Laboratorio di Telecomunicazioni - a.a. 2010/2011
Lezione n. 6

Sviluppo in serie di Fourier

docente L.Verdoliva

In questa quinta lezione affrontiamo il problema dell'analisi di un segnale periodico discreto nel dominio della frequenza, attraverso lo sviluppo in serie di Fourier, e della sintesi di tale segnale a partire dal suo spettro. Considereremo, inoltre, la ricostruzione di un segnale periodico continuo mediante un numero finito di armoniche.

1 Serie discreta di Fourier

Un segnale tempo discreto $x(n)$, periodico con periodo fondamentale N_0, può essere rappresentato dalla somma di N_0 esponenziali complessi relazionati armonicamente tra loro, cioè

$$x(n) = \sum_{k=0}^{N_0-1} X_k e^{j2\pi nk/N_0}$$

I coefficienti dello sviluppo sono numeri complessi e contengono le informazioni sull'ampiezza e fase delle N_0 sinusoidi discrete che rappresentano il segnale. I coefficienti dello sviluppo in serie possono essere calcolati mediante la seguente relazione:

$$X_k = \frac{1}{N_0} \sum_{n=0}^{N_0-1} x(n) e^{-j2\pi nk/N_0}$$

E' importante sottolineare che anche la sequenza dei coefficienti di Fourier è periodica dello stesso periodo del segnale, cioè $X_{k+N_0} = X_k$. Ricordiamo poi che X_0 è la componente continua, mentre X_k e X_{N_0-k} sono componenti a frequenza numerica k/N_0 e $-k/N_0$, rispettivamente, e che per segnali reali risultano complesse coniugate (hanno la stessa ampiezza, ma fase opposta), in modo da dar luogo alla ricostruzione mediante sinusoidi.

1.1 Gestione dei numeri complessi

Per implementare in Matlab la serie di Fourier sarà evidentemente necessario definire e gestire numeri complessi. In Matlab questo è molto semplice, ad esempio l'istruzione

```
>> x = complex(a,b);
```

definisce il numero complesso $x = a + jb$ (Matlab usa la i per indicare l'unità immaginaria, ma accetta anche j). Lo stesso effetto ha l'istruzione

```
>> x = a+b*i;
```

Sui numeri complessi sono definiti poi gli operatori `real`, `imag`, `abs`, `angle` e `conj`, con ovvio significato. Si noti che per matrici complesse, `X'` indica la trasposta Hermitiana, mentre per avere la trasposta semplice bisogna usare `X.'`. Per calcolare la serie di Fourier ci interessano soprattutto gli esponenziali complessi, che si possono definire nel seguente modo:

```
>> n=[0:N-1]; X = exp(-j*2*pi*n);
```

In questo modo si genera un vettore complesso X di lunghezza N.

1.2 Implementazione in Matlab

Proviamo a scrivere il codice in uno script per ricavare i coefficienti dello sviluppo in serie di un segnale periodico. Una prima soluzione è quella di implementare meccanicamente la formula mediante due cicli for innestati nel seguente modo

```
X = zeros(1,N);
for k=0:N-1,
   for n=0:N-1,
      X(k+1) = X(k+1)+x(n+1)*exp(-j*2*pi*k*n/N);
   end;
end;
X = X/N;
```

N.B. Per maggiore leggibilità del codice scriviamo N al posto di N_0.
Questa soluzione comporta una bassa occupazione in memoria, a costo di elevati tempi di calcolo. Il numero di cicli for può essere ridotto a uno solo calcolando ogni campione dello spettro come il prodotto scalare i campioni nel tempo e gli esponenziali complessi:

```
n = [0:N-1];
for k=0:N-1
    X(k+1) = x*exp(-j*2*pi*k*n/N).';
end;
X = X/N;
```

N.B. Ricordate che il calcolo del prodotto scalare si può ottenere anche scrivendo

```
X(k+1) = sum(x.*exp(-j*2*pi*k*n/N));
```

In realtà è possibile evitare del tutto cicli for se si riconosce che il calcolo degli N coefficienti di Fourier può essere visto come un prodotto matriciale, infatti risulta:

$$\begin{aligned}
X_0 &= \frac{1}{N_0}(x(0) + x(1) + \cdots + x(N_0-1)) \\
X_1 &= \frac{1}{N_0}(x(0) + x(1)\, e^{-j2\pi/N_0} + \cdots + x(N_0-1)\, e^{-j2\pi(N_0-1)/N_0}) \\
\ldots &= \ldots \\
X_{N_0-1} &= \frac{1}{N_0}(x(0) + x(1)\, e^{-j2\pi(N_0-1)/N_0} + \cdots + x(N_0-1)\, e^{-j2\pi(N_0-1)(N_0-1)/N_0})
\end{aligned}$$

Detti i vettori colonna $\mathbf{X} = [X_0, X_1, \ldots, X_{N_0-1}]$, $\mathbf{x} = [x_0, x_1, \ldots, x_{N_0-1}]$ e $\mathbf{W}$ la seguente matrice:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & e^{-j2\pi/N_0} & e^{-j4\pi/N_0} & ... & e^{-j2\pi(N_0-1)/N_0} \\ ... & ... & ... & ... & ... \\ 1 & e^{-j2\pi(N_0-1)/N_0} & e^{-j4\pi(N_0-1)/N_0} & ... & e^{-j2\pi(N_0-1)(N_0-1)/N_0} \end{bmatrix}$$

risulta che $\mathbf{X} = \frac{1}{N_0}\mathbf{W}\mathbf{x}$. In Matlab si ha:

```
n = [0:N-1];
k = [0:N-1];
W = exp(-j*2*pi*k'*n/N);
x=x.';                      % necessario se x è un vettore riga
X = (W*x)/N;
```

Lo spettro è calcolato quindi mediante una trasformazione lineare del segnale.
Con ragionamento analogo si perviene alla formula di ricostruzione: $\mathbf{x} = \mathbf{W}^*\mathbf{X}$.

```
x = W'*X;
```

1.3 Esempi ed esperimenti proposti

a) *Analisi e sintesi di un segnale periodico.* Scrivete una funzione che calcola i coefficienti dello sviluppo in serie di Fourier e un'altra che ricostruisce il segnale dal suo spettro, usando i seguenti prototipi:

```
function X = dfs(x);
function x = idfs(X);
```

dove `x` è il generatore del segnale e `X` è il vettore che contiene i suoi coefficienti di Fourier.

b) *Treno di impulsi rettangolari.* Usando la funzione `dfs`, scriviamo uno script per rappresentare graficamente spettro di ampiezza e di fase di un treno di impulsi rettangolari, in cui L è la durata dell'impulso e N il periodo del segnale.

```
x = [ones(1,L) zeros(1,N-L)];
X = dfs(x);
k=[0:N-1];
subplot(211); stem(k,abs(X),'filled'); title('Ampiezza');
subplot(212); stem(k,angle(X),'filled'); title('Fase');
```

Come primo esperimento consideriamo $L = 4$ e $N = 20$.

Fate attenzione alla visualizzazione dello spettro che è stato rappresentato nell'intervallo $[0, N[$, ovvero nell'intervallo $[0, 1[$ se si fa riferimento al valore di ν (frequenza normalizzata). Poiché le basse frequenze si trovano intorno a 0 e 1 (agli estremi dell'intervallo), mentre le alte frequenze intorno a 1/2 (al centro dell'intervallo), questa visualizzazione non è quella abituale, in cui le basse frequenze si trovano intorno all'origine e le alte frequenze lontano

dall'origine. D'altra parte è del tutto lecito, vista la periodicità dei coefficienti, riportare a destra della componente continua le componenti della seconda metà dello spettro, in modo da avere uno spettro più leggibile:

```
k=[-N/2:N/2-1];       % asse temporale centrato nell'origine
X=fftshift(X);        % traslazione dei coefficienti di Fourier
subplot(211); stem(k,abs(X),'filled'); title('Ampiezza');
subplot(212); stem(k,angle(X),'filled'); title('Fase');
```

La definizione dell'asse temporale però è valida solo quando N è pari, se consideriamo un valore dispari di N, l'asse temporale va definito nel seguente modo:

```
k=[(-(N-1)/2):((N-1)/2)];
```

Questo è il motivo per cui storicamente la serie discreta è sempre definita tra 0 e $N-1$.

- Provate adesso ad aumentare il periodo del segnale per $N = 30, 40, 50, ...$, lasciando L fisso, che considerazioni potete fare dall'osservazione dello spettro del segnale?
- Cosa succede invece se si aumenta anche il valore di L?

Considerazioni:

(a) Notate che la simmetria della fase è verificata in tutti punti, eccetto quelli corrispondenti ad un modulo pari a zero, in cui la fase risulta essere indeterminata. Il Matlab, a causa della precisione finita, si trova valori non identicamente nulli, cui associa il corrispondente valore di fase.

(b) Se il segnale è reale lo spettro di ampiezza è pari, mentre quello di fase è dispari, quindi in tal caso è possibile ridurre il costo computazionale, calcolando un numero inferiore di coefficienti di Fourier.

2 Serie continua di Fourier

Per un segnale periodico tempo continuo valgono le seguenti relazioni:

$$\begin{aligned} x(t) &= \sum_{k=-\infty}^{\infty} X_k e^{j2\pi k f_0 t} && \text{(equazione di sintesi)} \\ X_k &= \frac{1}{T_0}\int_{T_0} x(t) e^{-j2\pi k f_0 t} dt && \text{(equazione di analisi)} \end{aligned}$$

dove T_0 è il periodo del segnale, il suo reciproco $f_0 = 1/T_0$ è detta frequenza fondamentale, e l'integrale è esteso ad un periodo qualsiasi. Se il segnale è reale, si può usare la forma più significativa

$$x(t) = X_0 + \sum_{k=1}^{\infty} 2|X_k| \cos(2\pi k f_0 t + \angle X_k)$$

dalla quale si vede che $x(t)$ si può ricostruire come somma della componente continua (DC) X_0, pari alla media del segnale, più infinite componenti sinusoidali (AC) dette anche "armoniche", a frequenze $f_0, 2f_0, \ldots, kf_0, \ldots$, con ampiezza e fase opportune (definite proprio dai coefficienti di Fourier).

E' evidente che il calcolo dei coefficienti di Fourier attraverso un integrale non è possibile in Matlab, pertanto calcoleremo analiticamente tali coefficienti e li useremo per ricostruire il segnale con un numero finito di armoniche. In questo modo potremo valutare come migliora la rappresentazione del segnale all'aumentare del numero di armoniche e mostrare il cosiddetto "fenomeno di Gibbs" nel caso in cui il segnale presenti punti di discontinuità. In questi punti, infatti, la serie converge alla media fra il limite destro e quello sinistro, per cui anche considerando un numero elevato (ma finito) di armoniche non sarà possibile approssimare fedelmente il segnale nell'intorno di tali discontinuità.

2.1 I segnali "analogici" in Matlab

A rigore, non è possibile effettuare esperimenti in Matlab con segnali analogici, dato che un computer è in grado di gestire solo informazione numerica e dunque non è possibile in alcun momento disporre realmente della versione a tempo ed ampiezza continue del segnale.

Tuttavia, è possibile disporre di segnali "quasi" analogici, nel senso che gli istanti discreti di tempo possono essere tanto vicini fra loro da dare la sensazione della continuità (come i fotogrammi di un film) e le ampiezze discrete possono essere rappresentate con una precisione tale che l'errore risulti rilevabile solo attraverso un'analisi approfondita. In effetti, per quanto riguarda le ampiezze, il Matlab garantisce già un'elevatissima precisione, dato che rappresenta tutte le variabili in *double precision*, con 8 byte per ogni scalare. Per quanto riguarda i tempi, invece, dobbiamo noi campionare l'asse dei tempi in modo tanto fitto da garantire che il segnale sia pressochè costante lungo un intervallo di campionamento. Supponiamo di voler rappresentare graficamente la seguente funzione sinusoidale tempo continuo:

$$x_a(t) = 20\cos(2\pi(40)t - 0.4\pi)$$

Innanzitutto dobbiamo ottenere una rappresentazione discreta del segnale, a tal fine basta valutare $x(t)$ in determinati istanti di tempo $t_n = nT$, dove T rappresenta il passo di campionamento. In questo modo si ha la sequenza di campioni:

$$x(n) = x_a(nT) = 20\cos(2\pi(40)nT - 0.4\pi)$$

Per visualizzare questo segnale usiamo la seguente sequenza di comandi:

```
T=0.005;                        % passo di campionamento
tn=[-0.025:T:0.025];            % istanti di campionamento
xn=20*cos(80*pi*tn-0.4*pi);
plot(tn,xn);
axis([-0.025 +0.025 -20 20]); hold on;
stem(tn,xn,'filled'); hold off;
```

Il comando `plot` realizza un'interpolazione lineare, unisce cioè con dei segmenti i punti corrispondenti ai valori discreti della sinusoide. Appare evidente che il passo di campionamento

fornito (ν_0 =5 campioni per ciclo) non è sufficientemente piccolo da garantire una visualizzazione corretta del segnale. E' necessario diminuire T: si provi con un passo pari a $T/2 = 0.0025$ (10 campioni per ciclo) e con un passo pari a $T/40 = 0.000125$ (250 campioni per ciclo). Nel primo caso il numero di campioni usati è ancora insufficiente a rappresentare il segnale, nel secondo caso risultano più di quelli strettamente necessari. In generale, maggiore è il numero di campioni per ciclo, più accurata è la curva interpolata linearmente. E' evidente che la scelta di T dipende dalla frequenza della sinusoide, si provi ad aumentare f_0 a 2000 Hz, e si verifichi come un passo di campionamento di 0.000125 (5 campioni per ciclo) non garantisce una buona ricostruzione.

Queste considerazioni valgono ogni volta che vogliamo rappresentare un segnale analogico. Supponiamo, per esempio, di voler ottenere il grafico del segnale $x(t) = \text{rect}(2\text{t}/\text{T})$:

```
T = 10; dt = 0.01; t = [-T/2:dt:T/2];
x = (abs(t)<(T/4));
plot(t,x); grid on; xlabel('t'); ylabel('x(t)'); axis([-T/2 T/2 -1 2]);
```

Fate allora attenzione al passo di campionamento, che deve essere sufficientemente piccolo da garantire la corretta visualizzazione del segnale

2.2 Ricostruzione con un numero finito di armoniche

Dato un segnale periodico $x(t) = \text{rep}_T[x_g(t)]$ vogliamo valutare l'effetto della ricostruzione con un numero finito di armoniche, cioè approssimiamo il segnale con la somma di K armoniche:

$$\widehat{x}(t) = X_0 + \sum_{k=1}^{K} 2|X_k| \cos(2\pi k f_0 t + \angle X_k)$$

Ricordiamo che la formula di ricostruzione si semplifica per segnali pari nel seguente modo

$$\widehat{x}(t) = X_0 + \sum_{k=1}^{K} 2X_k \cos(2\pi k f_0 t)$$

mentre per segnali dispari risulta:

$$\widehat{x}(t) = \sum_{k=1}^{K} 2jX_k \sin(2\pi k f_0 t)$$

2.3 Esempi ed esperimenti proposti

a) *Treno di impulsi rettangolari.* Il segnale periodico $x(t) = \text{rep}_{T_0}[A\text{rect}(t/T)]$ ha coefficienti di Fourier $X_k = (AT/T_0)\text{sinc}(kT/T_0)$. Essendo il segnale pari, i coefficienti sono reali ed è possibile ricostruire il segnale come somma di soli coseni. Scriviamo il seguente script per visualizzare come viene ricostruito il segnale all'aumentare del numero di armoniche:

```
clc; clear all; close all;
K = input('Numero di armoniche: ');
A = input('Ampiezza del generatore: ');
```

```
T = input('Durata del generatore: ');
T_0 = input('Periodo del segnale: ');
t1=-T_0; dt=0.01; t2=T_0;
t=[t1:dt:t2];
n=[-K:K];
X_0 = A*T/T_0;
x   = X_0;
X_k = [zeros(1,K) X_0 zeros(1,K)];
figure(1);
for k=1:K
    subplot(2,1,1); stem(n,X_k,'filled');
    subplot(2,1,2); plot(t,x); pause;
    val_k = (A*T/T_0)*sinc(k*T/T_0);
    X_k(find((n==k)|(n==-k))) = val_k;
    x = x + 2*val_k*cos(2*pi*k*t/T_0);
end
```

Come primo esperimento fissate i seguenti valori $K = 40$, $A = 1$, $T = 2$ e $T_0 = 4$; notate come all'aumentare del numero di armoniche migliori la ricostruzione del segnale, tuttavia nei punti di discontinuità il segnale presenta delle fluttuazioni (*ripple*), e indipendentemente dal numero di armoniche usate nella ricostruzione il segnale presenta un valore massimo pari circa a 1.09A (fenomeno di Gibbs).

Adesso provate ad aumentare solo il periodo del segnale T_0, che considerazioni potete fare dall'osservazione dell'andamento dei coefficienti di Fourier e del segnale ricostruito?

b) *Treno di impulsi triangolari.* Convertite lo script dell'esempio precedente in una funzione

```
function [x,t] = sintesi(K,A,T,T_0);
```

e ripetete l'esperimento considerando un treno di impulsi triangolare: $x(t) = \mathrm{rep}_{T_0}[A\Lambda(t/T)]$, ricordando che in questo caso i coefficienti di Fourier sono $X_k = (AT/T_0)\mathrm{sinc}^2(kT/T_0)$. Potete notare come adesso bastano poche armoniche per ricostruire quasi perfettamente il segnale, ciò coerentemente col fatto che che l'onda triangolare non presenta discontinuità e quindi il suo contenuto frequenziale è concentrato alle basse frequenze (i coefficienti decadono come $1/k^2$).

c) *Errore di rappresentazione.* Per misurare l'errore che si commette nell'approssimare il segnale con un numero finito di armoniche si può rappresentare su di un grafico la differenza tra il segnale originale $x(t)$ e quello ricostruito: $x(t) - \widehat{x}(t)$. Una misura sintetica dell'errore è la media della differenza al quadrato, detta anche MSE (Mean Square Error), ovvero errore quadratico medio:

```
MSE = mean((x-xhat).^2)
```

Realizzate un grafico in cui mostrate come varia l'MSE all'aumentare del numero di armoniche per i due segnali visti negli esempi precedenti e commentate i risultati.