon$, $\forall k \geq k_0$. Posto $\gamma_k = \sum_{j=0}^{k} \alpha^{k-j}\epsilon_j$, si ha:

$$\gamma_k \leq \alpha^{k-k_0}\gamma_{k_0} + \sum_{j=k_0+1}^{k} \alpha^{k-j}\epsilon_j \leq \alpha^{k-k_0}\gamma_{k_0} + \epsilon[1/(1-\alpha)],$$

che mostra che la successione $\{\gamma_k\}$ converge a zero. Quindi, dalla (4.63) si ha che $\lim_{k\to\infty} x^{(k)} = x^*$. ■

Una condizione necessaria e sufficiente per la convergenza è stabilita dal seguente:

Teorema 4.10.4. *Il metodo iterativo*

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \ldots,$$

è convergente se e solo se

$$\rho(C) < 1 \tag{4.64}$$

dove

$$\rho(C) = \max_{i=1,\ldots,n} |\lambda_i|,$$

con λ_i i-mo autovalore di C, è il **raggio spettrale** *della matrice C.*

Dimostrazione Sia $\rho(C) < 1$. In tale ipotesi la matrice $I - C$ è non singolare (vedi **Teorema B.5** dell'**Appendice B**, **Parte prima**). Sia $\bar{x}$ l'unica soluzione del sistema lineare $x = Cx + d$ e sia

$$e^{(k)} = x^{(k)} - \bar{x}, \ \ \forall k \geq 0.$$

Si ha:

$$e^{(k+1)} = x^{(k+1)} - \bar{x} = Cx^{(k)} + d - (C\bar{x} + d) = C(x^{(k)} - \bar{x}) = Ce^{(k)}.$$

Applicando lo stesso ragionamento a $e^{(k)}$ si ottiene:

$$e^{(k+1)} = C(Ce^{(k-1)}) = C^2e^{(k-1)}.$$

Dopo $k+1$ passi si ha:

$$e^{(k+1)} = C^{k+1}e^{(0)}. \tag{4.65}$$

Poichè $\rho(C) < 1$, dal **Teorema B.17** dell'**Appendice B**, **Parte prima**, segue che:

$$\lim_{k\to\infty} e^k = 0, \ \ \forall e^{(0)}$$

e, quindi, il metodo (4.50) converge a $\bar{x}$ qualunque sia $x^{(0)}$.

Si supponga ora che il metodo sia convergente. Indicato con $\bar{x}$ il limite, si ricorda che $\bar{x}$ è soluzione del sistema associato al metodo, cioè $\bar{x} = C\bar{x} + d$. Si supponga per assurdo che $\rho(C) \geq 1$. Sia $\tilde{\lambda}$ un

autovalore di C tale che $|\tilde{\lambda}| = \rho(C)$ e sia $\tilde{x}$ un autovettore ad esso corrispondente. Posto $x^{(0)} = \bar{x} - \tilde{x}$ e $e^{(k)} = x^{(k)} - \bar{x}$, si ha:

$$e^{(k)} = C^k e^{(0)} = \tilde{\lambda}^k e^{(0)},$$

da cui, poiché $|\tilde{\lambda}| \geq 1$, si ricava che la successione $\{e^{(k)}\}$ non converge a zero, e che quindi la successione $\{x^{(k)}\}$ non converge a $\bar{x}$. Ciò contraddice l'ipotesi. ■

♣ **Esempio 4.23.** Si consideri il sistema lineare:

$$\begin{cases} 2x_1 - x_2 = 0 \\ x_1 - 3x_2 = 0, \end{cases} \tag{4.66}$$

e quello ottenuto semplicemente scambiando le sue equazioni:

$$\begin{cases} x_1 - 3x_2 = 0 \\ 2x_1 - x_2 = 0\,. \end{cases} \tag{4.67}$$

Per il primo sistema sia il metodo di Jacobi che di Gauss-Seidel convergono. Le matrici di iterazione dei metodi di Jacobi e G-S, applicati al primo sistema, sono:

$$C_J = \begin{pmatrix} 0 & 0.5000 \\ 0.3333 & 0 \end{pmatrix}, \qquad C_G = \begin{pmatrix} 0 & 0.5000 \\ 0 & 0.1667 \end{pmatrix}$$

aventi raggio spettrale, rispettivamente:

$$\rho(C_J) = 0.4082 < 1$$
$$\rho(C_G) = 0.1667 < 1$$

Per il secondo sistema, ad esempio partendo dai valori iniziali $x_1^{(0)} = x_2^{(0)} = 0.5$, entrambi i metodi non convergono. Le matrici di iterazione dei metodi di Jacobi e G-S, applicati al secondo sistema, sono:

$$C_J = \begin{pmatrix} 0 & 3 \\ 2 & 0 \end{pmatrix}, \qquad C_G = \begin{pmatrix} 0 & 3 \\ 0 & 6 \end{pmatrix}$$

aventi raggio spettrale, rispettivamente:

$$\rho(C_J) = 2.4495 > 1$$
$$\rho(C_G) = 6 > 1$$

Per entrambi i ristemi i risultati sono in accordo con le considerazioni sulla convergenza, dedotte nell'esempio 4.7.

♣

♣ **Esempio 4.24.** Nell'esempio 4.3 si è considerato il sistema lineare:

$$\begin{cases} x_1 & -2x_2 & +2x_3 & = & 1 \\ -x_1 & +x_2 & -x_3 & = & -1 \\ -2x_1 & -2x_2 & +x_3 & = & -3, \end{cases}$$

per il quale, come si evince dai risultati riportati nelle Tabelle 4.4 e 4.5, il metodo di Jacobi converge, mentre ciò non accade per il metodo di G-S. Si osservi che, per tale sistema, si ha:

$$\begin{aligned} \rho(C_J) &= 0.1255 \cdot 10^{-4} < 1 \\ \rho(C_G) &= 4.8284 > 1 \end{aligned}$$

Quindi, si può affermare che, in generale, non esiste un legame tra il comportamento (in termini di convergenza) tra i due metodi.
Si consideri ora il sistema lineare:

$$\begin{cases} x_1 & -2x_2 & -2x_3 & = & 5 \\ -x_1 & +x_2 & -x_3 & = & -1 \\ -2x_1 & -2x_2 & +x_3 & = & -3, \end{cases}$$

per il quale si ha:

$$\begin{aligned} \rho(C_J) &= 3.2361 > 1 \\ \rho(C_G) &= 12.3246 > 1 \end{aligned}$$

e quindi entrambi i metodi divergono. Si osservi che, considerando la decomposizione (4.56) per la matrice dei coefficienti di tale sistema, gli elementi della matrice $-(L+U)$ sono non negativi.
Si consideri ora il seguente sistema lineare:

$$\begin{cases} 5x_1 & -2x_2 & -1x_3 & = 3 \\ -x_1 & +4x_2 & -x_3 & = 2 \\ -3x_1 & -2x_2 & +7x_3 & = 2, \end{cases}$$

per il quale si ha:

$$\begin{aligned} \rho(C_J) &= 0.5944 < 1 \\ \rho(C_G) &= 0.3833 < 1 \end{aligned}$$

e quindi entrambi i metodi convergono. Si osservi che anche per tale sistema gli elementi della matrice $-(L+U)$ sono non negativi. Tale particolare comportamento è stabilito dal seguente [31]:

Teorema 4.10.5. [Teorema di Stein-Rosenberg]
Posto $A = D + L + U$, se la matrice $-(L+U)$ è non negativa, allora è vera una e una soltanto delle condizioni seguenti

- $\rho(C_J) = \rho(C_G) = 0$
- $\rho(C_J) = \rho(C_G) = 1$
- $0 < \rho(C_G) < \rho(C_J) < 1$
- $1 < \rho(C_J) < \rho(C_G)$

Quindi, in base a tale Teorema, per i sistemi lineari che ne soddisfano l'ipotesi, i metodi di Jacobi e di G-S o sono entrambi convergenti o sono entrambi divergenti. ♣

Per sintetizzare quanto detto finora, anche alla luce dei Teoremi 4.9.1 e 4.10.4, si può concludere che, se valgono le condizioni seguenti:

1. A è non singolare, e quindi il sistema lineare $Ax = b$ ha un'unica soluzione;
2. $C = I - M^{-1}A$, $d = M^{-1}b$, con M matrice di splitting non singolare;
3. $\rho(C) < 1$;

allora, qualunque sia $x^{(0)}$, la successione di vettori $\{x^{(k)}\}$ generata dal metodo (4.50) converge all'unica soluzione di $Ax = b$. Infatti, la condizione 2. assicura che il metodo iterativo (4.50) è consistente, cioè, se converge, converge alla soluzione del sistema $Ax = b$. La condizione 3., invece, assicura che il metodo converge qualunque sia il punto iniziale $x^{(0)}$.

Si osservi che la condizione per la convergenza (4.64) stabilita dal Teorema 4.10.4 non è di verifica facile perchè richiede la conoscenza dell'autovalore massimo della matrice di iterazione C, un problema che in generale è computazionalmente molto dispendioso. D'altra parte, nelle applicazioni, è necessario disporre di condizioni facilmente verificabili. In tale ottica, è importante ricordare la relazione seguente (vedi **Teorema B.12** dell'**Appendice B**, **Parte prima**):

$$\rho(A) \leq \|A\|, \tag{4.68}$$

che vale per ogni $A \in \Re^{n\times n}$ e per ogni norma matriciale $\|\cdot\|$ compatibile con una norma vettoriale (cfr. **Definizione B.5** dell'**Appendice B**, nella **Parte prima**).
Si riporta inoltre un risultato, che è utile in alcune applicazioni, il quale stabilisce una condizione sufficiente per la convergenza dei metodi iterativi basati sullo splitting di A, nel caso in cui A presenti particolari caratteristiche.

Teorema 4.10.6. *Se la matrice A del sistema (4.48) è* **simmetrica e definita positiva**[16]*, posto $A = M - R$, con M matrice non singolare, il metodo iterativo (4.54) converge se la matrice simmetrica:*

$$Q = M + M^T - A,$$

è **definita positiva**.

Dimostrazione Siano B e G le due matrici simmetriche definite positive tali che $A = B^2$ e $Q = G^2$ (l'esistenza e l'unicità di tali matrici è assicurata dal **Teorema B.19** dell'**Appendice B**, **Parte prima**). Considerata la matrice di iterazione del metodo, $C = I - M^{-1}A$, si ponga:

$$H = BCB^{-1} = I - BM^{-1}B,$$

[16]Sistemi lineari con tale caratteristica sono spesso ricorrenti nelle applicazioni. Un esempio è la risoluzione numerica di equazioni differenziali alle derivate parziali di tipo ellittico.

da cui, la matrice HH^T, i cui autovalori sono non negativi perchè semidefinita positiva (vedi **Teoremi B.9** e **B.10** dell'**Appendice B**, **Parte prima**), ha la seguente espressione:

$$\begin{aligned} HH^T &= (I - BM^{-1}B)(I - BM^{-T}B) = I - BM^{-T}B - BM^{-1}B + BM^{-1}AM^{-T}B = \\ &= I - BM^{-1}(M + M^T - A)M^{-T}B = I - T, \end{aligned} \tag{4.69}$$

dove:

$$T = BM^{-1}QM^{-T}B.$$

Poichè $Q = G^2$, la matrice T si scrive anche come:

$$T = (BM^{-1}G)(BM^{-1}G)^T.$$

Poichè B, G, e M sono matrici non singolari, la matrice $BM^{-1}G$ è non singolare. Di conseguenza, la matrice T, essendo il prodotto di una matrice non singolare per la sua trasposta, è simmetrica e definita positiva (**Teorema B.10** dell'**Appendice B**, **Parte prima**) e ha gli autovalori positivi. Dalla (4.69) segue che gli autovalori della matrice HH^T appartengono all'intervallo $[0, 1[$ e, quindi, $\rho(HH^T) < 1$. Se si considera la norma matriciale $\|\cdot\|_{2,B}$ (vedi **§B.3** dell'**Appendice B** della **Parte prima**), si ha:

$$\|C\|_{2,B} = \|BCB^{-1}\|_2 = \|H\|_2 = (\rho(HH^T))^{1/2} < 1.$$

Essendo, dunque, $\|C\| < 1$ ed, in particolare,

$$\rho(C) \leq \|C\| < 1,$$

per il Teorema 4.10.4 il metodo è convergente. ■

In generale, per la risoluzione di sistemi di equazioni lineari:

$$Ax = b, \quad A \in \Re^{n\times n},\ x, b \in \Re^n, \tag{4.70}$$

si utilizzano processi iterativi lineari, cioè ad esempio metodi stazionari ad un passo del tipo (4.51) in cui la funzione di iterazione G è lineare, $G(x) = Cx + d$, dove C è una matrice $\in \Re^{n\times n}$ e d è un vettore $\in \Re^n$. Il metodo (4.51) assume quindi la forma:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, ... \tag{4.71}$$

Da osservare che in tal caso il dominio della funzione G è tutto $\Re^n$ e, di conseguenza, il metodo (4.71) è sempre ben definito.

Considerati due punti x e $y \in \Re^n$, si ha:

$$\|Cx + d - Cy - d\| = \|C(x - y)\| \leq \|C\|\|x - y\|.$$

Segue che la funzione $G(x) = Cx + d$ è una contrazione se $\|C\| < 1$. In tale ipotesi, in base al **Teorema 4.10.2**, il metodo iterativo (4.71) genera una successione $\{x^{(k)}\}$ che converge ad un punto fisso di G, e quindi, ad una soluzione del sistema lineare $(I - C)x = d$, qualunque sia il punto iniziale $x^{(0)}$. Inoltre, nell'ipotesi che $\|C\| < 1$, la matrice $(I - C)$ è non singolare, e quindi il sistema $(I - C)x = d$ ammette una sola

soluzione. Se, infine, la matrice C è scelta in modo tale che il sistema $(I - C)x = d$ sia equivalente al sistema da risolvere (4.70), allora si può concludere che nell'ipotesi che $\|C\| < 1$ il metodo (4.71) converge alla soluzione.

4.11 Velocità di convergenza

Allo scopo di analizzare i fattori che determinano la velocità di convergenza, si suppone che il metodo iterativo (4.50) sia consistente e convergente. Posto:

$$e^{(k)} = x^{(k)} - x^*,$$

l'errore di troncamento analitico alla k-ma iterazione del metodo, per la (4.65) si ha:

$$e^{(k)} = C^k e^{(0)}.$$

Denotata con $\|\cdot\|$ una norma vettoriale ed una norma matriciale tra loro compatibili, si ha:

$$\|e^{(k)}\| \leq \|C^k\| \|e^{(0)}\|, \tag{4.72}$$

da cui discende che $\|C^k\|$ rappresenta una misura di quanto l'errore si riduce dopo k iterazioni.
In generale, si desidera arrestare il metodo iterativo quando:

$$\|e^{(k)}\| \leq \texttt{Tol} \|e^{(0)}\|, \tag{4.73}$$

con $\texttt{Tol} < 1$. In base alla (4.72), per soddisfare la (4.73), è sufficiente determinare un numero intero k tale che:

$$\|C^k\| \leq \texttt{Tol}. \tag{4.74}$$

Assumendo che $\|C\| < 1$ (condizione sufficiente per la convergenza), si ha che $\|C^k\| \leq \|C\|^k < 1$, e, applicando la funzione logaritmo naturale ad entrambi i membri della (4.74) e moltiplicandoli per k, si ha:

$$k \geq -\log(\texttt{Tol}) / \left(-\frac{1}{k} \log(\|C^k\|) \right).$$

Si evince quindi che il numero minimo di iterazioni, k, necessario per ottenere la richiesta riduzione (4.73) è inversamente proporzionale alla quantità:

$$-\frac{1}{k} \log(\|C^k\|).$$

Ciò conduce alla seguente definizione:

Definizione 4.11.1. (Velocità media di convergenza)
Relativamente al metodo iterativo:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \ldots ,$$

si definisce **velocità media di convergenza** *per k iterazioni la quantità*

$$R_k(C) = -\frac{1}{k}\log(\|C^k\|).$$

Si osservi che il valore di $R_k(C)$ dipende dalla norma matriciale che si utilizza.
Un'altra definizione di velocità di convergenza, che non presenta tale dipendenza, si ottiene dalla relazione:

$$\lim_{k\to\infty} \|C^k\|^{1/k} = \rho(C),$$

che vale, ovviamente, se $\rho(C) < 1$. Si ha dunque la seguente:

Definizione 4.11.2. (Velocità asintotica di convergenza)
Relativamente al metodo iterativo:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k = 0, 1, 2, \ldots ,$$

si definisce **velocità asintotica di convergenza** *il numero:*

$$R_\infty(C) = \lim_{k\to\infty} R_k(C) = -\log(\rho(C)) \ .$$

Il valore di $R_\infty(C)$ è determinato solo dal raggio spettrale della matrice di iterazione e, quindi, è indipendente dalla norma che si utilizza. Inoltre, quanto più piccolo è $\rho(C)$ tanto più grande è $R_\infty(C)$, e quindi tanto più velocemente il metodo converge alla soluzione.
Sulla base della definizione 4.11.2, si ha che una stima del minimo numero di iterazioni necessarie per ridurre l'errore di un fattore `Tol` è data da

$$k \geq \frac{-\log(\texttt{Tol})}{R_\infty(C)}. \tag{4.75}$$

♣ **Esempio 4.25.** Si considerino i sistemi lineari (a) e (b) dell'esempio 4.14. Per la matrice di iterazione del metodo di Jacobi applicato a tali sistemi si ha:

$$\begin{aligned} &\text{sistema (a)}: \quad \rho(C_J) = 0.7637; \\ &\text{sistema (b)}: \quad \rho(C_J) = 0.3333. \end{aligned}$$

Quindi, il metodo di Jacobi converge più velocemente per il sistema (b). Ciò è confermato dai risultati riportati nelle Tabelle 4.8 e 4.9: il numero di iterazioni effettuate per rendere la norma del massimo dell'errore minore di `Tol`$= 10^{-6}$ è 48 per il sistema (a) e 16 per il sistema (b).
Posto `Tol`$= 10^{-6}$ nella (4.75), si ha:

$$\text{sistema (a)} : k \simeq 51;$$
$$\text{sistema (b)} : k \simeq 13,$$

da cui si evince che la (4.75) fornisce una stima attendibile "a priori" del numero di iterazioni necessarie per ottenere la fissata riduzione dell'errore. ♣

♣ **Esempio 4.26.** In relazione al sistema (a) dell'esempio 4.14, per la matrice di iterazione del metodo di G-S si ha $\rho(C_G) = 0.5833$. Quindi, utilizzando la (4.75), si ha che una stima del numero di iterazioni necessarie per ridurre l'errore di `Tol`$= 10^{-6}$ è $k = 26$. L'attendibilità di tale stima è confermata dai risultati riportati nella Tabella 4.8. Inoltre, come descritto nell'esempio precedente, per la matrice di iterazione del metodo di Jacobi applicato al sistema (a) si ha $\rho(C_J) = 0.7637$ e, quindi, per tale sistema il metodo di G-S converge più velocemente, come confermano di nuovo i risultati riportati nella Tabella 4.8.

♣

Si osservi che, nonostante i risultati mostrati negli esempi precedenti, per alcuni sistemi lineari la stima (4.75) può fornire un valore che risulta sensibilmente inferiore al numero effettivo di iterazioni necessarie. L'utilizzo della seguente stima:

$$k \geq \frac{-\log(\texttt{Tol})}{R_k(C)},$$

in cui si considera la velocità media di convergenza, può fornire un valore più attendibile.

Osservazione 4.3. *Come illustrato nel §4.11, la velocità di convergenza di un metodo iterativo dipende dalle proprietà spettrali della matrice dei coefficienti; di qui il tentativo di trasformare il sistema lineare in esame in uno equivalente, che abbia, cioè, la stessa soluzione, ma presenti proprietà spettrali più vantaggiose ai fini della convergenza del metodo. La matrice che determina tale trasformazione prende il nome di* **precondizionatore**. *Ad esempio, se una matrice M approssima, in qualche modo, la matrice dei coefficienti, A, il sistema trasformato*

$$M^{-1}Ax = M^{-1}b$$

ha la stessa soluzione del sistema originale, $Ax = b$, ma le proprietà spettrali della sua matrice dei coefficienti, $M^{-1}A$, potrebbero garantire una più rapida convergenza del

metodo iterativo. In generale, comunque, un buon precondizionatore deve migliorare la velocità di convergenza del metodo, ma tanto da bilanciare il costo computazionale richiesto per la sua costruzione e la sua applicazione. Cioè, un efficiente precondizionatore M, deve poter avere le seguenti caratteristiche:

- *M è una buona approssimazione di A in qualche senso;*
- *il costo computazionale della costruzione di M non deve essere proibitivo tanto da annulare i vantaggi dell'aumentata velocità di convergenza;*
- *la matrice M deve essere facile da invertire nel senso che il sistema $My = c$ sia più facile da risolvere del sistema $Ax = b$.*

La individuazione di un opportuno precondizionatore per una data matrice è un attivo ed attuale campo di ricerca. Noi ci limitiamo ad introdurre alcuni classici metodi per l'accelerazione della convergenza.

4.12 Accelerazione della convergenza

4.12.1 Metodi di rilassamento: il metodo SOR

L'analisi effettuata nel paragrafo precedente ha mostrato che la convergenza di un metodo iterativo del tipo (4.50) è tanto più rapida quanto più piccolo è il raggio spettrale della matrice di iterazione del metodo. Di conseguenza, una delle tecniche più naturali per accelerare la convergenza è apportare modifiche al metodo in modo da rendere il più piccolo possibile $\rho(C)$. In generale tale problema è di non facile soluzione: lo si risolve solo in qualche caso particolare, ma rilevante per le applicazioni.
Come primo passo in tale direzione, si riscrive il metodo di G-S [17] :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n,$$

nella forma (aggiungendo e sottraendo $a_{ii} x_i^{(k)}$):

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}, \quad i = 1, \ldots, n, \tag{4.76}$$

dove:

$$r_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}, \quad i = 1, \ldots, n.$$

[17] Consideriamo il metodo di G-S perché quando converge, converge piú velocemente del metodo di Jacobi (Teorema 4.10.5).

Definizione 4.12.1. (Metodo SOR)
Considerato un numero reale ω, si modificano le (4.76) come segue [18]*:*

$$x_i^{(k+1)} = x_i^{(k)} + \omega r_i^{(k)}, \quad i = 1, \ldots, n. \tag{4.77}$$

Le formule (4.77) definiscono un metodo iterativo detto **metodo SOR** *(***S***uccessive* **O***ver* **R***elaxation) ed ω è detto* **parametro di rilassamento**.

Si può dedurre anche per un metodo SOR una formulazione matriciale. Si ha la seguente:

Proposizione 4.12.1. (Formulazione matriciale del Metodo SOR)
Un metodo SOR è un metodo che deriva da uno splitting di A del tipo (4.53), dove, in particolare:

$$M_\omega = \omega^{-1} D + L; \quad R_\omega = (\omega^{-1} - 1)D - U \ ,$$

con L, D e U definite tramite la decomposizione (4.56).

Dimostrazione Le (4.77) possono essere scritte come:

$$x_i^{(k+1)} + \omega \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} = (1-\omega) x_i^{(k)} - \omega \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \omega \frac{b_i}{a_{ii}}, \quad i = 1, \ldots, n,$$

da cui si ha:

$$[(I + \omega D^{-1} L) x^{(k+1)}]_i = [(1-\omega) x^{(k)}]_i - [\omega D^{-1} U x^{(k)}]_i + \omega [D^{-1} b]_i \ , \quad i = 1, \ldots, n,$$

e, quindi, in forma matriciale:

$$x^{(k+1)} = (I + \omega D^{-1} L)^{-1} [(1-\omega) I - \omega D^{-1} U] x^{(k)} + (I + \omega D^{-1} L)^{-1} \omega D^{-1} b. \tag{4.78}$$

Osservando che:

$$(I + \omega D^{-1} L) = \omega D^{-1} (\omega^{-1} D + L)$$

e

$$(1-\omega) I - \omega D^{-1} U = \omega D^{-1} [(1-\omega)\omega^{-1} D - U],$$

la (4.78) diventa:

$$x^{(k+1)} = \overbrace{(\omega^{-1} D + L)^{-1}}^{M_\omega^{-1}} \overbrace{[(\omega^{-1} - 1)D - U]}^{R_\omega} x^{(k)} + \overbrace{(\omega^{-1} D + L)^{-1} b}^{M_\omega^{-1} b} \tag{4.79}$$

e quindi la tesi. ■

È immediato verificare che, se $\omega \neq 0$, la matrice di splitting M_ω è non singolare e quindi il metodo SOR è consistente. Inoltre, nel caso $\omega = 1$, il metodo SOR si riduce al metodo

[18] Quando $\omega = 1$ il metodo SOR coincide con il metodo di Gauss-Seidel.

di G-S. In conclusione, in forma (4.50) un metodo SOR ha la seguente espressione:

METODO SOR

$$x^{(k+1)} = C_\omega x^{(k)} + d_\omega \quad , \quad C_\omega = M_\omega^{-1}\ R_\omega \quad , \quad M_\omega = (\omega^{-1}D + L)$$
$$d_\omega = M_\omega^{-1} b \quad , \quad R_\omega = (\omega^{-1} - 1)D - U \quad . \tag{4.80}$$

Per la convergenza del metodo SOR è fondamentale il risultato seguente:

Teorema 4.12.1. *Per la matrice di iterazione del metodo SOR ((4.77)) si ha:*

$$\rho(C_\omega) \geq |\omega - 1|.$$

Dimostrazione Considerate le matrici non singolari $E = D^{-1}L$ e $F = D^{-1}U$, la matrice di iterazione del metodo SOR può scriversi come:

$$C_\omega = (I + \omega E)^{-1}[(1 - \omega)I - \omega F].$$

Poichè $(I + \omega E)$ è un matrice triangolare inferiore con elementi diagonali uguali a 1, lo è anche la sua inversa che ha, quindi, determinante uguale a 1. La matrice $[(1 - \omega)I - \omega F]$ è una matrice triangolare superiore con elementi diagonali tutti uguali a $(1-\omega)$, il cui determinante è quindi $(1-\omega)^n$. Da quanto detto, il determinante di C_ω, $\mathtt{det}(C_\omega)$, è:

$$\mathtt{det}(C_\omega) = \mathtt{det}((I + \omega E)^{-1}) \cdot \mathtt{det}([(1 - \omega)I - \omega F]) = (1 - \omega)^n.$$

Poichè il prodotto degli autovalori di una matrice è uguale al suo determinante, si ha:

$$\rho(C_\omega) \geq (|\omega - 1|^n)^{1/n} = |\omega - 1|$$

e quindi la tesi. ■

Dal risultato del Teorema precedente, si evince che la condizione $0 < \omega < 2$ è necessaria per la convergenza del metodo SOR. Se la matrice A del sistema lineare da risolvere è simmetrica definita positiva si ha il seguente importante risultato, che è una conseguenza dei Teoremi 4.10.6 e 4.12.1:

Corollario 4.12.1. *Se A è* **simmetrica e definita positiva***, il metodo SOR è convergente se e solo se*

$$0 < \omega < 2.$$

Dimostrazione Che la condizione sia necessaria discende dal Teorema 4.12.1. Si supponga ora che $0 < \omega < 2$. Ricordando che la matrice di splitting del metodo SOR è $M_\omega = \omega^{-1}D + L$, si ha, essendo A simmetrica, per cui è $L^T = U$, che la matrice $Q = M_\omega + M_\omega^T - A = (2\omega^{-1} - 1)D$ è una matrice

diagonale, i cui elementi sono dati da $(2\omega^{-1}-1)a_{ii}$, $i=1,\ldots,n$, dove a_{ii}, $i=1,\ldots,n$, sono gli elementi diagonali di A. Poichè A è definita positiva, i suoi elementi diagonali sono tutti positivi; di conseguenza, anche la matrice Q, poichè $0<\omega<2$, ha tutti gli elementi positivi e, quindi, è definita positiva. In base al Teorema 4.10.6, si ha che il metodo SOR converge. ■

Il problema cruciale relativo all'utilizzo del metodo SOR è la scelta del parametro di rilassamento ω; tale scelta determina la velocità di convergenza del metodo e, quindi, la sua maggiore efficacia rispetto ai metodi di Jacobi e di G-S. Si consideri la seguente:

Definizione 4.12.2. (Valore ottimale)
Relativamente al metodo SOR (4.77), si definisce **valore ottimale di** ω *il valore* ω_{opt} *tale che:*

$$\rho(C_{\omega_{opt}}) = \min_{\omega} \rho(C_\omega).$$

In generale, la determinazione del valore ottimale del parametro di rilassamento del metodo SOR è un problema di non facile soluzione. Tuttavia, per alcune importanti classi di sistemi lineari il valore di ω_{opt} è noto o può essere stimato con sufficiente accuratezza.

♣ **Esempio 4.27.** Si consideri il seguente problema al contorno per l'equazione di Poisson:

$$\begin{cases} -u_{xx}-u_{yy}=f(x,y) & (x,y)\in\Omega=]0,1[\times]0,1[\\ u(x,y)=g(x,y) & (x,y)\in\delta\Omega \end{cases} \tag{4.81}$$

dove $\delta\Omega$ è la frontiera del quadrato unitario, $f(x,y)$ e $g(x,y)$ sono funzioni continue rispettivamante su Ω e $\delta\Omega$. La soluzione numerica di tale problema si può ottenere mediante il *metodo delle differenze finite* utilizzando uno schema a 5 punti. Più precisamente si considera una reticolazione del dominio quadrato $\Omega\cup\delta\Omega$ in quadratini di lato $h=1/(m+1)$ e si vuole calcolare un'approssimazione dei valori che la funzione incognita $u(x,y)$ assume nei punti interni di tale reticolazione (quelli nei punti di frontiera sono noti, nota la g, per la condizione imposta nella (4.81)), cioè nei punti:

$$(x_i,y_j),\ \ x_i=i\cdot h,\ y_j=j\cdot h,\ i,j=1,\ldots,m.$$

A tal fine, nei suddetti punti si approssimano le derivate parziali seconde con le usuali differenze centrali in modo che il problema di partenza (4.81) sia approssimato dal problema discreto:

$$\frac{1}{h^2}(4u_{i,j}-u_{i-1,j}-u_{i+1,j}-u_{i,j+1}-u_{i,j-1})=f_{i,j},\ i,j=1,\ldots,m, \tag{4.82}$$

dove $u_{i,j}$ e $f_{i,j}$ rappresentano i valori approssimati, rispettivamente di $u(x_i,y_j)$ e $f(x_i,y_j)$; inoltre $u_{i,j}$ rappresenta uno dei valori sulla frontiera se i o j è uguale a 0 o $m+1$. Numerando i punti interni della reticolazione (a partire da 1) da sinistra a destra e dal basso verso l'alto e portando a termine noto i valori sui punti della frontiera, posto

$$\mathbf{u}^T=(u_{11},u_{12},\ldots,u_{1m},u_{21},u_{22},\ldots,u_{2m},\ldots,u_{m1},u_{m2},\ldots,u_{mm}),$$

le (4.82) costituiscono un sistema di n equazioni lineari in n incognite, con $n = m^2$:

$$Au = b, \tag{4.83}$$

la cui soluzione rappresenta i valori approssimati della funzione incognita $u(x, y)$ nei punti interni della reticolazione. La matrice dei coefficienti ha la seguente struttura:

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix},$$

dove I è la matrice identità di ordine m e T è la matrice tridiagonale di ordine m:

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

La matrice A è quindi una T-matrice[19] simmetrica, in cui le matrici diagonali, T, sono non singolari. Inoltre, posto:

$$t = \frac{\pi}{m+1} = \pi h, \tag{4.84}$$

[19]

Definizione 4.12.3. (Matrice tridiagonale a blocchi)
Si definisce **matrice tridiagonale a blocchi** *(chiamata* T**-matrice**$)$ *una matrice* $A \in \Re^{n\times n}$ *del tipo:*

$$A = \begin{pmatrix} D_1 & F_1 & & & \\ E_2 & D_2 & F_2 & & \\ & \ddots & \ddots & \ddots & \\ & & E_{p-1} & D_{p-1} & F_{p-1} \\ & & & E_p & D_p \end{pmatrix},$$

dove D_i, E_i, F_i *sono matrici di dimensioni ripsettive* $n_i \times n_i$, $n_i \times n_{i-1}$ *e* $n_i \times n_{i+1}$ *con* $\sum_{i=1}^{p} n_i = n$.

Per le T-matrici valgono i seguenti risultati [31].

Teorema 4.12.2. *Dato un sistema lineare* $Ax = b$, *in cui* A *è una* T*-matrice, con matrici diagonali* D_i *non singolari, si ha:*

$$\rho(C_{GS}) = \rho^2(C_J),$$

e quindi:

$$R_\infty(C_{GS}) = 2R_\infty(C_J).$$

Sulla base del Teorema precedente, si ha che, per sistemi lineari in cui la matrice dei coefficienti è una T-matrice, se il metodo di Jacobi converge, allora converge anche il metodo di G-S con una velocità asintotica doppia.

è noto che gli autovalori della matrice A sono dati da:

$$\lambda_{i,j} = 4\left(\sin^2 i\frac{t}{2} + \sin^2 j\frac{t}{2}\right), \quad i,j = 1,\ldots,m.$$

Dalla conoscenza degli autovalori di A si determinano facilmente anche quelli della matrice di iterazione del metodo di Jacobi applicato al sistema (4.83), che nel caso specifico è data da $C_J = I - D^{-1}A = I - \frac{1}{4}A$. Gli autovalori di C_J sono quindi:

$$\mu_{i,j} = 1 - \frac{1}{4}\lambda_{i,j} = \frac{1}{2}(\cos it + \cos jt), \quad i,j = 1,...,m,$$

da cui, per il raggio spettrale di C_J si ha:

$$\rho(C_J) = \max_{i,j}|\mu_{i,j}| = \cos t < 1. \tag{4.85}$$

In conclusione, per il sistema (4.83) sono verificate tutte le ipotesi del Teorema 4.12.3, per cui il valore ottimale del parametro di rilassamento relativo al metodo SOR è:

$$\omega_{opt} = \frac{2}{1+(1-\cos^2 t)^{1/2}} = \frac{2}{1+\sin t}. \tag{4.86}$$

Come esempio di applicazione del risultato ottenuto, si consideri, in particolare, il caso in cui, in (4.81), si abbia:

$$f(x,y) = 0; \quad g(x,y) = x + y. \tag{4.87}$$

Con facili calcoli si verifica che in tali ipotesi e per $m = 2$, la matrice A ed il vettore b del sistema (4.83) sono:

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}; \quad b = \begin{pmatrix} u_{0,1}+u_{1,0} \\ u_{3,1}+u_{2,0} \\ u_{0,2}+u_{1,3} \\ u_{3,2}+u_{2,3} \end{pmatrix} = \begin{pmatrix} g\left(0,\frac{1}{3}\right)+g\left(\frac{1}{3},0\right) \\ g\left(1,\frac{1}{3}\right)+g\left(\frac{2}{3},0\right) \\ g\left(0,\frac{2}{3}\right)+g\left(\frac{1}{3},1\right) \\ g\left(1,\frac{2}{3}\right)+g\left(\frac{2}{3},1\right) \end{pmatrix} = \begin{pmatrix} 2/3 \\ 2 \\ 2 \\ 10/3 \end{pmatrix}. \tag{4.88}$$

La soluzione di tale sistema è $(2/3, 1, 1, 4/3)$. Inoltre, dalla (4.86) con $t = \frac{\pi}{3}$ si ha che:

$$\omega_{opt} = 1.0718$$

Nella Tabella 4.11 sono riportati i risultati ottenuti risolvendo il sistema (4.83), con A e f dati dalle (4.88), rispettivamente con il metodo di Jacobi, il metodo di G-S ($\omega = 1$) e il metodo SOR con $\omega = \omega_{opt} = 1.0718$. In particolare sono state utilizzate le **Procedure 4.3** e **4.4**, l'ultima opportunamente modificata per introdurre il parametro di rilassamento. Inoltre, si è posto `Tol`$=10^{-6}$ e $x_i^{(0)} = 0$, $i = 1,...,4$. Si osservi che, come si aspettava, la velocità di convergenza del metodo di G-S è quasi il doppio di quella del metodo di Jacobi. Inoltre, il numero di iterazioni (`IT`) richieste dal metodo SOR per soddisfare la tolleranza è inferiore rispetto a quelle richieste dal metodo di G-S.

Teorema 4.12.3. *Dato un sistema lineare $Ax = b$, in cui A è una T-matrice, con matrici diagonali D_i non singolari, se tutti gli autovalori della matrice di iterazione del metodo di Jacobi, C_J, sono reali e se $\rho(C_J) < 1$, allora il metodo SOR converge per $0 < \omega < 2$. Inoltre si ha:*

1) $\omega_{opt} = \frac{2}{1+(1-\rho^2(C_J))^{1/2}}$

2) $\rho(C_{\omega_{opt}}) = \omega_{opt} - 1.$

Metodo	IT	x_1	x_2	x_3	x_4	ERR
Jacobi	20	0.666674	0.999999	0.999999	1.33332	$0.71{\cdot}10^{-6}$
G-S	12	0.666674	0.999999	0.999999	1.33332	$0.49{\cdot}10^{-6}$
SOR	8	0.666674	1.000000	1.000000	1.33333	$0.76{\cdot}10^{-6}$

Tabella 4.11: Schema riassuntivo relativo alla risoluzione del sistema (4.83), con A e f dati dalle (4.88), con i metodi di Jacobi, G-S e SOR ($\omega = 1.0718$).

È da osservare che il miglioramento in velocità di convergenza, usando $\omega = \omega_{opt}$ nel metodo SOR, aumenta al crescere di m e, quindi, della dimensione del sistema. Infatti, per m abbastanza grande, e quindi, dalla (4.84), per $t << 1$, si ha che una stima accurata del raggio spettrale della matrice di iterazione del metodo SOR è data da:

$$\rho(C_{\omega_{opt}}) = \omega_{opt} - 1 = \frac{2}{1+\sin t} - 1 \simeq 1 - 2t.$$

Di conseguenza, una stima della velocità di convergenza del metodo SOR con $\omega = \omega_{opt}$ è:

$$R_{\omega_{opt}} \simeq -\log(1-2t) \simeq 2t. \tag{4.89}$$

Per il metodo di G-S si ha:

$$\rho(C_G) = \rho^2(C_J) = \cos^2 t \simeq 1 - t^2,$$

da cui la velocità di convergenza del metodo di G-S è:

$$R_G \simeq -\log(1-t^2) \simeq t^2.$$

Confrontando le velocità di convergenza dei due metodi, si ha che, per m abbastanza grande, il metodo SOR con valore ottimale del parametro di rilassamento richiede, rispetto al metodo di G-S, un numero di iterazioni minore di un fattore dato da:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq \frac{2}{t} \simeq \frac{2m}{\pi}.$$

Tale fattore è proporzionale a m, cioè aumenta al crescere di m. Ad esempio, se $m = 5$ si ha:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq 3,$$

mentre se $m = 10$ si ha:

$$\frac{R_{\omega_{opt}}}{R_G} \simeq 6\ .$$

In pratica, ciò significa che, se $m = 5$, il metodo SOR con parametro ottimale è circa 3 volte più veloce del metodo di G-S, mentre, se $m = 10$, circa 6. In Tabella 4.12 è riportato, al variare di m da 2 a 10, il numero di iterazioni richiesto dai metodi di Jacobi, G-S e SOR con parametro ottimale, per risolvere il sistema (4.83) derivante dal problema differenziale (4.81), con le funzioni f e g date dalle (4.87) ed utilizzando Tol=10^{-6}. I risultati ottenuti confermano le stime teoriche sul guadagno, in termini di velocità di convergenza, che si ha utilizzando il metodo SOR con il valore ottimale del parametro di rilassamento.

♣

m	n	IT (Jacobi)	IT (G-S)	IT (SOR)
2	4	20	12	8
3	9	38	21	12
5	25	84	45	18
7	49	142	77	24
9	81	214	116	30
10	100	254	138	32

Tabella 4.12: Risultati relativi alla risoluzione del sistema (4.83) al variare di m, utilizzando i metodi di Jacobi, G-S e SOR con parametro ottimale.

È importante osservare ancora che l'utilizzazione pratica dell'espressione analitica del valore ottimale del parametro di rilassamento fornita dal Teorema 4.12.3, dipende, tuttavia, dalla possibilità di disporre di una stima del raggio spettrale della matrice di iterazione del metodo di Jacobi. Tale stima, inoltre, a parte qualche caso particolare come quello illustrato nell'esempio 4.27, non è nota, e spesso calcolarla è computazionalmente molto dispendioso[20].

4.12.2 Accelerazione polinomiale

In questo paragrafo si introduce un'altra tecnica per accelerare la convergenza dei metodi iterativi del tipo (4.50). Tale tecnica, nota con il nome di **procedura di accelerazione polinomiale**, consiste nel costruire una nuova sequenza di vettori mediante combinazioni lineari dei vettori ottenuti con un fissato metodo iterativo, che è definito **metodo iterativo base** della procedura. Consideriamo il seguente problema.

Problema: formulazione (I)
Sia $\{x^{(k)}\}$ la successione di vettori generati, a partire da un vettore iniziale $x^{(0)}$, da un metodo base:

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0. \tag{4.90}$$

[20]Per tale motivo, la maggior parte delle principali librerie di software matematico, che contengono routine per la risoluzione di sistemi lineari con il metodo SOR, prevedono procedure *adattative* per il calcolo di ω_{opt}, cioè algoritmi numerici che determinino dinamicamente, ad ogni iterazione, una stima di ω_{opt}.

Si consideri la nuova sequenza di vettori $\{y^{(k)}\}$ *definiti dalle combinazioni lineari:*

$$y^{(k)} = \sum_{j=0}^{k} \alpha_{jk} x^{(j)}, \quad \alpha_{jk} \in \Re, \quad k \geq 0. \tag{4.91}$$

Si vogliono determinare i coefficienti α_{jk} *in modo che* $y^{(k)}$ *sia un'approssimazione più accurata della soluzione del sistema rispetto a* $x^{(k)}$*.*

Indicata con x^* la soluzione del sistema $Ax = b$, si osserva, innanzitutto, che se $x^{(0)} = x^{(1)} = ... = x^{(k)} = x^*$, allora deve risultare $y^{(k)} = x^*$. Sui coefficienti α_{jk} si impone quindi il seguente vincolo:

$$\sum_{j=0}^{k} \alpha_{jk} = 1, \quad \forall k \geq 0. \tag{4.92}$$

Tenendo conto della (4.92), il problema puó essere cosí riformulato

Problema: formulazione (II)

Indicati con

$$e^{(k)} = x^{(k)} - x^* \quad ed \quad \tilde{e}^{(k)} = y^{(k)} - x^*,$$

gli errori associati rispettivamente a $x^{(k)}$ *e ad* $y^{(k)}$*, per le (4.91) e (4.92) si ha:*

$$\tilde{e}^{(k)} = \sum_{j=0}^{k} \alpha_{jk} x^{(j)} - x^* = \sum_{j=0}^{k} \alpha_{jk} (x^{(j)} - x^*) = \sum_{j=0}^{k} \alpha_{jk} e^{(j)}. \tag{4.93}$$

Ricordando che:

$$e^{(j)} = C^j e^{(0)}, \ \ j \geq 0,$$

con $e^{(0)} = x^{(0)} - x^*$*, dalla (4.93) si ha:*

$$\tilde{e}^{(k)} = \left(\sum_{j=0}^{k} \alpha_{jk} C^j \right) e^{(0)}, \quad \forall k \geq 0, \tag{4.94}$$

dove $\tilde{e}^{(0)} = e^{(0)}$*. Introdotto il polinomio di grado k nella variabile reale z:*

$$p_k(z) = \alpha_{0,k} + \alpha_{1,k} z + ... + \alpha_{k,k} z^k, \ \ con \ \ p_k(1) = 1,$$

la (4.94) può essere riscritta nella forma:

$$\tilde{e}^{(k)} = p_k(C) \tilde{e}^{(0)}, \quad \forall k \geq 0, \tag{4.95}$$

dove $p_k(C) \in \Re^{n \times n}$ *ha la seguente espressione:*

$$p_k(C) = \alpha_{0,k} I + \alpha_{1,k} C + ... + \alpha_{k,k} C^k \ .$$

Il problema fondamentale legato all'utilizzo della procedura di accelerazione descritta è quello di trovare una successione di polinomi

$$\{p_k(z) \ : \ p_k(1) = 1\}$$

in modo da rendere minimo l'errore $\tilde{e}^{(k)}$, $\forall k \geq 0$. *Considerando la norma euclidea, dalla (4.95) si ha:*

$$\|\tilde{e}^{(k)}\|_2 \leq \|p_k(C)\|_2 \|\tilde{e}^{(0)}\|_2, \quad \forall k \geq 0,$$

e, quindi, per rendere piccola la norma dell'errore è necessario determinare una successione di polinomi $\{p_k(z) \ : \ p_k(1) = 1\}$ *tali che* $\|p_k(C)\|_2$ *sia piccola. Ciò conduce al seguente problema:*

$$\mathbf{P2}: \quad min_{p_k(1)=1} \|p_k(C)\|_2, \ \ \forall k \geq 0. \tag{4.96}$$

Si analizza qui di seguito una possibile soluzione di tale problema.

4.12.3 Accelerazione polinomiale di Chebyshev

Nel caso in cui la matrice C sia simmetrica si ha una ulteriore formulazione del problema.

Problema: formulazione (III)

Se la matrice di iterazione C *del metodo base (4.90) della procedura è simmetrica, i suoi autovalori* $\{\lambda_i\}_{i=1}^n$ *sono reali e, nell'ipotesi che il metodo base converga (*$\rho(C) < 1$*), si ha:*

$$-1 < a = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n = b < 1, \tag{4.97}$$

con a *e* b*, rispettivamente, il più piccolo ed il più grande tra gli autovalori di* C*. Si osservi che* $\{p_k(\lambda_i)\}_{i=1}^n$ *sono gli autovalori della matrice* $p_k(C)$ *(vedi* **Teorema B.18** *dell'***Appendice B**, *della* **Parte prima**). *Inoltre, dal fatto che* C *è simmetrica discende che lo è anche* $p_k(C)$*. Si ha quindi:*

$$\|p_k(C)\|_2 = \rho(p_k(C)) = \max_{1 \leq i \leq n} |p_k(\lambda_i)|.$$

Dalla (4.97) si ha:

$$\|p_k(C)\|_2 = \max_{1 \leq i \leq n} |p_k(\lambda_i)| \leq \max_{a \leq z \leq b} |p_k(z)|.$$

Di conseguenza, il problema **P2** *è ricondotto ad un nuovo problema: la determinazione di una successione di polinomi* $\{p_k(z) \ : \ p_k(1) = 1\}$ *che risolvano:*

$$\mathbf{P3}: \min_{p_k(1)=1} \left\{ \max_{a \leq z \leq b} |p_k(z)| \right\}, \ \ \forall k \geq 0. \tag{4.98}$$

La soluzione del problema è unica ed è definita in termini dei **polinomi di Chebyshev**, come mostra il Teorema che segue, la cui dimostrazione (come anche le principali proprietà dei polinomi di Chebyshev) è riportata in Appendice (Teorema C.2.2).

Teorema 4.12.4. *Sia $T_k(z)$ il polinomio di Chebyshev di prima specie di grado k. Si consideri la funzione $w : [a,b] \longrightarrow [-1,1]$ così definita:*

$$w(z) = \frac{2z-(b+a)}{b-a}.$$

Sia $\mathcal{P}_k$ l'insieme dei polinomi $p_k(z)$ di grado k nella variabile reale z tali che $p_k(1) = 1$. Il polinomio $H_k(z)$ definito da:

$$H_k(z) = \frac{T_k(w(z))}{T_k(w(1))}, \quad \forall k \geq 0, \tag{4.99}$$

è l'unico polinomio $\in \mathcal{P}_k$ che risolve il problema (4.98).

Il metodo iterativo base (4.90) e le (4.91), dove i coefficienti α_{jk} sono quelli dei polinomi (4.99), rappresentano la **procedura di accelerazione di Chebyshev**. Affinchè tale procedura costituisca un metodo effettivamente utilizzabile, è necessario tuttavia determinare una formula più efficiente della (4.91) per calcolare i vettori $y^{(k)}$; in tale formula, infatti, $y^{(k)}$ si ottiene a partire dai $k+1$ vettori $x^{(j)}$, $j = 0, ..., k$. Pertanto, l'uso della (4.91) comporta un'elevata complessità sia di tempo che di spazio quando k è grande. Sfruttando la nota relazione di ricorrenza a tre termini dei polinomi di Chebyshev (Teorema C.2.1 dell'Appendice) si può ottenere una formula più conveniente della (4.91) da un punto di vista computazionale. Si ha infatti il seguente Teorema, la cui dimostrazione è riportata in Appendice (Teorema C.2.3):

Teorema 4.12.5. *I polinomi definiti in (4.99) soddisfano la relazione di ricorrenza:*

$$\begin{cases} H_0(z) = 1, \quad H_1(z) = \gamma z + 1 - \gamma \ ; \\ H_{k+1}(z) = \beta_{k+1}(\gamma z + 1 - \gamma)H_k(z) + (1-\beta_{k+1})H_{k-1}(z), \quad k \geq 1, \end{cases} \tag{4.100}$$

dove:

$$\begin{cases} \gamma = 2/(2-(b+a)) \ ; \\ \beta_{k+1} = 2w(1)T_{(k)}(w(1))/T_{(k+1)}(w(1)) \ . \end{cases} \tag{4.101}$$

Dalla relazione (4.100) stabilita dal Teorema precedente segue che anche i valori $y^{(k)}$ della procedura di accelerazione basata sui polinomi $H_{(k)}(z)$ possono essere ottenuti mediante una formula di ricorrenza a tre termini. Si ha infatti il seguente:

Corollario 4.12.2. *I vettori $y^{(k)}$ della procedura di accelerazione di Chebyshev si possono calcolare, a partire da un arbitrario vettore $y^{(0)}$, mediante la formula:*

$$\begin{cases} y^{(1)} = \gamma(Cy^{(0)} + d) + (1-\gamma)y^{(0)} \ ; \\ y^{(k+1)} = \beta_{k+1}[\gamma(Cy^{(k)} + d) + (1-\gamma)y^{(k)}] + (1-\beta_{k+1})y^{(k-1)} \ . \end{cases} \tag{4.102}$$

Dimostrazione Ricordando la (4.91), $y^{(k)} = \sum_{j=0}^{k} \alpha_{jk} x^{(j)}$, si ricava che $y^{(0)} = x^{(0)}$. Inoltre, dalle (4.100) si ha $H_1(x(1)) = \gamma x^{(1)} + 1 - \gamma$. Ma, d'altra parte, $H_1(x(1)) = \alpha_{10} + \alpha_{11} x^{(1)}$, da cui si ha $\alpha_{11} = \gamma$ e $\alpha_{01} = 1 - \gamma$. Poichè $y^{(1)} = \alpha_{01} x^{(0)} + \alpha_{11} x^{(1)}$, e tenuto conto che $x^{(1)} = Cx^{(0)} + d$, si ha la prima uguaglianza delle (4.102).
Si consideri ora l'errore associato al vettore $y^{(k)}$ in (4.91). Dalla (4.95) e dalla (4.100) si ha:

$$\tilde{e}^{(k+1)}(z) = \{\beta_{k+1}[\gamma C + (1-\gamma)I]H_k(C) + (1-\beta_{k+1})H_{k-1}(C)\}\tilde{e}^{(0)},$$

da cui, utilizzando ancora la (4.95), si ha:

$$\tilde{e}^{(k+1)}(z) = \beta_{k+1}[\gamma C + (1-\gamma)I]\tilde{e}^{(k)} + (1-\beta_{k+1})\tilde{e}^{(k-1)}. \tag{4.103}$$

Aggiungendo x^* ad entrambi i termini della (4.103) si ottiene:

$$y^{(k+1)}(z) = \beta_{k+1}[\gamma C + (1-\gamma)I]y^{(k)} + (1-\beta_{k+1})y^{(k-1)} - \beta_{k+1}\gamma(C-I)x^*. \tag{4.104}$$

Poichè x^* è soluzione del sistema associato al metodo di base (4.90), cioè $x^* = Cx^* + d$, si ha $(C-I)x^* = -d$, per cui dalla (4.104) si ottiene la relazione (4.102). ■

Osservazione 4.4. *I coefficienti β_{k+1} della relazione di ricorrenza (4.102) sono determinati dalla formula (4.101). Si osserva, tuttavia, che tali coefficienti possono essere scritti nella forma seguente, più vantaggiosa da un punto di vista computazionale:*

$$\beta_1 = 1,\ \beta_2 = (1 - \tfrac{1}{2}\sigma^2)^{-1}$$
$$\beta_{k+1} = (1 - \tfrac{1}{4}\sigma^2\beta_k)^{-1},\ \ k \geq 2,$$

con:

$$\sigma = 1/(w(1)) = (b-a)/(2-(b+a)).$$

La relazione (4.102) mostra che non è necessario utilizzare un vettore ausiliario per memorizzare i vettori $x^{(k)}$ del metodo base (4.90). Infatti, anche se l'idea di partenza della procedura di accelerazione è quella di ottenere i vettori $y^{(k)}$ come combinazioni dei vettori $x^{(k)}$, la (4.102) consente di determinare direttamente gli $y^{(k)}$ in un modo simile al calcolo dei vettori $x^{(k)}$ nel metodo base. Si nota, tuttavia, che, mentre $x^{(k+1)}$ dipende solo da $x^{(k)}$, il vettore $y^{(k+1)}$ è ottenuto utilizzando i vettori calcolati nelle utlime due iterazioni, $y^{(k)}$ e $y^{(k-1)}$. Per tale motivo la procedura descritta è un **metodo iterativo a due passi**. Per la presenza di coefficienti il cui valore non è costante, ma dipende dall'iterazione (β_k), la procedura, al contrario dei metodi analizzati fino ad ora, appartiene alla classe dei **metodi non stazionari**. Inoltre, essa richiede l'uso di un vettore in più rispetto al metodo base. Ciò può pesare in maniera sensibile quando si vogliono risolvere sistemi lineari di grandi dimensioni[21].

[21] È da osservare, infine, che la procedura di accelerazione in esame richiede anche la conoscenza del più piccolo e del più grande autovalore della matrice di iterazione del metodo base. Nella maggior parte dei casi tali quantità non sono note, ed il loro calcolo è, in generale, computazionalmente dispendioso. Per tale motivo, sono state sviluppate opportune versioni della procedura, nelle quali si utilizzano solo delle stime dei precedenti valori.

4.13 Criteri di arresto

Nel paragrafo 4.4 si è analizzato un criterio per arrestare un metodo iterativo, il quale è basato su una stima dell'errore. Si ritorna su tale argomento con l'obiettivo di discutere dei concetti di base per la progettazione di criteri di arresto affidabili, cioè criteri il cui utilizzo consenta di ottenere una desiderata accuratezza.
In generale, l'effettivo utilizzo di un metodo iterativo richiede l'arresto del procedimento quando l'errore è divenuto sufficientemente piccolo. Poichè non è possibile calcolare l'errore in maniera diretta dal momento che la soluzione non è nota, per progettare un criterio di arresto soddisfacente occorre stimare l'errore ad ogni iterazione.
Si premette la seguente definizione:

Definizione 4.13.1. (Residuo)
Dato il sistema lineare $Ax = b$, sia $\bar{x}$ un vettore di dimensione n. Si definisce **residuo** *del sistema in $\bar{x}$ il vettore:*

$$r = b - A\bar{x}.$$

Si osservi che una norma del residuo è una misura di quanto il vettore $\bar{x}$ soddisfa il sistema.

Uno dei criteri di arresto più utilizzato si basa su una stima di quanto il residuo all'iterazione generica si è ridotto rispetto al residuo iniziale, cioè:

$$\frac{\|r^{(k)}\|}{\|r^{(0)}\|} \leq \texttt{Tol}, \tag{4.105}$$

dove $r^{(k)}$ è il residuo del sistema $Ax = b$ in $x^{(k)}$. Il Teorema che segue stabilisce la relazione che sussiste tra l'errore e il residuo alla generica iterazione.

Teorema 4.13.1. *Alla k-ma iterazione del metodo iterativo:*

$$x^{(k+1)} = Cx^{(k)} + d, \quad k \geq 0,$$

si ha:

$$\|x^{(k)} - x^*\| \leq \|A^{-1}\| \cdot \|r^{(k)}\|, \tag{4.106}$$

dove $r^{(k)}$ è il residuo in $x^{(k)}$. Inoltre, se $x^{(0)}$ è il vettore iniziale, si ha:

$$\frac{\|x^{(k)} - x^*\|}{\|x^{(0)} - x^*\|} \leq \mu(A) \cdot \frac{\|r^{(k)}\|}{\|r^{(0)}\|}, \tag{4.107}$$

dove $\mu(A) = \|A\| \cdot \|A^{-1}\|$ è l'indice di condizionamento della matrice A.

Dimostrazione Da

$$r^{(k)} = b - Ax^{(k)}, \tag{4.108}$$

si ha:

$$x^{(k)} - x^* = -A^{-1}r^{(k)},$$

da cui segue la (4.106). Inoltre, dalla (4.108), per $k = 0$ si ha:

$$r^{(0)} = -A(x^{(0)} - x^*),$$

da cui segue che:

$$\|r^{(0)}\| = \|A(x^{(0)} - x^*)\| \leq \|A\| \cdot \|x^{(0)} - x^*\|.$$

Dall'ultima relazione e dalla (4.106) si ricava che:

$$\frac{\|x^{(k)} - x^*\|}{\|x^{(0)} - x^*\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|r^{(k)}\|}{\|r^{(0)}\|}$$

e, quindi, la (4.107). ■

Si osservi, innanzitutto, che la (4.107) mette in relazione la riduzione dell'errore alla generica iterazione con quella del residuo. Inoltre, in base alla (4.106), se si arresta il metodo quando vale la (4.105), per l'errore assoluto si ha la seguente maggiorazione:

$$\|x^{(k)} - x^*\| \leq \|A^{-1}\| \cdot \|r^{(0)}\| \cdot \texttt{Tol}. \tag{4.109}$$

In generale, l'utilizzo del criterio (4.105) richiede ad ogni iterazione il calcolo del prodotto tra una matrice ed un vettore, e quindi un costo computazionale per iterazione circa uguale a n^2 flops. Tuttavia, per alcuni metodi, il residuo ad ogni iterazione è disponibile senza costi aggiuntivi. Ad esempio, per il metodo di Jacobi, il residuo al passo k può essere ottenuto senza dover calcolare un prodotto matrice-vettore; infatti per la i-ma componente del residuo si ha:

$$r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)} = b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)} - a_{ii}x_i^{(k)} = a_{ii}(x_i^{(k+1)} - x_i^{(k)}).$$

Si può verificare che per il metodo di G-S il residuo assume un'analoga espressione.

Osservazione 4.5. *È importante osservare che, anche se criteri di arresto basati sul residuo, come (4.105), sono utilizzati frequentemente, un residuo piccolo non comporta automaticamente un errore piccolo. Ciò è dovuto in generale al fatto che, come stabilito dal Teorema 4.13.1, la relazione tra l'errore e il residuo dipende dal condizionamento della matrice A.*

♣ **Esempio 4.28.** Per illustrare che la grandezza del residuo, $\|r^{(k)}\|$, può non essere un affidabile indicatore della grandezza dell'errore, si consideri il sistema lineare:

$$\begin{cases} 5x_1 + 0.0001x_2 + 0.0001x_3 = 5.0002 \\ 0.0001x_1 + 2.25000001x_2 + 0.00000001x_3 = 2.25010002 \\ 0.0001x_1 + 0.00000001x_2 + 0.00000001x_3 = 0.00010002 \end{cases}$$

la cui soluzione è $x_i^* = 1$, $i = 1, 2, 3$. Nella Tabella che segue sono riportati i valori di $\|r^{(k)}\|$ e $\|x^{(k)} - x^*\|$ nelle prime iterazioni del metodo di G-S (in particolare si è utilizzata la norma euclidea) partendo da $x_i^0 = 0$, $i = 1, 2, 3$. Si evince che ad un valore piccolo di $\|r^{(k)}\|_2$ non necessariamente corrisponde un errore con un comparabile ordine di grandezza. Poichè $\|A^{-1}\|_2 = 10^8$, dalla (4.106) si ha, per l'errore, la maggiorazione $\|x^{(k)} - x^*\|_2 \leq 10^8 \cdot \|r^{(k)}\|_2$, che giustifica i risultati ottenuti.

k	$\|r^{(k)}\|_2$	$\|x^{(k)} - x^*\|_2$
1	$1.60 \cdot 10^{-4}$	$0.40 \cdot 10^{0}$
2	$3.20 \cdot 10^{-5}$	$8.00 \cdot 10^{-2}$
3	$6.40 \cdot 10^{-6}$	$1.60 \cdot 10^{-2}$
4	$1.28 \cdot 10^{-6}$	$3.20 \cdot 10^{-3}$
5	$2.56 \cdot 10^{-7}$	$6.40 \cdot 10^{-4}$
6	$5.12 \cdot 10^{-8}$	$1.28 \cdot 10^{-4}$
7	$1.02 \cdot 10^{-8}$	$2.56 \cdot 10^{-5}$
8	$2.04 \cdot 10^{-9}$	$5.12 \cdot 10^{-6}$
9	$4.09 \cdot 10^{-10}$	$1.02 \cdot 10^{-6}$
10	$8.19 \cdot 10^{-11}$	$2.04 \cdot 10^{-7}$

♣

Si osservi, infine, che l'affidabilità del criterio di arresto (4.105) dipende anche dalla scelta di $x^{(0)}$. Infatti, se $x^{(0)}$ è "vicino" alla soluzione, la quantità $\|r^{(0)}\|$ può essere molto piccola e si corre il rischio di eseguire iterazioni inutili, cioè che non migliorano l'accuratezza del risultato; viceversa, se il punto iniziale è "distante" dalla soluzione, e quindi $\|r^{(0)}\|$ è un numero grande, il metodo può essere arrestato "troppo presto", ottenendo un'approssimazione poco accurata della soluzione.

♣ **Esempio 4.29.** Si consideri il sistema di equazioni lineari:

$$\begin{cases} 10000x_1 + 1x_2 + 4x_3 = 10005 \\ 0.5x_1 + 1x_2 + 0.4x_3 = 1.9 \\ 0.2x_1 + 0.6x_2 + 0.3x_3 = 1.1 \end{cases}$$

la cui soluzione è $x_i^* = 1$, $i = 1, 2, 3$. Applicando a tale sistema il metodo di Jacobi con $x_i^{(0)} = 0, i = 1, 2, 3$ ed utilizzando come criterio di arresto la (4.105) con `Tol`$= 10^{-6}$, il metodo si ferma dopo 71 iterazioni e gli ultimi valori calcolati, arrotondati alla sesta cifra decimale, sono: $x_1^{(71)} = 1.000000$, $x_2^{(71)} = 1.000380$, $x_3^{(71)} = 1.001114$. Si osserva che per le componenti x_2 e x_3 non si è ottenuta l'accuratezza desiderata. Poichè $\|r^{(0)}\|_2 \simeq 10005$ e $\|A^{-1}\|_2 \simeq 21.119$, dalla (4.109) si ha, per l'errore, la maggiorazione $\|x^{(71)} - x^*\|_2 \leq 0.211$. Si osservi che $\|x^{(71)} - x^*\|_2 = 0.001177$. ♣

4.14 Un cenno ai metodi iterativi non stazionari basati sui sottospazi di Krylov

Dalla (4.55) segue che un metodo iterativo stazionario ad un passo del tipo:

$$x^{(k+1)} = C\,x^{(k)} + d$$

è esprimibile anche nella forma:

$$x^{(k+1)} = (I - M^{-1}A)x^{(k)} + M^{-1}b = x^{(k)} + M^{-1}(b - A\,x^{(k)})$$

ovvero, posto $r^{(k)} = b - A\,x^{(k)}$, come:

$$x^{(k+1)} = x^{(k)} + M^{-1}r^{(k)} \tag{4.110}$$

dove M è la matrice derivante dallo splitting (4.53) di A.
La (4.110), nel caso particolare in cui M sia la matrice identica, prende il nome di *iterazione di Richardson.* Si noti che, a partire dall'iterazione di Richardson, segue:

$$b - A\,x^{(k+1)} = b - A\,x^{(k)} - A\,r^{(k)}$$

cioè:

$$r^{(k+1)} = (I - A)r^{(k)} = (I - A)^{(k+1)}r^{(0)}$$

ovvero, il *residuo* $r^{(k)}$ si esprime come il polinomio in A:

$$p_{k+1}(A) = (I - A)^{k+1}$$

di grado $k+1$, moltiplicato per il residuo iniziale. Questa è una proprietà comune a molti metodi iterativi, ed è particolarmente utile per derivare tecniche di accelerazione della convergenza. Sostituendo questa espressione del residuo nell'iterazione di Richardson, si ha:

$$x^{(k+1)} = x^{(k)} + r^{(k)} = x^{(k)} + (I - A)^{(k)}r^{(0)}$$

da cui, esprimendo a sua volta ciascuna iterata in funzione della precedente, si ha:

$$x^{(k+1)} = r^0 + r^{(1)} + r^{(2)} + \ldots + r^{(k)} = \sum_{j=0}^{k}(I - A)^j r^{(0)}$$

ovvero, l'iterata al passo $k+1$ appartiene al sottospazio

$$K_k = \{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{(k)}r^{(0)}\}$$

generato da A e da $r^{(0)}$. Tale sottospazio, di dimensione k, generato dai vettori

$$r^{(0)}, Ar^{(0)}, \ldots, A^{(k)}r^{(0)},$$

è detto *sottospazio di Krylov*.

L'iterazione di Richardson genera, quindi, al generico passo k, un'approssimazione della soluzione del sistema $Ax = b$, appartenente al sottospazio di Krylov di dimensione k. In particolare, invece della base standard, costruita a partire dal residuo iniziale, si costruiscono basi ortonormali, perché sono più convenienti da un punto di vista computazionale. L'ortonormalizzazione dei vettori della base, tipicamente basato sul procedimento di Gram-Schmidt modificato, conduce a relazioni di ricorrenza anche su tali vettori, e il metodo iterativo che ne deriva risulta, quindi, **non stazionario**, del tipo:

$$x^{(k+1)} = x^{(k)} + \alpha_{k+1} r^{(k)}$$

dove i coefficienti α_{k+1} sono scelti in modo da costruire *in qualche senso* la migliore approssimazione possibile, a ogni passo. A questa classe di metodi appartengono ad esempio i metodi del **Gradiente Coniugato** (CG) e del GMRES (**Generalized Minimal Residual**). Nel Gradiente Coniugato l'iterata al passo k viene calcolata come segue:

$$x^{(k+1)} = x^{(k)} + \alpha_{k+1}\, p^{(k+1)}$$

dove:

$$p^{(k+1)} = r^{(k+1)} + \beta_{k+1} p^{(k)}, \quad r^{(k+1)} = r^{(k)} - \alpha_{k+1} A p^{(k+1)}$$

con:

$$\alpha_{k+1} = \frac{p^{(k)T} \cdot r^{(k)}}{p^{(k+1)T} A p^{(k+1)}}, \quad \beta_{k+1} = -\frac{r^{(k+1)T} \cdot A p^{(k+1)}}{p^{(k)T} A p^{(k)}}$$

e $p^{(0)} = r^{(0)}$. Con tale scelta di α_{k+1} e di β_{k+1} al passo $k+1$ il residuo $r^{(k+1)}$ risulta ortogonale a tutti quelli precedenti ed inoltre $x^{(k+1)}$ viene individuato come il vettore che nel sottospazio di Krylov costruito al passo $k+1$ minimizza la A-norma[22] dell'errore al passo $k+1$, $e^{(k+1)} = x^{(k+1)} - x^\star$, dove con $x^\star$ abbiamo indicato la soluzione del sistema $Ax = b$.

Se la matrice non è simmetrica e non è neanche definita positiva, l'ortogonalizzazione dei vettori della base al passo $k+1$, porta a relazioni di ricorrenza con i vettori costruiti alle iterazioni precedenti, più articolate[23], come ad esempio nel caso del GMRES, in cui l'approssimazione $x^{(k)}$ viene costruita in modo da minimizzare, tra tutti i vettori appartenenti al sottospazio di Krylov di dimensione $k+1$, la norma euclidea del residuo $r^{(k+1)}$.

[22] Data una matrice $A \in \Re^{n\times n}$ simmetrica e definita positiva, ed un vettore $z \in \Re^n$, si definisce A-norma del vettore z la quantità $z^T A z$.

[23] Ad esempio, si usano tecniche di *restart*, ovvero la soluzione ottenuta dopo un prefissato numero di passi viene utilizzata come vettore iniziale per lo stesso algoritmo.

4.15 Software matematico disponibile per i metodi iterativi

Descriveremo il software matematico disponibile per i metodi iterativi classificandolo secondo la convenzione introdotta da *J. Rice* in *Numerical methods, Software and Analysis*, Academic Press, 1993 (cfr. **Parte prima, Cap. 3**). Parliamo, dunque, di:

1. routine individuali, o eventualmente raccolte in collezioni, pubblicate su riviste di software specializzate, ad esempio `TOMS` [28] e *JCAM* [9];

2. packages di routine che risolvono problemi in una specifica area computazionale; approfondiremo, a tal proposito, la libreria `SPARSKIT` [20], di dominio pubblico, specifica per la risoluzione di problemi di algebra lineare che coinvolgono matrici sparse;

3. packages di routine di base, ad esempio `SLATEC` [6], di dominio pubblico;

4. librerie di software *general - purpouse*, di tipo commerciale, come le librerie del `NAg` [13] e la `IMSL` [33], oppure le librerie *open-source*, `PETSc` [16] e `TRILINOS` [30]; queste ultime due, in particolare, contengono moduli mirati alla risoluzione di sistemi lineari mediante i più noti metodi iterativi, per i quali è prevista, inoltre, un'ampia scelta di precondizionatori.

5. Ambienti di risoluzione di problemi (PSE), ad esempio `MATLAB` [34] che mette a disposizione funzioni che implementano i più noti metodi iterativi; esistono, inoltre, pacchetti specifici sviluppati in `MATLAB` e di dominio pubblico, che contengono routine per l'implementazione di metodi iterativi e l'interazione ed il confronto con metodi diretti.

Analizziamo, brevemente, le singole classi.

1. Tra le routine dell'ACM-TOMS ne citiamo solo alcune destinate alla risoluzione numerica di sistemi di equazioni lineari mediante metodi iterativi. La routine `LSQR`, sviluppata in Fortran nel 1982, è finalizzata alla risoluzione di sistemi lineari sparsi, sottodeterminati o sovradeterminati e problemi di minimi quadrati.
   Fa parte di `CALGO` anche `ITPACK 2C` [10], un Package FORTRAN per la risoluzione di sistemi lineari sparsi e di grandi dimensioni, mediante metodi iterativi *adattativi accelerati* (*Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods*). Il package, pubblicato nel 1982, include routine che implementano metodi iterativi classici precondizionati, tra i quali: JCG, JSI, SOR, SSORCG, SSORSI, RSCG e RSSI.
   `PREQN` [17] è una collezione di subroutine, scritte in Fortran 77 e pubblicate nel 2001 sul TOMS, destinate al calcolo di precondizionatori per il metodo del Gradiente Coniugato (CG), quando quest'ultimo è applicato ad un insieme di sistemi lineari con matrici dei coefficienti simmetriche e definite positive.

Ricordiamo, infine, la libreria `sparse BLAS` (*Sparse Basic Linear Algebra Subprograms*) [22] scritta in Fortran 95 e pubblicata nel 2002, che contiene routine per operazioni di base tra vettori e matrici sparse.

2. `SPARSKIT` è un package per la risoluzione di problemi di algebra lineare con matrice sparsa.

   Sul sito di Netlib sono disponibili, inoltre, packages finalizzati alla risoluzione di sistemi lineari con matrice dei coefficienti sparsa. Ne citiamo solo alcuni.

   slap (*The Sparse Linear Algebra Package*) [21] è un package per la risoluzione di sistemi lineari di notevoli dimensioni, sparsi, con matrice dei coefficienti simmetrica o non simmetrica, definita positiva, mediante metodi iterativi precondizionati. Il software include routine che implementano metodi iterativi noti, tra i quali il metodo del Gradiente Coniugato (CG) precondizionato, del Gradiente Coniugato per le equazioni normali (CGNE) precondizionato, del Gradiente Bi-Coniugato (BiCG) precondizionato ed il metodo del minimo residuo generalizzato (GMRES).

   Nella directory di Netlib `linalg` [35] è disponibile **psblas1.0**, per l'implementazione parallela di solutori iterativi per sistemi lineari sparsi; sono implementati, in particolare, metodi iterativi basati su sottospazi di Krylov, per calcolatori a memoria distribuita. Il software è implementato in Fortran 90/Fortran 77 e C e richiede l'interfaccia BLACS per le comunicazioni in ambiente parallelo (*BLACS message passing*).

   Sempre di pubblico dominio è il package **UMFPACK** [4], un insieme di routine per la risoluzione di sistemi lineari non-simmetrici sparsi. Il software è scritto in ANSI/ISO C, con una interfaccia in `MATLAB` (Versione 6.0 e successive).

3. `SLATEC` è una libreria di dominio pubblico costituita da routine di base, scritte in Fortran 77. In particolare un capitolo, indicato con `D`, è dedicato all'algebra lineare e si articola in sezioni che riguardano:

   `D1` operazioni elementari tra vettori e matrici,

   `D2` soluzione di sistemi di equazioni lineari,

   `D3` calcolo del determinante,

   `D4` autovalori ed autovettori,

   `D5` decomposizione QR ed ortogonalizzazione di Gram-Schmidt,

   `D6` decomposizione ai valori singolari (SVD),

`D7` aggiornamento di decomposizioni matriciali,

`D8` risoluzione di sistemi di equazioni sovradeterminati o sottodeterminati e calcolo della pseudo-inversa di una matrice.

Della sezione `D2` fanno parte le routine di `SLATEC` che implementano metodi iterativi (Jacobi, Gauss-Seidel, GMRES, CG, BiCG), con o senza precondizionatore, da applicare a sistemi lineari con matrice dei coefficienti reale *sparsa* (simmetrica o non simmetrica).

4. La libreria NAg del Numerical Algorithm Group di Oxford è sicuramente una fonte molto ampia di software matematico. Per buona parte delle routine esistono versioni in FORTRAN e C, in singola e doppia precisione, nonché una versione parallela (in Fortran 77).
   Il Capitolo 5 della NAg's Fortran 90 Library, è dedicato alla risoluzione numerica di sistemi di equazioni lineari; il modulo `Module` 5.7 : `nag_sparse_lin_sys` contiene procedure che implementano metodi iterativi, con o senza precondizionatore, per la risoluzione di sistemi lineari sparsi con matrice dei coefficienti non simmetrica o complessa non Hermitiana. La routine `nag_sparse_gen_lin_sol` consente la scelta tra quattro metodi iterativi: GMRES(m), CGS, Bi-CGSTAB(l), TFQMR e prevede un'opzione sull'utilizzo del precondizionatore.

   La libreria dell'IMSL (*International Mathematics and Statistics Library*) è una collezione di software matematico, di tipo commerciale. La prima versione è stata pubblicata in Fortran nel 1970, seguita da una versione in linguaggio C, originariamente chiamata C/Base nel 1991, da una versione in Java nel 2002 ed, infine, da una versione in C# nel 2004.
   La versione `IMSL Fortran Numerical Library Version 6.0` comprende una sezione costituita da routine dedicate alla risoluzione di sistemi lineari mediante metodi iterativi; `PCGRC` risolve un sistema lineare a coefficienti reali, con matrice simmetrica e definita, usando il metodo del Gradiente Coniugato (CG) precondizionato (with reverse communication), `JCGRC` risolve un sistema lineare a coefficienti reali, con matrice simmetrica e definita, usando il metodo del Gradiente Coniugato (CG) con precondizionatore di Jacobi (with reverse communication), infine `GMRES` implementa il metodo omonimo (GMRES with reverse communication), per la risoluzione numerica di un sistema lineare.

   La libreria `PETSc` (*Portable Extensible Toolkit for Scientific Computation*), di tipo *open-source*, è una collezione di strutture dati e routine per la soluzione di problemi descritti mediante equazioni differenziali alle derivate parziali (PDE) e risolvibili sia in ambiente di calcolo sequenziale che parallelo.
   La libreria `PETSc` è adatta ad essere utilizzata per problemi su larga scala e molti progetti scientifici sono costruiti intorno ai suoi moduli. PETSc include, inoltre,

un gran numero di solutori per sistemi di equazioni lineari e non lineari che si interfacciano con software scritto in C, C++ e Fortran e che si basano su librerie di software per il calcolo scientifico ormai consolidate, quali BLAS e LAPACK; i solutori di `PETSc` utilizzano, inoltre, la libreria MPI per le comunicazioni in ambiente parallelo.
Tra i moduli principali che costituiscono la libreria citiamo il `KSP`, che contiene le implementazioni di molti dei più popolari metodi iterativi basati sui sottospazi di Krylov, incluso GMRES, CG, CGS, Bi-CG-Stab e LSQR; a tal proposito vale la pena ricordare, inoltre, il modulo `PC`, una collezione di precondizionatori, implementati sia in sequenziale che in parallelo (che comprende, ad esempio, *ILU(k)* (sequenziale), *LU*, *block Jacobi*, sia in sequenziale che in parallelo, e *ICC(0)*).

Routines destinate alla risoluzione di sistemi di equazioni lineari mediante metodi iterativi, in particolare basati sui sottospazi di Krylov, si possono trovare nella libreria di software matematico `TRILINOS`, di dominio pubblico.
`TRILINOS` è composta da una serie di pacchetti di routine sequenziali e parallele, autoconsistenti; tuttavia lo sviluppatore di software che utilizza lo strumento `TRILINOS` può sfruttare la possibile interazione tra i diversi pacchetti nonché quella tra questi ultimi e librerie di software di produzione esterna. I pacchetti di `TRILINOS` si servono anche del supporto delle librerie BLAS e LAPACK ed utilizzano la libreria MPI per le comunicazioni in ambiente parallelo. Essi sono scritti in C++, ma forniscono interfacce di supporto per utenti C e Fortran.

5. Tra gli ambienti di risoluzione di problemi (PSE) quello più utilizzato è sicuramente `MATLAB` il quale mette a disposizione nove funzioni che implementano i principali **metodi iterativi** per sistemi con matrice dei coefficienti sparsa; tra questi:

   ```
   bicg, cgs, gmres, lsqr, minres, symmlq, ...
   ```

4.15.1 La libreria `SPARSKIT`

`SPARSKIT` è una libreria di software per la risoluzione di problemi con matrici sparse. La sua organizzazione è illustrata in Figura 4.6.
Analizziamo, brevemente, alcuni pacchetti che costituiscono la libreria.

FORMATS

Utilizzando le procedure contenute nel file `formats.f` è possibile convertire una matrice sparsa da un formato di memorizzazione ad un altro. Il nome delle routine incluse nel file è costituito da sei lettere, le prime tre per il formato di input, le altre tre per il formato di output; ad esempio il nome della routine `COOCSR` fornisce informazioni sul

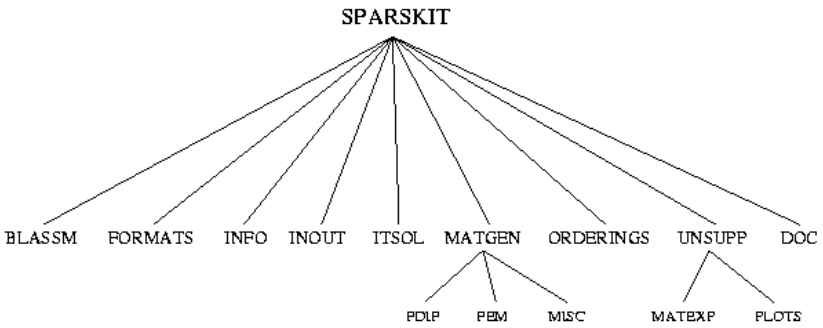

Figura 4.6: Organizzazione di `SPARSKIT`.

formato della matrice prima e dopo la conversione:

$$\underbrace{\texttt{COO}}_{\text{formato di input}} \qquad \underbrace{\texttt{CSR}}_{\text{formato di output}}$$

Analizziamo alcune delle sedici memorizzazioni supportate.

1. `DNS` (*Dense format*): è lo schema di memorizzazione in cui sono memorizzati tutti gli elementi della matrice sparsa.

2. `BND` (*Banded Linpack format*): è la memorizzazione utilizzata da Linpack per le matrici a banda.

♣ **Esempio 4.30.** Si consideri una matrice A di dimensione $m \times n$, con ku diagonali superiori e kl diagonali inferiori. Sia $m = n = 6$ e le ampiezze di banda inferiore e superiore rispettivamente $kl = 1$, $ku = 2$.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & 0 \\ 0 & 0 & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & 0 & a_{6,5} & a_{6,6} \end{pmatrix}.$$

Dato che la matrice ha solo quattro diagonali *"significative"*, si può considerare un array bidimensionale AB di dimensione $ku + kl + 1 \times n = 4 \times n$ nelle cui righe memorizzare le diagonali della matrice nel modo seguente:

$$AB = \begin{pmatrix} * & * & a_{1,3} & a_{2,4} & a_{3,5} & a_{4,6} \\ * & a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} & a_{5,6} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} & a_{6,6} \\ a_{2,1} & a_{3,2} & a_{4,3} & a_{5,4} & a_{6,5} & * \end{pmatrix}$$

dove il simbolo $*$ indica che al corrispondente elemento dell'array AB non è stato associato alcun valore. L'elemento $a_{i,j}$ è dunque memorizzato nella colonna di posto j. L'indice di riga in AB è individuato dalla corrispondenza: $i \rightarrow ku + 1 + i - j$. ♣

In generale, per memorizzare una generica matrice a banda A, di dimensione $m \times n$, con ampiezze di banda ku e kl, si utilizza un array bidimensionale di dimensione $(ku + kl + 1) \times n$, nel modo seguente:

$$AB(ku + 1 + i - j, j) = a_{i,j}$$
$$1 \leq ku + 1 + i - j \leq ku + kl + 1 \;\Rightarrow\; \max(1, j - ku) \leq i \leq \min(m, j + kl)\,.$$

Tale schema di memorizzazione è detto *a banda* (in inglese si parla di *band storage*) ed ha una complessità di spazio pari a $(ku + kl + 1)n$. Tale memorizzazione risulta "significativamente" vantaggiosa, rispetto alla usuale memorizzazione mediante un array bidimensionale di dimensione $m \times n$, se $ku, kl << min\{m, n\}$.

3. `CSR` (*Compressed Sparse Row format*) è la rappresentazione di base utilizzata da `SPARSKIT`.

♣ **Esempio 4.31.** Sia assegnata la matrice sparsa $A \in \Re^{m \times n}$ con nnz elementi non nulli. Sia $m = n = 6$ e $nnz = 12$. Memorizziamo la matrice A nei tre vettori AA, JA, IA. Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati per righe; il vettore di interi JA contiene l'indice di colonna degli elementi a_{ij} di A memorizzati in AA. La dimensione di JA è nnz; il vettore di reali IA contiene il puntatore al primo elemento non nullo di ogni riga di A. Quindi il valore di $IA(i)$ coincide con la posizione in AA (e JA) del primo elemento non nullo della $i-$esima riga. La dimensione di IA è $m + 1$ con $IA(m + 1) = nnz + 1$. Si osserva, inoltre, che $IA(i + 1) - IA(i)$ rappresenta il numero di elementi non nulli della riga i. Utilizzando tale schema la matrice

$$A = \begin{pmatrix} 7 & 0 & -3 & 0 & -1 & 0 \\ 2 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -2 & 0 & 6 \end{pmatrix}$$

è memorizzata nel modo seguente:

$$AA = (7, -3, -1, 2, 8, 1, -3, 5, -1, 4, -2, 6);$$

$$JA = (1, 3, 5, 1, 2, 3, 1, 4, 2, 5, 4, 6);$$

$$IA = (1, 4, 6, 7, 9, 11, 13).$$

Si noti che l'ultimo elemento di IA è uguale al numero totale degli elementi di AA più uno.

♣

4. `CSC` (*Compressed Sparse Column format*) consiste nella memorizzazione degli elementi non nulli per colonne.

♣ **Esempio 4.32.** Sia $A \in \Re^{m\times n}$ la matrice sparsa dell'esempio 4.31, con nnz elementi non nulli, $m = n = 6$ e $nnz = 12$. Memorizziamo la matrice A nei tre vettori AA, JA, IA. Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati per colonne; il vettore di interi JA contiene l'indice di riga degli elementi a_{ij} di A memorizzati in AA. La dimensione di JA è nnz; il vettore di reali IA contiene il puntatore al primo elemento non nullo di ogni colonna di A. Quindi il valore di $IA(j)$ coincide con la posizione in AA (e JA) del primo elemento non nullo della $j-$esima colonna. La dimensione di IA è $n+1$ con $IA(n+1) = nnz+1$. Si osserva, inoltre, che $IA(j+1) - IA(j)$ rappresenta il numero di elementi non nulli della colonna j. Utilizzando tale schema la matrice

$$A = \begin{pmatrix} 7 & 0 & -3 & 0 & -1 & 0 \\ 2 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -2 & 0 & 6 \end{pmatrix}$$

è memorizzata nel modo seguente:

$$AA = (7, 2, -3, 8, -1, -3, 1, 5, -2, -1, 4, 6);$$

$$JA = (1, 2, 4, 2, 5, 1, 3, 4, 6, 1, 5, 6);$$

$$IA = (1, 4, 6, 8, 10, 12, 13).$$

Si noti che l'ultimo elemento di IA è uguale al numero totale degli elementi di AA più uno.

♣

5. `COO` (*Coordinate format*) è la memorizzazione degli elementi non nulli di una matrice sparsa in qualunque ordine.

♣ **Esempio 4.33.** Sia $A \in \Re^{m\times n}$ una matrice sparsa con nnz elementi non nulli, $m = n = 5$ e $nnz = 12$. Memorizziamo la matrice A in tre vettori AA, JA, IA. Il vettore di reali AA contiene gli nnz elementi non nulli di A memorizzati *in qualunque ordine*; il vettore di interi JA contiene l'indice di riga degli elementi a_{ij} di A memorizzati in AA. La dimensione di JA è nnz; il vettore di reali IA contiene l'indice di colonna degli elementi a_{ij} di A memorizzati in AA. La dimensione di IA è nnz. Utilizzando tale schema la matrice

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix}$$

è memorizzata nel modo seguente:

$$AA = (12, 9, 7, 5, 1, 2, 11, 3, 6, 4, 8, 10);$$

$$JA = (5, 3, 3, 2, 1, 1, 4, 2, 3, 2, 3, 4);$$
$$IA = (5, 5, 3, 4, 1, 4, 4, 1, 1, 2, 4, 3).$$

♣

Tra le trentuno routine contenute in `formats.f` ricordiamo:

1. `CSRDNS`: converte il formato dal `CSR` al `DNS`;
2. `DNSCSR`: converte il formato dal `DNS` al `CSR`;
3. `COOCSR`: converte il formato dal `COO` al `CSR`;
4. `COICSR`: converte il formato dal `COO` al `CSR` effettuandolo *in-place*;
5. `CSRCOO`: converte il formato dal `CSR` al `COO`;
6. `CSRCSC`: converte il formato dal `CSR` al `CSC`, effettuando una trasposizione della matrice iniziale.

♣ **Esempio 4.34.** Sia assegnata una matrice di reali, *sparsa*, quadrata di dimensione 5, memorizzata in un array *dns*:

$$dns = \begin{pmatrix} 1 & 0 & 2 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 5 & 6 \\ 0 & 7 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

Convertire A dal formato *denso*, `DNS`, al formato `CSR`. Assegnando in input i parametri: $nrow = 5$, $ncol = 5$, un valore $nzmax \geq 9$, la variabile *dns* e $ndns = 5$, la routine `dnscsr` restituisce, in output, i tre vettori:

$$a = (1, 2, 3, 4, 5, 6, 7, 8, 9);$$
$$ja = (1, 3, 2, 4, 4, 5, 2, 5, 5);$$
$$ia = (1, 3, 5, 7, 9, 10).$$

e la variabile $ierr = 0$. ♣

Alcune delle routine contenute nel file `unary.f`, che eseguono operazioni di base sulle matrici, sono:

1. `SUBMAT`: estrae una sottomatrice quadrata o rettangolare da una matrice sparsa. Sia la matrice in input che quella in output sono nel formato `CSR`. La routine è *in-place*.

2. `COPMAT`: copia una matrice nel formato `CSR` in un'altra anch'essa in formato `CSR`.

3. `GETELM`: è una funzione il cui valore di output è l'elemento a_{ij} per ogni coppia (i, j) assegnata. Come parametro di output fornisce anche l'indirizzo dell'elemento negli array A e JA.

4. `GETDIA`: estrae la diagonale della matrice assegnata. Si può scegliere di non modificare la matrice di input o azzerare tutti i suoi elementi diagonali.

5. `CPERM`: effettua una permutazione delle colonne della matrice assegnata A, ovvero calcola la matrice $B = A \cdot Q$, con Q matrice di permutazione.

6. `RPERM`: effettua una permutazione delle righe della matrice assegnata A, ovvero calcola la matrice $B = P \cdot A$, con P matrice di permutazione.

7. `RETMX`: restituisce l'elemento massimo in valore assoluto per ciascuna riga della matrice assegnata A.

8. `INFDIA`: calcola il numero di elementi non nulli di ciascuna delle $2n - 1$ diagonali della matrice assegnata. Si noti che la prima diagonale considerata è quella denominata $-n$, costituita dal solo elemento di input $a_{n,1}$, mentre l'ultima è quella denominata n, costituita dal solo elemento $a_{1,n}$.

9. `RNRMS`: calcola le norme delle righe della matrice assegnata. Le norme $\|\cdot\|_1$, $\|\cdot\|_2$ e $\|\cdot\|_\infty$ sono supportate.

Analizziamo, ad esempio, la `INFDIA`.

subroutine infdia(n, ja, ia, ind, idiag):

Parametri di input:

- n: dimensione della matrice assegnata;
- ja: array delle colonne degli elementi non nulli della matrice assegnata nel formato `CSR` (secondo vettore della rappresentazione);
- ia: array di puntatori; terzo vettore della rappresentazione della matrice assegnata nel formato `CSR`.

Parametri di output:

- ind: array di interi di lunghezza $2n - 1$. Il Il $k-$esimo elemento del vettore ind contiene il numero di elementi non nulli della diagonale k.
- $idiag$: intero, contiene il numero totale di elementi non nulli trovati sulle diagonali della matrice assegnata.

♣ **Esempio 4.35.** Sia assegnata una matrice di reali, *sparsa*, quadrata di dimensione $n = 5$:

$$A = \begin{pmatrix} 0 & 3 & 2 & 0 & 0 \\ 1 & 0 & 0 & 2 & 3 \\ 0 & 6 & 8 & 0 & 7 \\ 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 8 & 0 \end{pmatrix}$$

Si supponga A memorizzata nel formato `CSR` di `SPARSKIT`; in particolare, il secondo e terzo vettore della rappresentzione sono, rispettivamente:

$$ja = (2, 3, 1, 4, 5, 2, 3, 5, 4, 1, 4)$$

$$ia = (1, 3, 6, 9, 10, 12).$$

Assegnando, in input, alla routine `infdia` la variabile $n = 5$ ed i vettori ja e ia, essa restituisce il vettore

$$ind = (1, 0, 0, 3, 2, 1, 3, 1, 0)$$

e la variabile $idiag = 11$.

♣

BLASSM

I moduli contenuti in `BLASSM` eseguono operazioni algebriche di base. In particolare il modulo `blassms.f` esegue operazioni che coinvolgono due matrici, quali: $A+B$, $A+\sigma\, B$, con $\sigma \in \Re$, $A \cdot B$, etc.

Descriviamo, brevemente, di seguito, le nove routines contenute in `blassms.f`:

1. `AMUB`: calcola il prodotto di due matrici, ovvero $C = A \cdot B$, dove sia A che B sono in formato `CSR`;
2. `APLB`: calcola la somma di due matrici, ovvero $C = A + B$, dove sia A che B sono in formato `CSR`;
3. `APLSB`: calcola $C = A + \sigma B$, dove $\sigma \in \Re$ e sia A che B sono matrici in formato `CSR`;
4. `APMBT`: calcola sia la somma $C = A + B^T$ che la differenza $C = A - B^T$;
5. `APLSBT` calcola l'operazione $C = A + \sigma\, B^T$, $\sigma \in \Re$;
6. `APLDIA`: calcola la somma di una matrice sparsa e di una matrice diagonale;

7. `DIAMUA`: calcola il prodotto di una matrice diagonale (posta a sinistra) ed una matrice sparsa, ovvero $C = D \cdot A$, con D matrice diagonale ed A matrice sparsa, entrambe memorizzate in formato `CSR`;

8. `AMUDIA`: calcola il prodotto di una matrice sparsa ed una matrice diagonale (posta a destra), ovvero $C = A \cdot D$, con D matrice diagonale ed A matrice sparsa, entrambe memorizzate in formato `CSR`;

9. `APLSCA`: *in-place* somma uno scalare alla diagonale di una matrice sparsa, ovvero esegue $A = A + \sigma I$, dove σ è uno scalare, A la matrice sparsa ed I la matrice identica.

Analizziamo in dettaglio la `AMUB`.

subroutine amub(nrow, ncol, job, a, ja, ia, b, jb, ib, c, jc, ic, nzmax, iw, ierr):

Parametri di input:

- *nrow*: intero, numero di righe della matrice A e della matrice C;
- *ncol*: intero, numero di colonne della matrice B e della matrice C;
- *job*: intero. Se $job = 0$ vengono creati solo i vettori jc e ic, ovvero viene creata solo la struttura della matrice C ma non sono calcolati i valori dei suoi elementi;
- a, ja, ia: array per la memorizzazione della matrice A nel formato `CSR`;
- b, jb, ib: array per la memorizzazione della matrice B nel formato `CSR`;
- *nzmax*: intero, rappresenta la lunghezza dei vettori di output, c e jc;
- *iw*: array di interi, area di lavoro di lunghezza uguale al numero di colonne della matrice A.

Parametri di output:

- c, jc, ic: array per la memorizzazione della matrice prodotto C nel formato `CSR`;
- *ierr*: intero, indicatore di errore; $ierr = 0$ indica che l'esecuzione si è conclusa correttamente mentre $ierr = k > 0$ indica che l'esecuzione è terminata durante il calcolo della k-esima riga della matrice C.

♣ **Esempio 4.36.** Siano assegnate due matrici di reali, A e B, *sparse*, quadrate, di dimensione 5 e si suppongano memorizzate nel formato `CSR` di `SPARSKIT`; in particolare, i vettori della rappresentazione di A sono:

$$a = (1, 4, 9, 2, 8, 5, 7)$$
$$ja = (1, 2, 2, 4, 2, 3, 5)$$
$$ia = (1, 3, 5, 6, 7, 8)$$

mentre quelli della rappresentazione di B sono:

$$b = (1, 2, 4, 2, 1, 3, 1)$$
$$jb = (2, 3, 1, 5, 5, 4, 1)$$
$$ib = (1, 3, 5, 6, 7, 8)$$

Assegnando, in input, alla routine `amub` le variabili $nrow = 5$, $ncol = 5$, $job = 1$, $nmax = 15$ e $iw[ncol]$, oltre che i sei vettori, a, ja, ia e b, jb, ib, essa restituisce, in output, la matrice prodotto, anch'essa memorizzata nel formato `CSR`:

$$c = (16, 1, 2, 8, 36, 6, 18, 32, 16, 5, 7)$$
$$ja = (1, 2, 3, 5, 1, 4, 5, 1, 5, 5, 1)$$
$$ia = (1, 5, 8, 10, 11, 12)$$

e la variabile $ierr = 0$. ♣

Il modulo `matvec.f` esegue operazioni di base che coinvolgono una matrice ed un vettore, ad esempio il prodotto matrice per vettore e la risoluzione di sistemi triangolari. Descriviamo, brevemente, alcune delle quindici routines contenute in `matvec.f`:

1. `AMUX`: esegue il prodotto di una matrice per un vettore ($y = Ax$). La matrice sparsa A è memorizzata nel formato `CSR`;

2. `ATMUX`: esegue il prodotto della trasposta di una matrice per un vettore ($y = A^T x$). La matrice sparsa A deve essere memorizzata nel formato `CSR`. Si noti come questa routine può eseguire anche il prodotto di una matrice sparsa A per un vettore, con A memorizzata nel formato `CSC`;

3. `LSOL`: risolve un sistema la cui matrice è triangolare inferiore ed unitaria (l'inversa coincide con la trasposta coniugata). La matrice è memorizzata nel formato `CSR`;

4. `LDSOL`: risolve un sistema la cui matrice è triangolare inferiore. La matrice è memorizzata nel formato `MSR` (*Modified Sparse Row*). Gli elementi della diagonale sono memorizzati in ordine inverso;

5. `LSOLC`: risolve un sistema la cui matrice è triangolare inferiore ed unitaria. La matrice è memorizzata nel formato `CSC`;

6. `USOL`: risolve un sistema la cui matrice è triangolare superiore ed unitaria. La matrice è memorizzata nel formato `CSR`;

7. `USOLC`: risolve un sistema la cui matrice è triangolare superiore ed unitaria. La matrice è memorizzata nel formato `CSC`.

Le altre routine contenute in `matvec.f` eseguono le stesse operazioni delle routine illustrate ma la matrice coinvolta nell'operazione è memorizzata in altri formati.

Analizziamo in dettaglio la `AMUX`.

subroutine amux(n, x, y, a , ja, ia):

Parametri di input:

- n: intero, numero di righe della matrice A;
- x: array di reali di lunghezza pari al numero di colonne della matrice A;
- a, ja, ia: array per la memorizzazione della matrice A nel formato `CSR`.

Parametri di output:

- y: array di reali di lunghezza n; contiene il prodotto $A \cdot x$.

♣ **Esempio 4.37.** Siano A una matrice di reali, quadrata, di dimensione $n = 5$ e x un vettore reale, anch'esso di dimensione $n = 5$. Sia A memorizzata nel formato `CSR` di `SPARSKIT`; in particolare, i vettori della sua rappresentazione siano:

$$a = (1, 4, 9, 2, 8, 5, 7)$$

$$ja = (1, 2, 2, 4, 2, 3, 5)$$

$$ia = (1, 3, 5, 6, 7, 8)$$

e sia

$$x = (1, 2, 3, 4, 5)$$

Assegnando, in input, alla routine `amux` le variabili $n = 5$, x ed i tre vettori, a, ja, ia, essa restituisce, in output, il vettore prodotto:

$$y = (9, 26, 16, 15, 35)$$

♣

INOUT

`INOUT` comprende routine per la lettura, scrittura e visualizzazione grafica delle strutture delle matrici sparse.

Analizziamo in dettaglio la `READMT`.

subroutine readmt(nmax,nzmax,job,iounit,a,ja,ia,rhs,nrhs,guesol,nrow, ncol,nnz,title,key,type,ierr):

Parametri di input:

- *nmax*: intero, massimo numero di colonne per la matrice di input;
- *nzmax*: massimo numero di elementi non nulli per la matrice di input;
- *job*: intero, contiene informazioni sulla lettura dei dati di input; se $job = 0$ legge i valori di *ncol*, *nrow*, *nnz*, *title*, *key*, *type* e *return*. La matrice non è letta; gli array a, ja, ia ed il vettore dei termini noti *rhs* non sono modificati. Se $job = 1$ sono letti solo gli arrays *ja* e *ia*. Se $job = 2$ sono letti gli arrays *a*, *ja* e *ia*. Se $job = 3$ sono letti gli arrays *a*, *ja* e *ia* ed il vettore dei termini noti *rhs*.
- *nrhs*: intero. In input contiene la dimensione di *rhs*;
- *iounit*: variabile logica, fornisce informazioni sulla matrice.

Parametri di output:

- *job*: in output è modificato al valore più alto che può assumere, in base ai dati di input; ad esempo, se, in input, $job = 2$ ma non è fornita una matrice da leggere, il valore della variabile è modificato, in output, in $job = 1$.
- a, ja, ia: vettori della rappresentazione della matrice, nel formato `CSC`;
- *rhs*: array di reali, se determinato, in base al valore di *job*, è di dimensione $nrow + 1$;
- *nrhs*: intero, contiene il numero dei vettori dei termini noti;
- *guesol*: dato di tipo alfanumerico, costituito da due caratteri; questi ultimi segnalano, rispettivamente, se, in coda alle componenti del vettore dei termini noti, sono assegnati, in input, un dato iniziale (*initial guess*) e/o la *soluzione "esatta"*.
- *nrow*: intero, numero di righe della matrice;
- *ncol*: intero, numero di colonne della matrice;
- *nnz*: intero, numero di elementi non nulli di A;

- *title*: dato di tipo alfanumerico, costituito da 72 caratteri; contiene il titolo (descrittivo) della matrice test;
- *key*: dato di tipo alfanumerico, costituito da 8 caratteri; identificativo per la matrice;
- *type*: dato di tipo alfanumerico, costituito da 3 caratteri; descrive il tipo della matrice.
- *ierr*: intero, indicatore di errore; $ierr = 0$ segnala che la matrice è stata letta, $ierr = 1$ che la matrice non può essere letta, perché $ncol + 1 > nmax$, $ierr = 3$ segnala che la matrice non può essere letta, perché $ncol + 1 > nmax$ e $nnz > nzmax$; se $ierr = 4$ vuol dire che il vettore dei termini noti, l'*initial guess* e la soluzione "esatta" non possono essere letti poiché memorizzati in formato *sparso*; infine, se $ierr = 5$ allora i tre vettori (il vettore dei termini noti, l'*initial guess* e la soluzione "esatta") non possono essere letti poiché la dimensione dell'array *rhs*, dichiarata in input nella variabile *nrhs*, non è sufficientemente grande per memorizzarli.

♣ **Esempio 4.38.** Sia assegnato, in input, il puntatore al file in cui è rappresentata la matrice in formato `H/B`:

```
title                                                                   key
        5    1    1    3    0
RUA          5    5   12    0
(6I3)     (12I3)     (5E15.8)     (5E15.8)
  1    4    6    8   11   13
  1    3    5    1    2    2    3    1    4    5    3    5
2.02765219E-01  6.03792479E-01  1.98814268E-01  1.52739270E-02  7.46785677E-01
8.46221418E-01  5.25152496E-01  8.38118445E-01  3.79481018E-01  8.31796018E-01
4.28892365E-01  1.89653748E-01
```

Il file descrive la matrice che, rappresentata in formato denso, è:

$$A = \begin{pmatrix} 2.02765219E-01 & 0 & 6.03792479E-01 & 0 & 1.98814268E-01 \\ 1.52739270E-02 & 7.46785677E-01 & 0 & 0 & 0 \\ 0 & 8.46221418E-01 & 5.25152496E-01 & 0 & 0 \\ 8.38118445E-01 & 0 & 0 & 3.79481018E-01 & 8.31796018E-01 \\ 0 & 0 & 4.28892365E-01 & 0 & 1.89653748E-01 \end{pmatrix}$$

In output, la funzione `readmt` fornisce, tra le altre variabili, i tre vettori:

$$\begin{aligned} a &= (2.02765219E-01, \quad 6.03792479E-01, \quad 1.98814268E-01, \quad 1.52739270E-02, \\ &\quad 7.46785677E-01, \quad 8.46221418E-01, \quad 5.25152496E-01, \quad 8.38118445E-01, \\ &\quad 3.79481018E-01, \quad 8.31796018E-01, \quad 4.28892365E-01 \quad 1.89653748E-01) \\ ja &= (1, 3, 5, 1, 2, 2, 3, 1, 4, 5, 3, 5) \\ ia &= (1, 4, 6, 8, 11, 13) \end{aligned}$$

♣

ITSOL

`ITSOL` comprende due driver principali: `ILUT` e `ITERS`:

- `ILUT` implementa l'algoritmo del GMRES (metodo iterativo per la risoluzione di un sistema lineare) precondizionato.
- `ITERS` implementa nove dei più utilizzati metodi iterativi per la risoluzione di sistemi lineari.

4.16 Risoluzione di sistemi lineari *sparsi* in ambiente `MATLAB`

In ambiente `MATLAB` le funzioni relative a *matrici sparse* sono collocate nella directory `sparfun` e sono organizzate in categorie. Se ne descrivono alcune di seguito.

- Funzioni che consentono la **conversione da un formato di memorizzazione ad un altro** che sia più "vantaggioso" in termini di complessità di spazio. Inoltre, la memorizzazione dei soli elementi non nulli e dei loro indici consente, attraverso l'utilizzo di opportune funzioni, di evitare operazioni aritmetiche che sarebbero eseguite su elementi nulli, riducendone, così, il costo computazionale richiesto.

♣ **Esempio 4.39.** Sia `S` una matrice sparsa, memorizzata in formato *denso* [24]; si individuino gli indici di riga e colonna degli elementi non nulli di `S`; si memorizzino questi ultimi in un array `s`:

```
S=[ 1.6000    2.7000         0         0         0         0
    0.3000         0         0         0   47.0000         0
    1.1300   71.0000         0         0         0    1.0000
    0.0100         0         0    0.2200    0.0300   84.0000
         0         0         0         0   14.0000   15.0000
         0         0    0.7000    0.0800         0         0 ]

>>  [i,j,s] = find(S);
```

I vettori di output `i` e `j` contengono, rispettivamente, gli indici di riga e di colonna degli elementi non nulli di `S`.
Costruire una matrice `T`, nel formato sparso di `MATLAB` , che conservi solo le informazioni relative agli elementi non nulli di `S`:

```
>>  [m,n] = size(S);
>>  T= sparse(i,j,s,m,n);
```

[24] Per *formato denso* si intende la memorizzazione di una matrice in un array bidimensionale contenente tutti gli elementi della matrice, compresi i nulli.

L'output visualizzato sarà:

```
T =

   (1,1)       1.6000
   (2,1)       0.3000
   (3,1)       1.1300
   (4,1)       0.0100
   (1,2)       2.7000
   (3,2)      71.0000
   (6,3)       0.7000
   (4,4)       0.2200
   (6,4)       0.0800
   (2,5)      47.0000
   (4,5)       0.0300
   (5,5)      14.0000
   (3,6)       1.0000
   (4,6)      84.0000
   (5,6)      15.0000
```

Lo stesso risultato si ottiene attraverso l'istruzione:

```
>> T=sparse(S)
```

L'output consiste negli elementi non nulli di S con i rispettivi indici di riga e di colonna. Gli elementi sono ordinati per colonna. Con

```
>> full(T)
```

si visualizza T nel formato denso:

```
ans =

    1.6000    2.7000         0         0         0         0
    0.3000         0         0         0   47.0000         0
    1.1300   71.0000         0         0         0    1.0000
    0.0100         0         0    0.2200    0.0300   84.0000
         0         0         0         0   14.0000   15.0000
         0         0    0.7000    0.0800         0         0
```

♣

- Funzioni che **generano** matrici sparse elementari, ad esempio, `speye`, `sprand`, `sprandn`, o che operano su matrici sparse, come `sprandsym`, `spdiags`.

♣ **Esempio 4.40.** Assegnata una matrice *a banda* A, di dimensione $m \times n = 7 \times 4$:

```
>>    A=[ 11     0   13    0
           0    22    0   24
           0     0   33    0
          41     0    0   44
           0    52    0    0
           0     0   63    0
           0     0    0   74 ]
```

e ampiezza di banda $w = 6$, estrarre tutte le diagonali non nulle. In particolare: costruire una matrice B di dimensione $min(m, n) \times \overline{w}$, con $\overline{w} \leq w$, le cui colonne siano le $\overline{w}$ diagonali non nulle di A; costruire, inoltre, un vettore d di lunghezza w, le cui componenti intere individuino la posizione, in A, delle sue diagonali non nulle:

```
>> [B,d]=spdiags(A)

B =

    41    11     0
    52    22     0
    63    33    13
    74    44    24


d =

    -3
     0
     2
```

Si osserva che, se la $i-$esima componente di d è $d(i) = 0$, allora l'$i-$esima colonna di B è la diagonale principale di A; se $d(i) = k > 0$, l'$i-$esima colonna di B è la $k-$esima diagonale sopra la principale, se, infine, $d(i) = k < 0$, l'$i-$esima colonna di B è la $k-$esima diagonale sotto la principale.

Viceversa, dati gli array $B \in \Re^{p \times \overline{w}}$ e $d \in \Re^{\overline{w} \times 1}$, con $p = 4$ e $\overline{w} = 3$, costruire la matrice C che abbia $m = 7$ righe, $n = p$ colonne e le $\overline{w}$ colonne di B al posto delle diagonali indicate dalle componenti del vettore d:

```
>> C = spdiags(B,d,m,n)

C =

   (1,1)       11
   (4,1)       41
   (2,2)       22
   (5,2)       52
   (1,3)       13
   (3,3)       33
   (6,3)       63
   (2,4)       24
   (4,4)       44
   (7,4)       74
```

La funzione `spdiags` può, dunque, sia estrarre elementi diagonali da una matrice sparsa, riducendo la complessità di spazio richiesta dalla memorizzazione dell'intera matrice, che, eventualmente, sostituire i suoi elementi diagonali con valori nuovi. ♣

- Funzioni che forniscono **informazioni quantitative** o **grafiche** sulla struttura delle matrici sparse. Ad esempio:
  - informazioni sugli elementi non nulli o sul loro numero sono rese dalle funzioni `nonzeros` e `nnz`, rispettivamente;
  - la visualizzazione grafica della distribuzione degli elementi non nulli di una matrice sparsa è resa possibile dalla funzione `spy` che genera una figura in cui è visualizzata la *struttura* della matrice assegnata in input. Ogni punto del grafico rappresenta la posizione di un elemento non nullo all'interno della matrice.

♣ **Esempio 4.41.** Sia S la matrice sparsa dell'esempio 4.39. Con

```
>> spy(S,'r*')
```

si ottiene il grafico in Fig.4.7. Agli elementi di A, a_{ij}, $i = 1, \ldots, 6$, $j = 1, \ldots, 6$, corrispondono i punti del grafico, di coordinate $x \in \{1, \ldots, 6\}$, $y \in \{1, \ldots, 6\}$.
Nella finestra grafica compare, *di default*, anche il numero, `nz`, di elementi non nulli della matrice.

♣

♣ **Esempio 4.42.** Esempi di matrici che possono risultare più o meno sparse sono le *matrici di adiacenze* di *grafi*.
Un grafo è un insieme di nodi opportunamente connessi tra loro. La sua matrice delle adiacenze è una matrice quadrata, i cui elementi sono uguali a 0 o 1 (binaria). In particolare, l'elemento (i, j) della matrice è 1 se e solo se esiste nel grafo un arco che va dal vertice i al vertice j, altrimenti è 0.
In molti casi ogni nodo è connesso solo a pochi altri nodi, per cui la matrice delle adiacenze risulta sparsa con, in particolare, pochi elementi non nulli per ciascuna riga.
Ad esempio, in `MATLAB` , il grafo della semisfera (con cui si rappresenta la cupola geodesica di Buckminster Fuller) può essere generato attraverso la funzione `bucky` che fornisce la matrice delle adiacenze, B, e la matrice, V, delle coordinate cartesiane xyz dei vertici. Combinando due semisfere si ottiene il grafo di una struttura detta anche *Buckyball*, la cui forma ricorda quella di un pallone da calcio o una molecola di carbonio-60.
La funzione `gplot` applicata alla matrice delle adiacenze fornisce la visualizzazione bidimensionale del grafo (Fig.4.8).

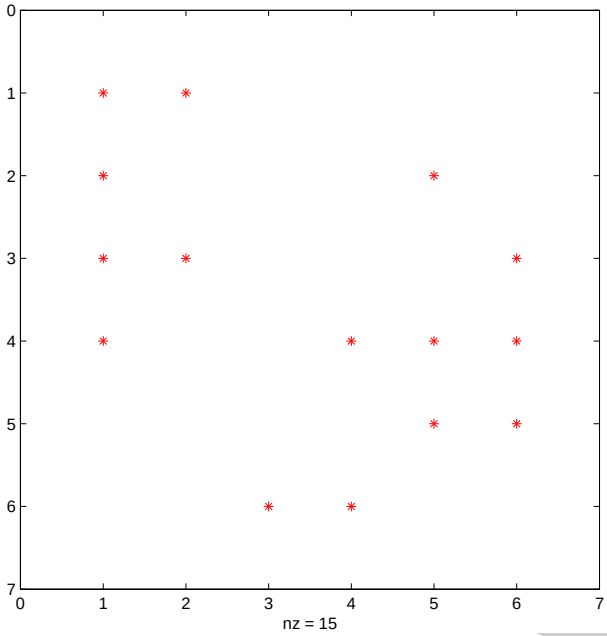

Figura 4.7: Grafico della struttura della matrice sparsa dell'esempio 4.39.

```
>> [B,V] = bucky;
>> % matrice sparsa con elementi nulli
>> H = sparse(60,60);
>> k = 31:60;
>> H(k,k) = B(k,k);

>> gplot(B-H,V,'b-');
>> hold on
>> gplot(H,V,'r-');
>> title('Backyball')
>> axis equal
```

La distribuzione degli elementi non nulli della matrice delle adiacenze della semisfera si può visualizzare attraverso la funzione `spy` che fornisce il grafico della struttura della matrice di input. Si osserva che la matrice B è sparsa, di dimensione 60×60, e simmetrica in quanto il nodo i è connesso al nodo j se, e solo se, il nodo j è connesso al nodo i. Il suo diagramma è riportato in Fig.4.9.

```
>> spy(B)
```

♣

- Funzioni applicabili a matrici sparse che lavorano equivalentemente anche su matrici piene; di esse esiste, inoltre, una versione non specifica per matrici sparse. Tra queste:

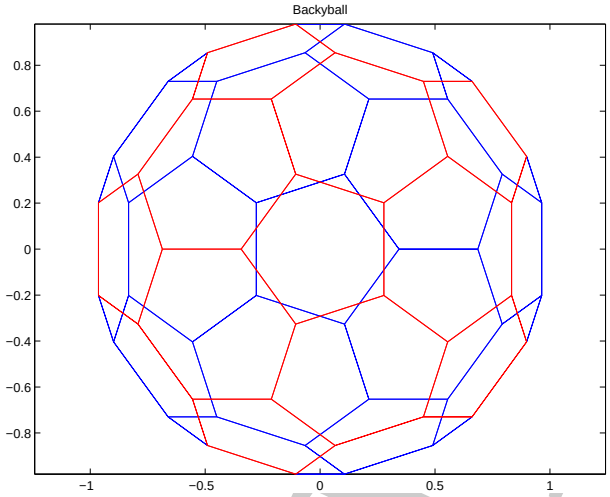

Figura 4.8: Visualizzazione bidimensionale del grafo della *Buckyball* dell'esempio 4.42.

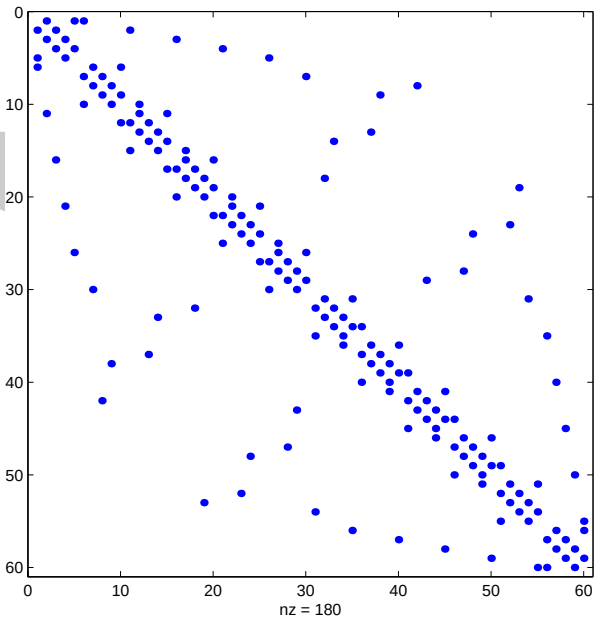

Figura 4.9: Diagramma della matrice delle adiacenze della semisfera dell'esempio 4.42.

```
eigs, svds,
normest,condest,sprank
```

Ad esempio, la function `eigs` consente di calcolare un numero arbitrario di autovalori di una matrice. Opzionalmente consente di calcolare gli autovettori corrispondenti. La matrice di input deve essere quadrata e, preferibilmente, *di grandi dimensioni e sparsa*, sebbene la funzione si possa applicare, equivalentemente, anche a matrici piene.

- Funzioni che implementano le usuali tecniche di fattorizzazione, ad esempio LU, Cholesky e QR, che, se applicate a matrici sparse, tengono conto della sparsità della matrice nell'esecuzione delle operazioni richieste dal metodo implementato:

  ```
  lu, chol, qr
  ```

- Infine, `MATLAB` mette a disposizione nove funzioni che implementano i principali **metodi iterativi** per sistemi con matrice dei coefficienti sparsa; tra questi:

  ```
  bicg, cgs, gmres, lsqr, minres, symmlq, ...
  ```

  Approfondimenti relativi a questi ultimi sono reperibili in letteratura (ad esempio in [2] e [19]).

4.17 Esercizi

4.17.1 Quesiti

Quesito 1 Sotto quali condizioni il metodo iterativo di Gauss-Seidel, per la risoluzione di un sistema di equazioni lineari $Ax = b$, converge?

Quesito 2 Il metodo iterativo di Gauss-Seidel è un caso particolare di metodo di tipo **SOR** per la risoluzione di un sistema di equazioni lineari? Perché?

Quesito 3 Quale fattore principale rende poco efficiente l'utilizzo dei metodi diretti basati su fattorizzazione matriciale, nella risoluzione di sistemi di equazioni lineari sparsi e di grandi dimensioni?

Quesito 4 Qual è la formulazione generale di un metodo iterativo *stazionario* per la risoluzione di un sistema di equazioni lineari del tipo $Ax = b$?

Quesito 5

(a) Cosa si intende per *splitting* di una matrice?

(b) Qual è la formulazione del metodo iterativo risultante dallo splitting, per la risoluzione di un sistema di equazioni lineari del tipo $Ax = b$?

(c) Quale condizione imposta sullo splitting garantisce che lo schema iterativo risultante sia convergente?

(d) Assegnata la matrice

$$A = \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix},$$

qual è la matrice di splitting per il metodo di Jacobi?

(e) Per la stessa matrice in (d), qual è la matrice di splitting per il metodo di Gauss-Seidel?

Quesito 6 Quale dei seguenti metodi iterativi, per la risoluzione di un sistema di equazioni lineari, è *stazionario*?

(a) Jacobi

(b) Gauss-Seidel

(c) SOR

Quesito 7 Descrivere la differenza tra i metodi iterativi di Jacobi e Gauss-Seidel.

(a) Quale dei due metodi converge più rapidamente?

(b) Quale dei due metodi richiede una minore complessità di spazio per la memorizzazione di approssimazioni successive?

Quesito 8 Quali sono i limiti del parametro di rilassamento ω nel metodo SOR?

Quesito 9 Cosa si intende per *precondizionamento*?

4.17.2 Esercizi numerici

Esercizio 1 Fornire vantaggi e svantaggi dei metodi iterativi rispetto ai metodi diretti, nella risoluzione di sistemi di equazioni lineari *sparsi* e di *grandi dimensioni*.

Esercizio 2 Di seguito sono elencate alcune proprietà che caratterizzano i metodi per la risoluzione di sistemi di equazioni lineari. Per ciascuna delle proprietà elencate specificare se essa descrive in maniera più idonea un metodo diretto o un metodo iterativo:

(a) Gli elementi della matrice non sono modificati nel calcolo della soluzione.

(b) Una buona stima a priori per la soluzione è utile.

(c) Gli elementi della matrice sono memorizzati esplicitamente utilizzando uno schema di memorizzazione standard come, ad esempio, un array.

(d) Il costo computazionale richiesto dipende dall'indice di condizionamento del problema.

(e) L'avere risolto un particolare sistema consente di risolvere *facilmente* un altro sistema con la stessa matrice dei coefficienti ma diverso vettore dei termini noti.

(f) Generalmente si utilizzano tecniche di accelerazione o precondizionatori.

(g) Generalmente si ottiene una fattorizzazione della matrice.

(h) La complessità di tempo può essere determinata *a priori*.

Esercizio 3 Sia A una matrice non singolare. Sia L la parte di A *strettamente triangolare inferiore* D la matrice diagonale con elementi uguali a quelli diagonali della matrice A e U la parte di A *strettamente triangolare superiore*.

(a) Esprimere lo schema iterativo di Jacobi in termini di L, D, e U.

(b) Esprimere lo schema iterativo di Gauss-Seidel in termini di L, D, e U.

Esercizio 4 Ordinare i seguenti metodi iterativi per la risoluzione di sistemi di equazioni lineari secondo la loro velocità di convergenza, dal più veloce al più lento:

(a) Gauss-Seidel

(b) Jacobi

(c) SOR con parametro di rilassamento ω ottimale.

Esercizio 5 Dimostrare che il metodo di Jacobi per la risoluzione di un sistema di equazioni lineari $Ax = b$ converge se la matrice dei coefficienti, A, è a diagonale dominante per righe.
[*Suggerimento*: Usare la norma-∞.]

Esercizio 6 Provare che il metodo SOR diverge se ω non appartiene all'intervallo $(0, 2)$.

Esercizio 7 Sia $A \in \Re^{n\times n}$ una matrice a diagonale strettamente dominante per righe. Provare che il metodo iterativo di Gauss-Seidel per la risoluzione di un sistema lineare del tipo $Ax = b$ è convergente.

Esercizio 8 Si consideri la matrice *tridiagonale*, del tipo

$$T_\alpha = tridiag(-1, \alpha, -1) = \begin{pmatrix} \alpha & -1 & & & & \\ -1 & \alpha & -1 & & & \\ & -1 & \alpha & -1 & & \\ & & -1 & \alpha & -1 & \\ & & & -1 & \alpha & -1 \\ & & & & -1 & \alpha \end{pmatrix}$$

con $\alpha \in \Re$. Gli autovalori di T_α sono esprimibili come:

$$\lambda_j = \alpha - 2\,cos(j\,\theta), \qquad j = 1, \ldots, n$$

dove $\theta = \pi/(n+1)$ e gli autovettori corrispondenti sono:

$$\mathbf{q}_j = [sin(j\,\theta), sin(2\,j\,\theta), \ldots, sin(n\,j\,\theta)]^T$$

Per quali valori di α la matrice è definita positiva?
[*Soluzione*: $\alpha \geq 2$].

Esercizio 9 Si consideri la matrice dell'esercizio precedente. Sia $\alpha = 2$.

- Il metodo iterativo di Jacobi converge se applicato alla risoluzione di un sistema di equazioni con T_2 come matrice dei coefficienti? In caso affermativo, quale è il suo *fattore di convergenza* [25]?
- Il metodo iterativo di Gauss-Seidel converge se applicato alla risoluzione di un sistema di equazioni con T_2 come matrice dei coefficienti? In caso affermativo, quale sará il suo *fattore di convergenza*?
- Per quali valori di ω il metodo SOR convergerà?

Esercizio 10 Si consideri la matrice pentadiagonale di dimensione n:

$$A = pentadiag_n(-1, -1, 10, -1, -1).$$

Assumiamo che sia: $n = 10$ e $A = M + N + D$, con $D = diag(8, \ldots, 8) \in \Re^{10\times 10}$, $M = pentadiag_{10}(-1, -1, 1, 0, 0)$ e $N = M^T$. Si analizzi la convergenza dei metodi iterativi seguenti, nella risoluzione del sistema $Ax = b$.

(a) $(M + D)x^{(k+1)} = -Nx^{(k)} + \mathbf{b}$,

(b) $Dx^{(k+1)} = -(M + N)x^{(k)} + \mathbf{b}$,

(c) $(M + N)x^{(k+1)} = -Dx^{(k)} + \mathbf{b}$.

[*Soluzione*: denotando, rispettivamente con $\rho(a)$, $\rho(b)$ e $\rho(c)$ il raggio spettrale delle matrici di iterazione dei tre metodi, si ha $\rho(a) = 0.1450$, $\rho(b) = 0.5$ e $\rho(c) = 12.2870$, il che implica la convergenza per i metodi (a) e (b) e la divergenza del metodo (c).]

Esercizio 11 Per risolvere il sistema lineare $Ax = b$ con:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 3 \\ 5 \end{bmatrix},$$

[25] Il *fattore di convergenza* di una sequenza è il limite

$$\rho = \lim_{k\to\infty} \left(\frac{\|d_k\|}{\|d_0\|} \right)^{1/k}$$

Si osserva che ρ coincide con il raggio spettrale della matrice di iterazione di un metodo iterativo stazionario ad un passo.

consideriamo il metodo iterativo seguente

$$x^{(k+1)} = B(\theta)x^{(k)} + g(\theta), \quad k \geq 0 \quad \text{e} \quad x^{(0)} \quad \text{assegnata}$$

dove θ è un parametro reale e

$$B(\theta) = \frac{1}{4}\begin{bmatrix} 2\theta^2+2\theta+1 & -2\theta^2+2\theta+1 \\ -2\theta^2+2\theta+1 & 2\theta^2+2\theta+1 \end{bmatrix}, \qquad g(\theta) = \begin{bmatrix} \frac{1}{2}-\theta \\ \frac{1}{2}-\theta \end{bmatrix}.$$

Provare che il metodo è consistente $\forall\, \theta \in \Re$. Determinare i valori di θ per cui il metodo è convergente e calcolare il valore *ottimale* di θ (i.e. il valore del parametro per cui la *velocità asintotica di convergenza* risulta massima).
[*Soluzione*: il metodo è convergente se, e solo se, $-1 < \theta < 1/2$ e la *velocità asintotica di convergenza* è massima se $\theta = (1-\sqrt{3})/2$.]

Esercizio 12 Per risolvere il seguente sistema lineare a blocchi

$$\begin{bmatrix} A_1 & B \\ B & A_2 \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

consideriamo i due metodi

(1) $A_1x^{(k+1)} + By^{(k)} = b_1$, $Bx^{(k)} + A_2y^{(k+1)} = b_2$;
(2) $A_1x^{(k+1)} + By^{(k)} = b_1$, $Bx^{(k+1)} + A_2y^{(k+1)} = b_2$.

Trovare condizioni sufficienti che garantiscano la convergenza dei due schemi, per qualsiasi scelta dei dati iniziali: $x^{(0)}, y^{(0)}$.
[*Soluzione*: il metodo (1) è un sistema (disaccoppiato) nelle incognite $x^{(k+1)}$ e $y^{(k+1)}$. Assumendo che A_1 e A_2 siano invertibili, il metodo (1) converge se $\rho(A_1^{-1}B) < 1$ e $\rho(A_2^{-1}B) < 1$.
A proposito del metodo (2) abbiamo un sistema (accoppiato) da risolvere ad ogni passo, nelle incognite $x^{(k+1)}$ e $y^{(k+1)}$. Risolvendo, formalmente, la prima equazione rispetto a $x^{(k+1)}$ (il che richiede che A_1 sia invertibile) e sostituendo nella seconda, si riscontra che il metodo (2) è convergente se $\rho(A_2^{-1}BA_1^{-1}B) < 1$ (A_2 deve essere invertibile).]

Esercizio 13 Consideriamo il sistema lineare $Ax = b$ con

$$A = \begin{bmatrix} 62 & 24 & 1 & 8 & 15 \\ 23 & 50 & 7 & 14 & 16 \\ 4 & 6 & 58 & 20 & 22 \\ 10 & 12 & 19 & 66 & 3 \\ 11 & 18 & 25 & 2 & 54 \end{bmatrix}, \qquad b = \begin{bmatrix} 110 \\ 110 \\ 110 \\ 110 \\ 110 \end{bmatrix}.$$

Verificare se i metodi di Jacobi e Gauss-Seidel possono essere applicati per risolvere il sistema.
[*Soluzione*: La matrice A non è a diagonale dominante nè simmetrica definita

positiva, per cui bisogna calcolare il raggio spettrale delle matrici di iterazione dei metodi di Jacobi e Gauss-Seidel per verificare se sono convergenti. Si riscontra che: $\rho_J = 0.9280$ e che $\rho_{GS} = 0.3066$, il che implica la convergenza per entrambi i metodi.]

Esercizio 14 Consideriamo il sistema lineare $Ax = b$ con

$$A = \begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}, \qquad b = \begin{bmatrix} 23 \\ 32 \\ 33 \\ 31 \end{bmatrix}.$$

Analizzare le proprietà di convergenza dei metodi di Jacobi e Gauss-Seidel applicati al sistema nelle loro formulazioni *puntuali* e *a blocchi* (partizionando la matrice A in blocchi 2×2).
[*Soluzione*: entrambi i metodi sono convergenti, essendo, in particolare, la forma a blocchi la più *veloce*. Inoltre: $\rho^2(B_J) = \rho(B_{GS})$.]

Esercizio 15 Per risolvere il sistema lineare $Ax = b$, consideriamo il metodo iterativo

$$Mx^{(k+1)} = Rx^{(k)} + \mathbf{b} \qquad k \geq 0$$

dove le matrici M e R si ottengono mediante uno *splitting* della matrice A: $A = M - R$; sia $M = D + \omega F$ e $R = -\beta F - E$, con ω e β numeri reali. Provare che il metodo è consistente solo se $\beta = 1 - \omega$. In tal caso esprimere gli autovalori della matrice di iterazione in funzione di ω e determinare per quali valori di ω il metodo è convergente. Calcolare, inoltre, il valore ottimale ω_{opt}, assumendo che sia $A = tridiag_{10}(-1, 2, -1)$.
[*Suggerimento*: Sfruttare il risultato dell'**Esercizio 8**.]

Esercizio 16 Sia $A \in \Re^{n\times n}$ tale che $A = (1+\omega)P-(N+\omega\, P)$, con $P^{-1}N$ non singolare e con autovalori reali $1 > \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Determinare i valori del parametro $\omega \in \Re$ in corrispondenza dei quali il metodo iterativo seguente:

$$(1 + \omega)Px^{(k+1)} = (N + \omega\, P)x^{(k)} + \mathbf{b}, \qquad k \geq 0,$$

converge, $\forall\, x^{(0)}$, alla soluzione del sistema lineare $Ax = b$. Determinare, inoltre, il valore di ω per cui la velocità asintotica di convergenza risulta massima.
[*Soluzione*: $\omega > -(1 + \lambda_n)/2$; $\omega_{opt} = -(\lambda_1 + \lambda_n)/2$.]

Esercizio 17 Si consideri il sistema lineare:

$$\begin{cases} 5x + 3y = 6 \\ 4x - 2y = 8 \end{cases}$$

È possibile utilizzare sia il metodo di Jacobi che quello di Gauss-Seidel per risolvere tale sistema? Perché?

Esercizio 18 L'iterazione di Jacobi può sempre essere usata per calcolare la soluzione del sistema seguente?

$$\begin{cases} 2x + y - 5z = & 9 \\ x - 5y - z = & 14 \\ 7x - y - 3z = & 26 \end{cases}$$

Esercizio 19 Si consideri il seguente sistema lineare *tridiagonale* e si assuma che la matrice dei coefficienti sia *a diagonale dominante*:

$$\begin{cases} d_1x_1+ & c_1x_2 & & & & & & = b_1 \\ a_1x_1+ & d_2x_2+ & c_2x_3 & & & & & = b_2 \\ & a_2x_2+ & d_3x_3+ & c_3x_4 & & & & = b_3 \\ & \ddots & \ddots & \ddots & & & & \vdots \\ & & & a_{N-2}x_{N-2}+ & d_{N-1}x_{N-1}+ & c_{N-1}x_N & & = b_{N-1} \\ & & & & a_{N-1}x_{N-1}+ & d_Nx_N & & = b_N \end{cases}$$

Scrivere un algoritmo che descriva un metodo iterativo per la risoluzione di tale sistema.

Esercizio 20 Dimostrare che, se A è *a diagonale dominante* e se Q è la matrice diagonale i cui elementi diagonali sono gli elementi di A, allora

$$\rho(I - Q^{-1}A) < 1$$

[*Suggerimento*: Basta applicare il **Teorema di Gershgorin** (cfr. **Teorema B.6** dell'**Appendice B**, **Parte prima**) alla matrice $I - Q^{-1}A$ e ricordare la definizione di raggio spettrale.]

Esercizio 21 Dimostrare che il processo iterativo di base, del tipo:

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b \qquad (k \geq 1)$$

è equivalente al seguente:

Assegnata $x^{(k)}$,

calcolare $r^{(k)} = b - Ax^{(k)}$,

risolvere rispetto a $z^{(k)}$ l'equazione $Qz^{(k)} = r^{(k)}$,

definire $x^{(k+1)} = x^{(k)} + z^{(k)}$.

Esercizio 22 Usando la notazione dell'esercizio precedente, dimostrare che

$r^{(k+1)} = (I - AQ^{-1})r^{(k)}$

$z^{(k+1)} = (I - Q^{-1}A)z^{(k)}$

Esercizio 23 Dimostrare che il metodo di Gauss-Seidel è un caso particolare di metodo SOR.

Esercizio 24 Siano $D = diag(A)$, $-C_L$ la parte strettamente triangolare inferiore di A (ovvero la matrice triangolare inferiore costituita dagli elementi di A sotto la diagonale principale) e $-C_U$ la parte strettamente triangolare superiore di A. Dimostrare che le matrici

$$\mathcal{J} = I - D^{-1}A$$
$$\mathcal{G} = I - (D - C_L)^{-1}A$$
$$\mathcal{S}_\omega = I - \omega(2-\omega)(D - \omega C_U)^{-1}D(D - \omega C_L)^{-1}A$$

sono le matrici di iterazione per i metodi di Jacobi, Gauss-Seidel e SSOR, rispettivamente.

Esercizio 25 Trovare la forma esplicita della matrice di iterazione $I - Q^{-1}A$ del metodo di Gauss-Seidel quando

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

Esercizio 26 Caratterizzare la famiglia di tutte le matrici $n \times n$ non singolari, A, per le quali l'algoritmo per il metodo di Gauss-Seidel risolve un sistema $Ax = b$ con *una sola iterazione*, a partire da $x^{(0)} = 0$.

Esercizio 27 Fornire un esempio di matrice, A, che non sia a diagonale dominante, in corrispondenza della quale, tuttavia, il metodo di Gauss-Seidel applicato ad un sistema $Ax = b$ converge.

Esercizio 28 Come si semplifica la tecnica di accelerazione di Chebyshev se il metodo iterativo di base è il metodo di Jacobi?

Esercizio 29 Dimostrare che, se il numero $\delta = \|I - Q^{-1}A\|$ è minore di 1, allora

$$\|x^{(k)} - x\| \leq \frac{\delta}{1-\delta}\|x^{(k)} - x^{(k-1)}\|$$

Esercizio 30 Sia $A \in \Re^{8\times 8}$:

$$A = \begin{bmatrix} 8 & -2 & 0 & 0 & \cdots \\ -2 & 8 & -2 & 0 & \cdots \\ 0 & -2 & 8 & -2 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & -2 & 8 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

1. Verificare che il metodo di Jacobi è convergente.
2. Verificare che il metodo di Gauss-Seidel è convergente.
3. Detta C_J la matrice di iterazione del metodo di Jacobi e C_{GS} quella del metodo di Gauss-Seidel, verificare che tra i raggi spettrali di tali matrici sussiste la relazione:
$$\rho(C_{GS}) = \rho^2(C_J).$$

Esercizio 31 Sia $A \in \Re^{8\times 8}$:

$$A = \begin{bmatrix} 7 & 1 & 0 & 0 & \cdots \\ 1 & 7 & 1 & 0 & \cdots \\ 0 & 1 & 7 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 & 7 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo SOR:

1. verificare che il metodo è convergente;
2. trovare il valore ottimale del parametro di rilassamento ω_{opt} del metodo SOR;
3. calcolare la matrice di iterazione $C_{\omega_{opt}}$ del metodo SOR.

Esercizio 32 Sia $A \in \Re^{3\times 3}$:

$$A = \begin{bmatrix} 8 & -1 & 0 \\ -2 & 4 & -1 \\ 3 & -1 & 7 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo di Jacobi:

1. verificare che il metodo è convergente;
2. costruire la matrice di iterazione C_J;
3. determinare il numero k minimo di iterazioni affinché l'errore sulla soluzione sia minore di 10^{-5}.

Esercizio 33 Sia $A \in \Re^{3\times 3}$:

$$A = \begin{bmatrix} 10 & -3 & 1 \\ -3 & 10 & 4 \\ 1 & 4 & 10 \end{bmatrix}$$

la matrice dei coefficienti di un sistema lineare:

$$Ax = b.$$

Supponendo di utilizzare il metodo iterativo di Gauss-Seidel:

1. verificare che il metodo è convergente;
2. costruire la matrice di iterazione C_{GS};
3. determinare il numero k minimo di iterazioni affinché l'errore sulla soluzione sia minore di 10^{-5}.

Esercizio 34 Si consideri un sistema lineare:

$$Ax = b, \qquad b \in \Re^3$$

dove la matrice dei coefficienti è

$$A = \begin{bmatrix} -1 & 0 & a \\ f & -1 & 0 \\ 0 & c & -1 \end{bmatrix},$$

con a, f, e c numeri reali. Studiare la convergenza del metodo iterativo di Gauss-Seidel al variare dei parametri a, f, e c.

4.17.3 Problemi da risolvere con il calcolatore

Problema 1 Implementare almeno due metodi iterativi per la risolvere un sistema lineare con matrice dei coefficienti simmetrica definita positiva. Confrontarne le prestazioni sia in termini di *velocità asintotica di convergenza* che di complessità di tempo totale, risolvendo un campione significativo di problemi test, sia ben-condizionati che mal-condizionati. Confrontare i risultati ottenuti con le stime teoriche per la *velocità asintotica di convergenza*.

Problema 2 Implementare il metodo di Jacobi per risolvere un sistema lineare $Ax = b$, $n \times n$, dove A è tridiagonale con elementi diagonali uguali a 2 e con elementi sub-diagonali and superdiagonali uguali a -1. Si assuma $\mathbf{b} = \mathbf{0}$, così che la soluzione "esatta" sia $\mathbf{x} = \mathbf{0}$ e ad ogni iterazione l'errore sia uguale al valore corrente di $\mathbf{x}$. Come valore inizale si assuma

$$x_j = sin\left(\frac{jk\pi}{n+1}\right), \qquad j = 1, \ldots, n.$$

Problema 3 Sviluppare un elemento di software matematico che implementi il metodo iterativo di Gauss-Seidel e testarlo per la risoluzione dei seguenti sistemi:

a. $\begin{cases} 3x + y + z = 5 \\ x + 3y - z = 3 \\ 3x + y - 5z = -1 \end{cases}$

b. $\begin{cases} 3x + y + z = 5 \\ 3x + y - 5z = -1 \\ x + 3y - z = 3 \end{cases}$

Analizzare cosa accade quando i due sistemi sono risolti mediante metodo di eliminazione di Gauss *senza* pivoting.

Problema 4 Applicare il metodo iterativo di Gauss-Seidel al sistema $Ax = b$ con:

$$A = \begin{bmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{bmatrix} \qquad b = \begin{bmatrix} 0.88824 \\ 0.74988 \end{bmatrix}$$

Utilizzare, come punto iniziale, $(0.33116, 0.7)^T$ e spiegare cosa accade.

Problema 5 Per ciascuno dei sistemi seguenti:

1. $\begin{cases} 4x - y = 15 \\ x + 5y = 9 \\ 3x + y - 5z = -1 \end{cases}$

2. $\begin{cases} 8x - 3y = 10 \\ -x + 4y = 6 \end{cases}$

3. $\begin{cases} -x + 3y = 1 \\ 6x - 2y = 2 \end{cases}$

4. $\begin{cases} 2x + 3y = 1 \\ 7x - 2y = 1 \end{cases}$

5. $\begin{cases} 5x - y + z = 10 \\ 2x + 8y - z = 11 \end{cases}$

6. $\begin{cases} 2x + 8y - z = 11 \\ 5x - y + z = 10 \\ -x + y + 4z = 3 \end{cases}$

7. $\begin{cases} x - 5y - z = -8 \\ 4x + y - z = 13 \\ 2x - y - 6z = -2 \end{cases}$

8. $\begin{cases} 4x + y - z = 13 \\ x - 5y - z = -8 \\ 2x - y - 6z = -2 \end{cases}$

- porre $x^{(0)} = \underline{0}$ ed utilizzare i metodo di Jacobi per calcolare le iterate: $x^{(1)}$, $x^{(2)}$ e $x^{(3)}$. Il metodo converge verso la soluzione?
- Porre $x^{(0)} = \underline{0}$ ed utilizzare i metodo di Gauss-Seidel per calcolare le iterate: $x^{(1)}$, $x^{(2)}$ e $x^{(3)}$. Il metodo converge verso la soluzione?
- Scrivere un elemento di software matematico che implementi i metodi di Jacobi e Gauss-Seidel.

Problema 6 Calcolare la soluzione dei seguenti sistemi *tridiagonali* di ordine 50:

a.
$$\begin{cases} 4m_1 + & m_2 & & = 3 \\ m_1 + & 4m_2 + & m_3 & = 3 \\ m_2 + & 4m_3 + & m_4 & = 3 \\ m_3 + & 4m_4 + & m_5 & = 3 \\ \vdots & \vdots & \vdots & \vdots \\ m_{48} + & 4m_{49} + & m_{50} & = 3 \\ & m_{49} + & 4m_{50} & = 3 \end{cases}$$

b.
$$\begin{cases} 4m_1 + & m_2 & & = 1 \\ m_1 + & 4m_2 + & m_3 & = 2 \\ m_2 + & 4m_3 + & m_4 & = 1 \\ m_3 + & 4m_4 + & m_5 & = 2 \\ \vdots & \vdots & \vdots & \vdots \\ m_{48} + & 4m_{49} + & m_{50} & = 1 \\ & m_{49} + & 4m_{50} & = 2 \end{cases}$$

Problema 7 Utilizzare il metodo iterativo di Gauss-Seidel per risolvere il seguente sistema lineare *a banda* di ordine 50:

$$\begin{cases} 12x_1 & -2x_2 & +x_3 & & & & = 5 \\ -2x_1 & +12x_2 & -2x_3 & +x_4 & & & = 5 \\ x_1 & -2x_2 & +12x_3 & -2x_4 & +x_5 & & = 5 \\ x_2 & -2x_3 & +12x_4 & -2x_5 & +x_6 & & = 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_{46} & -2x_{47} & +12x_{48} & -2x_{49} & +x_{50} & & = 5 \\ & x_{47} & -2x_{48} & +12x_{49} & -2x_{50} & & = 5 \\ & & x_{48} & -2x_{49} & +12x_{50} & & = 5 \end{cases}$$

Problema 8 Si consideri un sistema lineare:

$$Ax = b$$

dove

$$A = \begin{bmatrix} 10 & -1 & 2 & 3 \\ 1 & -1 & -1 & 2 \\ 2 & 3 & 20 & -1 \\ 3 & 2 & 1 & 20 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 14 \\ 12 \\ 24 \\ 26 \end{bmatrix}.$$

Dopo avere discusso la convergenza dei metodi iterativi di Jacobi e di Gauss-Seidel per ciascun metodo, sviluppare una function `MATLAB` per risolvere il sistema lineare. Inoltre,

1. determinare le velocità asintotiche di convergenza per ciascun metodo;
2. verificare i risultati ottenuti con le stime sperimentali dedotte applicando le function `MATLAB` elaborate.

Bibliografia

[1] Axelsson O. - *Iterative Solution Methods* - Cambridge University Press, 1994.

[2] Barlett R. et al. - *Templates for the Solution of Linear Systems. Building Blocks for Iterative Methods* - SIAM, Philadelphia, 1993.

[3] Cheney W., Kincaid D. R. - *Numerical Mathematics and Computing* - sixth ed., Pacific Grove, CA, Brooks/Cole, 2007.

[4] Davis T. A. - *UMFPACK* - `http://www.cise.ufl.edu/research/sparse/umfpack/`.

[5] Demmel J. W. - *Applied Numerical Linear Algebra* - SIAM, Philadelphia, 1997.

[6] Fong K. W., Jefferson T. H., Suyehiro T., Walton L. - *SLATEC Common Mathematical Library* - (1993), `http://www.netlib.org/slatec/`.

[7] Greenbaumm A. - *Iterative Methods for Solving Linear Systems* - SIAM, Philadelphia, 1997.

[8] Hageman L. A., Young D. M. - *Applied Iterative Methods* - Academic Press, 1981.

[9] *Journal of Computational and Applied Mathematics* (JCAM) - `http://www.elsevier.com/locate/cam/`.

[10] *ITPACK 2C* - `http://www.netlib.org/itpack/`.

[11] Kelley C. T. - *Iterative Methods for Linear and Nonlinear Equations* - SIAM, Philadelphia, 1995.

[12] Lancaster P., Tismenetsky M. - *The Theory of Matrices* - second edition, Academic Press, 1985.

[13] Numerical Algorithm Group - *NAg Library Manual, Mark 22* - Oxford (2009), `http://www.nag.com/`.

[14] Ortega J. M., Rheinboldt W. C. - *Iterative Solution of Nonlinear Equations in Several Veriables* - Academic Press, 1970.

[15] Ortega J. M. - *Matrix Theory* - Plenum Press, 1988.

[16] *Portable Extensible Toolkit for Scientific Computation* (PETSc) - http://www.mcs.anl.gov/petsc/petsc.html.

[17] *PREQN* - http://www.netlib.org/toms/809.

[18] Rice J. - *Matrix Computation and mathematical software* - McGrawHill, 1981.

[19] Saad Y. - *Iterative Methods for Sparse Linear Systems* - PWS Publishing Co, Boston, 2003.

[20] Saad Y. - *SPARSKIT: A basic tool-kit for sparse matrix computations* - http://www-users.cs.umn.edu/∼saad/software/SPARSKIT/sparskit.html.

[21] Seager Mark. K. - *SLAP, The Sparse Linear Algebra Package* - http://www.netlib.org/slap/.

[22] *Sparse Basic Linear Algebra Subprograms* (sparse BLAS) - http://math.nist.gov/spblas/.

[23] Stewart G. W. - *Afternotes on Numerical Analysis* - SIAM, Philadelphia, 1998.

[24] Stewart G. W. - *Introduction to matrix computations* - Academic Press, 1973.

[25] Stewart G. W. - *Matrix Algorithms Volume I* - SIAM, 1998.

[26] Stewart G. W. - *Matrix Algorithms Volume II* - SIAM, 2001.

[27] Stewart G. W., Sun Guang-Ji - *Matrix Perturbation Theory* - Elsevier Publishers, 1990.

[28] *Transactions on Mathematical Software (TOMS)* - http://www.netlib.org/toms.

[29] Trefethen L. N., Bau D. - *Numerical Linear Algebra* - SIAM, Philadelphia, 1997.

[30] *The Trilinos Project* - http://trilinos.sandia.gov/.

[31] Varga R. S. - *Matrix Iterative Analysis* - Prentice Hall, Englewood Cliffs, NJ, 1962.

[32] Van der Vost H. A. - *Iterative Krylov methods for large linear systems* - Cambridge, 2003.

[33] Visual Numerics - *IMSL C Numerical Library version 7.0 (2008), IMSL Fortran Numerical Library Version 6.0 (2007)* - http://www.vni.com.

[34] http://www.mathworks.com/.

[35] http://www.netlib.org/linalg/.

[36] Young D. M. - *Iterative Solution of Large Linear Systems* - Academic Press, NY, 1971.

Capitolo 5

La Trasformata discreta di Fourier e l'algoritmo FFT

5.1 Introduzione

In questo capitolo, trattiamo algoritmi per il calcolo di somme del tipo:

$$S_j = \sum_{k=0}^{N-1} s_k e^{-\frac{2\pi}{N} ijk} \ , \qquad i = \sqrt{-1}\,, \quad j = 0, 1, \ldots, N-1 \tag{5.1}$$

dove $\{s_k\}_{k=0,\ldots,N-1} \in \mathbb{C}^N$ è un vettore le cui componenti sono numeri complessi [1].

[1] Utilizzando l'identità di Eulero:

$$e^{i\theta} = \cos\theta + i\sin\theta \quad , \ \theta \in \Re, \tag{5.2}$$

segue che:

$$S_j = \sum_{k=0}^{N-1} (Re(s_k) + iIm(s_k)) \left[\cos\left(\frac{2\pi jk}{N}\right) - i\sin\left(\frac{2\pi jk}{N}\right)\right] \tag{5.3}$$

e quindi,

$$Re(S_j) = \sum_{k=0}^{N-1} \left[Re(s_k)\cos\left(\frac{2\pi jk}{N}\right) + Im(s_k)\sin\left(\frac{2\pi jk}{N}\right)\right] \tag{5.4}$$

e

$$Im(S_j) = \sum_{k=0}^{N-1} \left[Im(s_k)\cos\left(\frac{2\pi jk}{N}\right) - Re(s_k)\sin\left(\frac{2\pi jk}{N}\right)\right]. \tag{5.5}$$

Se, in particolare, i valori s_k sono numeri reali, allora:

$$Re(S_j) = \sum_{k=0}^{N-1} \left[s_k\cos\left(\frac{2\pi jk}{N}\right)\right] \tag{5.6}$$

e

La somma (5.1), a partire da un vettore $\{s_k\}_{k=0,\ldots,N-1}$, definisce il vettore $\{S_k\}_{k=0,\ldots,N-1}$. A tale operatore, come verrà precisato in seguito, viene dato il nome di **Trasformata Discreta di Fourier (DFT)**.
Come è facile osservare dalla (5.1), il calcolo diretto della DFT ha una complessità dell'ordine di $\mathcal{O}(N^2)$. Negli ultimi anni sono stati messi a punto versioni sempre più specializzate dell' algoritmo **FFT (Fast Fourier Transform)** originariamente proposto da Cooley e Tukey, nel 1965. In questo capitolo, dopo aver introdotto l'operatore DFT e alcune sue proprietà fondamentali, descriviamo l'idea e la metodologia alla base degli algoritmi FFT. Vedremo come, attraverso l'approccio *divide et impera,* il calcolo di una DFT di lunghezza $N = r \cdot q$ possa essere ricondotto a quello di q DFT di lunghezza r, e così proseguendo, ottenendo, ad esempio se $N = 2^m$, una riduzione della complessità di tempo da $\mathcal{O}(N^2)$ a $\mathcal{O}(N \log_2 N)$. Lo stesso approccio, alla base del software di libreria che implementa gli algoritmi FFT, combinato con tecniche di ottimizzazione delle prestazioni implementate dinamicamente in fase di compilazione e di esecuzione (*AEOS - Automated Empirical Optimization of Software*) ha fatto della FFTW (*Fast Fourier Transform in the West)*, la libreria sviluppata nel 1999 da M. Frigo e S. Johnson presso il MIT *(Massachusetts Institute of Technology)* per il calcolo della DFT di un vettore di lunghezza N, il miglior prodotto software in termini di efficienza, accuratezza ed affidabilità (*J. H. Wilkinson Prize for Numerical Software, 2000*).

♣ **Esempio 5.1.** Anche se è consuetudine attribuire a Cooley e Tukey la prima formulazione di un algoritmo della classe FFT per il calcolo della DFT, in effetti già il trattato sull'interpolazione scritto da *Carl Friedrich Gauss* nel 1805, pubblicato postumo nel 1866, contiene un chiaro riferimento all'idea che sottende il calcolo veloce di una DFT mediante algoritmi FFT [16]. A tal proposito, i contributi di Gauss furono citati solo a partire dal 1904 nell'enciclopedia matematica di Burkhardt e successivamente nel 1977 da Goldstine [15].
Consideriamo, quindi, il problema del quale si interessò Gauss [11]. Nella tabella 5.1 si riportano, così come apparsi nel lavoro di Gauss, i valori indicanti la posizione dell'asteroide Pallas, espressa in termini delle variabili (θ, X) che esprimono rispettivamente l'ascensione e la declinazione [2].
A partire da tali misure il problema consisteva nel disegnare la traiettoria dell'asteroide utilizzando un modello interpolante che fosse basato su un polinomio trigonometrico $p(\theta)$, di grado $N+1$, del tipo:

$$p(\theta) = a_0 + \sum_{i=1}^{N} \left[a_k \cos\left(\frac{2\pi k\theta}{360}\right) + b_k \sin\left(\frac{2\pi k\theta}{360}\right)\right] + a_N \cos\left(\frac{2\pi(N+1)\theta}{360}\right).$$

Poiché i dati a disposizione sono 12, come indicato nella tabella, e poiché la costruzione del polinomio $p(\theta)$ comporta la determinazione di $2(N+1)$ coefficienti in questo caso fu necessario fissare $N = 5$. Indicate con (θ_k, X_k), $k = 0, \ldots, 11$ le coppie relative ai valori dell'ascensione e declinazione riportati

$$Im(S_j) = \sum_{k=0}^{N-1} \left[-s_k \sin\left(\frac{2\pi jk}{N}\right)\right]. \tag{5.7}$$

[2] Ascensione e Declinazione sono le coordinate utilizzate in astronomia per localizzare la posizione di un oggetto sulla sfera celeste. Tali coordinate sono analoghe alle coordinate tipicamente utilizzate per definire la posizione di un oggetto sulla sfera terrestre, longitudine e latitudine.

Ascensione	0	30	60	90	120	150
Declinazione	408	89	-66	10	338	807
Ascensione	180	210	240	270	300	330
Declinazione	1238	1511	1583	1462	1183	804

Tabella 5.1: Posizione dell'asteroide Pallas espressa in termini delle coordinate (l'Ascensione è espressa in gradi e la Declinazione in minuti).

nella tabella, e imponendo le condizioni di interpolazione:

$$p(\theta_k) = X_k, \quad k = 0, \ldots, 11$$

si ottiene un sistema di equazioni lineari di ordine 12 la cui soluzione fornisce i coefficienti del polinomio p. Gauss per risolvere tale sistema, utilizzando opportunamente le proprietà di simmetria e periodicità delle funzioni trigonometriche scoprí un modo per *decomporre tale problema in sottoproblemi le cui soluzioni, una volta combinate tra loro, fornirono la soluzione del problema con* $N(3+4) = 5 \cdot 7 = 35$ *operazioni* [3]. Questo procedimento è proprio quello che è alla base dell'algoritmo di FFT. ♣

Da ora in poi indicheremo con w_N l'esponenziale complesso:

$$w_N = e^{\frac{2\pi i}{N}} \quad .$$

Le potenze w_N^k con $k = 0, 1, \ldots, N-1$, dette **radici primitive N-me dell'unità**[4], rappresentano le soluzioni nel campo complesso dell'equazione

$$z^N = 1, \quad \forall\, z \in \mathbb{C}\,,$$

e svolgono un ruolo particolarmente significativo nel calcolo di una DFT poichè la complessità computazionale di una DFT dipende, oltre che dalle operazioni floating point necessarie al calcolo della somma in (5.1), anche dalla loro valutazione. Come vedremo, alla base degli algoritmi FFT vi è una opportuna riformulazione della DFT (derivante dalla fattorizzazione del parametro N) che, sfruttando la periodicità e la simmetria delle funzioni trigonometriche, consente di evitare inutili e ridondanti valutazioni di tali funzioni.

[3] L'algoritmo di eliminazione dello stesso Gauss avrebbe richiesto circa $1210 = \frac{12^3}{3} + \frac{12^2}{2}$ operazioni floating point.

[4] Geometricamente, le radici primitive N -me dell'unità si possono associare ai vertici del poligono regolare a N lati inscritto nel cerchio goniometrico. Tali grandezze si ripetono ciclicamente, ovvero $\omega_N^k = \omega_N^{(k\, modulo\, N)}$. Una utile proprietà delle radici N-me dell'unità è la seguente:

$$1 + \omega^r + \omega^{2r} + \ldots + \omega^{(N-1)r} = 0, \tag{5.8}$$

per ogni intero positivo $r \in \mathcal{N}$. L'appellativo *primitive* deriva dal fatto che ciascuna radice ha la proprietà di riprodurre con le potenze di esponente da 0 a $N-1$ tutte le radici N-me dell'unità.

♣ **Esempio 5.2.** Supponiamo di conoscere in $N = 16$ punti $x_k \in [0, 2\pi]$:

$$x_k = \frac{2\pi}{16}k, \quad k = 0, ..., 15, \tag{5.9}$$

i valori y_k, $k = 0, \ldots, 15$ di una funzione periodica[5].

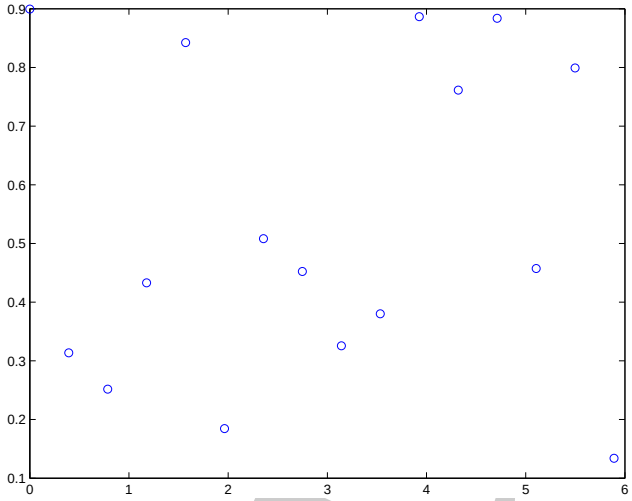

Figura 5.1: Rappresentazione dei punti di coordinate (x_k, y_k), $k = 0, 1, ..., 15$ con $x_k = \frac{2\pi}{N}k$, $y_k \in \mathbb{C}$, $N = 16$.

Al fine di costruire un modello che descriva i dati è ragionevole pensare ad un modello approssimante avendo assunto che le quantità y_k siano affette dall'errore (non trascurabile) introdotto dai dispositivi utilizzati per la loro aquisizione. Inoltre, essendo il fenomeno periodico, per descriverne l'andamento consideriamo come modello un polinomio trigonometrico di grado $N - 1 = 15$, del tipo:

$$p(x) = \frac{a_0}{2} + a_1 cos(x) + \ldots + a_{15} cos(15x)$$

In particolare, il problema è determinare i coefficienti a_j. A tal fine, imponiamo che

$$p^*(x) = \min_p ||y_i - p(x_i)||_2$$

ovvero, imponiamo che p^* sia il polinomio di migliore approssimazione relativo ai punti $(x_k, y_k)_{k=0,N-1}$ nel senso dei minimi quadrati, nello spazio dei polinomi trigonometrici di grado al più 15.
Tale problema conduce alla risoluzione del sistema (vedi **Capitolo 3**, **Parte prima**):

$$A^T \cdot Aa = A^T y$$

essendo:

$$A = \begin{pmatrix} 1 & \cos(x_1) & ... & ... & \cos(15x_1) \\ 1 & \cos(x_2) & ... & ... & \cos(15x_2) \\ 1 & \cos(x_3) & ... & ... & \cos(15x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos(x_{15}) & ... & ... & \cos(15x_{15}) \end{pmatrix}$$

[5]Piú in generale, i valori y_k possono essere valori corrispondenti ai nodi $x_k = \frac{T}{N}k$, $k = 0, \ldots, N-1$, con x_k appartenenti ad un generico intervallo $[0, T]$. In tal caso basta utilizzare la funzione $t = \frac{2\pi}{T}x$, che trasforma l'intervallo $[0, T]$ nell'intervallo $[0, 2\pi]$ e risolvere il problema in quest'ultimo intervallo.

$$A^T y = \begin{pmatrix} 2\sum_{i=0}^{15} y_i \\ 2\sum_{i=0}^{15} y_i \cos(x_i) \\ \vdots \\ \vdots \\ 2\sum_{i=0}^{15} y_i \cos(15x_i) \end{pmatrix}$$

e $a = (a_0, \ldots, a_{15})$, $y = (y_1, \ldots, y_{15})$.
Poiché [6]:

$$\sum_{i=0}^{15} \cos(jx_i)\cos(kx_i) = \begin{cases} 0 & j \neq k \\ \frac{16}{2} & j = k \neq 0 \\ 16 & j = k = 0 \end{cases} \tag{5.10}$$

$$\sum_{i=0}^{15} \cos(jx_i) = \begin{cases} 0 & j \neq 16 \\ 16 & j = 16 \end{cases} \tag{5.11}$$

la matrice del sistema diventa:

$$A^T \cdot A = \begin{pmatrix} 16 & 0 & 0 & \ldots & 0 \\ 0 & 16 & 0 & \ldots & 0 \\ 0 & 0 & 16 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \ldots & 16 \end{pmatrix}$$

la cui soluzione è, ovviamente:

$$a_j^* = \frac{2}{16}\sum_{k=0}^{15} y_k \cos\frac{2\pi kj}{16}, \qquad j = 0, \ldots, 15$$

Analogamente se:

$$p^*(x) = \frac{b_0}{2} + b_1^* \sin(x) + \ldots + b_{12}^* \sin(15x) \tag{5.12}$$

i coefficienti b_j^* sono:

$$b_j^* = \frac{2}{16}\sum_{k=0}^{15} y_k \sin\frac{2\pi kj}{16}, \qquad j = 0, 1, \ldots, 15$$

Tenendo conto delle (5.6) e (5.7) i coefficienti del polinomio p^*, a meno del segno e del fattore moltiplicativo 2/16, sono rispettivamente la parte reale e la parte immaginaria delle quantità S_j introdotte nella (5.1).

Il risultato al quale siamo pervenuti è di carattere generale. Posto $z = e^{ix}$, con $i = \sqrt{-1}$, assegnati N punti di coordinate $(z_k, y_k), k = 0, \ldots, N-1$, dove $z_k = e^{ix_k}$ con $x_k = \frac{2\pi}{N}k$ sussiste la seguente:

[6] Queste relazioni si ricavano utilizzando la proprietà (5.8) delle radici N-me dell'unità.

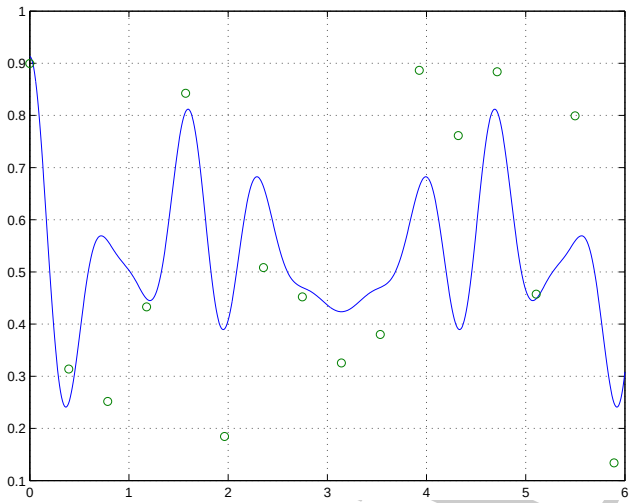

Figura 5.2: Grafico del polinomio trigonometrico p^* di migliore approssimazione nel senso dei minimi quadrati relativo ai punti $(x_i, y_i), \quad i = 0, 1, ..., 15$ dell'**esempio 5.2**.

Proposizione 5.1.1. *Esiste un unico polinomio trigonometrico del tipo*

$$p(x) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{N-1} z^{N-1}, \quad a_j = \frac{2}{N} \sum_{k=0}^{N-1} y_k e^{-\frac{2\pi}{N} ikj}, \qquad j = 0, 1, ..., N-1$$

interpolante gli N punti di coordinate $(z_k, y_k), k = 0, \ldots, N-1$.

e, analogamente per l'approssimazione nel senso dei minimi quadrati:

Proposizione 5.1.2. *Esiste un unico polinomio trigonometrico del tipo*

$$p(x) = b_0 + b_1 z + b_2 z^2 + \ldots + b_{M-1} z^{M-1}, \quad b_j = \frac{2}{N} \sum_{k=0}^{N-1} y_k e^{-\frac{2\pi}{N} ikj}, \qquad j = 0, 1, ..., M-1$$

con $M < N - 1$, di migliore approssimazione nel senso dei minimi quadrati relativo agli N punti $(z_k, y_k), k = 0, \ldots, N-1$.

♣

5.2 La Trasformata discreta di Fourier (DFT)

Come si è visto anche nell'esempio 5.2, un'applicazione che richiede il calcolo di una o più DFT è la costruzione del polinomio interpolante o approssimante per la descrizione di un insieme di dati con andamento periodico. A tale proposito, è importante notare che la conoscenza di N valori assunti da una funzione f in un intervallo $[0, T]$, in alcuni casi può essere sufficiente a determinare univocamente la funzione f a patto di considerare una opportuna distribuzione dei punti nell'intervallo che si sta considerando[7].

[7]Sussiste, infatti, il seguente (**Teorema di campionamento di Shannon** [26]):

Diamo la seguente:

Definizione 5.2.1. *Si definisce* **Trasformata Discreta di Fourier (DFT)** *di un vettore di numeri complessi* $\underline{f} = (f_0, \ldots, f_{N-1})$ *di lunghezza* N*, e si indica con DFT[f], il vettore di numeri complessi* $\underline{F} = (F_0, \ldots, F_{N-1})$ *ove:*

$$F_k = \sum_{j=0}^{N-1} f_j e^{-\frac{2\pi i}{N} jk} \qquad k = 0, \ldots N-1;\ i = \sqrt{-1} \tag{5.13}$$

Definizione 5.2.2. *Si definisce* **Trasformata Discreta Inversa di Fourier (IDFT)** *di un vettore le cui componenti sono numeri complessi* $\underline{F} = (F_0, \ldots, F_{N-1})$ *di lunghezza* N*, e si indica IDFT[F], il vettore di numeri complessi* $\underline{f} = (f_0, \ldots, f_{N-1})$ *ove:*

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{\frac{2\pi i}{N} jk} \qquad k = 0, N-1; i = \sqrt{-1} \tag{5.14}$$

I settori applicativi in cui si fa uso di DFT sono molteplici. D'altra parte, tale operatore si presenta come strumento fondamentale per la risoluzione di molti problemi della matematica numerica, come ad esempio, nella risoluzione di problemi di calcolo matriciale, se la matrice presenta particolari strutture, nella risoluzione di equazioni

Teorema 5.2.1. *Sia* f *una funzione la cui trasformata di Fourier è nulla al di fuori dell'intervallo* $[-f_{Ny}, f_{Ny}]$*. Siano assegnati i punti* $x_n = nh$*, con:*

$$h \leq 1/(2f_{Ny})$$

allora f *può essere univocamente ricostruita dai suoi campioni:*

$$f(x) = \sum_{n=-\infty}^{+\infty} f(x_n) \frac{\sin(\pi(x - x_n)/h)}{\pi(x - x_n)/h}$$

La quantità f_{Ny} è detta **frequenza di Nyquist**.

La serie al secondo membro è anche nota come *serie cardinale di f*, e la funzione f, somma delle serie cardinale, è detta *funzione cardinale di Witthaker*. Il teorema fu introdotto già da Witthaker nel 1915, e successivamente utilizzato da Shannon nel 1948 [26]. In pratica, nelle applicazioni accade che la trasformata di Fourier di una certa funzione si può considerare trascurabile da un certo intervallo in poi. In tal caso è possibile determinare il passo h in modo da controllare l'errore introdotto dalla sua rappresentazione in termini della *funzione cardinale di Witthaker*.

Se fissiamo l'intervallo $[0, T]$ e $h = \frac{T}{N}$, l'ipotesi che $h \leq 1/(2f_{Ny})$ implica che la massima frequenza di f che può essere calcolata è

$$f_{Ny} = \frac{1}{2T}$$

Il teorema stabilisce quindi che quanto più rapidamente la funzione f *oscilla* nell'intervallo $[0, T]$ (ovvero quanto maggiore è la sua frequenza $\omega = \frac{2\pi}{T}$) tanto più piccola deve essere la distanza dei punti. Inoltre, per ricostruire l'andamento di una funzione che nel suo periodo T ha un unico ciclo ($f_{N_y} = \frac{1}{T}$), dobbiamo conoscerla in almeno due punti ($h < \frac{T}{2}$). In caso contrario, nel ricostruire la funzione f utilizzando lo sviluppo in serie cardinale di Witthaker, le frequenze maggiori della frequenza di Nyquist si sovrappongono alle altre. Tale fenomeno nell'analisi di Fourier prende il nome di **aliasing**.

differenziali ordinarie e alle derivate parziali, nell'integrazione numerica di funzioni periodiche o oscillanti. L'algoritmo FFT è oggi uno dei 10 algoritmi con il maggior impatto scientifico e tecnologico [8].

5.2.1 Alcune proprietà della DFT

Descriviamo alcune delle proprietà più significative della DFT, ovvero consideriamo quelle proprietà che sono alla base delle applicazioni della DFT[9]. Le prime proprietà discendono direttamente dalla definizione dell'operatore DFT e riguardano la linearità, la periodicità e la simmetria. La linearità viene utilizzata specialmente nelle applicazioni in cui il vettore di cui bisogna calcolare la DFT si può decomporre nelle sue componenti armoniche e attraverso l'analisi della DFT di tali componenti si possono trarre informazioni significative sul vettore iniziale (questo è ad esempio il principio alla base dell'analisi armonica di funzioni periodiche). La proprietà di periodicità consente di prolungare *periodicamente* il vettore di cui calcolare la DFT senza alterarne il risultato numerico. Infine la simmetria trova applicazione negli schemi di memorizzazione alla base degli algoritmi per il calcolo di una DFT.

Proposizione 5.2.1. *Siano $f, g \in \mathbb{C}^N$ due vettori di numeri complessi, di lunghezza N e siano $\alpha\ , \beta \in \Re$, si ha:*

$$DFT[\alpha f + \beta g] = \alpha DFT[f] + \beta DFT[g]$$

ovvero la DFT è un operatore lineare.

Dimostrazione Dalla definizione di DFT discende che:

$$DFT[\alpha f + \beta g] = \sum_{j=0}^{N-1} (\alpha f + \beta g)_j \omega_N^{-jk} = \sum_{j=0}^{N-1} \alpha f_j \omega_N^{-jk} + \sum_{j=0}^{N-1} \beta g_j w_N^{jk} =$$

$$= \alpha \sum_{j=0}^{N-1} f_j \omega_N^{-jk} + \beta \sum_{j=0}^{N-1} g_j \omega_N^{-jk} = \alpha DFT[f] + \beta DFT[g].$$

■

[8] IEEE, Computing in Science and Engineering, 2000, n. 22.

[9] A tutte le proprietà dell'operatore DFT corrispondono analoghe proprietà dell'operatore di Trasformata di Fourier; d'altra parte, l'operatore DFT si può anche interpretare come discretizzazione della Trasformata di Fourier ottenuta mediante l'applicazione della formula trapezoidale composita all'integrale di Fourier.

♣ **Esempio 5.3.** Assegnati due vettori:

$$f = (1+i, -3, 5+7i, -2), \quad g = (3+4i, 4, 7, 1-i)$$

calcoliamo la DFT dei vettori f e g:

$$DFT[f] = (1+8i, -4-5i, 11+8i, -4-7i), \quad DFT[g] = (15+3i, -3-i, 5-5i, -5-7i)$$

Poiché $f+g = (4+5i, 1, 12+7i, -1-i)$, la DFT di h è:

$$DFT[h] = DFT[f+g] = (16+11i, -7-4i, 16-13i, -9) =$$

$$= (1+8i, -4-5i, 11+8i, -4-7i) + (15+3i, -3-i, 5-5i, -5-7i) = DFT[f] + DFT[g].$$

♣

Proposizione 5.2.2. *Il vettore $F = DFT[f]$, DFT del vettore di numeri complessi $f_0, \ldots, f_{N-1}$ di lunghezza N è N-periodico, ovvero:*

$$F_{N+k} = F_k, \quad \forall k \in \mathcal{N} \cup \{0\}$$

Dimostrazione Questa proprietà discende direttamente dalla periodicità dell'esponenziale complesso:

$$\omega_N^{-(k+N)n} = \omega_N^{-nk}$$

■

Le proprietà seguenti, che enunciamo solamente, lasciando la dimostrazione per esercizio, hanno interessanti implicazioni in termini di complessità di tempo e di spazio nel calcolo di una DFT di un vettore di numeri reali o di numeri immaginari puri[10].

Proposizione 5.2.3. *Valgono le seguenti proprietà* [11] *:*

1. *Se $f \in \Re^N$ è un vettore di numeri reali, allora $F = DFT[f]$ è un vettore hermitiano simmetrico, ovvero:*
$$F_k = \bar{F}_{N-k} \quad k = 1, ..., N/2$$
avendo indicato con $\bar{F}_{N-k}$ il numero complesso coniugato di F_{N-k}.

2. *Se $f \in \mathbb{C}^N$ è un vettore hermitiano simmetrico, ovvero $f_k = \bar{f}_{N-k}$, allora $F = DFT[f]$ è un vettore di numeri reali.*

[10] Nel seguito, all'occorrenza, il vettore DFT si intende prolungato per periodicità

[11] Nel seguito si farà riferimento al numero intero $N/2$, volendo intendere, nel caso in cui N sia dispari, alla parte intera di $N/2$ aumentata di un'unità.

3. *Se $g \in \mathbb{C}^N$ è un vettore le cui componenti sono numeri immaginari allora $G = DFT[g]$ è un vettore hermitiano antisimmetrico, ovvero:*

$$G_k = -\bar{G}_{N-k} \quad k = 1, ..., N/2$$

4. *Se $g \in \mathbb{C}^N$ è un vettore hermitiano antisimmetrico, ovvero $g_k = \bar{g}_{N-k}$, allora $G = DFT[g]$ è un vettore le cui componenti sono numeri immaginari.*

♣ **Esempio 5.4.** Calcoliamo la DFT del vettore $f = (1, 2, 3, 4, 5, 6)$ e del vettore $g = (i, -2i, 5i, -4i, 3i, 6i)$:

$$F = DFT[f] = (21, -3 - 5.1962\,i, -3 - 1.7321\,i, -3, -3 + 1.7321\,i, -3 + 5.1962\,i)$$

$$G = DFT[g] = (9\,i, -5.1962 + 3\,i, -8.6603 + 9\,i, -9i, 8.6603i + 9\,i, 5.1962 + 3\,i)$$

F è un vettore hermitiano simmetrico e G è hermitiano antisimmetrico. ♣

♣ **Esempio 5.5.** Calcoliamo la DFT di due vettori reali $x, y \in \Re^N$.
Costruiamo il vettore di numeri complessi $z = x + iy$, e consideriamo il vettore $Z = DFT[z]$. Poiché l'operatore DFT è lineare, segue che $Z = DFT[z] = DFT[x] + iDFT[y]$. Sia $X = DFT[x]$ e $Y = DFT[y]$, applicando la proprietà 1 della proposizione 5.2.3, risulta

$$X_k = \bar{X}_{N-k}, \quad \bar{X}_k = X_{N-k} \quad k = 1, \ldots, N/2$$

e

$$Y_k = \bar{Y}_{N-k}, \quad \bar{Y}_k = Y_{N-k} \quad k = 1, \ldots, N/2$$

Dall'uguaglianza:

$$Z_k = X_k + iY_k \ , \quad k = 0, \ldots, N-1 \tag{5.15}$$

sostituendo k con $N - k$ e passando ai coniugati segue:

$$\bar{Z}_{N-k} = X_k - iY_k \quad k = 0, \ldots, N-1 \tag{5.16}$$

Addizionando e sottraendo la (5.15) e la (5.16), si ha:

$$X_k = 1/2[\bar{Z}_{N-k} + Z_k], \quad Y_k = i/2[\bar{Z}_{N-k} - Z_k], \quad k = 1, \ldots, N/2$$

ovvero, per ottenere i due vettori X e Y, DFT rispettivamente dei vettori x e y, invece di calcolare due DFT di due vettori di numeri reali, basta calcolare una sola DFT di un vettore di numeri complessi. Inoltre, poiché si tratta di vettori hermitiani simmetrici, si ha anche un risparmio di occupazione di memoria.
Sia ad esempio, $N = 5$ e:

$$x = (5, 2, 1, -1, 3), \quad y(-2, 4, 1, 7, 3)\,;$$

costruiamo il vettore

$$z = x + iy = (5 - 2\,i, 2 + 4\,i, 1 + i, -1 + 7\,i, 3 + 3\,i)\ ;$$

calcoliamo la DFT di z:

$$Z = DFT[z] = (10 + 13\,i, 3.9694 - 6.5335\,i, 7.249 - 2.7011\,i, -5.3392 - 7.6809\,i, 9.1207 - 6.0845i)$$

da cui, applicando le relazioni

$$X_k = 1/2[\bar{Z}_{N-k} + Z_k], \quad Y_k = i/2[\bar{Z}_{N-k} - Z_k]\ , \quad k = 1, \ldots, N/2 \quad (X_0 = Re(Z_0),\, Y_0 = Im(Z_0))$$

si ottiene il vettore DFT di x e y. ♣

♣ **Esempio 5.6.** Calcoliamo la DFT di un vettore $x \in \Re^{2N}$.
Costruiamo i vettori $h = (h_j)_{j=0,\ldots,N-1}$ e $g = (g_j)_{j=0,\ldots,N-1}$ dove

$$h_j = f_{2j} \quad , \quad g_j = f_{2j+1}$$

e

$$z = h + ig$$

Siano $Z = DFT[z]$, $H = DFT[h]$ e $G = DFT[g]$. Osserviamo innanzitutto che, dalla definizione di DFT segue:

$$X_k = DFT[x]_k = DFT[h]_k + e^{-\frac{i\pi}{N}k} DFT[g]_k \quad , k = 0, \ldots, N-1$$

inoltre, poiché il vettore x è reale, il vettore $X = DFT[x]$ è hermitiano simmetrico cioè $X_{2N-k} = \bar{X}_k$. Ció significa che per calcolare la DFT del vettore x basta calcolare le sue prime N componenti. Il problema è stato quindi ricondotto al calcolo delle DFT dei due vettori di numeri reali, h e g, di lunghezza N. Come abbiamo già visto nell'esempio 5.5, questo si puó ottenere effettuando il calcolo della DFT del vettore z. In sintesi, abbiamo calcolato la DFT di x, vettore di numeri reali di lunghezza $2N$, calcolando la DFT di z, vettore di numeri complessi di lunghezza N.

Sia, ad esempio, $N = 3$ e consideriamo il vettore

$$x = (2, -5, 8, -3, -2, 4)$$

di lunghezza $2N = 6$. Siano $h = (2, 8, -2)$ e $g = (-5, -3, 4)$. Costruiamo il vettore

$$z = (2 - 5i, 8 - 3i, -2 + 4i)$$

e calcoliamo la sua DFT:

$$Z = (8 - 4i, -7.06 - 14.16i, 5.06 + 3.16i)$$

da cui si ricava che

$$DFT[h] = (8, -1 - 8.6i, -1 + 8.6i), \quad DFT[g] = (-4, -5.5 + 6.06i, -5.5 - 6.06i)$$

e, infine, ricaviamo le componenti di $X = DFT[x]$:

$$X = (4, 1.5 - 0.8i, -3.5 + 16.4i, 12, -3.5 - 16.4i, 1.5 + 0.8i)$$

♣

Sia:

$$W = \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & w_N^{-2(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \dots & \dots & \omega_N^{-(N-1)(N-1)} \end{pmatrix}$$

La matrice W è detta **matrice di Fourier**.

Proposizione 5.2.4. *La matrice $\widetilde{W} = \frac{1}{\sqrt{N}} W$ è unitaria , cioè $\widetilde{W} \cdot \widetilde{W}^H = I$, dove $\widetilde{W}^H$ è la matrice trasposta della matrice coniugata di $\widetilde{W}$.*

Dimostrazione

$$\begin{aligned} \tilde{W} \cdot \tilde{W}^T &= \frac{1}{N} \cdot \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & \omega^{-2(N-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \dots & \dots & \omega_N^{-(N-1)^2} \end{pmatrix} \cdot \begin{pmatrix} \omega_N^0 & \omega_N^0 & \omega_N^0 & \dots & \omega_N^0 \\ \omega_N^0 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ \omega_N^0 & \omega_N^{-2} & \omega_N^{-4} & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \omega_N^0 & \omega_N^{-(N-1)} & \omega_N^{-2(N-1)} & \dots & \omega_N^{-(N-1)^2} \end{pmatrix} \\ &= \frac{1}{N} \cdot \begin{pmatrix} \sum_{i=0}^{N-1} \omega_N^0 \cdot \omega_N^0 & \sum_{i=0}^{N-1} \omega_N^0 \cdot \omega_N^{-i} & \dots & \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-(N-1)i} \\ \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-i} & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-i} & \dots & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-(N-1)i} \\ \dots & \dots & \dots & \dots \\ \sum_{i=0}^{N-1} \omega_N^0 \cdot w^{-(N-1)i} & \sum_{i=0}^{N-1} \omega_N^{-i} \cdot w^{-(N-1)i} & \dots & \sum_{i=0}^{N-1} \omega_N^{-(N-1)i} \cdot w^{-(N-1)i} \end{pmatrix} \\ &= \frac{1}{N} \cdot \begin{pmatrix} N & 0 & 0 & \dots & 0 \\ 0 & N & 0 & \dots & 0 \\ 0 & 0 & N & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \end{aligned}$$

■

Il teorema che segue è alla base di tutte le applicazioni della DFT dove si fa uso dell'operatore di convoluzione e dell'operatore inverso di deconvoluzione. In breve, l'-operazione di convoluzione [12] tra due vettori a e b, trasforma il vettore a in un altro in cui ciascuna componente è ottenuta come media pesata della rispettiva componente e di quelle consecutive, laddove il peso è dato dalle componenti del vettore b. Tale operazione

12

Definizione 5.2.3. ***(Prodotto di Convoluzione)***
Dati due vettori $a = (a_0, a_1, ..., a_{N-1})$ e $b = (b_0, b_1, ..., b_{N-1})$ si definisce prodotto di convoluzione il vettore:

$$c = a * b$$

si usa spesso nell'elaborazione di dati che descrivono fenomeni con andamento periodico con l'obiettivo di ridurre *picchi* e brusche variazioni nel loro andamento che spesso denotano la presenza di *rumore*. L'operazione di convoluzione si esprime in termini matriciali come prodotto di una matrice *circolante* [13] per un vettore. Per tale motivo, utilizzando opportunamente la proprietà dell'operatore DFT nel calcolo del prodotto di convoluzione, si derivano in maniera naturale gli algoritmi basati sulla DFT per il calcolo con matrici circolanti.

Teorema 5.2.2. [di Convoluzione]
Siano $F = DFT[f]$ e $G = DFT[g]$ le DFT di due vettori f e g di lunghezza N. La DFT del prodotto di convoluzione tra f e g, $f * g$, è il vettore ottenuto come prodotto puntuale dei due vettori, ovvero G è il vettore le cui componenti sono ottenute effettuando il prodotto componente per componente dei vettori F e G:

$$DFT[f * g] = (F_0 G_0, F_1 G_1, ..., F_{N-1} G_{N-1}) = F. * G$$

avendo indicato con $.*$ il prodotto puntuale dei vettori F e G.

♣ **Esempio 5.7.** Consideriamo il vettore $\underline{f} = \{f_n\}_{n=0,...,23}$, di $N = 24$ punti ottenuti valutando la funzione

$$f(x) = \cos(2\pi x) + 1/2\cos(10\pi x) + 1/3\cos(12\pi x)$$

in $x_n = n \cdot \Delta x, \Delta x = 1/24$, $n = 0, \ldots, 23$. Assumiamo, inoltre, che tale vettore sia 24-periodico, ovvero $f_{n\pm 24} = f_n$. Consideriamo il vettore $\{g_n\}_{n=0,...,23}$, con

$$g_n = \frac{1}{8} f_{n-2} + \frac{1}{4} f_{n-1} + \frac{1}{4} f_n + \frac{1}{4} f_{n+1} + \frac{1}{8} f_{n+2}$$

Introdotto il vettore

$$\underline{h} = (\ldots, 0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \ldots)$$

---

la cui $j-esima$ componente è data da:

$$c_j = \sum_{k=0}^{j} a_k \cdot b_{j-k} \quad j = 0, ..., 2N-2.$$

[13]

Definizione 5.2.4. (Matrice Circolante)
Assegnato un vettore $v = (v_0, v_1, ..., v_{N-1})$ si definisce **matrice circolante** *di lunghezza N, generata dal vettore v, una matrice $C := C(v)$*

$$C := C(v) = \begin{bmatrix} v_0 & v_1 & \cdots & v_{N-2} & v_{N-1} \\ v_{N-1} & v_0 & \cdots & v_{N-3} & v_{N-2} \\ v_{N-2} & v_{N-1} & \cdots & v_{N-4} & v_{N-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v_1 & v_2 & \cdots & v_{N-1} & v_0 \end{bmatrix}.$$

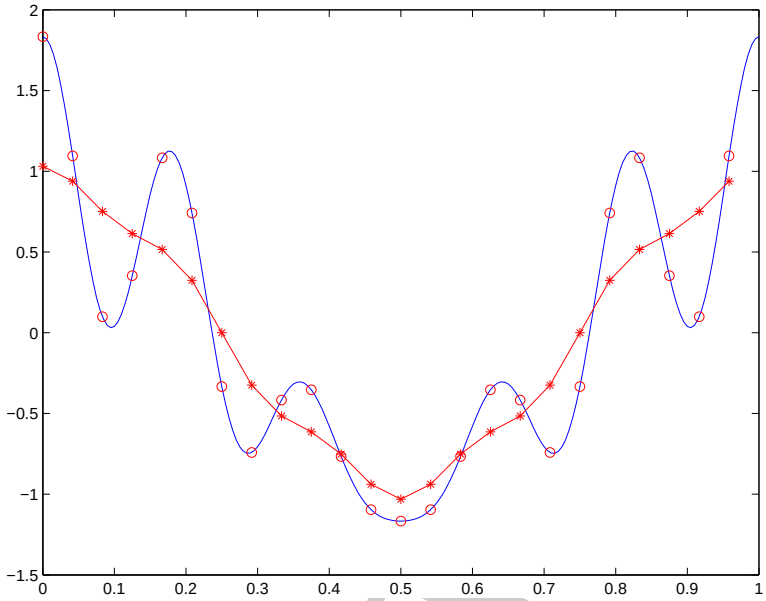

Figura 5.3: Grafico di $f(x) = cos(2\pi x) + \frac{1}{2}cos(10\pi x) + \frac{1}{3}cos(12\pi x)$ in $[0, 1)$. Con il simbolo 'o' si indicano i valori $f_n = f(x_n)$ e con il simbolo '*' si indicano i valori g_n, $n = 0, \ldots, 23$.

24-periodico, con $h_{\pm 2} = 1/8$, $h_{\pm 1} = 1/4$, $h_0 = 1/4$, si ha

$$g_n = \sum_{j=0}^{n} f_j \cdot h_{n-j} = \underline{f} * \underline{h}$$

In altre parole, il vettore $\{g_n\}$ è il prodotto di convoluzione del vettore $\{f_n\}$ con il vettore $\{h_n\}$. Come anche illustrato in Fig.5.3, il vettore $\{f_n\}$ è costituito dai valori assunti dalla funzione f, composta da tre funzioni cosinusoidali con frequenza rispettivamente $1, 5$ e 6; congiungendo a due a due i punti di coordinate (x_n, g_n) si ottiene una curva con meno picchi rispetto al grafico della funzione f e con un andamento molto più *dolce*. Il procedimento appena descritto è un esempio dell'operazione che trasforma un vettore, le cui componenti hanno valori che differiscono tra loro in maniera significativa, in un altro in cui picchi e brusche variazioni vengono smorzati.

♣

♣ **Esempio 5.8.** Consideriamo la matrice circolante C generata dal vettore $v = (5, 2, 7, 9, 4)$:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 \\ 4 & 5 & 2 & 7 & 9 \\ 9 & 4 & 5 & 2 & 7 \\ 7 & 9 & 4 & 5 & 2 \\ 2 & 7 & 9 & 4 & 5 \end{bmatrix}.$$

Si vuole utilizzare la DFT per il calcolo del prodotto matrice per vettore

$$C \cdot x' = y'$$

dove $x = (1, 2, 3, 4, 5)$ e $y = (86, 93, 75, 67, 84)$.

Applicando la proprietà delle matrici circolanti:

$$C = W^{-1}DW$$

dove W è la matrice di Fourier e $D = diag(F)$, con $F = DFT[v]$, si ha:

$$Cx' = y' \quad \Leftrightarrow \quad W^{-1}DWx' = y' \quad \Leftrightarrow \quad DWx' = Wy'$$

e

$$Wx' = DFT[x'], \quad Wy' = DFT[y'],$$

posto $X = DFT[x']$, $Y = DFT[y']$ e $F = DFT[v]$ si ha

$$Cx' = y' \quad \Leftrightarrow \quad F. * X' = Y'$$

avendo indicato con $.*$ il prodotto puntuale [14] dei vettori F e X'. Essendo

$$F = (27., -6.0902 + 3.0777i, 5.0902 - 0.7265i, 5.0902 + 0.7265i, -6.0902 - 3.0777i)$$

e

$$X' = (15, -2.5 - 3.441i, -2.5 - 0.8123i, -2.5 + 0.8123i, -2.5 + 3.441i)$$

effettuando il prodotto puntuale $F. * X'$ si ha:

$$Y' = (405, 25.82 + 13.26i, -13.32 - 2.32i, -13.32 + 2.32i, 25.82 - 13.26i)$$

ed applicando la DFT inversa di Y' si ottiene il vettore y':

$$y' = IDFT[Y] = \begin{pmatrix} 86 \\ 93 \\ 75 \\ 67 \\ 84 \end{pmatrix}.$$

♣

Consideriamo una matrice circolante C, generata da un vettore v: $C = C(v)$. Per il calcolo del prodotto matrice per vettore:

$$C \cdot x' = y'$$

si ha:

$$C \cdot x' = y' \Leftrightarrow W^{-1}DW \cdot x' = y' \Leftrightarrow DW \cdot x' = Wy' \tag{5.17}$$

[14] Il prodotto puntuale tra due vettori, a e b, è il vettore le cui componenti sono ottenute moltiplicando le componenti omologhe dei due vettori a e b.

Poiché:

$$Wx' = DFT[x'] \quad \text{e} \quad Wy' = DFT[y'],$$

posto $X = DFT[x']$ e $Y = DFT[y']$ la (5.17) diventa:

$$C \cdot x' = y' \Leftrightarrow F. * X' = Y' \tag{5.18}$$

avendo indicato con $.*$ il prodotto puntuale dei vettori F e X'. Quindi per effettuare il prodotto $Cx' = y'$ si può:

1. calcolare $X = DFT[x']$;
2. calcolare $F = DFT[v]$;
3. calcolare il prodotto puntuale $F. * X' = Y'$;
4. calcolare la DFT inversa $IDFT[Y] = y'$;

e ciò richiede il calcolo di 2 DFT e 1 IDFT.
Analogamente deriva lo schema relativo alla risoluzione di un sistema lineare avente come matrice dei coefficienti una matrice circolante.

♣ **Esempio 5.9.** Consideriamo la seguente matrice:

$$T = \begin{bmatrix} 5 & 2 & 7 \\ 4 & 5 & 2 \\ 9 & 4 & 5 \end{bmatrix}.$$

La matrice T è una matrice di *Toeplitz*[15]. Notiamo che T è una sottomatrice della matrice circolante C:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 \\ 4 & 5 & 2 & 7 & 9 \\ 9 & 4 & 5 & 2 & 7 \\ 7 & 9 & 4 & 5 & 2 \\ 2 & 7 & 9 & 4 & 5 \end{bmatrix}$$

introdotta nell'esempio 5.8. Pertanto, viene naturale domandarsi se, anche in questo caso, analogamente a quanto fatto per le matrici circolanti, è possibile utilizzare la DFT per eseguire le operazioni di calcolo matriciale involventi T. Considerato allora il vettore

$$x = (1, 2, 3)\,,$$

[15] **Definizione 5.2.5.** *La matrice $T := (t_{ij})$, $i, j = 1, \ldots, n$, è una matrice di Toeplitz se ha elementi costanti lungo ciascuna diagonale, ovvero*

$$t_{ij} = r_{j-i}, \quad \forall i, j = 1, \ldots, n$$

con $r_{-n+1}, \ldots, r_0, \ldots, r_{n-1}$ numeri reali.

calcoliamo ad esempio il vettore y, prodotto di T per x, $T \cdot x = y$. In particolare, si ha $y = (86, 93, 75)$. Si osserva che se introduciamo il vettore $x' = (1, 2, 3, 0, 0)$, il prodotto $C \cdot x'$ fornisce come vettore risultante il vettore y' le cui prime tre componenti coincidono con le componenti del vettore $y = T \cdot x$. Per calcolare y basta calcolare $y' = C \cdot x'$, utilizzando, quindi, la DFT. ♣

In generale, una qualsiasi matrice di Toeplitz $T \in \Re^{n \times n}$, si può *immergere* in una matrice circolante, ovvero T si può riguardare come sottomatrice di una matrice circolante $C \in \Re^{(2n-1)\times(2n-1)}$, ottenuta considerando come prima colonna di C il vettore v di lunghezza $2n-1$ le cui prime n componenti coincidono con le n componenti del vettore relativo alla prima colonna di T, e le ultime $n-1$ componenti sono le componenti del vettore relativo alla prima riga di T, prese nell'ordine inverso. Quanto detto si può esprimere sinteticamente, utilizzando ad esempio la notazione `MATLAB`, come:

$$v = (T(1:n, 1), T(1, n:-1:2))$$

Di conseguenza, per le operazioni di calcolo matriciale involventi una matrice di Toeplitz, di dimensione n, come ad esempio il prodotto matrice per vettore $T \cdot x$, si può utilizzare la DFT a patto di:

- **immergere** la matrice di Toeplitz di dimensione n in una matrice circolante C di dimensione $2n-1$;
- **prolungare** il vettore x aggiungendo $n-1$ componenti nulle;
- calcolare il prodotto $C \cdot x = y$.

♣ **Esempio 5.10.** Consideriamo la matrice tridiagonale a blocchi così definita:

$$M = \begin{bmatrix} T & -I & & & & \\ -I & T & -I & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & -I \\ & & & & -I & T \end{bmatrix}$$

di dimensione $N \times N$, dove $N = n \times n$, I è la matrice identica di dimensione $n \times n$ e T è la matrice tridiagonale di dimensione $n \times n$:

$$T = \begin{bmatrix} 4 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & -1 \\ & & & & -1 & 4 \end{bmatrix}$$

Consideriamo il problema della risoluzione del sistema lineare [16]:

$$Mu = b, \quad M \in \Re^{N \times N}$$

La matrice M è anche simmetrica e a diagonale dominante. La matrice tridiagonale T è anch'essa simmetrica e a diagonale dominante. Inoltre, M è *semplice* e i suoi autovalori sono noti e sono esprimibili come funzioni trigonometriche. In particolare, esiste una matrice ortogonale Q tale che:

$$QMQ^T = S \Leftrightarrow M = Q^T S Q$$

dove:

$$S = diag(\lambda_{jk}), \quad \lambda_{jk} = 4 - 2\cos\left(\frac{j\pi}{n+1}\right) - 2\cos\left(\frac{k\pi}{n+1}\right)$$

e Q è la matrice ortogonale definita come;

$$Q = diag(F_n^T)\ P\ diag(F_n^T) \in \Re^{N \times N}$$

dove:

$$F_n = (f_{ij}), \quad f_{ij} = \sqrt{\frac{2}{\pi}} \sin\frac{\pi ij}{n+1} \quad i, j = 1, \ldots, n$$

e P è una matrice di permutazione.
Se per il calcolo di $u = M^{-1}b$ sostituiamo a M l'espressione $M = Q^T S Q$, si ha:

$$u = Q^T S^{-1} Q b$$

ovvero u si può ottenere:

1. calcolando $b_1 = Qb$,
2. calcolando $b_2 = S^{-1}b_1$,
3. calcolando $b_3 = Q^T b_2$.

Questo procedimento può risultare particolarmente efficiente se si tiene conto che in ciascun passo possiamo utilizzare la DFT, e quindi l'algoritmo FFT. Osserviamo, infatti, che poiché:

$$\sin\left(\frac{\pi kj}{N}\right) = \frac{e^{\frac{\pi kj}{N}} - e^{\frac{-\pi kj}{N}}}{2i} = \frac{w_{N'}^{kj/2} - w_{N'}^{-kj/2}}{2i}$$

dove

$$w = e_{N'}^{i2\pi/N'}, \quad N' = 2N,$$

segue:

$$-2i \sum_{j=1}^{N-1} a_j \sin\left(\frac{\pi kj}{N}\right) = \sum_{j=1}^{N-1} a_j w_{N'}^{-kj} - \sum_{j=1}^{N-1} a_j w_{N'}^{kj}$$

Posto $r = N' - j$ si ha:

$$w^{kj} = w_{N'}^{kN'-kr}$$

[16] Come già visto nell'esempio 4.81 del **Capitolo 4**, questa matrice si ottiene, ad esempio, considerando l'equazione di Poisson, ovvero l'equazione differenziale alle derivate parziali di tipo ellittico:

$$\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} = f(x, y)$$

su un dominio quadrato $\Omega = (0, L) \times (0, L)$, con condizioni di Dirichlet sulla frontiera $u(x, y) = g(x, y), (x, y) \in \partial\Omega$ e per la sua risoluzione numerica discretizziamo le derivate parziali all'interno del dominio Ω con uno schema alle differenze finite su 5 punti.

da cui:

$$\sum_{j=1}^{N-1} \sin\left(\frac{\pi kj}{N}\right) = \sum_{j=0}^{N'-1} c_j w_{N'}^{-kj}$$

dove

$$c_j = \begin{cases} 0 & j = 0, j = N \\ ia_j & 1 \leq j \leq N-1 \\ -ia_{N'-j} & N+1 \leq j \leq N'-1 \end{cases}$$

In altre parole, il vettore

$$s_k = 2\sum_{j=1}^{N-1} a_j \sin\left(\frac{\pi kj}{N}\right), \quad k = 1, 2, \ldots, N-1$$

che definisce la trasformata discreta di seni del vettore $\{a_j\}_{j=1,2,\ldots,N-1}$ può essere ottenuto calcolando la DFT del vettore $\{c_j\}$, di lunghezza $2N$. Analogo discorso vale per il calcolo della trasformata discreta di soli coseni. Questo significa, ad esempio, che per calcolare il vettore b_1 al primo passo è necessario calcolare $2n$ DFT, analogamente per ottenere b_3 al terzo passo è necessario effettuare $2n$ DFT per un totale complessivo di $4n$ DFT di lunghezza $2N$. ♣

Un'altra applicazione della DFT che fa uso del prodotto di convoluzione riguarda il calcolo dei coefficienti del polinomio prodotto di due polinomi. Se si tiene conto della rappresentazione dei numeri nel sistema di numerazione posizionale (come quello decimale o binario) questa applicazione ha una interessante implicazione negli algoritmi numerici che fanno uso della multiprecisione [17].

♣ **Esempio 5.11.** [**Moltiplicazione di due polinomi**] Siano assegnati due polinomi:

$$p(x) = 1 + 5x + 17x^2, \qquad q(x) = 11 + 6x - 4x^2$$

Definiti i vettori $a, b \in \Re^5$:

$$a = (1, 5, 17, 0, 0) \quad \text{e} \quad b = (11, 6, -4, 0, 0)$$

che contengono nelle prime 3 posizioni i coefficienti di p e q rispettivamente, i coefficienti del polinomio z di quarto grado prodotto di p e di q si ottengono effettuando il prodotto di convoluzione dei vettori a e b, ovvero:

$$c = a * b = (11, 61, 213, 82, -68, 0, 0, 0, 0)$$

Il polinomio z è:

$$z(x) = p(x)q(x) = 11 + 61x + 213x^2 + 82x^3 - 68x^4$$

Per determinare i coefficienti di z utilizziamo l'operatore DFT. Posto:

$$A = DFT[a] = (23, -11.2082 - 14.7476i, 2.082 + 13.229i, 2.2082 - 13.229i, -11.2082 + 14.7476i)$$

$$B = DFT[b] = (13, 16.0902 - 3.3552i, 4.9098 - 7.3309i, 4.9098 + 7.3309i, +16.0902 + 3.3552i)$$

[17] [1]

utilizzando il **Teorema 5.2.2**:

$$C = DFT(c) = DFT[a * b] = A. * B =$$

$$= (299, -229.82 - 199.69i, 107.82 + 48.76i, 107.82 - 48.76i, -229.82 + 199.69i)$$

da cui:

$$c := IDFT(C) = (11, 61, 213, 82, -68)$$

Il calcolo dei coefficienti del prodotto di due polinomi si riduce, quindi, al calcolo di 3 DFT. ♣

Assegnati due polinomi

$$p(x) = \sum_{i=0}^{r} a_i x^i \ \in \Pi_r \,, \qquad q(x) = \sum_{j=0}^{s} b_j x^j \ \in \Pi_s$$

indicato con:

$$z(x) = p(x)q(x) = \sum_{k=0}^{r+s} c_k x^k \quad \in \Pi_{r+s} \tag{5.19}$$

il polinomio prodotto, i coefficienti di z:

$$\begin{cases} c_0 = a_0 \cdot b_0 \\ c_1 = a_0 b_1 + a_1 b_0 \\ \vdots \\ c_k = \sum_{h=0}^{k} a_h b_{k-h} \\ \vdots \end{cases}$$

si possono ottenere effettuando il prodotto di convoluzione tra il vettori a e b:

$$c = a * b$$

avendo indicato con $a = (a_0, a_1, ...a_{r+s})$ il vettore contenente nelle prime $r + 1$ posizioni i coefficienti del polinomio p e nelle restanti s posizioni lo zero e analogamente con $b = (b_0, b_1, ...b_{r+s})$ il vettore contenente nelle prime $s + 1$ posizioni i coefficienti del polinomio q e nelle restanti r posizioni lo zero, e con $c = (c_0, c_1, ...c_{r+s})$ il vettore contenente i coefficienti del polinomio prodotto z.

Applicando il **Teorema 5.2.2**, si puó ottenere il vettore c utilizzando la DFT. In particolare, per calcolare il vettore c bisogna:

1. calcolare la DFT dei coeffienti del polinomio p:

$$A = DFT[a]$$

2. calcolare la DFT dei coeffienti del polinomio q:

$$B = DFT[b]$$

3. effettuare il prodotto componente per componente di A e di B:

$$A. * B = C$$

4. calcolare i coefficienti di z mediante la DFT inversa:

$$c = IDFT[C]$$

5.3 La DFT come approssimazione della Trasformata di Fourier (FT)

Consideriamo il seguente integrale:

$$\int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt, \quad i = \sqrt{-1}, \tag{5.20}$$

dove $f : \Re \to \Re$ è una funzione per cui tale integrale esiste ed è finito [18]. Al variare di ω l'integrale in (5.20) definisce una funzione

$$F : \omega \to \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt$$

a cui si dà il nome di **Trasformata di Fourier (FT)** della funzione f. Sia $T \in \Re - \{\infty\}$, si ha:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt = \sum_{k=-\infty}^{+\infty} \int_{kT}^{(k+1)T} f(t)e^{-2\pi i\omega t}dt =$$

$$= \sum_{k=-\infty}^{+\infty} \int_{0}^{T} f(t+kT)e^{-2\pi i\omega(t+kT)}dt = \int_{0}^{T} \sum_{k=-\infty}^{+\infty} f(t+kT)e^{-2\pi i\omega(t+kT)}dt$$

Se introduciamo la funzione f_p, periodica di periodo T, definita come:

$$f_p(t) = \sum_{k=-\infty}^{+\infty} f(t+kT), \quad t \geq 0$$

segue che:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-2\pi i\omega t}dt = \int_{0}^{T} f_p(t)e^{-2\pi i\omega t}dt$$

[18] Ad esempio, tale è una funzione assolutamente integrabile su tutto l'asse reale che in ogni intervallo finito ha al più un numero finito di discontinuità di prima specie ed è a variazione limitata. Una siffatta funzione si dice che soddisfa le *condizioni di Dirichlet*.

ovvero la Trasformata di Fourier di f coincide con la trasformata di Fourier di f_p. Supponiamo di voler valutare la funzione $F(\omega)$ per $\omega = \omega_n = n/T$, $n = 0, 1, 2, \ldots$:

$$F(\omega_n) = \int_0^T f_p e^{-2\pi i \omega_n t} dt$$

e discretizziamo l'integrale con la formula di quadratura trapezoidale composita su N punti. Posto $h = \frac{T}{N}$ si ottiene:

$$F(\omega_n) = h \sum_{k=0}^{N-1} f_p(kh) e^{-2\pi i k h \frac{n}{T}} = h \sum_{k=0}^{N-1} f_p(kh) e^{-\frac{2\pi}{N} i k n}$$

Considerati i vettori $F = \{F(\omega_n)\}_{n=0,1,\ldots,N-1}$, e $f_p = \{f_p(kh)\}_{k=0,1,\ldots,N-1}$, segue che

$$F = DFT[f_p]$$

ovvero, la trasformata discreta di Fourier, costruita a partire dal vettore f_p, fornisce un'approssimazione dei valori assunti dalla Trasformata di Fourier della funzione f_p in un insieme di punti equidistanziati. In particolare poi, se f è nulla al di fuori dell'intervallo $[0, T]$, allora $f_p(v) = f(v)$, $v \in [0, T]$ e la $DFT[f_p](= DFT[f])$ **fornisce un'approssimazione della trasformata di Fourier di** f. In altre parole la FT e la DFT nei punti ω_n coincidono a meno di un errore. L'errore dipende **dall'errore di discretizzazione**, introdotto dalla formula di quadratura utilizzata, e dall'**errore di troncamento** introdotto dall'aver assunto $f_p = f$. Per quanto riguarda l'errore di discretizzazione, tale quantità tende a zero al tendere a zero di h. Come abbiamo già notato (vedi **Capitolo 2**, **§2.6)**, l'ordine di convergenza della formula di quadratura trapezoidale dipende dalla regolarità della funzione f. In particolare, a partire dalla stima dell'errore di discretizzazione delle formule di quadratura di Newton-Cotes, che garantisce che se la $f \in C^2([0, T])$ l'errore ha ordine 2, si possono ricavare stime per l'errore di discretizzazione che mostrano un'accuratezza di ordine sempre più alto quanto più la funzione è regolare. Ad esempio, se $f \in C^{2r+1}, r \geq 1$, utilizzando lo sviluppo dell'errore fornito dalla somma di Eulero McLaurin, l'errore di discretizzazione ha ordine $2r + 1$.

Teorema 5.3.1. *[6, 18]*
Fissato $r \geq 1$, sia $f \in C^{2r+1}([0, a])$, integrabile in $[0, +\infty]$ e sia:

$$M = \int_0^\infty |f^{(2r+1)}(x)| dx < \infty$$

Supponiamo inoltre che:

$$f^{(2k-1)}(0) = 0, \quad \lim_{x \to \infty} f^{(2k-1)}(x) = 0, \quad k = 1, \ldots, r$$

Si ha allora, per ogni fissato $h > 0$, per l'errore di discretizzazione E:

$$|E| \leq h^{2r+1} \frac{M \zeta(2r+1)}{2^{2r} \pi^{2r+1}}$$

dove ζ è la funzione zeta di Riemann.

In particolare, poi, se $f \in C^\infty$ e tutte le derivate di ordine dispari tendono a zero agli estremi dell'intervallo di integrazione, l'errore di discretizzazione tende a zero più velocemente di qualsiasi potenza di h. Infine, il teorema seguente riporta una maggiorazione dell'errore di discretizzazione di più semplice utilizzo negli algoritmi numerici, applicabile nel caso in cui la funzione f è la restrizione sull'asse reale di una funzione analitica:

Teorema 5.3.2. *[6, 18]*
Fissato $q \geq 0$. Sia G_q l'insieme delle funzioni g tali che $g(x+iy)$ sia analitica nella striscia $|y| \leq q$ e tali che

$$|g(x+iy)| \leq \frac{A}{|x|^\beta}, \quad |x| > \delta$$

con $A > 0$, $\delta > 0$ e $\beta > 1$. Se $f \in G_q$ e $\omega \in]0, \pi/h[$ allora:

$$E \leq \frac{2e^{-q\omega(p-1)}}{1-e^{-qp\omega}} M(q)$$

dove $p = \frac{2\pi}{h\omega} > 2$ e

$$M(q) = \max_{y=\pm q} \int_{-\infty}^{\infty} |f(t+iy)| dt$$

Per quanto riguarda l'errore di troncamento, ovvero l'errore introdotto dall'aver assunto che la funzione f sia nulla al di fuori dell'intervallo $[0, T]$, fissato h, tale quantità decresce esponenzialmente al crescere di N (e quindi di T) [19]. In generale, in base a tali considerazioni è possibile una scelta opportuna dei parametri N e h in modo da garantire dei risultati accettabili con un costo computazionale non elevato.

5.4 La Trasformata Veloce di Fourier (FFT)

Dalla definizione di DFT di un vettore di N numeri complessi si deduce che la valutazione diretta di ciascuna componente del vettore DFT richiede il calcolo di N moltiplicazioni complesse e $N-1$ addizioni complesse, ed inoltre la valutazione di N esponenziali complessi. *Supponendo noti gli esponenziali complessi*, il calcolo della DFT di un vettore di lunghezza N richiede, quindi:

$$T(N) = N \cdot [N \quad \text{molt. complesse} + (N-1) \quad \text{add. complesse}] = \mathcal{O}(N^2)$$

operazioni floating point. Dunque, a parte la valutazione degli esponenziali complessi, la complessità computazionale del calcolo di una DFT è equivalente a quella di un prodotto matrice-vettore.
Fino al 1965, anno di pubblicazione del primo algoritmo di *Fast Fourier Transform* (FFT) nonostante la già ampia diffusione della DFT, la sua applicazione presentava

[19] [4]

problemi legati al costo computazionale. L'idea di J. Cooley e J. Tukey, alla base degli algoritmi FFT, fu di *semplificare* opportunamente i calcoli decomponendo il problema, ovvero il calcolo di una DFT di lunghezza N, in sottoproblemi dello stesso tipo ma di dimensione non solo più piccola ma anche tale da consentire di eliminare una buona parte di operazioni inutili e ridondanti. Un algoritmo FFT implementa una metodologia di tipo *divide et impera*, applicandola più volte a ciascuno dei sottoproblemi così ottenuti, in modo da riorganizzare efficacemente le operazioni che coinvolgono gli esponenziali complessi.

Riguardo la valutazione degli esponenziali complessi, la difficoltà principale di un algoritmo per il calcolo di una DFT, e più in generale di tutti gli algoritmi numerici utilizzati nell'analisi di Fourier, è sempre stata la valutazione efficiente delle funzioni trigonometriche, indipendentemente dall'ambiente di calcolo a disposizione. Già nel 1925, Witthaker, avendo a disposizione le macchine calcolatrici per eseguire moltiplicazioni, oltre che le tavole trigonometriche, utilizzando la metodologia *divide et impera*, propose uno schema per il calcolo dei coefficienti di Fourier per $N = 6, 12, 24$, che faceva uso solo delle valutazioni negli angoli notevoli. Tale algoritmo, ad esempio per $N = 12$, impiegava solo 44 operazioni. Nel 1958, Goertzel propose un algoritmo basato su formule di ricorrenza per il calcolo di coefficienti di Fourier per un qualsiasi valore di N che utilizzava solo 2 valutazioni trigonometriche, successivamente reso stabile dalla modifica introdotta da Reinsch [20]. Chiaramente questo algoritmo, a differenza dell'algoritmo poposto da Witthaker, poteva essere eseguito per un *qualsiasi valore di* N, anche se la sua complessità computazionale ($\mathcal{O}(N^2)$) ne impediva l'implementazione effettiva per valori di N grandi. L'introduzione degli algoritmi FFT ha consentito lo sviluppo delle applicazioni dell'analisi di Fourier, tra le quali vi sono quelle caratteristiche delle discipline che prendono il nome di *digital signal processing* e di *digital image processing*. In tali applicazioni N è almeno dell'ordine di $\mathcal{O}(10^3)$.

A partire dall'idea introdotta da Cooley e Tukey, l'anno successivo Gentleman e Sande pubblicarono un articolo che descrive in maniera completa ed esaustiva l'algorimo FFT, corredandolo di analisi della complessità e della stabilità [21]. Oggi sono disponibili molteplici implementazioni dell'algoritmo FFT, ciascuna specializzata in base a vari fattori come il tipo di fattorizzazione di N e l'applicazione o meno del *bit reversal*. Il pacchetto software FFTW contiene tali implementazioni, utilizzabili in maniera del tutto *trasparente* perchè sono generate automaticamente da un compilatore interno alla libreria.

[20] [14]
[21] [12]

5.4.1 L'algoritmo di Cooley e Tukey

L'idea alla base di tutti gli algoritmi FFT è calcolare la DFT di un vettore di lunghezza $N = r \cdot q$ effettuando q DFT di lunghezza r o viceversa.

♣ **Esempio 5.12.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 3 \cdot 4 = 12$:

$$DFT[f]_k := F_k = \sum_{j=0}^{11} f_j w_{12}^{-jk} \tag{5.21}$$

posto:

$$\begin{array}{llll} k = & 3k_1 + k_0 & k_1 = 0,1,2,3 & k_0 = \quad 0,1,2 \\ j = & 4j_1 + j_0 & j_1 = 0,1,2 & j_0 = \quad 0,1,2,3 \end{array}$$

allora:

$$F(k) = \sum_{j_0=0}^{3} \sum_{j_1=0}^{2} f(4j_1 + j_0) w_{12}^{-(4j_1+j_0)(3k_1+k_0)} \tag{5.22}$$

Poiché:

$$w_{12}^{-(4j_1+j_0)(3k_1+k_0)} = w_{12}^{-12j_1k_1} w_{12}^{-4j_1k_0} w_{12}^{-3j_0k_1} w_{12}^{-j_0k_0} = w_{12}^{-12j_1k_1} w_3^{-j_1k_0} w_4^{-j_0k_1} w_{12}^{-j_0k_0}$$

e $w_{12}^{-12j_1k_1} = 1$, la (5.22) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{3} w_4^{-j_0k_1} \underbrace{\sum_{j_1=0}^{2} f(4j_1 + j_0) w_3^{-j_1k_0}}_{DFT\ di\ lunghezza\ 3} w_{12}^{-j_0k_0}}_{4\ DFT\ di\ lunghezza\ 3} \tag{5.23}$$

A parte il fattore esponenziale $w_{12}^{-j_0k_0}$, il calcolo di una DFT di lunghezza 12 si riconduce al calcolo di 4 DFT di lunghezza 3. In particolare, si osserva che i 4 vettori di cui si calcola la DFT, ottenuti da $f(4j_1 + j_0)$ fissando $j_0 = 0,1,2,3$ rispettivamente e facendo variare $j_1 = 0,1,2$, sono costruiti considerando le componenti del vettore f i cui indici distano di 4 unità. ♣

Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{N-1})$ di lunghezza $N = r_1 \cdot r_2$:

$$F(k) = \sum_{j=0}^{N-1} f_j w_N^{-jk} \tag{5.24}$$

Posto:

$$\begin{array}{llll} k = & r_1k_1 + k_0 & k_1 = 0,1,...,r_2 - 1 & k_0 = \quad 0,1,...,r_1 - 1 \\ j = & r_2j_1 + j_0 & j_1 = 0,1,...,r_1 - 1 & j_0 = \quad 0,1,...,r_2 - 1 \end{array}$$

allora:

$$F(k) = \sum_{j_0=0}^{r_2-1} \sum_{j_1=0}^{r_1-1} f(r_2 j_1 + j_0) w_N^{-(r_2 j_1 + j_0)(r_1 k_1 + k_0)} \tag{5.25}$$

Poiché:

$$w_N^{-(r_2 j_1 + j_0)(r_1 k_1 + k_0)} = w_N^{-N j_1 k_1} w_{12}^{-r_2 j_1 k_0} w_N^{-r_1 j_0 k_1} w_N^{-j_0 k_0} = w_N^{-N j_1 k_1} w_{r_1}^{-j_1 k_0} w_{r_2}^{-j_0 k_1} w_N^{-j_0 k_0}$$

e $w_N^{-N j_1 k_1} = 1$, la (5.25) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{r_2-1} w_{r_1}^{-j_0 k_1} \underbrace{\left[\sum_{j_1=0}^{r_1-1} f(r_2 j_1 + j_0) w_{r_2}^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\, r_1} w_N^{-j_0 k_0}}_{r_2\ DFT\ di\ lunghezza\ r_1} \tag{5.26}$$

A meno dell'esponenziale $w_N^{-j_0 k_0}$, il calcolo di una DFT di lunghezza $N = r_1 \cdot r_2$ si riconduce al calcolo di r_2 DFT di lunghezza r_1. Se poi r_1 si fattorizza nel prodotto $r_2 = r_3 \cdot r_4$, si può applicare tale idea alla DFT di lunghezza r_2, pervenendo via via a DFT di lunghezza più piccola.

♣ **Esempio 5.13.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_{11})$ di lunghezza $N = 4 \cdot 3 = 2^2 \cdot 3 = 12$:

$$DFT[f]_k := F(k) = \sum_{j=0}^{11} f_j w_{12}^{-jk} \tag{5.27}$$

posto:

$$\begin{array}{llllll} k = & 4k_1 + k_0 & k_1 = 0,1,2 & k_0 = & 0,1,2,3 \\ j = & 3j_1 + j_0 & j_1 = 0,1,2,3 & j_0 = & 0,1,2 \end{array}$$

si ha:

$$F(k) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{3} f(3j_1 + j_0) w_{12}^{-(4j_1 + j_0)(3k_1 + k_0)} \tag{5.28}$$

Poiché:

$$w_{12}^{-(3j_1 + j_0)(4k_1 + k_0)} = w_{12}^{-12 j_1 k_1} w_{12}^{-3 j_1 k_0} w_{12}^{-4 j_0 k_1} w_{12}^{-j_0 k_0} = w_{12}^{-12 j_1 k_1} w_4^{-j_1 k_0} w_3^{-j_0 k_1} w_{12}^{-j_0 k_0}$$

e $w_{12}^{-12 j_1 k_1} = 1$ allora la (5.28) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_0=0}^{2} w_3^{-j_1 k_1} \underbrace{\left[\sum_{j_1=0}^{3} f(3j_1 + j_0) w_4^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\, 4} w_{12}^{-j_0 k_0}}_{3\ DFT\ di\ lunghezza\ 4} \tag{5.29}$$

Il calcolo di una DFT di lunghezza 12 si riconduce al calcolo di 3 DFT di lunghezza 4. In questo caso gli $r_2 = 3$ vettori di lunghezza $r_1 = 4$ di cui si deve calcolare la DFT, ottenuti per $j = 0, 1, 2$ rispettivamente, sono costruiti considerando le componenti del vettore f che distano di $r_2 = 3$ unità.

Applichiamo lo stesso schema alla DFT più interna di lunghezza 4. Introduciamo il vettore F^* di lunghezza 3 così definito:

$$F^*(j_0) = \sum_{j_1=0}^{3} f(3j_1 + j_0) w_4^{-j_1 k_0} \quad j_1 = 0, 1, 2, 3 \tag{5.30}$$

Sia:

$$j_1 = \quad 2j_1' + j_0' \qquad j_1' = 0, 1, \quad j_0' = \quad 0, 1$$

allora:

$$F^*(j_0) = \underbrace{\sum_{j_1'=0}^{1} w_4^{-j_0' k_0} \underbrace{\left[\sum_{j_0'=0}^{1} f(3j_1 + j_0) w_2^{-j_1' k_0} \right]}_{DFT\ di\ lunghezza\ 2}}_{2\ DFT\ di\ lunghezza\ 2} \tag{5.31}$$

ovvero una DFT di lunghezza 4 è espressa come 2 DFT di lunghezza 2.
Dalle (5.29) e (5.31) segue quindi che:

$$F(k) = \underbrace{\sum_{j_0=0}^{2} w_3^{-j_1 k_1} \underbrace{\sum_{j_1'=0}^{1} w_2^{-j_0' k_0} \underbrace{\left[\sum_{j_0'=0}^{1} f(3j_1 + j_0) w_2^{-j_1' k_0} \right]}_{DFT\ di\ lunghezza\ 2}}_{2\ DFT\ di\ lunghezza\ 2} w_{12}^{-j_0 k_0}}_{3 \cdot 2 = 6\ DFT\ di\ lunghezza\ 2} \tag{5.32}$$

Si nota che la DFT più interna coinvolge coppie di componenti del vettore f che distano di 3 unità, come anche illustrato in Figura 5.4. In particolare, poiché

$$p_0 = 3j_1 + j_0 = 3(2j_1' + j_0') + j_0 = 6j_1' + 3j_0' + j_0$$

le 6 DFT vengono effettuate fissando nelle due sommatorie esterne $j_0 = 0$ e $j_1' = 0$ e quindi utilizzando le coppie di componenti del vettore f in corrispondenza di $p_0 = 0$ e $p_0 = 3$, poi, fissando $j_0 = 0$ e $j_1' = 1$ si ottiene $p_0 = 6$ e $p_0 = 9$, successivamente fissando $j_0 = 1$ e $j_1' = 0$, si ha $p_0 = 1$ e $p_0 = 4$, mentre per $j_0 = 1$ e $j_1' = 1$ si ha $p_0 = 7$ e $p_0 = 10$. Infine, per $j_0 = 2$ e $j_1' = 0$ segue $p_0 = 2$ e $p_0 = 5$ e per $j_0 = 2$ e $j_1' = 1$ si ha l'ultima coppia di componenti ovvero $p_0 = 8$ e $p_0 = 11$. ♣

Se assumiamo che N sia un multiplo di 2, scegliendo $r_1 = N/2$ e $r_2 = 2$, l'espressione (5.26) si scrive come:

$$F(k) = \underbrace{\sum_{j_0=0}^{1} w_{N/2}^{-j_0 k_1} \underbrace{\left[\sum_{j_1=0}^{N/2-1} f(2j_1 + j_0) w_{N/2}^{-j_1 k_0} \right]}_{DFT\ di\ lunghezza\ r_1 = N/2} w_N^{-j_0 k_0}}_{2\ DFT\ di\ lunghezza\ N/2} \tag{5.33}$$

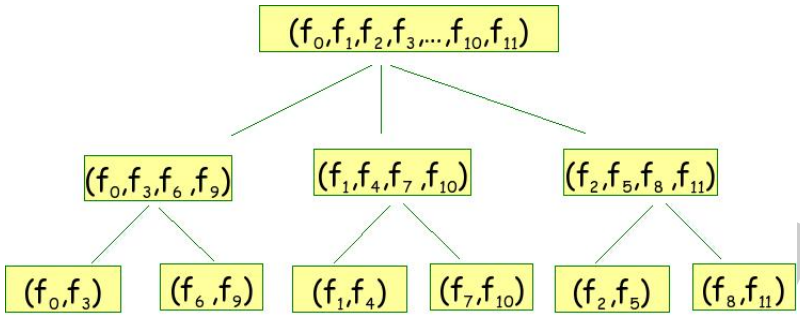

Figura 5.4: Al primo livello dell'albero: il calcolo della DFT di lunghezza 12 viene ricondotto a quello di 3 DFT di lunghezza 4. Al secondo livello dell'albero: il calcolo della DFT di lunghezza 12 viene ricondotto a quello di 6 DFT di lunghezza 2. La distanza fra le componenti è di $r_1 = 3$ unità.

e lo schema appena descritto porta alla decomposizione del vettore f in 2 vettori di lunghezza $N/2$.

Dalla (5.33) si osserva, in particolare, che fissato $j_0 = 0$ nella sommatoria esterna, l'algoritmo di Cooley e Tukey porta alla decomposizione del vettore f in una parte contenente le componenti con indice $2j_1$, con $j_1 = 0, \ldots, N/2$ e quindi le componenti con indice pari, e in un'altra parte, per $j_0 = 1$, contenente le componenti con indice dispari[22]. Siano quindi $\{f_{2j}\}_{j=0,\ldots,N/2}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$ i due vettori di lunghezza $N/2$ costruiti a partire dal vettore di componenti $\{f_j\}_{j=0,\ldots,N-1}$ e contenenti, rispettivamente, il primo le componenti di f corrispondenti a valori pari dell'indice j, il secondo le componenti corrispondenti a valori dispari dell'indice j.

♣ **Esempio 5.14.** Consideriamo il calcolo della DFT di un vettore di lunghezza $N = 16 = 2^4$.

$$DFT[f]_k := F(k) = \sum_{j=0}^{15} f(j) w_{16}^{-jk} \quad k = 0, \ldots, 15 \tag{5.34}$$

Fissato k, con $k = 0, \ldots, N-1$ si ha:

$$F(k) = \sum_{j=0}^{15} f(j) w_{16}^{-jk} = \sum_{j=0}^{7} f(2j) w_{16}^{-2jk} + \sum_{j=0}^{7} f(2j+1) w_{16}^{-(2j+1)k} =$$

$$= \sum_{j=0}^{7} f(2j) w_8^{-jk} + w_{16}^{-k} \sum_{j=0}^{7} f(2j+1) w_8^{-jk}$$

Osserviamo che, sfruttando la periodicità dell'esponenziale complesso coinvolto, ovvero dei termini ω_8^{-jk} e ω_8^{-k}, basta calcolare le componenti di $F(k)$ per $k = 0, \ldots, 7$, perché le altre 8 si possono ricavare dalle prime 8. Ad esempio, per $k = 9$ si ha:

$$\omega_8^{-j9} = \omega_8^{-j(8+1)} = \omega_8^{-j\cdot 8} \cdot \omega_8^{-j\cdot 1} = \omega_8^{-j\cdot 1} \quad (\omega_8^r = \omega_8^{r\, mod\, 8}).$$

[22] Questa particolare decomposizione fu evidenziata già nel 1942 da Danielson e Lanczos [5].

e quindi ci si riconduce al calcolo della componente con indice $k = 1$. Siano, allora,

$$F^E(k) = \sum_{j=0}^{7} f(2j) w_8^{-jk}, \quad k = 0, \ldots, 7$$

e

$$F^O(k) = \sum_{j=0}^{7} f(2j+1) w_8^{-jk}, \quad k = 0, \ldots, 7 \quad .$$

L'algoritmo di Cooley e Tukey, al primo passo, porta alla decomposizione del vettore F come:

$$F(k) = F^E(k) + \omega_{16}^{-jk} F^O(k), \quad k = 0, \ldots, 7$$

♣

Dalla (5.24) si ha:

$$F(k) = \sum_{j=0}^{N-1} f(j) \omega_N^{-jk} = \sum_{j=0}^{N/2-1} f(2j) \omega_N^{-2jk} + \sum_{j=0}^{N/2-1} f(2j+1) \omega_N^{-(2j+1)k}. \tag{5.35}$$

Poiché

$$\omega_N^{-(2j+1)k} = \omega_N^{-2jk} \cdot \omega_N^{-k} = \omega_{N/2}^{-jk} \cdot \omega_N^{-k}$$

segue

$$F(k) = \sum_{j=0}^{N/2-1} f(2j) \omega_{N/2}^{-jk} + w_N^{-k} \sum_{j=0}^{N/2-1} f(2j+1) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N-1$$

Siano

$$F^E(k) = \sum_{j=0}^{N/2-1} f(2j) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

e

$$F^O(k) = \sum_{j=0}^{N/2-1} f(2j+1) \omega_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

le due DFT di lunghezza $N/2$ ottenute a partire dai vettori $\{f_{2j}\}_{j=0,\ldots,N/2-1}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$. La (5.35) diventa quindi:

$$F(k) = F^E(k) + \omega_N^{-k} F^O(k) \quad k = 0, \ldots, N/2-1 \tag{5.36}$$

e sfruttando la periodicità dell'esponenziale complesso $\omega_{N/2}$, le altre $N/2$ componenti si ricavano a partire dalle prime $N/2$.

Se N è divisibile per 2^2, ovvero $r_1 = N/2$ è divisibile per 2, applicando la stessa idea alle due DFT, $\{F^E(k)\}_{k=0,\ldots,N/2-1}$ e $\{F^O(k)\}_{k=,\ldots,N/2-1}$, di lunghezza $N/2$, l'algoritmo

di Cooley e Tukey decompone ciascuna DFT in 2 DFT di lunghezza $N/4$, ottenute utilizzando le componenti con indice pari e quelle con indice dispari di ciascuno dei due vettori $\{f_{2j}\}_{j=0,\ldots,N/2-1}$ e $\{f_{2j+1}\}_{j=0,\ldots,N/2-1}$. Questo comporta che il calcolo del vettore F al primo passo viene ricondotto a quello di 4 DFT di lunghezza $N/4$.

♣ **Esempio 5.15. (Continua l'esempio 5.14)** Consideriamo i due vettori $\{F^E(k)\}_{k=0,\ldots,7}$ e $\{F^O(k)\}_{k=,\ldots,7}$ ottenute nell'esempio 5.14. Ciascuna DFT di lunghezza 8 si puó decomporre in 2 DFT di lunghezza 4. Ad esempio:

$$F^E(k) = \sum_{j=0}^{7} f(2j) w_8^{-jk} = \sum_{j=0}^{3} f(4j) w_8^{-2jk} + \sum_{j=0}^{3} f(4j+1) w_8^{-(2j+1)k} =$$

$$= \sum_{j=0}^{3} f(4j) w_4^{-jk} + w_8^{-k} \sum_{j=0}^{3} f(4j+1) w_4^{-jk}$$

Siano

$$F^{EE}(k) = \sum_{j=0}^{3} f(4j) w_4^{-jk} \qquad \text{e} \qquad F^{EO}(k) = \sum_{j=0}^{3} f(4j+1) w_4^{-jk}$$

le 2 DFT di lunghezza 4 costruite utilizzando i vettori $\{f_{4j}\}_{j=0,\ldots,3}$ e $\{f_{4j+1}\}_{j=0,\ldots,3}$. Al secondo passo, l'algoritmo di Cooley e Tukey porta alla decomposizione di $\{F^E(k)\}_{k=0,\ldots,N/2-1}$ come:

$$F^E(k) = F^{EE}(k) + w_8^{-k} F^{EO}(k) \quad k = 0, \ldots, 3$$

e quindi il calcolo del vettore F viene ricondotto al calcolo di 4 DFT di lunghezza 4. ♣

In generale, assumendo che N sia esprimibile come una potenza di 2, ad esempio $N = 2^m$, al passo k il calcolo della DFT di lunghezza N viene ricondotto a quello di k DFT di lunghezza 2^{m-k}, ovvero il vettore f viene decomposto in k vettori di lunghezza 2^{m-k}. Dopo $m-1$ passi, l'algoritmo di Cooley e Tukey decompone il vettore f in 2^{m-1} coppie di componenti, laddove ciascuna coppia definisce una DFT di lunghezza 2. L'algoritmo così ottenuto è la variante FFT radix-2 di Cooley e Tukey.

♣ **Esempio 5.16. (Continua l'esempio 5.15)** Riscrivendo ciascuna DFT di lunghezza 4 (ottenuta nell'esempio 5.15) come somma di 2 DFT di lunghezza 2, si ha:

$$F^{EE}(k) = \sum_{j=0}^{3} f(j) w_4^{-jk} = \sum_{j=0}^{1} f(8j) w_4^{-2jk} + \sum_{j=0}^{1} f(8j+1) w_4^{-(2j+1)k} =$$

$$= \sum_{j=0}^{1} f(8j) w_2^{-jk} + w_4^{k} \sum_{j=0}^{1} f(8j+1) w_2^{-jk}$$

Siano

$$F^{EEE}(k) = \sum_{j=0}^{1} f(8j) w_2^{-jk} \qquad \text{e} \qquad F^{EEO}(k) = \sum_{j=0}^{1} f(8j+1) w_2^{-jk}$$

le 2 DFT di lunghezza 2 ottenute condiderando rispettivamente le componenti (f_0, f_8) e (f_1, f_9). In conclusione, per $N = 16 = 2^4$, dopo 3 passi l'algoritmo di Cooley e Tukey porta alla decomposizione seguente:

$$F_k = F_k^E + F_k^O =$$
$$= (F_k^{EE} + F_k^{EO}) + (F_k^{OE} + F_k^{OO}) =$$
$$= (F_k^{EEE} + F_k^{EEO}) + (F_k^{EOE} + F_k^{EOO}) + (F_k^{OEE} + F_k^{OEO}) + (F_k^{OOE} + F_k^{OOO})$$

laddove la prima somma coinvolge 2 DFT di lunghezza 8, la seconda somma coinvolge 4 DFT di lunghezza 4 e l'ultima 8 DFT di lunghezza 2. Questo è lo schema dell'algoritmo FFT radix-2, introdotto da Cooley e Tukey. Si nota la natura ricorsiva dell'algoritmo di Cooley e Tukey. ♣

L'algoritmo di Cooley e Tukey decompone ripetutamente il vettore f, di lunghezza $N = 2^m$, separando le componenti con indice pari da quelle con indice dispari fino ad arrivare a 2^{m-1} coppie di componenti. Dopo questa prima fase di *decomposizione* si passa alla fase di *calcolo*: partendo dalla DFT più piccola, ovvero quella di lunghezza 2 si calcolano DFT di lunghezza 4 combinando 2 DFT di lunghezza 2, poi, combinando 2 DFT di lunghezza 4, si calcolano DFT di lunghezza 8 e così proseguendo, fino a ottenere 1 DFT di lunghezza N.

Questa versione dell'algoritmo prevede da un punto di vista implementativo una fase di *riordinamento* iniziale e una seconda fase di *calcolo* sul vettore ordinato. In tal modo, questa versione produce il vettore risultante secondo l'ordine naturale.

In maniera analoga derivano gli algoritmi FFT radix-r. I più comuni sono quelli per $r = 3$ e per $r = 5$.

♣ **Esempio 5.17.** Consideriamo la DFT di un vettore $f = (f_0, f_1, f_2, ...f_8)$ di lunghezza $N = 3 \cdot 3 = 9$:

$$DFT[f]_k := F_k = \sum_{j=0}^{8} f_j w_9^{jk} \tag{5.37}$$

Posto:

$$\begin{array}{llll} k = & 3k_1 + k_0 & k_1, k_0 = & 0, 1, 2 \\ j = & 3j_0 + j_1 & j_1, j_0 = & 0, 1, 2 \end{array}$$

si ha

$$F(k) = \sum_{j_0=0}^{2} \sum_{j_1=0}^{2} f(3j_0 + j_1) w_9^{-(3j_0+j_1)(3k_1+k_0)} \tag{5.38}$$

Poiché:

$$w_9^{-(3j_0+j_1)(3k_1+k_0)} = w_9^{-9j_0k_1} w_9^{-3j_0k_0} w_9^{-3j_1k_1} w_9^{-j_1k_0}$$

e $w_9^{-9j_0k_1} = 1$ la (5.38) si può scrivere come:

$$F(k) = \underbrace{\sum_{j_1=0}^{2} w_9^{3j_1k_1} \underbrace{\left[\sum_{j_0=0}^{2} f(3j_0+j_1) w_9^{-3j_0k_0}\right]}_{DFT\ di\ lunghezza\ 3} w_9^{-j_1k_0}}_{3\ DFT\ di\ lunghezza\ 3} \tag{5.39}$$

Il calcolo di una DFT di lunghezza 9 si riconduce al calcolo 3 DFT di lunghezza 3. ♣

La strategia descritta nell'esempio precedente è alla base della classe di algoritmi FFT *radix-3* in cui il vettore f ha lunghezza $N = 3^m$.

5.4.2 L'algoritmo di Gentleman e Sande

Una strategia leggermente diversa da quella alla base dello schema di Cooley e Tukey sottende la versione dell'algoritmo FFT radix-2 introdotta da Gentleman e Sande. Tale variante si basa sul decomporre ripetutamente il vettore DFT da calcolare. Si osserva che le componenti con indice pari della DFT si possono esprimere come una DFT di lunghezza $N/2$ costruita a partire da un vettore ottenuto combinando le componenti del vettore di input che distano di $N/2$. Analogo ragionamento si fa per le componenti con indice dispari della DFT da calcolare.

♣ **Esempio 5.18.** Consideriamo il calcolo della DFT di un vettore di lunghezza $N = 8 = 2^3$.

$$F(k) = \sum_{j=0}^{8} f_j w_8^{-jk} \tag{5.40}$$

Da:

$$F(k) = \sum_{j=0}^{8} f_j w_8^{-jk} = \sum_{j=0}^{3} f_j w_8^{-jk} + \sum_{j=0}^{3} f_{j+4} w_8^{-(j+4)k} \tag{5.41}$$

separiamo le componenti del vettore F con indice pari da quelle con indice dispari, ovvero consideriamo i due vettori:

$$(F_0, F_2, F_4, F_6) \quad e \quad (F_1, F_3, F_5, F_7)\,.$$

Sfruttando la proprietà di periodicità dell'esponenziale complesso, si ha:

$$F(2k) = \sum_{j=0}^{3} [f(j) + f(j+4)] w_4^{-jk} \quad , k = 0, \ldots, 3$$

e

$$F(2k+1) = \sum_{j=0}^{3} [f(j) - f(j+4)] w_4^{-j} w_4^{-jk} \quad , k = 0, \ldots, 3$$

Primo passo: L'espressione di $F(2k)$ e di $F(2k+1)$ suggerisce di costruire i 2 vettori di lunghezza 4:

$$y = (f_0 + f_4, f_1 + f_5, f_2 + f_6, f_3 + f_7)$$

e

$$z = (f_0 - f_4, (f_1 - f_5)w_4^{-1}, (f_2 - f_6)w_4^{-2}, (f_3 - f_7)w_4^{-3})$$

Per cui, i due vettori $\{F(2k)\}_{k=0,1,2,3}$ e $\{F(2k+1)\}_{k=0,1,2,3}$ si ottengono come DFT di lunghezza 4 costruite a partire dai vettori y e z:

$$F(2k) = \sum_{j=0}^{3}[f(j) + f(j+4)]w_4^{-jk} = \sum_{j=0}^{3} y(j)w_4^{-jk} \quad k = 0, \ldots, 3$$

e

$$F(2k+1) = \sum_{j=0}^{3} z(j)w_4^{-jk} \quad k = 0, \ldots, 3$$

Per ciascuno dei due vettori $\{F(2k)\}_{k=0,1,2,3}$ e $\{F(2k+1)\}_{k=0,1,2,3}$, separiamo il calcolo delle componenti con indice pari da quello delle componenti con indice dispari. Ad esempio, per quanto riguarda $\{F(2k)\}_{k=0,1,2,3}$, consideriamo

$$(F(0), F(4)) = \{F(2k)\}_{k=0,2}$$

e

$$(F(2), F(6)) = \{F(2k)\}_{k=1,3}$$

La coppia $(F(0), F(4))$ si può esprimere come DFT di lunghezza 2 costruita a partire dal vettore di componenti $(y_0 + y_2, y_1 + y_3)$. Cioé:

$$\{F(2k)\}_{k=0,2} = \sum_{j=0}^{1}[y(j) + y(j+2)]w_2^{-jk}$$

$$\{F(2k)\}_{k=1,3} = \sum_{j=0}^{1}[y(j) + y(j+2)]w_2^{-jk}w_4^{-j}$$

Analogamente, la coppia $(F(2), F(6))$ si puó esprimere come una DFT di lunghezza 2 costruita a partire dal vettore di componenti $(y_0 - y_2, (y_1 - y_3)w_4^{-1})$.
Secondo passo: questa analisi suggerisce di costruire i due vettori

$$y' = (y_0 + y_2, y_1 + y_3), \quad y'' = (y_0 - y_2, (y_1 - y_3)\omega_4^{-1})$$

per cui il vettore di componenti $\{F(2k)\}_{k=0,1,2,3}$ si puó ottenere calcolando 2 DFT di lunghezza 2 costruite a partire dai vettori y' e y''. Cioè:

$$F(2k)_{k=0,2} = \sum_{j=0}^{2} y'(j)w_2^{-jk}, \qquad F(2k+1)_{k=0,2} = \sum_{j=0}^{2} y''(j)w_2^{-jk}.$$

Analogo ragionamento vale per il vettore di componenti $\{F(2k+1)\}_{k=0,1,2,3}$. Separando le componenti con indice pari da quelle con indice dispari, cioè considerando i due sottovettori: $(F(1), F(5))$ e $(F(3), F(7))$, si trova che

$$(F(1), F(5)) = \{F(2k+1)\}_{k=0,2} = \sum_{j=0}^{1}[z(j) + z(j+2)]w_2^{-jk} = \sum_{j=0}^{1} z'_j w_2^{-jk} \quad k = 0, 2$$

$$(F(3), F(7)) = \{F(2k+1)\}_{k=1,3} = \sum_{j=0}^{1}[z(j) - z(j+2)]w_2^{-jk}w_4^{-j} = \sum_{j=0}^{1} z''(j)w_2^{-jk} \quad k = 1, 3$$

Si costruiscono quindi i vettori:

$$z' = (z_0 + z_2, z_1 + z_3), \quad z'' = (z_0 - z_2, z_1 - z_3)$$

per cui ciascuna coppia di componenti si può esprimere come DFT di lunghezza 2 costruita rispettivamente su z' e z''.

Il terzo passo comporta infine il calcolo delle DFT di lunghezza 2 costruite al secondo passo, e quindi il calcolo delle quantità:

$$y''' = y'_0 + y'_1, \quad y^{iv} = y'_0 - y'_1$$
$$y^{v} = y''_1 + y''_2 \quad y^{vi} = y''_1 - y''_2$$
$$z''' = z'_0 + z'_1, \quad z^{iv} = z'_0 - z'_1$$
$$z^{v} = z''_1 + z''_2 \quad z^{vi} = z''_1 - z''_2$$

Il vettore $F = DFT[f]$ è:

$$F = (y''', y^{iv}, y^{v}, y^{vi}, z''', z^{iv}, z^{v}, z^{vi}) = (F_0, F_4, F_2, F_6, F_1, F_5, F_3, F_7)$$

Come si può osservare, il vettore F ha le componenti non ordinate secondo l'ordine naturale. La componente F_4 si trova al posto della componente F_1 e viceversa, la componente F_6 si trova al posto della componente F_3 e viceversa. L'ordinamento con il quale si presentano le componenti del vettore DFT è detto **ordinamento del bit inverso** (*bit reversal*). ♣

Applicando la proprietà di periodicità dell'esponenziale complesso si ha:

$$F(k) = \sum_{j=0}^{N-1} f(j)w_N^{-jk} = \sum_{j=0}^{N/2-1} f(j)w_N^{-jk} + \sum_{j=0}^{N/2-1} f(j + N/2)w_N^{-(j+N/2)k}$$

da cui:

$$\begin{aligned} F(k) &= \sum_{j=0}^{N/2-1} [f(j)w_N^{-jk} + f(j + N/2)w_N^{(-j+N/2)k}] = \\ &= \sum_{j=0}^{N/2-1} [f(j) + (-1)^k f(j + N/2)]w_{N/2}^{-jk/2} \end{aligned}$$

l'ultima eguaglianza sussiste in quanto $w_N^{-N/2} = -1$ e $w_N^{-jk} = w_N^{-jk/2}$.
Consideriamo le componenti con indice pari:

$$F(2k) = \sum_{j=0}^{N/2-1} [f(j) + f(j + N/2)w_{N/2}^{-jk}], \quad k = 0, \ldots, N/2 - 1$$

e quelle con indice dispari

$$F(2k+1) = \sum_{j=0}^{N/2-1} [(f(j) - f(j+N/2))w_N^{-j}]w_{N/2}^{-jk}, \quad k = 0, \ldots, N/2-1$$

Introdotti i vettori $\{y_j\}_{j=0,\ldots,N/2-1}$ e $\{z_j\}_{j=0,\ldots,N/2-1}$ di componenti

$$y_j = f_j + f_{j+N/2}, \quad z_j = (f_j - f_{j+N/2})w_N^{-j}, \quad j = 0, \ldots, N/2-1$$

i due vettori di lunghezza $N/2$, $\{F_{2k}\}_{k=0,\ldots,N/2}$ e $\{F_{2k+1}\}_{k=0,\ldots,N/2}$, si possono riguardare come DFT di lunghezza $N/2$ costruite rispettivamente su $\{y_j\}_{j=0,\ldots,N/2-1}$ e $\{z_j\}_{j=0,\ldots,N/2-1}$:

$$F = DFT[f] = (\{F_{2k}\}, \{F_{2k+1}\})_{k=0,\ldots,N/2} = (DFT[y], DFT[z])$$

Al **primo passo**, l'algoritmo di Gentleman e Sande calcola i due vettori y e z.

Consideriamo ciascuna DFT e separiamo le componenti con indice pari da quelle con indice dispari. I due vettori così costruiti, di lunghezza $N/4$, sono esprimibili in termini di DFT di lunghezza $N/4$:

$$DFT[y] = \sum_{j=0}^{N/2-1} [f(j) + f(j+N/2)w_{N/2}^{-jk}] = \sum_{j=0}^{N/2-1} y(j)w_{N/2}^{-jk} =$$

$$= \sum_{j=0}^{N/4-1} (y(j) + y(j+N/4))w_{N/4}^{-jk} + \sum_{j=0}^{N/4-1} [(y(j) - y(j+N/4))\omega_{N/2}^{-j}]w_{N/4}^{-jk}$$

Se consideriamo le componenti con indice pari del vettore $DFT[y]$:

$$DFT[y](2k) = \sum_{j=0}^{N/4-1} (y(j) + y(j+N/4))\omega_{N/4}^{-jk} \quad k = 0, \ldots, N/4-1$$

e

$$DFT[y](2k+1) = \sum_{j=0}^{N/4-1} (y(j) - y(j+N/4))\omega_{N/2}^{-j}\omega_{N/4}^{-jk} \quad k = 0, \ldots, N/4-1$$

introdotti i vettori di lunghezza $N/4$, $\{y'\}_{j=0,\ldots,N/4-1}$ e $\{y''\}_{j=0,\ldots,N/4-1}$, di componenti

$$y'_j = (y_j + y_{j+N/4}), \quad y''_j = (y_j - y_{j+N/4})w_{N/2}^{-j},$$

segue che:

$$\{DFT[y]\} = \{DFT[y'], DFT[y'']\},$$

Analogamente, per il vettore $DFT[z]$ contenente le componenti con indice dispari della DFT di f:

$$DFT[z](k) = \sum_{j=0}^{N/2-1} [(f(j) - f(j+N/4))w_{N/2}^{-j}]w_{N/4}^{-jk} = \sum_{0}^{N/2-1} z(j)w_{N/2}^{-jk}$$

$$= \sum_{j=0}^{N/4-1} [(z(j)+z(j+N/4))w_{N/2}^{j}]w_{N/4}^{jk} + \sum_{j=0}^{N/4-1} [(z(j)-z(j+N/4))w_{N/2}^{-j}]w_{N/4}^{-jk}, \quad k = 0, N/4-1$$

Introdotti i vettori $\{z'\}_{j=0,l...,N/4-1}$ e $\{z''\}_{j=0,l...,N/4-1}$ di componenti

$$z'_j = z_j + z_{j+N/4}, \quad z''_j = (z_j - z_{j+N/4})w_{N/2}^{-j},$$

segue che

$$DFT[z] = \{DFT[z'], DFT[z'']\}$$

Al **secondo passo**, l'algoritmo di Gentleman e Sande calcola i vettori y' , y'', z' e z'' e il calcolo della DFT di lunghezza N viene ricondotto a quello di 4 DFT di lunghezza $N/4$:

$$F = DFT[f] = (DFT[y], DFT[z]) = (DFT[y'], DFT[y''], DFT[z'], DFT[z''])$$

Proseguendo in questo modo, dopo $m = \log_2(N)$ passi, l'algoritmo di Gentleman e Sande costruisce $N/2$ vettori di lunghezza 2. Ciascuna coppia è una DFT e fornisce una coppia di componenti della DFT del vettore f.

A differenza della variante di Cooley e Tukey, si nota che in questa strategia le DFT calcolate in un certo passo sono ottenute *utilizzando il vettore costruito al passo precedente*, per cui l'algoritmo di Gentleman e Sande prevede inizialmente una fase di calcolo, per un certo numero di passi dipendente dalla dimensione del vettore f, e poi, una volta calcolate le $N/2$ DFT di lunghezza 2, è necessaria una fase di riordinamento del vettore DFT, dal momento che tale vettore risulta nell'ordine *scrambled*, ovvero secondo l'ordinamento indotto dal *bit reversal*.
Questa variante è anche nota in letteratura come *decimazione nelle frequenze*, perché la decomposizione viene effettuata sulle componenti del vettore DFT , e nell'analisi armonica tale vettore fornisce informazioni sulle frequenze contenute nel vettore dato [23].

*L'**ordinamento del bit inverso** (bit reversal) è definito nel modo seguente: due interi positivi, j e k, sono in* relazione *secondo l'ordinamento del bit inverso se, considerata la rappresentazione binaria di j e invertendo l'ordine delle cifre binarie, si ottiene il numero intero decimale k. Una coppia di numeri naturali i_1, i_2 si dice* ordinata *secondo il bit inverso se i numeri interi naturali k_1, k_2, in relazione secondo l'ordinamento del bit inverso rispettivamente con i_1 e i_2, sono ordinati secondo l'ordinamento naturale. Un insieme finito di numeri interi risulta ordinato secondo il bit inverso, se una qualsiasi coppia di elementi di tale insieme risulta ordinata secondo il bit inverso. Un vettore, $v \in \Re^N$, è ordinato secondo l'ordinamento del bit inverso, se tale risulta il sottoinsieme dei numeri naturali costituito dai valori assunti dall'indice della generica componente del vettore. In altre parole, considerata la componente di indice j, questa si trova al posto*

[23] [12]

della componente di indice k dove k è l'indice in relazione con j secondo l'ordinamento del bit inverso.

♣ **Esempio 5.19.** Consideriamo 4 numeri interi ordinati (in senso crescente):

$$0, 1, 2, 3.$$

Consideriamo la rappresentazione binaria di ciascuno:

$$\begin{aligned}(0)_{base10} &= (00)_{base2}\\ (1)_{base10} &= (01)_{base2}\\ (2)_{base10} &= (10)_{base2}\\ (3)_{base10} &= (11)_{base2}\end{aligned}$$

e invertiamola. Si ottiene:

$$\begin{aligned}(00)_{base2} &= (0)_{base10}\\ (10)_{base2} &= (2)_{base10}\\ (01)_{base2} &= (1)_{base10}\\ (11)_{base2} &= (3)_{base10}\end{aligned}$$

I numeri interi

$$0\ ,\ 2\ ,\ 1\ ,\ 3$$

sono ordinati in senso crescente secondo la relazione del bit inverso. ♣

5.4.3 Complessità computazionale degli algoritmi FFT

In generale, quindi, gli algoritmi FFT effettuano il calcolo di una DFT di lunghezza N combinando opportunamente DFT di lunghezza inferiore. In base alla fattorizzazione del parametro N si distinguono essenzialmente tre tipologie di algoritmi FFT.

1. Se $N = r_1 r_2$, e r_1 e r_2 sono primi tra loro, gli algoritmi FFT calcolano r_1 DFT di lunghezza r_2 (*algoritmi FFT con fattori primi (Prime Factor Algorithms (PFA))*). Di questi il più noto è l'algoritmo di Rader [24]. La complessità di questi algoritmi è
$$T(N) = \mathcal{O}(N(r_1 + r_2))$$

2. Se $N = r_1^p r_2^q$ gli algoritmi FFT calcolano p DFT di lunghezza r_1 e q DFT di lunghezza r_2 (algoritmi a radici miste *(FFT mixed radix)*. Tipicamente, la radice in questo caso è un numero primo sufficientemente piccolo, ad esempio $r = 3, 5$. La complessità di questa classe di algoritmi è:
$$T(N) = \mathcal{O}(p \log_{r_1} N + q \log_{r_2} N)$$

[24] [24]

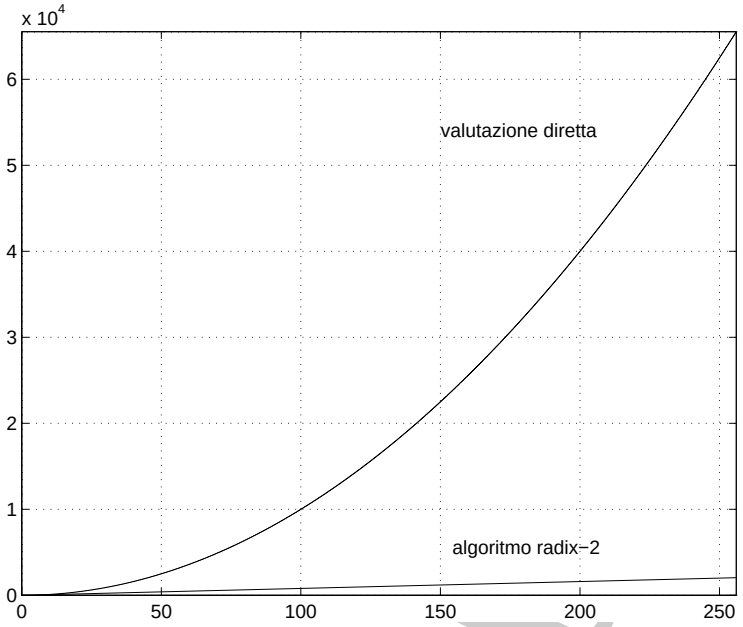

Figura 5.5: Confronto tra la complessità computazionale della valutazione diretta della DFT di una sequenza di numeri complessi di lunghezza N e la FFT della medesima sequenza.

3. Se $N = r^p$ gli algoritmi FFT calcolano p DFT di lunghezza r (algoritmi *radix-r*). In particolare, tra gli algoritmi radix-r, i più efficienti sono quelli del tipo radix-2 a cui appartengono anche quelli in cui r è una potenza di 2 come gli algoritmi radix-4 e radix-8. Ciò deriva dal fatto che in questi casi le funzioni trigonometriche assumono valori uguali a ± 1 e 0. La complessità di tempo $T(N)$ di questi algoritmi è:

$$T_{DFT}(N) = \mathcal{O}(N \log_2 N)$$

Nel grafico in Figura 5.5 sono confrontate la complessità computazionale della valutazione diretta della DFT e quella dell'algoritmo FFT radix-2. Si osserva come la differenza tra le due funzioni, rispettivamente $y = N^2$ e $y = N \log_2 N$, al crescere di N aumenta in maniera molto significativa; ad esempio per $N = 10^8$, risulta $N^2 = 10^{16}$ e $N \log_2 N \simeq 10^8 \cdot 26$ il che significa che, utilizzando un calcolatore con una potenza di calcolo di $1Glop/s = 10^9 flop/s$, nel primo caso occorrono $10^{16} \cdot 10^{-9} sec = 10^7 sec$ corrispondenti a circa 3 mesi, mentre nel secondo caso occorrono appena $26 \cdot 10^8 \cdot 10^{-9} sec = 2.6 sec$

Esiste una stretta relazione tra l'algoritmo radix-2 e gli algoritmi radix-r, infatti la complessità di tali algoritmi è data da:

$$T(N) = \mathcal{O}(Nr log_r N) = \mathcal{O}\left(Nr \frac{log_2 N}{log_2 r}\right) = \mathcal{O}\left(\frac{r}{log_2 r} \cdot N log_2 N\right)$$

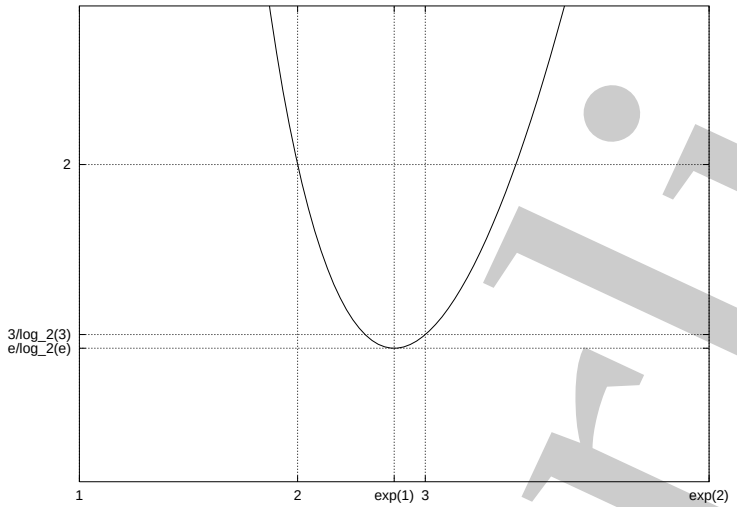

Figura 5.6: Rappresentazione del fattore di proporzionalità $\frac{r}{log_2 r}$.

Il fattore di proporzionalità $r/log_2 r$, come anche mostrato in Figura 5.6, assume il suo valore minimo per $r = e$; infatti:

$$\frac{d}{dr}\frac{r}{log_2 r} = \frac{log_2 r - log_2 e}{(log_2 r)^2} \geq 0 \Longleftrightarrow r \geq e$$

e ha in $r = 3$ un valore minore di quello che assume per $r = 2$:

$$\frac{3}{log_2 3} \simeq 1.893, \quad \frac{2}{log_2 2} = 2$$

quindi l'algoritmo FFT radix-3 è quello che ha la complessità computazionale minima; nonostante ciò l'algoritmo radix-2 è di gran lunga il più utilizzato, sia perché la maggior parte dei calcolatori ha un sistema aritmetico floating point in base 2, sia perché con questa scelta alcune potenze w_N^k sono semplificate (abbiamo visto che il numero complessivo di esponenziali da calcolare è uguale alla metà della lunghezza della sequenza di cui si vuole la FFT).

♣ **Esempio 5.20.** Riprendiamo l'esempio 5.10. Abbiamo visto che la risoluzione del sistema lineare

$$Mu = b$$

attraverso l'uso della DFT comporta il calcolo di $4n$ DFT di lunghezza $2N$. Se $N = 10^6$, questo significa effettuare

$$4n(2N \log_2(2N)) = 4 \cdot 10^3 \cdot 2 \cdot 10^6 \log_2(2 \cdot 10^6) = \mathcal{O}(10^9) \quad flop \tag{5.42}$$

La risoluzione diretta del sistema lineare con l'algoritmo di Cholesky richiede invece

$$T_{chol} = (1/3)N^3 = (1/3) \cdot 10^{18} \quad flop$$

mentre, se si utilizza l'algoritmo specializzato per matrici tridiagonali a blocchi, è necessaria una complessità dell'ordine

$$T_{bandchol} = \mathcal{O}(2N^2) = 2 \cdot 10^{12} \quad flop$$

Infine, dalla (4.42) del **Capitolo 4**, **§4.6**, la risoluzione del sistema mediante un metodo iterativo, in un numero k di iterazioni, richiede una complessità di tempo asintotica:

$$T(n) = k \cdot T_{it}(N) = \mathcal{O}(kN^2(1 - SP)) = \mathcal{O}(k \cdot 10^{12} \cdot (1 - SP)) \quad flop$$

Quest'ultima risulta essere maggiore della (5.42) se il numero di iterazioni k soddisfa la condizione:

$$k(1 - SP) > 10^{-3}$$

Dal confronto tra le complessità si deduce che, utilizzando un calcolatore con una prestazione di 1 Gflop/s, si ottiene una riduzione da 10 anni ($\frac{1}{3}10^{18} \cdot 10^{-9} = \frac{1}{3}10^9$ sec $\simeq$ 10 anni) con l'algoritmo di Cholesky a mezz'ora ($2 \cdot 10^{12} \cdot 10^{-9} = 2 \cdot 10^3$ sec $\simeq$ 30 min) con la versione a banda ed infine a 3 minuti ($8 \cdot 10^9 \cdot \log_2(2 \cdot 10^6) \cdot 10^{-9}$sec $\simeq$ 3 min) utilizzando la FFT.

♣

5.4.4 Aspetti implementativi dell'algoritmo FFT radix-2

Consideriamo l'algoritmo FFT radix-2 che si applica nel caso in cui $N = 2^m$. Come abbiamo visto, sia l'algoritmo di Cooley e Tukey sia la versione proposta da Gentleman e Sande hanno come operazione di base il calcolo di una DFT di lunghezza 2.

♣ Esempio 5.21. DFT di lunghezza 2

Sia $N = 2^1$, si vuole calcolare la DFT del vettore $\underline{f} = (f_0, f_1)$. Per definizione di DFT si ha:

$$F_k = \sum_{j=0}^{1} f_j \omega_2^{-jk} \quad k = 0, 1$$

ovvero:

$$k = 0 \quad \rightarrow \quad F_0 = f_0\omega_2^0 + f_1\omega_2^0 \tag{5.43}$$

$$k = 1 \quad \rightarrow \quad F_1 = f_0\omega_2^0 + f_1\omega_2^{-1} \tag{5.44}$$

il calcolo di F_0 ed F_1 richiede, quindi, 4 moltiplicazioni complesse. D'altra parte osservando che gli esponenziali complessi coinvolti sono:

$$\omega_2^0 = e^{-i\frac{2\pi}{2}0} = cos\left(\frac{2\pi}{2}0\right) - i\,sen\left(\frac{2\pi}{2}0\right) = 1$$
$$\omega_2^{-1} = e^{-i\frac{2\pi}{2}1} = cos\left(\frac{2\pi}{2}1\right) - i\,sen\left(\frac{2\pi}{2}0\right) = -1$$

la (5.43) e la (5.44) si riducono ad una somma ed una sottrazione tra numeri complessi:

$$\begin{array}{lll} k = 0 & \rightarrow & F_0 = f_0 + f_1 \\ k = 1 & \rightarrow & F_1 = f_0 - f_1 \end{array} \tag{5.45}$$

♣

Il calcolo di una DFT di lunghezza 2 richede 2 operazioni (somma e differenza) alle quali si associa lo schema grafico (mostrato in Figura 5.7), detto *schema a farfalla (o butterfly)*.

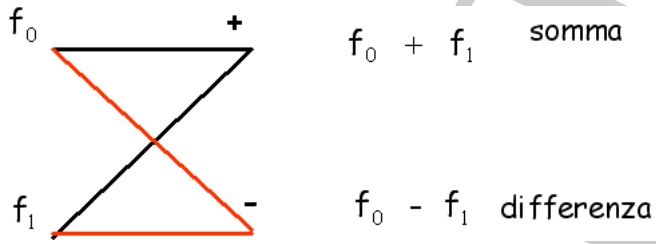

Figura 5.7: Schema butterfly.

Più in generale, gli algoritmi FFT si basano su un'operazione del tipo:

$$\begin{aligned} k = 0 \rightarrow \quad & F_0 = f_0 + f_1 \\ k = 1 \rightarrow \quad & F_1 = w(f_0 - f_1) \end{aligned}$$

dove w è un opportuno esponenziale complesso, come mostrato in Figura 5.8.

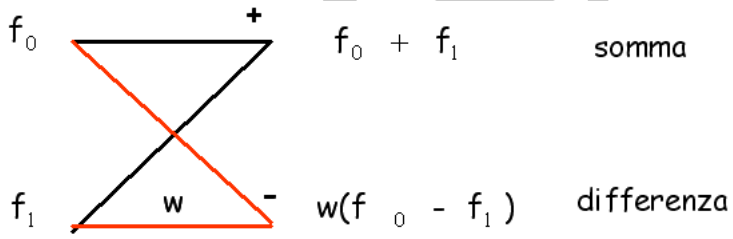

Figura 5.8: Schema generale di una butterfly.

Da un punto di vista implementativo tutti gli algoritmi FFT radix-2 ad ogni passo si compongono di operazioni di tipo *butterfly*.

♣ **Esempio 5.22. DFT di lunghezza 4**
Sia $N = 2^2 = 4$, si vuole calcolare la DFT del vettore:

$$\underline{f} = (f_0, f_1, f_2, f_3).$$

Dalla definizione di DFT si ha:

$$F_k = \sum_{j=0}^{3} f_j \omega_4^{-jk} \quad k = 0, 1, 2, 3$$

Poiché gli esponenziali complessi sono:

$$\begin{array}{l}\omega_4^{-0} = e^{-i\frac{2\pi}{4}0} = cos(\frac{2\pi}{4}0) - isen(\frac{2\pi}{4}0) = 1\\ \omega_4^{-1} = e^{-i\frac{2\pi}{4}1} = cos(\frac{2\pi}{4}1) - isen(\frac{2\pi}{4}1) = -i\\ \omega_4^{-2} = e^{-i\frac{2\pi}{4}2} = cos(\frac{2\pi}{4}2) - isen(\frac{2\pi}{4}2) = -1\\ \omega_4^{-3} = e^{-i\frac{2\pi}{4}3} = cos(\frac{2\pi}{4}3) - isen(\frac{2\pi}{4}3) = i\\ \omega_4^{-4} = e^{-i\frac{2\pi}{4}4} = cos(\frac{2\pi}{4}4) - isen(\frac{2\pi}{4}4) = 1\\ \omega_4^{-6} = e^{-i\frac{2\pi}{4}6} = cos(\frac{2\pi}{4}6) - isen(\frac{2\pi}{4}6) = -1\\ \omega_4^{-9} = e^{-i\frac{2\pi}{4}9} = cos(\frac{2\pi}{4}9) - isen(\frac{2\pi}{4}9) = -i\end{array} \tag{5.46}$$

segue:

$$\begin{array}{llll} F_0 = f_0 & + f_1 & +f_2 & + f_3\\ F_1 = f_0 & -if_1 & -f_2 & +if_3\\ F_2 = f_0 & - f_1 & +f_2 & - f_3\\ F_3 = f_0 & +if_1 & -f_2 & -if_3\end{array} \tag{5.47}$$

Se si raccolgono, in maniera opportuna, alcuni termini nella (5.47) si ottiene:

$$\begin{array}{l} F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)}\\ F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)}\\ F_2 = (f_0 + f_2) - (f_1 + f_3)\\ F_3 = (f_0 - f_2) + i(f_1 - f_3)\end{array} \tag{5.48}$$

e si separano le componenti del vettore DFT con indice pari da quelle con indice dispari, si ha:

$$\begin{array}{ll} F_0 = \boldsymbol{(f_0 + f_2) + (f_1 + f_3)} & F_1 = \boldsymbol{(f_0 - f_2) - i(f_1 - f_3)}\\ F_2 = (f_0 + f_2) - (f_1 + f_3) & F_3 = (f_0 - f_2) + i(f_1 - f_3)\end{array} \tag{5.49}$$

Secondo l'algoritmo di Gentleman e Sande, come descritto nel paragrafo 5.4.2, il calcolo del vettore F viene realizzato attraverso i seguenti passi:

- **passo 1:** si calcolano 2 vettori di lunghezza $2 = N/2$: (y_0, y_1) e (z_0, z_1), con:

$$\begin{array}{ll} y_0 = (f_0 + f_2) & z_0 = (f_0 - f_2)\\ y_1 = (f_1 + f_3) & z_1 = -i(f_1 - f_3)\end{array} \tag{5.50}$$

- **passo 2:** si calcolano 4 vettori di lunghezza $1 = N/4$:

$$\begin{array}{ll} y'_0 = y_0 + y_1 & z'_0 = z_0 + z_1\\ y'_1 = y_0 - y_1 & z'_1 = z_0 - z_1\end{array} \tag{5.51}$$

Il vettore (y'_0, y'_1, z'_0, z'_1) è la DFT del vettore F.

Esplicitando le componenti del vettore costruito al secondo passo, si ottiene:

$$(y'_0, y'_1, z'_0, z'_1) = (y_0 + y_1, y_0 - y_1, z_0 + z_1, z_0 - z_1) =$$

$$= f_0 + f_2 + f_1 + f_3, f_0 + f_2 - (f_1 + f_3), (f_0 - f_2) - i(f_1 - f_3), (f_0 - f_2) + i(f_1 - f_3)) = (F_0, F_2, F_1, F_3)$$

cioè si ottengono le componenti del vettore F disposte non nell'ordine naturale, bensì nell'ordine del *bit reversal (scrambled)*, ovvero, ad esempio, la componente di indice $2 = (100)_2$ si trova al posto della componente di indice $1 = (001)_2$.

Le operazioni riportate nella (5.50) si possono descrivere graficamente utilizzando lo schema della Figura 5.9, che corrisponde a due *butterfly* innestate.

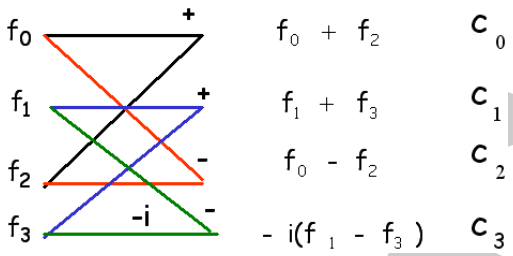

Figura 5.9: Schema del passo 1.

Osserviamo che per ottenere le componenti y_0 e z_0, risultato dell'operazione *butterfly* sulle componenti (f_0, f_2) del vettore iniziale, è necessario utilizzare **esclusivamente** le componenti (f_0, f_2). Analogamente, per ottenere le componenti y_1 e z_1 è necessario utilizzare solo le componenti (f_1, f_3). Questo fatto induce a implementare l'algoritmo *in place*, ovvero di utilizzare le medesime locazioni di memoria del vettore di input f per memorizzare le componenti dei vettori costruiti al primo passo. Naturalemnte, questo si può fare se i due vettori y e z ottenuti al primo passo si costruiscono calcolando le componenti omologhe rispettivamente del primo e del secondo vettore.
Introdotto il vettore

$$c = (y_0, y_1, z_0, z_1)$$

il primo passo dell'algoritmo di Gentleman e Sande equivale alla costruzione di tale vettore effettuando 2 butterly: la prima necessaria al calcolo delle componenti (c_0, c_2) e la seconda per il calcolo delle componenti (c_1, c_3). Inoltre, ciascuna *butterfly* può essere effettuata *in place*, ovvero il risultato può essere memorizzato utilizzando le stesse locazioni di memoria riservate per le componenti di input.
Analogamente al **secondo passo**, le operazioni riportate nella (5.51) si possono descrivere graficamente utilizzando lo schema della Figura 5.10.

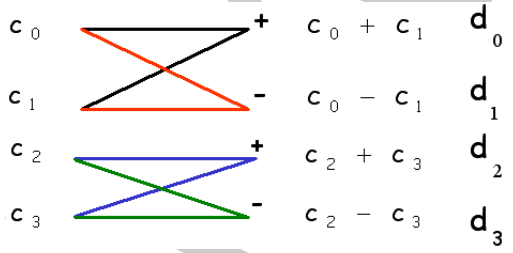

Figura 5.10: Schema del passo 2.

Anche al secondo passo, le componenti (y'_0, y'_1) risultato della butterfly sulle componenti (c_0, c_1) del vettore costruito al primo passo, sono ottenute utilizzando esclusivamente (c_0, c_1) e pertanto possono essere memorizzate nelle locazioni di memoria di queste ultime. Analogo risultato vale per l'altra coppia di componenti, ovvero per (z'_0, z'_1).
Anche in questo caso, introdotto il vettore

$$d = (y'_0, y'_1, z'_0, z'_1)$$

le sue componenti sono ottenute effettuando 2 *butterfly in place*.
In conclusione, per il calcolo di una DFT di lunghezza 4 sono stati effettuati 2 passi, in ciascuno dei quali è richiesto il calcolo di *2 butterfly*, il risultato di ciascuna *butterfly* può essere memorizzato sulle medesime componenti utilizzate per calcolare la *butterfly* stessa (algoritmo *in place*).

Nella Figura 5.11 sono mostrate le *butterfly* necessarie al calcolo di una DFT di lunghezza 4. ♣

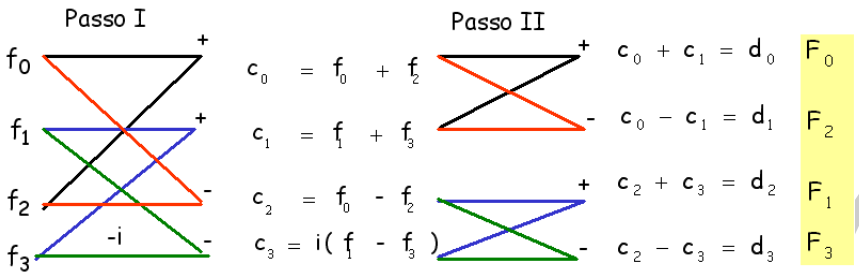

Figura 5.11: Schema di una DFT di lunghezza 4.

♣ **Esempio 5.23. DFT di lunghezza 8**
Supponiamo $N = 2^3 = 8$. Vogliamo effettuare la DFT di un vettore di lunghezza 8:

$$\underline{f} = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7).$$

Dalla definizione di DFT si ha:

$$F_k = \sum_{j=0}^{7} f_j \omega_8^{-jk} \quad k = 0, 1, ..., 7$$

Seguendo l'algoritmo di Gentleman e Sande si ha che al **passo 1**, si calcolano i due vettori y e z, di lunghezza 4 e componenti:

$$\begin{array}{ll} y_0 = (f_0 + f_4) & z_0 = (f_0 - f_4) \\ y_1 = (f_1 + f_5) & z_1 = (f_1 - f_5) \\ y_2 = (f_2 + f_6) & z_2 = (f_2 - f_6) \\ y_3 = (f_3 + f_7) & z_3 = (f_3 - f_7) \end{array} \tag{5.52}$$

Le componenti omologhe dei due vettori y e z si ottengono rispettivamente come somma e differenza di coppie di componenti del vettore f. In particolare, effettuando $N/2 = 4$ *butterfly* costruite sulle componenti del vettore f che distano di $N/2 = 4$ unità, si ottengono le componenti dei vettori y e z. Introdotto il vettore

$$c = (y_0, y_1, y_2, y_3, z_0, z_1, z_2, z_3)$$

il primo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su f.
Al **passo 2** si calcolano 4 vettori di lunghezza 2 y', y'', z' e z'', di componenti:

$$\begin{array}{l} y'_0 = c_0 + c_2 \\ y'_1 = c_1 + c_3 \\ y''_0 = c_0 - c_2 \\ y''_1 = c_2 - c_3 \\ z'_0 = c_4 + ic_6 \\ z'_1 = c_5 + ic_7 \\ z''_0 = c_4 - ic_6 \\ z''_1 = c_5 - ic_7 \end{array} \tag{5.53}$$

Al secondo passo, i 4 vettori y', y'', z' e z'' si ottengono effettuando $N/2 = 4$ *butterfly* costruite utilizzando le componenti dei vettori y e z costruiti al passo precedente. In particolare, per ciascuna

butterfly si combinano le componenti che distando di $N/2^2 = 2$ unità. Introdotto il vettore

$$d = (y'_0, y'_1, y''_0, y''_1, z'_0, z'_1, z''_0, z''_1)$$

il secondo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su c.
Al **passo 3** si calcolano le quantità:

$$\begin{array}{l} y^{'''} = (d_0 + d_1) \\ y^{iv} = (d_0 - d_1) \\ y^{v} = (d_2 + id_3) \\ y^{vi} = (d_2 - id_3) \\ z^{'''} = (d_4 + \omega_8^2 d_5) \\ z^{iv} = (d_4 - \omega_8^2 d_5) \\ z^{v} = (d_6 + \omega_8^6 d_7) \\ z^{vi} = (d_6 - \omega_8^6 d_7) \end{array} \tag{5.54}$$

Al terzo passo, queste 8 quantità si ottengono effettuando $N/2 = 4$ *butterfly* costruite utilizzando le componenti dei vettori costruiti al passo precedente. In particolare, per ciascuna *butterfly* si combinano le componenti che distando di $N/2^4 = 1$ unità. Introdotto il vettore

$$e = (y''', y^{iv}, y^{v}, z^{iii}, z^{iv}, z^{v}, z^{vi})$$

il terzo passo dell'algoritmo porta alla costruzione di tale vettore memorizzato su d.

In generale, al passo $k = 1, 2, 3$ sono necessarie $4 = N/2$ butterfly costruite utilizzando componenti del vettore costruito al passo precedente che distano di $N/2^k = 2^3/2^k = 2^{3-k}$ unità. Tali componenti sono anche dette *nodi duali*.

Osserviamo che se indichiamo con c_k il vettore di $N = 8$ componenti costruito al passo k dell'algoritmo, la relazione che lega la coppia di nodi duali costruita al passo k con la medesima coppia relativa al passo $k-1$ è si esprime in maniera naturale attraverso la formula ricorrente:

$$\begin{aligned} c_k(r) &= c_{k-1}(r) + \omega_8^{q/2^k} c_{k-1}(r + 2^{3-k}) \quad r = 1, N/2 \\ c_k(r + 2^{3-k}) &= c_{k-1}(r) + \omega_8^{q'/2^k} c_{k-1}(r + 2^{3-k}), \quad r = 1, N/2 \end{aligned}$$

Poiché $q' = q + 2^{k-1}$, segue che $\omega_N^{q'/2^k} = -\omega_N^{q/2^k}$ e quindi per ciascuna *butterfly* è necessario valutare un unico esponenziale complesso.

Infine, esplicitando i calcoli ci si accorge che il vettore $\underline{e} = (e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7)$ é la DFT del vettore $\underline{F}$ dove le componenti sono disposte non nell'ordine naturale ma nell'ordine $(F_0, F_4, F_2, F_6, F_1, F_5, F_3, F_7)$ (*scrambled*) ovvero, del bit inverso. ♣

Lo schema generale dell'algoritmo FFT radix-2, nella versione di Gentleman e Sande, per calcolare le componenti del vettore F prevede di combinare a due a due le componenti del vettore costruito al passo precedente in modo da effettuare somme e differenze secondo lo schema *butterfly*. In particolare:

- passo 1: si calcolano 2^{m-1} *butterfly* ottenute combiando le componenti di f a distanza $p = N/2$, come mostrato nella Figura 5.12.

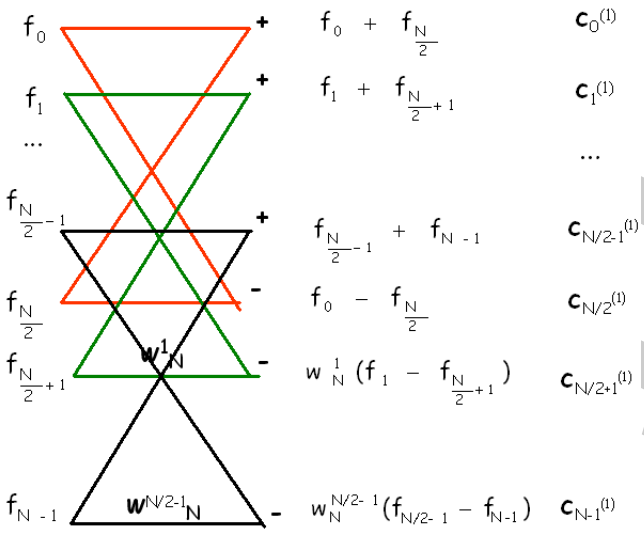

Figura 5.12: Schema del passo 1 di una DFT di lunghezza N.

- passo 2: si calcolano 2^{m-1} *butterfly* ottenute combinando le componenti di f a distanza $p = N/4$, come mostrato nella Figura 5.13.

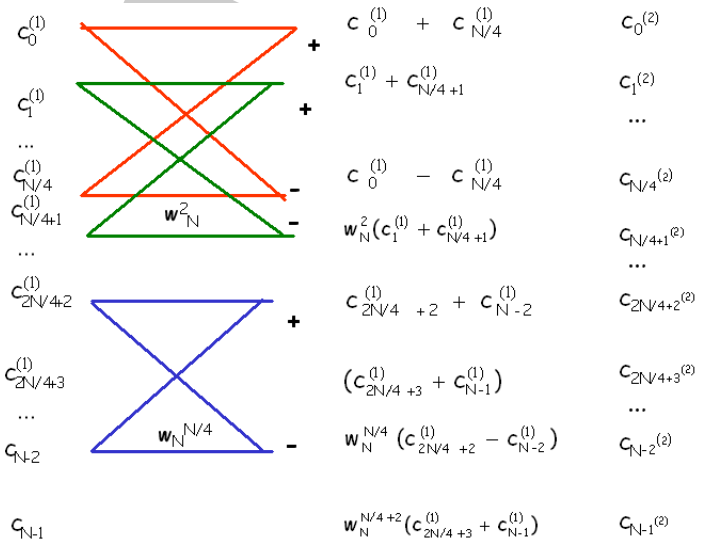

Figura 5.13: Schema del passo 2 di una DFT di lunghezza N.

- passo k: si calcolano 2^{m-1} *butterfly* ottenute combinando le componenti di f a distanza $p = N/2^k$, come mostrato nella Figura 5.14.

- ...

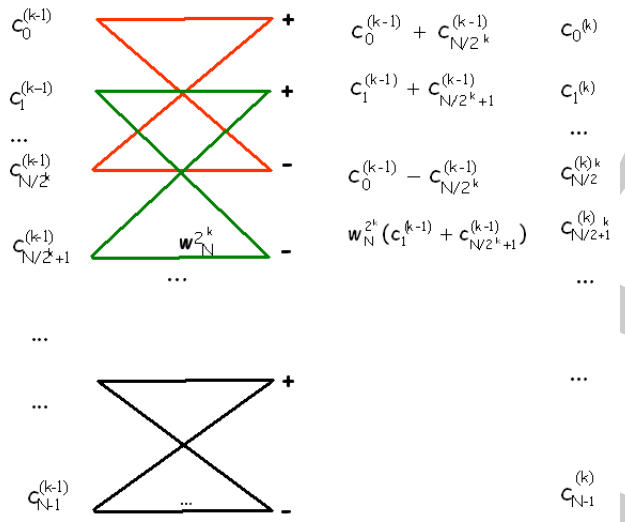

Figura 5.14: Schema del passo k di una DFT di lunghezza N.

- passo m: si calcolano 2^{m-1} *butterfly* ottenute combinando gli elementi di f a distanza $p = N/2^{m-1}$, come mostrato nella Figura 5.15.

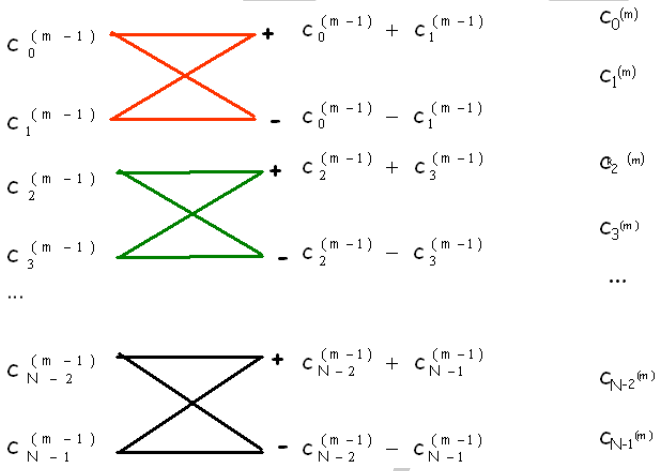

Figura 5.15: Schema del passo m di una DFT di lunghezza N.

Si nota che ad ogni passo ciascuna coppia di componenti ottenuta da un'operazione *butterfly*, può essere memorizzata sulle posizioni di memoria occupate dalle rispettive componenti utilizzate nell'operazione *butterfly*. Ciò è lecito perchè per ciascuna *butterfly*, la coppia di componenti coinvolte viene utilizzata esclusivamente per calcolare le componenti risultanti da quella butterfly. In altre parole, l'algoritmo può essere implementato *in place*, ovvero il vettore calcolato ad ogni passo può essere memorizzato sul vettore di input.
Il vettore c_k costruito al passo k dell'algoritmo si ottiene a partire dal vettore c_{k-1}

attraverso la formula ricorrente:

$$c_k(r) = c_{k-1}(r) + \omega_N^{q/2^k} c_{k-1}(r + 2^{m-k})$$
$$c_k(r + 2^{3-k}) = c_{k-1}(r) + \omega_N^{q'/2^k} c_{k-1}(r + 2^{m-k}) \quad (\omega_N^{q'/2^k} = \omega_N^{-q/2^k})$$

Il vettore costruito all'ultimo passo è la DFT del vettore F disposto nell'ordine *scrambled* (o anche del *bit inverso*).

L'algoritmo di Cooley e Tukey prevede, invece, una fase di *riordinamento* iniziale e una seconda fase di *calcolo* (in particolare, operazioni tipo butterfly tra componenti che distano di 1, poi di 2, poi di 4, poi di 8 componenti, e così via) sul vettore ordinato. In tal modo, questa versione produce il vettore risultante secondo l'ordine naturale.

Un aspetto interessante, da un punto di vista implementativo, è osservare che la decomposizione del vettore di input, nell'algoritmo di Cooley e Tukey, o del vettore DFT nella variante di Gentleman e Sande, si può effettuare utilizzando la rappresentazione binaria degli indici j e k. Infatti notiamo che, ad esempio, nell'algoritmo di Cooley e Tukey, separare le componenti di f con indice pari da quelle con indice dispari significa dimezzare l'indice $j = 0, \ldots, N-1$ e quindi per il numero intero j esiste una rappresentazione del tipo:

$$j = 2l + q, \quad j = 0, \ldots, \underbrace{N/2}_{2^{m-1}} - 1, \quad l, q = 0, 1$$

Analogamente, applicare questo ragionamento ai due sottovettori di lunghezza $N/2$, significa dimezzare l'indice l e quindi:

$$j = 2 \cdot \underbrace{(2r + s)}_{l} + q, \quad r = 0, \ldots, \underbrace{N/4}_{2^{m-2}} - 1, \quad q, s = 0, 1$$

e ancora, riapplicare questo ragionamento ai 4 vettori di lunghezza $N/4$, significa dimezzare r:

$$j = 2 \cdot (2 \cdot \underbrace{(2t + v)}_{r} + s) + q, \quad t = 0, \ldots, \underbrace{N/8}_{2^{m-3}} - 1, \quad v, s, q = 0, 1$$

Dopo $m-1$ passi si ha una rappresentazione di j del tipo:

$$j = 2^{m-1} \cdot j_{m-1} + 2^{m-2} \cdot j_{m-2} \ldots + j_0, \quad k = 0, \ldots, m-1, \quad j_k = 0, 1$$

Ovvero abbiamo rappresentato il numero intero j come numero binario. Applicando questo ragionamento agli indici j e k, la (5.36) diventa:

$$F(k) = \sum_{j_{m-1}=0}^{1} \sum_{j_{m-2}=0}^{1} \ldots \sum_{j_0=0}^{1} f(2^{m-1} \cdot j_{m-1} + 2^{m-2} \cdot j_{m-2} \ldots + j_0) \omega_N^{-jk}$$

Tenendo conto che

$$\omega_N^{-jk} = \omega_N^{-k(2^{m-1}\cdot j_{m-1}+2^{m-2}\cdot j_{m-2}\ldots+j_0)} = \omega_N^{-kj_{m-1}}\omega_{N/2}^{-kj_{m-2}}\cdots\omega_2^{-k_0 j_0}$$

segue:

$$\omega_N^{-jk} = \omega_N^{-kj_{m-1}}\cdot\omega_{2^{m-1}}^{-kj_{m-2}}\cdot\omega_{2^{m-2}}^{-kj_{m-3}}\cdots\omega_2^{-kj_0}$$

Inoltre, se:

$$k = 2^{m-1}\cdot k_{m-1} + 2^{m-2}\cdot k_{m-2}\ldots + k_0$$

si ha che:

$$\omega_2^{-kj_0} = \omega_2^{-j_0 k_{m-1}}$$

e quindi

$$F(k) = \sum_{j_{m-1}=0}^{1}\omega_N^{-kj_{m-1}}\sum_{j_{m-2}=0}^{1}\omega_{2^{m-1}}^{-kj_{m-2}}\cdots\sum_{j_0=0}^{1}f(2^{m-1}\cdot j_{m-1}+2^{m-2}\cdot j_{m-2}\ldots+j_0)\omega_2^{-j_0 k_{m-1}}$$

Fissato l'indice k, il calcolo di ciascuna componente del vettore DFT, espresso in questo modo, si puó interpretare come il calcolo di $m-1$ somme innestate, in cui ciascuna somma è una DFT di lunghezza 2. Questa è l'idea alla base dell'implementazione dell'algoritmo FFT radix-2 di Gentleman e Sande. Al contrario, partendo da somma più esterna, si hanno 2 DFT di lunghezza 2^{m-1} che corrispondono alle rimanenti $m-1$ sommatorie. Considerando le prime due sommatorie esterne, si hanno 4 DFT di lunghezza 2^{m-2} e così proseguendo verso l'interno, arriviamo al calcolo di 2^{m-1} DFT di lunghezza 2, ovvero alla versione dell'algoritmo FFT radix-2 proposta da Cooley e Tukey.

Da un punto di vista implementativo, entrambe le varianti si possono ottenere selezionando il valore (0/1) di un bit della rappresentazione binaria N (ad esempio, nella variante di Cooley e Tukey, il valore 0 del bit meno significativo nella rappresentazione binaria di j corrisponde all' indice pari, il valore 1 corrisponde all'indice dispari.). Questo significa che se consideriamo l'algoritmo di Cooley e Tukey e prima di iniziare i calcoli , si riordinano le componenti del vettore di input secondo l'ordine inverso (del bit reversal), ad ogni passo sarà necessario combinare sempre componenti che nel nuovo ordinamento si trovano in locazioni di memoria adiacenti. Inoltre, il vettore risultante dopo m passi è il vettore DFT ordinato secondo l'ordine naturale delle componenti. Nella variante di Gentleman e Sande, invece, tale riordinamneto iniziale non è necessario. Tale riordinamento dovrà essere applicato unicamente alla componenti del vettore DFT qualora si vogliano tali componenti nell'ordine naturale.

```
procedure fft2(input: f,N, out: f)
/# SCOPO: calcolo del vettore DFT di f
/# .....
     for k=1,m
/# ciclo sul numero di passi
        d := N/2^k;                          distanza nodi duali al passo k
        nesp := 2^(k-1);                     esponenziali diversi al passo k
        s := 0;
        for i=1,nesp                         ciclo sugli esponenziali diversi
         r := s;                             indice prima componente
            for l=0,d-1                      calcolo nodi duali
            r := l + s;                      indice primo elemento della coppia
            r1 := r + d;                     indice secondo elemento della coppia
            t := exp(z/2^k) * f(r1);
            f(r1) := f(r) - t;               calcolo coppia di nodi duali
            f(r) := f(r) + t;
            endfor l;
            s := s + 2d;                     salto
          endfor i;
       endfor k;
```

Procedura 5.1: Algoritmo di FFT radix-2: calcolo del vettore c_k, $k = 1, \ldots, m$.

5.4.5 Formulazione matriciale dell'algoritmo FFT radix-2

L'algoritmo FFT radix-2, nella versione di Gentleman e Sande, applicato ad un vettore di lunghezza N, produce la fattorizzazione della matrice di Fourier nel prodotto di $\log_2(N)$ matrici sparse e strutturate.

♣ **Esempio 5.24.** Sia $N = 2^1$; come nell'**esempio 5.21**, la DFT del vettore:

$$\underline{f} = (f_0, f_1)$$

è:

$$\begin{array}{l} F_0 = f_0 + f_1 \\ F_1 = f_0 - f_1 \end{array} \tag{5.55}$$

Se si considera la matrice di Fourier di dimensione 2:

$$W := A_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

si ha:

$$W \cdot f = F$$

La matrice A_1 è una matrice a blocchi:

$$A_1 = \begin{bmatrix} I_1 & I_1 \\ \Omega_2 & -\Omega_2 \end{bmatrix}$$

dove I_1 è la matrice identica di ordine 1 e Ω_2 è la matrice diagonale di ordine 1 del tipo:

$$\Omega_2 = diag(w_2^0) = diag(1)$$

♣

♣ **Esempio 5.25.** Sia $N = 2^2 = 4$; come nell'**esempio 5.22**, la DFT del vettore:

$$\underline{f} = (f_0, f_1, f_2, f_3)$$

è:

$$\begin{array}{l} F_0 = f_0 + f_1 + f_2 + f_3 \\ F_1 = f_0 - if_1 - f_2 + if_3 \\ F_2 = f_0 - f_1 + f_2 - f_3 \\ F_3 = f_0 + if_1 - f_2 - if_3 \end{array} \tag{5.56}$$

Se si definisce la matrice di Fourier di dimensione 4:

$$W := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

allora:

$$W \cdot f = F$$

Nell'**esempio 5.22** si è visto anche che, per calcolare le somme (5.56) è necessario:

- al **passo 1** calcolare le componenti del vettore c:

$$\begin{array}{ll} c_0 = (f_0 + f_2) & c_2 = (f_0 - f_2) \\ c_1 = (f_1 + f_3) & c_3 = -i(f_1 - f_3) \end{array} \tag{5.57}$$

Introdotta la matrice:

$$A_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & -i & 0 & i \end{bmatrix},$$

le operazioni (5.57) corrispondono al prodotto matrice per vettore:

$$A_1 \cdot f = c$$

La matrice A_1 è una matrice strutturata a blocchi, del tipo:

$$A_1 = \begin{bmatrix} I_2 & I_2 \\ \Omega_4 & -\Omega_4 \end{bmatrix}$$

essendo I_2 una matrice identica di ordine 2 e Ω_4 una matrice diagonale di ordine 2 definita come

$$\Omega_4 = diag(w_4^0, w_4^1) = diag(1, w_4^1).$$

Se si pone:

$$B_1 = A_1$$

risulta:

$$A_1 = I_1 \otimes B_1$$

dove il simbolo $\otimes$ denota il prodotto tensoriale di due matrici[25] e I_1 la matrice identica di ordine 1.

- Al **passo 2** calcolare le componenti del vettore d:

$$\begin{array}{ll} d_0 = (c_0 + c_1) & d_2 = (c_2 + c_3) \\ d_1 = (c_0 - c_1) & d_3 = (c_2 - c_3) \end{array} \tag{5.58}$$

Introdotta la matrice:

$$A_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

le operazioni (5.58) corrispondono al prodotto matrice per vettore:

$$A_2 \cdot c = d$$

La matrice A_2 è una matrice strutturata a blocchi, del tipo:

$$A_2 = \begin{bmatrix} I_1 & I_1 & 0 & 0 \\ \Omega_2 & -\Omega_2 & 0 & 0 \\ 0 & 0 & I_1 & I_1 \\ 0 & 0 & \Omega_2 & -\Omega_2 \end{bmatrix}$$

[25] Assegnate due matrici $A = (a_{i,j})_{i,j=1,...n} \in \Re^{n \times n}$ e $B = (b_{i,j})_{i,j=1,...m} \in \Re^{m \times m}$ il prodotto tensoriale di A e B è una matrice $C \in \Re^{mn \times mn}$ del tipo:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & ... & a_{1n}B \\ ... & ... & ... & ... \\ a_{n1}B & a_{n2}B & ... & a_{nn}B \end{bmatrix}$$

essendo I_1 una matrice identica di ordine 1 e

$$\Omega_2 = diag(w_2^0) = diag(1)$$

una matrice diagonale di ordine 1. Se si pone:

$$B_2 = \begin{bmatrix} I_1 & I_1 \\ \Omega_2 & -\Omega_2 \end{bmatrix}$$

risulta che:

$$A_2 = I_2 \otimes B_2,$$

dove I_2 è la matrice identica di ordine due.

In conclusione, la DFT di un vettore lunghezza 4 si ottiene effettuando due prodotti matrice-vettore:

1. $A_1 \cdot f = c$;
2. $A_2 \cdot c = d$.

essendo A_1 ed A_2 matrici strutturate a blocchi.
Si osserva, inoltre, che se:

$$W^* := A_2 \cdot A_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \\ 1 & i & -1 & -i \end{bmatrix}$$

W^* coincide con la matrice di Fourier W *a meno di uno scambio di righe*. In particolare, detta P la matrice di permutazione:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ottenuta dalla matrice identica I_4 di ordine 4 scambiando la seconda e la terza riga, si ha:

$$W^* \cdot P = W$$

In conclusione, l'algoritmo FFT radix-2 applicato ad un vettore di lunghezza 4 produce la fattorizzazione di W nel prodotto di due matrici strutturate a blocchi:

$$W = W^* \cdot P = A_2 \cdot A_1 \cdot P$$

da cui:

$$Wf = F \Leftrightarrow A_2 \cdot A_1 \cdot P \; f = F$$

Si osserva che la permutazione della seconda e terza riga, definita attraverso il prodotto a destra della matrice di Fourier W per la matrice di permutazione P, corrisponde di fatto alla permutazione delle componenti di f indotta dal bit reversal.

♣

In generale, per effettuare la DFT di un vettore di lunghezza $N = 2^m$:

$$\underline{f} = (f_0, f_1, ..., f_{2^m})$$

sono necessari m passi in ciascuno dei quali si effettua un prodotto matrice per vettore. In particolare:

- al **passo 1**, introdotta la matrice:

$$A_1 = I_1 \otimes B_1$$

con:

$$B_1 = \begin{bmatrix} I_{2^{m-1}} & I_{2^{m-1}} \\ \Omega_{2^m} & -\Omega_{2^m} \end{bmatrix}$$

dove $I_{2^{m-1}}$ ed

$$\Omega_{2^m} = (w^0_{2^m}, w^1_{2^m}, ..., w^{2^{m-1}}_{2^m})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-1} si calcola il vettore:

$$c^{(1)} = A_1 \cdot f$$

- al **passo 2**, introdotta la matrice:

$$A_2 = I_2 \otimes B_1$$

con

$$B_2 = \begin{bmatrix} I_{2^{m-2}} & I_{2^{m-2}} \\ \Omega_{2^{m-1}} & -\Omega_{2^{m-1}} \end{bmatrix}$$

dove $I_{2^{m-2}}$ ed

$$\Omega_{2^{m-1}} = (w^0_{2^{m-1}}, w^1_{2^{m-1}}, ..., w^{2^{m-2}}_{2^{m-1}})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-2}, si calcola il vettore:

$$c^{(2)} = A_2 \cdot c^{(1)}$$

- al **passo k**, introdotta la matrice:

$$A_k = I_{2^{k-1}} \otimes B_k$$

con

$$B_k = \begin{bmatrix} I_{2^{m-k}} & I_{2^{m-k}} \\ \Omega_{2^{m-k+1}} & -\Omega_{2^{m-k+1}} \end{bmatrix}$$

dove $I_{2^{m-k}}$ ed

$$\Omega_{2^{m-k}} = (w^0_{2^{m-k}}, w^1_{2^{m-k}}, ..., w^{2^{m-k+1}}_{2^{m-k}})$$

sono rispettivamente la matrice identica e una matrice diagonale di ordine 2^{m-k}, si calcola il vettore:

$$c^{(k)} = A_k \cdot c^{(k-1)}$$

In conclusione, la DFT di un vettore lunghezza $N = 2^m$ si ottiene effettuando m prodotti matrice per vettore:

1. $A_1 \cdot f = c^{(1)}$;

2. $A_2 \cdot c^{(1)} = c^{(2)}$;

   ...

m. $A_m \cdot c^{(m-1)} = c^{(m)}$

essendo A_1, A_2,..., A_m matrici strutturate a blocchi. Se si denota con W^* la matrice che si ottiene effettuando il prodotto delle matrici A_1, A_2, ..., A_m ovvero:

$$W^* = A_m \cdot A_{m-1} ... A_1$$

tale matrice coincide con la matrice di Fourier W "a meno di uno scambio di righe", ovvero:

$$W = W^* \cdot P$$

essendo P una matrice di permutazione di ordine 2^m costruita ordinando secondo la relazione del bit inverso le righe della matrice identica I_{2^m}.
In conclusione, l'algoritmo FFT radix-2, applicato ad un vettore di lunghezza $N = 2^m$, fattorizza la matrice di Fourier W nel prodotto di m matrici, ovvero:

$$W = W^* \cdot P = A_m \cdot A_{m-1} \cdots A_1 \cdot P \tag{5.59}$$

da cui:

$$W f = F \Leftrightarrow A_m \cdot A_{m-1} \cdot A_1 \cdots P \; f = F \tag{5.60}$$

Concludiamo questo paragrafo osservando che la matrice di Fourier W gode di una interessante proprietà riguardante l'andamento dei valori singolari. Si dimostra, infatti, che partizionando la matrice di Fourier in p^2 blocchi, di dimensione n/p, tali sottomatrici hanno valori singolari compresi tra zero e uno, e solo una porzione di questi sono vicini all'unità. In altri termini, effettuando una decomposizione in valori singolari "troncata" di siffatti blocchi, si perviene ad una formulazione a blocchi dell'algoritmo FFT radix-2 che, sebbene fornisca il vettore DFT corretto a meno di una prefissata tolleranza, avendo assunto numericamente nulli i valori singolari al di sotto di tale tolleranza, si presta ad una implementazione in ambiente di calcolo parallelo e distribuito più efficiente di quelle basate sugli algoritmi di Cooley e Tukey o Gentleman e Sande [26].

5.4.6 Stabilità dell'algortimo FFT radix-2

Sia $\Im$ un sistema aritmetico a precisione finita e sia $F = W \cdot f$, il vettore DFT di f. Indicato con $\widehat{F}$ il valore di F, calcolato in $\Im$ utilizzando l'algoritmo FFT, il teorema seguente mostra che la propagazione dell'errore nel sistema aritmetico $\Im$ cresce linearmente con m, ovvero che l'algoritmo FFT è *stabile* nel senso della f.e.a.

[26] [7]

Teorema 5.4.1. *Sia f un vettore di lunghezza $N = 2^m$ esattamente rappresentabile in $\Im$ ovvero $f = fl(f) = \widehat{f}$. Supponiamo che gli esponenziali complessi ω_k siano precalcolati e che su di essi sia stato commesso un errore $|\epsilon_{jk}|$ tale che:*

$$\hat{\omega}_j^k = fl(\omega_j^k) = \omega_j^k + \epsilon_{jk} \quad |\epsilon_{jk}| \leq u \tag{5.61}$$

essendo u la massima accuratezza relativa di $\Im$.
Allora:

$$\frac{||F - \widehat{F}||_2}{||F||_2} \leq \frac{m\eta}{1 - m\eta}, \quad \eta := \mu + \gamma_4(\sqrt{2} + \mu) \tag{5.62}$$

essendo $\gamma_4 = \frac{4u}{1-4u}$.

Dimostrazione Si osserva che per le matrici in (5.60):

$$||A_k||_2 = \sqrt{2} \tag{5.63}$$

e che:

$$|| \, |A_k| \, ||_2 = 2 = \sqrt{2}||A_k||_2$$

Denotata con $\widehat{A}_k$ la matrice definita dai valori degli esponenziali complessi calcolati $\widehat{\omega}_k^j$. Allora:

$$\widehat{F} = fl(\widehat{A}_m....\widehat{A}_1) = (\widehat{A}_m + \Delta\widehat{A}_m)...(\hat{A}_1 + \Delta\widehat{A}_1) \cdot P_r \cdot f \tag{5.64}$$

dove con P_r si indica una matrice di permutazione di ordine r. Poiché ciascuna matrice A_k ha solo due elementi non nulli per riga ed inoltre ciascun elemento è un numero complesso[27] si ha:

$$|\Delta\widehat{A}_k| \leq \gamma_4|\widehat{A_k}|$$

per cui:

$$||\Delta\widehat{A}_k||_2 \leq || \, |\Delta\hat{A}_k| \, ||_2 \leq \gamma_4|| \, |\widehat{A_k}| \, ||_2$$

Dall'ipotesi (5.61) segue che:

$$\widehat{A}_k = A_k + \Delta A_k \quad ||\Delta A_k|| \leq \sqrt{2}\mu = \mu||A_k||_2 \tag{5.65}$$

Utilizzando la (5.63) e la (5.65) si ottiene che:

$$|| \, |\widehat{A}_k| \, ||_2 \leq || \, |A_k| \, ||_2 + || \, |\delta A_k| \, ||_2 \leq (\sqrt{2} + \mu)||A_k||_2. \tag{5.66}$$

La (5.64), mediante la (5.65) si può scrivere come:

$$\hat{F} = (A_m + \Delta A_m + \Delta\widehat{A}_m)...(A_1 + \Delta A_1 + \Delta\widehat{A}_1) \cdot P_r \cdot f \tag{5.67}$$

Posto:

$$E_k = \Delta A_k + \Delta\widehat{A}_k \quad k = 1, ..., m$$

[27] Utilizzando un modello standard di aritmetica a precisione finita [17] se x ed y sono numeri complessi esattamente rappresentabili si assume che:

$$\begin{array}{ll} fl(x \pm y) = (x \pm y)(1 + \delta) & |\delta| \leq u \\ fl(xy) = (xy)(1 + \delta) & |\delta| \leq \sqrt{2}\frac{2u}{1-2nu} \\ fl(x/y) = (x/y)(1 + \delta) & |\delta| \leq \sqrt{2}\frac{4u}{1-4nu} \end{array}$$

si ottiene:

$$\hat{F} = (A_m + E_m)...(A_1 + E_1) \cdot P_r \cdot f \tag{5.68}$$

e dalla (5.65) e (5.66) si ha:

$$||E_k||_2 \leq (\mu + \gamma_4(\sqrt{2} + \mu))||A_k||_2 = \eta||A_k||_2.$$

Applicando il Lemma 5.4.1[28] si trova che:

$$||F - \widehat{F}||_2 \leq [(1+\eta)^m - 1]||A_m||_2...||A_1||_2||P_r||_2||f||_2 \leq$$

$$\frac{m\eta}{1-m\eta} 2^{m/2}||f||_2$$

poiché:

$$||f||_2 = n^{-1/}||F||_2 =^{-m/2} ||F||_2$$

segue la tesi. ■

5.5 Software matematico per la FFT

Esistono numerose implementazioni degli algoritmi FFT, disponibili sia come singoli elementi di software sia come librerie che come software proprietari specializzati per tipologie di processori . La maggior parte di questi software sono scritti in linguaggio C/C++, altri in Fortran 77/90. Le prime implementazioni risalgono al 1969 [29], l'ultima è del 2009 (`FFTW 3.2.1` [9]). Il sito netlib consente di accedere alle più diffuse librerie di software matematico tra cui, ad esempio, `FFTPACK` [8], `MPFUN` [20], `NAPACK` [23] e `SCILIB` [25].

Una delle motivazioni per cui esitono così tanti elementi di software che implementano l'algoritmo FFT nasce dalla necessità di specializzare l'algoritmo FFT in base alle differenti caratteristiche dell'architettura di calcolo e del problema (reale, complesso, simmetrico, hermitiano, radix-r, a radici miste, etc.) al fine di migliorarne le prestazioni. Una rivoluzione nello sviluppo di software matematico efficiente per il calcolo della FFT

[28] [17]

Lemma 5.4.1. *Se le matrici $X_j + \Delta X_j \in \Re^{n\times n}$ soddisfano la relazione $||\Delta X_j|| \leq \delta_j||X_j||$ per ogni j in una qualsiasi norma matriciale submoltiplicativa, allora:*

$$\left|\left|\prod_{j=0}^{m}(X_j + \Delta X_j) - \prod_{j=0}^{m} X_j\right|\right| \leq \left(\prod_{j=0}^{m}(1+\delta_j) - 1\right)\prod_{j=0}^{m} X_j$$

[29] [27]

si è avuta nel 1998, con il software *Fast Fourier Transform in the West* (`FFTW`)[30]. Il principio alla base del software `FFTW` è quello di *ottenere le massime prestazioni su una qualsiasi architettura e per un qualsiasi problema* (portabilità delle prestazioni) *attraverso una combinazione dinamica di un prefissato insieme di algoritmi elementari.* Ciascuna routine del pacchetto software `FFTW` è una composizione di componenti elementari, detti `codelets`; ciascun `codelet` rappresenta l'implementazione di un algoritmo FFT di base, come ad esempio, il calcolo della DFT per $N = 2, 3, 5, 7, \ldots$, l'algoritmo di Cooley e Tukey, l'algoritmo di Gentleman e Sande, etc.. Il software `FFTW` contiene `codelets` creati automaticamente, in fase di installazione della libreria, da un compilatore (*genfft*) ed utilizzati da un programma (detto `plan`) che manda in esecuzione i `codelets` necessari per il calcolo della DFT richiesta dall'utente. Il programma `plan` viene creato dal `planner`, il cui ruolo è quello di *prendere in input il problema fornito dall'utente e scegliere, attraverso tecniche automatiche e dinamiche, la combinazione migliore e più efficiente per risolvere quel problema*. Tale combinazione viene scelta in base alle prestazioni che i *codelets* raggiungono nell'ambiente di calcolo considerato. La scelta dei *codelets* da utilizzare viene fatta percorrendo l'albero ottenuto dalla fattorizzazione del parametro N.

L'aspetto innovativo della `FFTW` risiede proprio nella presenza dei moduli software, il `plan` e il `planner`, con cui interagisce l'utente. L'obiettivo è coniugare portabilità ed efficienza utilizzando tecniche di generazione automatica di codice (ovvero software che produce a sua volta del software). Tale approccio caratterizza la metodologia `AEOS` *(Automated Empirical Optimization of Software)* alla base dello sviluppo automatico di software efficiente e ottimizzato sulle architetture avanzate.

Nel seguito si descrive in breve un frammento di codice scritto in C, che utilizza codelets di `FFTW`.

[30] [10]

```
#include <fftw3.h>
...
{ fftw_complex *in, *out;
fftw_plan p; /* p contiene le specifiche del problema
/* allocazione delle variabili di input e di output */
in = fftw_malloc(sizeof(fftw_complex)*n);
out =fftw_malloc(sizeof(fftw_complex)*n);
/* Generazione del plan */
p= fftw_plan_dft_1d(n,in,out,FFTW_FORWARD,FFTW_ESTIMATE);
...
/* Esecuzione del plan (executor) */
fftw_execute(p);
...
/* deallocazioni della memoria occupata */
fftw_destroy_plan(p);
fftw_free(in);
fftw_free(out);}
```

Il `plan` definisce l'ordine con il quale devono essere eseguiti i *codelets*. Il plan ha una struttura ad albero corrispondente alla specifica fattorizzazione del parametro N, e può essere memorizzato per successivi impieghi.
Le istruzioni contenute nel `plan` vengono eseguite da un opportuno programma detto `executor`. L'`executor` è la componente software della libreria `FFTW` che si occupa del calcolo effettivo della DFT implementando le istruzioni del plan. L'utente deve solo selezionare il tipo di `plan` per il calcolo di FFT m-dimensionali. In particolare `FFTW` mette a disposizione:

`fftw_plan_dft_1d()`	DFT monodimensionale
`fftw_plan_dft_2d()`	DFT bidimensionale
`fftw_plan_dft_3d()`	DFT tridimensionale
`fftw_plan_dft()`	DFT m-dimensionale

Infine l'header file:

```
#include <fftw3.h>
```

contiene informazioni necessarie ad una corretta compilazione, e utilizza, tra l'altro, routine per l'allocazione ed il rilascio della memoria:

```
fftw_malloc(...)
fftw_free(...)
fftw_destroy_plan(...)
```

5.6 MATLAB e la Trasformata discreta di Fourier

In ambiente MATLAB le routine per la trasformata di Fourier sono collezionate nella directory datafun. Esse sono:

```
fft         - Calcola la Trasformata discreta di Fourier (DFT).
fft2        - Calcola la DFT bidimensionale.
fftn        - Calcola la DFT N-dimensionale.
ifft        - Calcola la DFT inversa (IDFT).
ifft2       - Calcola la IDFT bidimensionale.
ifftn       - Calcola la IDFT N-dimensionale.
fftshift    - Esegue uno shift delle componenti di una DFT.
ifftshift   - Funzione inversa di fftshift.
```

Ciascuna delle routine dedicate al calcolo della DFT (`fft, fft2, fftn, ifft, ifft2, ifftn`) si basa sulla libreria **FFTW**. Inoltre MATLAB mette a disposizione dell'utente anche la funzione `fftw`, che costituisce un'interfaccia alla libreria FFTW.

Dal *prompt* dei comandi è possibile richiamare le funzioni con le relative opzioni; ad esempio con:

```
>> Y = fft(X)
>> Y = fft(X,n)
```

- se X è una matrice, `fft` fornisce la Trasformata di Fourier di ciascuna colonna;
- se X è un array multidimensionale, `fft` lavora lungo la prima dimensione diversa da uno.

Inoltre, con:

```
>> Y = fft(X,[],dim)
>> Y = fft(X,n,dim)
```

si applica l'operatore DFT lungo la dimensione *dim*. Analogamente, con:

```
>> y = ifft(X)
>> y = ifft(X,n)
```

si calcola la *DFT inversa* del vettore X.

♣ **Esempio 5.26.** [31] Si consideri una funzione $x = x(t)$ composta da due funzioni sinusoidali, una di frequenza 50 Hz ed ampiezza 0.7 e l'altra di frequenza 120 Hz ed ampiezza 1; si assuma, inoltre, che la funzione x sia affetta da rumore casuale, $\eta(t)$, distribuito secondo distribuzione normale di media zero (Fig. 5.16).

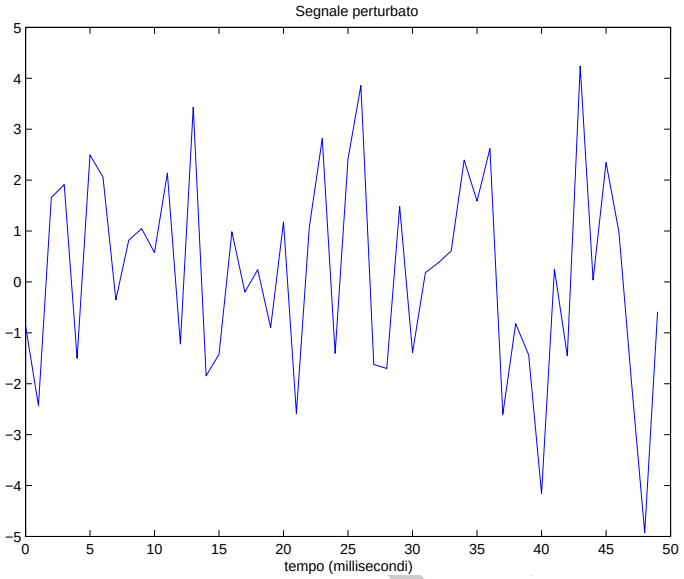

Figura 5.16: Grafico della funzione $y(t) = 0.7\sin(2\pi 50t) + \sin(2\pi 120t) + \eta(t)$.

```
>>Fs = 1000;      % Campionamento delle frequenze
>>T = 1/Fs;       % Campionamento del dominio del tempo
>>L = 1000;       % Lunghezza del segnale
>>t = (0:L-1)*T;  % Vettore del tempo
>>% Somma di una sinusoide a 50 Hz  ed una sinusoide a 120 Hz
>>x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
>>y = x + 2*randn(size(t));  % Aggiunta di rumore
>>plot(Fs*t(1:50),y(1:50))
>>title('Segnale perturbato')
>>xlabel('tempo (millisecondi)')
```

La funzione `randn(N)` genera una matrice di numeri casuali, $N \times N$, appartenenti ad una distribuzione normale di media 0 e varianza 1.
La trasformazione nel dominio delle frequenze è realizzata attraverso il calcolo della DFT del segnale y, mediante la funzione che implementa l'algoritmo FFT:

[31] `http://www.mathworks.com/`

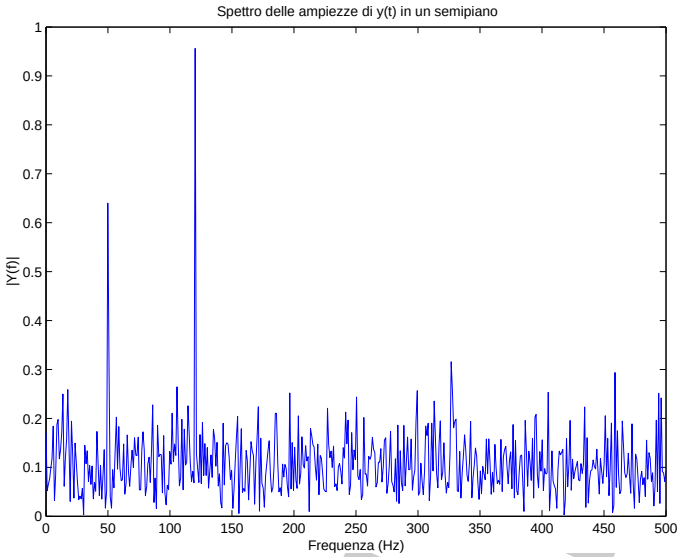

Figura 5.17: Frequenze ed ampiezze della funzione $y(t)$.

```
>>NFFT = 2^nextpow2(L);
>>Y = fft(y,NFFT)/L;
>>%costruzione di un vettore di frequenze equispaziate in [0,Fs/2];
>>f=(Fs/2)*linspace(0,1,NFFT/2);
>>plot(f,2*abs(Y(1:NFFT/2))); %Grafico dello spettro delle ampiezze
>>title('Spettro delle ampiezze di y(t) in un semipiano')
>>xlabel('Frequenza (Hz)')
>>ylabel('|Y(f)|')
```

Poiché la routine `fft` risulta *più veloce* per potenze di due, si calcola il primo p tale che $2^p \geq |L|$ ($2^{10} = 1024$) e si assegna `NFFT` come dimensione alla function `fft` [32].
Il motivo principale per cui le ampiezze non risultano esattamente 0.7 e 1 (Fig.5.17) è la presenza del rumore. Ripetendo le istruzioni (incluso il calcolo del vettore y) si otterranno approssimazioni diverse di 0.7 e di 1. Un ulteriore motivo è la lunghezza finita del segnale; incrementando il valore di L si otterranno approssimazioni mediamente più accurate.

♣

5.6.1 Problemi da risolvere con le routine di MATLAB

Problema 1 Assegnato un vettore x di dimensione $n \geq 1000$, ad esempio con la funzione `MATLAB rand`, $x = rand(n, 1)$, calcolare la DFT di x:

$$y = \texttt{fft}(x)$$

[32] Si osserva che $\texttt{fft}(X, N)$ calcola una DFT di un vettore X di lunghezza N, inserendo zeri se X ha meno di N componenti e troncando se ne ha di più.

Applicare al vettore y la funzione `ifft`:

$$xinv = \texttt{ifft}(\texttt{fft}(x))$$

e stimare l'ordine di grandezza dell'errore:

$$E = \frac{\|x - xinv\|}{\|x\|}$$

Problema 2 Prodotto di due polinomi

Siano

$$a = (1, 5, 17, 0, 0) = (a_0, a_1, ...a_{r+s})$$

e

$$b = (11, 6, -4, 0, 0) = (b_0, b_1, ...b_{r+s})$$

i vettori dei coefficienti, ordinati secondo le potenze crescenti, di due polinomi di grado r e s rispettivamente. A partire dai vettori:

```
>>A=(23,-11.2082+14.7476i,2.082-13.229i,2.2082+13.229i,-11.2082-14.7476i)
>>B=(13,16.0902+3.3552i,4.9098+7.3309i,4.9098-7.3309i,16.0902-3.3552i)
```

con $A = DFT[a]$ e $B = DFT[b]$,

1. calcolare

   ```
   >> ifft(A.*B)=ifft(fft(a).*fft(b))
   ```

2. verificare che il vettore ottenuto coincide con il risultato fornito dalla funzione `conv` di `MATLAB` che, applicata ad a e b, ne calcola il prodotto di convoluzione:

   ```
   >> conv(a,b)
   ```

3. Quante DFT occorrono per il calcolo dei coefficienti del prodotto di due polinomi?
4. Confrontare la complessità di tempo richiesta dal calcolo dei coefficienti di un polinomio prodotto mediante DFT con il numero di operazioni necessarie (asintoticamente) per il calcolo diretto dei coefficienti dello stesso polinomio.

Problema 3 Prodotto matrice circolante per vettore

Si consideri la matrice circolante C generata dal vettore $v = (5, 2, 7, 9, 4, 1, 2, 3)$:

$$C = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 & 1 & 2 & 3 \\ 3 & 5 & 2 & 7 & 9 & 4 & 1 & 2 \\ 2 & 3 & 5 & 2 & 7 & 9 & 4 & 1 \\ 1 & 2 & 3 & 5 & 2 & 7 & 9 & 4 \\ 4 & 1 & 2 & 3 & 5 & 2 & 7 & 9 \\ 9 & 4 & 1 & 2 & 3 & 5 & 2 & 7 \\ 7 & 9 & 4 & 1 & 2 & 3 & 5 & 2 \\ 2 & 7 & 9 & 4 & 1 & 2 & 3 & 5 \end{bmatrix}$$

ed il vettore $x = (1, 2, 3, 4, 5, 6, 7, 8)$. Utilizzare la DFT per il calcolo del prodotto matrice per vettore

$$C \cdot x' = y'$$

1. Calcolare, utilizzando `MATLAB` , il prodotto $y' = Cx'$. Verificare che

   ```
   >>  fft(c).*fft(x)=fft(y)
   ```

   ovvero che attraverso il calcolo di 3 DFT si perviene al vettore soluzione y:

   ```
   >> y=ifft(fft(c).*fft(x))
   ```

2. Confrontare la complessità di tempo richiesta dal calcolo delle 3 DFT con quella del prodotto matrice vettore eseguito righe per colonne.

5.7 Esercizi

5.7.1 Quesiti

Quesito 1 L'algoritmo FFT (Fast Fourier transform) calcola sia la trasformata discreta di Fourier che la sua inversa con la stessa complessità di tempo?

Quesito 2 Fornire due applicazioni della Trasformata Discreta di Fourier (DFT).

Quesito 3 La DFT di un vettore di lunghezza n è legata all'interpolazione trigonometrica mediante un insieme di n funzioni trigonometriche di base.

1. Perché il calcolo della DFT non richiede la risoluzione di un sistema lineare per il calcolo dei coefficienti delle funzioni di base?
2. Qual è il caso peggiore, dal punto di vista della complessità computazionale, per il calcolo della DFT? Per quale valore di n si verifica?
3. Qual è il caso migliore, dal punto di vista della complessità computazionale, per il calcolo della DFT? Per quale valore di n si verifica?
4. Spiegare il motivo della differenza tra le complessità di tempo nei due casi.

Quesito 4 Perché si utilizza l'algoritmo FFT per calcolare il prodotto di convoluzione di due vettori?

5.7.2 Esercizi numerici

Esercizio 1 Verificare, attraverso opportuni esempi, che l'operatore DFT è lineare, ovvero verificare che, se f, $g \in \mathbb{C}^N$ sono vettori di numeri complessi di lunghezza N e se $\alpha, \beta \in \Re$, si ha:

$$DFT[\alpha f + \beta g] = \alpha DFT[f] + \beta DFT[g]$$

al variare dei vettori complessi f e g e degli scalari α e β.

Esercizio 2 Verificare, attraverso opportuni esempi, che, se $f = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{C}^N$ è un vettore di numeri complessi di lunghezza N e se

$$f_{sim} = (f_0, f_{N-1}, f_{N-2}, ..., f_1)$$

è il suo vettore simmetrico, si ha:

$$\frac{1}{N} DFT[DFT[f]] = f_{sim}.$$

Esercizio 3 Verificare, attraverso opportuni esempi, le seguenti proprietà dell'operatore DFT:

1. Se $f \in \Re^N$ è un vettore di numeri reali, allora $F = DFT[f]$ è un vettore hermitiano simmetrico, ovvero tale che le sue componenti, a meno della prima, soddisfino la condizione:

   $$F_k = \bar{F}_{N-k} \quad k = 1, ..., N/2$$

   avendo indicato con $\bar{F}_{N-k}$ il numero complesso coniugato di F_{N-k}.
2. Se $f \in \mathbb{C}^N$ è un vettore hermitiano simmetrico, ovvero $f_k = \bar{f}_{N-k}$, allora $F = DFT[f]$ è un vettore di numeri reali.
3. Se g è un vettore le cui componenti sono numeri immaginari puri (parte reale nulla) allora $G = DFT[g]$ è un vettore hermitiano antisimmetrico, ovvero:

   $$G_k = -\bar{G}_{N-k} \quad k = 1, ..., N/2.$$

4. Se g è un vettore hermitiano antisimmetrico, ovvero $g_k = \bar{g}_{N-k}$, allora $G = DFT[g]$ è un vettore le cui componenti sono numeri immaginari.

Si considerino, ad esempio, i vettori

$$f = (1, 2, 3, 4, 5, 6) \quad \text{e} \quad g = (i, -2i, 5i, -4i, 3i, 6i).$$

Si calcolino $DFT[f]$ e $DFT[g]$ e si effettuino opportune considerazioni sui risultati.

Esercizio 4 Per un fissato vettore, x, perché la prima componente della sua DFT, y_0, è sempre uguale alla somma delle componenti di x?

Esercizio 5 La matrice di Fourier F_n, definita da: $\{F_n\}_{mk} = w^{mk}$ è simmetrica. Per quali valori di n, se esistono, F_n è Hermitiana?

Esercizio 6 Se y è la DFT di un vettore reale, x, di lunghezza n, dove n è una potenza di due, provare che y_0 e $y_{n/2}$ devono essere reali.

Esercizio 7 Se $y = DFT(x)$, dimostrare che

$$x = \frac{1}{n} \overline{DFT(\overline{y})}$$

dove per $\overline{y}$ si intende il vettore le cui componenti sono i complessi coniugati di ciascuna delle componenti di y.

5.7.3 Problemi da risolvere con il calcolatore

Problema 1 Assegnate due matrici circolanti C_1 e C_2, progettare ed implementare una procedura per il calcolo del prodotto righe per colonne di C_1 per C_2 utilizzando la FFT.
Discutere la riduzione della complessità computazionale.

Problema 2 Descrivere lo schema ad albero in base al quale si esprime il calcolo della FFT mixed-radix nel caso $N = 15$.

Problema 3 Assegnato il vettore di numeri complessi:

$$h = (36, -4-9.6569i, -4-4i, -4-1.6569i-4, -4+1.6569i, -4+4i, -4+9.6569i)$$

si calcoli, utilizzando la libreria **fftw3**, la DFT $u := fft(h)$ del vettore h.
(a) Quale proprietà ha il vettore u?
(b) Indicato con $g = 2h$ il vettore che si ottiene da h moltiplicando le sue componenti per due, calcolare il vettore $v := fft(g)$ DFT del vettore g.
(c) Individuare il legame che sussiste tra u e v ed il calcolo della FFT mixed-radix nel caso $N = 15$.

Problema 4 Assegnato il vettore di numeri reali:

$$h = (1, 2, 3, 4, 5, 6, 7, 8)$$

si calcoli, utilizzando la libreria **fftw3**, la DFT $u := fft(h)$ del vettore h.
(a) Quale proprietà ha il vettore u?
(b) Calcolare il vettore $v := fft(u)$ DFT del vettore u.
(c) Individuare il legame che sussiste tra h e v.

Problema 5 Assegnati i vettori di numeri reali:

$$u = (-1, 3, 4, 8)$$

$$v = (11, -4, 7, 9)$$

utilizzando una opportuna routine della libreria **fftw3**
(a) si calcolino i vettori $r = fft(u)$ ed $s = fft(v)$.
(b) Sia $G = (37, -1+18i, 5, -1-18i)$ si calcoli il vettore $g := ifft(G)$ trasformata inversa di G.
(c) Individuare il legame che sussiste tra g ed i vettori u e v.

Problema 6 Assegnato il vettore di numeri reali:

$$u = (42, 32, 4, 8)$$

utilizzando una opportuna routine della libreria **fftw3**
(a) si calcoli il vettore $r = fft(u)$.
(b) Si illustri la proprietà di cui gode il vettore r.
(c) Descrivere lo schema ad albero in base al quale si esprime il calcolo della FFT mixed-radix nel caso $N = 20$.

Problema 7 Utilizzando i moduli di **FFTPACK**[33] scrivere delle routine che eseguano le seguenti operazioni:

1. assegnato un vettore x di dimensione n, calcolare la FFT di x (verificare la correttezza dei calcoli eseguendo una IFFT del risultato ottenuto e stimando l'ordine di grandezza dell'errore relativo tra x e $IFFT(FFT(x))$);
2. assegnata una matrice quadrata A di dimensione n calcolare la FFT bidimensionale di A (verificare la correttezza dei calcoli eseguendo una IFFT del risultato ottenuto e stimando l'ordine di grandezza dell'errore relativo tra A e $IFFT(FFT(A))$);
3. assegnata una matrice circolante A di dimensione n ed un vettore b di dimensione n eseguire il prodotto matrice vettore.

Problema 8 Ripetere gli stessi esercizi proposti utilizzando la libreria **FFTW**.

Problema 9

1. Per ciascun valore di m, $m = 1, \ldots, 5$, calcolare la DFT di un vettore $x_k = cos(mk\pi), k = 0, \ldots, 7$. Discutere i risultati.
2. Tracciare un grafico delle due funzioni $cos(\pi t)$ e $cos(3t)$ sull'intervallo $0 \leq t \leq 7$. Calcolare la DFT di ciascun vettore $x_k = cos(\pi k)$ e $x_k = cos(3k), k = 0, \ldots, 7$, e confrontare i risultati. Spiegare perché le DFT possono essere così differenti sebbene le funzioni siano così simili.

Problema 10 Gauss analizzò l'orbita dell'asteroide Pallas basata sull'osservazione dei dati:

θ	0	30	60	90	120	150
x	408	89	-66	10	338	807
θ	180	210	240	270	300	330
x	1238	1511	1583	1462	1183	804

dove θ rappresenta l'ascensione, in gradi, e x la declinazione, in minuti.

1. Rappresentare i dati mediante la funzione:

$$f(\theta) = a_0 + \sum_{k=1}^{5}[a_k cos(2\pi k\theta/360) + b_k sin(2\pi k\theta/360)] + a_6 cos(2\pi 6\theta/360),$$
$$\theta \in [0, 400]$$

2. Tracciare il grafico dei dati e della funzione calcolata al punto precedente.
3. Utilizzare una routine che implementi l'algoritmo FFT per calcolare la DFT y del vettore x.

[33] Utilizzare le routine **dfftf**,**dfftb** e **dffti** di FFTPACK.

4. Che relazione si può individuare tra parte reale e coefficiente dell'immaginario di y ed i parametri a_k e b_k calcolati al primo punto? (*Suggerimento*: Potrebbe essere necessario scalare rispetto alla lunghezza del vettore o alla sua radice quadrata, in base alla particolare routine FFT che si utilizza).

Problema 11 Sia x un vettore di numeri casuali di lunghezza n, ad esempio $n = 8$. Utilizzare una routine FFT per calcolare la DFT di x. A questo punto, realizzare uno shift delle componenti di x in modo circolare, ad esempio uno shift *ciclico* (o *end-around*) di *un solo posto* verso destra e calcolare la DFT del vettore shiftato e confrontarla con la DFT del vettore iniziale. Considerare, dunque, il modulo delle componenti di ciascuna delle due trasformate (tale vettore è detto "spettro delle ampiezze") e confrontarli. Provare ad eseguire lo shift di un numero maggiore di componenti, verso destra, e confrontare i vettori trasformate discrete di Fourier, con e senza shift, ed i loro relativi spettri delle ampiezze. Quale conclusione si può trarre?

Problema 12 Sia x il vettore $x_k = 1$, $k = 0, \ldots, n-1$, dove n *non è* una potenza di due. Sia $\hat{x}$ lo stesso vettore prolungato con componenti nulle, in modo da rendere la sua dimensione una potenza di due (i.e., $\hat{x}_k = 1$, $k = 0, \ldots, n-1$ e $\hat{x}_k = 0$, $k = n, \ldots, m-1$, dove m è la più piccola potenza di due maggiore di n). Ponendo $n = 5$ e $m = 8$, utilizzare una routine *FFT mixed-radix* per calcolare la DFT sia di x che di $\hat{x}$ e confrontare i risultati. Le trasformate discrete di Fourier coincidono? Cosa si può concludere sull'inserimento di zeri al fine di rendere la dimensione del vettore una potenza di due? Ripetere l'esercizio con altri valori n e di m per determinare se le considerazioni sui risultati sono coerenti.
Ripetere l'esercizio con altri vettori x (i.e., vettori con componenti non costanti o non periodiche) e confrontare le trasformate discrete di Fourier risultanti.

Problema 13 Usando una routine FFT mixed-radix misurare il tempo necessario per calcolare la DFT di un vettore di lunghezza n, per ciascun valore intero $n = 1, 2, 3, \ldots, 1024$. Tracciare il grafico del tempo di esecuzione in funzione di n, usando la scala logaritmica per l'asse delle ordinate. Noto il numero di operazioni eseguite al secondo, dal calcolatore in cui si implementa la routine, verificare che i limiti superiore ed inferiore dei risultati siano in accordo con i valori teorici della complessità di tempo, compresi tra $\mathcal{O}(n\, log_2 n)$ e $\mathcal{O}(n^2)$.

Problema 14 Utilizzare una routine per il calcolo di tutti gli autovalori della matrice *scalata* $(1/\sqrt{n})F_n$, $n = 1, 2, 3, \ldots, 16$. Quanti sono gli autovalori distinti?

Problema 15 Dimostrare empiricamente che la matrice di Fourier F_n diagonalizza una matrice circolante. A tal fine, generare una matrice circolante random C di ordine n e poi calcolare $(1/n)F_n C F_n^H$, con F_n^H matrice *trasposta coniugata* di F_n. La matrice ottenuta dovrebbe risultare diagonale. Cosa comporta questo risultato, riguardo gli autovalori di C? Provare, ad esempio, con $n = 8$.

Problema 16 Calcolare la DFT utilizzando routine standard che effettuano il prodotto di una matrice per un vettore, mediante matrici di Vandermonde. Confrontare le *prestazioni* con quelle di una routine FFT standard, eseguendo test su vettori di lunghezza potenza di due e su vettori che non abbiano lunghezza potenza di due. In particolare provare ad utilizzare, come dimensione, numeri primi *abbastanza grandi*.

Problema 17 Implementare una routine per il calcolo del prodotto di convoluzione tra due vettori. Usare una routine FFT per trasformare i vettori, calcolare il prodotto puntuale delle trasformate discrete di Fourier e, poi, trasformarle nel dominio originale attraverso la trasformazione inversa.

Bibliografia

[1] Bailey D. H. - *Multiprecision Translation and Execution of Fortran Programs* - ACM Trans. on Mathematical Software, vol. 19, no. 3, pp. 288-319 (1993).

[2] Briggs W., Henson V. E. - *The DFT, An Owner Manual for the Discrete Fourier Transform* - Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[3] Brigham E. - *The Fast Fourier Transform and its Applications* - Prentice Hall Signal Processing Series, 1988.

[4] Cooley J. W., Tukey J. W. - *An Algorithm for the Machine Calculation of Complex Fourier Series* - Mathematics of Computation, vol.19, no. 90, pp. 297-301 (1965).

[5] Danielson G. C., Lanczos C. - *Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids* - J.Franklin Inst., vol. 233, pp. 365-380 e pp. 435-452 (1942).

[6] Davis P. J., Rabinowitz P. - *Methods of Numerical Integration*, Academic Press, New York, 1975

[7] Edelman A., McCorquodale P., Toledo S. - *The Future Fast Fourier Transform?* - SIAM J. Sci. Comp., Vol. 20, No. 3, pp. 1094-1114 (1999).

[8] *FFTPACK* - `http://www.netlib.org/fftpack/`.

[9] *FFTW* - `http://www.fftw.org/`.

[10] Frigo M. and Johnson S. G. - *FFTW: An Adaptive Software Architecture for the FFT* - ICASSP conference proceedings, vol. 3, pp. 1381-1384 (1998).

[11] Gauss C. F. - *Theoria interpolationis methodo nova tractata* - Gottingen, 1866.

[12] Gentleman W. M., Sande G. - *Fast Fourier transforms: for fun and profit* - AFIPS Joint Computer Conferences Proceedings, vol. 29, pp. 563-578 (1966).

[13] Giunta G., Murli A. - *Algorithm 649. A Package for Computing Trigonometric Fourier Coefficients Based on Lyness's Algorithm* - ACM Transaction on Mathematical Software, vol. 13, no. 1, pp.97-107 (1987).

[14] Goertzel G. - *An algorithm for the evaluation of finite trigonometric series* - American Mathematical Monthly, vol.65, no.1, pp.34-35 (1958).

[15] Goldstine H. H. - *A History of Numerical Analysis from the 16th through the 19th century* - Springer-Verlag, 1977.

[16] Heideman M. T., Johnson D. H., Burrus C. S. - *Gauss and the history of the fast Fourier transform* - Archive for History of Exact Sciences, vol. 34, no. 3, pp. 265-277 (1985).

[17] Higham N. - *Accuracy and Stability of Numerical Algorithms* - SIAM, Philadelphia, II ed., 2002.

[18] Maddalena M. R., Murli A. - *Su Alcuni Algoritmi Utilizzabili per il Calcolo dell'Integrale di Fourier (Premesse per la Costruzione di un Software)* - Quaderni I.A.C., Roma, Serie III, n. 101 (1979).

[19] Morris L. R. - *Digital Signal Processing Software* - Toronto, Canada: DSPSW, Inc., 1982, 1983.

[20] *MPFUN* - `http://www.netlib.org/mpfun/`.

[21] Murli A. - *Alcune Formule per il Calcolo Numerico della Trasformata di Fourier* - Calcolo, vol. IV, fasc. IV (1967).

[22] Murli A. - *Il Calcolo Numerico degli Integrali Trigonometrici*, Calcolo, vol. V, suppl. no. 1 (1968).

[23] *NAPACK* - `http://www.netlib.org/napack/`.

[24] Rader C. M. - *Discrete Fourier transforms when the number of data points is prime* - Proceedings of the IEEE, vol. 56, No. 6., pp. 1107-1108 (1968).

[25] *SCILIB* - `http://www.netlib.org/scilib/`.

[26] Shannon C. E., Weaver W. - *The Mathematical Theory of Communication* - University of Illynois Press, 1949.

[27] Singleton R. C. - *A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage* - IEEE Trans. on Audio and Electroacoustics, vol. 15, issue 2, pp. 91-98 (1967).

[28] Van Loan C. - *Computational Frameworks for the Fast Fourier Transform* - SIAM, 1992.

[29] Walker J. - *Fast Fourier Transforms* - CRC-Press, 2nd edition, 1996.

Capitolo 6

Sul condizionamento dell'interpolazione polinomiale di Lagrange

6.1 Un confronto tra le formule per la costruzione del polinomio interpolante di Lagrange

Nel **§3.2** del **Capitolo 3**, **Parte prima**, abbiamo illustrato alcune rappresentazioni del polinomio interpolante di Lagrange. In particolare:

1. la rappresentazione *standard*,
2. la formula di Lagrange,
3. la formula di Newton,

laddove la rappresentazione *standard* fa riferimento alla base dei monomi, di uno spazio di polinomi. Dal momento che ciascuna esprime lo stesso ed unico polinomio interpolante in un *modo diverso*, è naturale domandarsi quando usare una rappresentazione e quando un'altra. A tal fine valutiamo l'*efficacia computazionale* di ciascuna rappresentazione, nell'ambito della costruzione e valutazione del polinomio interpolante di Lagrange, confrontandole cioè in base ai criteri di condizionamento, stabilità ed efficienza.

6.1.1 Condizionamento, stabilità ed efficienza per il problema di interpolazione di Lagrange

Sia $\mathcal{P}$ il problema di interpolazione di Lagrange. I dati di $\mathcal{P}$ sono i nodi $(x_i)_{i=1,\ldots,n}$ ed i valori $(y_i)_{i=1,\ldots,n}$ corrispondenti ai nodi, mentre la soluzione è la costruzione e la valutazione dell'unico polinomio interpolante $p \in \Pi_m$, $m = n-1$. Pertanto, il problema

$\mathcal{P}$ si può descrivere come segue:

$$\mathcal{P} : (x_i, y_i)_{i=1,\ldots,n} \in \Re^{n\times n} \longrightarrow p \in \Pi_{n-1} \tag{6.1}$$

Poiché un polinomio in Π_{n-1} è individuato unicamente dai suoi coefficienti in una fissata base di Π_{n-1}, cioè:

$$p(x) = \sum_{i=1}^{n} c_i b_i(x) \qquad (b_i \in \Pi_{n-1}),$$

il problema $\mathcal{P}$ può essere decomposto nei due sottoproblemi:

1. $\mathcal{P}_1$: dati i nodi $(x_i)_{i=1,\ldots,n}$ ed i valori $(y_i)_{i=1,\ldots,n}$, calcolo dei coefficienti $(c_i)_{i=1,\ldots,n}$ di p nella base $\{b_i\}_{i=1,\ldots,n}$,
2. $\mathcal{P}_2$: dato il vettore $(c_i)_{i=1,\ldots,n}$ dei coefficienti di p nella base $\{b_i\}_{i=1,\ldots,n}$, costruzione del polinomio $p(x) = \sum_{i=1}^{n} c_i b_i(x)$, ovvero valutazione di p, per ogni fissato valore x.

Cioè:

$$\mathcal{P} : \overbrace{(x_i, y_i)_{i=1,\ldots,n}}^{\mathcal{P}_1} \longrightarrow \overbrace{(c_i)_{i=1,\ldots,n} \longrightarrow p(x) = \sum_{i=1,\ldots,n} c_i b_i(x)}^{\mathcal{P}_2}$$

Al fine di studiare condizionamento, stabilità ed efficienza del problema $\mathcal{P}$ e quindi dei problemi $\mathcal{P}_1$ e $\mathcal{P}_2$, formuliamo tali problemi in termini matriciali.

Come visto nel **Capitolo 3**, **Parte prima**, le varie rappresentazioni del polinomio interpolante di Lagrange (la rappresentazione standard, la formula di Lagrange e la formula di Newton), si possono esprimere tutte nella forma:

$$p(x) = \sum_{i=1}^{n} c_i b_i(x)$$

dove:

$$b_i(x) = \begin{cases} x^{i-1} & \text{rappresentazione nella base standard} \\ l_i(x) & \text{formula di Lagrange} \\ \Pi_{j=1}^{i-1}(x - x_j) & \text{formula di Newton} \end{cases}$$

e:

$$y_i = p(x_i) = \sum_{j=1}^{n} b_j(x_i) c_j \qquad i = 1, \ldots, n \tag{6.2}$$

Posto:

$$a_{ij} = b_j(x_i) \quad i = 1, \ldots, n; \quad j = 1, \ldots, n\,,$$

la (6.2) si puó riscrivere come:

$$y_i = \sum_{j=1}^{n} a_{ij} c_j \quad i = 1, \ldots, n$$

cioè, introdotti i vettori:

$$c = [c_1, c_2, \ldots, c_n]^T \qquad \text{e} \qquad y = [y_1, y_2, \ldots, y_n]^T$$

la (6.2) diviene:

$$A \cdot \underline{c} = \underline{y} \tag{6.3}$$

con

$$A = (a_{ij})_{i=1,\ldots,n;j=1,\ldots,n} \tag{6.4}$$

Il problema $\mathcal{P}_1$ equivale quindi alla risoluzione del sistema di equazioni lineari (6.3).

Sia M l'operatore che al vettore di componenti $\{c_i\}_{i=1,\ldots,n} \in \Re^n$ associa il polinomio in Π_{n-1} che ha come coefficienti, in una fissata base $\{b_i\}_{i=1,\ldots,n}$ di Π_{n-1}, le componenti di tale vettore:

$$M : \{c_i\}_{i=1,\ldots,n} \in \Re^n \longrightarrow p = \sum_{i=1}^{n} c_i b_i \in \Pi_{n-1}$$

Posto $z = (z_1, \ldots, z_m) \in \Re^m$, con $m \geq 1$, sia, inoltre:

$$W : p \in \Pi_{n-1} \longrightarrow (p(z_1), \ldots, p(z_m)) \in \Re^m;$$

il problema $\mathcal{P}_2$ si può descrivere attraverso l'operatore che, fissato $z = (z_1, \ldots, z_m) \in \Re^m$, agisce come:

$$W \circ M : \{c_i\}_{i=1,\ldots,n} \in \Re^n \overset{M}{\longrightarrow} p \in \Pi_{n-1} \overset{W}{\longrightarrow} \{p(z_i)\}_{i=1,\ldots,m} \in \Re^m.$$

che, in forma matriciale si rappresenta come [1]:

$$\underbrace{\begin{bmatrix} b_1(z_1) & \ldots & b_n(z_1) \\ \ldots & \ldots & \ldots \\ b_1(z_m) & \ldots & b_n(z_m) \end{bmatrix}}_{T} \cdot \underbrace{\begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}}_{\underline{c}} = \underbrace{\begin{bmatrix} p(z_1) \\ \vdots \\ p(z_m) \end{bmatrix}}_{\underline{v}}$$

ed è, dunque, ricondotto ad un'operazione di prodotto tra una matrice ed un vettore, del tipo:

$$T \cdot \underline{c} = \underline{v}.$$

[1] Si osservi che, se $\underline{z} = \underline{x}$, la matrice T coincide con la matrice A del problema $\mathcal{P}_1$; inoltre, in questa ipotesi, il problema $\mathcal{P}_2$ si rappresenta ancora, in termini matriciali, come $A \cdot \underline{c} = \underline{y}$, ma interpretato come problema *diretto*, in cui noti A ed il vettore $\underline{c}$, $\underline{y}$ è calcolato come prodotto tra una matrice ed un vettore. D'altra parte questo si può considerare un caso degenere, in quanto, in un problema di interpolazione, i nodi ed i valori corrispondenti si suppongono dati, per cui non ha senso richiedere la valutazione del polinomio in corrispondenza degli x_i, $i = 1, \ldots, n$, essendo, naturalmente, $p(x_i) = y_i$, $i = 1, \ldots, n$.

Nei paragrafi successivi studieremo il condizionamento, la stabilità e l'efficienza di $\mathcal{P}_1$ e $\mathcal{P}_2$, rispettivamente per la formula di Lagrange, di Newton e per la rappresentazione nella base standard.

6.1.2 Formula di Lagrange

Condizionamento della formula di Lagrange

Condizionamento del problema $\mathcal{P}_1$

Per costruzione si ha:

$$a_{ij} = l_j(x_i) = \delta_{ji} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad i, j = 1, \ldots, n$$

cioè la matrice A nella (6.4) è la matrice identica $A \equiv I$. In questo caso il problema $\mathcal{P}_1$ è *perfettamente* condizionato ($\mu_A(A) = \mu_R(A) = 1$).

Condizionamento del problema $\mathcal{P}_2$

Si ha:

Proposizione 6.1.1. *L'indice di condizionamento assoluto del problema $\mathcal{P}_2$ è:*

$$\mu_A(\mathcal{P}_2) = \Lambda_n$$

mentre quello relativo è:

$$\mu_R(\mathcal{P}_2) = \frac{\Lambda_n \cdot \|y\|}{\|v\|}$$

dove Λ_n indica la costante di Lebesgue, $\underline{y} = \underline{c}$, con $(c_i)_{i=1,\ldots,n}$ vettore dei coefficienti di p, $\underline{v} = (p(z_1), \ldots, p(z_m))$ vettore delle valutazioni di p.

Dimostrazione Rappresentiamo $\mathcal{P}_2$ in forma matriciale, come

$$T \cdot \underline{c} = \underline{v},$$

con, in particolare,

$$T = \begin{bmatrix} l_1(z_1) & \ldots & l_n(z_1) \\ \ldots & \ldots & \ldots \\ l_1(z_m) & \ldots & l_n(z_m) \end{bmatrix}$$

e con

$$\underline{c} = \underline{y} \qquad \text{e} \qquad \underline{v} = (p(z_1), \ldots, p(z_m))$$

In tal modo

$$v_j = p(z_j) = \sum_{i=1}^{n} y_i l_i(z_j), \qquad j = 1, \ldots, m.$$

Inoltre,

$$\begin{aligned}\|T\|_\infty &= \max_{1\le i\le m}\sum_{j=1}^{n}|b_j(z_i)| = \\ &= \max_{1\le i\le m}\sum_{j=1}^{n}|l_j(z_i)|,\end{aligned}$$

$$\|c\|_\infty = \max_{1\le i\le n}|c_i| = \max_{1\le i\le n}|y_i| = \|y\|_\infty$$

e

$$\|v\|_\infty = \max_{1\le i\le m}|p(z_i)| = \max_{1\le i\le m}\left|\sum_{j=1}^{n} y_j l_j(z_i)\right| \le \max_{1\le i\le m}\sum_{j=1}^{n}|y_j||l_j(z_i)|.$$

Poniamo

$$\lambda(z_i) = \sum_{j=1}^{n}|l_j(z_i)|, \quad \forall i$$

e:

$$\|\lambda(z)\|_\infty = \Lambda_n, \qquad z = (z_1, \ldots, z_m)$$

si ha, allora:

$$\|T\|_\infty = \Lambda_n$$

Quindi

$$\mu_A(\mathcal{P}_2) = \Lambda_n$$

e

$$\mu_R(\mathcal{P}_2) = \frac{\|T\|\|c\|}{\|v\|} = \frac{\Lambda_n \cdot \|y\|}{\|v\|}$$

■

L'indice di condizionamento assoluto della costruzione e valutazione del polinomio interpolante di Lagrange è, quindi:

$$\mu_A(\mathcal{P}) = \mu_A(\mathcal{P}_1)\mu_A(\mathcal{P}_2) = \|A^{-1}\| \cdot \|T\| = \|I\| \cdot \Lambda_n = \Lambda_n,$$

mentre quello relativo è:

$$\mu_R(\mathcal{P}) = \mu_R(\mathcal{P}_1)\mu_R(\mathcal{P}_2) = \|I\|\|I^{-1}\| \cdot \frac{\Lambda_n \cdot \|y\|}{\|v\|} = \frac{\Lambda_n \cdot \|y\|}{\|v\|}.$$

Si dimostra che[2]:

$$\Lambda_n > \frac{2}{\pi}\log n + c \quad \text{con} \quad \frac{1}{2} < c < \frac{3}{4}$$

per qualsiasi scelta dei nodi, cioè l'andamento asintotico della costante di Lebesgue è *almeno* logaritmico. Se i nodi sono gli zeri del polinomio di Chebyshev di grado n, allora

$$\Lambda_n \le \frac{2}{\pi}\log n + 1$$

ossia per questa scelta di nodi si ha l'andamento *ottimale* cioè quello *al più* logaritmico.

[2] [5]

Stabilità della formula di Lagrange

Per studiare la stabilità utilizziamo la **backward error analysis** (b.e.a.) e la **forward error analysis** (f.e.a.).

Stabilità del problema $\mathcal{P}_1$

Il problema $\mathcal{P}_1$, espresso dalla formula di Lagrange, è *perfettamente* stabile. Dal momento che i coefficienti della rappresentazione del polinomio in questa base sono dati del problema, non si effettua alcuna operazione floating point.

Stabilità del problema $\mathcal{P}_2$

Consideriamo il problema della valutazione del polinomio interpolante espresso nella formula di Lagrange rappresentato in forma matriciale come

$$T \cdot \underline{c} = \underline{v},$$

con

$$T = \big(l_j(z_i)\big)_{i,j=1,\dots,n}$$

dove:

$$\begin{aligned} l_i(x) = & \prod_{j=1, i\neq j}^{n} \frac{(x-x_j)}{(x_i-x_j)} = \\ & = \frac{(x-x_1)(x-x_2)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_1)(x_i-x_2)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} \end{aligned} \tag{6.5}$$

e con

$$\underline{c} = \underline{y} \qquad \text{e} \qquad \underline{v} = (p(z_1), \dots, p(z_m))$$

Consideriamo le operazioni aritmetiche floating point necessarie per la valutazione di p supponendo, per semplicità, che sia $m = 1$. Applicando la **f.e.a.**, si ha:

Proposizione 6.1.2. *Sia $\widetilde{l_i(x)}$ il valore calcolato di l_i in x. Indichiamo con δ_k l'errore introdotto durante l'esecuzione in $\mathcal{F}$ di una qualsiasi delle operazione aritmetiche floating point definita tra k operandi. Si ha:*

$$\widetilde{l_i(x)} = l_i(x)(1 + \delta_{4n-5}) \qquad\qquad |\delta_{4n-5}| \leq \rho_n \cdot u \tag{6.6}$$

dove:

$$\rho_n = (4n-5) \cdot 1.06$$

rappresenta il fattore di amplificazione dell'errore relativo sul valore calcolato di $l_i(x)$, i-mo polinomio fondamentale di Lagrange.

Dimostrazione Durante il calcolo di $\widetilde{l_i(x)}$, applicando la definizione (6.5), si effettuano:

- per il numeratore: $n-2$ moltiplicazioni e $n-1$ sottrazioni;
- per il denominatore: $n-2$ moltiplicazioni e $n-1$ sottrazioni;
- 1 divisione.

In totale si effettuano $4n-5$ operazioni floating point [3]. ■

Proposizione 6.1.3. *Indichiamo con* $\widetilde{p(x)}$ *il valore calcolato di* $p(x)$*, in un fissato valore* x*. La* **f.e.a.** *mostra che*

$$|\widetilde{p(x)} - p(x)| \leq \rho'_n \cdot u \cdot p(|x|)$$

dove

$$\rho'_n = (5n-5) \cdot 1.06$$

e

$$p(|x|) = \sum_{i=1}^{n} |y_i||l_i(x)|$$

Dimostrazione Per il calcolo di $p(x)$ teniamo conto, analogamente a quanto fatto per il calcolo dei polinomi fondamentali di Lagrange, solo del numero di operazioni floating point effettuate, supponendo che l'ordine con cui queste operazioni sono eseguite, sia quello stabilito dalla formula stessa. Supponendo, inoltre, che i dati $(x_i, y_i)_{i=1,\ldots,n}$ siano esattamente rappresentabili in $\mathcal{F}$, si ha:

$$\begin{aligned}\widetilde{p(x)} = \quad & fl(\textstyle\sum_{i=1}^{n} y_i l_i(x)) = \\ & = (y_1 \cdot \widetilde{l}_1(x)) \cdot (1+\delta_n) + (y_2 \cdot \widetilde{l}_2(x)) \cdot (1+\delta_{n-1}) + \\ & + \cdots + (y_n \cdot \widetilde{l_n(x)}) \cdot (1+\delta_1)\end{aligned}$$

con:

$$|\delta_i| \leq i \cdot 1.06\, u \quad i = 1, \ldots, n$$

[3] Per semplificare l'analisi della propagazione dell'errore di roundoff durante l'esecuzione di una o più operazioni floating point, è opportuno richiamare alcune notazioni, in particolare si ricorda il risultato [4]:

Lemma 6.1.1. *Assegnato un insieme finito di valori*

$$\{\delta_i\}_{i=1,\ldots,n}$$

ed una costante u*, tali che*

$$|\delta_i| \leq u = \frac{1}{2}\beta^{1-t}, \qquad per \qquad i = 1, \ldots, n \qquad e \qquad nu < 0.1,$$

allora

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \eta_n,$$

dove $|\eta_n| \leq 1.06\, n\, u$.

errore di roundoff introdotto da ciascun prodotto $y_i \cdot l_i(x)$ e dalla somma di tutti gli addendi[4]. Per la (6.6) si ha:

$$\begin{aligned}
\widetilde{p(x)} &= y_1 l_1 \cdot (1+\delta_{4n-5})(1+\delta_n) + y_2 l_2 \cdot (1+\delta_{4n-5})(1+\delta_{n-1}) + \\
&\quad + \cdots + y_n l_n \cdot (1+\delta_{4n-5}) \cdot (1+\delta_1) = \\
&= y_1 l_1 \cdot (1+\delta_{5n-5}) + y_2 l_2 \cdot (1+\delta_{5n-6}) + \cdots + \\
&\quad + y_n l_n \cdot (1+\delta_{4n-4}) = \\
&= \textstyle\sum_{i=1}^{n} y_i l_i (1+\delta_{5n-4-i})
\end{aligned}$$

da cui segue che

$$\widetilde{p(x)} - p(x) = \sum_{i=1}^{n} y_i l_i(x) \delta_{5n-4-i}$$

Quindi

$$|\widetilde{p(x)} - p(x)| \leq \sum_{i=1}^{n} |y_i||l_i(x)||\delta_{5n-4-i}|$$

L'errore di roundoff, δ_{5n-4-i}, ricondotto sul dato y_i, è tale che:

$$|\delta_{5n-4-i}| \leq (5n-4-i) \cdot 1.06 \cdot u \leq (5n-5) \cdot 1.06 \cdot u, \quad \forall i = 1, \ldots, n$$

Dunque, l'analisi della propagazione dell'errore di roundoff in avanti (**f.e.a.**), mostra che

$$|\widetilde{p(x)} - p(x)| \leq \rho'_n \cdot u \cdot p(|x|)$$

avendo posto

$$\rho'_n = (5n-5) \cdot 1.06$$

e

$$p(|x|) = \sum_{i=1}^{n} |y_i||l_i(x)|$$

■

Per quanto riguarda l'errore relativo, si ha:

Proposizione 6.1.4. *Se il problema $\mathcal{P}_2$ è ben condizionato ($\mu_R(\mathcal{P}_2) \approx 1$), l'errore relativo nella soluzione calcolata dipende* **linearmente** *da n:*

$$\frac{|p(x) - \widetilde{p(x)}|}{|p(x)|} \leq \rho'_n \cdot u \cdot \frac{\|y\| \cdot \Lambda_n}{\|v\|} = \rho'_n \cdot u \cdot \mu_R(\mathcal{P}_2)$$

dove:

$$\rho'_n = (5n-5) \cdot 1.06$$

[4]In questa analisi si assume che $\widetilde{p(x)}$ sia calcolato utilizzando l'algoritmo di Horner e quindi come:

$$\widetilde{p(x)} = (((y_1 l_1 + y_2 l_2) + y_3 l_3) + \ldots + y_n l_n)$$

Dimostrazione

$$\begin{aligned}\frac{|p(x)-\widetilde{p(x)}|}{|p(x)|} &\leq \rho_n' \cdot u \cdot \frac{p(|x|)}{|p(x)|} \\ &\leq \rho_n' \cdot u \cdot \frac{\sum_{i=1}^n |y_i| \cdot |l_i(x)|}{|p(x)|} \leq \\ &\leq \rho_n' \cdot u \cdot \|y\|_\infty \frac{\sum_{i=1}^n |l_i(x)|}{|p(x)|} \leq \\ &\leq \rho_n' \cdot u \cdot \frac{\|y\|_\infty \cdot \Lambda_n}{|p(x)|}\end{aligned}$$

ed osservando che, se $m = 1$ e $z \in \Re^1$

$$\|v\|_\infty = |p(z)|$$

segue

$$\frac{|p(x)-\widetilde{p(x)}|}{|p(x)|} \leq \rho_n' \cdot u \cdot \frac{\|y\| \cdot \Lambda_n}{\|v\|} = \rho_n' \cdot u \cdot \mu_R(\mathcal{P}_2)$$

■

Tale risultato stabilisce che l'algoritmo per la valutazione del polinomio interpolante espresso nella formula di Lagrange è **stabile nel senso della f.e.a.**.

Si perviene allo stesso risultato anche applicando la **b.e.a.**. Infatti, posto

$$\widetilde{y_i} = y_i(1 + \delta_{5n-4-i}) \;\; i = 1, \ldots, n \tag{6.7}$$

si ha:

$$\widetilde{p(x)} = \sum_{i=1}^n \widetilde{y_i} l_i(x) \tag{6.8}$$

cioè $\widetilde{p(x)}$ può interpretarsi come il risultato in aritmetica a precisione infinita calcolato a partire da dati perturbati, cioè dai coefficienti $\widetilde{y_i}$. Dunque, la b.e.a. mostra che il valore del polinomio interpolante espresso dalla formula di Lagrange, in corrispondenza di un fissato x, che si suppone rappresentato esattamente nel sistema aritmetico a precisione finita, può essere considerato come il valore in aritmetica a precisione infinita, assunto dal polinomio che differisce da $p(x)$ per una perturbazione sui coefficienti al più di:

$$\max_{i=1,\ldots,n} |y_i - \tilde{y}_i| = \max_{i=1,\ldots,n} |\delta_{5n-4-i}| = |\delta_{5n-5}| \leq (5n-5) \cdot 1.06 \cdot u = \rho_n' \cdot u$$

una quantità che dipende **linearmente** da n. In altre parole l'algoritmo per la valutazione del polinomio interpolante espresso nella formula di Lagrange è **stabile nel senso della backward error analysis**.

Efficienza

Calcoliamo la complessità computazionale della costruzione del polinomio interpolante utilizzando la formula di Lagrange. Il calcolo *diretto* dei polinomi fondamentali di Lagrange richiede un totale di $4n-5$ *flops* (di cui $2n-3$ moltiplicazioni e $2n-2$ addizioni) per ciascun polinomio l_i. Essendo n i polinomi l_i, la costruzione del polinomio

$$p(x) = \sum_{i=1}^{n} l_i(x) y_i$$

richiede

$$n((4n-5)+1) = \mathcal{O}(n^2) \quad flops$$

Per risolvere il problema $\mathcal{P}_2$, e, dunque, valutare il polinomio interpolante in un fissato valore x, occorre valutare ciascun polinomio fondamentale di Lagrange in x, effettuando nuovamente tutte le operazioni descritte, con una complessità di tempo asintotica di $\mathcal{O}(n^2)$ *flops*.

6.1.3 Formula di Newton

Condizionamento della formula di Newton

Condizionamento del problema $\mathcal{P}_1$

Per costruzione si ha:

$$a_{ij} = b_j(x_i) = \prod_{k=1}^{j-1} (x_i - x_k) \qquad i, j = 1, \ldots, n$$

È possibile esprimere il calcolo dei coefficienti del polinomio interpolante in forma matriciale.

♣ **Esempio 6.1.** Assegnati i nodi ed i valori corrispondenti

$$(x_i, y_i)_{i=1,\ldots,n},$$

si vogliono determinare i coefficienti del polinomio interpolante di Lagrange, espresso nella formula di Newton.

Al passo $k = 1$, della procedura `diffdivise` nel **§3.2.7** del **Capitolo 3**, **Parte prima**, si calcolano le differenze divise del *primo* ordine

$$a_i = y[x_i, x_{i+1}] = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad i = 1, 2, 3$$

In particolare, il numeratore si può esprimere come

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}}_{N_1} \cdot \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}}_{\underline{y}} = \begin{pmatrix} y_1 \\ y_2 - y_1 \\ y_3 - y_2 \\ y_4 - y_3 \end{pmatrix}$$

mentre, i rapporti

$$\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad i = 1, 2, 3$$

come:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{x_2 - x_1} & 0 & 0 \\ 0 & 0 & \frac{1}{x_3 - x_2} & 0 \\ 0 & 0 & 0 & \frac{1}{x_4 - x_3} \end{pmatrix}}_{D_1^{-1}} \cdot \underbrace{\begin{pmatrix} y_1 \\ y_2 - y_1 \\ y_3 - y_2 \\ y_4 - y_3 \end{pmatrix}}_{N_1 \underline{y}} = \underbrace{\begin{pmatrix} y_1 \\ \frac{y_2 - y_1}{x_2 - x_1} \\ \frac{y_3 - y_2}{x_3 - x_2} \\ \frac{y_4 - y_3}{x_4 - x_3} \end{pmatrix}}_{\underline{y}^{(1)}}$$

dove si osserva, in particolare, che le ultime tre componenti del vettore

$$\underline{y}^{(1)} = D_1^{-1} N_1 \underline{y}$$

sono le differenze divise del primo ordine

$$\begin{aligned} y[x_1, x_2] &= \frac{y_2 - y_1}{x_2 - x_1} \\ y[x_2, x_3] &= \frac{y_3 - y_2}{x_3 - x_2} \\ y[x_3, x_4] &= \frac{y_4 - y_3}{x_4 - x_3} \end{aligned}$$

Al passo $k = 2$, si calcolano le differenze divise del *secondo* ordine

$$a_i = \frac{y[x_{i+1}, x_{i+2}] - y[x_i, x_{i+1}]}{x_{i+2} - x_i} \quad i = 1, 2$$

In particolare, il numeratore si può esprimere come

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}}_{N_2} \cdot \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_2, x_3] \\ y[x_3, x_4] \end{pmatrix}}_{\underline{y}^{(1)}} = \begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_2, x_3] - y[x_1, x_2] \\ y[x_3, x_4] - y[x_2, x_3] \end{pmatrix}$$

mentre, i rapporti

$$\frac{y[x_{i+1}, x_{i+2}] - y[x_i, x_{i+1}]}{x_{i+2} - x_i} \quad i = 1, 2$$

come:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{x_3 - x_1} & 0 \\ 0 & 0 & 0 & \frac{1}{x_4 - x_2} \end{pmatrix}}_{D_2^{-1}} \cdot \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_2, x_3] - y[x_1, x_2] \\ y[x_3, x_4] - y[x_2, x_3] \end{pmatrix}}_{N_2 \underline{y}^{(1)}} = \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ \frac{y[x_2, x_3] - y[x_1, x_2]}{x_3 - x_1} \\ \frac{y[x_3, x_4] - y[x_2, x_3]}{x_4 - x_2} \end{pmatrix}}_{\underline{y}^{(2)}}$$

dove si osserva, in particolare, che le ultime due componenti del vettore

$$\underline{y}^{(2)} = D_2^{-1} N_2 \underline{y}^{(1)}$$

sono le differenze divise del secondo ordine

$$\begin{aligned} y[x_1, x_2, x_3] &= \frac{y[x_2, x_3] - y[x_1, x_2]}{x_3 - x_1} \\ y[x_2, x_3, x_4] &= \frac{y[x_3, x_4] - y[x_2, x_3]}{x_4 - x_2} \end{aligned}$$

Al passo $k = 3$, si calcola la differenza divisa del *terzo* ordine

$$a_i = \frac{y[x_{i+1}, x_{i+2}, x_{i+3}] - y[x_i, x_{i+1}, x_{i+2}]}{x_{i+3} - x_i} \quad i = 1$$

In particolare, il numeratore si può esprimere come

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}}_{N_3} \cdot \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_1, x_2, x_3] \\ y[x_2, x_3, x_4] \end{pmatrix}}_{\underline{y}^{(2)}} = \begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_1, x_2, x_3] \\ y[x_2, x_3, x_4] - y[x_1, x_2, x_3] \end{pmatrix}$$

mentre, il rapporto

$$\frac{y[x_{i+1}, x_{i+2}, x_{i+3}] - y[x_i, x_{i+1}, x_{i+2}]}{x_{i+3} - x_i} \quad i = 1$$

come:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{x_4 - x_1} \end{pmatrix}}_{D_3^{-1}} \cdot \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_1, x_2, x_3] \\ y[x_2, x_3, x_4] - y[x_1, x_2, x_3] \end{pmatrix}}_{N_3 \underline{y}^{(2)}} = \underbrace{\begin{pmatrix} y_1 \\ y[x_1, x_2] \\ y[x_1, x_2, x_3] \\ \frac{y[x_2, x_3, x_4] - y[x_1, x_2, x_3]}{x_4 - x_1} \end{pmatrix}}_{\underline{y}^{(3)}}$$

dove si osserva, in particolare, che l'ultima componente del vettore

$$\underline{y}^{(3)} = D_3^{-1} N_3 \underline{y}^{(2)}$$

è la differenza divisa del terzo ordine

$$y[x_1, x_2, x_3, x_4] = \frac{y[x_2, x_3, x_4] - y[x_1, x_2, x_3]}{x_4 - x_1}$$

Quindi, dopo il passo $k = 3$ si ha:

$$\begin{aligned} \underline{y}^{(1)} &= D_1^{-1} N_1 \underline{y} \\ \underline{y}^{(2)} &= D_2^{-1} N_2 \underline{y}^{(1)} = D_2^{-1} N_2 D_1^{-1} N_1 \underline{y} \\ \underline{y}^{(3)} &= D_3^{-1} N_3 \underline{y}^{(2)} = \underbrace{D_3^{-1} N_3}_{L_3} \underbrace{D_2^{-1} N_2}_{L_2} \underbrace{D_1^{-1} N_1}_{L_1} \underline{y} \end{aligned}$$

avendo posto

$$L_k = D_k^{-1} \cdot N_k, \qquad k = 1, 2, 3$$

Definendo

$$\underline{y}^{(k)}, \quad \text{come il vettore } \underline{y} \text{ costruito al passo } k,$$
$$\underline{y}^{(0)} = \underline{y}$$

si ha, al generico passo k,

$$\underline{y}^{(k)} = L_k \cdot \underline{y}^{(k-1)} \quad k = 1, 2, 3$$

ed, in particolare, al termine dell'algoritmo, si ottiene:

$$\underline{y}^{(3)} = L_3 \cdot L_2 \cdot L_1 \underline{y}^{(0)} \equiv L \cdot \underline{y}$$

♣

Introduciamo le matrici N_k e D_k^{-1}, definite di seguito. La matrice

$$N_k = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 1 & \dots & \dots & 0 \\ 0 & \dots & \dots & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & -1 & 1 & 0 \\ 0 & \dots & \dots & \underbrace{\dots}_{k} & \dots & -1 & 1 \end{pmatrix}$$

ammette una rappresentazione a blocchi come riportata in Figura 6.1. In essa, I_k è la matrice identica di ordine k mentre B_k è la matrice bidiagonale, di dimensione $(n-k) \times (n-k+1)$, definita come:

$$B_k = \begin{pmatrix} -1 & 1 & 0 & \dots & \dots & 0 \\ 0 & -1 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & -1 & 1 & 0 \\ 0 & \dots & \dots & \dots & -1 & 1 \end{pmatrix}$$

La matrice D_k^{-1} è, invece, del tipo

$$D_k^{-1} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & \dots & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 1 & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \frac{1}{x_{k+1}-x_1} & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & \frac{1}{x_{k+2}-x_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & \dots & \frac{1}{x_{n-1}-x_{n-k+1}} & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & \frac{1}{x_n-x_{n-k}} \end{pmatrix}$$

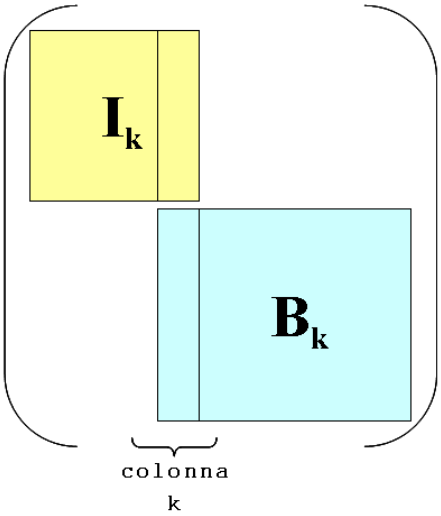

Figura 6.1: Rappresentazione a blocchi della matrice N_k

ed ammette una rappresentazione a blocchi come riportata in Figura 6.2.
In essa, I_k è la matrice identica di ordine k mentre S_{n-k} è la matrice diagonale, di ordine $n-k$

$$S_{n-k} = \begin{pmatrix} \frac{1}{x_{k+1}-x_1} & \dots & \dots & \dots & 0 \\ 0 & \frac{1}{x_{k+2}-x_2} & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \frac{1}{x_{n-1}-x_{n-k+1}} & 0 \\ 0 & \dots & \dots & \dots & \frac{1}{x_n-x_{n-k}} \end{pmatrix}$$

Proposizione 6.1.5. *Sia*

$$L_k = D_k^{-1} N_k \tag{6.9}$$

e

$$\underline{y}^{(0)} = \underline{y},$$

L'indice di condizionamento assoluto del problema $\mathcal{P}_1$ è:

$$\mu_A(\mathcal{P}_1) = \|L\|$$

mentre quello relativo è:

$$\mu_R(\mathcal{P}_1) = \frac{\|L\|\|y\|}{\|c\|}.$$

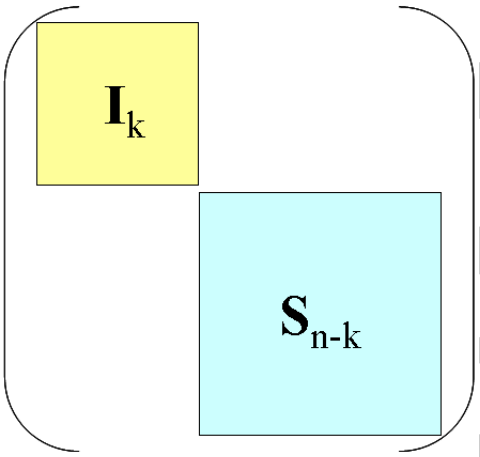

Figura 6.2: Rappresentazione a blocchi della matrice D_k^{-1}

Dimostrazione Ripercorrendo l'algoritmo per il calcolo delle differenze divise, si ha:

$$\begin{aligned} \underline{y}^{(1)} &= L_1 \cdot \underline{y}^{(0)} \\ \underline{y}^{(2)} &= L_2 \cdot \underline{y}^{(1)} = L_2 \cdot L_1 \cdot \underline{y}^{(0)} \\ &\vdots \end{aligned}$$

Al termine dell'algoritmo, dopo n passi, si ottiene

$$\underbrace{\underline{y}^{(n)}}_{c} = \underbrace{L_n \cdot L_{n-1} \cdots L_1}_{L} \underbrace{\underline{y}^{(0)}}_{y}$$

ovvero il calcolo del vettore delle differenze divise equivale al calcolo del vettore $\underline{c}$, come prodotto tra la matrice L ed il vettore $\underline{y}$:

$$\underline{c} = L \cdot \underline{y}.$$

Il problema $\mathcal{P}_1$, nel caso della formula di Newton, può, dunque, essere espresso mediante l'operatore che associa:

$$(x_i, y_i)_{i=1,\ldots,n} \in \Re^{n \times n} \longrightarrow \underline{c} = L_n \cdots L_1 \cdot \underline{y}.$$

L'indice di condizionamento assoluto del problema $\mathcal{P}_1$ coincide, dunque, con quello di un prodotto tra la matrice $L = L_n \cdots L_1$ ed il vettore $\underline{y}$ ed è:

$$\mu_A(\mathcal{P}_1) = \|L\|$$

mentre quello relativo è:

$$\mu_R(\mathcal{P}_1) = \frac{\|L\|\|y\|}{\|c\|}.$$

■

Condizionamento del problema $\mathcal{P}_2$

Proposizione 6.1.6. *si ha:*

$$\mu_A(\mathcal{P}_2) = \|T\|$$

e

$$\mu_R(\mathcal{P}_2) = \frac{\|T\|\|c\|}{\|v\|}.$$

Dimostrazione Relativamente al problema $\mathcal{P}_2$:

$$T \cdot \underline{c} = \underline{v},$$

con

$$T = (t_{ij})_{i,j=1,\ldots,n} = \left(\prod_{k=1}^{i-1}(x_j - x_k)\right)_{i,j=1,\ldots,n},$$

$$\underline{c} = (c_1, \ldots, c_n) = \left(\sum_{j=1}^{n} l_{1j}y_j, \ldots, \sum_{j=1}^{n} l_{nj}y_j\right)$$

e

$$\underline{v} = (p(z_1), \ldots, p(z_m)).$$

si ha:

$$v_k = p(z_k) = \sum_{i=1}^{n} c_i \prod_{j=1}^{i-1}(z_k - x_j), \qquad k = 1, \ldots, m.$$

Inoltre:

$$\|T\|_\infty = \max_{1\le i\le m} \sum\nolimits_{j=1}^{n} |b_j(z_i)| = \max_{1\le i\le m} \sum\nolimits_{j=1}^{n} |z_i - x_j|,$$

$$\|c\|_\infty = \max_{1\le i\le n} |c_i| = \max_{1\le i\le n} \left|\sum_{j=1}^{n} l_{ij}y_j\right| \le \max_{1\le i\le n} \sum_{j=1}^{n} |l_{ij}||y_j|$$

e

$$\|v\|_\infty = \max_{1\le i\le m} |p(z_i)| = \max_{1\le i\le m} \left|\sum_{j=1}^{n} c_j \prod_{k=1}^{j-1}(z_i - x_k)\right| \le$$

$$\le \max_{1\le i\le m} \sum_{j=1}^{n} |c_j| \prod_{k=1}^{j-1} |z_i - x_k| \le$$

$$\le \max_{1\le i\le m} \sum_{j=1}^{n} \left(\sum_{r=1}^{n} |l_{jr}||y_r|\right) \prod_{k=1}^{j-1} |z_i - x_k|.$$

Quindi,

$$\mu_A(\mathcal{P}_2) = \|T\|$$

e

$$\mu_R(\mathcal{P}_2) = \frac{\|T\|\|c\|}{\|v\|}.$$

■

Stabilità della formula di Newton

Stabilità del problema $\mathcal{P}_1$

Utilizziamo la **f.e.a.** per analizzare come si propaga l'errore di roundoff durante il calcolo del vettore $\underline{c}$. Sussiste la seguente:

Proposizione 6.1.7. *Sia*

$$x_1 < x_2 < \ldots < x_n$$

Se il problema $\mathcal{P}_1$ è ben condizionato ($\mu_R(\mathcal{P}_1) \approx 1$), l'errore relativo nella soluzione calcolata dipende **linearmente** *da n, infatti la* **f.e.a.** *conduce al risultato seguente:*

$$\frac{\|c - \bar{c}\|_\infty}{\|c\|_\infty} \leq \delta(n,u) \cdot \mu_R(\mathcal{P}_1)$$

con

$$\delta(n,u) = 3nu + \mathcal{O}(u^2)$$

Tale risultato stabilisce che l'algoritmo per il calcolo delle differenze divise è **stabile nel senso della f.e.a.**.

Stabilità del problema $\mathcal{P}_2$

Per la valutazione del polinomio p espresso nella formula di Newton, consideriamo l'algoritmo di Horner come riportato nella Procedura 6.1.

```
procedure Horner_Newton(in: n, x, a, x̃ ; out: p)
⋮
/# INIZIO ISTRUZIONI:
   p := a(n);
   for i = n − 1, 1 step -1 do
      p := p · (x̃ − x(i)) + a(i);
   endfor
end Horner_Newton
```

Procedura 6.1: Algoritmo di Horner per la valutazione di p, espresso nella formula di Newton, in $\tilde{x}$.

Proposizione 6.1.8. *L'analisi della propagazione dell'errore in avanti (***f.e.a.***), durante l'esecuzione delle operazioni floating point effettuate dall'algoritmo di Horner per la valutazione del polinomio interpolante p in un punto x, mostra che:*

$$|p(x) - \widetilde{p(x)}| \leq \gamma_{2n} \sum_{i=1}^{n} |c_i| \prod_{j=1}^{i-1} |x - x_j|$$

e, dunque

$$|p(x) - \widetilde{p(x)}| \leq \gamma_{2n} \cdot p(|x|),$$

avendo posto

$$p(|x|) = \sum_{i=1}^{n} |c_i| \prod_{j=1}^{i-1} |x - x_j|$$

e

$$\gamma_{2n} = \frac{2nu}{1 - 2nu} = 2nu(1 - 2nu)^{-1} = 2nu + \mathcal{O}(u^2)$$

Dimostrazione Supponiamo che i dati $(x_i, y_i)_{i=1,\ldots,n}$ siano esattamente rappresentabili in $\mathcal{F}$. Consideriamo l'algoritmo di Horner ed indichiamo con $p^{[i]}$ il valore della variabile p aggiornato al passo $i - esimo$ e con $\widetilde{p^{[i]}}$ il valore calcolato di $p^{[i]}$ in x. Si ha:

$$\begin{aligned} \widetilde{p^{[i]}} &= fl(fl(p^{[i+1]} \cdot (\tilde{x} - x_i)) + c_i) \\ &= \Big((p^{[i+1]} \cdot (x - x_i))(1 + \delta) + c_i\Big)(1 + \epsilon) \end{aligned}$$

con $|\delta| \leq u$ e $|\epsilon| \leq u$ errori di roundoff introdotti rispettivamente nell'effettuare le operazioni di prodotto e di somma. Dopo n passi:

$$\begin{aligned} \widetilde{p(x)} = \widetilde{p^{[1]}} = \quad & (x - x_1)(x - x_2)\ldots(x - x_{n-1})c_n(1 + \delta_{2n}) + \\ & + (x - x_1)(x - x_2)\ldots(x - x_{n-2})c_{n-1}(1 + \delta_{2n-2}) + \\ & + \ldots + (x - x_1)c_2(1 + \delta_2) + c_1(1 + \delta_1) \end{aligned}$$

Posto

$$\widetilde{c_k} = c_k(1 + \delta_{2k})$$

si ha

$$\widetilde{p(x)} = \sum_{i=1}^{n} \widetilde{c_i} \prod_{j=1}^{i-1} (x - x_j)$$

e

$$\widetilde{p(x)} - p(x) = \sum_{i=1}^{n} c_i \delta_{2i} \prod_{j=1}^{i-1} (x - x_j)$$

Ciascun δ_{2k}, errore di roundoff introdotto sul coefficiente c_k, nell'effettuare un'operazione floating point a precisione finita tra k operandi, è tale che:

$$|\delta_k| \leq \frac{ku}{1 - ku} := \gamma_k, \quad k = 1, \ldots, 2n$$

Pertanto, l'analisi della propagazione dell'errore effettuata in avanti (**f.e.a.**), mostra che:

$$|p(x) - \widetilde{p(x)}| \leq \gamma_{2n} \sum_{i=1}^{n} |c_i| \prod_{j=1}^{i-1} |x - x_j|$$

e, dunque

$$|p(x) - \widetilde{p(x)}| \leq \gamma_{2n} \cdot p(|x|),$$

avendo posto

$$p(|x|) = \sum_{i=1}^{n} |c_i| \prod_{j=1}^{i-1} |x - x_j|$$

■

Per l'errore relativo si ha:

Proposizione 6.1.9. *Se il problema $\mathcal{P}_2$ è ben condizionato ($\mu_R(\mathcal{P}_2) \approx 1$), l'errore relativo nella soluzione calcolata dipende* **linearmente** *da n:*

$$\begin{aligned} \frac{|p(x) - \widetilde{p(x)}|}{|p(x)|} &\leq \gamma_{2n} \cdot \frac{\|T\| \cdot \|L\| \cdot \|y\|}{\|v\|} \leq \\ &= \gamma_{2n} \cdot \mu_R(\mathcal{P}_2) \end{aligned}$$

dove:

$$\gamma_{2n} = \frac{2nu}{1 - 2nu} = 2nu(1 - 2nu)^{-1} = 2nu + \mathcal{O}(u^2)$$

Dimostrazione Si ha:

$$\begin{aligned} \frac{|p(x) - \widetilde{p(x)}|}{|p(x)|} &\leq \gamma_{2n} \cdot \frac{p(|x|)}{|p(x)|} \leq \\ &\leq \gamma_{2n} \cdot \frac{\sum_{i=1}^{n} |c_i| \prod_{j=1}^{i-1} |x - x_j|}{|p(x)|} \leq \\ &\leq \gamma_{2n} \cdot \|c\|_\infty \frac{\sum_{i=1}^{n} \prod_{j=1}^{i-1} |x - x_j|}{|p(x)|} = \\ &= \gamma_{2n} \cdot \frac{\|c\|_\infty \cdot \|T\|_\infty}{|p(x)|} \end{aligned}$$

ed osservando che, se $m = 1$ e $z \in \Re^1$

$$\|v\|_\infty = |p(z)|$$

segue

$$\begin{aligned} \frac{|p(x) - \widetilde{p(x)}|}{|p(x)|} &\leq \gamma_{2n} \cdot \frac{\|T\| \cdot \|L\| \cdot \|y\|}{\|v\|} \leq \\ &\leq \gamma_{2n} \cdot \frac{\|T\| \cdot \|L_1\| \cdots \|L_n\| \cdot \|y\|}{\|v\|} = \\ &= \gamma_{2n} \cdot \mu_R(\mathcal{P}_2) \end{aligned}$$

dove:

$$\gamma_{2n} = \frac{2nu}{1 - 2nu} = 2nu(1 - 2nu)^{-1} = 2nu + \mathcal{O}(u^2)$$

■

Tale risultato stabilisce che l'algoritmo di Horner per la valutazione del polinomio interpolante espresso nella formula di Newton è **stabile nel senso della forward error analysis**.

Effettuando la b.e.a. si osserva che $\widetilde{p(x)}$ è il risultato in aritmetica a precisione infinita calcolato a partire da dati perturbati, cioè dai coefficienti $\widetilde{c_k}$. Dunque, il valore fornito dall'algoritmo di Horner, in corrispondenza di un fissato x, che si suppone rappresentato esattamente nel sistema aritmetico a precisione finita, può essere considerato come il valore in aritmetica a precisione infinita, assunto dal polinomio che differisce da $p(x)$ per una perturbazione sui coefficienti al più di γ_{2n} che dipende **linearmente** da n. *In altre parole l'algoritmo di Horner è* **stabile nel senso della b.e.a.**.

Efficienza

Il problema $\mathcal{P}_1$, ovvero il calcolo dei coefficienti della formula di Newton, ha la complessità asintotica di tempo di un prodotto tra una matrice ed un vettore, essendo dell'ordine di $\mathcal{O}(n^2)$, mentre il problema $\mathcal{P}_2$, ovvero la valutazione del polinomio, noti i suoi coefficienti, utilizzando l'algoritmo di Horner, richiede $\mathcal{O}(n)$ operazioni floating point.

6.2 L'algoritmo di Björck e Pereyra per la risoluzione di sistemi lineari con matrice di Vandermonde

Nell'effettuare il confronto tra formula di Lagrange e formula di Newton, di proposito non abbiamo tenuto conto della rappresentazione standard perché, essendo la matrice relativa al calcolo dei coefficienti, una matrice di Vandermonde, l'indice di condizionamento calcolato rispetto alla norma matriciale indotta dalla norma del massimo risulta asintoticamente di ordine esponenziale.
A tal proposito, di seguito riportiamo alcune limitazioni inferiori dell'indice di condizionamento, in norma infinito, di una matrice di Vandermonde, indicata con V, di dimensione n, al variare della distribuzione dei nodi x_i, $i = 1, \ldots, n$.

$$\mu_\infty(V) > \begin{cases} n^{n+1} & x_i = 1/(i+1) \\ n^{-1/2}2^{n-2} & x_i \text{ arbitrari} \\ (4\pi)^{-1}\sqrt{2}\cdot 8^n & x_i \in [0,1] \text{ equispaziati} \\ 2^{n-1} & x_i \geq 0 \\ 2^{n/2} & x_i + x_{n+1-i} = 0 \ (i = 1, \ldots, n) \\ \frac{3^{3/4}}{4}(1+\sqrt{2})^n & x_i = \cos\left[(2i-1)\frac{\pi}{2n}\right] \ (i = 1, \ldots, n) \end{cases}$$

con $\mu_\infty(V) = \|V^{-1}\|_\infty \|V\|_\infty$. Dunque, la rappresentazione nella base standard dà origine ad un problema il cui indice di condizionamento cresce **esponenzialmente** con n.

L'algoritmo di Björck e Pereyra[5] trasforma il problema in uno equivalente meglio condizionato, consentendo di calcolare la soluzione del sistema di Vandermonde con la massima accuratezza possibile.

♣ **Esempio 6.2.** Riportiamo l'esempio preso in considerazione da A. Björck e V. Pereyra nel 1970 che fece pensare alla possibilità di ottenere soluzioni accurate da sistemi lineari con matrici di Vandermonde, anche se molto mal condizionate.
Consideriamo il sistema lineare:

$$V \cdot \underline{c} = \underline{b}$$

con:

$$V = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_n \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_0^n & \alpha_1^n & \cdots & \alpha_n^n \end{bmatrix}$$

matrice di Vandermonde, definita a partire dai seguenti valori:

$$\alpha_i = \frac{1}{(i+3)} \quad i = 1, \ldots, n$$

Supponiamo che il termine noto del sistema sia del tipo:

$$b_i = \frac{1}{2^i} \quad i = 1, \ldots, n$$

Si dimostra che la soluzione è:

$$x_i = (-1)^i \begin{pmatrix} n+1 \\ i+1 \end{pmatrix} \left(1 + \frac{i+1}{2}\right)$$

Indicata con $\widetilde{x_i}$ la soluzione calcolata adoperando un calcolatore **IBM 360/50** con 16 cifre significative e precisione di macchina $u = 2.22 \times 10^{-16}$ e denotato con e_n l'errore relativo sulla soluzione, trovarono:

n	5	10	15	20	25	30
e_n/u	4	5	10	54	81	280

dove e_n/u è il rapporto tra l'errore relativo sulla soluzione e la precisione di macchina. Per $n = 30$ tale rapporto è 280 volte la precisione di macchina ovvero l'errore relativo è:

$$e_n = 280 \times 2.220 \times 10^{-16} = 6.210 \times 10^{-14}$$

Considerato che l'indice di condizionamento, in norma del massimo, di V è:

n	5	10	15	20	25	30
$\mu_\infty(V)$	4.69×10^4	4.42×10^{15}	1.39×10^{18}	2.25×10^{22}	3.16×10^{26}	7.17×10^{47}

[5]Nel 1970 A. Björck e V. Pereyra pubblicarono un algoritmo per la risoluzione di un sistema di equazioni lineari con matrice dei coefficienti una matrice di Vandermonde. Tale algoritmo derivava da un rifacimento di un algoritmo analogo pubblicato nel 1967 da uno degli autori.
In questo articolo gli autori notarono, tra l'altro, che: *[...] È come se almeno certi problemi relativi alle matrici di Vandermonde, che tradizionalmente erano stati considerati fortemente mal condizionati, effettivamente potessero essere risolti con una buona accuratezza.[...]* .

il fatto che la soluzione calcolata fosse così accurata risultò molto sorprendente [6]. ♣

Si consideri la matrice di Vandermonde generata dai nodi:

$$x_0, \cdots, x_m$$

con $m = n - 1$. L'algoritmo BP per il calcolo della soluzione del sistema

$$Vc = b$$

con $b = (b_0, \ldots, b_m)$, consiste di due passi:

- **Passo (a)** Si calcolano i coefficienti $(a_k)_{k=0,\ldots,m}$ del polinomio $p \in \Pi_m$ che interpola i punti $P_k = (x_k, b_k), \quad k = 0, \ldots, m$, espresso nella forma di Newton:

$$p(x) = \sum_{k=0}^{m} a_k \prod_{i=0}^{k-1} (x - x_i) \tag{6.10}$$

- **Passo (b)** A partire dai coefficienti $(a_k)_{k=0,\ldots,m}$ del **Passo (a)** si ricavano i coefficienti $(\tilde{c}_k)_{k=0,\ldots,m}$ di p, espresso nella base standard:

$$p(x) = \sum_{k=0}^{m} \tilde{c}_k x^k \tag{6.11}$$

[6] Questo risultato è stato chiarito da N. Higham il quale, effettuando un'analisi della propagazione dell'errore di roundoff nell'algoritmo di A. Björck e V. Pereyra, ha ricavato una maggiorazione dell'errore in termini di componenti della matrice e del vettore dei termini noti (cioè una maggiorazione **componentwise**) in cui il fattore di amplificazione dell'errore, rappresentato dall'indice di condizionamento di Skeel della matrice V, è dell'ordine di 10^2.

procedure BP(**in:** m, x, b ; **out:** $\tilde{c}$)

/# SCOPO: Risolvere un sistema duale di Vandermonde, $V^T \underline{\tilde{c}} = \underline{b}$

/# SPECIFICHE PARAMETRI:

/# PARAMETRI DI INPUT:

var: m	: **intero**	{$m+1$ *è la dimensione del* }
		{*sistema*}
var: $x(m+1)$	: **array di reali**	{*vettore dei nodi che generano*}
		{*la matrice*}
var: $b(m+1)$	: **array di reali**	{*vettore dei termini noti del* }
		{*sistema*}

/# PARAMETRI DI OUTPUT:

var: $\tilde{c}(m+1)$	: **array di reali**	{*soluzione calcolata*}
		{*memorizzata in* $\tilde{c}^{(0)}(m+1)$}

/# VARIABILI LOCALI:

var: j, k	: **interi**	{*contatori*}
var: $a(m+1)$	: **array di reali**	{*vettore dei coefficienti del*}
		{*polinomio, espresso*}
		{*nella formula di Newton*}

/# INIZIO ISTRUZIONI:

% **Passo (a)** calcolo dei coefficienti $a_j = a_j^{(m)}$ $(j = 0, ..., m)$

$a_j^{(0)} = b_j$ $(j = 0, ..., m)$

for $k = 0, m-1$ **do**

$a_j^{(k+1)} = \frac{|a_j^{(k)} - a_{j-1}^{(k)}|}{|x_j - x_{j-k+1}|}$ $(j = m, m-1..., k+1)$

endfor

Procedura 6.2: Algoritmo BP per la risoluzione del sistema $V^T \underline{\tilde{c}} = \underline{b}$ - continua

% **Passo (b)** calcolo dei coefficienti $\tilde{c}_j = \tilde{c}_j^{(0)}$ $(j = 0, ..., m)$
$\tilde{c}_j^{(m)} = a_j^{(m)}$ $(j = 0, ..., m)$
for $k = m-1, 0$ **step** -1 **do**
$\tilde{c}_j^{(k)} = \tilde{c}_j^{(k+1)} - x_k \cdot \tilde{c}_{j+1}^{(k+1)}$ $(j = k, ..., m-1)$
endfor
return $\tilde{c}$
end BP

Procedura 6.2: Algoritmo BP per la risoluzione del sistema $V^T\underline{\tilde{c}} = \underline{b}$ - fine

Analizziamo nel dettaglio ciascun passo, al fine di ottenere una rappresentazione in termini matriciali dell'algoritmo.
Nel passo (a) si calcolano, e si memorizzano nel vettore $(a_0, \cdots, a_m)$, i coefficienti del polinomio espresso in (6.10); tali coefficienti sono le differenze divise relative ai punti P_k assegnati. Analogamente a quanto descritto nel §6.1.3, il calcolo del vettore delle differenze divise si può esprimere mediante lo schema:

$$\begin{aligned} &\underline{a}^{(0)} = \underline{b} \\ &\underline{a}^{(k+1)} = L_k \cdot \underline{a}^{(k)} \quad k = 0, 1, \cdots, m-1 \end{aligned} \tag{6.12}$$

con L_k definita nella (6.9). Dunque, il calcolo dei coefficienti del polinomio, espresso dalla (6.10), si riconduce al prodotto matrice per vettore:

$$\underline{a} = L \cdot \underline{b} \qquad (\underline{a}^{(m)} = \underline{a}). \tag{6.13}$$

Il passo (b) dell'algoritmo BP consiste nella trasformazione dalla base di Newton alla base standard; tale operazione può essere descritta in termini di operazioni matriciali.

♣ **Esempio 6.3.** Consideriamo un polinomio di secondo grado, espresso nella base di Newton, del tipo:

$$p(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

con x_0 e x_1 valori assegnati. Esplicitando le operazioni si ottiene:

$$\begin{aligned} p(x) &= a_0 + a_1 x - a_1 x_0 + a_2 x^2 - a_2 x x_0 - a_2 x x_1 + a_2 x_1 x_2 = \\ &= a_0 - a_1 x_0 + a_2 x_1 x_2 + (a_2(-x_0 - x_1) + a_1)x + a_2 x^2 \end{aligned}$$

ovvero il polinomio espresso nella base standard. In particolare i coefficienti di p nella base standard sono:

$$\tilde{c}_2 = a_2, \qquad \tilde{c}_1 = a_2(-x_0 - x_1) + a_1, \qquad \tilde{c}_0 = a_0 - a_1 x_0 + a_2 x_1 x_2$$

e possono essere espressi mediante la trasformazione:

$$\begin{pmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{pmatrix} = \begin{pmatrix} 1 & -x_0 & x_0x_1 \\ 0 & 1 & -x_0 - x_1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

Posto

$$U = \begin{pmatrix} 1 & -x_0 & x_0x_1 \\ 0 & 1 & -x_0 - x_1 \\ 0 & 0 & 1 \end{pmatrix}$$

si ha

$$\underline{\tilde{c}} = U \cdot \underline{a}.$$

♣

Il vettore $\tilde{c}^{(k)}$, calcolato alla $k-esima$ iterazione del passo (b), si può esprimere mediante lo schema:

$$\begin{aligned} &\underline{\tilde{c}}^{(m)} = \underline{a}^{(m)} \\ &\underline{\tilde{c}}^{(k)} = U_k \cdot \underline{\tilde{c}}^{(k+1)} \quad k = m-1, m-2, \cdots, 0 \end{aligned} \tag{6.14}$$

essendo U_k una matrice bidiagonale del tipo:

$$U_k = \begin{bmatrix} 1 & 0 & & & & & \\ & 1 & \ddots & & & & \\ & & \ddots & 0 & & & \\ & & & 1 & -x_k & & \\ & & & & 1 & \ddots & \\ & & & & & \ddots & -x_k \\ & & & & & & 1 \end{bmatrix}$$

Il passo (b) dell'algoritmo BP può essere, dunque, descritto come:

I iterazione: si calcola

$$\tilde{c}^{(m-1)} = U_{m-1} \cdot \tilde{c}^{(m)}$$

II iterazione: si calcola

$$\tilde{c}^{(m-2)} = U_{m-2} \cdot \tilde{c}^{(m-1)} = U_{m-2}U_{m-1}\tilde{c}^{(m)}$$

...

m-esima iterazione: si calcola

$$\tilde{c}^{(0)} = U_0 \cdot \tilde{c}^{(1)} = U_0 \cdots U_1 \cdots U_{m-1} \cdot \tilde{c}^{(m)}$$

Alla fine del passo (b), ricordando che $\tilde{c} = \tilde{c}^{(0)}$ e ponendo:

$$U := U_0 \cdots U_{m-2} U_{m-1}$$

si calcolano i coefficienti nella base standard di p, espresso nella (6.10), effettuando un prodotto matrice per vettore:

$$\underline{\tilde{c}} = U \cdot \underline{\tilde{c}}^{(m)} = U \cdot \underline{a}. \tag{6.15}$$

Proposizione 6.2.1. *L'algoritmo BP fornisce la soluzione del sistema:*

$$V^T \underline{\tilde{c}} = \underline{b}$$

effettuando due prodotti matrice per vettore

$$\underline{\tilde{c}} = U \cdot (L \cdot b) \tag{6.16}$$

Ovvero, l'algoritmo BP produce una fattorizzazione della matrice di Vandermonde del tipo:

$$V^T = L^{-1} U^{-1} = (UL)^{-1}$$

♣ **Esempio 6.4.** Sia A una matrice di Vandermonde di dimensione 8 tale che:

$$A = (a_{ij}) = j^{2(i-1)}, \ \ i, j = 1, \ldots, 8$$

Si ha:

$$\mu_\infty(A) = 0.168 \times 10^{14} \tag{6.17}$$

Consideriamo come termine noto il vettore:

$$b = [1, 1, 1 \ldots, 1]$$

e perturbiamo il sistema in modo tale che:

$$|\Delta A| = 8u|A|, \ \ |\Delta b| = 8u|b|;$$

la soluzione calcolata $\widetilde{x}$ è tale che:

$$\frac{\|x - \widetilde{x}\|_\infty}{\|x\|_\infty} = 0.240 \times 10^{-11}$$

cioè $\widetilde{x}$ ha circa 12 cifre significative corrette e, dunque, la stima dell'indice di condizionamento, in questo caso, è risultata essere troppo pessimistica.

Osserviamo che gli elementi della matrice A hanno un ordine di grandezza che varia da 1 fino a circa 10^{14}, per cui, nel calcolo dell'indice di condizionamento di A nella (6.17), incide il contributo degli elementi più grandi. In questi casi può essere più significativo misurare la sensibilità del problema relativamente al singolo elemento di A cioè **componente per componente**.

♣

L'analisi del condizionamento di una matrice di Vandermonde, V, quando si usa come misura l'indice di condizionamento $\mu = \|V\|\|V^{-1}\|$, puó portare a risultati poco significativi, soprattutto se i coefficienti della matrice hanno ordini di grandezza molto diversi tra loro. In questi casi può essere più significativa l'informazione fornita da una misura che tiene conto di ciascun elemento della matrice (**componentwise**).

Il risultato seguente dà informazioni sulla propagazione dell'errore relativo componente per componente, nell'algoritmo BP:

Teorema 6.2.1. *Sia $\mathcal{F}$ un sistema aritmetico a precisione finita, con u massima accuratezza relativa e si supponga di risolvere mediante l'algoritmo BP il sistema lineare $V^T \cdot \tilde{c} = b$ con V^T matrice di Vandermonde con dati iniziali esattamente rappresentabili, allora la soluzione calcolata $\hat{c}$ soddisfa la maggiorazione:*

$$\frac{||\tilde{c} - \hat{c}||_\infty}{||\tilde{c}||_\infty} \leq \lambda(m,u) \cdot k_S(U) \cdot k_S(L) \tag{6.18}$$

essendo $\lambda(m,u) = 7mu + \mathcal{O}(u^2)$, $k_s(U) = \||U| \cdot |U^{-1}|\|_\infty$ e $k_s(L) = \||L| \cdot |L^{-1}|\|_\infty$.

Dimostrazione Poiché si dimostra [4] che

$$|\tilde{c} - \hat{c}| \leq \lambda(m,u)|U_0| \cdots |U_{m-1}||L_{m-1}| \cdots |L_0||b|$$

con $\lambda(m,u) = (1+u)^{7m} - 1 = 7m\,u + \mathcal{O}(u^2)$, segue che

$$|\tilde{c} - \hat{c}| \leq \lambda(m,u)|U||L||b|. \tag{6.19}$$

Essendo L una matrice invertibile si ha che:

$$a = Lb \Rightarrow b = L^{-1}a \Rightarrow |b| \leq |L^{-1}||a| \tag{6.20}$$

per cui sostituendo la (6.20) nella (6.19) si ottiene:

$$|\tilde{c} - \hat{c}| \leq \lambda(m,u)|U||L||L^{-1}||a||b|. \tag{6.21}$$

Essendo U una matrice invertibile si ha che:

$$\tilde{c} = Ua \Rightarrow a = U^{-1}\tilde{c} \Rightarrow |a| \leq |U^{-1}||\tilde{c}| \tag{6.22}$$

per cui sostituendo la (6.22) nella (6.21) si ottiene:

$$|\tilde{c} - \hat{c}| \leq \lambda(m,u)|U||U^{-1}||\tilde{c}||L||L^{-1}|. \tag{6.23}$$

Passando alla norma infinito in (6.23) e dividendo per $||\tilde{c}||_\infty$ si ottiene la tesi. ■

Data una matrice A, la quantità $k_s(A) = \||A| \cdot |A^{-1}|\|$ prende il nome di **indice di condizionamento di Skeel** [7].

Se la matrice V è ben condizionata (rispetto all'**indice di condizionamento di Skeel** $k_s(U) \simeq 1$ e $k_s(L) \simeq 1$), l'errore relativo nella soluzione $\hat{c}$ dipende **linearmente** da n. In altri termini l'algoritmo BP è **stabile nel senso della forward error analysis.**

L'indice di condizionamento di Skeel è particolarmente indicato nei casi in cui, per stimare la sensibilità di un problema alle perturbazioni sui dati, l'informazione fornita dall'indice di condizionamento, calcolato utilizzando le classiche norme $(1, 2, \infty, \ldots)$, può far perdere il contributo del singolo dato, soprattutto se i coefficienti hanno ordini di grandezza molto diversi tra loro (matrice *non equilibrata*). Se la matrice non è equilibrata, la limitazione dell'errore relativo sulla soluzione di un problema può essere molto più affidabile se effettuata *componente per componente* (**componentwise**) e quindi se si utilizza come misura del condizionamento del problema l'indice di condizionamento di Skeel. Le matrici di Vandermonde, come quelle dell'esempio 6.4, sono tipologie di matrici non equilibrate.

Si dimostra che

$$k_s(A) = \min\{k_\infty(DA) : D \text{ diagonale}\}$$

[7] Anche per l'indice di condizionamento di Skeel si puó dimostrare un risultato analogo a quello già visto per μ_∞, ovvero che tale quantità rappresenta il fattore di amplificazione dell'errore nei dati sulla soluzione di un sistema lineare. Vale il seguente:

Teorema 6.2.2. [Teorema del condizionamento (*componentwise*)]
Sia $Ax = b$ (con A non singolare) e si consideri il sistema perturbato:

$$(A + \Delta A)(x + \Delta x) = (b + \Delta b)$$

Considerato il più piccolo $\epsilon > 0$, tale che

$$|\delta A| \leq \epsilon |A|$$

e

$$|\delta b| \leq \epsilon |b|,$$

si ha:

$$\frac{\|\Delta x\|}{\|x\|} \leq \epsilon \cdot k_s(A)$$

dove $k_s(A) = \||A^{-1}| \cdot |A^{-1}|\|$, con $\|\cdot\|$ una qualsiasi norma vettoriale, tale che

$$\| \, |\cdot| \, \| = \|\cdot\|.$$

Dimostrazione È analoga a quella del **Teorema 2.4**, **§2.7.1** del **Capitolo 2**, **Parte prima**, pur di sostituire la norma con il valore assoluto. ■

ed inoltre, indicata con D_R la matrice diagonale tale da *equilibrare* le righe di A (cioè le righe di $D_R A$ hanno uguale ordine di grandezza, avendo norma 1 unitaria, cioè $D_R|A|\underline{e} = \underline{e}$) si ha:

$$\frac{k_\infty(A)}{k_\infty(D_R)} \leq k_s(A) \leq k_\infty(A)$$

pertanto, se la matrice A non è *equilibrata per righe*, allora:

$$k_\infty(D_R) >> 1 \quad \text{e} \quad k_s(A) << k_\infty(A)$$

♣ **Esempio 6.5.** Sia ω_C la minima perturbazione relativa sui dati del sistema lineare dell'esempio 6.2, misurata componente per componente, affinché la soluzione, $\tilde{c}$, calcolata dall'algoritmo di eliminazione di Gauss, si possa considerare la soluzione esatta del medesimo problema perturbato,

$$(V + \Delta V)\tilde{c} = b + \Delta b,$$

con [8]

$$\frac{|\Delta V|}{|V|} \leq \omega_C, \quad \frac{|\Delta b|}{|b|} \leq \omega_C.$$

Allora, si ha:

$$\omega_C = 0.110 \times 10^{-11}$$

e

$$k_s(A) \approx 10^2$$

Applicando il Teorema 6.2.2 risulta:

$$\frac{\|c - \widetilde{c}\|_\infty}{\|c\|_\infty} \leq \omega_C \cdot k_s(A) \approx 10^{-9}$$

in accordo con quanto ottenuto nell'esempio 6.2.
In altre parole, l'analisi dell'errore effettuata componente per componente, fornisce una limitazione dell'errore relativo sulla soluzione $\widetilde{c}$, molto più realistica.

♣

La complessità di tempo dell'algoritmo BP, per la risoluzione del problema $\mathcal{P}_1$ del calcolo dei coefficienti del polinomio interpolante di Lagrange, espresso nella rappresentazione standard, si può stimare come somma di quella richiesta ad ogni passo.
Il passo (a) consiste nel *calcolo dei coefficienti nella base di Newton*, ovvero nel prodotto matrice per vettore:

$$\underline{a} = L \cdot \underline{b} \quad \Leftrightarrow \quad \mathcal{O}(n^2) \quad \text{operazioni f.p.}$$

[8] Le disuguaglianze riguardanti la matrice ed il vettore vanno intese componente per componente.

Il passo (b) consiste nel *cambiamento di base* dei coefficienti, dalla base di Newton alla base standard, ovvero nel prodotto matrice per vettore:

$$\underline{\tilde{c}} = U \cdot \underline{a} \quad \Leftrightarrow \quad \mathcal{O}(n^2) \quad \text{operazioni f.p.}$$

Il problema $\mathcal{P}_2$, ovvero la valutazione del polinomio, noti i suoi coefficienti, richiede, utilizzando l'algoritmo di Horner, $\mathcal{O}(n)$ operazioni f.p.

Confrontiamo, infine, le tre rappresentazioni del polinomio interpolante di Lagrange, riferendoci, in particolare, per la rappresentazione standard, alla risoluzione del problema $\mathcal{P}_1^*$ mediante l'algoritmo BP.

formula	*stabilità*		*efficienza*	
	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_1$	$\mathcal{P}_2$
Lagrange	1	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Newton	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
rappresentazione standard	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$

Confronto tra la formula di Lagrange, la formula di Newton e la rappresentazione standard

Si può osservare che, utilizzando l'algoritmo BP, la rappresentazione del polinomio interpolante nella base standard è paragonabile alla formula di Newton, ed inoltre sono entrambe complessivamente più convenienti della formula di Lagrange.

Bibliografia

[1] Björck A., Pereyra V. - *Solution of Vandermonde Systems of Equations* - Mathematics of Computation, vol.24, no.112, pp. 893-903 (1970).

[2] Gautschi W., Inglese G. - *Lower bounds for the condition number of Vandermonde matrices* - Numerische Mathematik, vol.52, no.3, pp.241-250 (1987).

[3] Higham N. - *Accuracy and Stability of numerical algorithms* - SIAM, Philadelphia, 1996.

[4] Higham N. - *Accuracy and Stability of numerical algorithms* - SIAM, Philadelphia, II ed., 2002.

[5] Rivlin T. - *An Introduction to the Approximation of Functions* - Dover, 1969.

[6] Skeel R. - *Scaling for numerical stability in Gaussian elimination* - Journal of the ACM, vol.26, no.3, pp.494-526 (1979).

Capitolo 7

Generazione numerica di funzioni elementari

7.1 Introduzione

Scopo del capitolo è fornire un cenno ad alcune metodologie ed agli strumenti alla base dell'**approssimazione numerica delle funzioni elementari** [1]. Più precisamente, questo problema prende il nome di **generazione di funzioni elementari** perché l'obiettivo è *generare, con la massima accuratezza possibile, i valori di una funzione nota*, per qualsiasi valore dell'argomento nel dominio di definizione della funzione.

In generale, assegnata una funzione f, un **problema di approssimazione** consiste nel costruire una funzione f^* tale che:

1. f^* sia semplice da valutare,
2. su tutto il dominio di definizione lo "scostamento" di f da f^* sia il minimo possibile.

La prima richiesta è una caratteristica tipica della risoluzione computazionale di un qualsiasi problema. Il concetto di semplicità, infatti, è inteso, in generale, come sinonimo di efficienza computazionale. La seconda richiesta stabilisce una misura dell'accuratezza richiesta. In base al tipo di informazioni disponibili si distinguono essenzialmente due tipi di problemi di approssimazione:

1. la funzione f è nota analiticamente, ovvero f è nota in un insieme arbitrario di punti. In questo caso, come misura dello scostamento si utilizza tipicamente la norma del massimo perché riflette la richiesta di una elevata accuratezza in ogni punto **(migliore approssimazione uniforme o approssimazione minimax)**.

[1] Le funzioni elementari (esponenziale, trigonometriche, logaritmo, etc.) sono anche dette, utilizzando la terminologia dei linguaggi di programmazione ad alto livello, *built-in*, in quanto sono le funzioni predefinite dal compilatore.

2. la funzione f è nota in un insieme determinato e limitato di punti e tali valori sono affetti da errore non trascurabile. In questo caso, come misura dello scostamento si utilizza la norma-2 **(migliore approssimazione nel senso dei minimi quadrati)** o la norma-1, perché entrambe rappresentano una richiesta sulla distanza media e sono meno sensibili ad errori inerenti i dati.

La generazione delle funzioni elementari rientra tra i problemi **di migliore approssimazione uniforme** [2].

Definizione 7.1.1. *Il problema della* **migliore approssimazione uniforme di una funzione** *f consiste nel determinare una funzione f^*, in generale espressa come combinazione lineare di funzioni di base*

$$g_1, \ldots, g_r,$$

definita su un intervallo $[a, b]$, tale da minimizzare la distanza da f, misurata utilizzando la norma infinito, cioè tale che

$$f^* = \underset{c}{arg\ min} \|f - g\|_\infty = \underset{c}{arg\ min} \max_{a \le x \le b} |f(x) - g(x)|$$

tra tutte le funzioni del tipo

$$g = \sum_{j=1}^{r} c_j g_j$$

oppure

$$f^* = \underset{c}{arg\ min} \|f - g\|_\infty = \underset{c}{arg\ min} \max_{1 \le i \le m} |f(x_i) - g(x_i)|,$$

se f è nota solo su m punti

$$\{x_1 < x_2 \ldots < x_m\} \quad con \quad x_1 = a \quad e \quad x_m = b.$$

Il supporto teorico di tale problema trae la sua origine dal *Teorema di Weierstrass*. Nel 1885, Weierstrass dimostrò che, assegnata una funzione f, nella sola ipotesi di continuità in un intervallo chiuso e limitato, esiste un polinomio che approssima uniformemente f con una accuratezza arbitrariamente elevata. Il risultato è certamente notevole, soprattutto perché il polinomio è la più semplice funzione da costruire, da valutare, da derivare, da integrare, etc.

Sono stati necessari circa 60 anni per passare dai polinomi di Bernstein (che forniscono una dimostrazione costruttiva del Teorema di Weierstrass) ai polinomi di Chebyshev, riscoperti per le loro proprietà di minimizzare il massimo errore nell'approssimazione di siffatte funzioni, ed ai polinomi razionali, utilizzati di fatto negli algoritmi

[2] Essendo la distanza misurata in norma infinito, si usa anche l'appellativo "minimax" richiedendo il calcolo del minimo di un massimo.

numerici per la generazione delle funzioni elementari perché, rispetto ai polinomi, a parità di costo computazionale, forniscono una migliore accuratezza uniforme: il polinomio razionale ottenuto dal rapporto tra un polinomio di grado n ed uno di grado m, fornisce la stessa approssimazione di un polinomio di grado $n+m$, con un costo computazionale di ordine uguale al massimo tra n e m. In particolare, si utilizzano i polinomi razionali di Chebyshev costruiti utilizzando l'approssimazione di Padé.
In questo capitolo, oltre ad alcuni risultati preliminari che enunceremo solamente, circa l'esistenza, l'unicità e la costruzione di approssimazioni minimax polinomiali e razionali, descriveremo brevemente l' **algoritmo di Remez** alla base della generazione delle funzioni elementari delle librerie matematiche dei compilatori dei linguaggi di programmazione ad alto livello.

È opportuno precisare sin da ora che la soluzione fornita dall'algoritmo di Remez è un'approssimazione numerica della soluzione attesa, calcolata con la massima accuratezza possibile ovvero *corretta a meno della precisione del sistema aritmetico floating point utilizzato*. In altre parole, l'algoritmo di Remez implementa un procedimento iterativo il cui limite è la soluzione del problema di migliore approssimazione uniforme. Il procedimento si arresta quando l'iterata corrente viene calcolata con la massima accuratezza del sistema aritmetico floating point, ovvero dista dalla soluzione attesa al piú quanto la precisione della macchina. In pratica, viene calcolata un'**approssimazione di ordine** u con $u =$ precisione del sistema aritmetico.

Definizione 7.1.2. *Data una funzione g, definita su un intervallo $[a,b]$ e fissato $\epsilon > 0$, la funzione f^+ tale che*

$$\|g - f^+\|_\infty < \epsilon$$

è detta ***approssimazione di ordine*** *ϵ della funzione g.*

7.2 La migliore approssimazione uniforme delle funzioni elementari

Cominciamo a trattare il problema della costruzione del polinomio algebrico P^*, di grado n ([3]), di migliore approssimazione uniforme, ovvero, assumiamo di considerare

[3]In questo capitolo, siamo interessati principalmente alla costruzione del polinomio di migliore approssimazione uniforme, piuttosto che alle problematiche legate alla definizione del polinomio stesso, tra cui la scelta del grado di tale polinomio. Per il momento, ci limitiamo a dire che il grado del polinomio di migliore approssimazione uniforme dipende dalla regolarità della funzione f da approssimare. A tal proposito, i *Teoremi di Jackson* forniscono svariate stime asintotiche dell'errore introdotto dal polinomio di migliore approssimazione uniforme [18]. Ad esempio, per funzioni dotate di derivata continua di ordine qualsiasi, l'errore ha un andamento asintotico di ordine esponenziale con n.

come funzioni di base i monomi, ovvero $g_j(x) = x^{j-1}$, $j = 1, \ldots, n+1$ e quindi

$$P^*(x) = \sum_{j=1}^{n+1} c_j x^{j-1} \quad .$$

Successivamente, descriveremo brevemente come si specializzano tali risultati ai polinomi razionali.

L'esistenza del polinomio di migliore approssimazione uniforme di una funzione continua discende dal teorema seguente, che fornisce un risultato di carattere generale:

Teorema 7.2.1. *Sia V uno spazio normato e W un sottospazio di V a dimensione finita. Per ogni elemento di V, $v \in V$, esiste almeno un elemento di W, $w^* \in W$, che realizza la migliore approssimazione di v:*

$$\|v - w^*\| \leq \|v - w\|, \quad \forall\, w \in W$$

Una condizione sufficiente atta a garantire che tale soluzione sia anche l'unica è che lo spazio S sia *strettamente convesso* rispetto alla norma ivi introdotta. Questa proprietà non è verificata dalla norma infinito sullo spazio delle funzioni continue, pertanto per dimostrare l'unicità di tale soluzione è necessario ricorrere a specifiche caratterizzazioni della migliore approssimazione polinomiale. In particolare, come vedremo, si fa uso dell'esistenza del cosiddetto *insieme delle alternanze* di punti per la funzione errore, cosí definito:

Definizione 7.2.1. *Sia g una funzione continua in un intervallo $[a, b]$. La funzione g è detta "alternare s volte" su $[a, b]$ se esistono $s + 1$ punti*

$$a \leq x_1 < x_2 < \ldots < x_{s+1} \leq b$$

tali che

$$g(x_i) = (-1)^i E, \quad i = 1, \ldots, s+1$$

con $E = \|g\|_\infty$ oppure $E = -\|g\|_\infty$.

L'insieme $\{x_i\}_{i=1,\ldots,s+1}$ è detto "insieme delle alternanze" per la funzione g.

L'idea alla base dell'algoritmo di Remez consiste nel sostituire il problema del calcolo del polinomio di migliore approssimazione uniforme sull'insieme di definizione della funzione da approssimare f con un problema analogo ristretto ad un sottoinsieme finito dell'insieme di definizione della funzione f.

Indichiamo con S il problema sull'intervallo $[a, b]$:

$$S: \text{Calcolare} \quad P^* \in \Pi_n \quad \text{tale} \quad \text{che:} \quad \|P^* - f\|_\infty = \min_{P \in \Pi_n} \|P - f\|_\infty \quad \text{su} \quad [a, b]$$

e con $S^{(m)}$ il problema su un insieme finito $X^{(m)}$ contenuto in $[a, b]$:

$$S^{(m)}: \text{Calcolare} \quad P^{(m)} \in \Pi_n \quad \text{tale} \quad \text{che:} \quad \|P^{(m)} - f\|_\infty = \min_{P \in \Pi_n} \|P - f\|_\infty \quad \text{su} \quad X^{(m)}.$$

Posto:

$$\delta = \|P^* - f\|_\infty, \quad \epsilon_m = \|P^{(m)} - P^*\|_\infty \quad \text{e} \quad \delta_m = \|P^{(m)} - f\|_\infty,$$

poiché

$$\|P^* - f\|_\infty \leq \|P^* - P^{(m)}\|_\infty + \|P^{(m)} - f\|_\infty$$

si ha:

$$\delta \leq \epsilon_m + \delta_m$$

Sostituendo il problema S con il problema $S^{(m)}$, e quindi calcolando $P^{(m)}$ in luogo di P^*, l'approssimazione di f dipende da ϵ_m e δ_m. A tal proposito osserviamo che, una volta calcolato $P^{(m)}$, e quindi una volta ottenuto un certo valore di δ_m, resta determinato anche il valore di ϵ_m anche se, non essendo noto P^*, il valore di ϵ_m non è noto a priori. Naturalmente, si auspica di poter scegliere m in modo da rendere sia δ_m sia ϵ_m più piccoli possibile. Nell'algoritmo di Remez, questo risultato si ottiene facendo crescere opportunamente m, dal momento che, per quanto riguarda ϵ_m, si dimostra che $\epsilon_m \to 0$, per $m \to \infty$ [4], mentre, per quanto riguarda δ_m, per ogni fissato m, per definizione $P^{(m)}$ viene costruito in modo da avere minima distanza da f, cioè δ_m è minimo.

Nell'algoritmo di Remez, l'insieme $X^{(m)}$ è definito in base ai punti in cui la distanza tra la funzione f e il polinomio di migliore approssimazione uniforme assume valore massimo, con segni alterni, come stabilito dal teorema seguente [5].

Teorema 7.2.2. *Sia $f \in C^0([a, b])$ e sia $P^* \in \prod_n$ il polinomio di grado al piú n di migliore approssimazione uniforme di f su X:*

$$P^* = \sum_{j=1}^{n+1} c_j^* x^{j-1} \quad .$$

Affinché P^* realizzi la migliore approssimazione su X, della funzione f, è necessario e sufficiente che la distanza $r = f - P^*$ presenti, su X, almeno $n+2$ **alternanze***: ovvero, devono esistere $n + 2$ punti di X, $x_0 < \cdots < x_{n+1}$ tali che $r(x_i) = -r(x_{i-1}) = \pm\|r\|_\infty$.*

Tale risultato è noto come **Teorema di equioscillazione di Chebyshev**. Utilizzando tale caratterizzazione si dimostra anche che il polinomio di migliore approssimazione uniforme è unico.

[4] [18]

[5] Per la dimostrazione di questo teorema e di quelli seguenti si veda [18].

♣ **Esempio 7.1.** Il polinomio di Taylor di una funzione f, in un intervallo $[a, b]$, non realizza la migliore approssimazione in norma infinito,

$$t(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k,$$

in quanto la differenza

$$|f(x) - t(x)| = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

con $\xi \in [a, b]$, non assume mai, su tutto l'intervallo di approssimazione, valori uguali in modulo e di segno opposto. ♣

Teorema 7.2.3. *Sia $f \in C^0([a, b])$ e sia $P^* \in \prod_n$ il polinomio di grado al piú n di migliore approssimazione uniforme di f su X:*

$$P^* = \sum_{j=1}^{n+1} c_j^* x^{j-1} \quad .$$

Allora P^* è unico.

Si noti che il massimo della funzione $r = f - P^*$ non è noto a priori. Il teorema seguente fornisce l'espressione di tale quantità, fissato un insieme arbitrario di punti.

Teorema 7.2.4. *Sia $X = \{x_0, \ldots, x_{n+1}\}$. Allora*

$$\max_{x \in X} r(x) = |\lambda|$$

dove

$$\lambda = \frac{\sum_{i=0}^{n+1} f(x_i)/\omega'(x_i)}{\sum_{i=0}^{n+1} (-1^i)/\omega'(x_i)}$$

con

$$\omega(x) = (x - x_0) \ldots (x - x_{n+1})$$

Ció premesso, resta da stabilire come individuare l'insieme delle alternanze di punti. Nell'algoritmo di Remez, a partire da un insieme di $m \geq n + 2$ [6] punti scelto arbitrariamente, ad ogni passo si aggiunge un nuovo punto o si scambia con uno dei precedenti finché non sia soddisfatta la condizione di equioscillazione per la funzione r.

[6] Si noti che se $m \leq n + 1$, considerato il polinomio $P^* \in \prod_n$ interpolante gli m punti, sull'insieme $(x_1, \ldots, x_m)$ risulta banalmente

$$0 = \|f - P^*\|_\infty = min_{q \in \prod_n} \|f - q\|$$

ovvero che il polinomio interpolante gli m punti è il polinomio di migliore approssimazione uniforme sull'insieme $(x_1, \ldots, x_m)$. Pertanto, come è stato fatto anche nel caso del problema di migliore approssimazione nel senso dei minimi quadrati (**§3.5.1** del **Capitolo 3**, **Parte prima**) al fine di escludere tale eventualità, si assume $m \geq n + 2$.

7.2.1 L'algoritmo di Remez

Fissati $m \geq n+2$ punti $(x_1, \ldots, x_m)$, consideriamo la matrice

$$A(\mathbf{x}) = \begin{pmatrix} x_1^0 & x_1^1 & \ldots & x_1^n \\ x_2^0 & x_2^1 & \ldots & x_2^n \\ \ldots & \ldots & \ldots & \ldots \\ x_m^0 & x_m^1 & \ldots & x_m^n \end{pmatrix}, \qquad A(\mathbf{x}) \in \Re^m \times \Re^{(n+1)}$$

dove abbiamo indicato con $\mathbf{x}$ il vettore di componenti $(x_1, \ldots, x_m)$.

Posto $\mathbf{b} = (f(x_1), \ldots, f(x_m))$, $\mathbf{c} = (c_1, \ldots, c_{n+1})$, chiamiamo **residuo** la funzione vettoriale di $\Re^{(n+1)} \times \Re^m$ in $\Re^m$:

$$r(\mathbf{c}, \mathbf{x}) = A(\mathbf{x})\mathbf{c} - \mathbf{b} = \sum_{i=1}^{n+1} c_i \mathbf{x}^{i-1} - f(\mathbf{x}).$$

dove $\mathbf{x}^{i-1}$ indica il vettore le cui componenti sono ottenute come potenze *(i-1)*-me delle componenti omologhe di $\mathbf{x}$. Osserviamo che, se fissiamo $\mathbf{x}$, il residuo r dipende solo dal vettore $\mathbf{c}$. In questo caso, il calcolo del vettore $\mathbf{c}$ per cui è minima la norma infinito del residuo, equivale al calcolo della soluzione **nel senso minimax** del sistema lineare $A\mathbf{c}=\mathbf{b}$. Se, invece, fissiamo il vettore $\mathbf{c}$, il residuo dipende solo dal vettore $\mathbf{x}$. In questo caso, il massimo della funzione residuo si ottiene calcolando la derivata prima di f e di $P^{(m)}$, attraverso l'espressione di f e di $P^{(m)}$.

Descriviamo, brevemente, l'algoritmo di Remez:

Dati di input:

- $[a, b] \subset \Re$: insieme di definizione di f;
- $f(x)$, $x \in [a, b]$: funzione da approssimare;
- n: grado del polinomio di migliore approssimazione minimax;
- $m \geq n+2$: numero dei punti iniziali;

Dati di output:

- $\mathbf{c} = (c_1, \ldots, c_{n+1}) \in \Re^{n+1}$, coefficienti del polinomio $P^{(m)} \in \Pi_n$, approssimazione minimax della funzione f.

Algoritmo di Remez:

per $k = 0, 1, 2, \ldots$

1. Si considera l'insieme $X^{(k)} = \{x_1, \ldots, x_{m+k}\}$:

- per $k = 0$, l'insieme $X^{(0)} = \{x_1, \ldots, x_m\}$, con $m \geq n + 2$, si fissa arbitrariamente [7];

2. si calcola il vettore $\mathbf{c}^{(k)} \in \Re^{n+1}$, come soluzione minimax su $X^{(k)}$ del sistema $\mathbf{Ac} = \mathbf{b}$;

3. si valuta il residuo $r = Ac^{(k)} - b$;

4. detto $P^{(k)}$ il polinomio di grado n avente come coefficienti le componenti del vettore $\mathbf{c}^{(k)}$, cioè

$$P^{(k)} = \sum_{i=1}^{n+1} c_i^{(k)} x^{i-1},$$

utilizzando l'espressione di $P^{(k)}$ e di f, si calcola il punto di massimo per il modulo della funzione residuo, rispetto alla variabile x in $X^{(k)}$. Detto $x^{(k)}$ il punto in cui il modulo del residuo assume valore massimo:

$$|r(\mathbf{c}^{(k)}, x^{(k)})| = \max_{x \in X^{(k)}} \{|r(\mathbf{c}^{(k)}, x)|\},$$

si costruisce l'insieme

$$X^{(k+1)} = X^{(k)} \cup \{x^{(k)}\}.$$

- Se sui punti dell'insieme $X^{(k+1)}$ la funzione residuo assume il valore massimo sull'intervallo $[a, b]$ allora l'algoritmo si arresta;

- altrimenti, poiché, per il **Teorema 7.2.2**, i valori $r(\mathbf{c}^{(k)}, x_i^{(k)}),\ \forall i$, sono dello stesso ordine di grandezza e di segno alterno, la funzione $r(\mathbf{c}^{(k)}, x),\ x \in [a, b]$, in ciascun intervallo $(x_{i-1}^{(k)}, x_i^{(k)}),\ i = 2, \ldots, m + k + 1$ ammette uno zero, che indichiamo con z_i. Poniamo, inoltre, $z_0 = a$ e $z_{m+k+2} = b$.
Introduciamo le quantità:

$$\sigma_i = sign\ r(x_i^{(k)}), \quad i = 0, \ldots, m + k + 1$$

e indichiamo con $y_i \in [z_i, z_{i+1}]$, $i = 0, \ldots, m + k + 1$, il punto per cui $\sigma_i r(y_i)$ assume valore massimo nell'intervallo $[z_i, z_{i+1}]$.
Poiché il massimo sull'intervallo $[a, b]$ non è assunto in corrispondenza dei punti y_i, esiste un punto $y \in [a, b]$, diverso dai punti y_i, in corrispondenza del quale il residuo assume il valore massimo:

$$\|r\|_\infty = \|f - P^{(k)}\|_\infty = |f(y) - P^{(k)}(y)| \ > \ |f(y_i) - P^{(k)}(y_i)|, \quad i = 0, \ldots, m + k + 1,$$

[7] In generale, si scelgono come punti iniziali gli zeri del polinomio di Chebyshev trasformati nell'intervallo $[a, b]$ secondo la trasformazione

$$y = \frac{(b - a)x + (a + b)}{2}.$$

Si sostituisce, quindi, uno degli y_i con questo valore y. Il punto y deve sostituire quel valore y_i tale che il nuovo insieme $X^{(k+1)} = \{y_0, \ldots, y_{m+k+1}\}$ sia ancora ordinato e che su di esso la funzione r assuma segni alterni.[8] L'algoritmo continua ritornando al passo 1.

♣ **Esempio 7.2.** Sia $f(x) = x^2$ e $X = [0,1]$. Posto $P^*(x) = c_1 g_1(x)$ con $g_1(x) = 1 + x$, si vuole calcolare il coefficiente c_1 tale che $P^*(x)$ sia l'approssimazione minimax di $f(x)$ su X.

1. **Passo k=0**: Sia $m = 2$ e $X^{(0)} = \{x_1, x_2\} = \{0, 1\}$ e

$$A = \begin{pmatrix} g_1(x_1) \\ g_1(x_2) \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} f(x_1) \\ f(x_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

- Calcolo del vettore $\mathbf{c}^{(0)} \in \Re$, soluzione minimax su $X^{(0)} = \{0, 1\}$ del sistema $A\mathbf{c} = \mathbf{b}$. Tale sistema ammette come soluzione $c_1^{(0)} = 1/3$.

  – Risoluzione nel senso minimax del sistema **Ac=b**.
  La matrice $A \in \Re^{2\times 1}$ ha rango 1, quindi per il **Teorema B.1.1**, sia $J = \{1, 2\}$ e $\lambda = (\lambda_1, \lambda_2) \in \Re^2$ un vettore non nullo per cui:

$$\begin{aligned} A^T\lambda &= \mathbf{0} \\ \lambda_i\theta_i &\geq 0 \qquad i = 1. \end{aligned} \tag{7.1}$$

  Considerato il sistema

$$\begin{aligned} A^T\lambda = \mathbf{0} &\Leftrightarrow \begin{pmatrix} 1 & 2 \end{pmatrix}\lambda = \mathbf{0} \\ &\Leftrightarrow \lambda_1 + 2\lambda_2 = 0 \end{aligned} \tag{7.2}$$

  la soluzione appartiene all'insieme $\{(\lambda_1, -\lambda_1/2) : \lambda_1 \in \Re\}$. Imponendo che $\lambda^T\mathbf{b} \geq 0$, si ha

$$\begin{aligned} \lambda^T\mathbf{b} \geq \mathbf{0} &\Leftrightarrow \begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \lambda_2 \geq 0 \\ &\Leftrightarrow \lambda_1 \leq 0. \end{aligned}$$

  In tal modo, dovendo valere anche la (7.1), si ha $\theta_1 = -1$, $\theta_2 = 1$. Per il **Corollario B.1.1**, se $\mathbf{c}$ risolve $A\mathbf{c} = \mathbf{b}$ in senso minimax in $X^{(0)}$, risolve anche il sistema

$$\begin{cases} b_1 - g_1(x_1)c_1 = \theta_1 h \leftrightarrow 0 - c_1^{(0)} &= -h \\ b_2 - g_1(x_2)c_1 = \theta_2 h \leftrightarrow 1 - 2c_1^{(0)} &= h \end{cases}$$

  che ammette come soluzione $c_1^{(0)} = 1/3$, $h = 1/3$.

[8] Si accenna, brevemente, alla regola per l'aggiunta di y nell'insieme $\{y_i\}$. Se $y \in [z_i, y_i)$, per qualche $i \geq 1$, allora si sostituisce y_{i-1} con y. Se $y \in [a, y_0)$, allora si sostituisce y_n con y_{n-1}, y_{n-1} con y_{n-2}, $\ldots$, y_1 con y_0 ed, infine, y_0 con y. Analogamente se $y \in (y_n, b]$.

- Utilizzando l'espressione simbolica della funzione f e di P^* calcoliamo il valore di x in cui la funzione

$$|r(c_1^{(0)},x)| = |P^{(0)}(x) - f(x)| = |c_1^{(0)}(1+x) - x^2| = |c_1^{(0)} + c_1^{(0)}x - x^2| = \left|\frac{1}{3} + \frac{1}{3}x - x^2\right|,$$

considerata come funzione nella variabile x, assume il massimo. A tale scopo, calcolando la derivata prima del residuo e ponendola uguale a zero, si ottiene:

$$\frac{1}{3} - 2x = 0 \leftrightarrow x = \frac{1}{6} \leftrightarrow x^{(0)} = \underset{x\in X^{(0)}}{arg\ max}|r(c_1^{(0)},x)| = \frac{1}{6}.$$

Risulta inoltre:

$$|\mathbf{r}(\mathbf{c}^{(0)},x^{(0)})| = \left|r\left(\frac{1}{3},\frac{1}{6}\right)\right| = \frac{13}{36}.$$

2. **Secondo passo (k=1):**

- l'algoritmo si ripete su $X^{(1)} = X^{(0)} \cup \{x^{(0)}\} = \left\{0, \frac{1}{6}, 1\right\}$. In particolare, bisogna calcolare il vettore $c^{(1)} \in \Re^2$ soluzione minimax del sistema $\mathbf{Ac} = \mathbf{b}$ dove:

$$A = \begin{pmatrix} g_1(x_1) \\ g_1(x_2) \\ g_1(x_3) \end{pmatrix} = \begin{pmatrix} 1 \\ 7/6 \\ 2 \end{pmatrix} \quad \text{e} \quad \mathbf{b} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{pmatrix} = \begin{pmatrix} 0 \\ 1/36 \\ 1 \end{pmatrix}.$$

Ragionando in maniera analoga al primo passo, si trova $\mathbf{c}^{(1)} = c_1^{(1)} = 37/114$, soluzione minimax del sistema $A\mathbf{c} = \mathbf{b}$ su $X^{(1)}$. Qui di seguito, descriviamo il procedimento:

– la matrice $A \in \Re^{3\times 1}$ ha rango 1, quindi per il **Teorema B.1.1**, determiniamo un sottoinsieme J di $\{1,2,3\}$ costituito da al più 2 interi, ed un vettore non nullo $\lambda = (\lambda_1,\lambda_2,\lambda_3) \in \Re^3$, per cui:

$$\begin{aligned} \lambda_3 &= 0 \\ A^T\lambda &= \mathbf{0} \\ \lambda_i\theta_i &\geq 0 \qquad i = 1,2. \end{aligned} \tag{7.3}$$

Considerato il sistema

$$\begin{aligned} A^T\lambda = \mathbf{0} &\Leftrightarrow \begin{pmatrix} 1 & 7/6 & 2 \end{pmatrix}\lambda = \mathbf{0} \\ &\Leftrightarrow \lambda_1 + 7/6\lambda_2 + 2\lambda_3 = 0 \end{aligned}$$

la soluzione appartiene all'insieme $\{(\lambda_1, -\frac{6}{7}\lambda_1, 0) : \lambda_1 \in \Re\}$. Imponendo che $\lambda^T\mathbf{b} \geq 0$, si ha

$$\begin{aligned} \lambda^T\mathbf{b} \geq \mathbf{0} &\Leftrightarrow \begin{pmatrix} \lambda_1 & -\frac{6}{7}\lambda_1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ \frac{1}{36} \\ 1 \end{pmatrix} = -\frac{1}{42}\lambda_1 \geq 0 \\ &\Leftrightarrow \lambda_1 \leq 0 \end{aligned} \tag{7.4}$$

In tal modo, dovendo valere anche la (7.3), si ha $\theta_1 = -1$, $\theta_2 = 1$ e θ_3 arbitrario. Per il **Corollario B.1.1**, se $\mathbf{c}$ è soluzione del sistema $A\mathbf{c} = \mathbf{b}$ nel senso minimax, risolve lo stesso problema definito ponendo $b_i = f(x_i)$ $i = 1,2$ e considerando, come A, la matrice di dimensione 2×1 con $i-esima$ riga il vettore $[g_1(x_i)]$, $i = 1,2$, ovvero:

$$b_i - \alpha_i^T\mathbf{c} = \hat{\theta}_i h, \qquad i \in \{1,2,3\} \qquad \Leftrightarrow \qquad \begin{cases} -c_1^{(1)} &= -h \\ \frac{1}{36} - \frac{7}{6}c_1^{(1)} &= h \\ 1 - 2c_1^{(1)} &= -h \end{cases} \tag{7.5}$$

Se $J = \{1, 2\}$ si ottiene il sistema costituito dalle prime due equazioni delle (7.5):

$$\begin{cases} & -c_1^{(1)} & = & -h \\ \frac{1}{36} & -\frac{7}{6}c_1^{(1)} & = & h \end{cases}$$

che ammette come soluzione $c_1^{(1)} = 1/78$, $h = 1/78 \approx 0.0128$. Per rendere h minimo su tutto $X^{(1)}$ deve essere

$$|r_i| \leq h, \quad i = 1, 2, 3. \tag{7.6}$$

In tal caso, invece, le condizioni (7.6) non sono tutte verificate essendo

$$|r_i| \leq h, \quad i = 1, 2,$$

ma $|r_3| = \left|1 - 2 \cdot \frac{1}{78}\right| = 0.9744 > h$. Se si assume $J = \{1, 3\}$ si ottiene il sistema

$$\begin{cases} & -c_1^{(1)} & = & -h \\ 1 & -2c_1^{(1)} & = & -h \end{cases}$$

che ammette come soluzione $c_1^{(1)} = 1$, $h = 1$. Anche in questo caso le condizioni (7.6) non sono tutte vere essendo

$$|r_i| \leq h, \quad i = 1, 3,$$

ma $|r_2| = \left|\frac{1}{36} - \frac{7}{6}\right| = 1.1389 > h$. Al contrario, se $J = \{2, 3\}$ si ottiene il sistema

$$\begin{cases} \frac{1}{36} & -\frac{7}{6}c_1^{(1)} & = & h \\ 1 & -2c_1^{(1)} & = & -h \end{cases}$$

che ammette come soluzione $c_1^{(1)} = 37/114$, $h = -20/57 \approx -0.3509$ ed, in tal caso, le condizioni (7.6) sono tutte soddisfatte, essendo $|r_i| \leq h, \quad i = 2, 3$ ed anche

$$|r_1| = \left|0 - \frac{37}{114}\right| = 0.3246 < -h.$$

- Calcolo del valore in cui la funzione

$$|r(c_1^{(1)}, x)| = |P^{(1)}(x) - f(x)| = |c_1^{(1)}(1+x) - x^2| = \left|\frac{37}{114}(1+x) - x^2\right| = \left|\frac{37}{114} + \frac{37}{114}x - x^2\right|,$$

nella variabile x, assume il valore massimo:

$$x^{(1)} = \underset{x \in X}{arg\ max} |r(c_1^{(1)}, x)| = 37/228.$$

Effettuando la derivata prima di r ed eguagliandola a zero, si ha:

$$|\mathbf{r}(\mathbf{c}^{(1)}, x^{(1)})| = \left|r\left(\frac{37}{114}, \frac{37}{228}\right)\right| \simeq 0.35089642967067.$$

- L'algoritmo si ripete su $X^{(2)} = X^{(1)} \cup \{x^{(1)}\} = \left\{0, \frac{37}{228}, \frac{1}{6}, 1\right\} \approx \{0,\ 0.16228\ldots,\ 0.16667\ldots, 1\}$.

♣

♣ **Esempio 7.3.** Sia $f(x) = x^2$ e $X = [0, 2]$. Si ponga $P^* = c_1 g_1(x) + c_2 g_2(x)$ con $g_1(x) = x$ e $g_2(x) = e^x$. A partire da $m = 3$ e $X^{(0)} = \{0, 1, 2\}$, come si vede dalla **Tabella 7.1**, la soluzione $(c_1^{(k)}, c_2^{(k)})$ calcolata dopo $k = 18$ iterazioni è

$$P^{(18)}(x) = 0.18421184\, x + 0.41863683\, e^x.$$

k	**1**	**2**	**3**	**10**	**16**	**17**	**18**
x_1	0	0	0.2347	0.4055	0.406356	0.406356	0.406356
x_2	1	0.4789	0.4789	0.4074	0.406395	0.406382	0.406378
x_3	2	2	2	2	2	2	2
$c_1^{(k)}$	0.3061	-0.1911	0.0674	0.1844	0.18423193	0.18421687	0.18421184
$c_2^{(k)}$	0.4038	0.5224	0.4512	0.4186	0.41863139	0.41863547	0.41863683

Tabella 7.1: Evoluzione dell'algoritmo di Remez per la funzione $f(x) = x^2$ nell'intervallo $[0, 2]$, come nell'esempio 7.3. I coefficienti c_1 e c_2 vengono calcolati con la massima accuratezza possibile (8 cifre significative) dopo 18 iterazioni. Già alla decima iterazione i valori calcolati, di $c_1^{(k)}$ e $c_2^{(k)}$, possono ritenersi corretti alla terza cifra significativa; in effetti, dalla decima iterazione in poi le prime tre cifre significative di $c_1^{(k)}$ e $c_2^{(k)}$ non cambiano.

♣

7.2.2 Studio della convergenza

L'algoritmo di Remez genera una successione di approssimazioni che, sotto opportune ipotesi, converge in maniera *quadratica*, cioè

$$\|P^{(k)} - P^*\|_\infty \leq A\theta^{2k}, \qquad \text{per} \qquad k \to \infty,$$

con A e θ numeri positivi, indipendenti da k ed, in particolare, con $\theta < 1$.
Di seguito si enuncia, senza dimostrazione [9] il teorema relativo alla convergenza lineare; esiste un teorema analogo per la convergenza quadratica. Siano

$$\underline{c}^{(k)} = (c_1^{(k)}, \ldots, c_{n+1}^{(k)})$$

[9] [3]

il vettore dei coefficienti e

$$P^{(k)} = \sum_{i=1}^{n+1} c_i^{(k)} x^{i-1}$$

il polinomio approssimante, costruito al passo k dell'algoritmo di Remez; vale il seguente:

Teorema 7.2.5. *La successione dei $P^{(k)} \in \Pi_n$, generati dall'algoritmo di Remez, converge uniformemente alla migliore approssimazione uniforme, $P^* \in \Pi_n$, secondo la seguente disuguaglianza:*

$$\|P^{(k)} - P^*\|_\infty \leq A\theta^k, \qquad per \qquad k \to \infty,$$

dove $0 < \theta < 1$.

Proposizione 7.2.1. *Posto*

$$E_k = \max_{0 \leq i \leq n+1} |f(x_i^{(k-1)}) - P^{(k)}(x_i^{(k-1)})|,$$

si ha:

$$\forall\, k > 0 \qquad E_k < E_{k+1} \leq \rho_n(f) = \|f - P^*\|_\infty$$

avendo indicato con P^* l'approssimazione minimax costruita all'ultimo passo dell' algoritmo di Remez.

Il procedimento iterativo si arresta quando l'iterata corrente dista dalla soluzione attesa al piú quanto la precisione del sistema aritmetico floating point utilizzato. In altre parole, anche la soluzione numerica fornita dall'algoritmo di Remez rappresenta un'**approssimazione di ordine** u della funzione data.

7.2.3 Esempio di studio: l'approssimazione minimax della funzione esponenziale

Passo 1 Si consideri il problema di approssimare e^x sull'intervallo $(-\infty, \infty)$. Al fine di approssimare la funzione con la stessa accuratezza su tutto l'asse reale, il primo passo, nella costruzione di software numerico alla base della generazione delle funzioni elementari *built-in* dei compilatori di linguaggi di programmazione ad alto livello, prevede che il problema sia ricondotto su un intervallo chiuso e limitato. Poiché, per ogni $x \in (-\infty, \infty)$,

$$e^x = 2^{\log_2 e^x} = 2^{x \log_2 e} \tag{7.7}$$

e

$$x \log_2 e = Y + F \tag{7.8}$$

con Y numero intero e F numero reale tale che $-1 < F < 1$, dalla (7.8) e (7.7) segue:

$$2^{Y+F} = 2^Y \cdot 2^F$$

e, poiché

$$2^F = e^{\log 2^F} = e^{F \log 2}$$

si ha:

$$2^{x \log_2 e} = 2^Y e^{F \log 2}$$

Essendo Y un numero intero, in un sistema aritmetico in base 2 la moltiplicazione di $e^{F \log 2}$ per 2^Y comporta una variazione dell'esponente di $e^{F \log 2}$. In tal modo, il problema della valutazione di e^x, $x \in (-\infty, \infty)$, è ricondotto alla valutazione della funzione esponenziale $e^{F \log 2}$ con $F \log 2 \in (-\log 2, \log 2)$.
Nel caso in cui si desideri ridurre tale argomento in un intervallo di centro l'origine e di ampiezza minore, è sufficiente riscrivere la (7.8) come [10]:

$$x \log_2 e = m + \frac{j}{2^L} + \frac{q}{2^{L+1}}, \quad \frac{q}{2^{L+1}} \in \left(-\frac{1}{2^L}, \frac{1}{2^L}\right), \quad m, j, q, L \text{ numeri interi} \quad (7.9)$$

Poiché

$$\log_2 e = \frac{\log e}{\log 2} = \frac{1}{\log 2},$$

dalla (7.9), si ha:

$$x \log_2 e = x \frac{1}{\log 2} = \left(m + \frac{j}{2^L}\right) + \frac{q}{2^{L+1}} \quad \Leftrightarrow \quad x = \left(2^L m + j\right) \frac{\log 2}{2^L} + r \quad (7.10)$$

con

$$r = \log 2 \cdot \frac{q}{2^{L+1}} \in I = \left[-\frac{\log 2}{2^{L+1}}, \frac{\log 2}{2^{L+1}}\right] \subset \left(-\frac{\log 2}{2^L}, \frac{\log 2}{2^L}\right)$$

Pertanto, segue che:

$$e^x = e^{\left(2^L m + j\right)\frac{\log 2}{2^L}} \cdot e^r = e^{m \cdot \log 2} \cdot e^{\frac{j}{2^L} \log 2} \cdot e^r = 2^m \cdot \left(2^{\frac{j}{2^L}} \cdot e^r\right)$$

e il problema è ricondotto al calcolo di una approssimazione di e^r con $r \in I$ [11].

[10] Ad esempio:

$$123.45 = \underbrace{123}_{Y} + \underbrace{\frac{45}{100}}_{F} \qquad \text{con} \qquad F \in (-1, 1)$$

e:

$$123.45 = \underbrace{123}_{m} + \underbrace{\frac{4}{10}}_{j/10} + \underbrace{\frac{5}{100}}_{q/10^2}$$

dove

$$\frac{5}{100} = \frac{5}{10^2} \in \left(-\frac{1}{10}, \frac{1}{10}\right)$$

[11] La scelta dell'intervallo I, in generale, dipende da considerazioni legate alla rappresentazione floating point dei numeri macchina e dall'errore di roundoff introdotto nella loro rappresentazione. In questa sede, ci limitiamo a evidenziare che tale l'intervallo deve essere quanto piú piccolo possibile e che gli estremi dell'intervallo dipendono dalla base del sistema di numerazione dell'ambiente di calcolo.

Passo 2 Nel secondo passo il valore $f(r) = e^r - 1$ è approssimato con il valore assunto, in r, dal polinomio razionale, p, approssimazione minimax di f, per cui si pone:

$$e^r := p(r) + 1.$$

Passo 3 Valutazione di e^x:

$$e^x = 2^m(2^{j/2^L} + 2^{j/2^L}p(r)). \tag{7.11}$$

In particolare, posto

$$B_j = \log 2^{j/2^L},\ j = 0, 1, 2, \ldots, 2^L - 1,$$

da cui:

$$e^{B_j} = 2^{j/2^L},\ j = 0, 1, 2, \ldots, 2^L - 1,$$

si ha:

$$e^x = 2^m e^{B_j}(1 + p(r)). \tag{7.12}$$

Di seguito riportiamo tre proposizioni, per la cui dimostrazione si rimanda all'**Appendice B.2.1**, in cui si fornisce una stima dell'errore introdotto in ciascun passo, avendo fissato, ad esempio, $L = 5$.

Proposizione 7.2.2. *L'errore di roundoff, commesso nella rappresentazione floating point, R, dell'argomento r, introdotto al passo 1, ammette la seguente maggiorazione:*

$$\boxed{|R - r| < 2^{-34}.}$$

Proposizione 7.2.3. *Per l'errore introdotto al passo 2 vale la stima*

$$\boxed{|e^t - 1 - p(t)| < 2^{-33.2} \quad \textit{per ogni} \quad t \in [-0.010831, 0.010831].}$$

Proposizione 7.2.4. *L' errore di roundoff, δ, introdotto nella valutazione di e^x al passo 3 soddisfa la seguente maggiorazione:*

$$|\delta| \leq \boxed{\begin{array}{ll} 0.5240 \cdot 2^{-24} & \textit{se } j = 0 \textit{ e } r < 0, \\ 0.5120 \cdot 2^{-23} & \textit{se } j = 0 \textit{ e } r \geq 0, \\ 0.5295 \cdot 2^{-23} & \textit{se } j \geq 1. \end{array}}$$

Infine, supponendo che nel calcolo di $2^{\mathtt{m}}$ e $2^{j/32}$ non si introduca errore di roundoff, l'errore globale si stima in base all'approssimazione del termine

$$2^{j/32}(1 + p(r)) = 2^{j/32}e^r.$$

Per la dimostrazione si rimanda all'**Appendice B.2.1**.

Proposizione 7.2.5. *Per l'errore globale introdotto nei passi 1-2-3 nella generazione numerica di $y = e^x$ risulta:*

$$\boxed{|(y - fl(y))/y| < 0.54\, ulp}$$

dove $ulp(x)$ denota la distanza tra i due numeri macchina più vicini a x.

7.2.4 L'approssimazione minimax razionale

Nella generazione numerica delle funzioni matematiche elementari l'algoritmo di Remez si utilizza, in particolare, per costruire approssimazioni minimax di tipo razionale. Infatti, il polinomio razionale

$$R_{m,k} = \frac{p_m}{q_k}$$

ha una distanza massima da f piú piccola di quella ottenuta con il polinomio algebrico di migliore approssimazione uniforme di f di grado m [12]. Tale polinomio viene costruito utilizzando l'approssimazione razionale di Padé mediante i polinomi razionali di Chebyshev. L'approssimazione di Padé fornisce, infatti, stime calcolabili utili alla costruzione del polinomio razionale di migliore approssimazione uniforme mentre i polinomi di Chebyshev rispetto a qualsiasi altra rappresentazione polinomiale hanno minima l'oscillazione massima e per questo sono utilizzati nella maggior parte dei problemi di approssimazione polinomiale [13].

[12] Questo deriva dal fatto che il polinomio razionale di migliore approssimazione unforme minimizza la distanza in norma infinito dalla funzione f tra tutti i polinomi razionali $R_{m,k}$ con $m, k \geq 0$ quindi, in particolare, anche per quelli in corrispondenza di $k = 0$, ovvero i polinomi algebrici di grado al piú m. [17]

[13] [15]

L'algoritmo è applicabile anche nel caso in cui si desidera un'approssimazione del tipo

$$s \cdot R_{mk},$$

dove s è una funzione continua che non si annulla nell'intervallo (a, b). Scegliendo, ad esempio,

$$s(x) = \frac{1}{f(x)}$$

e la funzione da approssimare identicamente uguale ad uno, è possibile calcolare la migliore approssimazione minimax *relativa*, di f, su un intervallo in cui essa non si annulla, essendo

$$1 - \frac{R_{mk}(x)}{f(x)} = \frac{1}{f(x)}[f(x) - R_{mk}(x)].$$

Prima di descrivere cosa sia una approssimazione di Padé, richiamiamo alcuni risultati riguardanti l'esistenza e la caratterizzazione del polinomio razionale di migliore approssimazione uniforme.

Teorema 7.2.6. *Sia f una funzione continua in $[a, b]$. Fissati due interi m e k, esiste ed è unico il polinomio razionale $R_{m,k}$ di migliore approssimazione uniforme.*

Sia

$$r_{mk} = \max_{a \le x \le b} |f(x) - R_{mk}(x)|$$

la distanza massima di f da un polinomio razionale del tipo R_{mk}, in un intervallo $[a, b]$. Il seguente teorema fornisce la caratterizzazione necessaria per la determinazione dell'approssimazione razionale minimax di f, su un intervallo fissato.

Teorema 7.2.7. *Esiste un unico polinomio razionale R^*_{mk}, che minimizza r_{mk}. Inoltre, se:*

$$R^*_{mk}(x) = \frac{P^*(x)}{Q^*(x)} = \frac{\sum_{j=0}^{m-\nu} a_{j+\nu} x^j}{\sum_{j=0}^{k-\mu} b_{j+\mu} x^j}, \tag{7.13}$$

dove

$$0 \le \mu \le k \quad 0 \le \nu \le m \quad a_m, b_k \neq 0$$

e dove P^*/Q^* è irriducibile, allora, se $r^*_{mk} \neq 0$, il numero dei punti consecutivi dell'intervallo $[a, b]$, in corrispondenza dei quali la distanza assume il valore massimo r^*_{mk}, con segni alterni, non è minore di

$$T = \underbrace{m + k}_{N} + 2 - d,$$

dove $d = min(\mu, \nu)$.

In particolare il teorema afferma che, quando $k = 0$, il numero dei punti in cui l'errore assume valore massimo, in modulo, è, almeno, $m + 2$.

L'approssimazione di Padé

Assegnata una funzione $f \in C^\infty([-c, c])$ con $0 < c < 1$ e considerato il polinomio di Taylor $p \in \Pi_n$ relativo a f, ottenuto troncando lo sviluppo in serie di Mc Laurin di f:

$$p(x) = \sum_{i=0}^{n} \frac{f^{(i)}(0)}{i!} x^i,$$

p e f sono uguali in $x = 0$ con tutte le prime n derivate.
Supponiamo di voler determinare il polinomio razionale del tipo

$$R_{mk}(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{j=1}^{k} b_j x^j} \tag{7.14}$$

che soddisfi una condizione analoga. Sia $N = m + k$ l'*indice* di R_{mk}.

Definizione 7.2.2. *Il polinomio R_{mk} è un'approssimazione di Padé, di indice N, di una funzione f analitica in $[a, b]$, se i polinomi P e Q sono tali che la differenza*

$$r(x) = f(x) - R_{mk}(x)$$

abbia quante più derivate nulle in $x = 0$.[14]

Per quanto riguarda la quantitá $r = f - R_{mk}$, vale il seguente[15]:

Teorema 7.2.8. *Se*

$$R_{mk}(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{j=1}^{k} b_j x^j}$$

è una approssimazione di Padé di indice $N = m + k$ della funzione

$$f(x) = \sum_{j=0}^{\infty} c_j x^j,$$

allora:

$$\left| f(x) - \frac{P(x)}{Q(x)} \right| = \left| \frac{\sum_{j=N+1}^{\infty} d_j x^j}{Q(x)} \right|, \quad con \quad d_j = \sum_{i=0}^{k} c_{j-i} b_i$$

[14] Si osserva che, per $k = 0$, l'approssimazione coincide con lo sviluppo in serie di Mc Laurin di f.
[15] [3]

Tra tutte le approssimazioni di Padé, quelle con $m = k$ oppure $m = k + 1$, restituiscono la *minima distanza massima*, tra tutte quelle di indice N e sono, dunque, di tipo *minimax*.

♣ **Esempio 7.4.** Determiniamo l'approssimazione di Padé (7.14) di e^x. Indicando con $c_i,\ i = 0, 1, 2, \ldots$ i coefficienti dello sviluppo in serie di McLaurin della funzione esponenziale:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, \tag{7.15}$$

la funzione errore diventa:

$$\left| e^x - \frac{P_m(x)}{Q_k(x)} \right| = \left| \frac{\left(\sum_{j=0}^{\infty} c_j x^j\right)\left(\sum_{j=0}^{k} b_j x^j\right) - \sum_{j=0}^{m} a_j x^j}{1 + \sum_{j=1}^{k} b_j x^j} \right|$$

Affinché le prime N derivate di $e^x - R_{mk}(x)$ siano uguali a zero in $x = 0$, deve risultare:

$$\left| e^x - \frac{P_m(x)}{Q_k(x)} \right| = \left| \frac{\sum_{j=N+1}^{\infty} d_j x^j}{Q(x)} \right| \tag{7.16}$$

Assumendo $N = 4$, $m = 2$ e $k = 2$, la (7.16) è vera imponendo le condizioni [16]:

$$\begin{aligned}
&\textstyle\sum_{j=0}^{k} c_{N-s-j} b_j = 0 \qquad && s = 0, 1, \ldots, N - m - 1 \\
& && c_i = 0, \quad if \quad i < 0, \\
& && b_0 = 1 \\
&a_r = \textstyle\sum_{j=0}^{r} c_{r-j} b_j = 0 && r = 0, 1, \ldots, m \\
& && b_j = 0, \quad if \quad j > k
\end{aligned}$$

che, dalla (7.15), essendo

$$c_0 = 1, \qquad c_1 = 1, \qquad c_2 = 1/2, \qquad c_3 = 1/6, \qquad c_4 = 1/24$$

danno luogo al sistema di $N + 1 = 5$ equazioni in $N + 1 = 5$ incognite:

$$\begin{cases}
\frac{1}{24} b_0 + \frac{1}{6} b_1 + \frac{1}{2} b_2 = 0 & \Rightarrow \quad \frac{1}{24} + \frac{1}{6} b_1 + \frac{1}{2} b_2 = 0 \\
\frac{1}{6} b_0 + \frac{1}{2} b_1 + b_2 = 0 & \Rightarrow \quad \frac{1}{6} + \frac{1}{2} b_1 + b_2 = 0 \\
a_0 = c_0 b_0 & \Rightarrow \quad a_0 = 1 \\
a_1 = b_0 + b_1 & \Rightarrow \quad a_1 = 1 + b_1 \\
a_2 = \frac{1}{2} b_0 + b_1 + b_2 & \Rightarrow \quad a_2 = \frac{1}{2} + b_1 + b_2
\end{cases}$$

[16] Non è restrittivo supporre che sia $b_0 = 1$; in effetti, non può essere nullo poiché l'approssimazione deve esistere in $x = 0$ ed, inoltre, il valore di $R_{mk}(x)$ deve restare invariato se il numeratore ed il denominatore sono divisi per la stessa costante.

Calcolando b_1 e b_2 dalle prime due equazioni e sostituendo nelle seguenti si trova:

$$b_1 = -1/2, \qquad b_2 = 1/12, \qquad a_0 = 1, \qquad a_1 = 1/2, \qquad a_2 = 1/12$$

così che:

$$R_{2,2}(x) = \frac{12 + 6x + x^2}{12 - 6x + x^2}$$

Analogamente si calcolano le altre funzioni della *tavola di Padé* $R_{m,k}$, $m, k = 0, \ldots, N$:

$$R_{4,0}(x) = 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \tfrac{1}{24}x^4$$

$$R_{3,1}(x) = \tfrac{24+18x+6x^2+x^3}{24-6x}$$

$$R_{1,3}(x) = \tfrac{24+6x}{24-18x+6x^2-x^3}$$

$$R_{0,4}(x) = \tfrac{1}{1-x+\frac{1}{2}x^2-\frac{1}{6}x^3+\frac{1}{24}x^4}$$

In Tabella 7.2 si descrivono i valori dell'errore $e^x - R_{mk}$, al variare di m e k. In accordo con quanto affermato, si osserva che, in $[-.69, .69] \subset (-\log 2, \log 2)$, la funzione $R_{m,k}$ con $m = k = 2$ è di tipo minimax, in quanto fornisce il minimo errore massimo. Tuttavia, nell'intervallo di ampiezza maggiore, $[-1.0, 1.0]$, $R_{3,1}$ fornisce la migliore approssimazione di tipo minimax. Osserviamo, inoltre, che agli estremi dell'intervallo l'errore cresce molto rapidamente.

	(m,k)				
$x \in [-1,1]$	(4,0)	(3,1)	(2,2)	(1,3)	(0,4)
-1.00	$-.712 \times 10^{-2}$	$.121 \times 10^{-2}$	$-.54 \times 10^{-3}$	$.53 \times 10^{-3}$	$-.135 \times 10^{-2}$
$-.75$	$-.175 \times 10^{-2}$	$.33 \times 10^{-3}$	$-.16 \times 10^{-3}$	$.18 \times 10^{-3}$	$-.50 \times 10^{-3}$
$-.69$	$-.117 \times 10^{-2}$	$.22 \times 10^{-3}$	$-.11 \times 10^{-3}$	$.13 \times 10^{-3}$	$-.37 \times 10^{-3}$
$-.50$	$-.240 \times 10^{-3}$	$.49 \times 10^{-4}$	$-.27 \times 10^{-4}$	$.32 \times 10^{-4}$	$-.104 \times 10^{-3}$
$-.25$	$-.78 \times 10^{-5}$	$.18 \times 10^{-5}$	$-.11 \times 10^{-5}$	$.14 \times 10^{-5}$	$-.52 \times 10^{-5}$
$.25$	$.86 \times 10^{-5}$	$-.23 \times 10^{-5}$	$.18 \times 10^{-5}$	$-.29 \times 10^{-5}$	$.129 \times 10^{-4}$
$.50$	$.284 \times 10^{-3}$	$-.88 \times 10^{-4}$	$.73 \times 10^{-4}$	$-.134 \times 10^{-3}$	$.653 \times 10^{-3}$
$.69$	$.147 \times 10^{-2}$	$-.50 \times 10^{-3}$	$.45 \times 10^{-3}$	$-.88 \times 10^{-3}$	$.463 \times 10^{-2}$
$.75$	$.225 \times 10^{-2}$	$-.79 \times 10^{-3}$	$.72 \times 10^{-3}$	$-.147 \times 10^{-2}$	$.783 \times 10^{-2}$
1.00	$.995 \times 10^{-2}$	$-.394 \times 10^{-2}$	$.400 \times 10^{-2}$	$-.899 \times 10^{-2}$	$.5162 \times 10^{-1}$

Tabella 7.2: Errore $e^x - R_{mk}(x)$, nei punti $x \in [-1, 1]$.

♣

Le conclusioni dell'esempio 7.4 suggeriscono di approssimare funzioni, laddove è possibile, su intervalli di ampiezza piccola, in cui le funzioni $R_{m,k}$, con $m = k$ oppure $m = k + 1$, forniscono il minimo errore massimo.
Una caratteristica tipica delle approssimazioni di Padé è il comportamento della funzione che misura la distanza dalla funzione da approssimare. Come accade per il polinomio

ottenuto troncando lo sviluppo in serie di Mc Laurin della funzione f, anche per le approssimazioni di Padé la distanza dalla funzione f aumenta all'aumentare della distanza di x da $x_0 = 0$. È auspicabile, invece, che il comportamento di tale funzione sia sufficientemente omogeneo sull'intervallo di approssimazione, perché nella costruzione dell'approssimazione minimax, ciò garantisce che la distanza *in ogni punto* dell'intervallo di approssimazione sia sufficientemente piccola. I polinomi di Chebyshev soddisfano questa condizione [17]. Per questo motivo, nella costruzione delll'approssimazione razionale di Padé delle funzioni elementari, di fatto, si utilizza la rappresentazione polinomiale in base di Chebyshev.

Osservazione 7.1. *Consideriamo lo sviluppo in serie di Fourier di una funzione f, espresso in termini di polinomi di Chebyshev, T_n, $n = 1, 2, \ldots$:*

$$f(ax) = \frac{1}{2}c_0(a) + \sum_{n=1}^{\infty} c_n(a) \cdot T_n(x), \qquad -1 \leq x \leq 1 \tag{7.17}$$

I coefficienti $c_j(a)$, $j = 0, 1, \ldots$ dello sviluppo (7.17) dipendono da tutti i valori $f(z)$ assunti da f nell'intervallo $(-a, a)$; inoltre, essi decrescono al crescere di n, per cui l'errore di troncamento

$$e_n(x) = f(ax) - \sigma_n(a, x),$$

dove $\sigma_n(a, x) = \sum_{n=1}^{n} c_n(a) \cdot T_n(x)$, si può stimare con il primo termine della serie trascurato, ovvero

$$e_n(x) = \mathcal{O}(c_{n+1}(a) \cdot T_{n+1}(x)).$$

$T_{n+1}(x) = cos(n + 1)\theta$ oscilla tra $+1$ e -1, per $x \in [-1, 1]$:

$$|T_{n+1}(x)| \leq 1, \quad x \in [-1, 1]$$

e si annulla $n + 1$ volte in $(-1, 1)$, nei punti

$$x_j = cos\left(\frac{2j+1}{n+1} \cdot \frac{\pi}{2}\right), \qquad j = 0, \ldots, n.$$

Dunque, l'errore di troncamento, il cui comportamento è, essenzialmente, quello di $T_{n+1}(x)$, **ha $n + 1$ zeri e $n + 2$ valori estremi, uguali in valore assoluto (in particolare i loro moduli sono uguali ad 1) e di segni alterni e sono assunti in corrispondenza dei punti**

$$x_j = cos\left(\frac{j\pi}{n+1}\right), \qquad j = 0, \ldots, n + 1;$$

[17]

Teorema 7.2.9. (Chebyshev) *Tra tutti i polinomi di grado r, con coefficiente di x^r uguale ad uno (monici), il polinomio di Chebyshev di grado r, moltiplicato per $1/2^{r-1}$ oscilla con ampiezza massima minima, nell'intervallo $[-1, 1]$.*

questi ultimi includono gli n punti in cui $T'_{n+1} = 0$ *ed i due estremi dell'intervallo* $[-1, 1]$. *Pertanto, il comportamento del polinomio* $\sigma_n(a, x)$ *è simile a quello del polinomio* P^*, *di grado minore o uguale a n, che realizza la migliore approssimazione polinomiale in senso minimax.*

Detta, infine, $T_{m,0}$ la funzione calcolata troncando la somma nella (7.17) dopo $j = m$ termini, utilizzando i polinomi di Chebyshev si generano approssimazioni razionali analoghe a quelle di Padé:

$$T_{m,k}(x) = \frac{\sum_{j=0}^{m} a_j T_j(x)}{\sum_{j=0}^{k} b_j T_j(x)}.$$

Attraverso i valori in Tabella 7.3 è possibile un confronto in accuratezza tra le approssimazioni $R_{2,2}(x)$ e $T_{2,2}(x)$.

$x \in [-1, 1]$	$e^x - R_{2,2}(x)$	$e^x - T_{2,2}(x)$
-1.00	$-.54 \times 10^{-3}$	$-.42 \times 10^{-4}$
$-.75$	$-.16 \times 10^{-3}$	$.47 \times 10^{-4}$
$-.50$	$-.27 \times 10^{-4}$	$-.18 \times 10^{-4}$
$-.25$	$-.11 \times 10^{-5}$	$-.7 \times 10^{-4}$
0.0	0.0×10^{0}	-0.2×10^{-4}
$.25$	$.18 \times 10^{-5}$	0.88×10^{-4}
$.50$	$.73 \times 10^{-4}$	$.84 \times 10^{-4}$
$.75$	$.72 \times 10^{-3}$	$-.122 \times 10^{-3}$
1.00	$.400 \times 10^{-2}$	$.189 \times 10^{-3}$

Tabella 7.3: Errore introdotto dalle approssimazioni razionali $R_{2,2}(x)$ **e** $T_{2,2}(x)$ **di** e^x**, nei punti** $x \in [-1, 1]$**.**

Bibliografia

[1] Atkinson K. E. - *An Introduction to Numerical Analysis* - J. Wiley and Sons, New York, II ed., 1989.

[2] Bui H. T., Tahar S. - *Design and Synthesis of an IEEE-754 Exponential Function* - Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, vol. 1, pp.450-455 (1999).

[3] Cheney E. W. - *Introduction to Approximation Theory* - International series in pure and applied mathematics, McGraw-Hill Book Company, New York, 1966.

[4] Cody W. and Waite W. - *Software Manual for the Elementary Functions* - Prentice-Hall, Englewood Cliffs, N.J., 1980.

[5] Harrison J. R. - *Floating point verification in HOL Light: the exponential function* - Technical Report number 428, University of Cambridge Computer Laboratory. UK (1997).

[6] Hastings C. Jr - *Approximations for digital computers* - Princeton University Press, 1955

[7] Kogbetliantz E. G. - *Generation of Elementary Functions* - Rockefeller Institute for Medical Research, International Business Machines Corporation (in `Ralston A., Wilf H. - Mathematical methods for digital computers - Vol. I, John Wiley & Sons, Inc., 1960`).

[8] Liu Z. A. - *Berkeley elementary function test suite. Master's thesis* - Computer Science Div., Dept. Electrical Engineering and Computer Science, Univ. of California at Berkeley, 1987.

[9] Mathis R. F. - *Elementary functions package for Ada* - Proceedings of the 1987 annual ACM SIGAda International Conference on Ada, ACM, pp.95-100 (1987).

[10] Paul G. and Wilson M. W. - *Should the Elementary Function Library Be Incorporated Into Computer Instruction Sets?* - IBM Corporation, ACM, Vol. 2, No. 2, pp. 132-142 (1976).

[11] Ping Tank Peter Tang - *Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic* - ACM, Vol. 15, No. 2, pp.144-157 (1989).

[12] Ping Tank Peter Tang - *Table-Driven Implementation of the Logarithm Function in IEEE Floating-Point Arithmetic* - ACM, Vol. 6, No. 4, pp.378-400 (1990).

[13] Ping Tank Peter Tang - *Accurate and Efficient Testing of the Exponential and Logarithm Functions* - ACM, Vol. 16, No. 3, pp.185-200 (1990).

[14] Ralston A. - *Rational Chebyshev Approximation* - State University of New York at Buffalo (in `Ralston A., Wilf H. - Mathematical methods for digital computers - Vol. II, John Wiley & Sons, Inc., 1967`).

[15] Rivlin T. - *An Introduction to the Approximation of Functions* - Dover publications, New York, 1969.

[16] Sterbenz Pat H. - *Floating point computation* - Prentice-Hall, Englewood Cliffs, NJ, 1974.

[17] Stoer J., Bulirsch R. - *Introduction to numerical analysis* - Springer-Verlag, New York, Inc., third edition, 2002.

[18] Watson G. A. - *Approximation Theory and Numerical Methods* - John Wiley & Sons, New York, 1980.

Capitolo 8

Sulla convergenza dei polinomi interpolanti

8.1 Introduzione

In questo capitolo affronteremo un particolare problema di approssimazione che, come vedremo, è strettamente legato all'interpolazione. In breve, il quesito che ci poniamo è analizzare sotto quali condizioni un polinomio **interpolante** i valori assunti da una funzione f, possa essere utilizzato anche come modello **approssimante** la funzione stessa. Più precisamente tale problema può essere così formulato: assegnata una funzione f su un intervallo chiuso e limitato $[a, b]$, si vuole determinare una successione di polinomi $\{p_n\}_{n=0}^{\infty}$ tale che essa converga uniformemente a f, ovvero che converga rispetto alla norma infinito:

$$\lim_{n \to +\infty} \|f - p_n\|_\infty = 0$$

dove

$$\|f - p_n\|_\infty = \max_{x \in [a,b]} |f(x) - p_n(x)|$$

L'obiettivo è costruire tale successione utilizzando i polinomi interpolanti i valori assunti dalla funzione f.

Introduciamo la seguente definizione:

Definizione 8.1.1. *Sia f una funzione definita su un intervallo chiuso e limitato $[a, b]$, una* **matrice di interpolazione** *per f è una matrice triangolare inferiore* infinita*, X, del tipo:*

$$
\begin{array}{ccccccccc}
x_0^{(0)} & & & & & & & & \\
x_0{}^{(1)} & x_1{}^{(1)} & & & & & & & \\
x_0{}^{(2)} & x_1{}^{(2)} & x_2{}^{(2)} & & & & & & \\
x_0{}^{(3)} & x_1{}^{(3)} & x_2{}^{(3)} & x_3{}^{(3)} & & & & & \\
\dots & \dots & \dots & \dots & \dots & & & & \\
\dots & \dots & \dots & \dots & \dots & \dots & & & \\
x_0{}^{(n)} & x_1{}^{(n)} & x_2{}^{(n)} & x_3{}^{(n)} & \dots & x_{n-1}^{(n)} & x_n^{(n)} & & \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots &
\end{array}
$$

tale che:

$$a \ \leq \ x_0{}^{(k)} \ < \ldots < \ x_k{}^{(k)} \ \leq \ b, \quad k = 0, 1..., n$$

e su cui f è nota.

Se ogni riga di X è utilizzata come insieme di nodi di interpolazione, la matrice X individua una successione di polinomi interpolanti che indicheremo con:

$$\{L_n(f, X)\}_{n=0}^{\infty} \equiv \{L_n\}_{n=0}^{\infty},$$

dove $L_n(f, X) \equiv L_n \in \Pi_n$ è l'unico polinomio di Lagrange, di grado n, interpolante f negli $n + 1$ punti appartenenti alla *n+1*-esima riga di X, $n = 0, 1, \ldots$.
Ci si chiede ora se esistano delle condizioni sulla funzione f e sulla matrice di interpolazione X, tali che la successione $\{L_n\}_{n=0}^{\infty}$ converga a f in modo uniforme, cioè se esistano delle condizioni per cui:

$$\lim_{n \to +\infty} E_n(X) = 0,$$

avendo posto:

$$E_n(X) = \max_{x \in [a,b]} |f(x) - L_n(x)|$$

♣ **Esempio 8.1.** Consideriamo la funzione esponenziale $f(x) = e^x$ nell'intervallo $[-1, 1]$. Essendo f una funzione analitica, si può esprimere come:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \quad \forall x \in [-1, 1].$$

Posto

$$p_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + ... + \frac{x^n}{n!},$$

ed utilizzando la stima di Lagrange per il resto n-esimo $R_{n+1}(x)$, si ha che:

$$R_{n+1}(x) = |e^x - p_n(x)| = \frac{|x^{n+1}|}{(n+1)!} e^{\xi}, \quad |\xi| < |x|, \quad \forall x \in [-1, 1];$$

passando alla norma infinito si può fornire una stima dell'errore di approssimazione $E_n(f, p_n)$:

$$E_n(f, p_n) = ||e^x - p_n||_\infty = \max_{x \in [-1,1]} |e^x - p_n(x)| \leq \frac{e}{(n+1)!}$$

e quindi:

$$E(f, p_n) \to 0, \quad n \to +\infty.$$

In Tabella 8.1 si riportano i valori della stima $\frac{e}{(n+1)!}$. Si osserva che, fissando una tolleranza sull'errore uguale a 10^{-8}, i polinomi di grado $n \geq 11$ forniscono un'approssimazione di f che soddisfa la tolleranza richiesta (Figura 8.1).

n	$e/(n+1)!$
0	$2.7183e+000$
1	$1.3591e+000$
2	$4.5305e-001$
3	$1.1326e-001$
4	$2.2652e-002$
5	$3.7754e-003$
6	$5.3934e-004$
7	$6.7418e-005$
8	$7.4909e-006$
9	$7.4909e-007$
10	$6.8099e-008$
11	$5.6749e-009$
12	$4.3653e-010$

Tabella 8.1: Stima dell'errore di approssimazione $E_n(f, p_n)$ tra la funzione esponenziale $f(x) = e^x$ ed i suoi polinomi di Taylor sull'intervallo $[-1, 1]$.

Costruiamo la matrice X, della Definizione 8.1.1, nel seguente modo:

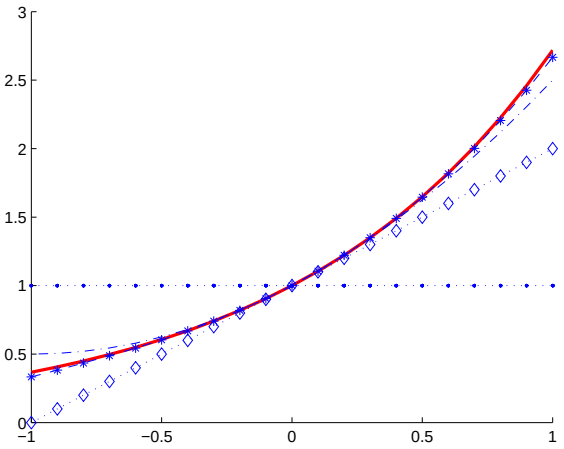

Figura 8.1: La funzione esponenziale $f(x) = e^x$ sull'intervallo $[-1, 1]$ (linea continua) ed i suoi polinomi di Taylor di grado $n = 0$ ('-•'), $n = 1$ ('-◇'), $n = 2$ ('-.') e $n = 3$ ('-*').

$$\begin{array}{cccccc} -1 & & & & & \\ -1 & 1 & & & & \\ -1 & 0 & 1 & & & \\ -1 & -0.33333 & 0.33333 & 1 & & \\ -1 & -0.5 & 0 & 0.5 & 1 & \\ -1 & -0.6 & -0.2 & 0.2 & 0.6 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \end{array} \tag{8.1}$$

dove la riga *n+1*-esima di X contiene gli $n+1$ nodi equidistanziati $x_0^{(0)} = -1$ e $x_i^{(n)} = -1+hi,\ i = 0, ..., n$ con $h = \frac{2}{n}$, per $n = 1, 2, \dots$. La successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ costruita su X converge uniformemente a $f(x) = e^x$. Infatti, $\forall x \in [-1, 1]$, essendo f infinitamente derivabile, si dimostra (cfr. §8.3):

$$|f(x) - L_n(x)| = \left| \frac{f^{n+1}(\xi)}{(n+1)!} \cdot \Pi_{i=0}^{n}(x - x_i^{(n)}) \right|, \tag{8.2}$$

dove $\min\{x, x_0^{(n)}, x_1^{(n)}, ..., x_n^{(n)}\} < \xi < \max\{x, x_0^{(n)}, x_1^{(n)}, ..., x_n^{(n)}\}$.
Inoltre [1] si può dimostrare che:

$$\left| \Pi_{i=0}^{n}(x - x_i^{(n)}) \right| \leq n! \frac{h^{n+1}}{4}$$

da cui:

$$|f(x) - L_n(x)| \leq \frac{\max\limits_{x \in [-1,1]} \left| f^{n+1}(x) \right|}{(n+1)!} \cdot n! \frac{h^{n+1}}{4} = \frac{e}{(n+1)!} \frac{n!}{4} \cdot \left(\frac{2}{n} \right)^{n+1} = \frac{e}{2n(n+1)} \left(\frac{2}{n} \right)^{n} \tag{8.3}$$

e quindi:

[1] Il risultato sussiste per una qualsiasi distribuzione uniforme di nodi, del tipo $x_i = x_{i-1} + h$, $h > 0$, $i = 1, \dots, n$ e x_0 fissato.

$$\lim_{n\to+\infty} E_n(X) = \lim_{n\to+\infty} \left(\max_{x\in[a,b]} |f(x) - L_n(x)| \right) = 0.$$

In Tabella 8.2 si riportano i valori della stima (8.3). Si osserva che, fissata la tolleranza sull'errore di 10^{-8}, il polinomio di grado 10 fornisce un'approssimazione uniforme che soddisfa la tolleranza richiesta. Ciò costituisce un *miglioramento* dal punto di vista computazionale rispetto all'approssimazione polinomiale di grado 11 ottenuta utilizzando i polinomi di Taylor (Figura 8.2).

n	$\frac{e}{2n(n+1)}\frac{2^n}{n^n}$
1	$1.3591e+000$
2	$2.2652e-001$
3	$3.3559e-002$
4	$4.2473e-003$
5	$4.6392e-004$
6	$4.4390e-005$
7	$3.7722e-006$
8	$2.8804e-007$
9	$1.9958e-008$
10	$1.2652e-009$
11	$7.3910e-011$
12	$4.0024e-012$

Tabella 8.2: Stima dell'errore di approssimazione $E_n(f, L_n)$ tra la funzione esponenziale $f(x) = e^x$ ed i polinomi interpolanti di Lagrange sull'intervallo $[-1, 1]$.

♣

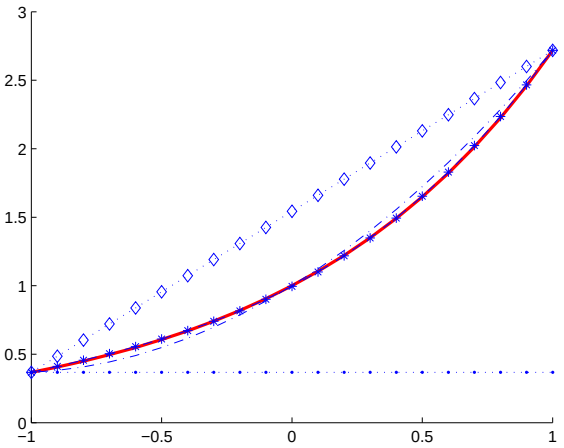

Figura 8.2: La funzione $f(x) = e^x$ nell'intervallo $[-1, 1]$ (linea continua) ed i polinomi interpolanti di Lagrange di grado $n = 0$ ('-•'), $n = 1$ ('-◇'), $n = 2$ ('-.') e $n = 3$ ('-*').

♣ **Esempio 8.2.** Consideriamo la funzione $f(x) = \frac{1}{1+25x^2}$ nell'intervallo $[-1, 1]$ (*funzione di Runge*) e sia X la matrice triangolare inferiore definita nell'esempio 8.1. Se si osserva il grafico della f e dei polinomi interpolanti (Figura 8.3), si può facilmente intuire che in prossimità degli estremi dell'intervallo $[-1,\ 1]$ al crescere di n si ha:

$$\lim_{n\to+\infty} |f(x) - L_n(x)| = +\infty.$$

La successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ determinata da X, quindi, non converge uniformemente a $f(x)$; in seguito si dimostrerà che:

$$\lim_{n\to+\infty} E_n(X) = \lim_{n\to+\infty} \|f - L_n\|_\infty = +\infty$$

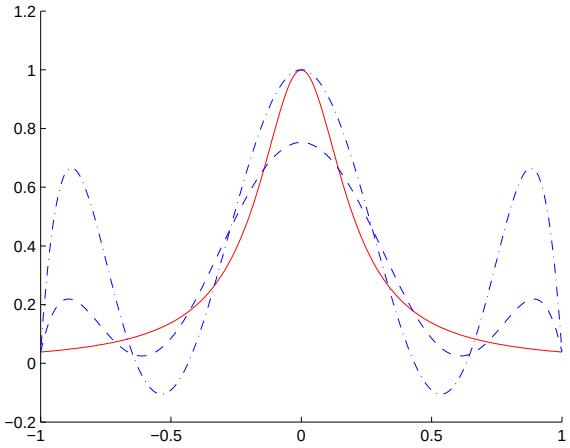

Figura 8.3: La funzione $f(x) = \frac{1}{1+25x^2}$ (linea continua) ed i polinomi interpolanti di Lagrange di grado $n = 6$ ('-.') e $n = 7$ ('- -'), determinati dalla distribuzione dei nodi equidistanziati nell'intervallo $[-1, 1]$.

♣

L'esempio 8.2 mostra come, in generale, non sia possibile determinare un insieme X "universalmente" efficace per ogni funzione continua in un intervallo $[a, b]$. A tal proposito sussiste, infatti, il teorema seguente [2]:

[2]Laddove non riportate, per le dimostrazioni dei teoremi seguenti si veda [4].

Teorema 8.1.1. [Teorema di Faber]
Data $X = \{x_i^{(n)}\}$, $i = 0, ..., n$, $n \in \mathcal{N}$ *una matrice di interpolazione formata da punti di un intervallo chiuso e limitato* $[a, b]$, *esiste almeno un funzione* $f^* \in C[a, b]$ *tale che la successione dei polinomi interpolanti la* f^*, *costruita utilizzando le righe di* X *come nodi di interpolazione, diverga, ovvero:*

$$\lim_{n \to +\infty} E_n(X) = \lim_{n \to +\infty} \|f^* - L_n(f^*, X)\|_\infty = +\infty$$

Tuttavia, fissata una funzione continua $f \in C[a, b]$, è sempre possibile determinare almeno un insieme X per cui la successione dei polinomi interpolanti $\{L_n(f, X)\}_{n=0}^{\infty}$ converge uniformemente:

Teorema 8.1.2. *Data* $f \in C[a, b]$, *esiste una matrice di interpolazione* $X^* = \{x_i^{(n)}\}$, $i = 0, ..., n$, $n \in \mathcal{N}$, *costitutita da punti dell'intervallo* $[a, b]$, *tale che:*

$$\lim_{n \to +\infty} E_n(X^*) = \lim_{n \to +\infty} \|f - L_n(f, X^*)\|_\infty = 0$$

♣ **Esempio 8.3.** Il fenomeno di Runge per la funzione $f(x) = \frac{1}{1+25x^2}$, visto nell'esempio 8.2, può essere evitato utilizzando i nodi di Chebyshev, definiti nel modo seguente:

$$\hat{x_i} = cos\left(\frac{i\pi}{n}\right) \quad i = 0, ..., n.$$

Tali punti detti nodi Chebyshev sono le ascisse di punti equidistanziati sulla circonferenza di raggio unitario e si addensano, al crescere di $n \in \mathcal{N}$, in prossimità degli estremi dell'intervallo $[-1, 1]$.
Mostreremo, in seguito, che la successione di polinomi interpolanti determinata da tali punti converge uniformemente a f, continua e derivabile con continuità, nell'intervallo $[-1, 1]$ (Figura 8.4).

♣

8.2 Preliminari

Al fine di analizzare sotto quali condizioni la successione dei polinomi di Lagrange interpolanti una funzione converga alla funzione stessa, premettiamo alcuni risultati fondamentali.

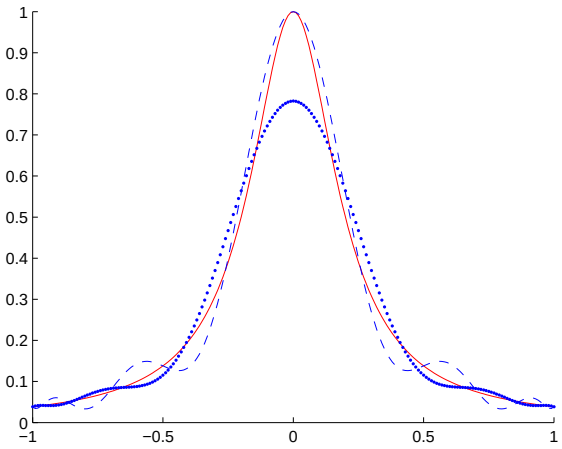

Figura 8.4: La funzione di Runge $f(x) = \frac{1}{1+25x^2}$ (linea continua) ed i polinomi di Lagrange interpolanti f, di grado $n = 11$ ('•') e $n = 12$ ('- -'), determinati dalla distribuzione dei nodi di Chebyshev nell'intervallo $[-1, 1]$.

Definizione 8.2.1. *Sia f una funzione definita in un intervallo $I \subseteq \Re$ e sia $\delta > 0$, si definisce* ***modulo di continuità*** *di f su I la funzione:*

$$\omega(f, \delta) = \omega(\delta) = \sup_{|x_1 - x_2| \leq \delta} |f(x_1) - f(x_2)|$$

con x_1 e $x_2 \in I$.

Per il modulo di continuità valgono le seguenti proprietà:

- se $0 < \delta_1 \leq \delta_2$ allora
$$\omega(\delta_1) \leq \omega(\delta_2);$$
- la funzione f è uniformente continua nell'intervallo $[a, b]$ se e solo se:
$$\lim_{\delta \to 0} \omega(\delta) = 0;$$
- se $\lambda > 0$, allora $\forall \delta > 0$
$$\omega(\lambda\delta) \leq (1 + \lambda)\omega(\delta).$$

♣ **Esempio 8.4.** Si consideri la funzione $f(x) = sin(x^2)$. Comunque si scelga un $\delta > 0$ esiste un $\nu \in \mathcal{N}$ tale che $\forall x_1,\ x_2\ > \nu$:

$$\sup_{|x_1 - x_2| \leq \delta} |f(x_1) - f(x_2)| = 2.$$

Quindi il modulo di continuità di $f(x)$ è la funzione identicamente uguale a 2:

$$\omega(\delta) \equiv 2, \quad \forall \delta > 0$$

data dalla differenza tra il massimo e il minimo di $f(x)$, come mostrato in Figura 8.5. ♣

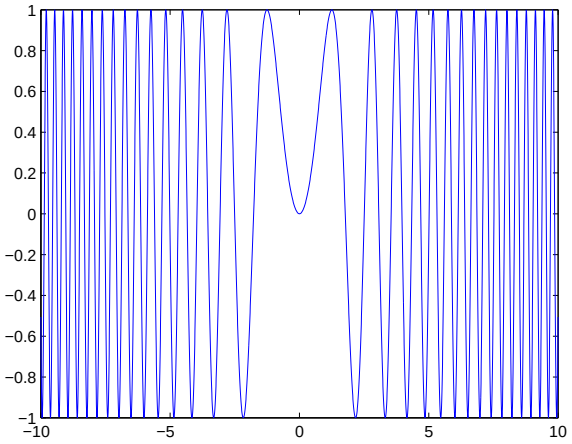

Figura 8.5: Grafico della funzione $f(x) = sin(x^2)$ nell'intervallo $[-10, 10]$.

Si consideri ora una classe di funzioni intermedia tra $C^0([a, b])$ e $C^1([a, b])$.

Definizione 8.2.2. *Una funzione f si dice hölderiana in $[a, b]$ se esiste una costante $M > 0$ ed un numero reale $0 < \alpha \leq 1$ tale che $\forall x, y \in [a, b]$ si ha:*

$$|f(x) - f(y)| < M|x - y|^{\alpha}.$$

Si noti che la hölderianità implica la continuità uniforme e che per $\alpha = 1$, si ritrova la lipschitzianità.
È immediato verificare il seguente teorema che fornisce una stima del modulo di continuità di una funzione hölderiana:

Teorema 8.2.1. *Se f è una funzione hölderiana di grado α (α-hölderiana), il modulo di continuità $\omega(\delta)$ soddisfa la seguente diseguaglianza:*

$$\omega(\delta) < M\delta^{\alpha}, \quad \forall \delta > 0, \tag{8.4}$$

con M costante reale positiva. In particolare, per $\alpha = 1$, il modulo di continuità $\omega(\delta)$ delle funzioni lipschitziane verifica la seguente limitazione:

$$\omega(\delta) < M\delta, \quad \forall \delta > 0 \tag{8.5}$$

ovvero $\omega(\delta)$ ha un andamento sublineare.

Sia $p \in \Pi_n$ il polinomio di migliore approssimazione uniforme di f in Π_n (cfr. **Capitolo 7**) e sia:

$$\mathcal{E}_n(f) = \|f - p\|_{\infty} \tag{8.6}$$

l'errore di migliore approssimazione uniforme polinomiale. Vale il seguente teorema:

Teorema 8.2.2. [Teorema di Jackson]
Sia $f \in C^k([a,b])$ con $k \geq 0$, l'errore di migliore approssimazione uniforme $\mathcal{E}_n(f)$ soddisfa la seguente maggiorazione:

$$\mathcal{E}_n(f) \leq \begin{cases} 6\ \omega(\frac{b-a}{2n}) & se\ f \in C^0([a,b]) \\ C_k(\frac{b-a}{2n})^k & se\ f \in C^k([a,b]) \quad k \geq 1 \end{cases} \tag{8.7}$$

con C_k costante reale positiva.

Per semplicità consideriamo la trasformazione $t = 2\frac{(x-a)}{(b-a)} - 1$ dell'intervallo $[a,b]$ nell'intervallo $[-1,1]$.
Il Teorema 8.2.2 afferma che per funzioni di classe $C^k([-1,1])$ con $k \geq 1$, la quantità $\mathcal{E}_n(f)$ è un infinitesimo di ordine k, mentre per funzioni continue essa è un infinitesimo di ordine uguale a $\omega\left(\frac{1}{n}\right)$. Se, dunque, si considerano funzioni α-hölderiane, per la (8.4) $\mathcal{E}_n(f)$ è un infinitesimo di ordine $0 < \alpha \leq 1$; in particolare, per le funzioni lipschitziane ($\alpha = 1$), $\mathcal{E}_n(f)$ ha ordine di infinitesimo uguale a 1.

Definizione 8.2.3. *Sia X una matrice di interpolazione costituita da punti appartenenti all'intervallo $[a,b]$. Si definisce* ***costante di Lebesgue*** *di ordine $n \in \mathcal{N} \cup \{0\}$ la quantità:*

$$\Lambda_n = \max_{x \in [a,b]} \lambda_n(x), \quad n = 0, 1, \ldots$$

dove $\lambda_n(x)$ indica la somma dei moduli dei polinomi fondamentali di Lagrange $l_i^{(n)}$, $i = 0, ..., n$, relativi alla $n+1$-esima riga di X:

$$\lambda_n(x) = \sum_{i=0}^{n} |l_i^{(n)}(x)|.$$

8.3 La convergenza dei polinomi interpolanti

Teorema 8.3.1. *Sia $f \in C^0([-1,1])$ e X una matrice di interpolazione. Se:*

$$E_n(X) = \max_{x\in[-1,1]} |f(x) - L_n(x)|, \quad n = 0, 1, \ldots \tag{8.8}$$

e $\{L_n\}_{n=0}^{\infty}$ è la successione dei polinomi interpolanti determinata da X, si ha:

$$E_n(X) \leq \mathcal{E}_n(f)\ (1 + \Lambda_n), \quad n = 0, 1, \ldots \tag{8.9}$$

Nell'intervallo $[-1, 1]$, per ogni $n \in \mathcal{N}$, la (8.7), per $k = 0$, si scrive:

$$\mathcal{E}_n(f) \leq\ 6\ \omega\left(\frac{1}{n}\right)$$

e la (8.9) diventa:

$$E_n(X) \leq 6\ (1 + \Lambda_n)\ \omega\left(\frac{1}{n}\right) \tag{8.10}$$

La (8.10) garantisce che la successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ converga uniformemente a f, se il prodotto $\Lambda_n\ \omega\left(\frac{1}{n}\right)$ tende a zero per $n \to +\infty$. È importante, dunque, conoscere una stima asintotica delle costanti di Lebesgue. A tal fine si enunciano i seguenti teoremi:

Teorema 8.3.2. *Per ogni scelta di X:*

$$\Lambda_n(X) > \frac{2}{\pi^2} log(n) - 1, \qquad n \geq 1. \tag{8.11}$$

Questo risultato ha un'importante conseguenza che si ritrova nel Teorema 8.1.1.

Teorema 8.3.3. *[3]*
Sia X la matrice di interpolazione costitutita da nodi equidistanziati nell'intervallo $[-1,1]$:

$$x_i = x_0 + hi,\ i = 0, ..., n, \quad con \quad h = \frac{2}{n} \quad e \quad n \geq 1$$

allora

$$\Lambda_n(X) \geq \frac{2^{n+1}}{e\, n(log(n) + \gamma)}, \qquad n \geq 1 \tag{8.12}$$

dove e *indica il numero di Nepero e γ la costante di Eulero.*

Teorema 8.3.4. *Sia X la matrice di interpolazione costitutita dagli zeri di Chebyshev nell'intervallo $[-1,1]$:*

$$\hat{x_i} = cos\left(\frac{i\pi}{n}\right) \quad i = 0, ..., n \quad n \geq 1$$

allora:

$$\Lambda_n(X) < \frac{2}{\pi} log(n) + 4, \qquad n \geq 1. \tag{8.13}$$

Dalle (8.11), (8.12) e (8.13) si deduce che, per approssimare una funzione definita nell'intervallo $[-1,1]$ mediante polinomi interpolanti, una buona scelta dei nodi di interpolazione X è rappresentata dagli zeri di Chebyshev. In tal caso, le costanti di Lebesgue sono di ordine al più logaritmico e dunque non sono richieste proprietà di regolarità della funzione affinchè il prodotto $\Lambda_n\, \omega\big(\frac{1}{n}\big)$ tenda a zero per $n \to +\infty$.

♣ **Esempio 8.5.** Si consideri nuovamente la funzione di Runge $f(x) = \frac{1}{1+25x^2}$ e la matrice di interpolazione X costituita dai nodi di Chebyshev nell'intervallo $[-1,1]$:

$$\hat{x_i} = cos\left(\frac{i\pi}{n}\right) \quad i = 0, ..., n$$

Per tale distribuzione di nodi, la (8.13) mostra che la successione delle costanti di Lebesgue cresce con andamento asintotico al più logaritmico:

$$\Lambda_n \leq \mathcal{O}\big(log(n)\big) \quad (n \to \infty). \tag{8.14}$$

La funzione $f(x) = \frac{1}{1+25x^2}$ è una funzione $C^\infty([-1,1])$ e quindi, per il Teorema di Jackson, per ogni valore di k arbitrariamente fissato, si ha:

$$\omega\left(\frac{1}{n^k}\right) \leq \mathcal{O}\left(\frac{1}{n^k}\right) \quad (n \to \infty) \tag{8.15}$$

Utilizzando la (8.10), la (8.14) e la (8.15) segue:

$$E_n(X) \leq 6\ (1+\Lambda_n)\ \omega\left(\frac{1}{n}\right) \leq 6\ \left(1+\mathcal{O}\big(log(n)\big)\right)\mathcal{O}\left(\frac{1}{n^k}\right) \quad (n \to \infty) \tag{8.16}$$

ovvero, per $n \to +\infty$, $E_n(X)$ tende a 0 e la successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ converge uniformemente a f in $[-1,1]$. ♣

♣ **Esempio 8.6.** Si consideri la funzione $f(x) = |x|$ nell'intervallo $I = [-1,1]$. f è continua ma non derivabile in I. Si verifica facilmente che f è una funzione hölderiana di grado $\alpha = 1$, ovvero è lipschitziana, con costante di Lipschitz uguale a 1:

$$||x| - |y|| \leq |x-y|, \quad \forall x, y \in [-1,1]$$

Per la (8.5) si ha dunque che:

$$\omega\left(\frac{1}{n}\right) < \mathcal{O}\left(\frac{1}{n}\right)$$

Considerata la distribuzione dei nodi equidistanziati X:

$$x_i = -1 + hi,\ i = 0, ..., n, \quad \text{con} \quad h = \frac{2}{n} \quad \text{e} \quad n \geq 1,$$

per la (8.12):

$$\Lambda_n(X) = \mathcal{O}\left(\frac{2^{n+1}}{nlog(n)}\right),$$

da cui:

$$\Lambda_n(X)\omega\left(\frac{1}{n}\right) = \mathcal{O}\left(\frac{2^{n+1}}{n^2 log(n)}\right) \longrightarrow +\infty \quad (n \to +\infty).$$

La successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ non converge uniformemente a f, come si osserva dalla Figura 8.6.

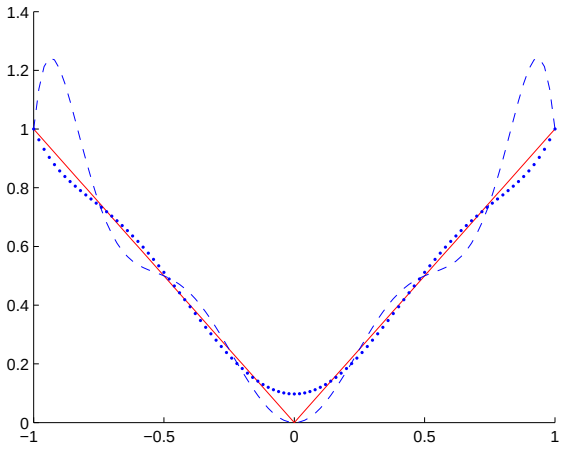

Figura 8.6: Grafico della funzione $f(x) = |x|$ (linea continua) e dei polinomi di Lagrange interpolanti la f su nodi equidistanziati, con $h = 2/n$, di grado $n = 7$ ('•') e $n = 8$ ('- -').

Se invece si utilizza la distribuzione X costituita dai nodi di Chebyshev nell'intervallo $[-1, 1]$:

$$\hat{x_i} = cos\left(\frac{i\pi}{n}\right) \quad i = 0, ..., n, \qquad n \geq 1,$$

si ha:

$$\Lambda_n = \mathcal{O}\big(log(n)\big),$$

da cui:

$$\Lambda_n \omega\left(\frac{1}{n}\right) = \mathcal{O}\left(\frac{log(n)}{n}\right) \longrightarrow 0, \quad n \to +\infty \tag{8.17}$$

La (8.17) garantisce la convergenza uniforme a f dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$, come si osserva anche dalla Figura 8.7. ♣

Il teorema seguente fornisce condizioni utili ai fini della convergenza uniforme dei polinomi interpolanti, nell'ipotesi in cui la funzione f sia sufficientemente regolare in un intervallo $[a, b]$.

Teorema 8.3.5. *Sia $f \in C^n([a, b])$, con derivata continua in ogni punto $x \in (a, b)$. Si consideri una distribuzione arbitraria di $n + 1$ punti:*

$$a \leq x_0 \ < x_1 \ < ... < x_n \leq b,$$

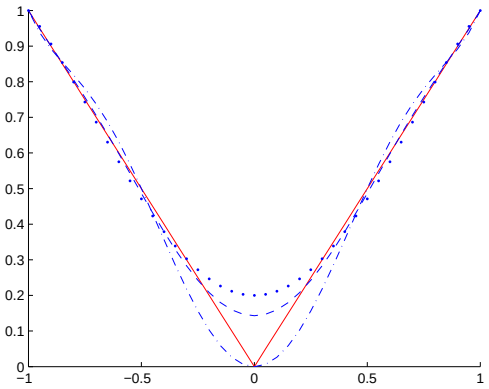

Figura 8.7: Grafico della funzione $f(x) = |x|$ (linea continua) e dei polinomi di Lagrange interpolanti la f su nodi di Chebyshev, di grado $n = 5$ ('•'), $n = 6$ ('-.') e $n = 7$ ('- -').

allora, $\forall x \in [a, b]$ *esiste* $\xi \in (a, b)$ *tale che:*

$$f(x) - L_n(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_n)}{(n + 1)!} f^{n+1}(\xi), \tag{8.18}$$

dove $\min\{x, x_0, x_1, ..., x_n\} < \xi < \max\{x, x_0, x_1, ..., x_n\}$ *dipende da* $x, x_0, x_1, ..., x_n$ *e da* f.

Dimostrazione Se $x = x_k$, $\forall k = 0, 1, ..., n$, poiché $L_n(x_k) = f(x_k)$, la differenza $f(x_k) - L_n(x_k)$ si annulla e quindi l'uguaglianza (8.18) è banalmente vera $\forall \xi \in [a, b]$. Se $x \neq x_k$, con $k = 0, ..., n$, si consideri la quantità:

$$K(x) = \frac{f(x) - L_n(x)}{(x - x_0)(x - x_1)\dots(x - x_n)}$$

e la funzione W nella variabile $t \in [a, b]$:

$$W(t) = f(t) - L_n(t) - (t - x_0)(t - x_1)...(t - x_n)K(x).$$

La funzione $W(t)$ si annulla nei punti $t = x_k$, $\forall k = 0, ..., n$, e nel punto $t = x$. Applicando il Teorema di Rolle generalizzato, esiste ξ, con $\min\{x, x_0, x_1, ..., x_n\} < \xi < \max\{x, x_0, x_1, ..., x_n\}$, tale che:

$$W^{n+1}(\xi) = f^{n+1}(\xi) - (n + 1)!K(x) = 0, \tag{8.19}$$

da cui:

$$K(x) = \frac{f^{n+1}(\xi)}{(n + 1)!}$$

e quindi l'asserto. ■

Utilizzando la (8.18) segue:

Corollario 8.3.1. *Sia $f \in C^{n+1}([a,b])$ e X una distribuzione arbitraria di punti di $[a,b]$, allora*

$$E_n(X) \leq \frac{|x-x_0||x-x_1|\cdots|x-x_n|}{(n+1)!} \cdot \max_{x\in[a,b]} |f^{n+1}(x)|. \tag{8.20}$$

♣ **Esempio 8.7.** Riprendiamo l'esempio 8.1. Per il Teorema 8.3.5 ed il Corollario 8.3.1 si ha la (8.2) da cui segue la (8.3) e, dunque, la convergenza uniforme a $f(x) = e^x$ della successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$, costruita sulla matrice di interpolazione (8.1).

♣

♣ **Esempio 8.8.** Un'approssimazione della funzione $f(x) = arcosen(x)$ in $x = 0.5335$ è ottenuta mediante il polinomio interpolante di primo grado ($n = 1$) relativo ai nodi $x = 0.5330$ e $x = 0.5340$. Per stimare l'errore di approssimazione nel punto $x = 0.5335$ si utilizza la (8.20): poichè la derivata terza $f'''(x) = (1+2x^2)(1-x^2)^{-\frac{5}{2}}$ è positiva sull'intervallo $0.5330 < x < 0.5340$, il valore massimo della derivata seconda $f''(x) = x(1-x^2)^{-\frac{3}{2}}$ è assunto nel punto $x = 0.5340$. Dalla (8.20) si ottiene quindi:

$$E_1(0.5335) \leq \frac{0.5340}{(1-(0.5340)^2)^{\frac{3}{2}}} \frac{(0.0005)^2}{2} \leq \ 1.2 \times 10^{-7}$$

Un calcolo diretto della differenza tra $f(x) - L_1(x)$ restituisce, infatti, 1.101×10^{-7}. ♣

Corollario 8.3.2. *Sia $f \in C^{\infty}([a,b])$. Condizione sufficiente affinché, per ogni scelta di X, la successione dei polinomi interpolanti $\{L_n(f,X)\}$ converga uniformemente a f su $[a,b]$ è che:*

$$|f^n(x)| \leq M\ L^n, \qquad \forall n \in \mathcal{N} \cup \{0\}$$

con M e L costanti reali positive.

♣ **Esempio 8.9.** Si consideri la funzione $f(x) = cos(x)$ nell'intervallo $I = [-1, 1]$ e la matrice X costruita utilizzando i punti distribuiti in modo casuale nell'intervallo I:

$$\begin{array}{cccccccc}
-5.8186e-01 & & & & & & & \\
-5.8186e-01 & -5.4810e-01 & & & & & & \\
-5.8186e-01 & -5.4810e-01 & -3.3210e-01 & & & & & \\
-5.8186e-01 & -5.4810e-01 & -3.3210e-01 & -1.3419e-01 & & & & \\
-5.8186e-01 & -5.4810e-01 & -3.3210e-01 & -1.3419e-01 & 0 & & & \\
-5.8186e-01 & -5.4810e-01 & -3.3210e-01 & -1.3419e-01 & 0 & 3.1024e-02 & & \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots &
\end{array} \tag{8.21}$$

aggiungendo, all'n-esima riga, l'n-esimo elemento calcolato come $x_n^{(n)} = -1+2\cdot rand$, con $rand$ numero scelto, in maniera casuale, da una distribuzione uniforme di punti appartenenti all'intervallo $(0, 1)$.

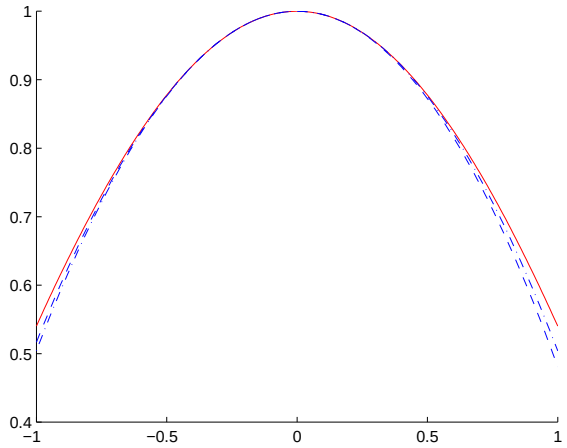

Figura 8.8: Grafico della funzione $f(x) = cos(x)$ (linea continua) e dei polinomi di Lagrange interpolanti la f nei punti della matrice (8.21), di grado $n = 2$ ('-.') e $n = 3$ ('- -').

La funzione $f \in C^\infty([-1, 1])$ ed è semplice verificare che le sue derivate sono equilimitate sull'intervallo $[-1, 1]$. Per il Corollario 8.3.2 la successione dei polinomi $\{L_n\}_{n=0}^\infty$, converge uniformemente a f, come mostrato in Figura 8.8. Inoltre:

$$E_n(X) \leq \frac{2^{n+1}}{(n+1)!}, \quad n \geq 1.$$

♣

Si noti che per utilizzare la (8.20), è necessario calcolare le derivate di f fino all'ordine $(n+1)$-esimo, per poi determinare un limite superiore della derivata f^{n+1} nell'intervallo

$[a, b]$. Tale procedimento richiede un costo computazionale elevato già per funzioni elementari. Nel caso in cui f sia definita nel campo complesso $\mathbb{C}$ ed ivi analitica, si riescono ad avere utili stime per l'errore che prescindono dalla conoscenza dalle derivate di f. Si mostrerà un risultato secondo il quale se i nodi di interpolazione, considerati nel piano complesso, sono tutti confinati in una regione limitata T di $\mathbb{C}$, in cui f risulti analitica, allora i polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ convergono uniformemente a f in una regione S strettamente contenuta in T. Tale risultato fa uso della **formula integrale di Cauchy**:

Teorema 8.3.6. [**Teorema di Hermite**]
Sia $f(z)$ una funzione analitica su una regione chiusa, semplicemente connessa, R di $\mathbb{C}$. Sia Γ una curva semplice, chiusa e rettificabile, che giace in R ed il cui interno contiene i punti di una distribuzione X, $x_0, x_1, ..., x_n$, con $n \in \mathcal{N}$. Allora, $\forall z \in R$, si ha:

$$E_n(z) = f(z) - L_n(z) = \frac{1}{2\pi i} \int_{\Gamma} \frac{(z - x_0)(z - x_1)...(z - x_n) f(t)}{(t - x_0)(t - x_1)...(t - x_n)(t - z)} dt. \tag{8.22}$$

Tale espressione di $E_n(z)$ prende il nome di ***resto integrale di Cauchy****.*

Dimostrazione Si consideri l'integrale:

$$I = \frac{1}{2\pi i} \int_{\Gamma} \frac{f(t)}{(t - x_0)(t - x_1)...(t - x_n)} dt. \tag{8.23}$$

La funzione integranda nella (8.23) è analitica o ha poli semplici in corrispondenza dei punti $x_0, x_1, ...x_n$; utilizzando la teoria dei residui, si ha:

$$I = \sum_{k=0}^{n} \frac{f(x_k)}{(x_k - x_0)(x_k - x_1)...(x_k - x_{k-1})(x_k - x_{k+1})...(x_k - x_n)}\,. \tag{8.24}$$

Inoltre, si dimostra[3] che tale quantità è uguale alla differenza divisa di ordine n, relativa ai punti $\{(x_k, f(x_k))\}_{k=0,...,n}$, da cui:

$$y[x_0, x_1, ...x_n, z] = \frac{1}{2\pi i} \int_{\Gamma} \frac{(z - x_0)(z - x_1)...(z - x_n) f(t)}{(t - x_0)(t - x_1)...(t - x_n)(t - z)} dt, \quad t \in R$$

e che:

$$E_n(z) = f(z) - p_n(z) = y[x_0, x_1, ...x_n, z](z - x_0)(z - x_1)...(z - x_n).$$

Per il resto n-esimo dell'interpolazione polinomiale, $E_n(z)$, vale, dunque, l'asserto. ■

Teorema 8.3.7. *Siano R, S e T regioni limitate di $\mathbb{C}$ semplicemente connesse, tali che $R \subset S \subset T$, le cui frontiere sono indicate rispettivamente con C_R, C_S, C_T, e sia $f(z)$*

[3] [1]

una funzione analitica in T e sulla frontiera C_T. Sia δ la minima distanza da C_T a C_R e Δ la massima distanza da C_S a C_R ed indichiamo con X una matrice di interpolazione costituita da punti appartenenti a R. Condizione sufficiente, affinché la successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$, determinata da X, converga uniformente a f in S, è che:

i) C_T sia una curva semplice, chiusa e rettificabile;

ii) $C_S \bigcap C_T = \emptyset$;

iii) $\frac{\Delta}{\delta} < 1$.

Dimostrazione Utilizzando la (8.22) si ottiene, $\forall z \in S$:

$$E_n(z) = f(z) - L_n(z) = \frac{1}{2\pi i}\int_{C_T} \frac{(z-x_{n0})(z-x_{n1})...(z-x_{nn})f(t)}{(t-x_{n0})(t-x_{n1})...(t-x_{nn})(t-z)}dt. \tag{8.25}$$

Passando ai moduli:

$$|E_n(z)| \leq \frac{1}{2\pi}\int_{C_T} \frac{|z-x_{n0}||z-x_{n1}|...|z-x_{nn}||f(t)|}{|t-x_{n0}||t-x_{n1}|...|t-x_{nn}||t-z|}dt. \tag{8.26}$$

Per $x_{nk} \in R$ e $z \in C_S$, si ha $|z - x_{nk}| < \Delta$; inoltre, per $x_{nk} \in R$ e $t \in C_T$, $|t - x_{nk}| > \delta$. Se

$$M = \max_{t \in C_T} |f(t)| \quad \text{e} \quad d = \min_{\substack{t \in C_T \\ z \in S}} |t - z|,$$

allora:

$$|E_n(z)| \leq \frac{1}{2\pi}\int_{C_T} \frac{\Delta^{n+1}}{\delta^{n+1}}\frac{M}{d}ds = \frac{M\ L(C_T)}{2\pi d}\left(\frac{\Delta}{\delta}\right)^{n+1} \tag{8.27}$$

dove $L(C_T)$ è la lunghezza di C_T. La stima sussiste, uniformemente, per $z \in S$. Per ipotesi, $\frac{\Delta}{\delta} < 1$, quindi:

$$\lim_{n\to\infty} E_n(z) = 0,$$

uniformemente in S. Resta, dunque, soddisfatta la tesi. ■

♣ **Esempio 8.10.** Si consideri $f(x) = (e^{x^2-4} - 1)^{\frac{1}{3}}$ nell'intervallo $I = [-1, 1]$. Si vuole dimostrare, utilizzando il Teorema 8.3.7, che la successione dei polinomi interpolanti, determinata dai nodi equidistanziati, nell'intervallo $[-1, 1]$, con passo di disretizzazione $h = 2/n$, converge uniformemente a f. Se

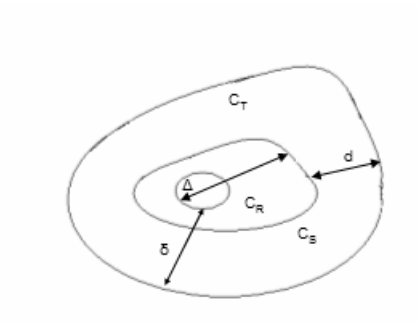

Figura 8.9: Disposizione dei luoghi $R \subset S \subset T$ nel piano complesso $\mathbb{C}$ e rappresentazione grafica di d, δ e Δ.

si riguarda f nel campo complesso, essa è una funzione analitica sull'insieme $T = \{z \in \mathbb{C} \ : \ |z| \leq 2\}$. Allora:

$$|f(z)| = |e^{z^2-4} - 1|^{\frac{1}{3}} \leq (|e^{z^2-4}| + 1)^{\frac{1}{3}} = (e^{Re(z^2-4)} + 1)^{\frac{1}{3}} = (e^{x^2-y^2-4} + 1)^{\frac{1}{3}}$$

da cui:

$$M = \max_{z \in C_T} |f(z)| \leq \ (e^{4-4} + 1)^{\frac{1}{3}} = 2^{\frac{1}{3}}.$$

Se $R = \{z \in \mathbb{C} \ : \ |z| \leq 1\}$ e $S = \{z \in \mathbb{C} \ : \ |z| \leq 2 - \epsilon, \ 0 < \epsilon < 1\}$, si ha:

$$|E_n(z)| \leq \frac{M \ L(C_T)}{2\pi\epsilon} \left(\frac{\Delta}{\delta}\right)^{n+1} = \frac{2^{\frac{1}{3}} 4\pi}{2\pi\epsilon} (1-\epsilon)^{n+1}, \quad \forall z \in S$$

da cui:

$$|E_n(X)| \leq \frac{2^{\frac{4}{3}}}{\epsilon} (1-\epsilon)^{n+1}$$

ovvero $E_n(X)$ tende a zero per $n \to +\infty$ (vedi Figura 8.10).

♣

Definizione 8.3.1. *Siano $\xi_1, \xi_2, ..., \xi_n \in \mathbb{C}$, non necessariamente distinti. Per ogni $\varrho > 0$ l'insieme dei punti $z \in \mathbb{C}$ che soddisfano l'equazione:*

$$|(z - \xi_1)(z - \xi_2)...(z - \xi_k)| = \varrho^k \tag{8.28}$$

è detto ***lemniscata*** *di raggio ϱ e fuochi $\xi_1, \xi_2, ..., \xi_k$. L'insieme dei punti di $\mathbb{C}$ che soddisfano la diseguaglianza:*

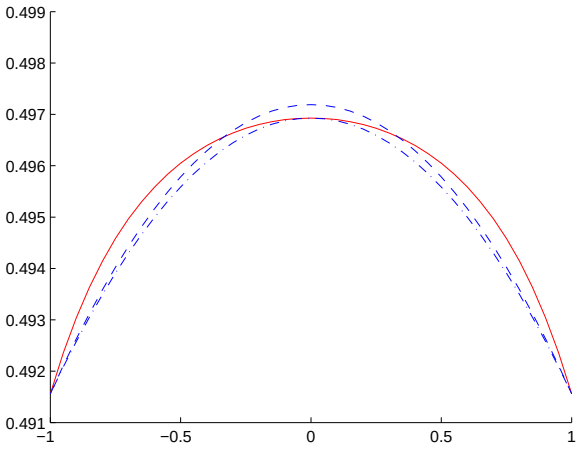

Figura 8.10: Grafico della funzione $f(x) = (e^{x^2-4} - 1)^{1/3}$ (linea continua) e dei polinomi di Lagrange interpolanti la f su nodi equidistanziati, con $h = 2/n$, di grado $n = 2$ ('-.') e di grado $n = 3$ ('- -').

$$|(z - \xi_1)(z - \xi_2)...(z - \xi_k)| < \varrho^k \tag{8.29}$$

è detto ***interno della lemniscata*** *e si indica con* $\mathcal{L}_\varrho$.

♣ **Esempio 8.11.** Si consideri il luogo dei punti $z = x + iy \ \in \mathbb{C}$, che soddisfano l'equazione:

$$[(x - a)^2 + y^2][(x + a)^2 + y^2] = b^4.$$

Tale luogo è una lemniscata di fuochi $\xi_1 = -a$ e $\xi_2 = a$, appartenenti a $\Re$, e raggio $\varrho = b$:

$$|(z - \xi_1)(z - \xi_2)| = \varrho^2,$$

che in coordinate polari diventa:

$$r = \pm a\sqrt{cos(2\theta) \pm \sqrt{\left(\frac{b}{a}\right)^4 - sin^2(2\theta)}} \tag{8.30}$$

definita per i valori di $\theta \in \Re$, per i quali entrambi i radicandi della (8.30) risultano positivi.
Se $0 < \varrho < \frac{1}{2}|\xi_2 - \xi_1|$, ovvero se $b < a$, otteniamo due ovali che cicondano i fuochi, detti **Ovali di Cassini**. Se $0 < \varrho = \frac{1}{2}|\xi_2 - \xi_1|$, ovvero se $b = a$, otteniamo un'unica curva chiusa, detta **lemniscata di Bernoulli**, che circonda i due fuochi, con un punto doppio in $\frac{1}{2}|\xi_1 + \xi_2| = \frac{1}{2}|-a + a| = 0$. Se

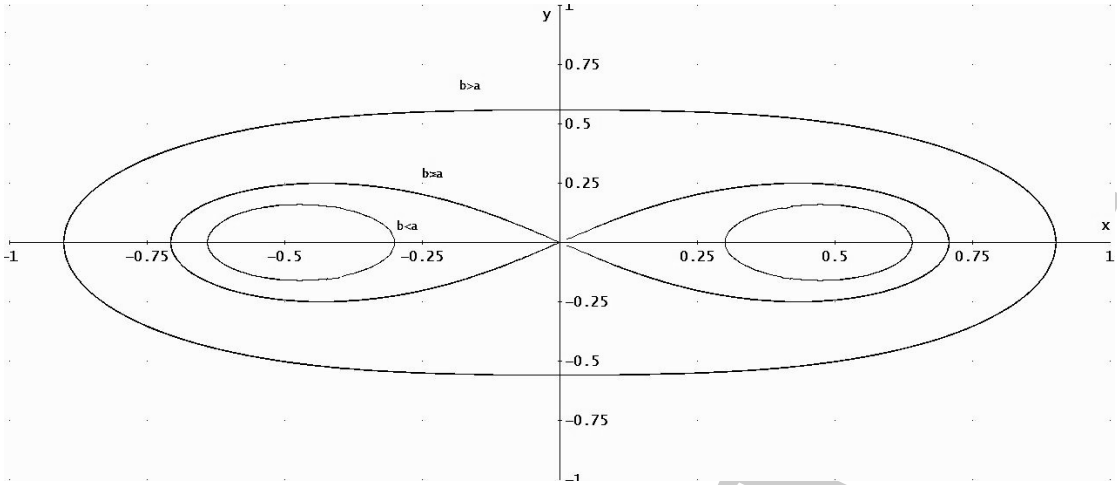

Figura 8.11: Le lemniscate di fuochi $\xi_1 = -0.5$ e $\xi_2 = 0.5$, di raggio rispettivamente $b = 0.4 < a$, $b = 0.5 = a$ e $b = 0.6 > a$.

$\varrho > \frac{1}{2}|\xi_2 - \xi_1|$, ovvero se $b > a$, si ottiene una quartica di tipo ellittico che circonda i due fuochi (Figura 8.11).

♣

Si fornisce ora un teorema di convergenza per i polinomi interpolanti quando i nodi di interpolazione appartengono all'interno di una lemniscata.

Teorema 8.3.8. *Sia $\mathcal{L}_\varrho$ l'interno della lemniscata di raggio ϱ e fuochi $\xi_1, \xi_2, \ldots, \xi_k$:*

$$\mathcal{L}_\varrho = \{z \in \mathbb{C} : |(z-\xi_1)(z-\xi_2)...(z-\xi_k)| < \varrho^k\} \tag{8.31}$$

e siano $x_0, x_1, \ldots, x_n$, $n \in \mathcal{N}$, punti di $\mathcal{L}_\varrho$ appartenenti ad una matrice di interpolazione X. Sia f una funzione analitica in $\mathcal{L}_\varrho$ ma non in $\mathcal{L}_{\varrho_1}$, con $\varrho_1 > \varrho$. Condizione sufficiente affinché la successione dei polininomi interpolanti $\{L_n\}_{n=0}^{\infty}$, determinata da X, converga a f in $\mathcal{L}_\varrho$:

$$\lim_{n\to+\infty} L_n(z) = f(z), \quad \forall z \in \mathcal{L}_\varrho. \tag{8.32}$$

è che sia soddisfatto il limite:

$$\lim_{n\to+\infty} |(z-x_0)(z-x_1)...(z-x_n)|^{\frac{1}{n+1}} = |(z-\xi_1)(z-\xi_2)...(z-\xi_k)|^{\frac{1}{k}}, \quad \forall z \in \mathcal{L}_\varrho, \tag{8.33}$$

uniformemente in ogni insieme S limitato, tale che

$$\inf_{t\in S}|x_i - t| > \delta > 0, \qquad i = 0, 1, 2, \ldots, n$$

Inoltre tale convergenza è uniforme in ogni insieme chiuso contenuto in $\mathcal{L}_\varrho$. All'esterno di $\mathcal{L}_\varrho$ il limite non esiste. Più precisamente si ha:

$$E_n(X) \leq M\left(\frac{\varrho'}{\varrho}\right)^{n+1}, \qquad per \quad z \in \mathcal{L}_{\varrho'}, \quad \varrho' < \varrho \tag{8.34}$$

con M costante reale positiva; inoltre:

$$\lim_{n\to\infty} \sup |a_n|^{1/n} = \frac{1}{\rho}, \tag{8.35}$$

avendo posto:

$$L_n(z) = a_0 + a_1(z - x_0) + \cdots + a_n(z - x_0)(z - x_1)\cdots(z - x_{n-1})$$

interpolante f nei nodi $x_0, x_1, \ldots, x_n$.

Dimostrazione Dimostriamo la convergenza uniforme dei polinomi interpolanti all'interno di $\mathcal{L}_\varrho$. Poiché un insieme chiuso S contenuto in $\mathcal{L}_\varrho$ deve giacere anche in qualche $\mathcal{L}_{\varrho'}$, con $\varrho' < \varrho$, allora la (8.34) implica la convergenza uniforme (8.32) in S. Basta provare, dunque, la (8.34). Sia $\varrho' \in \Re^+$, tale che $\varrho' < \varrho$ e sufficientemente vicino a ϱ, così che tutti i punti $x_0, x_1, \ldots, x_n$ cadano in $\mathcal{L}_{\varrho'}$. Sia $\varrho'' \in \Re^+$ tale che $\varrho' < \varrho'' < \varrho$. Utilizzando la stima integrale di Cauchy, si ha:

$$f(z) - L_n(z) = \frac{1}{2\pi i}\int_{\Gamma_{\varrho''}} \frac{(z - x_0)(z - x_1)...(z - x_n)f(t)}{(t - x_0)(t - x_1)...(t - x_n)(t - z)}dt, \quad z \in \mathcal{L}_{\varrho'} \tag{8.36}$$

Dalla (8.33) si ha:

$$\lim_{n\to+\infty} |(z - x_0)(z - x_1)...(z - x_n)|^{\frac{1}{n+1}} = \varrho'$$

uniformemente per $z \in \Gamma'_\varrho$, ovvero sulla frontiera di $\mathcal{L}_{\varrho'}$. Fissato, dunque, un ϵ_1, esiste un ν_1 sufficientemente grande, tale che $\forall n > \nu_1$ si ha :

$$|(z - x_0)(z - x_1)...(z - x_n)| \leq (\varrho' + \epsilon_1)^{n+1}, \quad \forall z \in \Gamma'_\varrho \tag{8.37}$$

e, per il principio del massimo, la disuguaglianza sussiste in tutto $\mathcal{L}_{\varrho'}$. Scegliamo ϵ_1 tale che, posto $\varrho''' = \varrho' + \epsilon_1$, sia $\varrho' < \varrho''' < \varrho''$.
Con considerazioni analoghe, dalla (8.33), si ha:

$$\lim_{n\to+\infty} |(t - x_0)(t - x_1)...(t - x_n)|^{\frac{1}{n+1}} = \varrho'', \quad \forall t \in \Gamma_{\varrho''} \tag{8.38}$$

uniformemente. Fissato, dunque, un ϵ_2, esiste un ν_2 sufficientemente grande, tale che $\forall n > \nu_2$:

$$|(t - x_0)(t - x_1)...(t - x_n)| \geq (\varrho'' - \epsilon_2)^{n+1}, \quad \forall t \in \Gamma_{\varrho''}. \tag{8.39}$$

Scegliamo ϵ_2 tale che, posto $\varrho^{iv} = \varrho'' - \epsilon_2$, sia $\varrho''' < \varrho^{iv} < \varrho''$.
Utilizziamo le (8.37) e (8.39) per stimare la (8.36). Per ogni $n \geq \nu = \max\{\nu_1, \nu_2\}$:

$$|f(z) - L_n(z)| \leq \frac{1}{2\pi}\int_{\Gamma_{\varrho''}} \left(\frac{\varrho'''}{\varrho^{iv}}\right)^{n+1} \frac{|f(t)|}{|t-z|}dt, \quad z \in \mathcal{L}_{\varrho'} \tag{8.40}$$

Sia $L(\Gamma_{\varrho''})$ la lunghezza di $\Gamma_{\varrho''}$, $m = \max_{t\in\Gamma_{\varrho''}} |f(t)|$, δ la minima distanza da $\Gamma_{\varrho'}$ a $\Gamma_{\varrho''}$, considerando che $\varrho' < \varrho''' < \varrho^{iv} < \varrho'' < \varrho$, allora, $\forall n > \max\{\nu_1, \nu_2\}$, definitivamente si ha:

$$|f(z) - L_n(z)| \leq \frac{mL(\Gamma_{\varrho''})}{2\pi\delta}\left(\frac{\varrho'''}{\varrho^{iv}}\right)^{n+1} \leq M\left(\frac{\varrho'}{\varrho}\right)^{n+1}, \quad z \in \mathcal{L}_{\varrho'}, \tag{8.41}$$

con M costante reale positiva, indipendente da n. Dall'arbitrarietà della scelta di ϱ'', con $\varrho' < \varrho'' < \varrho$, si ottiene la tesi (8.34).

Si consideri, infine, il numero μ definito da:

$$\lim_{n\to\infty} \sup |a_n|^{1/n} = \frac{1}{\mu}$$

Proveremo che $\mu = \rho$, dunque la (8.35). Supponiamo, dapprima, che sia $\mu < \rho$. Per quanto provato, il limite $\lim_{n\to+\infty} L_n(z)$ esiste in $\mathcal{L}_\varrho$ ma, per le ipotesi [4], il limite non esiste nei punti z esterni a $\mathcal{L}_\mu$ e distinti dai fuochi $\xi_1, \xi_2, \ldots, \xi_k$. Dunque, nei punti $z(\neq \xi_1, \xi_2, \ldots, \xi_k)$ appartenenti a $\mathcal{L}_\varrho$ ma esterni a $\mathcal{L}_\mu$ il limite esiste e non esiste, conducendo ad un assurdo. Viceversa, se fosse $\mu > \rho$, si proverebbe [5] che la serie dei polinomi $L_n(z)$ converge uniformemente ad una funzione analitica, in ogni sottoregione chiusa di $\mathcal{L}_\mu$. Questo è impossibile, essendo, per ipotesi, $\mathcal{L}_\mu$ la massima lemniscata di analiticità. Questo prova la (8.35) e, dunque, che c'è divergenza all'esterno di $\mathcal{L}_\varrho$. ■

♣ **Esempio 8.12.** Nell'esempio 8.2 si è osservato che la successione dei polinomi $\{L_n\}_{n=0}^{\infty}$, interpolanti la funzione $f(x) = \frac{1}{1+25x^2}$ sui nodi equidistanziati nell'intervallo $I = [-1, 1]$, non converge uniformemente a f su I. Determiniamo almeno un sottointervallo $I' \subset I$ tale che $\{L_n\}_{n=0}^{\infty}$ converga uniformemente a f su I'. A tal fine, si consideri l'interno della lemniscata $\mathcal{L}_\varrho$ definita nel seguente modo:

$$|(z - 0.4)(z + 0.4)| < 0.44^2, \quad (\varrho = 0.44).$$

$f(x)$ è analitica in $\mathcal{L}_\varrho$ perchè quest'ultima non contiene le singolarità $\{-\frac{1}{5}i, \frac{1}{5}i\}$ di f in $\mathbb{C}$, come mostrato anche in Figura 8.12. Dal Teorema 8.3.8, la successione dei polinomi $\{L_n\}_{n=0}^{\infty}$, interpolanti la funzione di Runge, individuata da nodi equidistanziati del tipo $x_0 = -0.6$ e $x_i = -0.6 + hi$, $i = 0, ..., n$ con $h = \frac{2}{n}$, per $n = 1, 2, \ldots$, appartenenti alla parte reale di $\mathcal{L}_\varrho$, ovvero all'intervallo $I' = [-0.6, 0.6]$, converge uniformemente a f, $\forall x \in I'$ (Figura 8.13).

♣

[4] [1]
[5] [1]

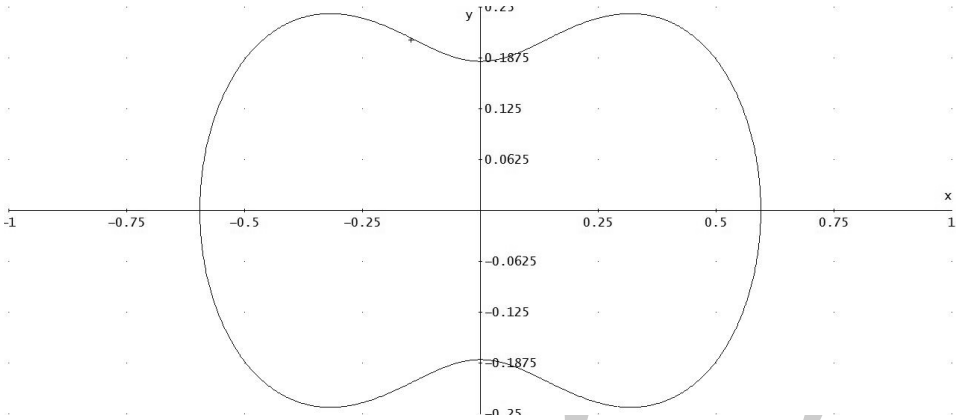

Figura 8.12: La lemniscata di fuochi $\xi_1 = -0.4$ e $\xi_2 = 0.4$ e di raggio $\varrho = 0.44$.

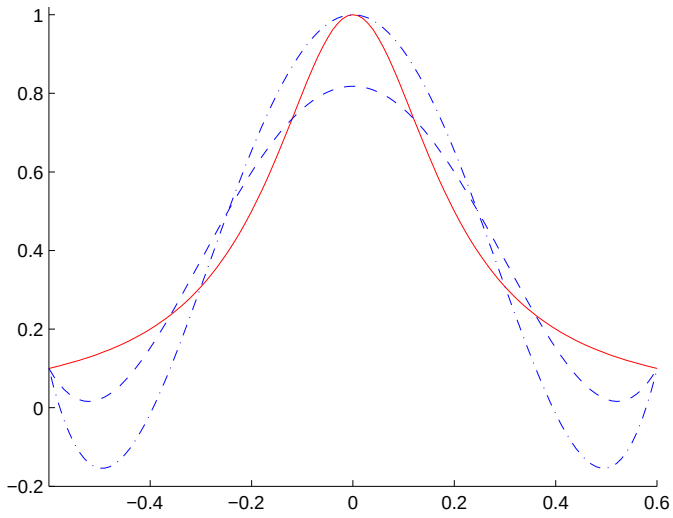

Figura 8.13: La funzione di Runge (linea continua) ed i polinomi di Lagrange interpolanti f su nodi equidistanziati, con $h = 2/n$, nell'intervallo $[-0.6, 0.6]$, di grado $n = 4$ ('-.') e $n = 5$ ('- -').

Il teorema seguente, che conclude questa trattazione, specializza il Teorema 8.3.8 al caso di nodi equidistanziati nell'intervallo $[-1, 1]$.

Teorema 8.3.9. *Sia X una distribuzione di punti equidistanziati appartenenti all'intervallo $[-1, 1]$:*

$$x_k = \frac{k}{n}, \quad (k = 0, \pm 1, ..., \pm n), \quad n \in \mathcal{N}$$

e f una funzione definita nell'intervallo $[-1, 1]$. Condizione necessaria e sufficiente affinché la successione dei polinomi interpolanti $\{L_n\}_{n=0}^{\infty}$ converga a f nell'intervallo $[-1, 1]$ è che f nel campo complesso sia olomorfa all'interno del luogo dei punti $A(z) \subset \mathbb{C}$, con:

$$A(z) \equiv \left\{ z \in \mathbb{C},\, z = x + iy,\, x, y \in \Re : \frac{1}{4} \left| (1+z)^{1+z} (1-z)^{1-z} \right| e^{\pi y} = 1 \right\}$$

la cui equazione in coordinate cartesiane è:

$$\frac{1}{2}(1+x) log[(1+x)^2 + y^2] + \frac{1}{2}(1-x) log[(1-x)^2 + y^2] +$$
$$+\pi |y| - y \left(arctg \frac{y}{1+x} + arctg \frac{y}{1-x} \right) = 2\, log(2)$$

Dimostrazione Si consideri la funzione:

$$\phi(x) = \frac{F_m(z) - F_m(x)}{z - x}, \quad con\ x \in [-1, 1], \tag{8.42}$$

dove $F_m(x)$ è il polinomio composto dai seguente fattori:

$$F_m(x) = (x - x_0)(x - x_1)...(x - x_m), \tag{8.43}$$

con $x_0, x_1, ..., x_m$ appartenenti all'intervallo $[-1, 1]$. La funzione $\phi(x)$ ha grado m, perchè il numeratore di (8.42) è un polinomio divisibile per $z - x$. Lo stesso rimane vero se dividiamo per $F_m(z)$ ottenendo la funzione nelle variabili x e z:

$$\overline{\phi}(x, z) = \frac{F_m(z) - F_m(x)}{F_m(z)(z - x)} = \frac{1}{z - x} - \frac{F_m(x)}{F_m(z)(z - x)} \tag{8.44}$$

Di conseguenza, $\overline{\phi}(x, z)$, considerata come funzione di x, ammette una interpolazione polinomiale "*esatta*", come somma di $m + 1$ addendi, comunque siano distribuiti i nodi $x_0, x_1, ..., x_m$. Il secondo termine della (8.44), invece, svanisce in corrispondenza dei nodi. Tale termine, portato a sinistra della seconda uguaglianza, nella (8.44), si può, dunque, considerare come il resto dell'*interpolazione polinomiale, in corrispondenza di nodi arbitrariamente distribuiti*, della funzione $\frac{1}{z-x}$. Dalla (8.44), infatti, la funzione nella variabile x:

$$v(x) = \frac{1}{z - x}, \quad x \in [-1, 1] \tag{8.45}$$

può essere scritta come:

$$\frac{1}{z-x} = \frac{F_m(z) - F_m(x)}{F_m(z)(z-x)} + \frac{F_m(x)}{F_m(z)(z-x)} \tag{8.46}$$

Per ipotesi X è una distribuzione di $2n+1$ punti equidistanziati:

$$x_k = \frac{k}{n}, \quad (k = 0, \pm 1, ..., \pm n).$$

Posto, allora, $m = 2n+1$, la funzione $F_{2n+1}(x)$ diviene:

$$F_{2n+1}(x) = x\left(x^2 - \frac{1}{n^2}\right)\left(x^2 - \frac{4}{n^2}\right)\cdots\left(x^2 - \frac{n^2}{n^2}\right) \tag{8.47}$$

che scriveremo come:

$$F_{2n+1}(x) = \frac{(2n+1)!}{n^{2n+1}}\,\psi_{2n+1}(x) \tag{8.48}$$

dove:

$$\psi_{2n+1}(x) = nx\frac{(n^2x^2-1)(n^2x^2-4)...(n^2x^2-n^2)}{(2n+1)!} \tag{8.49}$$

Sostituendo, nella (8.46), le (8.48) e (8.49), la funzione (8.45) può essere riscritta come:

$$\frac{1}{z-x} = n\sum_{k=0}^{2n}\frac{\psi_k(x)}{(k+1)\psi_{k+1}(z)} + \eta_{2n+1}(x,z) \tag{8.50}$$

avendo posto $\psi_0(x) = 1$ e:

$$\eta_{2n+1}(x,z) = \frac{\psi_{2n+1}(x)}{\psi_{2n+1}(z)}\frac{1}{z-x}$$

Applicando il Teorema integrale di Cauchy alla funzione $f(x)$, si ha che, $\forall x \in [-1,1]$:

$$f(x) = \frac{1}{2\pi i}\oint\frac{f(z)dz}{z-x} \tag{8.51}$$

dove l'integrazione è estesa ad una curva chiusa Γ che contiene il punto $z = x \in [-1,1]$ ed i punti $z = x_i$ della distribuzione di X, ma non contiene le singolarità di $f(z)$ in $\mathbb{C}$. Sostituendo la (8.50) nella (8.51) si ottiene un sviluppo integrale di $f(x)$ con un resto che assume la forma:

$$\overline{\eta}_{2n+1}(x) = \frac{\psi_{2n+1}(x)}{2\pi i}\oint\frac{f(z)\,dz}{(z-x)\psi_{2n+1}(z)}\,. \tag{8.52}$$

La funzione $\psi_{2n+1}(x)$ al numeratore resta limitata da ± 1 in tutto l'intervallo $[-1,1]$. Per studiare la convergenza della successione interpolante, definita su punti equidistanziati, alla funzione $f(x)$, focalizziamo l'attenzione sulla funzione $\psi_{2n+1}(z)$ posta al denominatore, investigandone le proprietà al crescere di n. Poiché l'andamento di $\psi_{2n+1}(z)$ è proporzionale all'$n-$esima potenza della quantità [6]

$$\left[\frac{(1+z)^{1+z}(1-z)^{1-z}}{4}\right] e^{\pi y},$$

basta studiarne il valore assoluto. Si osserva che

$$A(z) = \left\{ z \in \mathbb{C},\, z = x + iy,\, x, y \in \Re : \frac{1}{4} \left|(1+z)^{1+z}(1-z)^{1-z}\right| e^{\pi y} = 1 \right\}$$

è il luogo dei punti di $\mathbb{C}$ descritto da una curva di tipo ellittico (rappresentata in Figura 8.14), la cui equazione cartesiana è:

$$\frac{1}{2}(1+x)log[(1+x)^2+y^2] + \frac{1}{2}(1-x)log[(1-x)^2+y^2] + \pi|y| - y\left(arctg\frac{y}{1+x} + arctg\frac{y}{1-x}\right) = 2\,log(2)$$

e che interseca l'asse x in -1 e 1.

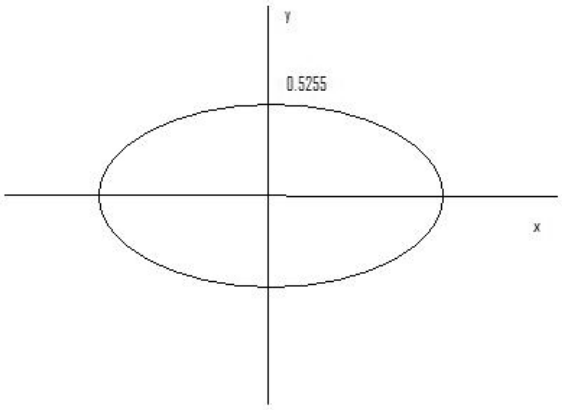

Figura 8.14: $A(z) = \left\{ z \in \mathbb{C},\, z = x + iy,\, x, y \in \Re : \frac{1}{4} \left|(1+z)^{1+z}(1-z)^{1-z}\right| e^{\pi y} = 1 \right\}$.

Si dimostra che, se $A(z) < 1$, allora l'$n-$esima potenza della quantità tende a zero, per $n \to \infty$, dunque il resto nella (8.52) cresce con n; al contrario, se $A(z) > 1$, allora l'$n-$esima potenza della quantità tende a ∞, per $n \to \infty$, dunque il resto nella (8.52) è infinitesimo con n. Con queste considerazioni si prova che l'integrando della (8.52) diverge all'interno della regione $A(z)$ mentre converge verso lo zero al di fuori, per $n \to +\infty$. Conseguentemente, se $f(z)$ è olomorfa all'interno di $A(z)$, per ogni cammino chiuso Γ completamente esterno alla curva $A(z)$, l'integrale (8.52) converge a zero per $n \to +\infty$, quindi la successione dei polinomi interpolanti $\{L_n(x)\}_{n=0}^{\infty}$ su nodi equidistanziati in $[-1,1]$, converge a $f(x)$ su tutto l'intervallo di interpolazione. Se $f(z)$ ammette una o più singolarità all'interno di $A(z)$, la curva Γ sarà contenuta all'interno di $A(z)$, in modo tale da non includere tali singolarità; in tal caso l'integrando in (8.52) diverge per $n \to +\infty$ e, quindi, non si verifica la convergenza dei polinomi interpolanti sull'intervallo $[-1,1]$ a $f(x)$ [7].

[6] [2]

[7] Una applicazione concreta di tale teorema è riportata in [6] e [28] del **Capitolo 2**.

■

♣ **Esempio 8.13.** Il Teorema 8.3.9 prova il carattere divergente della successione $\{L_n\}_{n=0}^{\infty}$, dei polinomi interpolanti la funzione di Runge $f(x) = \frac{1}{1+25x^2}$, su nodi equidistanziati nell'intervallo $[-1, 1]$, del tipo $x_0 = -1$ e $x_i = -1 + hi,\ i = 0, ..., n$ con $h = \frac{2}{n}$, per $n = 1, 2, \ldots$. Infatti le singolarità di f sul campo complesso $\mathbb{C}$ sono:

$$z_1 = +\frac{1}{5}i \quad \text{e} \quad z_2 = -\frac{1}{5}i$$

e cadono all'interno della regione critica $A(z)$, come mostrato in Figura 8.15.

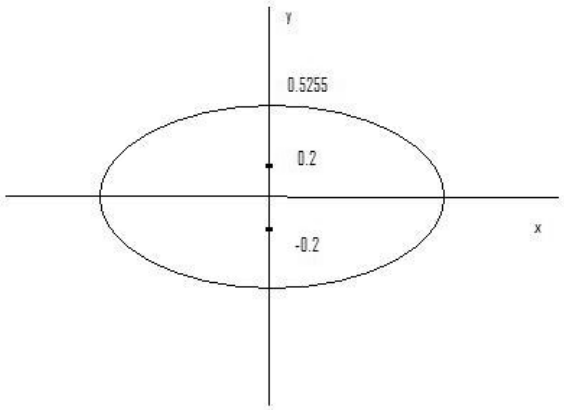

Figura 8.15: Le discontinuità di $f(z)$, $z_1 = +\frac{1}{5}i$ *e* $z_2 = -\frac{1}{5}i$, cadono all'interno di $A(z) = \left\{z \in \mathbb{C},\, z = x + iy,\, x, y \in \Re : \frac{1}{4}\left|(1+z)^{1+z}(1-z)^{1-z}\right| e^{\pi y} = 1\right\}$.

♣

Bibliografia

[1] Davis P. J. - *Interpolation and approximation* - Blaisdell, New York, 1975.

[2] Lanczos C. - *Applied Analysis* - Prentice Hall, Inc, 1956.

[3] Natanson I. - *Constructive Function Theory* - Frederick Ungar Publishing Co., New York, vol. III, 1965.

[4] Rivlin T. - *An Introduction to the Approximation of Functions* - Dover, 1969.

Appendice A

Il calcolo numerico di π

Il problema del calcolo delle cifre di π è stato oggetto di studio sin dall'antichità. Il valore di π ha attirato l'attenzione di molti matematici, dai tempi di Archimede ai nostri giorni e sono state utilizzate diverse metodologie che hanno condotto ad approssimazioni sempre piú accurate del numero π [1].
Di seguito riportiamo alcuni risultati classici, poiché riteniamo utile, dal punto di vista didattico, evidenziare come risultino ancora attuali alcune questioni di calcolo numerico dell'antichità.
Uno dei primi risultati significativi è dovuto ad Archimede (250 a.c.) che riuscì a dimostrare che:

$$3.140845704225352112... = \frac{223}{71} = 3 + \frac{10}{71} < \pi < 3 + \frac{1}{7} = \frac{22}{7} = 3.142857142857142857...$$

approssimando la lunghezza della circonferenza con raggio unitario con il perimetro dei poligoni inscritti e circoscritti, arrivando ad utilizzare poligoni fino a 96 lati.
A partire da Leibniz (1688 d.c), il calcolo delle cifre di π si basa sull'approssimazione della funzione arcotangente. Attualmente, negli algoritmi numerici, il modo più accurato per calcolare un'approssimazione di π secondo la precisione del sistema è

$$\pi = 4 \cdot arctg(1)$$

[1] L'interesse che accomuna i matematici antichi e quelli moderni, nei confronti del calcolo dei decimali di π, è legato sia all'uso che ne hanno fatto le civiltà nel corso dei secoli (basti pensare che già i costruttori egizi stabilirono l'inclinazione delle pareti della piramide di Cheope in modo tale che il rapporto tra l'altezza e la larghezza della base corrispondesse a quello esistente tra il raggio e la circonferenza di un cerchio), che a quello che se ne fa attualmente in numerosi campi, tra i quali, ad esempio, la crittografia; d'altra parte, lo stimolo a determinare sempre più cifre di quello sviluppo infinito che rappresenta lo stesso π, è legato anche alle problematiche ancora aperte, che riguardano l'espressione e la sequenza dei decimali del numero. Ci si chiede, ad esempio, se ciascuna cifra si ripete infinitamente spesso, o se π è *semplicemente normale in base 10*, nel senso che ogni sua cifra appare ugualmente spesso, in senso asintotico, nel suo sviluppo decimale, oppure se è *normale in base 10*, cioè se ogni blocco di cifre di una fissata lunghezza appare ugualmente spesso, in senso asintotico, nel suo sviluppo decimale, o anche se π è *normale*, intendendo con ciò, che ogni blocco di cifre di una fissata lunghezza appare ugualmente spesso, in senso asintotico, nella sua rappresentazione in una qualsiasi base.

che fa uso dell'approssimazione della funzione arcotangente, considerata come funzione elementare (*built-in*) del compilatore.
Nella tabella seguente riportiamo una breve sintesi dello **sviluppo storico** del calcolo delle cifre di π:

Autore	**Anno**	**Risultato**	**Cifre corrette**
Re Salomone	975 a.c	3.0	1
Archimede	250 a.c.	3.14	3
Aryabhata	499 d.c.	3.1416	4
Fibonacci	1220	3.141818	4
Viete	1593	3.141592653	9
Newton	1665	3.1415926535897932	16
Sharp	1699	3.(72 cifre)	71
Machin	1706	3.(100 cifre)	100
Eulero	1755	3.(20 cifre)	20
Vega	1789	3.(140 cifre)	126
Vega	1794	3.(137 cifre)	136
Ferguson	1946	3.(700 cifre)	620

Tra gli autori indicati in tabella vale la pena evidenziare Eulero, che adottò per primo la lettera greca π e che ottenne il risultato in solo un'ora di calcoli!
Il valore ottenuto da Ferguson nel Luglio del 1946 rappresenta l'ultimo calcolo delle cifre di π eseguito senza usare il calcolatore.
Nella tabella seguente si riportano, invece, alcuni dei risultati ottenuti mediante l'uso del calcolatore.

Autore	**Anno**	**Cifre corrette**	**Macchina**
Ferguson	1947	710	Desk calculator
Ferguson e Wrench	1947	808	Desk calculator
Shants e Wrench Jr	1961	100 265	IBM 7090
Guilloud e Filliatre	1966	250 000	IBM 7030
Guilloud e Dichampt	1967	500 000	CDC 6600
Kanada e Ushiro	1983	10 013 395	Hitachi S-810/20
Kanada et al.	1987	134 214 700	NEC SX-2
Kanada e Takahashi	1999	206 158 430 000	Hitachi SR8000
Kanada et al.	2002	1 241 100 000 000	Hitachi SR8000/MP

In particolare il risultato raggiunto da Y. Kanada e Y. Ushiro nel 1983 è il primo ottenuto mediante *supercalcolatore*. Dal 1980 Yasumasa Kanada è uno dei maggiori esponenti, tra gli interessati al calcolo di π con un elevato numero di cifre. La maggior parte dei suoi calcoli è stata realizzata utilizzando supercalcolatori ed è basata su

moderni algoritmi iterativi.

Le prime 151 cifre del numero π calcolate con un elaboratore IBM 704 sono riportate qui di seguito [2]:

$\pi = 3.14159265358979323856264338327$
$95923078164062862089986280348253$
$42117067982148086513282306647$
$09384460955058223172535940812 8 \ldots$

A.1 *Metodo I - Archimede 240 a.C.*

Si consideri una circonferenza di centro l'origine e raggio unitario. Detta A_n l'area del poligono regolare con 2^n lati inscritto nella circonferenza, si ha:

$$\pi = \lim_{n \to \infty} A_n$$

Il metodo proposto da Archimede è quello di approssimare il valore di π con A_n.

Come si può anche notare dalla Figura A.1, per $n = 2$, A_2, l'area del quadrato inscritto nella circonferenza può essere calcolata come 4 volte l'area del triangolo AOB.
Quindi:

$$A_2 = 4 \cdot \frac{AO \cdot BH}{2} = 2^2 \cdot \frac{\sin(\beta_2)}{2}$$

essendo $AO = 1$ perchè raggio di una circonferenza unitaria, $\beta_2 = A\hat{O}B$ e
$BO = BH = \sin(\beta_2)$.
Per $n = 3$, A_3, l'area dell'ottagono inscritto nella circonferenza, A_3 può essere calcolata come 8 volte l'area del triangolo AOB, come illustrato in Figura A.2.

[2] L'ultimo risultato relativo al calcolo delle cifre di π risale al Settembre 2002, quando il Prof. Yasumasa Kanada dell'Università di Tokyo, *Information Technology Center*, insieme ad altri nove collaboratori, calcolò π con 1.241.100.000.000 cifre decimali, superando di più di sei volte il numero delle cifre del loro stesso *Guinness World Record*, costituito da 206.158.430.000 cifre decimali e raggiunto nel 1999. Il calcolo richiese circa 602 ore e fu realizzato con un supercomputer Hitachi (Hitachi SR8000).

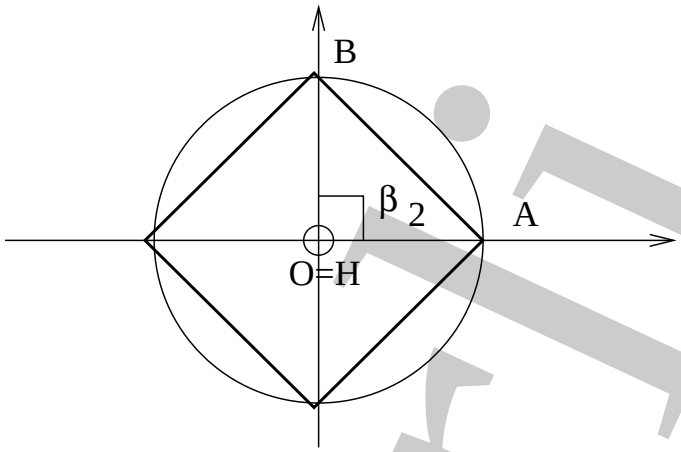

Figura A.1: Quadrato inscritto nella circonferenza di raggio unitario e centro l'origine. L'area del quadrato è $A_2 = 4 \cdot \frac{AO \cdot OB}{2}$.

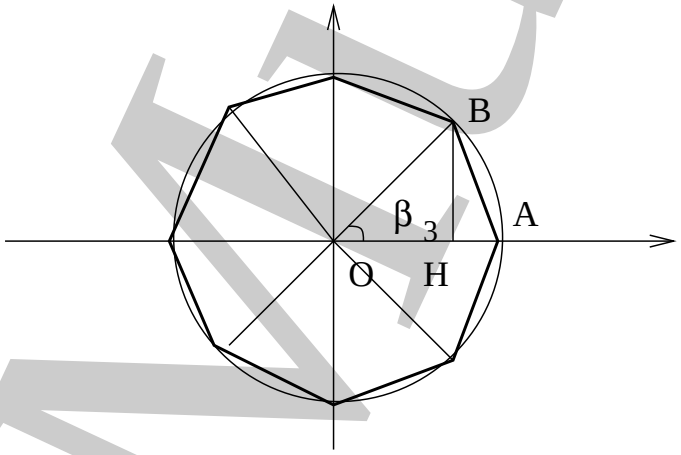

Figura A.2: Per $n = 3$ si ottiene un ottagono inscritto nella circonferenza. L'area dell'ottagono è $A_3 = 8 \cdot \frac{AO \cdot BH}{2}$.

Quindi:

$$A_3 = 8 \cdot \frac{AO \cdot BH}{2} = 2^3 \cdot \frac{1 \cdot \sin(\beta_3)}{2} \qquad (AO = 1,\ \beta_3 = A\hat{O}B)$$

In generale, per un fissato n, l'area del poligono con 2^n lati, A_n, è 2^n volte l'area dei triangoli che lo compongono, e quindi:

$$A_n = \frac{2^n AO \cdot BH}{2} = 2^{n-1} \sin(\beta_n) \qquad (AO = 1,\ \beta_n = A\hat{O}B) \tag{A.1}$$

Posto:

$$b_n = 2^{n-1} \tag{A.2}$$

la (A.1) diventa:

$$A_n = b_n \cdot \sin(\beta_n) \tag{A.3}$$

Il problema del calcolo di A_n è ricondotto al calcolo di $\sin(\beta_n)$.
Nel metodo proposto da Archimede, il valore di $\sin(\beta_n)$ viene calcolato a partire da quello di $\sin(2\beta_n)$ attraverso la formula di bisezione degli angoli:

$$\sin(\beta_n) = \sqrt{\frac{1-\cos(2\beta_n)}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2(2\beta_n)}}{2}} \tag{A.4}$$

dove $0 \leq \beta_n < \pi/2$. Posto $s_n = \sin(\beta_n)$ si ha:

$$s_2 = \sin(\beta_2) = 1$$

e, in generale:

$$\sin(2\beta_n) = \sin\left(2\frac{2\pi}{2^n}\right) = \sin(\beta_{n-1}) = s_{n-1}$$

essendo $\beta_n = \frac{2\pi}{2^n}$. Dalla (A.4) si ottiene la formula ricorrente:

$$s_2 = 1, \qquad s_n = \sqrt{\frac{1-\sqrt{1-s_{n-1}^2}}{2}}\ , \ n > 2$$

Inoltre:

$$b_2 = 2, \qquad b_i = 2 \times b_{i-1,}\ \ i > 2$$

pertanto la (A.3), diventa:

$$A_n = b_n \cdot s_n = 2 \cdot b_{n-1} \cdot \sqrt{\frac{1-\sqrt{1-s_{n-1}^2}}{2}}$$

In conclusione, una stima di π si ottiene attraverso approssimazioni successive ottenute dallo schema iterativo seguente:

$$\begin{cases} b_2 = & 2\,, \quad s_2 = 1 \\ b_i = & 2 \cdot b_{i-1} \\ s_i = & \sqrt{\frac{1-\sqrt{1-s_{i-1}^2}}{2}}, \quad i > 2 \\ A_i = & b_i \cdot s_i, \quad i > 2 \end{cases} \tag{A.5}$$

A.1.1 Analisi della stabilità dell'algoritmo

Volendo implementare lo schema iterativo descritto dalla (A.5) in un sistema aritmetico a precisione finita $\Im$, bisogna tenere conto della propagazione dell'errore di *round-off*. Consideriamo, quindi, i primi i passi dell'algoritmo di Archimede:

$$
\begin{aligned}
&A_2 = b_2 \cdot s_2 = 2 \cdot 1 = 2 \\
&A_3 = b_3 \cdot s_3 = 2b_2 \cdot \sqrt{\tfrac{1-\sqrt{1-s_2^2}}{2}} = 2^2 \cdot \sqrt{\tfrac{1}{2}} = 2^2 \cdot \tfrac{\sqrt{2}}{2} = 2 \cdot \sqrt{2} \\
&A_4 = b_4 s_4 = 2b_3 s_4 = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-s_3^2}}{2}} = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-\frac{1}{2}}}{2}} = 2^2 \cdot \sqrt{2-\sqrt{2}} \\
&A_5 = b_5 s_5 = 2b_4 s_5 = 2^4 \cdot \sqrt{\tfrac{1-\sqrt{1-s_4^2}}{2}} = 2^3 \cdot \sqrt{\tfrac{1-\sqrt{1-\frac{2-\sqrt{2}}{4}}}{2}} = 2^2 \cdot \sqrt{2-\sqrt{2-\sqrt{2}}} \\
&\vdots \\
&A_i = b_i \cdot s_i = 2b_{i-1} \cdot s_i = 2^{i-1} \cdot \sqrt{\tfrac{1-\sqrt{1-s_{i-1}^2}}{2}} = 2^{i-2} \cdot \underbrace{\sqrt{2-\sqrt{2-\ldots-\sqrt{2}}}}_{i-2 \text{ radicali innestati}} \quad , i > 2
\end{aligned}
$$

Si osserva che al passo i-esimo si moltiplica il termine precedente per $\mu = 2^{i-2}$ che è, quindi, anche il fattore di amplificazione dell'errore di round-off. Lo stesso fattore, in base 10, è:

$$\mu \simeq 10^{0.301 \cdot (i-2)}$$

Segue, allora, che l'algoritmo descritto è **instabile**.

A.1.2 Analisi dell'errore di troncamento analitico

Per dare una stima dell' errore di troncamento che si commette arrestando al passo n il processo iterativo descritto, si utilizza lo sviluppo in serie di Mc Laurin della funzione $\sin(x)$.
Lo sviluppo in serie di Mc Laurin della funzione $\sin(x)$ è dato da :

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \tag{A.6}$$

Ricordando la (A.1) si ha:

$$A_n = 2^{n-1} \sin \frac{\pi}{2^{n-1}}$$

e quindi:

$$A_n = 2^{n-1}\left[\frac{\pi}{2^{n-1}} - \frac{1}{6}\frac{\pi^3}{8^{n-1}} + \frac{1}{120}\frac{\pi^5}{32^{n-1}} - \ldots\right] = \pi - \frac{1}{6}\frac{\pi^3}{4^{n-1}} + \frac{1}{120}\frac{\pi^5}{16^{n-1}} - \ldots$$

per l'errore di troncamento analitico si ha:

$$|e_n| = \mathcal{O}\left(\left|\frac{\pi^3}{6} \cdot \frac{1}{4^{n-1}}\right|\right) = \mathcal{O}\left(\left|\frac{2}{3}\pi^3 \cdot \frac{1}{4^n}\right|\right) \approx \mathcal{O}\left(21 \cdot \frac{1}{4^n}\right) = \mathcal{O}\left(\frac{1}{4^n}\right)$$

e:

$$|e_{n+1}| < \frac{1}{4}|e_n|$$

A.2 *Metodo II - Viete 1593*

Si consideri una circonferenza C di centro l'origine e raggio unitario. Indicata con $l(C)$ la lunghezza della circonferenza C, si ha:

$$l(C) = 2 \cdot \pi$$

e:

$$\pi = \lim_{n\to\infty} \frac{P_n}{2}$$

essendo P_n il perimetro del poligono regolare con 2^n lati inscritto nella circonferenza.
Il metodo proposto da Viete è quello di approssimare il valore di π con $P_n/2$.
Per ogni fissato valore di n, si indichi con a_n la lunghezza del lato del poligono di 2^n lati e con p_n il semiperimetro dello stesso poligono, si ottiene:

$$p_n = \frac{2^n a_n}{2} = 2^{n-1} a_n \tag{A.7}$$

Ricaviamo una relazione che lega il lato a_n del poligono di 2^n lati con il lato a_{n+1} del poligono con 2^{n+1} lati.
Per $n = 2$, consideriamo i triangoli AKC e AOB, come illustrato nella Figura A.3. Gli angoli $A\widehat{K}C$ ed $A\widehat{O}B$ sono due angoli, rispettivamente alla circonferenza ed al centro, che insistono sullo stesso arco $(\widehat{AC})$ [3] per cui:

$$A\widehat{K}C = \frac{A\widehat{O}B}{2}, \quad A\widehat{O}B = \frac{\pi}{4} \Rightarrow A\widehat{K}C = \frac{\pi}{8}$$

L'angolo $C\widehat{A}K$ è uguale all' angolo $A\widehat{K}C$, essendo angoli alla base di un triangolo isoscele (ACK); per cui se $\frac{a_2}{2}$ è un cateto del triangolo ABC, a_3 ne è l'ipotenusa, si ha:

[3] Ricordiamo il seguente risultato di geometria:

Teorema A.2.1. *In una circonferenza, l'angolo alla circonferenza è la metà del corrispondente angolo al centro che insiste sullo stesso arco.*

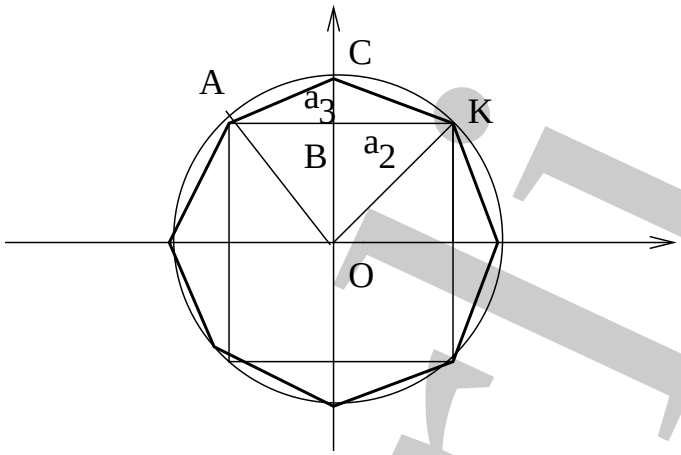

Figura A.3: Metodo di Viete per il calcolo di π. a_2 è il lato del quadrato inscritto e a_3 è il lato dell'ottagono inscritto.

$$a_3 \cdot \cos\left(\frac{\pi}{8}\right) = a_3 \cdot \cos\left(\frac{\pi}{2^3}\right) = \frac{a_2}{2}.$$

In generale, la relazione che lega il lato a_n del poligono di 2^n lati al lato a_{n+1} del poligono con 2^{n+1} lati è la seguente:

$$a_{n+1} \cdot \cos\left(\frac{\pi}{2^{n+1}}\right) = \frac{a_n}{2} \tag{A.8}$$

da cui si ottiene:

$$a_n = 2 \cdot a_{n+1} \cdot \cos\left(\frac{\pi}{2^{n+1}}\right) \tag{A.9}$$

Posto $\beta_i = \frac{\pi}{2^i}$ si ha quindi:

$$p_i = 2^{i-1} a_i = 2^{i-1} \cdot 2 \cdot \cos(\beta_{i+1}) \cdot a_{i+1} = 2^i \cos(\beta_{i+1}) \cdot a_{i+1} = p_{i+1} \cdot \cos(\beta_{i+1}) = p_{i+1} \cdot c_{i+1}$$

da cui:

$$p_{i+1} = \frac{p_i}{c_{i+1}}$$

Riferendoci sempre alla Figura A.3 il triangolo OAB è isoscele e retto in B e l'angolo $O\widehat{A}B = \frac{\pi}{4}$, per cui:

$$\frac{a_2}{2} = \cos\left(\frac{\pi}{4}\right) \quad \Rightarrow \quad a_2 = 2 \cdot 1 \cdot \frac{\sqrt{2}}{2} = \sqrt{2}$$

e:

$$p_2 = 2 \cdot a_2 = 2 \cdot \sqrt{2}$$

Ponendo $c_i = \cos(\beta_i)$ ed utilizzando la formula di bisezione per il coseno:

$$\cos(\beta_i) = \sqrt{\frac{1+\cos(2\beta_i)}{2}} = \sqrt{\frac{1+\cos(\beta_{i-1})}{2}}$$

$$c_1 = \cos\left(\frac{\pi}{2}\right) = 0$$

e:

$$c_i = \sqrt{\frac{1+c_{i-1}}{2}}, \quad i \geq 2$$

In conclusione, resta individuata la seguente formula ricorrente:

$$\begin{cases} p_2 = & 2 \cdot \sqrt{2} \\ c_2 = & \sqrt{\frac{1}{2}} = \frac{\sqrt{2}}{2} \\ p_{i+1} = & \frac{p_i}{\sqrt{\frac{1+c_i}{2}}}, \qquad i \geq 2 \end{cases}$$

A.2.1 Analisi della stabilità dell'algoritmo

Essendo:

$$\frac{\sqrt{2}}{2} < c_i = \cos\left(\frac{\pi}{2^i}\right) < 1, \qquad i \geq 2$$

e quindi

$$1 < \frac{1}{c_i} < \sqrt{2}$$

cioè il fattore di amplificazione dell'errore nella formula di ricorrenza

$$p_{i+1} = \frac{p_i}{c_{i+1}}$$

è limitato e quindi il metodo di Viete per il calcolo di π è **stabile**.

A.2.2 Analisi dell'errore di troncamento analitico

Per determinare una stima dell'errore di troncamento analitico commesso ad ogni passo n si osserva che:

$$a_n = 2 \cdot \sin\frac{2\pi}{2^{n+1}} \quad \Rightarrow \quad p_n = 2^{n-1} a_n \; = \; 2^n \sin\frac{\pi}{2^n}$$

Utilizzando lo sviluppo in serie di Taylor della funzione seno, calcolato in $\frac{\pi}{2^n}$, si ha:

$$p_n \; = \; 2^n \left[\frac{\pi}{2^n} - \frac{1}{6}\frac{\pi^3}{8^n} + \frac{1}{120}\frac{\pi^5}{32^n} - \dots\right] = \pi - \frac{1}{6}\frac{\pi^3}{4^n} + \frac{1}{120}\frac{\pi^5}{16^n} - \dots$$

da cui:

$$|e_n| < \left| \frac{1}{6}\frac{\pi^3}{4^n} + \frac{1}{120}\frac{\pi^5}{8^n} - \dots \right| = \mathcal{O}\left(\frac{1}{6}\pi^3 \cdot \frac{1}{4^n}\right) = \mathcal{O}\left(\frac{1}{4^n} \cdot 5.1677\dots\right) = \mathcal{O}\left(\frac{1}{4^n}\right)$$

A.3 *Metodo III - Leibniz 1688*

Il valore di π è ottenuto dallo sviluppo in serie di Mc Laurin della funzione $arctg(x)$ [4]:

$$arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

Per $x = 1$ si ha, infatti:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

indicata con q_i la somma parziale dei primi i addendi è possibile scrivere la formula ricorrente seguente:

$$q_1 = 1, \quad q_i = q_{i-1} + \frac{(-1)^{i-1}}{2i-1} \tag{A.10}$$

dove $i = 2, \dots, n-1$.

A.3.1 Analisi della stabilità dell'algoritmo

In un sistema aritmetico a precisione finita $\Im$, l'errore di round-off E_n, nel calcolo di una somma di n numeri x_i, si può stimare attraverso la formula seguente:

$$|E_n| \leq (n-1)u \sum_{i=1}^{n} |x_i| + \mathcal{O}(u^2), \quad \text{con} \quad u \textit{ massima accuratezza}$$

e, quindi, cresce linearmente con il numero degli addendi, n. Il metodo di Leibniz per il calcolo di π è, dunque, **stabile**.

[4] La funzione $arctg(\cdot)$ fornita dal compilatore di un linguaggio di programmazione evoluto (Fortran, C, C++,...) non viene calcolata facendo uso dello sviluppo in serie di Mc Laurin (cfr. **Capitolo 7**).

A.3.2 Analisi dell'errore di troncamento analitico

Utilizzando la formula ricorrente espressa nella (A.10), si ricava che l'errore di troncamento analitico è:

$$|e_n| = \mathcal{O}((2n)^{-1})$$

A.4 *Metodo IV - Integrazione Numerica*

Il valore di π è calcolato utilizzando la formula di quadratura trapezoidale.
Infatti la funzione che definisce l'arco di circonferenza unitaria nel primo quadrante è:

$$f(x) = \sqrt{1 - x^2}$$

Per cui integrando $f(x)$ in $[0, 1]$ si ottiene $\frac{\pi}{4}$.

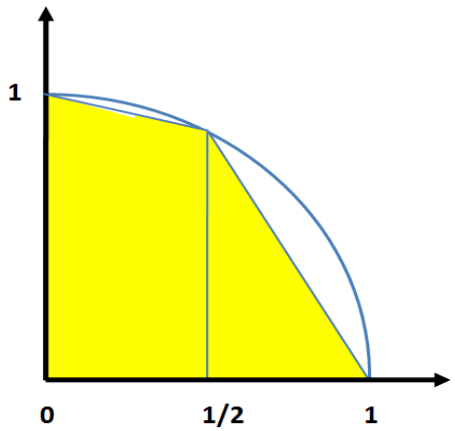

Figura A.4: Metodo di integrazione numerica.

Utilizzando la formula di quadratura trapezoidale composita per il calcolo numerico dell'integrale si ottiene:

$$\frac{\pi}{4} \simeq s_n = \frac{1}{2^n}[1 + 2b_1 + 2b_2 + \cdots + 2b_{n-1}]$$

dove:

$$b_i = \sqrt{1 - \frac{i^2}{n^2}}$$

A.4.1 Analisi della stabilità dell'algoritmo

In un sistema aritmetico a precisione finita $\Im$, l'errore di round-off E_n, nel calcolo della somma s_n, si può stimare nel modo seguente:

$$|E_n| \leq \frac{1}{2^{n-1}}(n-1)u\sum_{i=1}^{n-1}|b_i| + \mathcal{O}(u^2)$$

e, quindi, cresce linearmente con n, numero degli addendi. Il metodo di integrazione numerica è, dunque, **stabile**.

A.4.2 Analisi dell'errore di troncamento analitico

L'errore di troncamento analitico che si commette è:

$$|e_n| < k \cdot n^{-3/2}$$

essendo k una costante positiva.

A.5 Algoritmi implementati in FORTRAN

Di seguito vengono proposti alcuni programmi in Fortran relativi agli algoritmi descritti nel paragrafo precedente. I test hanno l'obiettivo di mettere in evidenza gli effetti della propagazione dell'errore sul risultato finale. Come vedremo, utilizzando il metodo di Archimede, per $n = 7$ il risultato è accurato a 4 cifre significative, ma già per $n = 14$, il risultato calcolato è completamente errato. Ciò a causa dell'instabilità dell'algoritmo. Con il metodo di Viete, invece, l'errore introdotto si mantiene limitato e si ottiene un valore di π corretto a 8 cifre significative per $n = 16$. Infine, se si utilizza il metodo di Leibniz, sebbene l'algoritmo sia stabile, a causa della lenta velocità di convergenza dello sviluppo in serie di McLaurin, solo per $n = 500000$ si ottiene un risultato accurato a 6 cifre significative.

Programma Fortran per il calcolo di π con il metodo di Archimede

```
=================================================
    program  CalcoloMetodoArchimede
C Fase  di dichiarazione delle variabili.
     integer n,step
     real  p,bi,si,error,pi,error1,n1
C Stampa a video del problema da risolvere
  write(*,*)    '                                          '
  write(*,*)    '  Calcolo di PI                           '
  write(*,*)    '  (1)   pi=b(i-1)s(i-1)                   '
  write(*,*)    '  (2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)   '
  write(*,*)    '                                          '
  write(*,*)    '                                          '
     write(*,*) 'Inserire n'
     read*,n
     print*,n
C Fase di  Calcolo
     bi=2.
     si=1.
     step=0
     p=0

5    continue
     p=bi*si
     bi=2*bi
     si=sqrt((1-sqrt(1-si**2))/2)
     step=step+1
C Terminazione della fase di Calcolo.
     if(step.lt.n) goto 5
     n1=n
     error=4**n1
     error=21*1/error
     pi=4*atan(1.)
     error1=abs(pi-p)/pi
     print*, 'PI calcolato=',p
     print*, 'Errore analitico stimato < di',error
C Confronto con la funzione di libreria
     print*, 'PI mediante funzioni built-in =',pi
     print*, 'Errore relativo',error1
     stop
     end
=================================================
```

Prova di esecuzione

Consideriamo alcune prove di esecuzione dell'algoritmo proposto da Archimede.

```
==================================================

Calcolo di PI
(1)   pi=b(i-1)s(i-1)
(2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

Inserire n
7
  PI calcolato=  3.1412857
  Errore analitico stimato < di  0.0012817383
  PI mediante funzione built-in=  3.1415927
  Errore relativo  0.000097747594
==================================================

==================================================

Calcolo di PI
(1)   pi=b(i-1)s(i-1)
(2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

Inserire n
14
  PI calcolato=  2.828427(*)
  Errore analitico stimato < di  7.8231096E-8
  PI mediante funzioni built-in=  3.1415927
  Errore relativo  0.099683724
==================================================
```

(*) Il risultato della seconda esecuzione è del tutto errato. Dunque, nel sistema aritmetico del calcolatore, che esegue le operazioni in registri a doppia precisione, per $n = 14$ l'errore di *round off* è del 100% (algoritmo instabile).

Eseguendo l'implementazione in doppia precisione l'instabilità dell'algoritmo comporta un errore di *round off* del 100% per $n = 34$; si ottengono, infatti, i risultati seguenti:

```
==================================================

Calcolo di PI
(1)   pi=b(i-1)s(i-1)
(2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)
```

```
 Inserire n
 7
   PI calcolato=  3.141277250932773
   Errore analitico stimato < di  0.00128173828125
   PI mediante funzione built-in=  3.141592653589793
   Errore relativo  0.00010039578385817655
================================================

================================================

 Calcolo di PI
 (1)   pi=b(i-1)s(i-1)
 (2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

 Inserire n
 14
   PI calcolato=  3.1415926343386995
   Errore analitico stimato < di  7.82310962677002E-8
   PI mediante funzioni built-in=  3.141592653589793
   Errore relativo  6.127813415313911E-9
================================================
================================================

 Calcolo di PI
 (1)   pi=b(i-1)s(i-1)
 (2)   s(i)=sqrt(1-sqrt(1-s(i-1)^2)/2)

 Inserire n
 34
   PI calcolato=  4.
   Errore analitico stimato < di  7.115076756936123E-20
   PI mediante funzioni built-in=  3.141592653589793
   Errore relativo  0.2732395447351627
================================================
```

Programma Fortran per il calcolo di π con il metodo di Viete

```
================================================
    program  CalcoloMetodoViete
C Fase  di dichiarazione delle variabili.
     integer n,step
```

```
      real  p,bi,si,error,pi,error1,n1
C Stampa a video del problema da risolvere
  write(*,*)    '                                        '
  write(*,*)    '    Calcolo di PI                       '
  write(*,*)    '    (1)   pi=p(i-1)/ci                  '
  write(*,*)    '    (2)   c(i)=sqrt(1-c(i-1))/2 )       '
  write(*,*)    '                                        '
  write(*,*)    '                                        '

      write(*,*) 'Inserire n'
      read*,n
C Fase di  Calcolo
      ci=0.
      per=2.
      step=1
      p=2.


5   continue


      step=step+1
      ci=sqrt((1+ci)/2)
      p=per/ci
      per=p
C Terminazione della fase di Calcolo.
       if(step.lt.n) goto 5
       n1=n
       error=4**n1
       print*,'errore=',error
       error=5.2*1/error
       pi=4*atan(1.)
       error1=abs(pi-p)/pi
       print*, 'PI calcolato=',p
       print*, 'Errore analitico stimato < di',error
C Confronto con la funzione di libreria
         print*, 'PI mediante funzioni built-in =',pi
         print*, 'Errore relativo',error1
         stop
         end
```

Prova di esecuzione

Consideriamo alcune prove di esecuzione dell'algoritmo proposto da Viete.
Per $n = 7$ si ottiene un'accuratezza di 4 cifre significative corrette e per $n = 16$ il risultato ha 8 cifre significative corrette.

```
================================================

 Calcolo di PI
 (1)   pi=p(i-1)/ci
 (2)   c(i)=sqrt(1-c(i-1))/2 )

 Inserire n
           7
   PI calcolato=   3.1412773
   Errore analitico stimato < di  0.0003173828
   PI mediante funzioni built-in =   3.1415927
   Errore relativo  0.00010040378
================================================

================================================

 Calcolo di PI
 (1)   pi=p(i-1)/ci
 (2)   c(i)=sqrt(1-c(i-1))/2 )

 Inserire n
          16
   PI calcolato=   3.1415927
   Errore analitico stimato < di  1.2107193E-9
   PI mediante funzioni built-in =   3.1415927
   Errore relativo  0.
================================================
```

Programma Fortran per il calcolo di π con il metodo di Leibniz

```
================================================
     program  CalcoloMetodoLeibniz

C Fase  di dichiarazione delle variabili.

   integer z
```

```
      real  p,i,error,pi,error1,n1,step,n

C Stampa a video del problema da risolvere

      write(*,*)    '                                                  '
      write(*,*)    '  ------------------------------------------------'
      write(*,*)    '   Calcolo di PI                                  '
      write(*,*)    '    Sviluppo in serie di Taylor di arctg(x)       '
      write(*,*)    '                                                  '
      write(*,*)    '  ------------------------------------------------'
       write(*,*) 'Inserire ordine dello sviluppo'
       read*,n

C Fase di  Calcolo

        qi=1.
        step=1

5       continue

        step=step+1
        i=(-1)**(step-1)
        i=i/(2*step-1)
        qi=qi+i

C Terminazione della fase di Calcolo.

         if(step.le.n-1) goto 5
         p=4*qi
         n1=n
         error=1/n1
         error=2*error
         pi=4*atan(1.)
         error1=abs(pi-p)/pi
         print*, 'PI calcolato=',p
         print*, 'Errore analitico stimato < di',error

C Confronto con la funzione di libreria

        print*, 'PI mediante funzioni built-in =',pi
        print*, 'Errore relativo',error1
        stop
        end
=============================================
```

Prova di esecuzione

Consideriamo alcune prove di esecuzione dell'algoritmo proposto da Leibniz.

```
================================================

------------------------------------------
Calcolo di PI
 Sviluppo in serie di Taylor di arctg(x)

------------------------------------------

Inserire ordine dello sviluppo
70
  PI calcolato= 3.1273072
  Errore analitico stimato < di 0.028571429
  PI mediante funzioni built-in = 3.1415927
  Errore relativo 0.0045472365
================================================


================================================

------------------------------------------
Calcolo di PI
 Sviluppo in serie di Taylor di arctg(x)

------------------------------------------

Inserire ordine dello sviluppo
200
  PI calcolato= 3.1365926
  Errore analitico stimato < di 0.01
  PI mediante funzioni built-in = 3.1415927
  Errore relativo 0.0015915858
================================================


================================================

------------------------------------------
Calcolo di PI
 Sviluppo in serie di Taylor di arctg(x)

------------------------------------------

Inserire ordine dello sviluppo
700
  PI calcolato= 3.140165
  Errore analitico stimato < di 0.0028571428
  PI mediante funzioni built-in= 3.1415927
  Errore relativo 0.00045443524
================================================


================================================
```

```
 Calcolo di PI
  Sviluppo in serie di Taylor di arctg(x)

 Inserire ordine dello sviluppo
 500000
   PI calcolato= 3.141594
   Errore analitico stimato < di 0.000004
   PI mediante funzioni built-in = 3.1415927
   Errore relativo 3.7945495E-7
==================================================
```

A causa della lenta convergenza dello sviluppo in serie di Taylor, solo con 500000 passi si ottengono 6 cifre significative corrette.

Bibliografia

[1] http://numbers.computation.free.fr/Constants/constants.html

[2] http://www.geocities.com/hjsmithh/Pi/index.html

[3] http://www.super-computing.org/

[4] O'Connor J. J. , Robertson E. F. -*A chronology of Pi* - September 2000, World Wide Web site at the adress: http://www−groups.dcs.st−and.ac.uk/∼history/HistTopics/Pi_chronology.html

[5] O'Connor J. J. , Robertson E. F. -*A history of Pi* - August 2001, World Wide Web site at the adress: http://www−groups.dcs.st−and.ac.uk/∼history/HistTopics/Pi_through_the_ages.html

[6] Sebah P. , Gourdon X. - π *and its computation through the ages* - April 2003, World Wide Web site at the adress: http://numbers.computation.free.fr/Constants/Pi/piCompute.html

Appendice B

Approfondimenti sulla generazione delle funzioni elementari

B.1 Alcuni richiami

La costruzione del polinomio di migliore approssimazione uniforme si riconduce al calcolo della soluzione minimax di un sistema lineare sovradeterminato. A tale proposito, i teoremi seguenti [1] forniscono utili caratterizzazioni.
Sia $\overline{I}(=\overline{I}(\mathbf{c})) \subseteq \{1, 2, \ldots, m\}$ l'insieme degli indici i corrispondenti alle componenti di $\mathbf{r} \in \Re^m$ che sono uguali (in modulo) al massimo del vettore $\mathbf{r}(\mathbf{c}, \mathbf{x})$; posto $\theta_i = sign(r_i)$ allora

$$r_i = \theta_i \|\mathbf{r}\|, \qquad i \in \overline{I}.$$

Teorema B.1.1. *Un vettore* $\mathbf{c} \in \Re^{n+1}$ *soddisfa la*

$$\mathbf{c} = \underset{\mathbf{c}\in\Re^{n+1}}{arg\ min} \|\mathbf{r}(\mathbf{c}, \mathbf{x})\| = \underset{\mathbf{c}\in\Re^{n+1}}{arg\ min} \max_{1\le i\le m} |\mathbf{r}(\mathbf{c}, x_i)| \tag{B.1}$$

se, e solo se, esiste $I \subseteq \overline{I}$*, ed un vettore non nullo* $\lambda = (\lambda_1, \ldots, \lambda_m) \in \Re^m$*, tale che*

$$\begin{aligned} \lambda_i &= 0 \qquad i \notin I \\ A^T\lambda &= \mathbf{0} \\ \lambda_i\theta_i &\geq 0 \qquad i \in I. \end{aligned} \tag{B.2}$$

Il sottoinsieme I *è detto* ***sottoinsieme estremale****.*

Corollario B.1.1. *Se* $\mathbf{c}$ *soddisfa la (B.1), allora è soluzione del medesimo problema in* $\Re^t$*, definito ponendo* $b_i = f(x_i)$ $i = 1, 2, \ldots, t$*, mentre* A *è la matrice di dimensione* $t \times n$ *avente come* $i-esima$ *riga il vettore* $[g_1(x_i), \ldots, g_{n+1}(x_i)]$*,* $i = 1, 2, \ldots, t$*.*

[1] Per la dimostrazione si veda [2].

Teorema B.1.2. *Se A ha rango t, una soluzione $\mathbf{c}$ della (B.1) esiste sempre, con $\overline{I}$ contenente almeno $(t+1)$ interi.*

Teorema B.1.3. *Se A ha rango t e J è un sottoinsieme di dimensione $(t+1)$ di $\{1, 2, \ldots, m\}$, allora*

$$\min_{\mathbf{c}} \left\{ \max_{i \in J} |r_i(\mathbf{c})| \right\} \leq \min_{\mathbf{c}} \|\mathbf{r}(\mathbf{c})\|,$$

dove l'uguaglianza sussiste quando J è un **sottoinsieme estremale** *di $\{1, 2, \ldots, m\}$.*

Sia $\overline{E}(= \overline{E}(\mathbf{c}))$ l'insieme degli $\mathbf{x} \in X$, con $X = [a, b]$ intervallo in cui costruire la soluzione del problema di approssimazione minimax, per i quali, fissato $\mathbf{c}$,

$$|\mathbf{r}(\mathbf{c}, \mathbf{x})| = \|\mathbf{r}(\mathbf{c}, \mathbf{x})\|.$$

Il seguente teorema, l'analogo del **Teorema B.1.1** e dovuto a Kirchberger (1903), fornisce una condizione necessaria e sufficiente al calcolo del vettore $\mathbf{c}$.

Teorema B.1.4. *Un vettore $\mathbf{c} \in \Re^{n+1}$ è soluzione del problema di approssimazione minimax (B.1) se, e solo se, esiste $E \subset \overline{E}$ contenente $t \leq n+1$ punti $x_1, x_2, \ldots, x_t$, ed un vettore non nullo $\lambda \in \Re^t$, tale che*

$$\begin{aligned} \sum_{i=1}^{t} \lambda_i g_j(\mathbf{x}_i) &= 0 \qquad j = 1, 2, \ldots, n, \\ \sum_{i=1}^{t} \lambda_i g_j(x_i) &= 0 \qquad j = 1, 2, \ldots, n, \\ \lambda_i \theta_i &\geq 0 \qquad i = 1, 2, \ldots, t, \end{aligned}$$

dove $\theta_i = sign(r(\mathbf{c}, x_i)),\ i = 1, 2, \ldots, t.$

B.2 Esempio di studio: generazione della funzione esponenziale

Riprendiamo l'esempio di studio del paragrafo 7.2.3, relativo alla valutazione di e^x, $x \in (-\infty, \infty)$. Sia $X = fl(x)$, $R = fl(r)$.

Passo 1 Fissato $L = 5$, si scala x nell'intervallo

$$I = \left[-\frac{\log 2}{64}, \frac{\log 2}{64}\right] \simeq [-0.108 \times 10^{-1}, 0.108 \times 10^{-1}],$$

ponendo

$$X = \frac{(32 \cdot m + j)\log 2}{32} + R, \quad R \in I. \tag{B.3}$$

Il primo passo consiste nel calcolare i valori di m e j. Moltiplichiamo l'equazione (B.3) per la costante $32/\log 2$. Si ottiene

$$X \cdot \frac{32}{\log 2} = (32 \cdot m + j) + R \cdot \frac{32}{\log 2}. \tag{B.4}$$

Poiché

$$R \in I \Leftrightarrow |R| \leq \frac{\log 2}{64}$$

si ha

$$R \cdot \frac{32}{\log 2} \leq 0.5.$$

Dalla (B.4) segue che $32 \cdot m + j$ rappresenta la parte intera di $X \cdot \frac{32}{\log 2}$. Ovvero, se indichiamo con

$$N = \left\lfloor X \cdot \frac{32}{\log 2} \right\rfloor$$

la parte intera di $X \cdot \frac{32}{\log 2}$, risulta che

$$N = 32 \cdot m + j$$

ed, ancora, che

$$N = N_1 + N_2$$

avendo posto $N_1 = 32 \cdot m$ e $N_2 = j$. Da cui:

$$\begin{aligned} m &:= N_1/32 \\ j &:= N_2 \end{aligned}$$

Il primo passo consiste, quindi, nel calcolare N, N_1, N_2:

$$\begin{aligned} N &:= \texttt{INTRND}(X \cdot Inv\_L) \\ N_2 &:= N \, mod \, 32 \\ N_1 &:= N - N_2 \end{aligned}$$

dove, in particolare, `Inv_L` è la costante floating point uguale a $32/\log 2$, arrotondata alla precisione della macchina, e `INTRND` è la funzione dell'aritmetica dell'IEEE, che arrotonda un numero floating point all'intero più vicino, secondo

la modalità di default dello stesso standard (*round-to-nearest integer*). I valori assunti da `Inv_L` sono riportati in [1].
L'argomento R è espresso come somma di due numeri macchina, nella precisione fissata (*working-precision numbers*), R_1 e R_2, per il calcolo dei quali sono necessarie alcune considerazioni. Il valore di $\log 2/32$ si approssima mediante due costanti, L_1 e L_2:

$$\frac{\log 2}{32} \simeq L_1 + L_2.$$

Scegliendo L_1 in modo che abbia solo pochi degli ultimi bit uguali a zero, la somma con un opportuno L_2 determinerà la stringa di bit con cui si approssima il numero $\log 2/32$ con una precisione molto più alta di quella con cui si sta lavorando. I bit non nulli della rappresentazione di L_1 costituiscono la parte anteriore dell'approssimazione, detta la *leading part*, i bit seguenti, della rappresentazione di L_2, ne costituiscono la cosiddetta *trailing part*. La loro somma, $L_1 + L_2$, approssima $\log 2/32$ con la massima precisione. I loro valori, in singola e doppia precisione, sono riportati in [1].
Posto $M = fl(m)$ e $J = fl(j)$, si ha:

$$\begin{aligned} X &= (32 \cdot M + J) \cdot \frac{\log 2}{32} + (R_1 + R_2) \simeq \\ &\simeq (N_1 + N_2) \cdot (L_1 + L_2) + (R_1 + R_2) = \\ &= N_1 \cdot L_1 + N_2 \cdot L_1 + N \cdot L_2 + (R_1 + R_2) \\ &= (N_1 \cdot L_1 + N_2 \cdot L_1 + R_1) + (N \cdot L_2 + R_2). \end{aligned}$$

In particolare si definisce

$$R_2 := -N \cdot L_2;$$

inoltre, se è richiesta la singola precisione e se $|N| < 2^9$, $N \cdot L_1$ è rappresentabile esattamente, in singola precisione, in quanto, in questa ipotesi, la rappresentazione binaria di L_1 ha gli ultimi nove bit uguali a zero; analogamente, quando $|N| \geq 2^9$, sia $N_1 \cdot L_1$ che $N_2 \cdot L_1$ sono rappresentabili esattamente. Da queste considerazioni segue, dunque, che

$$(N_1 + N_2) \cdot L_1 = N \cdot L_1$$

è rappresentabile esattamente in singola precisione e tale è anche

$$R_1 = X - N \cdot L_1.$$

Nel caso $|N| < 2^9$ si può, allora, esprimere R_1 come

$$R_1 := (X - N \cdot L_1),$$

mentre nel caso $|N| \geq 2^9$,

$$R_1 := (X - N_1 \cdot L_1) - N_2 \cdot L_1. \tag{B.5}$$

Passo 2 Il polinomio è valutato mediante l'**algoritmo di Horner**. I coefficienti sono riportati in [1], sia per la singola che per la doppia precisione, in formato esadecimale.
In particolare, lavorando in singola precisione, le costanti necessarie al secondo passo sono i coefficienti A_1 ed A_2, del polinomio approssimante:

$$p(t) = t + A_1 t^2 + A_2 t^3.$$

In doppia precisione, il polinomio approssimante è:

$$p(t) = t + A_1 t^2 + A_2 t^3 + A_3 t^4 + A_4 t^5 + A_5 t^6.$$

Passo 3 Con la scelta $L = 5$,

$$B_j = \log 2^{j/32}, \quad j = 0, 1, 2, \ldots, 31$$

e

$$exp(B_j) = 2^{j/32}, \quad j = 0, 1, 2, \ldots, 31.$$

L'esame dell'equazione

$$e^x = 2^m e^{B_j}(1 + p(r)) \tag{B.6}$$

rivela che, per ottenere il risultato finale, devono essere determinati solo 2^m ed i valori $2^{j/32}$, $j = 0, 1, \ldots, 31$. Il metodo utilizzato è definito *table driven* proprio perché si costruisce una tabella in cui sono memorizzati i valori $2^{j/32}$, $j = 0, 1, \ldots, 31$. Ragionando come per L_1 e L_2, la cui somma approssima $\log 2/32$, i $2^{j/32}$ sono stimati mediante due numeri macchina, `S_lead(J)` e `S_trail(J)`, che rappresentano, rispettivamente, la *leading part* e la *trailing part* dell'approssimazione di $2^{j/32}$. Si ha, allora,

$$2^{\mathtt{J}/32} \approx \mathtt{S\_lead(J)} + \mathtt{S\_trail(J)}, \quad j = 0, 1, \ldots, 31.$$

I 32 valori di `S_lead(J)` e di `S_trail(J)` [2], sia per la singola che per la doppia precisione, sono scelti in modo tale che gli ultimi sei bit di `S_lead(J)` siano uguali a zero.

Da cui:

$$\begin{aligned} \mathtt{S}: &= \mathtt{S\_lead(J)} + \mathtt{S\_trail(J)} \\ \mathtt{exp}: &= 2^{\mathtt{M}} \cdot (\mathtt{S\_lead(J)} + (\mathtt{S\_trail(J)} + \mathtt{S} \cdot \mathtt{P})) \end{aligned}$$

Prima di eseguire i tre passi relativi all'algoritmo descritto, nella fase di implementazione dell'algoritmo è necessario prevedere l'insorgere di *situazioni eccezionali*.

[2] Riportati in [1].

– Se l'argomento di input, X, denota la presenza di un NaN, dovrebbe essere segnalata una situazione eccezionale, quale un'operazione non valida, e restituito NaN.

– Se X è $+\infty$, dovrebbe essere restituito $+\infty$ (come risultato ottenuto per $e^{+\infty}$), senza segnalare alcuna situazione eccezionale. Quando X è $-\infty$, dovrebbe essere restituito $+0$ (come risultato ottenuto per $e^{-\infty}$), senza segnalare alcuna situazione eccezionale.

– Per evitare che il valore della funzione esponenziale in corrispondenza del dato di input dia luogo ad una situazione eccezionale di overflow oppure di underflow, si introducono due soglie, `THRESHOLD_1` e `THRESHOLD_2`, sull'ordine di grandezza di X. In particolare, se $|X| \geq$ `THRESHOLD_1`, dovrebbe essere restituito $+\infty$ con un segnale di overflow, oppure $+0$ con un segnale di underflow ed indicazione di risultato inesatto. Quando $|X| \leq$ `THRESHOLD_2`, dovrebbe essere restituito $1 + X$; la soglia `THRESHOLD_2` è fissata in modo che l'esponenziale di qualsiasi argomento con ordine di grandezza minore sia arrotondato correttamente a 1. [3]

B.2.1 Analisi degli errori

Analizziamo gli errori introdotti ad ogni passo.
L'analisi degli errori è svolta assumendo la modalità di arrotondamento di default per lo standard IEEE. La notazione utilizzata prevede:

- lettere maiuscole, del tipo `X,Y,P,Q,...` per denotare numeri macchina, in singola precisione;

- il simbolo $fl(\cdot)$ per indicare la rappresentazione floating point di un numero reale "·". In tal modo, l'esecuzione dell'istruzione

$$\texttt{A} := \texttt{B} \cdot \texttt{C}$$

in singola precisione, fornisce il valore

$$\texttt{A} = fl(\texttt{B} \cdot \texttt{C}).$$

[3] Sebbene la rappresentazione in singola precisione dello standard IEEE cada negli intervalli

$$\pm[2^{-149}, 2^{128}(1 - 2^{-24})],$$

talvolta le situazioni di overflow incontrate nelle operazioni, tranne nelle conversioni, vengono affrontate dividendo il risultato a precisione infinita, per una costante 2^{α} ed arrotondando. La costante α (*bias*), è scelta uguale a 192 nella singola precisione e 1536 nella doppia. Con questa tecnica è possibile, quindi, rappresentare con un'accuratezza maggiore anche i numeri floating point che appartengono all'intervallo

$$\pm[2^{-341}, 2^{320}(1 - 2^{-24})].$$

Affinché il valore calcolato di e^X cada in tale intervallo, le soglie `THRESHOLD_1` e `THRESHOLD_2`, per le implementazioni in singola precisione, sono, dunque, $341 \cdot \log 2$ e 2^{-25}, rispettivamente. Per la doppia precisione sono $2610 \cdot \log 2$ e 2^{-54}, rispettivamente. I loro valori, sia nella singola che nella doppia precisione dello standard IEEE, sono riportati, nella rappresentazione esadecimale, in [1].

- Sia, inoltre, x un numero reale; si indica con δ_x la differenza seguente

$$\delta_x := fl(x) - x = X - x.$$

Si osserva che $\delta_x = 0$ per tutti i numeri reali esattamente rappresentabili e

$$|\delta_x| \leq 2^{k-1} ulp(1) = ulp(2^{k-1}),$$

per $2^k \leq |x| < 2^{k+1}$, dove $ulp(x)$ denota la distanza tra i due numeri macchina più vicini a x.

Osservazione B.1. *Lavorando in singola precisione, si supponga, ad esempio, che x sia tale che $2^k \leq |x| < 2^{k+1}$, da cui si deduce che x è un numero reale floating point del tipo $0.\underbrace{\ldots}_{\geq 24} \times 2^k$, con un numero di cifre maggiore o uguale della precisione del sistema $t = 24$; in tal caso $fl(x)$ sarà del tipo $0.\underbrace{\ldots}_{=24} \times 2^k$ e l'errore commesso nell'approssimazione di x con $fl(x)$ risulterà minore o uguale di $1/2$, sulla $24-$esima cifra. Dunque*

$$|fl(x) - x| = 0.0 \ldots 0 d_{25} \times 2^k = 0.d_{25} \times 2^{k-24} \leq 2^{k-24},$$

avendo indicato con d_{25} la $25-$esima cifra del modulo della differenza, $d_{25} \leq \frac{1}{2}\beta$, con β base del sistema aritmetico utilizzato; in particolare risulta $ulp(1) = 2^{-23}$ per la singola precisione e $ulp(1) = 2^{-52}$ per la doppia e, dunque,

$$\begin{array}{rcll} |\delta_x| & \leq & 2^{k-24} & \textit{in singola precisione} \\ |\delta_x| & \leq & 2^{k-53} & \textit{in doppia precisione} \end{array}$$

- Utilizzando le notazioni introdotte, sussiste, dunque, la relazione

$$fl(\mathtt{A}\, op\, \mathtt{B}) = \mathtt{A}\, op\, \mathtt{B} + \delta_{\mathtt{A}\, op\, \mathtt{B}}$$

in assenza di overflow ed underflow, per ciascun valore dei parametri `A` e `B` e per ciascuna delle quattro operazioni $+, -, \cdot, /$.

Siano $\mathtt{R}_1, \mathtt{R}_2, \mathtt{N}, \mathtt{N}_1, \mathtt{N}_2, \mathtt{M}$ e `J` i numeri macchina dei corrispondenti valori, ottenuti secondo le note relative all'implementazione del **passo 1** (cfr. §B.2). Di seguito si stima la differenza tra $\mathtt{R} = \mathtt{R}_1 + \mathtt{R}_2$ e r, dove

$$r = \mathtt{X} - \mathtt{N} \cdot \log 2/32.$$

Osservazione B.2. **(sulla rappresentazione binaria delle grandezze in esame)** $\mathtt{N}_2$ *ha al più cinque bit significativi, essendo per definizione, il resto della divisione di* $\mathtt{N}$ *per 32 e, quindi,* $0 \leq \mathtt{N}_2 \leq 2^5 - 1 = 31$*;* $\mathtt{N}_1$ *ha, invece, al più nove bit significativi, mentre i suoi ultimi cinque bit sono uguali a zero. In effetti, esso è uguale alla differenza tra* $\mathtt{N}$ *ed il suo resto* `mod 32`*,* $\mathtt{N}_2$*, quest'ultimo ottenuto mediante un'operazione di AND bit a bit (che restituisce 1 solo se entrambi i bit sono uguali a 1), eseguita tra gli ultimi 5 bit di* N *e la sequenza "11111"; il risultato avrà un bit uguale a 1 solo dove il corrispondente bit di* N *è 1, da cui, nella sottrazione* $\mathtt{N}_1 = \mathtt{N} - \mathtt{N}_2$ *gli ultimi 5 bit di* $\mathtt{N}_1$ *risultano, inevitabilmente, uguali a 0.* $\mathtt{N}_1$ *è, inoltre, multiplo di* $2^5 = 32$*, avendo come primo bit significativo, eventualmente non nullo, il sesto (da destra verso sinistra, nella stringa di bit con cui si rappresenta),*

$\pm$	$\cdots$	*1*	*0*	*0*	*0*	*0*	*0*

,

che corrisponde al fattore di 2^5*, nella conversione dal sistema binario al decimale. Si deduce, infine, che* $\mathtt{N}_1$ *ha un ordine di grandezza strettamente minore di* 2^{14}*, essendo, complessivamente, rappresentabile mediante 14 bit (9 significativi e 5 nulli).*

Lemma B.2.1. *L'argomento* $\mathtt{R} = \mathtt{R}_1 + \mathtt{R}_2$ *approssima il dato di input,* $\mathtt{X} = fl(x)$*, con un errore*

$$\mathtt{R}_1 + \mathtt{R}_2 - \mathtt{X} = -(\mathtt{N} \cdot \mathtt{L}_1 + fl(\mathtt{N} \cdot \mathtt{L}_2))$$

per cui

$$\mathtt{R}_1 + \mathtt{R}_2 = \mathtt{X} - \mathtt{N}(\mathtt{L}_1 + \mathtt{L}_2) - \delta_{\mathtt{N} \cdot \mathtt{L}_2},$$

con, in particolare,

$$fl(\mathtt{N} \cdot \mathtt{L}_2) = \mathtt{N} \cdot \mathtt{L}_2 + \delta_{\mathtt{N} \cdot \mathtt{L}_2}. \tag{B.7}$$

Dimostrazione $\mathtt{L}_1, \mathtt{L}_2$ sono scelti in modo tale che $\mathtt{L}_1 + \mathtt{L}_2$ sia un'approssimazione di $\log 2/32$ corretta a 49 cifre decimali, ovvero

$$|\mathtt{L}_1 + \mathtt{L}_2 - \log 2/32| < 2^{-49};$$

si fissa, poi,

$$|\mathtt{L}_2| < 2^{-24}. \tag{B.8}$$

Poiché la rappresentazione binaria di $\mathtt{L}_1$ ha gli ultimi nove bit uguali a zero, il valore $\mathtt{N} \cdot \mathtt{L}_1$ è rappresentabile esattamente, in singola precisione, ogni volta che $|\mathtt{N}| < 2^9$. Analogamente, quando $|\mathtt{N}| \geq 2^9$, sia $\mathtt{N}_1 \cdot \mathtt{L}_1$ che $\mathtt{N}_2 \cdot \mathtt{L}_1$ sono rappresentabili esattamente. Da queste considerazioni segue, dunque, che

$$(\mathtt{N}_1 + \mathtt{N}_2) \cdot \mathtt{L}_1 = \mathtt{N} \cdot \mathtt{L}_1$$

è rappresentabile esattamente in singola precisione e tale è anche

$$\mathtt{R}_1 = \mathtt{X} - \mathtt{N} \cdot \mathtt{L}_1.$$

Al contrario $\mathtt{N} \cdot \mathtt{L}_2$ non risulta rappresentabile esattamente in singola precisione, per cui, nell'analisi degli errori, bisogna porre $\mathtt{R}_2$ uguale ad un valore arrotondato alla singola precisione,

$$\mathtt{R}_2 = fl(-\mathtt{N} \cdot \mathtt{L}_2).$$

Allora

$$\mathtt{R_1} + \mathtt{R_2} = \mathtt{X} - \mathtt{N} \cdot \mathtt{L_1} + fl(-\mathtt{N} \cdot \mathtt{L_2})$$

e, dalla (B.7), segue, dunque, l'asserto. ■

Proposizione B.2.1. *L'errore di roundoff, commesso nella rappresentazione floating point, R, dell'argomento r, ammette il seguente limite superiore*

$$\boxed{|R - r| = |(\mathtt{R_1} + \mathtt{R_2}) - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| < 2^{-34}.}$$

Dimostrazione Dall'essere

$$\begin{aligned} |(\mathtt{R_1} + \mathtt{R_2}) - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| &= |(\mathtt{X} - \mathtt{N}(\mathtt{L_1} + \mathtt{L_2}) - \delta_{\mathtt{N}\cdot\mathtt{L_2}}) - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| \\ &\leq |N| \cdot |\mathtt{L_1} + \mathtt{L_2} - \log 2/32| + |\delta_{\mathtt{N}\cdot\mathtt{L_2}}| \end{aligned} \tag{B.9}$$

si deduce la stima desiderata maggiorando i termini al secondo membro. Innanzitutto risulta $|\mathtt{N}| \leq 341 \cdot 32 \simeq 2^{14} \cdot 0.66$; infatti l'input

$$\begin{aligned} \mathtt{X} \in\ & [-\mathtt{THRESHOLD\_1}, -\mathtt{THRESHOLD\_2}] \cup \{0\} \cup \\ & \cup [\mathtt{THRESHOLD\_2}, \mathtt{THRESHOLD\_1}] \end{aligned}$$

e si deduce, in particolare, che

$$|\mathtt{X}| \leq 341 \cdot \log 2.$$

Allora,

$$\begin{aligned} |N| &= \mathtt{INTRND}\left(\frac{|X| \cdot 32}{\log 2}\right) \leq \mathtt{INTRND}\left(341 \cdot \log 2 \cdot \frac{32}{\log 2}\right) \leq \\ &\leq \mathtt{INTRND}\left(2^9 \cdot 0.66601\ldots \times 32\right) \leq \mathtt{INTRND}(10911.744) = \\ &= 10911 < 10912 = 341 \times 32. \end{aligned}$$

Tenendo conto, poi, della (B.8), $|\mathtt{N} \cdot \mathtt{L_2}| < 2^{14} \cdot 2^{-24} \cdot 2^{-0.6} = 2^{-10.6}$, essendo, infatti, $2^{-0.6} \simeq 0.66$. Di conseguenza, poiché da

$$|\mathtt{N} \cdot \mathtt{L_2}| < 2^{-11.6+1} = 2^{k+1}$$

segue

$$|\delta_{\mathtt{N}\cdot\mathtt{L_2}}| \leq 2^{k-24} = 2^{-11.6-24} = 2^{-11.6} \cdot 2^{-24},$$

si deduce che

$$\begin{aligned} |(\mathtt{R_1} + \mathtt{R_2}) - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| &< 341 \cdot 32 \cdot 2^{-49} + 2^{-11.6} \cdot 2^{-24} \\ &< 2^9 \cdot 2^5 \cdot 2^{-49} + 2^{-35.6}. \end{aligned}$$

In particolare, allora, una stima dell'errore commesso nella rappresentazione dell'argomento scalato nell'intervallo desiderato, è:

$$|(\mathtt{R_1} + \mathtt{R_2}) - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| < 2^{-34},$$

da cui l'asserto. ■

Proposizione B.2.2. *Per l'errore di approssimazione vale la stima*

$$|e^t - 1 - p(t)| < 2^{-33.2} \quad \textit{per ogni} \quad t \in [-0.010831, 0.010831].$$

Dimostrazione Il risultato si prova applicando il secondo algoritmo di Remez per approssimare la funzione $e^t - 1$ nell'intervallo $[-0.010831, 0.010831]$. ■

Proposizione B.2.3. *Gli errori di roundoff introdotti soddisfano alle seguenti maggiorazioni*

$$\begin{array}{llll} |\textit{errore} & \textit{di} & \textit{roundoff}| \leq 0.5240 \cdot 2^{-24} & \textit{se } j = 0 \textit{ e } r < 0, \\ |\textit{errore} & \textit{di} & \textit{roundoff}| \leq 0.5120 \cdot 2^{-23} & \textit{se } j = 0 \textit{ e } r \geq 0, \\ |\textit{errore} & \textit{di} & \textit{roundoff}| \leq 0.5295 \cdot 2^{-23} & \textit{se } j \geq 1. \end{array}$$

Dimostrazione Stimiamo la differenza tra il valore

$$fl(2^{\mathtt{M}} \cdot fl(\mathtt{S\_lead(J)} + fl(\mathtt{S\_trail(J)} + fl(\mathtt{S\_lead(J)} + \mathtt{S\_trail(J)}) \cdot \mathtt{P})))$$

ed il valore corrispondente

$$2^{\mathtt{M}} \cdot (\mathtt{S\_lead(J)} + (\mathtt{S\_trail(J)} + (\mathtt{S\_lead(J)} + \mathtt{S\_trail(J)}) \cdot \mathtt{P})).$$

Finché il risultato finale non realizza un underflow, la moltiplicazione per $2^{\mathtt{M}}$ è esatta. Così, è sufficiente considerare il caso in cui $\mathtt{M} = 0$. In questa circostanza, il risultato finale appartiene all'intervallo $(\frac{1}{2}, 2)$. Si nota anche che il risultato finale appartiene all'intervallo $(\frac{1}{2}, 1)$ se, e solo se, $J = 0$ e $r < 0$. Di conseguenza,

$$\begin{array}{ll} 1\,ulp = 2^{-24} & \text{per} \quad J = 0 \quad \text{e} \quad r < 0, \\ 1\,ulp = 2^{-23} & \text{altrimenti.} \end{array}$$

Inoltre l'ordine di grandezza di

$$\mathtt{Q} = \mathtt{R} \cdot \mathtt{R} \cdot (\mathtt{A}_1 + \mathtt{R} \cdot (\mathtt{A}_2 + \mathtt{R} \cdot (\ldots + \mathtt{R} \cdot \mathtt{A}_n) \ldots)),$$

è

$$\frac{1}{2}\left(\frac{\log 2}{64}\right)^2 < 2^{-13}.$$

In tal modo gli errori di roundoff, che si accumulano nel calcolarlo, sono praticamente nulli. Per semplificare le espressioni che seguono, nell'analisi degli errori, si adottano i simboli

$$\begin{array}{rcl} \mathtt{S}_1 : & = & \mathtt{S\_lead(J)} \\ \mathtt{S}_2 : & = & \mathtt{S\_trail(J)} \\ \mathtt{S} : & = & fl(\mathtt{S\_lead(J)} + \mathtt{S\_trail(J)}) \end{array}$$

In conclusione, calcoliamo la differenza tra

$$|fl(fl(\mathtt{S}_1) + fl(\mathtt{S}_2) + \mathtt{S}) - (\mathtt{S}_1 + \mathtt{S}_2 + \mathtt{S})|.$$

Posto $E_0 = \mathtt{S}_1 + \mathtt{S}_2 + \mathtt{S}$, indichiamo, poi, con E_1 il primo risultato calcolato, E_2 il secondo e così via. E_6 è il risultato finale calcolato e l'errore di roundoff è, quindi, la differenza

$$E_0 - E_6.$$

Si ha, allora,

$$\begin{aligned}
E_0 : &= \mathtt{S}_1 + \mathtt{S}_2 + (\mathtt{S}_1 + \mathtt{S}_2) \cdot (\mathtt{R}_1 + (\mathtt{R}_2 + \mathtt{Q})) \\
E_1 : &= \mathtt{S}_1 + \mathtt{S}_2 + (\mathtt{S}_1 + \mathtt{S}_2) \cdot (\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})) \\
E_2 : &= \mathtt{S}_1 + \mathtt{S}_2 + (\mathtt{S}_1 + \mathtt{S}_2) \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})) \\
E_3 : &= \mathtt{S}_1 + \mathtt{S}_2 + \mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})) \\
E_4 : &= \mathtt{S}_1 + \mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q}))) \\
E_5 : &= \mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})))) \\
E_6 : &= fl(\mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})))))
\end{aligned}$$

Si denotano, poi, con $F_1, F_2, \ldots, F_6$ i valori

$$\begin{aligned}
F_1 : &= \mathtt{R}_2 + \mathtt{Q} \\
F_2 : &= \mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q}) \\
F_3 : &= \mathtt{S}_1 + \mathtt{S}_2 \\
F_4 : &= \mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})) \\
F_5 : &= \mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q}))) \\
F_6 : &= \mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q}))))
\end{aligned}$$

A questo punto, si stima

$$|\text{errore di roundoff}| = |E_0 - E_6| \leq \sum_{i=1}^{6} |E_{i-1} - E_i|,$$

laddove

$$\begin{aligned}
|E_0 - E_1| &= |\mathtt{S}_1 + \mathtt{S}_2| \cdot |(\mathtt{R}_2 + \mathtt{Q}) - fl(\mathtt{R}_2 + \mathtt{Q})| \\
&= |F_3| \cdot |\delta_{F_1}|, \\
|E_1 - E_2| &= |\mathtt{S}_1 + \mathtt{S}_2| \cdot |(R_1 + fl(\mathtt{R}_2 + \mathtt{Q})) - fl(R_1 + fl(\mathtt{R}_2 + \mathtt{Q}))| \\
&= |F_3| \cdot |\delta_{F_2}|, \\
|E_2 - E_3| &= |fl(F_2)| \cdot |\delta_{F_3}| \\
|E_3 - E_4| &= |\delta_{F_4}|, \\
|E_4 - E_5| &= |\delta_{F_5}|, \\
|E_5 - E_6| &= |\delta_{F_6}|.
\end{aligned} \tag{B.10}$$

Per calcolare una stima dei $|\delta_{F_i}|$ per $i = 1, \ldots, 6$, è sufficiente conoscere la parte positiva degli intervalli binari, simmetrici rispetto allo 0, cui possono appartenere i valori $|F_i|$. Si osserva che ciascuno degli F_i è il risultato ottenuto da operazioni su valori il cui range è noto. Di conseguenza, a meno che l'ordine di grandezza più elevato, raggiunto da quei valori, sia molto vicino ad una potenza di due, gli intervalli binari più a destra, in cui essi si possono trovare, sono proprio gli intervalli cercati. Si tabulano, di seguito, i risultati (**Tabella B.1**).

Valore	Intervallo	Stime
$\lvert\mathtt{R}_2\rvert$	$[0, 2^{-10.78}]$	$\lvert\delta_{F_1}\rvert \leq 2^{-35}$.
$\lvert p(r)\rvert$	$[0, 2^{-6.52}]$	$\lvert\delta_{F_2}\rvert \leq 2^{-31}$, $\lvert fl(F_2)\rvert \leq 2^{-6.5}$.
$\lvert 2^{j/32}\rvert$	$[0, 2^{31/32}]$	$\lvert\delta_{F_3}\rvert = 0$ for $j = 0$;
		$\lvert\delta_{F_3}\rvert \leq 2^{-24}$ altrimenti.
$\lvert 2^{j/32}p(r)\rvert$	$2^{j/32}[0, 2^{-6.52}]$	$\lvert\delta_{F_4}\rvert \leq 2^{-31}$ for $j = 0$;
		$\lvert\delta_{F_4}\rvert \leq 2^{-30}$ altrimenti.
$\lvert\mathtt{S}_2 + 2^{j/32}p(r)\rvert$	$2^{j/32}[0, 2^{-6.49}]$	$\lvert\delta_{F_5}\rvert \leq 2^{-31}$ for $j = 0$;
		$\lvert\delta_{F_5}\rvert \leq 2^{-30}$ altrimenti.
$\lvert 2^{j/32}e^r\rvert$	$2^{j/32}[2^{-1/64}, 2^{1/64}]$	$\lvert\delta_{F_6}\rvert \leq 2^{-25}$ for $j = 0$ e $r < 0$;
		$\lvert\delta_{F_6}\rvert \leq 2^{-24}$ altrimenti.

Tabella B.1: Risultati.

Tenendo conto che $|F_3| = |\mathtt{S}|$ approssima $2^{j/32}$ e che

$$\begin{array}{ll} |F_3| = 1, & \text{se} \quad j = 0, \\ |F_3| = 2^{j/32} \leq 2^{31/32} = 1.9571\ldots \leq 2 & \text{se} \quad j \geq 1, \quad j \in \{1, \ldots, 31\} \end{array}$$

si calcola

$$\begin{array}{rcl} |\text{errore di roundoff}| & \leq & |F_3| \cdot |\delta_{F_1}| + |F_3| \cdot |\delta_{F_2}| + |fl(F_2)| \cdot |\delta_{F_3}| + \\ & + & |\delta_{F_4}| + |\delta_{F_5}| + |\delta_{F_6}| \end{array}$$

Sostituendo i valori numerici si deducono le stime seguenti, in ciascuno dei casi indicati.
Quando $j = 0$ e $r < 0$,

$$|\text{errore di roundoff}| \leq 0.5240 \cdot 2^{-24}$$

Quando $j = 0$ e $r \geq 0$,

$$|\text{errore di roundoff}| \leq 0.5120 \cdot 2^{-23}$$

Quando $j \geq 1$,

$$|\text{errore di roundoff}| \leq 0.5295 \cdot 2^{-23}$$

■

Supponendo che, nell'equazione (B.6), $2^{\mathtt{M}}$ ed i $2^{j/32}$ siano calcolati esattamente, l'errore globale si stima in base all'approssimazione del termine

$$2^{j/32}(1 + p(r)) = 2^{j/32}e^r.$$

Proposizione B.2.4. *L'errore globale risulta*

$$\boxed{|2^{j/32}e^r - fl(\mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})))))| < 0.54\, ulp}$$

Dimostrazione Aggiungendo e sottraendo la quantità

$$2^{j/32}(e^{\mathtt{R}_1+\mathtt{R}_2} - 1 - p(\mathtt{R}_1 + \mathtt{R}_2)),$$

raggruppando, opportunamente, per le proprietà del valore assoluto segue:

$$\begin{aligned} |\text{errore}| &= |2^{j/32}e^r - fl(\mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})))))| \\ &\leq 2^{j/32} \cdot |e^r - e^{\mathtt{R}_1+\mathtt{R}_2}| + \\ &+ 2^{j/32} \cdot |e^{\mathtt{R}_1+\mathtt{R}_2} - 1 - p(\mathtt{R}_1 + \mathtt{R}_2)| + \\ &+ |2^{j/32}(p(\mathtt{R}_1 + \mathtt{R}_2) + 1) - \\ &- fl(\mathtt{S}_1 + fl(\mathtt{S}_2 + fl(\mathtt{S} \cdot fl(\mathtt{R}_1 + fl(\mathtt{R}_2 + \mathtt{Q})))))| \end{aligned} \tag{B.11}$$

Nella (B.11) si possono eseguire alcune maggiorazioni. Innanzitutto è

$$|e^r - e^{\mathtt{R}_1+\mathtt{R}_2}| \leq 1.01|r - (\mathtt{R}_1 + \mathtt{R}_2)|,$$

potendo stimare:

$$\begin{aligned} &\left|\int_0^r e^t dt - \int_0^{\mathtt{R}_1+\mathtt{R}_2} e^t dt\right| \leq \left|\int_{\mathtt{R}_1+\mathtt{R}_2}^r e^t dt\right| \leq \\ &\leq \max_{t \in [-\frac{\log 2}{64}, \frac{\log 2}{64}]} |e^t| \cdot |r - (\mathtt{R}_1 + \mathtt{R}_2)| \end{aligned}$$

e, per quanto riportato in **Tabella B.1**,

$$|e^r| \in [2^{-1/64}, 2^{1/64}]$$

con $2^{1/64} = 1.0109$. Inoltre,

$$|e^{\mathtt{R}_1+\mathtt{R}_2} - 1 - p(\mathtt{R}_1 + \mathtt{R}_2)| \leq 2^{-33.2}$$

per la **Proposizione B.2.2**. Si analizzano, dunque, i singoli casi.
Quando $j = 0$ e $r < 0$, $1\, ulp = 2^{-24}$ e

$$\begin{aligned} |\text{errore}| &\leq 1.01 \underbrace{|r - (\mathtt{R}_1 + \mathtt{R}_2)|}_{\leq 2^{-34}} + 2^{-33.2} + 0.5240 \cdot 2^{-24} \\ &\leq 2^{-24}(1.01 \cdot 2^{-10} + 2^{-9.2} + 0.524) \\ &\leq 0.5267 \times 2^{-24} = 0.5267\, ulp. \end{aligned}$$

Quando $j = 0$ e $r \geq 0$, $1\, ulp = 2^{-23}$ e

$$\begin{aligned} |\text{errore}| &\leq 1.01 \underbrace{|r - (\mathtt{R}_1 + \mathtt{R}_2)|}_{\leq 2^{-34}} + 2^{-33.2} + 0.5120 \cdot 2^{-23} \\ &\leq 2^{-23}(1.01 \cdot 2^{-11} + 2^{-10.2} + 0.512) \\ &\leq 0.5134 \times 2^{-23} = 0.5134\, ulp. \end{aligned} \tag{B.12}$$

Quando $j \geq 1$, $1\, ulp = 2^{-23}$ e

$$\begin{aligned} |\text{errore}| &\leq 1.01 \cdot 2^{31/32} \cdot \underbrace{|r - (\mathtt{R}_1 + \mathtt{R}_2)|}_{\leq 2^{-34}} + 2^{31/32}2^{-33.2} + 0.5351 \cdot 2^{-23} \\ &\leq 2^{-23}(1.01 \cdot 2^{-11}2^{31/32} + 2^{-10.2}2^{31/32} + 0.5295) \\ &\leq 0.5321 \times 2^{-23} = 0.5321\, ulp. \end{aligned} \tag{B.13}$$

In tal modo, laddove il risultato finale non realizza un underflow, l'errore complessivo è inferiore a $0.54\,ulp$, cioè la tesi. ■

Consideriamo, ora, il caso in cui il risultato finale appartenga al primo intervallo binario $[2^{-127}, 2^{-126})$ di *gradual underflow*. Esso avrà, allora, solo 23 bit significativi. Così l'errore di $0.54\,ulp$, riscontrato utilizzando 24 bit significativi, consiste solo di $\frac{1}{2}(0.54)\,ulp = 0.27\,ulp$. D'altra parte, la moltiplicazione per $2^{\mathtt{M}}$, introduce un errore che può essere grande quanto $\frac{1}{2}\,ulp$. Di qui, la limitazione per l'errore diventa

$$(0.5 + 0.54/2)ulp = 0.77\,ulp.$$

Ragionamenti simili conducono ad un limite, per l'errore complessivo, di

$$(0.5 + 2^{-j}(0.54))\,ulp,$$

quando il risultato appartiene all'intervallo $[2^{-126-j}, 2^{-125-j})$, $j = 1, 2, \ldots, 22$. Dunque, anche quando il risultato finale risente di un gradual underflow, l'errore totale può ancora non essere superiore a $0.77\,ulp$.
Applicando la stessa analisi per l'implementazione in doppia precisione, si ottiene:

$$\begin{array}{rcll} |\mathtt{R}_1 + \mathtt{R}_2 - (\mathtt{X} - \mathtt{N} \cdot \log 2/32)| \leq & 2^{-77} & & \\ |e^t - 1 - p(t)| < & 2^{-63.2} & & \\ |\text{errore di roundoff}| < & 0.5235 \cdot 2^{-53} & \text{se} \quad \mathtt{J} = 0 \quad \text{e} \quad r < 0; & \\ < & 0.5267 \cdot 2^{-52} & \text{altrimenti.} & \end{array}$$

Quindi, anche lavorando in doppia precisione, l'errore globale si mantiene inferiore a $0.54\,ulp$, quando il risultato non realizza underflow, inferiore a $0.77\,ulp$, quando si verifica un underflow.

Bibliografia

[1] Ping Tank Peter Tang - *Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic* - ACM, Vol. 15, No. 2, pp.144-157 (1989).

[2] Watson G. A. - *Approximation Theory and Numerical Methods* - John Wiley and sons, 1980.

Appendice C

I polinomi ortogonali

C.1 Generalità

Assegnato uno spazio di funzioni $\mathcal{F}$ definite su un sottoinsieme X (continuo o discreto) del campo reale[1] $\Re$ è possibile fornire la seguente:

Definizione C.1.1. (Prodotto scalare tra due funzioni)
Il **prodotto scalare** *tra due funzioni, $f \in \mathcal{F}$ e $g \in \mathcal{F}$, è definito come:*

$$(f,g) = \begin{cases} \int_a^b f(x)g(x)\,dx & X = [a,b] \quad \text{(caso continuo)} \\ \sum_{i=0}^n f(x_i)g(x_i) & X = \{x_0, ..., x_n\} \quad \text{(caso discreto)} \end{cases}$$

Tale definizione induce la norma sullo spazio $\mathcal{F}$:

$$||f||_2 = \sqrt{(f,f)}$$

Più in generale, è possibile definire il prodotto scalare tra f e g, rispetto ad una funzione peso ammissibile ψ, mediante l'espressione:

$$(f,g) = \begin{cases} \int_a^b \psi(x)f(x)g(x)\,dx & X = [a,b] \quad \text{(caso reale)} \\ \sum_{i=0}^n \psi(x_i)f(x_i)g(x_i) & X = \{x_0, ..., x_n\} \quad \text{(caso discreto)} \end{cases}$$

Sia dato un insieme di funzioni $\{\phi_0, ..., \phi_n\}$ definite in $X \subset \Re$. Vale la seguente

Definizione C.1.2. (Insieme ortogonale di funzioni)
L'insieme di funzioni $\{\phi_0, ..., \phi_n\}$ definite su $X \subset \Re$è un insieme di **funzioni ortogonale** *in X se e solo se*

$$\begin{aligned} (\phi_i, \phi_j) &= 0 && \forall i \neq j \\ (\phi_j, \phi_j) &= ||\phi_j||_2^2 \neq 0 && \forall j \end{aligned}$$

Se inoltre accade che

$$||\phi_j||_2 = 1 \quad \forall j$$

l'insieme di funzioni si dice **ortonormale**.

[1] Tutta la teoria dei polinomi ortogonali può essere sviluppata con ovvie generalizzazioni nel campo complesso.

♣ **Esempio C.1.** I polinomi $p_0(x) = 1$, $p_1(x) = x$ e $p_2(x) = 3x^2 - 1$ costituiscono un insieme di polinomi ortogonali in $[-1, 1]$. Infatti:

$$\int_{-1}^{1} 1x\,dx = 0 \qquad \int_{-1}^{1} x(3x^2 - 1)\,dx = 0 \qquad \int_{-1}^{1} 1(3x^2 - 1)\,dx = 0$$

♣

Per le funzioni ortogonali valgono le seguenti proprietà:

Proprietà C.1.1. *Due funzioni f e g sono ortogonali tra loro se e solo se*

$$||f + g||_2^2 = ||f||_2^2 + ||g||_2^2$$

Proprietà C.1.2. *Se l'insieme di funzioni $\{\phi_0, ..., \phi_n\}$ è un insieme ortogonale, le funzioni $\phi_0, ..., \phi_n$ sono linearmente indipendenti.*

Nello sviluppo di metodi per l'integrazione numerica, particolare attenzione meritano le famiglie di **polinomi ortogonali**:

$$p_0, p_1, \ldots, p_k, \ldots$$

dove l'indice k indica anche il grado del polinomio p_k. Per le famiglie di polinomi ortogonali sussiste il seguente

Teorema C.1.1. *Sia $\mathcal{P} = \{p_0, p_1, \ldots, p_k, \ldots\}$ una famiglia di polinomi ortogonali in un insieme $X \subset \Re$ rispetto ad una funzione peso ψ e sia q un generico polinomio di grado m. Si ha:*

1. *il sottoinsieme $\mathcal{P}_m = \{p_0, p_1, \ldots, p_m\} \subset \mathcal{P}$ è una base per Π_m, cioè q ammette un'unica rappresentazione del tipo:*

$$q(x) = b_0 p_0(x) + b_1 p_1(x) + \cdots + b_m p_m(x)$$

2. *Il polinomio q è ortogonale ad ogni $p_k \in \mathcal{P}$ con $k > m$, cioè:*

$$(q, p_k) = 0 \quad k > m$$

3. *Ogni polinomio $p_k \in \mathcal{P}$ ha k zeri semplici in $]a, b[$.*

4. Per ogni famiglia di polinomi $\mathcal{P}$ esistono dei coefficienti A_k, B_k e C_k tali che

$$p_k(x) = (A_k x + B_k)p_{k-1}(x) - C_k p_{k-2}(x) \tag{C.1}$$

dove

$$A_k = \frac{d_k}{d_{k-1}} \qquad C_k = \frac{A_k}{A_{k-1}} = \frac{d_k d_{k-2}}{d_{k-1}^2} \qquad B_k = \frac{A_k(xp_{k-1}, p_{k-1})}{(p_{k-1}, p_{k-1})} \tag{C.2}$$

con d_k coefficiente di grado più alto del polinomio p_k.

Dimostrazione [2]

1. La dimostrazione segue immediatamente dal fatto che la famiglia $\mathcal{P}_m$ è composta da $m+1$ funzioni linearmente indipendenti nello spazio Π_m che ha dimensione $m+1$ e pertanto costituisce una base per lo spazio dei polinomi di grado al più m.

2. Per la proprietà precedente e per la linearità del prodotto scalare, è possibile scomporre il prodotto tra q e p_k come:

   $$(q, p_k) = b_0(p_0, p_k) + b_1(p_1, p_k) + \cdots + b_m(p_m, p_k),$$

   per l'ortogonalità si ha:

   $$(p_0, p_k) = (p_1, p_k) = \cdots = (p_m, p_k) = 0 \quad \forall k > m$$

   da cui la tesi.

3. Se p_k fosse di segno costante in $[a, b]$ si avrebbe, poichè p_0 è una funzione costante, che

   $$(p_k, p_0) = \int_a^b \psi(x)p_k(x)p_0(x)\,dx \neq 0$$

   contro l'ipotesi di ortogonalità, per cui p_k ha almeno uno zero in $]a, b[$. Sia $\overline{x}$ tale zero. Se tale zero fosse uno zero doppio si avrebbe che il polinomio $g(x) = p_k(x)/(x - \overline{x})^2$ avrebbe grado $k-2$, e per la proprietà precedente risulterebbe:

   $$0 = \left(p_k, \frac{p_k}{(x-\overline{x})^2}\right) = \left(1, \left(\frac{p_k}{x-\overline{x}}\right)^2\right) > 0$$

   e ciò è un assurdo, per cui p_k ha solo zeri semplici. Siano $x_1, x_2, \ldots, x_r$ gli zeri di p_k. Ovviamente $r \leq k$ ed in tal caso p_k può essere espresso come

   $$p_k(x) = q_{k-r}(x)(x - x_1)\cdots(x - x_r) \quad \text{con } q_{k-r} \text{ a segno costante in } [a, b]$$

   Posto $s_r(x) = (x - x_1)\cdots(x - x_r)$ si ha

   $$p_k(x)s_r(x) = q_{k-r}(x)(x - x_1)^2 \cdots (x - x_r)^2$$

   da cui si nota che il polinomio $p_k(x)s_r(x)$ non cambia segno in $[a, b]$. Ciò vuol dire che $(p_k, s_r) \neq 0$ (cioè i polinomi p_k e s_r non sono ortogonali tra loro), e quindi $k = r$.

4. La dimostrazione di tale proprietà procede per induzione ed è lasciata per esercizio al lettore.

[2] Il teorema è dimostrato nel caso continuo, cioè nel caso in cui $X = [a, b]$. Analogamente sussiste la dimostrazione nel caso discreto $X = \{x_0, \ldots, x_m\}$.

■

Una importante proprietà dei polinomi ortogonali è l'uguaglianza nota come *uguaglianza di Christoffel - Darboux*:

Proprietà C.1.3. *Per una famiglia di polinomi ortogonali $p_0, p_1, \ldots, p_n, \ldots$ vale la:*

$$\sum_{k=0}^{n} p_k(x)p_k(t) = \frac{d_n}{d_{n+1}} \frac{[p_{n+1}(x)p_n(t) - p_n(x)p_{n+1}(t)]}{(x-t)}$$

dove d_n è il coefficiente di x^n nell'n-mo polinomio ortogonale.

Dimostrazione Dalla (C.1) del Teorema C.1.1 si ha:

$$p_{k+1}(x)p_k(t) - p_k(x)p_{k+1}(t) =$$

$$[(A_{k+1}x - B_{k+1})p_k(x) - C_{k+1}p_{k-1}(x)]p_k(t) - p_k(x)[A_{k+1}t - B_{k+1}p_k(t) - C_{k+1}p_{k-1}(t)] =$$

$$= A_{k+1}(x-t)p_k(x)p_k(t) + C_{k+1}[p_k(x)p_{k-1}(t) - p_{k-1}(x)p_k(t)] \tag{C.3}$$

Per la (C.2), dividendo la (C.3) per $(x-t)$ si ha

$$\frac{d_k}{d_{k+1}} \frac{p_{k+1}(x)p_k(t) - p_k(x)p_{k+1}(t)}{(x-t)} = p_k(x)p_k(t) + \frac{d_{k-1}}{d_k} \frac{p_k(x)p_{k-1}(t) - p_{k-1}(x)p_k(t)}{(x-t)} \tag{C.4}$$

che vale anche per $k = 0$ con d_{-1} arbitrario e $p_{-1}(x) = 0$. La (C.4) equivale a:

$$p_k(x)p_k(t) = \frac{d_k}{d_{k+1}} \frac{p_{k+1}(x)p_k(t) - p_k(x)p_{k+1}(t)}{(x-t)} - \frac{d_{k-1}}{d_k} \frac{p_k(x)p_{k-1}(t) - p_{k-1}(x)p_k(t)}{(x-t)} \tag{C.5}$$

Riscrivendo la (C.5) per $k = 0, 1, \ldots, n$ e sommando tra loro le equazioni ottenute si ha la tesi. ■

Di seguito vengono fornite le caratteristiche delle principali famiglie di polinomi ortogonali [3]:

[3] Si ricorda che la funzione *delta di Kronecker* si definisce come:

$$\delta_{n,m} = \begin{cases} 1 & \text{se } n = m \\ 0 & \text{se } n \neq m \end{cases}$$

Polinomi di Legendre

Funzione peso: $\psi(x) = 1$, intervallo di ortogonalità: $[-1, 1]$

$$||p_k||_2^2 = \int_{-1}^{1} p_k(x)p_k(x)\,dx = 2/(2k+1)$$
$$\int_{-1}^{1} p_n(x)p_m(x)dx = \frac{2}{2n+1}\delta_{m,n}, \quad n, m = 0, 1, 2, ...$$

con δ_{mn} simbolo di Kronecker. Relazione ricorrente: $p_0(x) = 1$, $p_1(x) = x$

$A_k = (2k-1)/k, \qquad B_k = 0, \qquad C_k = (k-1)/k$

Polinomi di Laguerre

Funzione peso: $\psi(x) = x^\alpha e^{-x} \quad \alpha > -1$, intervallo di ortogonalità: $[0, \infty[$

$$||p_k||_2^2 = \int_0^{\infty} \psi(x)p_k(x)p_k(x)\,dx = \frac{(k+\alpha)!}{k!}$$
$$\int_0^{\infty} \psi(x)p_n(x)p_m(x)\,dx = \frac{(n+\alpha)!}{n!}\delta_{m,n}, \quad n, m = 0, 1, 2, ...$$

con δ_{mn} simbolo di Kronecker. Relazione ricorrente: $p_0(x) = 1$, $p_1(x) = 1 - x$

$A_k = -1/(k+1), \qquad B_k = 2k + \alpha + 1, \qquad C_k = (k+\alpha)/(k+1)$

Polinomi di Hermite

Funzione peso: $\psi(x) = e^{-x^2}$, intervallo di ortogonalità: $]-\infty, \infty[$

$$||p_k||_2^2 = \int_{-\infty}^{\infty} \psi(x)p_k(x)p_k(x)\,dx = \sqrt{\pi}2^k k!$$
$$\int_{-\infty}^{\infty} \psi(x)p_n(x)p_m(x)\,dx = \sqrt{\pi}2^n n!\delta_{m,n}, \quad n, m = 0, 1, ...$$

con δ_{mn} simbolo di Kronecker. Relazione ricorrente: $p_0(x) = 1$, $p_1(x) = 2x$

$A_k = 2, \qquad B_k = 0, \qquad C_k = 2k$

Polinomi di Chebyshev

Funzione peso: $\psi(x) = 1/\sqrt{1-x^2}$, intervallo di ortogonalità: $[-1, 1]$

$$\int_{-1}^{1} \psi(x) p_n(x) p_m(x)\, dx = \begin{cases} \pi & \text{se } n = m = 0 \\ \pi \cdot 1/2 \cdot \delta_{m,n} & \text{se } n, m > 0, \end{cases}$$

con δ_{mn} simbolo di Kronecker. Relazione ricorrente: $p_0(x) = 1$, $p_1(x) = x$

$$A_k = \begin{cases} 2 & \text{se } k > 0 \\ 1 & \text{se } k = 0 \end{cases}, \quad B_k = 0, \quad C_k = k/(k+1)$$

Polinomi di Jacobi

Funzione peso: $\psi(x) = (1-x)^\alpha (1+x)^\beta$ con $\alpha, \beta > -1$, intervallo di ortogonalità: $[-1, 1]$

$$\int_{-1}^{1} \psi(x) p_m(x) p_n(x)\, dx =$$

$$\frac{2^{\alpha+\beta+1}\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{(2n+\alpha+\beta+1)n!\Gamma(n+\alpha+\beta+1)}\delta_{m,n}, \quad n, m = 0, 1, ...$$

Relazione ricorrente: $p_0(x) = 1$, $p_1(x) = \frac{1}{2}(2+\alpha+\beta)x + \frac{1}{2}(\alpha-\beta)$

$A_k = (2k+\alpha+\beta+2)(2k+\alpha+\beta+1)(2k+\alpha+\beta)$

$B_k = (2k+\alpha+\beta+1)(\alpha^2-\beta^2)$

$C_k = 2(k+\alpha)(k+\beta)(2k+\alpha+\beta+2)$

Si può dimostrare che gli zeri dei polinomi ortogonali:

- non contengono gli estremi dell'intervallo di integrazione definito dal prodotto scalare (intervallo di ortogonalità)
- costituiscono degli insiemi disgiunti al variare del grado del polinomio tra due interi consecutivi.

La verifica di tali proprietà viene fornita nel caso dei polinomi di Chebyshev nell'esempio che segue:

♣ **Esempio C.2.** I polinomi di Chebyshev possono essere rappresentati come:

$$p_k(x) = cos[k\, arccos(x)] \qquad x \in [-1, 1]$$

Infatti, per note proprietà delle formule trigonometriche, si ha:

$$cos[(k+1)\phi] + cos[(k-1)\phi] = 2cos(\phi) \cdot cos(k\phi)$$

da cui, posto $x = cos(\phi)$ e $\tau_k(x) = cos(k\, arccos(x))$ si ha che

$$\tau_{k+1}(x) = 2\tau_1(x)\tau_k(x) - \tau_{k-1}(x) \quad \text{con} \quad \tau_0(x) = 1 \text{ e } \tau_1(x) = x$$

pertanto risulta $\tau_{k+1} \equiv p_{k+1}$. Conseguenza di ciò è che gli zeri di p_k, polinomio di Chebyshev di grado k sono:

$$x_i = cos\left(\frac{2i-1}{2k}\right)\pi \quad i = 1, 2, ..., k$$

da cui si verificano le proprietà prima elencate. Analoghi risultati si possono provare per le altre famiglie di polinomi ortogonali. ♣

C.2 I polinomi di Chebyshev

Definizione C.2.1. (Polinomio di Chebyshev di prima specie)
Il polinonio di Chebyshev di prima specie di grado k, $k \geq 0$, è definito da:

$$T_k(z) = \begin{cases} \cos(k \cos^{-1} z) & -1 \leq z \leq 1,\ k \geq 0 \\ \cosh(k \cosh^{-1} z) & |z| \geq 1,\ k \geq 0 \end{cases} \tag{C.6}$$

Un'altra espressione dei polinomi di Chebyshev è la seguente:

$$T_k(z) = \frac{1}{2}[(z + \sqrt{z^2-1})^k + (z - \sqrt{z^2-1})^k].$$

Un'importante proprietà dei polimomi di Chebyshev è data dal seguente:

Teorema C.2.1. *Nell'intervallo $[-1, 1]$ i polinomi di Chebyshev di prima specie soddisfano la seguente relazione di ricorrenza a tre termini:*

$$\begin{aligned} &T_0(z) = 1;\ \ T_1(z) = z; \\ &T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z),\ \ k \geq 1. \end{aligned} \tag{C.7}$$

Dimostrazione Per $-1 \leq z \leq 1$, si può scrivere $T_k(z) = \cos k\theta$, con $\cos\theta = z$. Ricordando la seguente identità trigonometrica:

$$\cos[(k-1)\theta] + \cos[(k+1)\theta] = 2\cos\theta\cos k\theta,$$

si ha immediatamente l'asserto. ■

Dalla definizione di $T_k(z)$, si ricava che:

$$\begin{aligned} &1) \quad T_k(1) = 1, && \forall k \geq 0; \\ &2) \quad T_k(z_j) = (-1)^j, \quad z_j = \cos(\tfrac{j\pi}{k}), \quad j = 0, 1, ..., k, && \forall k \geq 0; \\ &3) \quad \max_{-1 \leq z \leq 1} |T_k(z)| = 1. \end{aligned} \tag{C.8}$$

Teorema C.2.2. *Sia $T_k(z)$ il polinomio di Chebyshev di prima specie di grado k. Si consideri la funzione $w : [a, b] \longrightarrow [-1, 1]$ così definita:*

$$w(z) = \frac{2z - (b + a)}{b - a}.$$

Sia $\mathcal{P}_k$ l'insieme dei polinomi $p_k(z)$ di grado k nella variabile reale z, tali che $p_k(1) = 1$. Il polinomio $H_k(z)$ definito da:

$$H_k(z) = \frac{T_k(w(z))}{T_k(w(1))}, \quad \forall k \geq 0, \tag{C.9}$$

è l'unico polinomio $\in \mathcal{P}_k$ che risolve il problema:

$$\min_{p_k(1)=1} \left\{ \max_{a \leq z \leq b} |p_k(z)| \right\}, \quad \forall k \geq 0.$$

Dimostrazione Il polinomio $H_k(z)$ è tale che $H_k(1) = 1$. Quindi $H_k(z) \in \mathcal{P}_k$. Inoltre, poichè la funzione $w(z)$ trasforma l'intervallo $a \leq z \leq b$ nell'intervallo $-1 \leq w \leq 1$, si ha:

$$\max_{a \leq z \leq b} |H_k(z)| = \max_{-1 \leq w(z) \leq 1} \frac{|T_k(w(z))|}{|T_k(w(1))|} = \frac{1}{|T_k(w(1))|}.$$

Tale valore massimo è ottenuto, con segni alterni, in $k + 1$ punti distinti, $a = z_0 < z_1 < ... < z_k = b$. Si assuma ora che esista un polinomio $q_k(z) \in \mathcal{P}_k$ tale che:

$$\max_{a \leq z \leq b} |q_k(z)| < \max_{a \leq z \leq b} |H_k(z)|.$$

Considerato il polinomio al più di grado k dato dalla differenza tra i polinomi $q_k(z)$ e $H_k(z)$, $R_k(z) = q_k(z) - H_k(z)$, si ha che nei punti $\{z_j\}_{j=0}^k$ esso cambia segno. Per la continuità di $R_k(z)$ ci devono essere k punti s_j, $1 \leq j \leq k$, con $a \leq z_{j-1} < s_j < z_j \leq b$, tali che $R_k(s_j) = 0$. Poichè $R_k(1) = 0$, il polinomio $R_k(z)$ ha quindi $k + 1$ zeri e di conseguenza è necessariamente il polinomio nullo. Ciò dimostra che $H_k(z)$ è un polinomio di grado k che risolve il problema. Si supponga ora per assurdo che esista un altro polinomio $q_k(z) \in \mathcal{P}_k$ che risolva il problema. Con un ragionamento

analogo a quello precedente, si verifica che il polinomio $R_k(z) = q_k(z) - H_k(z)$ ha $k+1$ zeri e quindi è il polinomio identicamente nullo. Ciò dimostra l'unicità. ■

Teorema C.2.3. *Dati $a, b \in \Re$, con $-1 < a < b < 1$ e considerata la funzione $w : [a,b] \longrightarrow [-1,1]$ così definita:*

$$w(z) = \frac{2z - (b+a)}{b-a},$$

i polinomi $H_k(z)$ dati da (4.99) soddisfano la seguente relazione di ricorrenza:

$$\begin{cases} H_0(z) = 1, \quad H_1(z) = \gamma z + 1 - \gamma; \\ H_{k+1}(z) = \beta_{k+1}(\gamma z + 1 - \gamma)H_k(z) + (1 - \beta_{k+1})H_{k-1}(z), \quad k \geq 1, \end{cases} \tag{C.10}$$

dove:

$$\begin{cases} \gamma = 2/(2 - (b+a)); \\ \beta_{k+1} = 2w(1)T_{(k)}(w(1))/T_{(k+1)}(w(1)). \end{cases} \tag{C.11}$$

Dimostrazione Sulla base della definizione (4.99) dei polinomi $H_k(z)$, è immediato verificare che le prime due uguaglianze delle (C.10) sono vere.
Posto $\alpha_{k+1} = T_{k+1}(w(1))$, dalla (4.99) si ha $\alpha_{k+1}H_{k+1}(z) = T_{k+1}(w(z))$. Quindi, utilizzando la relazione di ricorrenza a tre termini (C.7), si ha:

$$\begin{aligned} H_{k+1}(z) = \quad & \alpha_{k+1}^{-1}T_{k+1}(w(z)) = \alpha_{k+1}^{-1}[2w(z)T_k(w(z)) - T_{k-1}(w(z))] = \\ & = 2\alpha_{k+1}^{-1}\alpha_k w(z)H_k(z) - \alpha_{k+1}^{-1}\alpha_{k-1}H_{k-1}(z). \end{aligned} \tag{C.12}$$

Sempre utilizzando la (C.7) si ottiene inoltre:

$$\alpha_{k+1} = T_{k+1}(w(1)) = 2w(1)T_k(w(1)) - T_{k-1}(w(1)) = 2\alpha_k w(1) - \alpha_{k-1},$$

da cui:

$$1 = 2\alpha_{k+1}^{-1}\alpha_k w(1) - \alpha_{k+1}^{-1}\alpha_{k-1}. \tag{C.13}$$

Posto $\gamma = 2/(2-(b+a))$, si ha, innanzitutto, che $w(1)(\gamma z + 1 - \gamma) = w(z)$. Inoltre, posto $\beta_{k+1} = 2w(1)\alpha_k\alpha_{k+1}^{-1}$:

$$2\alpha_{k+1}^{-1}\alpha_k w(z) = \beta_{k+1}(\gamma z + 1 - \gamma). \tag{C.14}$$

Infine, dalla (C.13) si ha:

$$-\alpha_{k+1}^{-1}\alpha_{k-1} = (1 - \beta_{k+1}). \tag{C.15}$$

Sostituendo le espressioni (C.14) e (C.15) in (C.12) si ha quindi l'asserto. ■

C.3 I polinomi di Stieltjes

Descriviamo ora alcuni polinomi introdotti da Stieltjes nel 1894 che hanno una stretta relazione con le formule di Gauss- Kronrod descritte nel §2.1.5.
Sia l_n il polinomio di Legendre di grado n e sia:

$$q_n(z) = \int_{-1}^{1} \frac{l_n(x)}{z - x}\, dx$$

una funzione di variabile complessa (talvolta chiamata funzione associata di l_n). Tale funzione è definita sull'intero piano complesso tranne che sul segmento reale $[-1, 1]$. È allora possibile scrivere lo sviluppo in serie di potenze (sfruttando anche l'ortogonalità dei polinomi l_n). Per opportune costanti b_i, c_i, d_i:

$$q_n(z) = \frac{1}{z^{n+1}} \left(b_n + \frac{b_{n+1}}{z} + \cdots \right)$$

da cui:

$$\frac{1}{q_n(z)} = z^{n+1} \left(c_n + \frac{c_{n+1}}{z} + \cdots \right)$$

L'ultima espressione può essere riscritta come:

$$\frac{1}{q_n(z)} = s_{n+1}(z) + \frac{d_1}{z} + \frac{d_2}{z^2} + \cdots$$

con $s_{n+1}(z)$ polinomio di variabile complessa che definisce il polinomio reale $s_{n+1}(x)$ di grado $n + 1$ per $x \in [-1, 1]$. Tali polinomi prendono il nome di **polinomi di Stieltjes**. Lo stesso Stieltjes dimostrò l'ortogonalità di tali polinomi rispetto alla famiglia dei polinomi di Legendre l_n, cioè:

$$\int_{-1}^{1} l_n(x) s_{n+1}(x) x^k\, dx = 0 \qquad \forall k = 0, 1, \ldots, n$$

Inoltre Szegö nel 1934 ha dimostrato che gli zeri ξ_j del polinomio di Stieltjes $s_{n+1}(x)$ di grado $n + 1$ hanno la proprietà di "separare" gli zeri x_i del polinomio di Legendre $l_n(x)$ di grado n e di appartenere all'intervallo $]-1, 1[$ cioè

$$-1 < \xi_1 < x_1 < \xi_2 < \ldots < \xi_n < x_n < \xi_{n+1} < 1\,.$$

Bibliografia

[1] Kronrod A. - *Nodes and weights of quadrature formulas* - Consultants bureau, 1965

[2] Monegato G. - *Positivity of weights of extended Gauss-Legendre quadrature rules* - Mathematics of Computation, vol. 32, No. 141, pp. 243-245 (1978).

[3] Monegato G. - *Stieltjes polynomials and related quadrature rules* - SIAM Review, vol. 24, pp. 137-158 (1982).

[4] Szegö G. - *Ortogonal polynomials* - American Mathematical Society Colloquium Publications, Volume XXIII, 1939.

Testi generali di riferimento

Bibliografia

[Atkinson] Atkinson K. E. - *An Introduction to Numerical Analysis* - J. Wiley and Sons, New York, II ed., 1989.

[Cheney] Cheney E. W. - *Introduction to Approximation Theory* - International series in pure and applied mathematics, McGraw-Hill Book Company, New York, 1966.

[Cheney] Cheney W., Kincaid D. R. - *Numerical Mathematics and Computing* - 6th ed., Pacific Grove, CA, Brooks/Cole, 2007.

[Cody] Cody W. and Waite W. - *Software Manual for the Elementary Functions* - Prentice-Hall, Englewood Cliffs, N.J., 1980.

[Comincioli] Comincioli V. - *Analisi Numerica: metodi, modelli, applicazioni* - McGraw-Hill, seconda edizione, 1995.

[Cowell] Cowell W. - *Source and Development of Mathematical Software* - Prentice Hall, 1984.

[Dahlquist,Björck] Dahlquist G., Björck A. - *Numerical Methods* - Prentice-Hall, 1974.

[Dahlquist,Björck] Dahlquist G., Björck A. - *Numerical Methods In Scientific Computing* - vol. 1, Society For Industrial & Applied Mathematics, United States, 2008

[Dahlquist,Björck] Dahlquist G., Björck A. - *Numerical Methods In Scientific Computing* - vol. II
(*working copy*: `http://www.mai.liu.se/~akbjo/NMbook.html`)

[Davis] Davis P. J. - *Interpolation and approximation* - Blaisdell, New York, 1975.

[Demmel] Demmel J. W. - *Applied Numerical Linear Algebra* - SIAM, Philadelphia, 1997.

[Forsythe,Malcom,Moler] Forsythe G., Malcom M. and Moler C. - *Computer methods for mathematical computations* - Prentice-Hall, 1977.

[Forsythe,Moler] Forsythe G., Moler C. - *Computer Solution of linear algebraic systems* - Prentice-Hall, 1967.

[Gear] Gear C. W. - *Applications and Algorithms in Science and Engineering: Module A (Introduction to Computers, Structured Programming and Applications)* - Science Research Associates, 1978.

[Gear] Gear C. W. - *Computer Applications and Algorithms* - Science Research Associates, 1986.

[Gear] Gear C. W. - *Computer Organization and Programming* - Mc Graw Hill, 4th ed., 1985.

[Gear] Gear C. W. - *Introduction to Computer Science* - Science Research Associates, 1971.

[Golub,Van Loan] Golub G. H., Van Loan C. F. - *Matrix Computations* - third ed., The Johns Hopkins University Press, 1997.

[Heath] Heath M. T. - *Scientific Computing: An Introductory Survey* - Second Edition, McGraw-Hill, New York, 2002.

[Higham] Higham N. - *Accuracy and Stability of numerical algorithms* - SIAM, Philadelphia, II ed., 2002.

[Johnston] Johnston R. L. - *Numerical Methods, a Software Approach* - John Wiley & Sons, 1982.

[Kahaner,Moler,Nash] Kahaner D., Moler C., Nash S. - *Numerical Methods and Software* - Prentice-Hall, Inc., 1989.

[Kincaid,Cheney] Kincaid D. and Cheney W. - *Numerical Analysis: Mathematics of Scientific Computing* - 3rd Edition, Brooks/Cole, Pacific Grove, 2002.

[Lanczos] Lanczos C. - *Applied Analysis* - Prentice Hall, Inc., 1956.

[Maron] Maron M. J. - *Numerical Analysis. A Practical Approach. Second edition* - MacMillan Pub. Comp. and Collier MacMillan Pub., 1987.

[Ralston,Rabinowitz] Ralston A., Rabinowitz P. - *A first course in numerical analysis* - Dover Publications, 2nd ed., 2001.

[Ralston,Wilf] Ralston A., Wilf H. - *Mathematical methods for digital computers* - Vol. I (1960) e Vol. II (1967), John Wiley & Sons, Inc.

[Rice] Rice J. R. - *Mathematical Aspects of Scientific Software* - Springer-Verlag, New York, 1988.

[Rice] Rice J. R. - *Matrix Computations and Mathematical Software* - Mc Graw Hill, 1981.

[Rice] Rice J. R. - *Numerical Methods, Software and Analysis* - McGraw-Hill, Second Edition, Academic Press, 1992.

[Rivlin] Rivlin T. - *An Introduction to the Approximation of Functions* - Dover, 1969.

[Sterbenz] Sterbenz Pat H. - *Floating point computation* - Prentice-Hall 1974.

[Stewart] Stewart G. W. - *Afternotes on Numerical Analysis* - SIAM, Philadelphia, 1998.

[Stoer,Bulirsch] Stoer J., Bulirsch R. - *Introduction to numerical analysis* - Springer-Verlag, third edition, 2002.

[Ueberhuber] Ueberhuber C. W. - *Numerical Computation: Methods, Software and Analysis* - Springer-Verlag Berlin and Heidelberg GmbH & Co. K, vol. 1 e vol. 2, 1997.

[Wallis] Wallis P. J. L. - *Improving Floating-Point Programming* - John Wiley & Sons, 1990.

[Watson] Watson G. A. - *Approximation Theory and Numerical Methods* - John Wiley & Sons, 1980.