Agostino De Marco
Domenico P. Coiro

Elementi

di

Dinamica e simulazione di volo

Quaderno 15

Simulazione di volo con il software JSBSim

Marzo 2017
ver. 2017.a

Dichiarazione di Copyright

Quaderno 15

Simulazione di volo con il software JSBSim

Can we actually 'know' the universe?
My God, it's hard enough finding your way around in Chinatown.
– Woody Allen

Indice

15.1 Desktop simulation con JSBSim

JSBSim (www.jsbsim.org) è un software open source che permette di eseguire simulazioni di volo ingegneristiche sulla base di opportuni modelli matematici di velivolo. Nel gergo degli specialisti della simulazione del volo, JSBSim è una *Flight Dynamics Model* (FDM) *library*. Questa libreria fornisce le funzionalità necessarie per: (*a*) definire completamente le caratteristiche di un aeromobile, (*b*) definire le condizioni iniziali della simulazione, (*c*) definire gli input ambientali e gli input provenienti dal pilota, (*d*) integrare le equazioni del moto, (*e*) generare in output i dati richiesti (ad esempio posizione, assetto, velocità e altri parametri di stato e di controllo) in un opportuno formato.

Il software JSBSim può essere integrato all'interno di altri programmi di simulazione. Un esempio è dato dai *desktop flight simulator*, cioè quelle applicazioni per personal computer — come FlightGear (www.flightgear.org), X-Plane (www.x-plane.com/) o Microsoft Flight Simulator X (www.microsoft.com/games/flightsimulatorx) — che attraverso l'uso di librerie grafiche sofisticate e di dispositivi di input dedicati (joystick e joypad) offrono all'utente la percezione (visiva e non solo) del volo e del pilotaggio.

Figura 15.1 La diffusione a livello mondiale della comunità di utenti di JSBSim (dati aggiornati fino all'anno 2011).

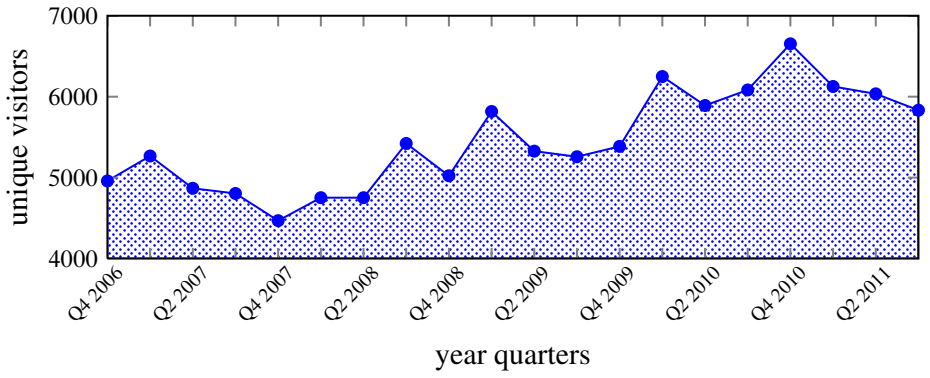


Figura 15.2 Numero di visitatori unici del sito `www.jsbsim.org` dal 2006 al 2011.

Queste applicazioni poggiano il loro funzionamento sulla rappresentazione grafica del velivolo e di altri oggetti in uno scenario prescelto. La rappresentazione, oltre che dalla sofisticazione degli effetti grafici, dipende fondamentalmente da come è gestito lo stato dell'aeromobile e da come esso è aggiornato nel tempo, in funzione degli input dell'utente e del modello di ambiente esterno.

Dal punto di vista del pilotaggio, il grado di realismo dei simulatori di volo è tanto maggiore quanto più è accurato e complesso il *Flight Dynamics Model* con il quale si rappresenta il velivolo e la sua interazione con l'esterno.

JSBSim è utilizzato come FDM di default nel simulatore di volo FlightGear, un'applicazione open source supportata da un'ampia comunità di utenti e di sviluppatori. FlightGear è un software a codice aperto nato per offrire un'alternativa alle costose applicazioni di *desktop flight simulation* ed è oggi da considerarsi un'applicazione sofisticata e professionale, grazie soprattutto alle notevoli possibilità di configurazione. Il processo di *build* di FlightGear prevede la compilazione di JSBSim in forma di libreria statica; questa viene

collegata insieme ad altre librerie per poi produrre il file eseguibile di FlightGear.

Un altro esempio notevole di uso di JSBSim è il software open source di simulazione virtuale OpenEaagles (`www.openeaagles.org`). Questo framework permette di costruire applicazioni di simulazione distribuite *human-in-the-loop* utilizzando il FDM realistico di JSBSim per le istanze dei velivoli.

Oltre alla modalità d'uso come FDM incorporato in altri simulatori, JSBSim può essere usato in una modalità *standalone*, nella quale il programma può processare i comandi impartiti in modo *non* interattivo. Per questo scopo tra i sorgenti della libreria viene distribuito anche il sorgente `JSBSim.cpp`, un programma che può essere compilato per ottenere un'applicazione a sé stante, nominata `JSBSim.exe` sui sistemi Windows o `JSBSim` sui sistemi Unix/Linux e Mac OS X. Il programma `JSBSim` viene eseguito attraverso una finestra dei comandi; con delle opportune opzioni passate dalla riga di comando, esso accetta in input determinati file e genera un output configurabile dall'utente.

L'esecuzione del programma `JSBSim` è *non* interattiva, nel senso che le azioni esterne sono programmate *a priori* e rappresentate in un apposito linguaggio di *scripting* sotto forma di "eventi" della simulazione. Per tale motivo queste simulazioni sono dette *batch simulations*, al contrario di quelle interattive che vengono dette *pilot-in-the-loop simulations*. Quando si vogliono eseguire delle simulazioni batch con JSBSim è possibile inserire sia le configurazioni iniziali che gli eventi in un unico file di input detto *script file*. Ad esempio, si potrà avere uno script file dal nome `take_off_run_B737.xml` da poter fornire in input a `JSBSim`. Digitando alla riga di comando:

```
$ JSBSim --script=take_off_run_B737.xml ⟨invio⟩
```

si lancerà la simulazione di una corsa di decollo del velivolo Boeing 737. Più avanti si forniranno gli elementi per comprendere la logica degli script file.

15.2 Installare JSBSim

Se non interessa compilare `JSBSim` sulla propria macchina sarà possibile non approfondire tutti i dettagli forniti in questa sezione e scaricare subito un'archivio contenente i file binari pre-compilati all'indirizzo: `wpage.unina.it/agodemar/DSV-DQV/JSBSim.rar`. Si troveranno gli eseguibili per le piattaforme Windows, Linux (Ubuntu) e Mac OS X, ottenuti compilando una versione recente di JSBSim. Conviene tuttavia non trascurare la lettura di questa parte poiché essa contiene informazioni utili a capire l'organizzazione dei file nell'albero delle cartelle della distribuzione ufficiale di JSBSim.

Per usare `JSBSim` è necessario utilizzare la console dei comandi del proprio sistema operativo (questo argomento verrà introdotto poco più avanti). Si deve arrivare a preparare un'apposita cartella, che chiameremo ⟨*JSBSim root*⟩, nella quale copiare il file eseguibile `JSBSim` e dalla quale lanciare le simulazioni. Gli utenti meno esperti possono approfittare di questa circostanza per acquisire un po' di familiarità con la riga di comando.

15.2.1 Repository con la distribuzione ufficiale

I sorgenti di JSBSim sono liberamente scaricabili da un archivio online di riferimento ospitato da SourceForge (`sourceforge.net`). Le istruzioni su come scaricare il software dalla repository ufficiale si possono consultare all'indirizzo `jsbsim.sourceforge.net/`

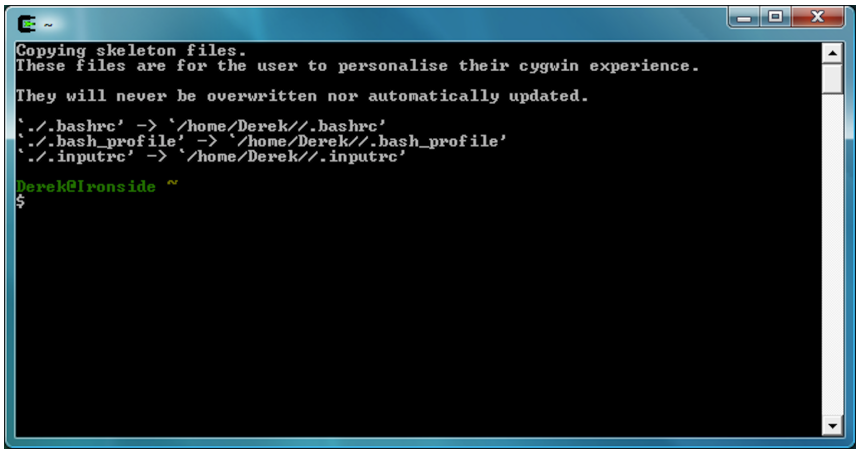


Figura 15.3 La shell dei comandi bash di Cygwim.

`download.html`. Qui ci interessa illustrare con qualche dettaglio in più la procedura per scaricare e compilare la versione più recente della libreria.

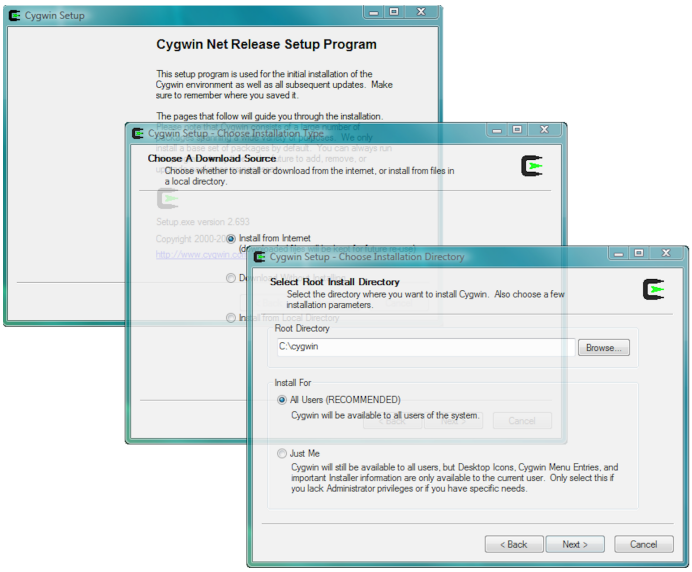
Si deve tenere in mente che un progetto open source come JSBSim è supportato da una comunità di utenti e di sviluppatori molto attiva, come si vede dalle figure 15.1 e 15.2. Questo fa sì che, nel corso del tempo, vengano corretti errori (*bug fixes*), aggiornate alcune caratteristiche (*code refinements*), aggiunte nuove funzionalità richieste dagli utenti (*new features*) o eliminati file obsoleti (*code clean-up*). Per questo motivo la repository ufficiale del codice sorgente viene aggiornata con una certa frequenza (spesso anche per modifiche minori) e conviene scaricare versioni recenti di JSBSim per usare una distribuzione che incorpora tutte le funzionalità più aggiornate. Per navigare nella distribuzione ufficiale del codice sorgente si vedano i seguenti indirizzi: `jsbsim.cvs.sourceforge.net` (espandere l'albero facendo clic su "JSBSim") oppure `jsbsim.git.sourceforge.net` (espandere l'albero facendo clic su "tree").

15.2.2 Riga di comando

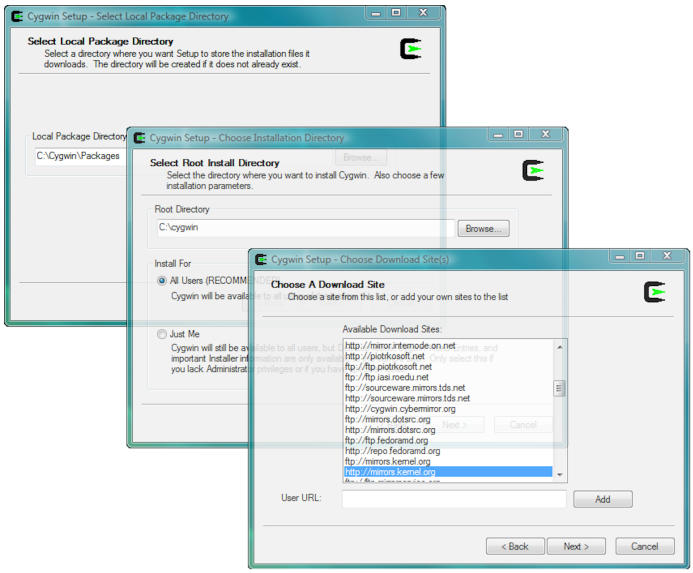
Prima di andarsi a procurare il codice di JSBSim è necessario preparare l'ambiente di lavoro. A seconda se si sta utilizzando un sistema operativo Windows, Linux o Mac OS X si dovrà lavorare con strumenti leggermente diversi. In ogni caso, si dovrà utilizzare una finestra dei comandi. Una finestra dei comandi — detta anche "console" o "shell" — è quell'ambiente che consente di eseguire delle attività di amministrazione del computer senza utilizzare l'interfaccia grafica del sistema operativo ma scrivendo semplicemente dei comandi e premendo il tasto d'invio.

Finestre di comando in Windows

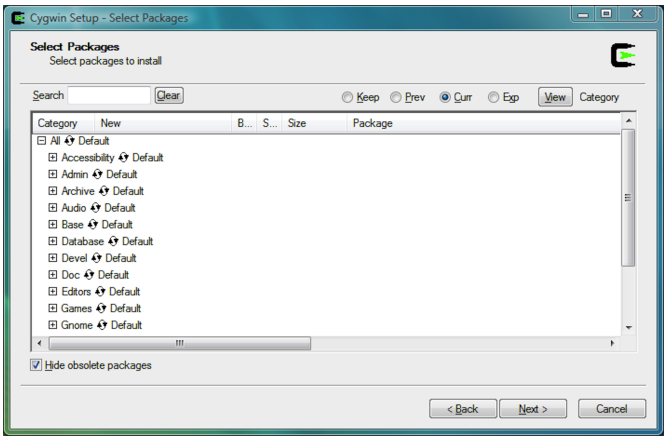
In Windows la finestra dei comandi è chiamata "prompt dei comandi". Per aprire una finestra del prompt dei comandi, l'utente deve fare clic sul pulsante Start, scegliere Tutti i programmi, Accessori e quindi Prompt dei comandi. Il prompt dei comandi è una funzionalità di Windows che rappresenta il punto di ingresso per la digitazione dei comandi MS-DOS (Microsoft Disk Operating System) e di altri comandi per il computer. Attraverso la finestra dei comandi di Windows è possibile eseguire attività di amministrazione anche sofisticate. Ad esempio si possono raccogliere sequenze di comandi ed



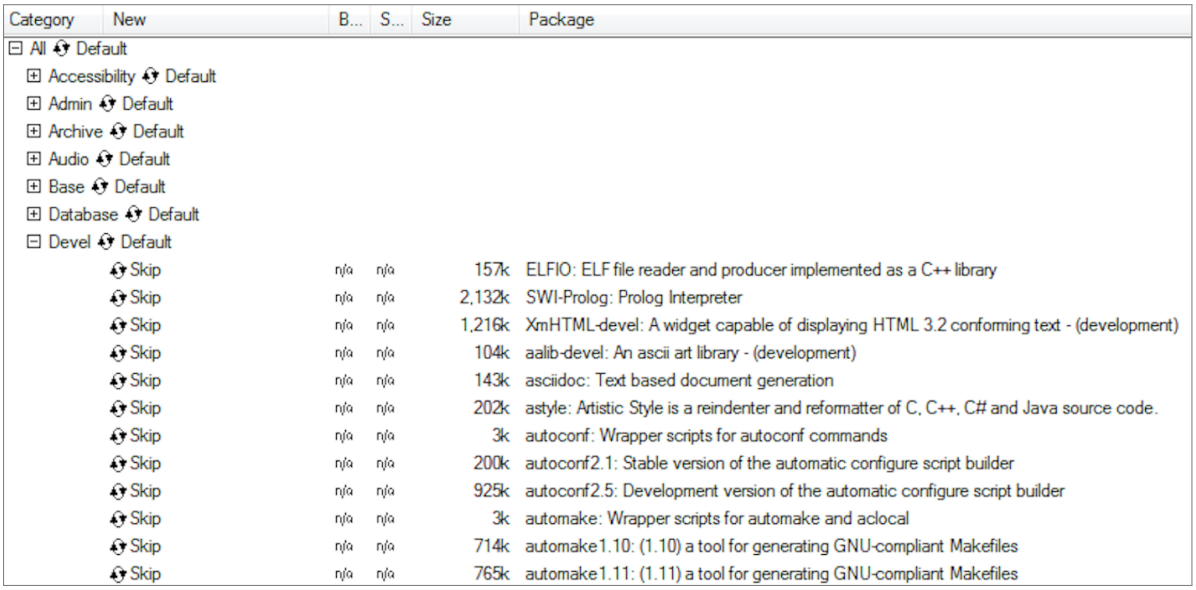
(a) Avvio del setup, scelta della modalità di installazione (di default, da internet) e della cartella di installazione (di default, `c:\cygwin`).



(b) Scelta del mirror dal quale scaricare i pacchetti. Un mirror italiano è `ftp://bo.mirror.garr.it`.



(c) Pannello di selezione dei pacchetti installabili.



(d) Espansione della categoria di pacchetti "Devel" dal quale selezionare `gcc`, `gcc-core`, `gcc-g++`, `make`, `automake`, `autoconf`, `binutils`, `libtool`.

Figura 15.4 Alcune schermate del setup di Cygwin. La procedura viene lanciata dal programma `cygwin.com/setup.exe` che permette di scegliere l'istallazione dei diversi pacchetti della distribuzione.

inserirle in file di estensione `.bat` (*DOS batch scripts*); queste procedure batch sono eseguibili dall'interprete dei comandi DOS.

Oltre al prompt dei comandi DOS gli utenti hanno la possibilità di usare Cygwin (`www.cygwin.com`), un software che permette di sfruttare, anche su una macchina con sistema operativo Windows, la potenza e la flessibilità delle shell dei comandi di Linux. Cygwin offre di default la shell *bash* (*Bourne Again SHell*) attraverso la quale è possibile utilizzare molti dei software tipici del mondo Linux, tra cui la *GNU Compiler Collection* (GCC). In pratica, Cygwin rappesenta un emulatore di terminale dei comandi di Linux all'interno di Windows. Una schermata della shell bash di Cygwin è mostrata nella figura 15.3 a pagina 6. Alcuni passi del procedimento di installazione di Cygwin sono mostrati dalla figura 15.4 nella pagina precedente.

Per una semplice guida in italiano all'installazione di Cygwin si veda il documento "L'installazione di Cygwin: un tutorial per immagini", di Fabio Ruini, `www.fabioruini.eu/unimore/labI/tutorial_cygwin_rev.pdf`. Una valida guida in inglese può essere consultata all'indirizzo `www.physionet.org/physiotools/cygwin/`. Su YouTube è possibile trovare diversi video-tutorial; un esempio è il seguente: `youtu.be/oImEq9JpF8Q`.

Finestre del terminale in Linux e Mac OS X

Sia in Linux che in Mac OS X sono fornite direttamente dal sistema operativo delle specifiche applicazioni che svolgono il ruolo di console dei comandi. Nel mondo Linux una shell molto usata è `xterm`. In Mac OS X esiste l'applicazione `Terminale`. Le funzionalità di queste applicazioni sono molto simili. Per gli utenti Mac è necessario installare del software aggiuntivo, gli Apple Developer Tools (`developer.apple.com/technologies/tools/`), per poter utilizzare i compilatori GCC.

⋆ ⋆ ⋆

In pratica, sia in Windows (con una shell bash di Cygwin) che in Linux (con `xterm`) che in Mac OS X (con `Terminale`) è possibile lavorare da console dei comandi secondo modalità sostanzialmente dentiche. In tutti e tre gli ambienti è possibile disporre di:

- utilità di amministrazione del sistema (comandi di base e programmi vari come `cd`, `ls`, `pwd`, `cp`, `mv`, `mkdir`, `grep`, `sed`, `gawk`, `tar`, eccetera),
- compilatore multitarget GCC (in particolare `gcc` per il linguaggio C, e `g++` per il linguaggio C++),
- strumenti di configurazione automatica delle compilazioni (in particolare i programmi `autoconf`, `automake`, `make`).

Per un approfondimento sulla shell bash si veda la "Guida avanzata di scripting Bash – un'approfondita esplorazione dell'arte dello scripting di shell" di Mendel Cooper (`wpage.unina.it/agodemar/DSV-DQV/guida_bash.pdf`). Per un approfondimento sui compilatori GCC si veda "Una introduzione a GCC — per i compilatori GNU `gcc` e `g++`" di Brian Gough, traduzione in lingua italiana di Andrea Montagner (`wpage.unina.it/agodemar/DSV-DQV/introgcc1_0.pdf`).

15.2.3 I programmi `cvs` e `git`

Il *Concurrent Versioning System* (CVS) è un sistema di controllo delle versioni di un software. Il controllo delle versioni è un concetto fondamentale per quelle applicazioni che vengono sviluppate da più programmatori che lavorano contemporaneamente su diversi

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ ls
AUTHORS                 JSBSimScript.xsd        autom4te.cache          examples
COPYING                 JSBSimScript.xsl        check_cases             include
CVS                     JSBSimSystem.xsd        config.guess            install-sh
ChangeLog               Makefile                config.log              libtool
INSTALL                 Makefile.am             config.status           ltmain.sh
JSBSim.cbx              Makefile.in             config.sub              missing
JSBSim.dox              NEWS                    configure               scripts
JSBSim.vcproj           README                  configure.in            src
JSBSim.vcxproj          aclocal.m4              data_output             systems
JSBSim.xsd              admin                   data_plot
JSBSim.xsl              aircraft                depcomp
JSBSimMainLoopFlow.dot  autogen.sh              engine
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$
```

Figura 15.5 Lista di file e cartelle nella directory principale di una distribuzione ufficiale di JSBSim. In questo caso particolare si vede una schermata del terminale di Mac OS X e la cartella ⟨*JSBSim root*⟩ è `/Users/agodemar/agodemar/jsbsim/JSBSim`.

aspetti del codice sorgente. In pratica `cvs` è anche il nome di un programma (`ximbiot.com/cvs/`) che permette di gestire e sincronizzare tutto il lavoro e tutti i cambiamenti in un dato insieme di file. Tipicamente questo insieme di file è l'implementazione di un software (che sia esso in fase di progetto, in via di sviluppo o già rilasciato) e `cvs` permette a molti sviluppatori (potenzialmente distanti) di collaborare.

L'applicazione `cvs` è installabile in Windows attraverso il programma di setup di Cygwin. In Linux e in Mac OS X `cvs` è normalmente uno dei programmi pre-installati.

In particolare, un programma come `cvs` permette di copiare in maniera anonima una intera repository di un software open source. Questa è la caratteristica che interessa a chi vuole procurarsi il codice sorgente di JSBSim scaricandolo dall'indirizzo ufficiale. Inoltre, una volta effettuata una copia locale della repository, sarà possibile, sempre tramite `cvs`, scaricare eventuali aggiornamenti del codice.

Il programma `git` (`git-scm.com/`) è un'alternativa a `cvs`. In Windows si può installare `git` sia attraverso Cygwin che attraverso un programma di setup dedicato. Anche per Mac OS X esiste un programma di installazione dedicato mentre per il mondo Linux esistono svariate possibilità di scaricare pacchetti pre-compilati o di ottenere i sorgenti per poi compilarli localmente.

15.2.4 Scaricare una distribuzione aggiornata di JSBSim

Per procurarsi una distribuzione aggiornata di JSBSim basta aprire una console dei comandi, spostarsi in un'opportuna cartella di lavoro e dare i seguenti comandi:

```
$ cvs -d:pserver:anonymous@jsbsim.cvs.sourceforge.net:/cvsroot/↩
   jsbsim login ⟨invio⟩
⟨invio⟩ (alla richiesta di password)
$ cvs -z3 -d:pserver:anonymous@jsbsim.cvs.sourceforge.net:/cvsroot/↩
   jsbsim co -P JSBSim ⟨invio⟩
```

☛ Il simbolo "↩" nell'esempio precedente indica che la riga di comando si è spezzata per motivi di spazio. In bash l'utente deve scrivere l'intero comando su un'unica riga senza andare a capo. In alternativa può dare ␣\⟨*invio*⟩ e continuare il comando su una nuova riga.

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd aircraft/
MacBook-Pro-di-Agostino-De-Marco:aircraft agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/aircraft
MacBook-Pro-di-Agostino-De-Marco:aircraft agodemar$ ls
737                     LM                      ah1s                    fokker50
A320                    MD11                    aircraft_template.xml   minisgs
A4                      Makefile                ball                    mk82
B17                     Makefile.am             blank                   p51d
B747                    Makefile.in             c172p                   pa28
Boeing314               OV10                    c172r                   paraglider
C130                    SGS                     c172x                   pc7
CVS                     Short_S23               c182                    pogo-jsbsim
Concorde                Shuttle                 c310                    sgs126
DHC6                    Submarine_Scout         dr1                     sgs233
F4N                     T37                     f104                    t6texan2
F80C                    T38                     f15                     weather-balloon
J246                    X15                     f16                     x24b
L17                     XB-70                   f22
L410                    ZLT-NT                  fokker100
MacBook-Pro-di-Agostino-De-Marco:aircraft agodemar$
```

Figura 15.6 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/aircraft/`.

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd engine
MacBook-Pro-di-Agostino-De-Marco:engine agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/engine
MacBook-Pro-di-Agostino-De-Marco:engine agodemar$ ls
AJ26-33A.xml                    Makefile                    Tay-651.xml                 propC6v.xml
AJ26-33_nozzle.xml              Makefile.am                 WrightGR-2600.xml           propC8v.xml
BR-725.xml                      Makefile.in                 XLR99.xml                   propDA-R352_6-123-F_2.xml
BR710.xml                       MerlinV1650.xml             Zenoah_G-26A.xml            propHO-V373-D.xml
CF6-80C2.xml                    Oberursel-UrII.xml          ah1s_rotor.xml              propHS139v.xml
CFM56.xml                       Olympus593Mrk610.xml        ah1s_tail_rotor.xml         prop_75in2f.xml
CFM56_5.xml                     P51prop.xml                 avco_lycoming_t53.xml       prop_81in2v.xml
CVS                             PT6A-27.xml                 direct.xml                  prop_Clark_Y7570.xml
Dr1_propeller.xml               PT6A-68.xml                 electric147kW.xml           prop_PT6.xml
Estes_E9.xml                    PW125B.xml                  electric_1mw.xml            prop_SSZ.xml
F100-PW-229.xml                 R-1820-97.xml               engIO360C.xml               prop_deHavilland5000.xml
F119-PW-1.xml                   RB211-524.xml               engIO470D.xml               prop_generic2f.xml
GE-CF6-80C2-B1F.xml             RL10.xml                    engIO540AB1A5.xml           s64_rotor.xml
HamiltonStd6243A-3.xml          RL10_nozzle.xml             engRRMerlin61.xml           t56.xml
J33-A-35.xml                    RollsRoyce.xml              engRRMerlinXII.xml          t56_prop.xml
J52.xml                         SRB.xml                     eng_O-200.xml               test_turbine.xml
J69-T25.xml                     SRB_nozzle.xml              eng_PegasusXc.xml           twin_pratt_and_whitney_t73.xml
J79-GE-11A.xml                  SSME.xml                    eng_RRhawk.xml              vrtule2.xml
J85-GE-5.xml                    SSME_nozzle.xml             eng_io320.xml               xlr99_nozzle.xml
JT9D-3.xml                      T76.xml                     engtm601.xml
LM_descent_nozzle.xml           TRENT-900.xml               prop30FP2B.xml
LMdescent.xml                   Tay-620.xml                 propC10v.xml
MacBook-Pro-di-Agostino-De-Marco:engine agodemar$
```

Figura 15.7 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/engine/`.

Si avvierà il download dell'intero albero di cartelle che compongono la distribuzione aggiornata di JSBSim. Nella cartella dalla quale si è dato il comando `cvs` verrà creata la sottocartella `JSBSim` che corrisponderà alla ⟨*JSBSim root*⟩. La figura 15.5 nella pagina precedente mostra la lista dei file presenti nella directory principale di JSBSim.

Con alcune sub-directory di ⟨*JSBSim root*⟩ gli utenti devono acquisire familiarità per poter agevolmente utilizzare i modelli predefiniti di velivolo e lanciare le simulazioni batch. Queste sono: la sottocartella `aircraft/` (figura 15.6), la sottocartella `engine/` (figura 15.7) e la sottocartella `scripts/` (figura 15.8 nella pagina successiva).

Il codice sorgente si troverà nella cartella ⟨*JSBSim root*⟩`/src/` e nelle rispettive sub-directory. Esplorare questa parte di albero è utile solo se si ha intenzione di modificare il programma `JSBSim` e ricompilarlo. Per chi ha esperienza di programmazione in C++ a volte è molto utile guardare il codice sorgente per rendersi conto direttamente delle funzionalità del programma.

Per chi preferisce usare `git` il comando che 'clona' l'archivio online è il seguente:

```
$ git clone git://jsbsim.git.sourceforge.net/gitroot/jsbsim/jsbsim
⟨invio⟩
```

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd scripts/
MacBook-Pro-di-Agostino-De-Marco:scripts agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/scripts
MacBook-Pro-di-Agostino-De-Marco:scripts agodemar$ ls
737_cruise.xml                  Short_S23_3.xml                 c172_cruise_8K.xml
737_cruise_steady_turn.xml      Short_S23_4.xml                 c172_elevation_test.xml
B737_Runway.xml                 StellarJ_Fly.xml                c172_elevator_doublet.xml
B747_script1.xml                T37.xml                         c172_runway_at_rest_cg_shift.xml
CVS                             T38.xml                         c3101.xml
Concorde_rotate_test.xml        WK450.xml                       c3104.xml
Concorde_runway_test.xml        ah1s_flight_test.xml            cannonball.xml
F4N_runway_test.xml             b171.xml                        f16_runway_test.xml
J2461.xml                       b737_runway_new.xml             f16_test.xml
L410.xml                        ball.xml                        kml_output.xml
Makefile                        ball_chute.xml                  mk82_script.xml
Makefile.am                     ball_orbit.xml                  ov10_runway.xml
Makefile.in                     blank.xml                       plotfile.xml
Postfix.kml                     c1721.xml                       sim_primer.xml
Prefix.kml                      c1722.xml                       weather-balloon.xml
Short_S23_1.xml                 c1723.xml                       x151.xml
Short_S23_2.xml                 c1724.xml                       x152.xml
MacBook-Pro-di-Agostino-De-Marco:scripts agodemar$
```

Figura 15.8 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/scripts/`.

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd systems/
MacBook-Pro-di-Agostino-De-Marco:systems agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/systems
MacBook-Pro-di-Agostino-De-Marco:systems agodemar$ ls
Autopilot.xml           Makefile                catapult.xml            refuel.xml
BLC.xml                 Makefile.am             fg_glue.xml             rpm_governor.xml
CVS                     Makefile.in             flaps.xml               speedbrakes.xml
FCS-pitch.xml           NWS.xml                 gear.xml                sperry-a2-autopilot.xml
FCS-roll.xml            afcs.xml                holdback.xml            tail_wheel_lock.xml
FCS-yaw.xml             airship_added_mass.xml  hook.xml
GNCUtilities.xml        alpha_buffet.xml        jsb_glue.xml
MacBook-Pro-di-Agostino-De-Marco:systems agodemar$
```

Figura 15.9 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/systems/`.

Nella cartella di lavoro si otterrà la creazione della directory `jsbsim` che costituirà la ⟨*JSBSim root*⟩ della repository locale.

15.2.5 Compilare i sorgenti

Per compilare JSBSim si deve avere un compilatore C++. La via più semplice è quella basata sul compilatore GCC, dopo essersi assicurati di avere installato i programmi `autoconf`, `automake` e `make`. Queste tre utilità sono normalmente presenti in Linux o in Mac OS X dopo aver installato gli Apple Developer Tools. In Windows questi programmi si installano attraverso il setup di Cygwin (spuntando le applicazioni da installare nella categoria “devel”).

La compilazione di `JSBSim` sul proprio sistema operativo va preparata spostandosi nella ⟨*JSBSim root*⟩ e lanciando il bash script `autogen.sh`. Successivamente basterà dare il comando `make` per compilare i sorgenti ed ottenere l’eseguibile `JSBSim`. Con riferimento all’esempio della figura 15.5 a pagina 9, in cui la cartella ⟨*JSBSim root*⟩ coincide in particolare con `/Users/agodemar/agodemar/jsbsim/JSBSim/`, si potrà dire:

```
$ cd /Users/agodemar/agodemar/jsbsim/JSBSim ⟨invio⟩
$ ./autogen.sh ⟨invio⟩
(seguono numerosa attività di configurazione e di verifica)
$ make ⟨invio⟩
(segue l’output della compilazione di tutti i sorgenti)
```

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd src
MacBook-Pro-di-Agostino-De-Marco:src agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/src
MacBook-Pro-di-Agostino-De-Marco:src agodemar$ ls
CVS                FGJSBBase.cpp      JSBSim.cpp         Makefile.am        math
FGFDMExec.cpp      FGJSBBase.h        JSBSim.minimal.cpp Makefile.in        models
FGFDMExec.h        FGJSBBase.o        JSBSim.o           initialization     simgear
FGFDMExec.o        JSBSim             Makefile           input_output       utilities
MacBook-Pro-di-Agostino-De-Marco:src agodemar$
```

Figura 15.10 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/src/`.

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd aircraft/737/
MacBook-Pro-di-Agostino-De-Marco:737 agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/aircraft/737
MacBook-Pro-di-Agostino-De-Marco:737 agodemar$ ls -l
total 160
-rw-r--r--  1 agodemar  staff  33306 23 Gen  2011 737.xml
drwxr-xr-x  5 agodemar  staff    170  9 Feb 09:43 CVS
-rw-r--r--  1 agodemar  staff    403 14 Nov  2010 INSTALL
-rw-r--r--  1 agodemar  staff   9825  9 Feb 09:44 Makefile
-rw-r--r--  1 agodemar  staff     39 18 Apr  2008 Makefile.am
-rw-r--r--  1 agodemar  staff   8856  9 Feb 09:44 Makefile.in
-rwxr-xr-x  1 agodemar  staff    489  4 Apr  2008 cruise_init.xml
-rwxr-xr-x  1 agodemar  staff    471  4 Apr  2008 cruise_steady_turn_init.xml
-rwxr-xr-x  1 agodemar  staff    548 25 Feb  2010 reset00.xml
MacBook-Pro-di-Agostino-De-Marco:737 agodemar$
```

Figura 15.11 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/aircraft/737/`.

Dopo aver compilato con successo, la situazione dei file nella sottocartella `src/` è data dalla figura 15.10. La lista dei file e delle cartelle si ha con i seguenti comandi:

```
$ cd src ⟨invio⟩ (ci si sposta nella cartella dei sorgenti)
$ pwd ⟨invio⟩ (stampa il percorso della cartella corrente)
$ ls ⟨invio⟩ (stampa la lista di file e cartelle, figura 15.10)
$ cp JSBSim .. ⟨invio⟩ (copia l'eseguibile nella cartella superiore)
$ cd .. ⟨invio⟩ (ci si sposta di nuovo in ⟨JSBSim root⟩)
```

Si noterà la presenza del file eseguibile `JSBSim` (`JSBSim.exe` in Windows/Cygwin). Tipicamente questo file è copiato nella cartella superiore, cioè ⟨*JSBSim root*⟩, da dove è opportuno lanciare tutte le simulazioni.

15.3 Caratteristiche di JSBSim

JSBSim è un software multipiattaforma, sviluppato in linguaggio C++ standard, compilabile sia sui sistemi operativi Windows che sui sistemi Unix/Linux e Mac OS X. Attualmente JSBSim è distribuito con licenza *GNU Library General Public License* (GPL) o *Lesser* GPL (LGPL) che ne consente il riutilizzo in progetti proprietari anche a scopo commerciale.

L'acronimo "JSB" rappresenta le iniziali di Jon S. Berndt, il coordinatore del team di sviluppatori (`jsbsim.sourceforge.net/about.html`), ideatore e progettista della libreria. L'architettura software di JSBSim è stata sviluppata negli anni, a partire dal 1996, secondo il paradigma della programmazione a oggetti. Questo stile di programmazione si presta particolarmente alla logica della simulazione del volo ed è ben supportato dal linguaggio C++.

```
MacBook-Pro-di-Agostino-De-Marco:JSBSim agodemar$ cd aircraft/c172x/
MacBook-Pro-di-Agostino-De-Marco:c172x agodemar$ pwd
/Users/agodemar/agodemar/jsbsim/JSBSim/aircraft/c172x
MacBook-Pro-di-Agostino-De-Marco:c172x agodemar$ ls -l
total 264
drwxr-xr-x  5 agodemar  staff    170  9 Feb 09:43 CVS
-rw-r--r--  1 agodemar  staff    438 14 Nov  2010 INSTALL
-rw-r--r--  1 agodemar  staff   9903  9 Feb 09:44 Makefile
-rw-r--r--  1 agodemar  staff    105 25 Feb  2011 Makefile.am
-rw-r--r--  1 agodemar  staff   8928  9 Feb 09:44 Makefile.in
-rwxr-xr-x  1 agodemar  staff   9465  7 Set 04:37 c172ap.xml
-rw-r--r--  1 agodemar  staff  68358 18 Ago  2011 c172x.xml
-rw-r--r--  1 agodemar  staff    445  8 Mag  2011 elevator_doublet_init.xml
-rw-r--r--  1 agodemar  staff    644 28 Apr  2006 output.xml
-rw-r--r--  1 agodemar  staff    626 18 Ago  2011 reset00.xml
-rw-r--r--  1 agodemar  staff    332  7 Set 04:37 reset01.xml
-rwxr-xr-x  1 agodemar  staff    564 25 Feb  2011 reset_at_rest.xml
MacBook-Pro-di-Agostino-De-Marco:c172x agodemar$
```

Figura 15.12 Lista di file e cartelle nella directory ⟨*JSBSim root*⟩`/aircraft/c172x/`.

Il software JSBSim presenta due aspetti importanti: (*a*) la definizione di diversi modelli di velivolo non richiede la ricompilazione del codice, (*b*) l'input è basato su un linguaggio di *markup* che lo rende semplice da leggere e da interpretare.

15.3.1 Modelli 'data driven'

Un'importante caratteristica di JSBSim risiede nella sua natura di applicazione di tipo *data driven*. Molti codici di calcolo e FDM sviluppati in passato — in un'epoca in cui era prevalente il paradigma di programmazione procedurale — spesso erano ideati per la simulazione di un velivolo specifico. Alcuni di essi sono in uso ancora oggi ma non offrono la possibilità di cambiare le caratteristiche del modello di aeromobile senza una ricompilazione del codice sorgente. Ciò è dovuto al fatto che la logica e i parametri del modello sono "scolpiti" nel codice (*hard coded*) ed un'eventuale cambiamento, piccolo o grande che sia, richiede una modifica dei sorgenti.

Il concetto fondamentale sul quale si basa l'architettura di JSBSim è quello della "esposizione" all'utente delle variabili della simulazione. Sia a tempo di lettura dell'input — cioè di configurazione iniziale del modello — che in qualsiasi istante dell'esecuzione, JSBSim rende visibili e modificabili i valori di numerose variabili interne. Ad esempio, la variabile

`aero/alpha-deg`

rappresenta l'angolo d'attacco del velivolo (α_B) espresso in gradi; il suo valore numerico è conservato in un'area di memoria, `aero/`, che raggruppa grandezze collegate al modello aerodinamico in uso. In generale, i gruppi di variabili in JSBSim sono organizzati come se fossero i percorsi di un file system; ciascun gruppo contiene i valori di un certo numero di variabili così come una directory contiene dei file. Nell'esempio precedente, sarà possibile accedere al valore di `aero/alpha-deg` in qualsiasi momento della simulazione. Analogamente,

`position/h-sl-ft` e `velocities/q-rad_sec`

rappresentano, rispettivamente, l'altitudine $h = -z_E$ in ft e la componente q della velocità angolare in assi velivolo espressa in rad/s.

Attualmente JSBSim espone all'utente più di 200 variabili interne predefinite. Esse sono raggruppate secondo i seguenti percorsi: `accelerations/`, `aero/`, `atmosphere/`, `attitude/`, `fcs/` (*Flight Control System*), `forces/`, `gear/`, `ic/` (*Initial Conditions*), `inertia/`, `metrics/`, `moments/`, `position/`, `propulsion/`, `simulation/`, `systems/`, `velocities/`.

L'insieme delle variabili definite per un dato modello di velivolo in JSBSim viene detto catalogo (*catalog*). Esso è stampabile dal programma `JSBSim` con l'apposita opzione `--catalog`. Ad esempio, la seguente riga di comando:

```
$ JSBSim --aircraft=c172p --catalog > c172p_catalog.txt ⟨invio⟩
```

attraverso l'operatore > di redirezione dell'output crea il file di testo `c172p_catalog.txt` con la lista completa delle variabili della simulazione, valida per il modello di velivolo `c172p` (si veda più avanti il paragrafo 15.4).

Al meccanismo dell'esposizione a run-time delle variabili della simulazione si affianca la possibilità offerta all'utente di poter definire delle variabili personalizzate. Esse saranno utilizzabili nello specifico modello matematico di velivolo insieme alle variabili predefinite.

La logica dell'esposizione delle variabili della simulazione e della possibilità di definire variabili e funzioni personalizzate hanno permesso agli sviluppatori di JSBSim di generalizzare la gestione del modello di aeromobile e della simulazione. Gli utenti hanno così la possibilità di definire *dall'esterno* il modello *completo* di uno specifico velivolo, attraverso un file di configurazione.

Pertanto, con il software JSBSim la simulazione è completamente configurabile attraverso i dati. Quando si vuole aggiungere un nuovo modello di velivolo alla galleria di modelli disponibili, basta predisporre un'apposita cartella con uno o più file nominati opportunamente, senza la necessità di scrivere nuovo codice e senza la necessità di ricompilare la libreria.

15.3.2 JSBSim-ML

JSBSim si aspetta che i dati relativi a un determinato modello di velivolo siano stati raccolti all'interno di in uno o più file di configurazione e scritti in un formato denominato JSBSim-ML (*JSBSim Markup Language*). Il file di configurazione principale (che eventualmente punterà ad altri file di dati) viene caricato solo al tempo di esecuzione.

Dal punto di vista sintattico il formato JSBSim-ML risponde alle specifiche del linguaggio XML (*eXtensible Markup Language*). Anche i file di inizializzazione ed il linguaggio di scripting di JSBSim si basano su XML. In generale, l'uso del formato XML presenta diverse caratteristiche positive. Oltre alla diffusione sempre maggiore, in particolare nelle applicazioni che utilizzano la connettività in rete, e alla reperibilità di *parser* efficienti e disponibili liberamente, una caratteristica concettualmente fondamentale è la possibilità di organizzare i dati in maniera strutturata. Il formato XML aumenta la leggibilità dei dati e riduce lo sforzo di programmazione necessario alla loro gestione, riducendo inoltre i problemi di scambio delle informazioni in specifici campi applicativi.

Dal punto di vista dei contenuti informativi JSBSim-ML è in larga parte compatibile con il formato DAVE-ML (*Dynamic Aerospace Vehicle Exchange Markup Language*), proposto dalla NASA (`daveml.org/`) e reso standard a partire dal 2011 (ANSI/AIAA Flight Dynamics Model Exchange Standard, ANSI/AIAA S-119-2011e, ISBN: 1600867965).

DAVE-ML nasce con l'obiettivo di unificare lo standard di rappresentazione dei dati per i modelli di simulazione in campo aerospaziale.

Per approfondimenti sulle specifiche del linguaggio JSBSim-ML e per i dettagli sull'uso di JSBSim gli utenti dovrebbero consultare in primo luogo il manuale di riferimento della libreria: `jsbsim.sourceforge.net/JSBSimReferenceManual.pdf`. Un'altra fonte importante di informazioni è la pagina web `jsbsim.sourceforge.net/documentation.html` che raccoglie diversi documenti utili sia per gli utilizzatori che per gli sviluppatori.

15.4 Il Flight Dynamics Model di un velivolo

15.4.1 Organizzazione dei file

Per caratterizzare completamente un velivolo ed eseguire simulazioni batch con JSBSim è necessario creare un insieme di file. Essi devono essere posizionati in opportune sottocartelle della directory in cui risiede l'eseguibile `JSBSim`. Detta ⟨*JSBSim root*⟩ tale cartella, si deve predisporre:

- Il file di configurazione del velivolo, che deve essere presente all'interno del percorso ⟨*JSBSim root*⟩`/aircraft/` in un'ulteriore sottocartella che deve avere lo stesso nome dell'aereo. Ad esempio, per il velivolo Tecnam P2006T si potrà creare il file `p2006t.xml` nella cartella ⟨*JSBSim root*⟩`/aircraft/p2006t/`.
- Il file di configurazione del motore, che deve essere presente nel percorso ⟨*JSBSim root*⟩`/engine/`. Ad esempio, il Tecnam P2006T è equipaggiato con due motori Rotax 912S; il file `p2006t.xml` farà riferimento ai dati del motore contenuti nel file `rotax912s.xml` posizionato nella cartella ⟨*JSBSim root*⟩`/engine/`.
- Il file di specifica del propulsore, che deve essere presente anch'esso nella sottocartella ⟨*JSBSim root*⟩`/engine/`. Ad esempio, il Tecnam P2006T monta sui motori una coppia di propulsori MTV-21-A-C-F; il file `p2006t.xml` farà riferimento ai dati del propulsore contenuti nel file `prop_mtv21.xml` presente nella cartella ⟨*JSBSim root*⟩`/engine/`.
- Il file di inizializzazione dello stato della simulazione, che contiene le condizioni iniziali. Questo file può avere un nome qualsiasi e deve essere presente nella stessa cartella in cui si trova il file di configurazione del modello di velivolo. Ad esempio, per il Tecnam P2006T si potrà avere il file `init_in_air_naples.xml` nella cartella ⟨*JSBSim root*⟩`/aircraft/p2006t/`. Nella galleria dei modelli di velivolo distribuiti con JSBSim si potranno trovare esempi di questo tipo di file, spesso nominati come `reset.xml`, `reset00.xml` o `reset01.xml`.
- Eventuali file di definizione dell'autopilota, per quei velivoli che ne sono dotati, che devono essere presenti nella stessa cartella del file di configurazione del velivolo (o anche nella cartella ⟨*JSBSim root*⟩`/systems/`).
- Un eventuale file di scripting. Tipicamente esso potrà risiedere nella sottocartella ⟨*JSBSim root*⟩`/scripts/`.

La figura 15.13 a pagina 17 riporta la rappresentazione visiva di una possibile organizzazione dei file per il modello di velivolo Tecnam P2006T. La stessa cartella ⟨*JSBSim root*⟩`/aircraft/` contiene una galleria di modelli di velivolo distribuiti con JSBSim. In essa si potranno trovare esempi già pronti per l'uso.

Si osservi che l'organizzazione dei file suggerita qui corrisponde a quella che `JSBSim` si aspetta di default. D'altra parte il programma possiede delle opzioni di lancio da riga di comando che permettono di caricare determinati file secondo specifici percorsi. Per conoscere la lista delle opzioni disponibili basta spostarsi nella cartella ⟨*JSBSim root*⟩ e dare il comando:

```
$ ./JSBSim --help ⟨invio⟩
```

Dalla riga di comando precedente si otterrà il seguente output:

```
JSBSim version 1.0 Feb 21 2012 11:15:51

Usage: jsbsim [script file name] [output file names] <options>

options:
  --help  returns this message
  --version  returns the version number
  --outputlogfile=<filename>  sets (overrides) the name of the first data output file
  --logdirectivefile=<filename>  specifies the name of a data logging directives file
                                 (can appear multiple times)
  --root=<path>  specifies the JSBSim root directory (where aircraft/, engine/, etc. reside)
  --aircraft=<filename>  specifies the name of the aircraft to be modeled
  --script=<filename>  specifies a script to run
  --realtime  specifies to run in actual real world time
  --nice  specifies to run at lower CPU usage
  --nohighlight  specifies that console output should be pure text only (no color)
  --suspend  specifies to suspend the simulation after initialization
  --initfile=<filename>  specifies an initilization file
  --catalog specifies that all properties for this aircraft model should be printed
            (catalog=aircraftname is an optional format)
  --property=<name=value> e.g. --property=simulation/integrator/rate/rotational=1
  --simulation-rate=<rate (double)> specifies the sim dT time or frequency
                    If rate specified is less than 1, it is interpreted as
                    a time step size, otherwise it is assumed to be a rate in Hertz.
  --end-time=<time (double)> specifies the sim end time

NOTE: There can be no spaces around the = sign when
      an option is followed by a filename
```

15.4.2 Il file di configurazione principale

Il file di configurazione del modello di velivolo è il più importante tra i file di input. Al suo interno sono presenti più sezioni, ciascuna individuata da un *marcatore* o *tag* XML. Le diverse sezioni definiscono i diversi elementi del modello di aeromobile elaborato da `JSBSim`, che sono: la geometria, la distribuzione delle masse, la disposizione dei carrelli, la propulsione, le caratteristiche delle superfici di controllo, i dati aerodinamici, le direttive di output.

Un esempio di struttura del file di configurazione è il seguente:

```
<fdm_config>    ← il tag radice
  <fileheader>
    ...         ← informazioni su autore, riferimenti, etc.
  </fileheader>
  <metrics>
    ...         ← informazioni sulla geometria
```

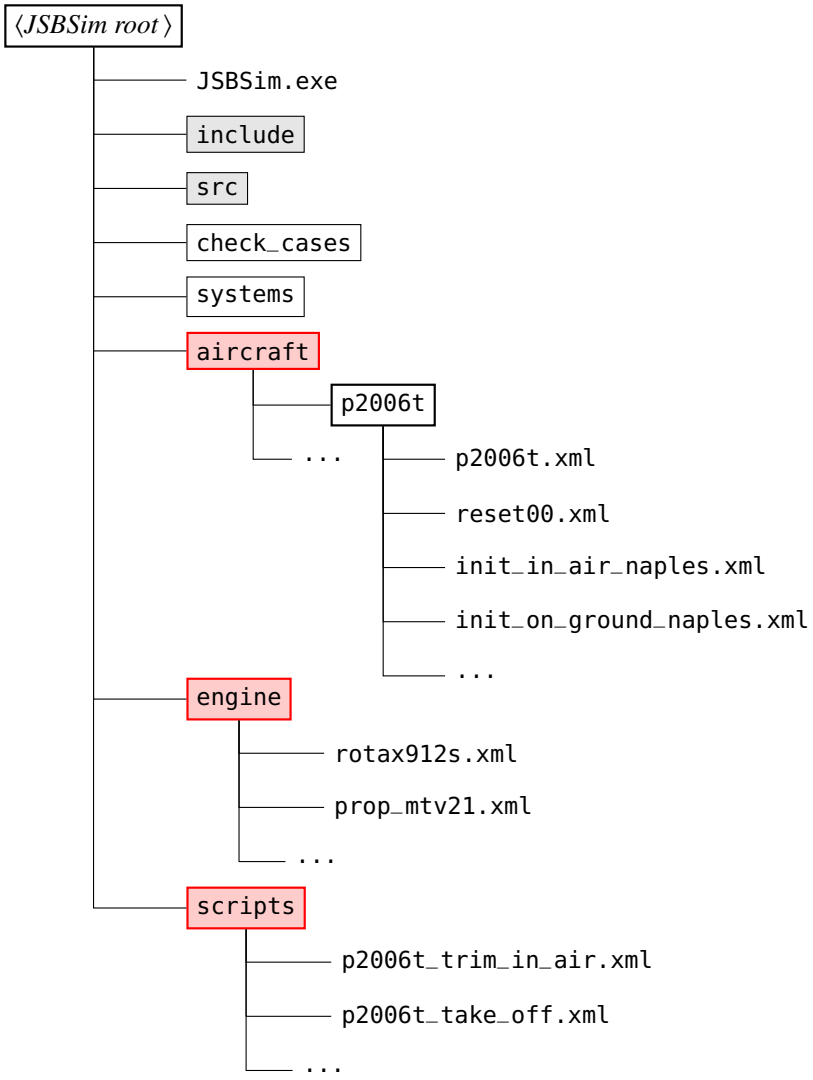


Figura 15.13 Organizzazione delle sottocartelle di ⟨*JSBSim root*⟩.

```
  </metrics>
  <mass_balance>
    ...          ← informazioni sui pesi e le inerzie
  </mass_balance>
  <ground_reactions>
    ...          ←caratteristiche del comportamento al suolo
    ...          ← (carrelli ed altri punti di contatto)
  </ground_reactions>
  <propulsion>
    ...          ← dati sul sistema propulsivo
  </propulsion>
  <flight_control>
    ...          ← definizioni dei controlli di volo (autopilota)
  </flight_control>
  <aerodynamics>
    ...          ←database aerodinamico
  </aerodynamics>
  <output>
    ...          ← direttive di log
  </output>
</fdm_config>    ← chiusura del tag radice
```

Figura 15.14 Un velivolo Cessna 172 Skyhawk (172M) in fase di decollo (en.wikipedia.org/wiki/Cessna_172).

Come si osserva dal listato precedente, ogni sezione è caratterizzata da una coppia di tag. Un tag è un marcatore che ha la forma <⟨*parola-chiave*⟩>, cioè una parola chiave delimitata da parentesi angolari, "<" e ">". Trattandosi di input per JSBSim, i marcatori sono da considerarsi parole chiave di JSBSim-ML. All'inizio della singola sezione è presente un marcatore "di apertura"; alla fine della sezione è presente un analogo marcatore "di chiusura" dalla forma </⟨*parola-chiave*⟩>.

Ad esempio, il tag `<metrics>` rappresenta l'inizio della sezione contenente informazioni sulle dimensioni principali del velivolo. Questo marcatore deve essere accompagnato da un tag di chiusura simile a quello di apertura, con la differenza che nel marcatore che chiude la sezione la parola chiave è preceduta dal carattere "/" (*slash*). Pertanto, il tag `</metrics>` chiude la sezione dei dati metrici.

Altre parti del file di configurazione saranno delimitate in modo analogo. Ad esempio, la sezione con i dati sul sistema propulsivo è delimitata dai marcatori `<propulsion>` e `</propulsion>`, quella con i dati del modello aerodinamico è delimitata dala coppia di tag `<aerodynamics>` e `</aerodynamics>`, e così via.

In generale, data la particolare strutturazione delle informazioni contenute nei file XML, non è importante che il susseguirsi delle diverse sezioni nel file di configurazione di JSBSim corrisponda alla sequenza sopra indicata. Questo è uno dei vantaggi del formato XML e dei linguaggi di markup che su di esso si basano. D'altra parte, è importante che le diverse sezioni riportate nel listato precedente sia *tutte* presenti nel file di input.

Nei paragrafi seguenti daremo qualche dettaglio in più sul formato dei dati, spiegando la sintassi di alcune delle sezioni più importanti del file di configurazione. Per fare degli esempi pratici, considereremo il velivolo Cessna 172 Skyhawk mostrato nella figura 15.14. Il file di input `c172p.xml` che ne fornisce il modello pronto per l'uso si trova nella cartella ⟨*JSBSim root*⟩`/aircraft/c172p/`.

Il tag `<metrics>`

La sezione `metrics` del file di configurazione definisce le grandezze caratteristiche del velivolo e la posizione di alcuni punti chiave. Le diverse informazioni all'interno di questa parte dovranno a loro volta essere marcate da opportune parole chiave di JSBSim-ML.

La sezione `metrics` del modello `c172p` è la seguente:

```
...
<metrics>
```

```
        <wingarea unit="FT2">   174.0 </wingarea>
        <wingspan unit="FT">     35.8 </wingspan>
        <chord unit="FT">         4.9 </chord>
        <htailarea unit="FT2">   21.9 </htailarea>
        <htailarm unit="FT">     15.7 </htailarm>
        <vtailarea unit="FT2">   16.5 </vtailarea>
        <vtailarm unit="FT">     15.7 </vtailarm>
        <location name="AERORP" unit="IN">
            <x> 43.2 </x>
            <y>  0.0 </y>
            <z> 59.4 </z>
        </location>
        <location name="EYEPOINT" unit="IN">
            <x> 37.0 </x>
            <y>  0.0 </y>
            <z> 48.0 </z>
        </location>
        <location name="VRP" unit="IN">
            <x> 42.6 </x>
            <y>  0.0 </y>
            <z> 38.5 </z>
        </location>
    </metrics>
    ...
```

Tutte le parole chiave utilizzate per delimitare le informazioni metriche hanno un significato intuitivo. Ad esempio il tag `<wingarea>` serve evidentemente ad assegnare la superficie di riferimento S del velivolo, che è l'area della forma in pianta dell'ala. Analogamente, `<wingspan>` si riferisce all'apertura alare b e `<chord>` alla corda di riferimento $\bar{c}$. I tag `<htailarm>` e `<vtailarm>` indicano i bracci $|x_{\mathrm{B,ac,H}}|$ e $|z_{\mathrm{B,ac,V}}|$, rispettivamente, del centro aerodinamico del piano orizzontale di coda e del centro aerodinamico del piano verticale di coda.

Un'aspetto molto interessante che mette in luce la potenza del formato XML può essere notato osservando il modo in cui vengono scritti i marcatori di apertura nel frammento di file precedente. Ad esempio, la sottosezione `wingarea` con il valore di S viene aperta con un marcatore `<wingarea ...>` in cui sono presenti altri elementi sintattici dopo il gruppo "`<wingarea`" e prima della seconda parentesi angolare "`>`". All'interno del marcatore viene inserita una seconda parola chiave, `unit`, che aggiunge un'informazione collaterale a quella principale data dal valore numerico di S.

La parola chiave `unit` è un esempio di *attributo del tag*, una forma espressiva prevista dal formato XML. Gli attributi offrono un meccanismo ausiliario di specifica delle informazioni e tipicamente corrispondono a delle stringhe letterali. Il valore di un attributo è destinato ad essere opportunamente utilizzato dal codice che legge il file XML. Nel caso specifico, evidentemente l'opzione `unit="FT2"` indica che l'utente vuole specificare un dato numerico inteso come quantità espressa in ft^2.

I programmatori di JSBSim hanno previsto le eventualità in cui un utente disponga di dati espressi in unità di misura diverse, a volte nel Sistema Internazionale, a volte nel sistema anglosassone; hanno quindi offerto la possibilità di specificare attraverso l'attributo `unit` l'informazione aggiuntiva sull'unità di misura usata. Se l'utilizzatore non usa questa opzione le quantità indicate nei vari tag di JSBSim-ML sono da esprimersi di default in

Tabella 15.1 Conversioni delle unità di misura supportate da JSBSim-ML.

unit=	unità
"M"	m
"FT"	ft
"IN"	in
"M2"	m^2
"FT2"	ft^2
"IN3"	in^3
"M3"	m^3
"FT3"	ft^3
"CC"	cm^3
"LTR"	l

unit=	unità
"KG"	kg
"SLG*FT2"	slug ft^2
"KG*M2"	kg m^2
"RAD"	rad
"DEG"	deg
"N"	N
"LBS"	lb
"N/M"	N/m
"LBS/FT"	lb/ft
"N/M/SEC"	N/(m s)
"LBS/FT/SEC"	lb/(ft s)

unit=	unità
"WATTS"	W
"HP"	hp
"KTS"	kt
"M/S"	m/s
"N*M"	N m
"FT*LBS"	lb ft
"PA"	N/m^2
"ATM"	atm
"PSF"	lb/ft^2
"PSI"	lb/in^2
"INHG"	in HG

unità di misura anglosassoni (ft, slug, lb, eccetera). Nell'esempio precedente, qualora l'utente conosca l'area della forma in pianta in m^2, si potrà usare questo dato specificando `unit="M2"` nel tag di apertura della sottosezione `wingarea`. Un ragionamento analogo vale per tutte le altre quantità specificate nel file di input. Ad esempio, nel tag successivo `<wingspan ...>` si potrà usare `unit="FT"` o `unit="M"`. Nel tag `<chord ...>` potrebbe essere utile usare `unit="FT"`, `unit="M"` o anche `unit="IN"`.

L'attributo `unit` è utile ed è largamente usato in tutte le sezioni del file di input. Ad esempio, laddove si andrà a specificare il valore di un momento di inerzia sarà possibile usare `unit="SLG*FT2"` oppure `unit="KG*M2"`; dove servirà fornire il valore del coefficiente d'attrito volvente delle ruote del carrello si potrà scegliere se usare la forma `unit="LBS/FT/SEC"` o `unit="N/M/SEC"`. Una lista di unità utilizzabili in JSBSim-ML è riportata nella tabella 15.1.

Le sottosezioni rimanenti della sezione `metrics` sono tre istanze diverse di coppie di marcatori `<location ...>...</location>`, che si distinguono per avere valori diversi dell'attributo `name`. Ciascuna di esse indica un punto solidale al velivolo, con coordinate espresse nel riferimento costruttivo $\{O_C, x_C, y_C, z_C\}$. Tali coordinate sono delimitate dalle coppie di marcatori `<x>...</x>`, `<y>...</y>` e `<z>...</z>`.

La posizione con attributo `name="AERORP"` è importante per la corretta interpretazione del modello aerodinamico (si veda il tag `<aerodynamics>`). Il punto in questione è l'*aerodynamic reference point*, cioè il polo P_{ARP}, di coordinate $(x_{C,ARP}, y_{C,ARP}, z_{C,ARP})$, rispetto al quale sono riferiti i momenti riportati nel database aerodinamico. Nel modello di velivolo considerato abbiamo una posizione individuata da

$$\begin{cases} x_{C,ARP} = 43{,}2\,\text{in} = 1{,}10\,\text{m} \\ y_{C,ARP} = 0 \qquad (P_{ARP} \text{ nel piano di simmetria longitudinale}) \\ z_{C,ARP} = 59{,}4\,\text{in} = 1{,}51\,\text{m} \end{cases} \tag{15.1}$$

A questo punto si osservi quanto segue. Per definizione di riferimento costruttivo, l'asse x_C è orientato verso la poppa della fusoliera e l'asse z_C è orientato nel verso piedi-testa del pilota. La lunghezza della fusoliera di un Cessna 172 è pari a circa 8 m. Come mostrato dalla figura 15.15, per questo particolare FDM l'origine O_C si trova in prossimità del pannello principale degli strumenti di bordo. Le espressioni (15.1) indicano che il

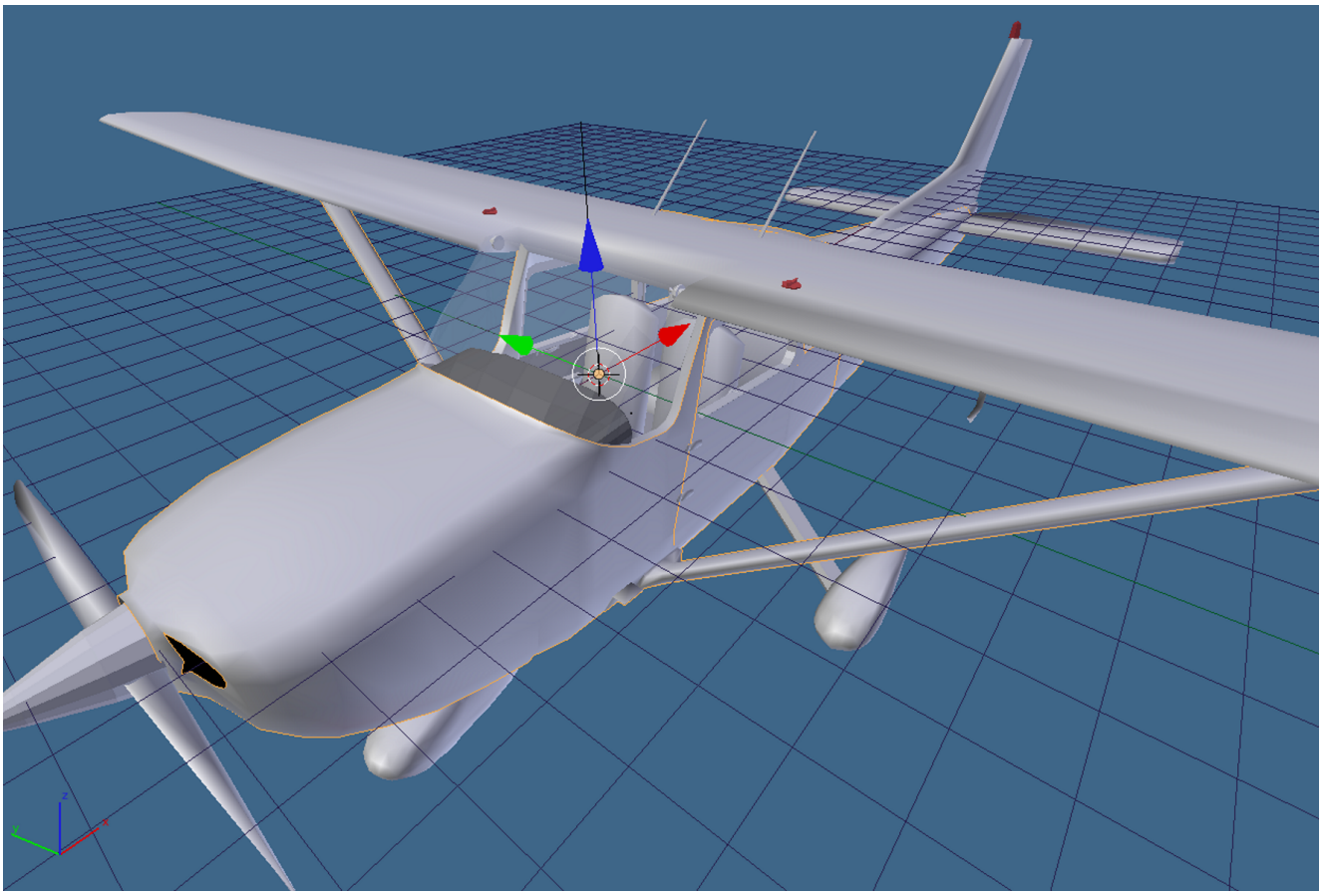


Figura 15.15 Una schermata del software di modellazione 3D Blender (`www.blender.org`). Nella scena è presente un modello di Cessna 172 ed è visualizzato il sistema di riferimento costruttivo $\{O_C, x_C, y_C, z_C\}$. L'origine O_C in questo caso si trova in un punto all'interno della cabina, in prossimità del pannello principale degli strumenti di bordo.

punto P_{ARP} è spostato verso poppa rispetto a O_C di circa 1 m ed è collocato all'altezza della corda di mezzeria dell'ala. Ciò si vede anche dalla figura 15.16 nella pagina successiva. Non sorprenderebbe scoprire che P_{ARP} è posizionato nel centro aerodinamico dell'ala o nel centro aerodinamico della configurazione ala-fusoliera. Come si vedrà più avanti, i dati aerodinamici di questo velivolo provengono da fonti liberamente accessibili che riportano risultati ottenuti in galleria del vento. Tipicamente gli sperimentatori fissano il punto di riduzione delle forze in una posizione convenzionale lungo la corda media aerodinamica corrispondente a uno dei punti prima menzionati.

Nel punto P_{ARP} va concentrata la forza aerodinamica risultante calcolabile istante per istante durante le simulazioni ("a run-time") a partire dai dati aerodinamici, dallo stato del velivolo e dai movimenti dei comandi. Quando questo punto è diverso dal baricentro `JSBSim` calcola i necessari momenti di trasporto per poter inserire nelle equazioni del moto i valori corretti dei momenti baricentrici e propagare correttamente lo stato della simulazione.

I punti associati ai valori `"EYEPOINT"` e `"VRP"` sono importanti nel caso in cui si utilizza la libreria JSBSim come motore di un simulatore grafico come FlightGear. L'*eye point* (EP) corrisponde alla posizione degli occhi del pilota, un punto caratteristico di cui è importante calcolare l'accelerazione rispetto ad un osservatore fisso. Il cosiddetto *visual reference point* (VRP) è utile a collocare il modello tridimensionale del velivolo all'interno dello scenario utilizzato dall'ambiente grafico. Se la simulazione avviene in

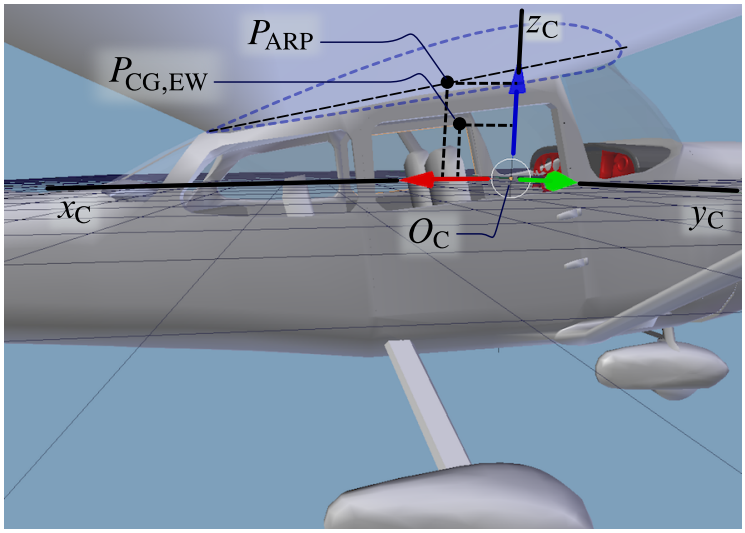


Figura 15.16 Nel sistema di assi costruttivi sono rappresentati i punti P_{ARP} e $P_{CG,EW}$, polo dei momenti aerodinamici e baricentro del velivolo al suo Empty Weight. Inoltre, vengono evidenziati il profilo e la corda (virtuali) della sezione alare di mezzeria.

alta quota, il ruolo del VRP non è evidente. Al contrario, nelle simulazioni in cui il volo avviene in prossimità del suolo, se il VRP non è definito correttamente il velivolo potrebbe apparire in una posizione errata. Ad esempio, in una fase di rullaggio la scena potrebbe mostrare il modello tridimensionale dell'aeromobile irrealisticamente sospeso a mezz'aria o addirittura immerso nel terreno anziché con le ruote poggiate sulla pista. Un punto consigliato in cui collocare il VRP è l'estremità anteriore della fusoliera.

Il tag `<mass_balance>`

Nella sezione `mass_balance` deve essere specificato il quadro completo delle masse del velivolo. In particolare, si devono definire: il peso a vuoto (Empty Weight), i momenti ed i prodotti di inerzia, le coordinate del baricentro, i pesi aggiuntivi (ad esempio, dei singoli piloti, passeggeri e bagagli) e le posizioni dei rispettivi baricentri.

Il frammento di input riportato qui di seguito si riferisce al modello `c172p`:

```
...
  <mass_balance>
      <ixx unit="SLUG*FT2"> 948 </ixx>
      <iyy unit="SLUG*FT2"> 1346 </iyy>
      <izz unit="SLUG*FT2"> 1967 </izz>
      <ixy unit="SLUG*FT2"> -0 </ixy>
      <ixz unit="SLUG*FT2"> -0 </ixz> ⟵ approssimazione
      <iyz unit="SLUG*FT2"> -0 </iyz>
      <emptywt unit="LBS"> 1500 </emptywt>
      <location name="CG" unit="IN">
          <x> 41 </x>
          <y> 0 </y>
          <z> 36.5 </z>
      </location>
      <pointmass name="Pilot">
          <weight unit="LBS"> 180 </weight>
          <location name="POINTMASS" unit="IN">
              <x> 36 </x>
              <y> -14 </y>
              <z> 24 </z>
          </location>
      </pointmass>
```

```
        <pointmass name="Co-Pilot">
            <weight unit="LBS"> 0 </weight>
            <location name="POINTMASS" unit="IN">
                <x> 36 </x>
                <y> 14 </y>
                <z> 24 </z>
            </location>
        </pointmass>
        <pointmass name="Left Passenger">
            <weight unit="LBS"> 0 </weight>
            <location name="POINTMASS" unit="IN">
                <x> 70 </x>
                <y> -14 </y>
                <z> 24 </z>
            </location>
        </pointmass>
        <pointmass name="Right Passenger">
            <weight unit="LBS"> 0 </weight>
            <location name="POINTMASS" unit="IN">
                <x> 70 </x>
                <y> 14 </y>
                <z> 24 </z>
            </location>
        </pointmass>
        <pointmass name="Baggage">
            <weight unit="LBS"> 0 </weight>
            <location name="POINTMASS" unit="IN">
                <x> 95 </x>
                <y>  0 </y>
                <z> 24 </z>
            </location>
        </pointmass>
    </mass_balance>
```

L'espressività del formato XML permette una semplice interpretazione in prima lettura del listato precedente. Si riconoscono i valori dei momenti e dei prodotti d'inerzia del velivolo rispetto agli assi costruttivi, contraddistinti dai marcatori `<ixx>`, `<iyy>`, ..., `<iyz>`. Con il marcatore `<emptywt>` viene indicato il peso a vuoto. Nella condizione di peso a vuoto il velivolo ha un baricentro nella posizione $P_{\mathrm{CG,EW}}$, di coordinate $\left(x_{\mathrm{C,CG,EW}}, y_{\mathrm{C,CG,EW}}, z_{\mathrm{C,CG,EW}}\right)$. Esse sono assegnate dal marcatore `<location>` con attributo `name="CG"`. Per il modello `c172p` le coordinate in assi costruttivi sono

$$\begin{cases} x_{\mathrm{C,CG,EW}} = 41{,}0\,\mathrm{in} = 1{,}04\,\mathrm{m} \\ y_{\mathrm{C,CG,EW}} = 0 \qquad (P_{\mathrm{CG,EW}} \text{ nel piano di simmetria longitudinale}) \\ z_{\mathrm{C,CG,EW}} = 36{,}5\,\mathrm{in} = 0{,}93\,\mathrm{m} \end{cases} \tag{15.2}$$

Con successivi marcatori `<pointmass>` vengono specificate le masse aggiuntive (trattate come pesi quando il valore è una quantità in lb), ciascuna identificata da un diverso valore dell'attributo `name`. Nell'esempio considerato sono indicati il peso e la posizione del pilota utilizzando i marcatori `<weight>` e `<location>`.

Oltre alla massa del pilota, sono assegnate anche le masse del co-pilota, dei due passeggeri destro e sinistro e del bagaglio. Queste masse aggiuntive sono fittiziamente

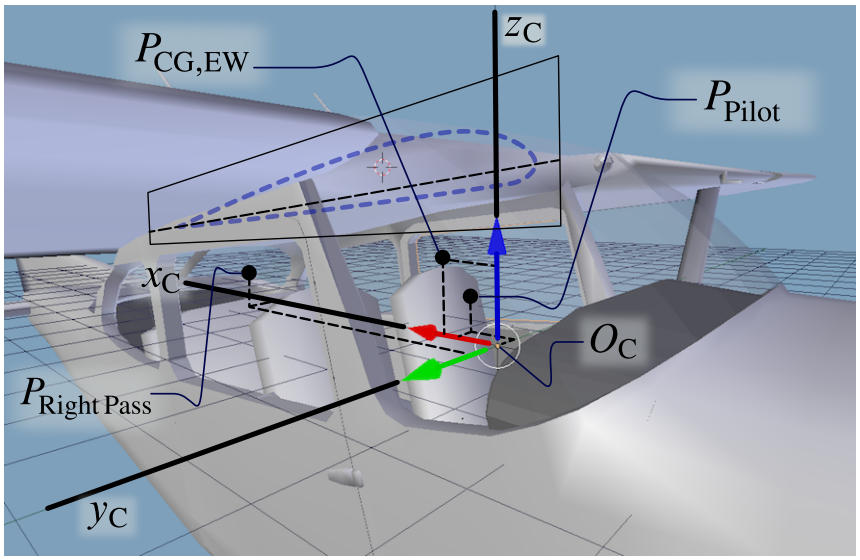


Figura 15.17 Nel sistema di assi costruttivi, oltre al punto $P_{\text{CG,EW}}$, sono rappresentati a titolo d'esempio due punti, P_{Pilot} e $P_{\text{Right Pass}}$, in cui sono concentrate due masse aggiuntive, quella del pilota e quella del passeggero destro.

poste uguali a zero lasciando all'utilizzatore il compito di inserire eventuali valori non nulli. La figura 15.17 mostra a titolo d'esempio due punti, P_{Pilot} e $P_{\text{Right Pass}}$, in cui sono da concentrarsi le masse del pilota e del passeggero destro.

La gestione del FDM da parte di JSBSim prevede la lettura dei dati inerziali e di massa e successivamente il calcolo automatico della posizione del baricentro G del velivolo nella sua condizione di peso assegnata. Anche i momenti e i prodotti d'inerzia saranno calcolati rispetto a questa condizione di peso e riferiti alla terna di assi velivolo standard $\{G, x_{\text{B}}, y_{\text{B}}, z_{\text{B}}\}$.

Il tag <ground_reactions>

Nella sezione `ground_reactions` vengono assegnati i punti in cui JSBSim deve calcolare, eventualmente, le forze di reazione nei confronti del veicolo per effetto del contatto con il suolo. Questi punti sono detti in gergo "punti di contatto". Il numero di punti di contatto può variare a seconda del modello di velivolo. In ogni caso, per avere simulazioni realistiche conviene sempre definirne almeno tre in modo che siano non allineati e opportunamente disposti rispetto al baricentro. Questo permette di simulare correttamente le situazioni in cui il velivolo è fermo sulla pista.

In particolare, un punto di contatto può coincidere con la parte inferiore della ruota di un carrello. Si può anche designare un particolare punto della struttura del velivolo — ad esempio le estremità alari o l'estremità posteriore della fusoliera — come punto di contatto per il quale si vuole monitorare la distanza dal suolo.

Un carrello, che è un dispositivo appositamente progettato per assolvere a certe funzioni, è modellato in JSBSim come una ruota collegata ad un ammortizzatore. Questa caratteristica rende possibile la simulazione realistica di fasi del moto come il decollo o l'atterraggio. In quanto punto di contatto, la definizione di un carrello prevede nello specifico l'assegnazione di: (a) un coefficiente `static_friction` di attrito statico fra ruota e suolo (adimensionale), (b) un coefficiente `dynamic_friction` di attrito dinamico (adimensionale), (c) un coefficiente `rolling_friction` di attrito volvente (adimensionale), (d) un fattore di rigidezza `spring_coeff`, quantità dimensionale espressa in N/m o in lb/ft, (e) un fattore di smorzamento viscoso `damping_coeff`, quantità dimensionale espressa in N/(m s) o in lb/(ft s).

La sezione `ground_reactions` del modello `c172p` si presenta come segue:

```
...
  <ground_reactions>
      <contact type="BOGEY" name="NOSE">
          <location unit="IN">
              <x> -6.8 </x>
              <y> 0 </y>
              <z> -19.5</z>
          </location>
          <static_friction> 0.8 </static_friction>
          <dynamic_friction> 0.5 </dynamic_friction>
          <rolling_friction> 0.02 </rolling_friction>
          <spring_coeff unit="LBS/FT"> 1800 </spring_coeff>
          <damping_coeff unit="LBS/FT/SEC"> 600 </damping_coeff>
          <max_steer unit="DEG"> 10 </max_steer>
          <brake_group> NONE </brake_group>
          <retractable>0</retractable>
      </contact>
      <contact type="BOGEY" name="LEFT_MAIN">
          <location unit="IN">
              <x> 58.2 </x>
              <y> -43 </y>
              <z> -15.5 </z>
          </location>
          <static_friction> 0.8 </static_friction>
          <dynamic_friction> 0.5 </dynamic_friction>
          <rolling_friction> 0.02 </rolling_friction>
          <spring_coeff unit="LBS/FT"> 5400 </spring_coeff>
          <damping_coeff unit="LBS/FT/SEC"> 1600 </damping_coeff>
          <max_steer unit="DEG"> 0.0 </max_steer>
          <brake_group> LEFT </brake_group>
          <retractable>0</retractable>
      </contact>
      <contact type="BOGEY" name="RIGHT_MAIN">
          <location unit="IN">
              <x> 58.2 </x>
              <y> 43 </y>
              <z> -15.5 </z>
          </location>
          <static_friction> 0.8 </static_friction>
          <dynamic_friction> 0.5 </dynamic_friction>
          <rolling_friction> 0.02 </rolling_friction>
          <spring_coeff unit="LBS/FT"> 5400 </spring_coeff>
          <damping_coeff unit="LBS/FT/SEC"> 1600 </damping_coeff>
          <max_steer unit="DEG"> 0.0 </max_steer>
          <brake_group> RIGHT </brake_group>
          <retractable>0</retractable>
      </contact>
      <contact type="BOGEY" name="TAIL_SKID">
          <location unit="IN">
              <x> 188 </x>
              <y> 0 </y>
              <z> 8 </z>
          </location>
```

```
            <static_friction> 0.2 </static_friction>
            <dynamic_friction> 0.2 </dynamic_friction>
            <rolling_friction> 0.2 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 20000 </spring_coeff>
            <damping_coeff unit="LBS/FT/SEC"> 1000 </damping_coeff>
            <max_steer unit="DEG"> 0.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
        <contact type="BOGEY" name="LEFT_TIP">
            <location unit="IN">
                <x> 43.2 </x>
                <y> -214.8 </y>
                <z> 59.4 </z>
            </location>
            <static_friction> 0.2 </static_friction>
            <dynamic_friction> 0.2 </dynamic_friction>
            <rolling_friction> 0.2 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 10000 </spring_coeff>
            <damping_coeff unit="LBS/FT/SEC"> 2000 </damping_coeff>
            <max_steer unit="DEG"> 0.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
        <contact type="BOGEY" name="RIGHT_TIP">
            <location unit="IN">
                <x> 43.2 </x>
                <y> 214.8 </y>
                <z> 59.4 </z>
            </location>
            <static_friction> 0.2 </static_friction>
            <dynamic_friction> 0.2 </dynamic_friction>
            <rolling_friction> 0.2 </rolling_friction>
            <spring_coeff unit="LBS/FT"> 10000 </spring_coeff>
            <damping_coeff unit="LBS/FT/SEC"> 2000 </damping_coeff>
            <max_steer unit="DEG"> 0.0 </max_steer>
            <brake_group> NONE </brake_group>
            <retractable>0</retractable>
        </contact>
    </ground_reactions>
```

La sezione `ground_reactions` raggruppa un certo numero di marcatori `<contact>`, ciascuno dei quali identifica un particolare punto di contatto del velivolo. La posizione deI punti di contatto nel sistema di assi costruttivi è assegnata, al solito, attraverso un marcatore `<location>`.

Nel listato precedente ogni marcatore `<contact>` ha attributo `type="BOGEY"` e ciò indica a JSBSim di effettuare il calcolo delle reazioni modellando il contatto come una ruota-ammortizzatore. I marcatori si distinguono per il loro attributo `name`. Si riconosce il contatto `"NOSE"` associato al ruotino di prua e quelli associati alle due ruote del carrello principale, `"LEFT_MAIN"` e `"RIGHT_MAIN"`. Il quadro completo dei punti di contatto definiti per il modello `c172p` è mostrato visivamente nella figura 15.18.

A tempo di simulazione — in cui l'aeromobile può trovarsi in numerose condizioni di volo possibili, in particolare in prossimità del suolo — viene interrogata la distanza

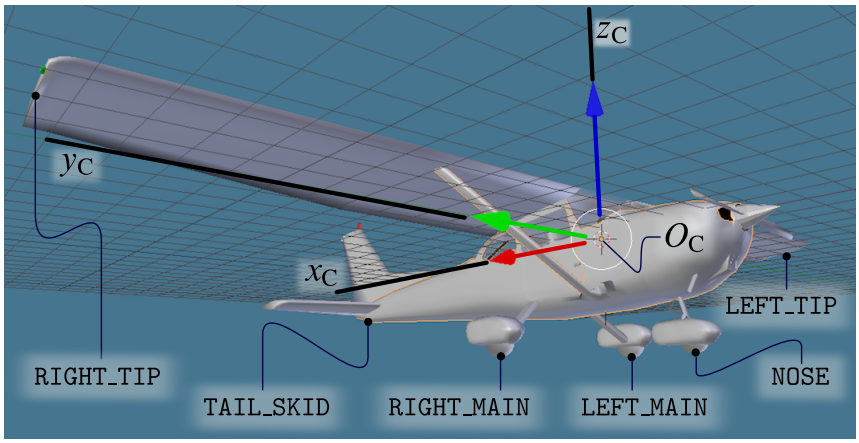


Figura 15.18 Le posizioni dei punti di contatto con il suolo definiti nella sezione `ground_reactions`.

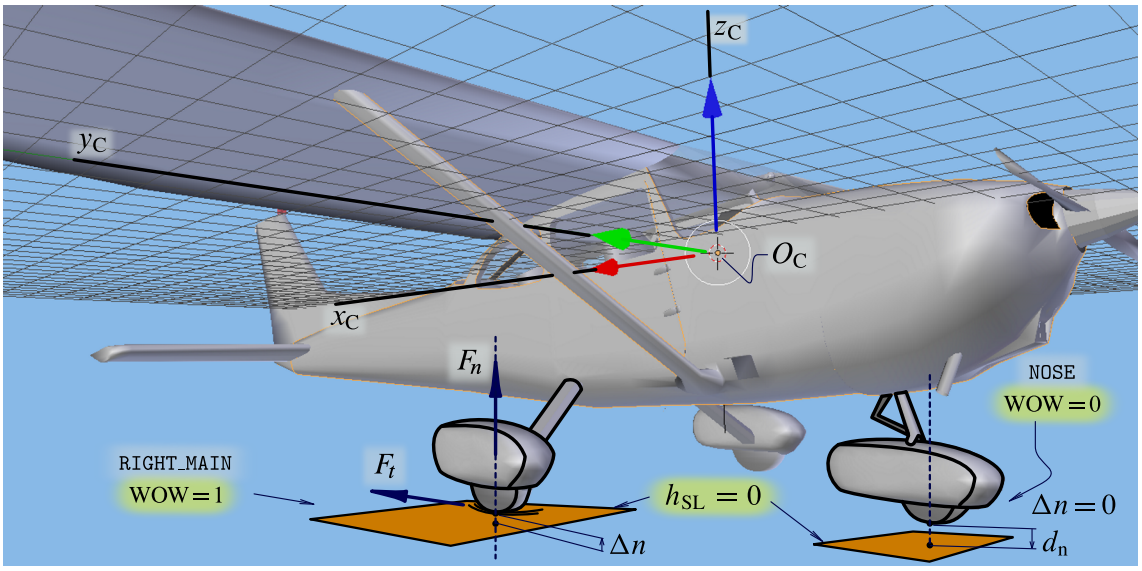


Figura 15.19 Situazione in cui WOW $= 1$ per la ruota destra del carrello principale e WOW $= 0$ per la ruota anteriore.

di ciascun punto di contatto dal livello di quota zero (tipicamente, in mancanza di una definizione del terreno, è una quota *above sea level* anziché *above ground level*). Un valore sufficientemente piccolo di questa distanza innesca per ogni contatto la riassegnazione di una variabile booleana interna di JSBSim chiamata WOW (*Weight On Wheels*). Questo flag vale 0 se il punto è in aria mentre vale 1 se il punto è sul suolo. Negli istanti in cui WOW è vera vengono calcolate le forze di reazione del suolo rispetto al velivolo ed i rispettivi momenti baricentrici. Il calcolo di queste azioni esterne è possibile grazie all'input contenuto nelle sottosezioni `contact`.

La figura 15.19 mostra una situazione in cui la ruota di prua si trova ad una distanza d_n non nulla dal suolo. Per $d_n \geq \varepsilon$, dove ε è un opportuno valore di soglia, si ha WOW $= 0$. Non appena d_n diventa più piccolo di ε JSBSim imposta WOW $= 1$ e predispone il calcolo della compressione dell'ammortizzatore. Quest'ultima situazione compete al punto di contatto `RIGHT_MAIN` rappresentato nella stessa figura 15.19. La quantità Δn è una distanza lineare in senso ortogonale al suolo che indica il movimento del punto di contatto `RIGHT_MAIN` rispetto alla struttura del velivolo per effetto della deformazione della ruota destra del carrello principale. Nel disegno sono riportate anche la forza normale F_n e tangenziale F_t esercitate dal suolo sulla ruota all'istante considerato.

Il FDM calcola, istante per istante, le grandezze Δn, $\mathrm{d}\Delta n/\,\mathrm{d}t$ ed

$$F_\mathrm{n} = k\ \Delta n + b\ \frac{\mathrm{d}\Delta n}{\mathrm{d}t} \tag{15.3}$$

con k assegnato in `spring_coeff` e b assegnato in `damping_coeff`. La quantità F_n è una componente di forza in assi Terra e viene trasformata per ottenerne le componenti di forza e di momento baricentrico corrispondenti in assi velivolo.

Analogamente, la forza tangenziale

$$F_\mathrm{t} = \mu\ F_\mathrm{n} \tag{15.4}$$

viene calcolata usando un opportuno valore del coefficiente d'attrito μ a seconda dello stato della ruota: ferma, strisciante o rotolante. I valori utilizzati dal FDM sono quelli assegnati in `static_friction`, `dynamic_friction` e `rolling_friction`. Anche F_t viene trasformata opportunamente per ottenerne le componenti di forza e di momento baricentrico in assi velivolo.

Le (15.3)-(15.4) sono relazioni semplificate che illustrano i concetti di base per il modello delle azioni esterne dovute al contatto con il suolo. Per approfondimenti sulla procedura completa di calcolo delle reazioni del terreno è possibile fare riferimento direttamente al codice di JSBSim o alla parte del manuale d'uso destinata agli sviluppatori.

Ai punti di contatto di tipo `"BOGEY"` può essere associato il movimento intorno ad un asse per modellare i carrelli girevoli. Il marcatore da usare in questi casi è `<max_steer>`, per indicare l'escursione massima della rotazione direzionale. Di quelli definiti sopra l'unico ad avere un valore non nullo di questa impostazione è il carrello di prua.

Per i punti di contatto di tipo `"BOGEY"` si può impostare un ulteriore flag tramite il marcatore `<retractable>` per indicare se si tratta di un carrello retraibile o fisso. Per i carrelli retraibili l'implementazione del FDM prevede l'assegnazione ad una apposita variabile booleana di un valore vero o falso a seconda se lo stato del carrello è esteso o retratto. Evidentemente, quando il carrello è retratto il calcolo delle azioni esterne non contempla gli effetti degli ammortizzatori e delle ruote.

Va osservato che nell'esempio precedente i punti di contatto relativi alle estremità alari, `"LEFT_TIP"` e `"RIGHT_TIP"`, e all'estremità posteriore della fusoliera, `"TAIL_SKID"`, hanno anch'essi attributo `type="BOGEY"`. Essi sono definiti come se fossero carrelli ma con valori inusitati della costante di rigidezza. In questi casi JSBSim offre la possibilità di un'impostazione alternativa per questo tipo di punti di contatto che possono essere configurati assegnando `type="STRUCTURE"`.

il tag `<external_reactions>`

Il marcatore `<external_reactions>` è molto simile a `<ground_reactions>` e può essere utilizzato per incorporare nella simulazione gli effetti di forze applicate in punti arbitrari della struttura del velivolo. Questa sezione è utile quando si vuole modellare l'azione di traino tramite un cavo, l'azione di un paracadute di frenaggio oppure il lancio in volo con una catapulta.

Per i dettagli sulle sottosezioni di `<external_reactions>` previste da JSBSim-ML si veda il manuale d'uso di JSBSim.

il tag <propulsion>

La sezione relativa al tag <propulsion> si riferisce allo specifico tipo di motore utilizzato. Vengono definiti il numero e la collocazione dei motori e viene indicato il tipo di propulsore che equipaggia il singolo motore trasformandone la potenza disponibile in potenza utile. Con il marcatore <propulsion> si definiscono anche i serbatoi di combustibile ed il loro livello di riempimento. Le informazioni di dettaglio riguardanti il particolare motore ed i propulsori sono solitamente conservate a parte in un file diverso da quello di configurazione principale, che si trova nella cartella ⟨*JSBSim root*⟩/engines/.

Ecco come si presenta la sezione propulsion per il modello c172p:

```
...
<propulsion>
  <engine file="eng_io320"> ⟵ ⟨JSBSim root⟩/engines/eng_io320.xml
    <location unit="IN">
      <x> -19.7 </x>
      <y> 0 </y>
      <z> 26.6 </z>
    </location>
    <orient unit="DEG">
      <roll> 0.0 </roll>
      <pitch> 0 </pitch>
      <yaw> 0 </yaw>
    </orient>
    <feed>0</feed>⟵ alimentazione dal serbatoio 0
    <feed>1</feed>⟵ alimentazione dal serbatoio 1
    <thruster
        file="prop_75in2f"> ⟵ ⟨JSBSim root⟩/engines/prop_75in2f.xml
      <location unit="IN"> ⟵ punto di applicazione della spinta
        <x> -37.7 </x> ⟵ X_C,T
        <y>   0   </y> ⟵ Y_C,T
        <z>  26.6 </z> ⟵ Z_C,T
      </location>
      <orient unit="DEG"> ⟵ orientamento della spinta
        <roll > 0.0 </roll >
        <pitch> 0.0 </pitch> ⟵ μ_T
        <yaw  > 0.0 </yaw  > ⟵ ξ_T
      </orient>
      <sense> 1 </sense> ⟵ rotazione destrorsa
      <p_factor> 5 </p_factor> ⟵disassamento della spinta
    </thruster>                          (P-factor effect)
  </engine>
  <tank type="FUEL">    <!-- Tank number 0 -->
    <location unit="IN">
      <x> 56 </x>
      <y> -112 </y>
      <z> 59.4 </z>
    </location>
    <capacity unit="LBS"> 185 </capacity>
    <contents unit="LBS"> 100 </contents>
  </tank>
  <tank type="FUEL">    <!-- Tank number 1 -->
    <location unit="IN">
      <x> 56 </x>
```

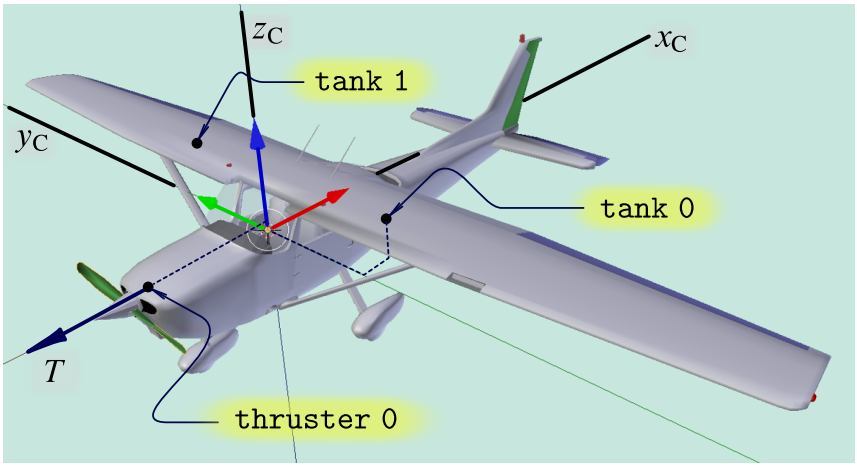


Figura 15.20 Le posizioni dell'entità `thruster` e degli elementi `tank` nel modello `c172p`.

```
      <y> 112 </y>
      <z> 59.4 </z>
    </location>
    <capacity unit="LBS"> 185 </capacity>
    <contents unit="LBS"> 100 </contents>
  </tank>
</propulsion>
...
```

Dal frammento di input su riportato si riconosce la gerarchia di sottosezioni del tag `<propulsion>`. Ciascuna sottosezione `engine` e `tank` definiscono, rispettivamente, il singolo motore e il singolo serbatoio. Il numero di istanze di queste sottosezioni corrisponde al numero di motori e serbatoi presenti. Nel caso di oggetti multipli dello stesso tipo JSBSim associa ad essi un indice che inizia da 0. Ad esempio, per un quadrimotore i motori saranno numerati da 0 a 3; per un modello con due serbatoi essi saranno identificati come serbatoio 0 e serbatoio 1.

Per ciascun `engine` vanno poi specificate le sottosezioni `location`, `orient`, `feed` e `thruster`. Per ciascun `tank` vanno specificate le sottosezioni `location`, `capacity`, e `contents`.

Un tag importante è `<thruster>` con il quale si definiscono il punto di applicazione della spinta $(X_{C,T}, Y_{C,T}, Z_{C,T})$ ed il suo orientamento (ξ_T, μ_T) rispetto agli assi costruttivi.

Per visualizzare le posizioni del propulsore e dei centri di massa iniziali dei due serbatoi si faccia riferimento alla figura 15.20.

Sono diversi i tipi di moto-propulsione modellabili in JSBSim. Quello preso in considerazione in questo contesto è un motore a pistoni accoppiato ad un'elica. Nella distribuzione ufficiale di JSBSim e dalle applicazioni riportate dagli utenti si hanno esempi di propulsione elettrica, a getto e a razzo. Un esempio di velivolo bimotore a elica è dato dal file ⟨*JSBSim root*⟩`/aircraft/c310/c310.xml`. Un buon esempio di modello di velivolo con motore turbofan è dato dal file ⟨*JSBSim root*⟩`/aircraft/737/737.xml`. Il file ⟨*JSBSim root*⟩`/engine/TRENT-900.xml` contiene i dati del motore Rolls-Roice Trent 900 che equipaggia gli Airbus A380.

Il tag `<flight_control>`

Il tag `<flight_control>` consente di definire il sistema di comando del velivolo (*Flight Control System*, FCS). Al suo interno è infatti possibile assegnare le caratteristiche dei

comandi (volantino e pedaliera) e delle superfici aerodinamiche di governo, definire gli eventuali sottosistemi di servoassistenza e specificare le leggi di controllo implementate da un autopilota (se presente). Tipicamente, il comando della manetta del singolo motore è automaticamente definito a partire dalle caratteristiche del tag `<propulsion>`.

Ciascun dispositivo di comando del velivolo, primario o secondario, è associato a un "canale", corrispondente ad un'instanza del tag `<channel>` all'interno della sezione `flight_control`. Ad esempio, esisteranno i canali associati al controllo, rispettivamente, del moto di rollio, di beccheggio ed d'imbardata. Inoltre, per quanto riguarda i comandi secondari, potrà esistere il canale di comando dei flap, degli spoiler, o anche il canale per la movimentazione dei carrelli quando questi sono retraibili. In ogni caso, ciascun canale avrà concettualmente, da un lato, l'input del pilota — cioè il "comando", assegnato a run-time dall'utilizzatore della simulazione —, dall'altro, il movimento di un determinato dispositivo — ad esempio, la deflessione dell'elevatore o dei flap oppure l'allungamento dei carrelli.

Nel caso del `c172p` il marcatore `<flight_control>` racchiude la seguente struttura:

```
  ...
<flight_control name="FCS: c172">
    <channel name="Roll"> ← canale di controllo del rollio
        ...
    </channel>
    <channel name="Pitch"> ← canale di controllo del beccheggio
        ...
    </channel>
    <channel name="Yaw"> ← canale di controllo dell'imbardata
        ...
    </channel>
    <channel name="Flaps"> ← canale di controllo dell'ipersostentazione
        ...
    </channel>
</flight_control>
...
```

La definizione dei comandi primari passa attraverso un'adeguata mappatura delle deflessioni delle parti mobili degli impennaggi. Le escursioni angolari massima e minima di una data superficie di governo devono essere associate alle due posizioni estreme del relativo comando azionato dal pilota. Le variazioni posizionali dei comandi presenti in cabina sono inoltre "normalizzate" rispetto alle loro posizioni estreme. Ad esempio, per il canale del beccheggio JSBSim gestisce le posizioni normalizzate del comando dell'equilibratore con una variabile il cui valore appartiene all'intervallo $[-1, 1]$; i valori estremi sono corrispondenti alle deflessioni, rispettivamente, $\delta_{e,\min}$ e $\delta_{e,\max}$.

Il canale di comando del beccheggio del `c172p` è definito come segue:

```
<flight_control name="FCS: c172">
...
<channel name="Pitch"> ← canale di controllo del beccheggio
    <summer name="Pitch Trim Sum"> ← combina elevatore e aletta tab
        <input>fcs/elevator-cmd-norm</input> ← comando normalizzato
        <input>fcs/pitch-trim-cmd-norm</input> ← comando normalizzato
        <clipto>  ← somma limitata all'intervallo [−1, 1]
            <min>-1</min>
            <max>1</max>
```

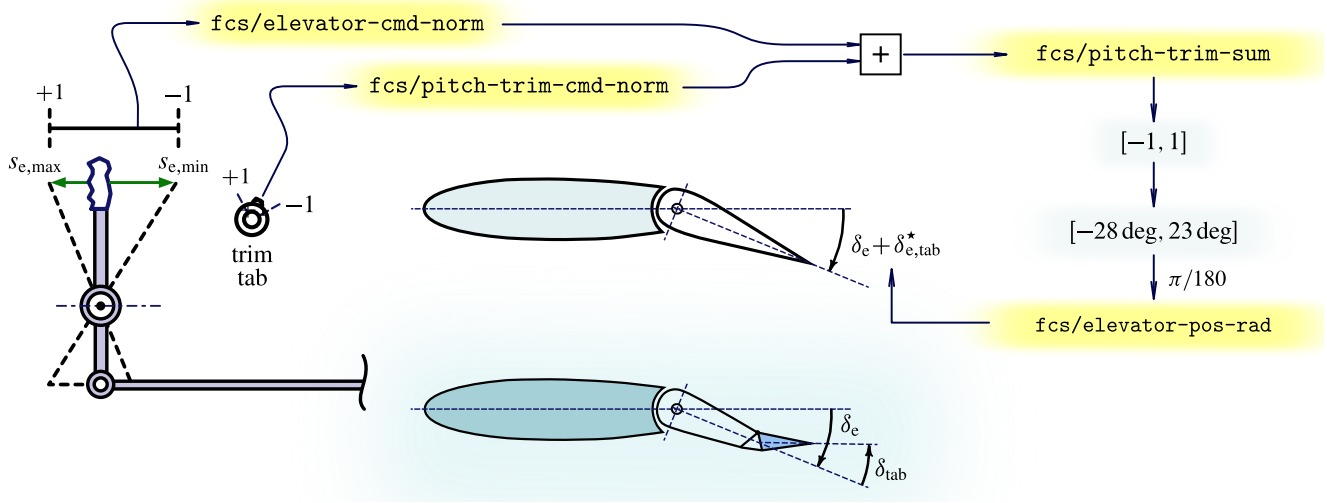


Figura 15.21 La logica di gestione dell'input del pilota per il canale di controllo del beccheggio. L'azione combinata sulla barra e sul trim tab è mappata nell'intervallo $[-1, 1]$. L'output del canale è una variabile reale `fcs/elevator-pos-rad` corrispondente ad una deflessione equivalente dell'equilibratore $\delta_e^\star = \delta_e + \delta_{e,tab}^\star$, con valore appartenente all'intervallo $[\delta_{e,min}, \delta_{e,max}]$. L'impennaggio è rappresentato in condizione di deflessione equivalente $\delta_e^\star$ (in alto) che modella le deflessioni effettive δ_e e δ_{tab} (in basso).

```
        </clipto>
    </summer> ← definisce la variabile fcs/pitch-trim-sum
    <aerosurface_scale name="Elevator Control"> ← conversione deg→rad
        <input>fcs/pitch-trim-sum</input>
        <gain>0.01745</gain> ← fattore di proporzionalità, π/180 = 1/57,3
        <range>
            <min>-28</min> ← δe + δ⋆e,tab minima, in deg
            <max>23</max> ← δe + δ⋆e,tab massima, in deg
        </range>
        <output>fcs/elevator-pos-rad</output>
    </aerosurface_scale>
    <aerosurface_scale name="Elevator Position Normalized">
        <input>fcs/elevator-pos-deg</input>
        <domain> ← deflessioni minima e massima in deg
            <min>-28</min>
            <max>23</max>
        </domain>
        <range> ← deflessioni minima e massima normalizzate
            <min>-1</min>
            <max>1</max>
        </range>
        <output>
            fcs/elevator-pos-norm ← deflessione normalizzata
        </output>
    </aerosurface_scale>
</channel>
...
</flight_control>
```

Il frammento di input precedente fornisce anche un esempio di gestione della deflessione di un'aletta tab di compensazione. Per un equilibratore dotato di aletta tab il FDM di JSBSim definisce una deflessione totale pari alla somma della deflessione geometrica reale δ_e e di una deflessione equivalente $\delta_{e,tab}^\star$ dovuta alla deflessione δ_{tab}. In tal caso sarà

la somma $\delta_e^\star = \delta_e + \delta_{e,tab}^\star$ ad essere prima limitata all'intervallo $[\delta_{e,min}, \delta_{e,max}]$ poi mappata con l'intervallo $[-1, 1]$. L'angolo $\delta_e^\star$ è interpretabile come una deflessione equivalente del solo equilibratore che incorpora gli effetti della deflessione effettiva del trim tab.

La figura 15.21 a fronte riporta una rappresentazione dello schema di gestione del canale di controllo del beccheggio. Nel blocco `<summer>` viene definita la grandezza $\delta_e^\star$ che corrisponde a `fcs/pitch-trim-sum`. Essa rappresenta l'input sul canale del pitch mappato nel dominio $[-1, 1]$. Gli ingressi di questo blocco sono la posizione normalizzata della barra `fcs/elevator-cmd-norm` e della rotellina che muove l'aletta di trim `fcs/pitch-trim-cmd-norm`. Esse vengono sommate ed inviate al blocco corrispondente al tag `<aerosurface_scale>`. All'interno di quest'ultimo il comando normalizzato viene trasformato nella deflessione equivalente $\delta_e^\star$ espressa in radianti, cioè in `fcs/elevator-pos-rad`.

Si deve osservare che la gestione del canale di input è di fondamentale importanza per la correttezza ed il realismo della simulazione. Da una parte, si gestisce effettivamente ciò che è percepito dal pilota, cioè il movimento del comando — e non l'escursione dell'equilibratore. Dall'altra, il movimento dei comandi è messo in relazione alla deflessione della superficie di governo; questo permette di entrare nel database aerodinamico gestito dal FDM e di determinare a run-time i valori delle forze e dei momenti aerodinamici necessari a propagare lo stato della simulazione.

Si lascia per esercizio al lettore l'interpretazione del seguente frammento di input che definisce i canali di controllo del rollio e dell'imbardata:

```
<flight_control name="FCS: c172">
...
<channel name="Roll">
    <summer name="Roll Trim Sum">
        <input>fcs/aileron-cmd-norm</input>
        <input>fcs/roll-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
    </summer>
    <aerosurface_scale name="Left Aileron Control">
        <input>fcs/roll-trim-sum</input>
        <gain>0.01745</gain>
        <range>
            <min>-20</min>
            <max>15</max>
        </range>
        <output>fcs/left-aileron-pos-rad</output>
    </aerosurface_scale>
    <aerosurface_scale name="Left Aileron Position Normalized">
        <input>fcs/left-aileron-pos-deg</input>
        <domain>
            <min>-20</min>
            <max>15</max>
        </domain>
        <range>
            <min>-1</min>
            <max>1</max>
```

```
        </range>
        <output>fcs/left-aileron-pos-norm</output>
    </aerosurface_scale>
    <aerosurface_scale name="Right Aileron Control">
        <input>fcs/roll-trim-sum</input>
        <gain>-0.01745</gain>
        <range>
            <min>-20</min>
            <max>15</max>
        </range>
        <output>fcs/right-aileron-pos-rad</output>
    </aerosurface_scale>
    <aerosurface_scale name="Right Aileron Position Normalized">
        <input>fcs/right-aileron-pos-deg</input>
        <domain>
            <min>-15</min>
            <max>20</max>
        </domain>
        <range>
            <min>1</min>
            <max>-1</max>
        </range>
        <output>fcs/right-aileron-pos-norm</output>
    </aerosurface_scale>
</channel>
<channel name="Yaw">
    <summer name="Yaw Trim Sum">
        <input>fcs/rudder-cmd-norm</input>
        <input>fcs/yaw-trim-cmd-norm</input>
        <clipto>
            <min>-1</min>
            <max>1</max>
        </clipto>
    </summer>
    <aerosurface_scale name="Rudder Control">
        <input>fcs/yaw-trim-sum</input>
        <gain>0.01745</gain>
        <range>
            <min>-16</min>
            <max>16</max>
        </range>
        <output>fcs/rudder-pos-rad</output>
    </aerosurface_scale>
    <aerosurface_scale name="Rudder Position Normalized">
        <input>fcs/rudder-pos-deg</input>
        <domain>
            <min>-16</min>
            <max>16</max>
        </domain>
        <range>
            <min>-1</min>
            <max>1</max>
        </range>
        <output>fcs/rudder-pos-norm</output>
```

```
    </aerosurface_scale>
</channel>
</flight_control>
...
```

La parte rimanente della sezione <flight_control> definisce il comando dei flap:

```
<flight_control name="FCS: c172">
...
<channel name="Flaps">
    <kinematic name="Flaps Control">
        <input>fcs/flap-cmd-norm</input>
        <traverse>
            <setting>
                <position>0</position>
                <time>0</time>
            </setting>
            <setting>
                <position>10</position>
                <time>2</time>
            </setting>
            <setting>
                <position>20</position>
                <time>1</time>
            </setting>
            <setting>
                <position>30</position>
                <time>1</time>
            </setting>
        </traverse>
        <output>fcs/flap-pos-deg</output>
    </kinematic>
    <aerosurface_scale name="Flap Position Normalizer">
      <input>fcs/flap-pos-deg</input>
      <domain>
        <min>0</min>  <!-- Flaps actual minimum pos. -->
        <max>30</max>  <!-- Flaps actual maximum pos. -->
      </domain>
      <range>
        <min>0</min>  <!-- Flaps normalized minimum pos. -->
        <max>1</max>  <!-- Flaps normalized maximum pos. -->
      </range>
      <output>fcs/flap-pos-norm</output>
    </aerosurface_scale>
</channel>
...
</flight_control>
```

Nella definizione del canale dei flap compare il tag <kinematic>, destinato a modellare la risposta temporale di un oggetto mobile ad un comando di azionamento. La posizione corrente dei flap viene specificata con position mentre con time si va a definire il tempo che esso impiega a raggiungere una nuova posizione comandata.

Il tag <aerodynamics>

I marcatori <aerodynamics> ... </aerodynamics> delimitano la sezione più articolata dell'intero file di configurazione del FDM. All'interno del blocco aerodynamics, attraverso la possibilità offerta da JSBSim-ML di definire "funzioni" potenzialmente molto complesse, l'utente può arrivare a riprodurre in maniera completa il comportamento aerodinamico del velivolo. Il grado di fedeltà del modello aerodinamico di un FDM dipenderà in pratica dalla quantità e dalla qualità delle informazioni disponibili, cioè dalla completezza del database aerodinamico utilizzato.

Per migliorare la leggibilità del file di configurazione principale del FDM, soprattutto quando il database aerodinamico è costituito da numerose funzioni tabulari (*lookup table*), l'insieme di dati può essere conservato in un file separato, residente nella stessa cartella. In tal caso è possibile incorporare il file dei dati usando l'attributo file del tag <aerodynamics ...>. Normalmente la sezione aerodynamics è incorporata nel file di configurazione principale (si vedano i file di esempio distribuiti con JSBSim).

All'interno della sezione aerodynamics devono trovarsi 6 sottosezioni axis, tante quanti sono i gradi di libertà del moto. Per ciascun tag di apertura <axis ...> sarà dichiarato un valore differente dell'attributo name che specificherà il tipo di azione aerodinamica — cioè quale componente di forza o di momento si sta definendo. All'interno dei singoli blocchi axis, attraverso dei costrutti <function>, si specificheranno opportunamente le modalità di calcolo delle azioni aerodinamiche a tempo di esecuzione.

La struttura logica della sezione dei dati aerodinamici è la seguente:

```
...
<aerodynamics>
    ...     ← definizioni globali
    <axis name="DRAG"> ← oppure "X" oppure "AXIAL"
        ...
    </axis>
    <axis name="SIDE"> ← oppure "Y" oppure "SIDE"
        ...
    </axis>
    <axis name="LIFT"> ← oppure "Z" oppure "NORMAL"
        ...
    </axis>
    <axis name="ROLL">
        ...
    </axis>
    <axis name="PITCH">
        ...
    </axis>
    <axis name="YAW">
        ...
    </axis>
    ...     ← eventuali definizioni aggiuntive
</aerodynamics>
...
```

I valori "DRAG", "SIDE" e "LIFT" assegnati all'attributo name nei primi tre blocchi axis designano le componenti di forza lungo gli assi aerodinamici, rispettivamente, x_A, y_A e z_A. Il polo dei momenti aerodinamici è il punto P_{ARP} definito nella sezione metrics. È intorno a degli assi paralleli a quelli aerodinamici e passanti per P_{ARP} che si intendono

espresse le componenti di momento designate dai valori `"ROLL"`, `"PITCH"` e `"YAW"` nelle rimanenti tre istanze di `axis`. Le funzionalità interne di JSBSim provvedono poi a trasformare i momenti suddetti in momenti baricentrici intorno agli assi velivolo.

L'utente ha anche la possibilità di specificare le azioni aerodinamiche riferendosi ad altri assi. Ad esempio, se si dispone di dati riferiti agli assi velivolo, le prime tre istanze di `axis` avranno attributi `name` uguali, rispettivamente, ai valori `"X"`, `"Y"` e `"Z"` e serviranno a definire le modalità di calcolo delle componenti X_A, Y_A e Z_A della forza aerodinamica lungo gli assi body x_B, y_B e z_B. Si deve tener presente che in tal caso le componenti $\mathcal{L}_A$, $\mathcal{M}_A$ e $\mathcal{N}_A$ rappresentano dei momenti intorno ad assi paralleli agli assi velivolo e passanti per P_{ARP}.

All'interno di ciascuna sezione `axis` l'utente inserirà un certo numero di istanze del tag `<function>`. Prima di approfondire i dettagli della sezione `axis`, ci si soffermi sul costrutto `<function>`. Un blocco `function` di JSBSim-ML definisce una variabile dipendente in funzione dei valori assunti a run-time da altre variabili; queste ultime appartengono al catalogo delle funzioni predefinite di JSBSim oppure sono a loro volta delle variabili definite dall'utente tramite altri costrutti `<function>`. La sintassi di una `function` è ispirata al linguaggio di markup MathML (`http://www.w3.org/TR/MathML2`).

Due esempi di `function`, definite per il modello `c172p` nella sezione `aerodynamics` prima di tutti i blocchi `axis`, sono riportati nel frammento seguente:

```
<function name="aero/function/kCDge"> ← definisce K_CD,ge , si veda eq. (15.6)
  <description>Change_in_drag_due_to_ground_effect</description>
  <product> ← K_CD,ge = al prodotto di ...
    <value>1.0</value> ← 1× ...
    <table> ← una funzione tabulare ...
      <independentVar>
        aero/h_b-mac-ft     ← ... di h/(b/2), variabile indipendente
      </independentVar>
      <tableData> ← tabella di coppie (h/(b/2), K_CD,ge)
        0.0000  0.4800  ← riga 1
        0.1000  0.5150  ← riga 2
        0.1500  0.6290
        0.2000  0.7090
        0.3000  0.8150
        0.4000  0.8820  ← (h/(b/2))_i , (K_CD,ge)_i
        0.5000  0.9280
        0.6000  0.9620
        0.7000  0.9880
        0.8000  1.0000
        0.9000  1.0000
        1.0000  1.0000
        1.1000  1.0000
      </tableData> ← si veda figura 15.22
    </table>
  </product>
</function>
<function name="aero/function/kCLge"> ← definisce K_CL,ge
  <description>Change_in_lift_due_to_ground_effect</description>
  <product>
    <value>1.0</value>
    <table>
      <independentVar>aero/h_b-mac-ft</independentVar>
```

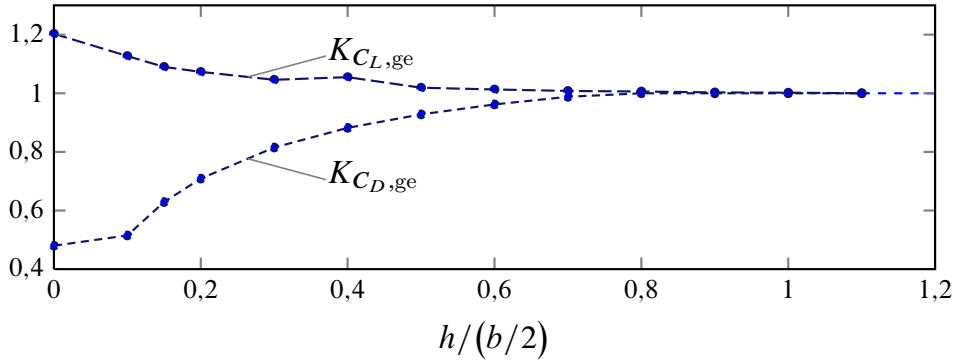


Figura 15.22 Le funzioni tabulari che definiscono le variabili `aero/function/kCLge` e `aero/function/kCDge` nel modello aerodinamico del `c172p`.

```
      <tableData>
        0.0000  1.2030   ← riga 1
        0.1000  1.1270   ← riga 2
        0.1500  1.0900
        0.2000  1.0730
        0.3000  1.0460
        0.4000  1.0550   ← (h/(b/2))_i , (K_CL,ge)_i
        0.5000  1.0190
        0.6000  1.0130
        0.7000  1.0080
        0.8000  1.0060
        0.9000  1.0030
        1.0000  1.0020
        1.1000  1.0000
      </tableData> ← si veda figura 15.22
    </table>
  </product>
</function>
```

Il primo costrutto `<function ...>` definisce la funzione tabulare $K_{C_D,\mathrm{ge}}$ della quota adimensionalizzata $h/(b/2) = -z_{\mathrm{E},G}/(b/2)$, introducendo una nuova variabile della simulazione dal nome `aero/function/kCDge`. La grandezza $h/(b/2)$ — associata alla variabile predefinita `aero/h_b-mac-ft` — è importante ai fini della modellazione dell'effetto suolo, che diventa significativo quando il velivolo ha una distanza dal terreno minore, all'incirca, della semiapertura alare. Pertanto, nel modello aerodinamico si introduce un fattore $K_{C_D,\mathrm{ge}}$ che moltiplica il coefficiente di resistenza del velivolo in area libera. Un analogo termine moltiplicativo $K_{C_L,\mathrm{ge}}$, associato alla variabile `aero/function/kCLge`, si definisce per il coefficiente di portanza. Le due funzioni tabulari sono rappresentate nella figura 15.22.

Nel frammento di input precedente si può osservare come una `function` si possa definire attraverso l'uso dei marcatori `<product>`, `<value>`, `<table>`, `<independentVar>` e `<tableData>`. La potenza espressiva del formato XML rende di facile interpretazione la definizione dei due fattori che modellano l'effetto suolo, soprattutto se ci si aiuta con il grafico della figura 15.22. Questo è un primo esempio di curve aerodinamiche non analitiche ma definite attraverso delle *lookup table*.

Visti gli esempi di costrutti `<function>` su riportati, a questo punto si osservi come si articola la sezione `axis` che definisce la portanza nel modello `c172p`. Nell'esempio

specifico si assume che la grandezza aerodinamica $L(t)$ possa calcolarsi come segue:

$$\begin{aligned} L &= L_{\text{basic}}\big|(\alpha_{\text{B}}, \phi_{\text{hyst}}) + \Delta L\big|\delta_{\text{f}} + \Delta L\big|\delta_{\text{e}} + \Delta L\big|\dot{\alpha}_{\text{B}} + \Delta L\big|q \\ &= \bar{q}_\infty S \Big[C_{L,\text{basic}}(\alpha_{\text{B}}, \phi_{\text{hyst}}) \\ &\qquad + \Delta C_L(\delta_{\text{f}}) + \Delta C_L(\delta_{\text{e}}) + \Delta C_L(\dot{\alpha}_{\text{B}}) + \Delta C_L(q) \Big] \end{aligned} \tag{15.5}$$

dove α_{B}, δ_{f}, δ_{e}, $\dot{\alpha}_{\text{B}}$ e q sono ben note; lo scalare adimensionale ϕ_{hyst} è un fattore variabile tra 0 e 1, che tipicamente assume valori non nulli in prossimità dello stallo dell'ala quando è spesso necessario incorporare nel modello gli effetti di isteresi aerodinamica.

La formulazione (15.5) si esprime in JSBSim-ML come segue:

```
<aerodynamics>
...    ← si definiscono K_CD,ge , K_CL,ge , ecc.
...
<axis name="LIFT"> ← si definisce L in base all'equazione (15.5)
  <function name="aero/coefficient/CLwbh">
      <description>Lift_due_to_alpha</description>
        ...          ← definisce L_basic|(α_B, φ_hyst)
  </function>
  <function name="aero/coefficient/CLDf">
      <description>Delta_lift_due_to_flap_deflection</description>
        ...          ← definisce ΔL|δ_f
  </function>
  <function name="aero/coefficient/CLDe">
      <description>Lift_due_to_Elevator_Deflection</description>
        ...          ← definisce ΔL|δ_e
  </function>
  <function name="aero/coefficient/CLadot">
      <description>Lift_due_to_alpha_rate</description>
        ...          ← definisce ΔL|α̇_B
  </function>
  <function name="aero/coefficient/CLq">
      <description>Lift_due_to_pitch_rate</description>
        ...          ← definisce ΔL|q
  </function>
</axis>
...
</aerodynamics>
...
```

Qui l'aspetto chiave è la cosiddetta tecnica del *coefficient build-up*, nel gergo della simulazione del volo. In altri termini, nel frammento di input precedente (così come in generale in tutte le istanze di `<axis>`) le variabili dipendenti definite da ciascuna `function` vengono sommate ottenendo la portanza (e in generale la componente aerodinamica risultante, di forza o di momento).

Il "build-up" non è altro che la somma a secondo membro della (15.5). Il termine "coefficient" è fuorviante essendo gli addendi di tale somma delle quantità dimensionali. Tuttavia la portanza totale può scriversi come prodotto di $q_\infty S$ per una somma di coefficienti aerodinamici. Da ciò discende anche la consuetudine di nominare le `function` del listato precedente come:

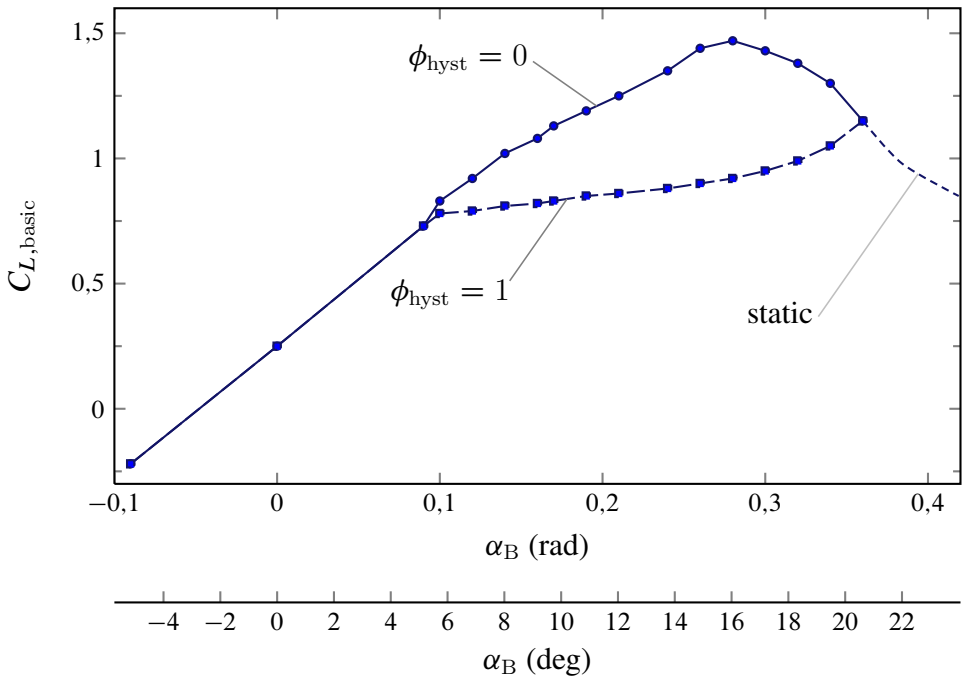


Figura 15.23 La funzione tabulare di due variabili $C_{L,\mathrm{basic}}(\alpha_{\mathrm{B}}, \phi_{\mathrm{hyst}})$ corrispondente alla `function` denominata `aero/coefficient/CLwbh` nel modello aerodinamico del `c172p`.

`aero/coefficient/CLwbh`	(relativo al Wing-Body-Horizontal tail),
`aero/coefficient/CLDf`	(dovuto a δ_{f}),
`aero/coefficient/CLDe`	(dovuto a δ_{e}),
`aero/coefficient/CLadot`	(dovuto ad $\dot{\alpha}_{\mathrm{B}}$),
`aero/coefficient/CLq`	(dovuto a q).

Ma, beninteso, i valori di queste variabili della simulazione sono quantità dimensionali, in questo caso delle forze. A tempo di esecuzione essi verranno sommati per ottenere $L(t)$; il vettore forza $-L(t)\boldsymbol{k}_{\mathrm{A}}$, noto $\alpha_{\mathrm{B}}(t)$, verrà poi opportunamente scomposto lungo gli assi velivolo.

A questo punto non resta che approfondire i dettagli sui singoli contributi alla portanza $L(t)$. Il primo addendo L_{basic} a secondo membro della (15.5), detto contributo "di base" (*basic*) alla portanza istantanea, è quello relativo al velivolo completo per valori nulli delle variabili δ_{f}, δ_{e}, $\dot{\alpha}_{\mathrm{B}}$ e q. Il coefficiente aerodinamico corrispondente $C_{L,\mathrm{basic}}$ è rappresentato graficamente nella figura 15.23 ed è definito come segue:

```
<axis name="LIFT">
  ...
  <function name="aero/coefficient/CLwbh"> ← si veda L_basic nella (15.5)
    <description>Lift_due_to_alpha</description>
    <product> ← L_basic = al prodotto di ...      — si veda la formula (15.6)
      <property>aero/qbar-psf</property> ← q̄_∞ × ...
      <property>metrics/Sw-sqft</property> ← S × ...
      <property>aero/function/kCLge</property> ← K_CL,ge × ...
      <table> ← C_L,basic, funzione tabulare di due variabili ...
        <independentVar lookup="row">
          aero/alpha-rad            ← di α_B (che scorre lungo le righe) ...
        </independentVar>
        <independentVar lookup="column">
          aero/stall-hyst-norm      ← di φ_hyst (che scorre lungo le colonne)
```

```
        </independentVar>
        <tableData>
           0.0000     1.0000   ← φ_hyst = 0 e φ_hyst = 1, colonne 1 e 2
-0.0900   -0.2200    -0.2200   ← riga 1
 0.0000    0.2500     0.2500   ← riga 2
 0.0900    0.7300     0.7300      ...
 0.1000    0.8300     0.7800
 0.1200    0.9200     0.7900   ← α_B i , C_L,basic(α_B i, 0) , C_L,basic(α_B i, 1)
 0.1400    1.0200     0.8100
 0.1600    1.0800     0.8200
 0.1700    1.1300     0.8300
 0.1900    1.1900     0.8500
 0.2100    1.2500     0.8600
 0.2400    1.3500     0.8800
 0.2600    1.4400     0.9000
 0.2800    1.4700     0.9200
 0.3000    1.4300     0.9500
 0.3200    1.3800     0.9900
 0.3400    1.3000     1.0500
 0.3600    1.1500     1.1500
        </tableData>
      </table>
    </product>
  </function>
  ...
</axis>
```

Il frammento precedente corrisponde a ciò che è richiesto da JSBSim-ML per definire la funzione

$$L_{\text{basic}}(t) = \bar{q}_\infty \, S \, K_{C_L,\text{ge}}\left(\frac{h}{b/2}\right) C_{L,\text{basic}}(\alpha_\text{B}, \phi_\text{hyst}) \tag{15.6}$$

Secondo questo modello la dipendenza temporale di L_{basic} discende, evidentemente, dalla dipendenza dal tempo delle variabili $\bar{q}_\infty$, $h = -z_{\text{E},G}$, α_B e ϕ_hyst. Normalmente la portanza, essendo una quantità dimensionale, dipende dalla quota e dalla velocità di volo attraverso la dipendenza da $\bar{q}_\infty$; la (15.6) mostra che in situazioni di vicinanza al suolo è possibile formulare una dipendenza dalla quota attraverso la funzione $K_{C_L,\text{ge}}$ su una scala di lunghezza paragonabile a $b/2$.

Inoltre, alla dipendenza di L_{basic} dall'angolo d'attacco, che è tipicamente riscontrabile nella variazione con α_B di $C_{L,\text{basic}}$, si aggiunge la dipendenza da un indice di isteresi aerodinamica ϕ_hyst. Quest'ultimo è calcolato attraverso le funzionalità interne di JSBSim tenendo conto del valore corrente l'angolo d'attacco e della rapidità $\dot{\alpha}_\text{B}$ con cui esso cambia nel tempo. Questo effetto instazionario che si innesca agli alti angoli d'attacco rappresenta una sofisticazione del modello aerodinamico (che non sempre è necessaria) ed è diverso dall'effetto instazionario rappresentato dalla derivata aerodinamica $C_{L\dot{\alpha}}$. Dall'esame della figura 15.23 si intuisce che, ad un dato istante t, quando $\alpha_\text{B}(t)$ è sufficientemente elevato, l'indice $\phi_\text{hyst}(t)$ può assumere un valore tra 0 e 1 e determinare un valore $C_{L,\text{basic}}(t)$ differente da quello statico. Ciò mostra come nelle simulazioni realistiche l'aerodinamica può essere modellata, seppur in maniera semplice, tenendo conto della storia del volo.

Passando agli altri contributi alla portanza $L(t)$, il secondo addendo $\Delta L\big|\delta_\text{f}$ a secondo membro della (15.5) rappresenta la variazione di portanza rispetto al valore $L_{\text{basic}}(t)$ dovuta alla deflessione dei flap. Questa funzione viene definita come segue:

```
<axis name="LIFT">
...
<function name="aero/coefficient/CLDf"> ← si veda ΔL|δf nella (15.5)
  <description>Delta_lift_due_to_flap_deflection</description>
  <product> ← ΔL|δf = al prodotto di ...      — si veda la formula (15.7)
    <property>aero/qbar-psf</property> ← q̄∞× ...
    <property>metrics/Sw-sqft</property> ← S× ...
    <property>aero/function/kCLge</property> ← K_CL,ge× ...
    <table> ← ΔCL(δf) , funzione tabulare di ...
      <independentVar>fcs/flap-pos-deg</independentVar> ← δf
      <tableData> ← tabella di coppie (δf, ΔCL)
         0.0000  0.0000
        10.0000  0.2000
        20.0000  0.3000
        30.0000  0.3500
      </tableData>
    </table>
  </product>
</function>
...
</axis>
```

Questa è la codifica in JSBSim-ML dell'espressione:

$$\Delta L\big|\delta_\mathrm{f} = \bar{q}_\infty \, S \, K_{C_L,\mathrm{ge}} \, \Delta C_L\big(\delta_\mathrm{f}\big) \tag{15.7}$$

cioè di un prodotto di variabili, tra le quali si compare la funzione $\Delta C_L(\delta_\mathrm{f})$. Nelle simulazioni realistiche è tipico descrivere questo incremento del coefficiente di portanza come funzione tabulare per tener conto di effetti non lineari. In alcuni casi questa grandezza può essere approssimata con il prodotto $C_{L\delta_\mathrm{f}}\,\delta_\mathrm{f}$ dove il coefficiente di proporzionalità con la deflessione del flap è il cosiddetto fattore di efficacia della ipersostentazione (*flap effectiveness*). La (15.7) mostra che anche per $\Delta L\big|\delta_\mathrm{f}$ è necessario tener conto dell'effetto suolo.

Il terzo addendo $\Delta L\big|\delta_\mathrm{e}$ a secondo membro della (15.5) rappresenta la variazione di portanza dovuta alla deflessione dell'equilibratore. Questa funzione viene definita come segue:

```
<axis name="LIFT">
...
<function name="aero/coefficient/CLDe"> ← si veda ΔL|δe nella (15.5)
  <description>Lift_due_to_Elevator_Deflection</description>
  <product> ← ΔL|δe = al prodotto di ...      — si veda la formula (15.8)
    <property>aero/qbar-psf</property> ← q̄∞× ...
    <property>metrics/Sw-sqft</property> ← S× ...
    <property>fcs/elevator-pos-rad</property> ← δe× ...
    <value>0.4300</value> ← C_Lδe in 1/rad
  </product>
</function>
...
</axis>
```

Ciò implementa in JSBSim-ML l'espressione:

$$\Delta L\big|\delta_\mathrm{e} = \bar{q}_\infty \, S \, C_{L_{\delta_\mathrm{e}}} \, \delta_\mathrm{e} \tag{15.8}$$

cioè di un prodotto di variabili, tra le quali si riconosce la potenza di controllo $C_{L_{\delta_e}}$ (*elevator effectiveness*). In questo caso la potenza di controllo è semplicemente una costante ed in JSBSim-ML viene introdotta attraverso i marcatori `<value> ... </value>`.

Il contributo $\Delta L\big|\dot{\alpha}_B$ viene codificato nella `function`:

```
<axis name="LIFT">
...
<function name="aero/coefficient/CLadot"> ← si veda ΔL|α̇B nella (15.5)
  <description>Lift_due_to_alpha_rate</description>
  <product> ← ΔL|α̇B = al prodotto di ...      — si veda la formula (15.9)
    <property>aero/qbar-psf</property> ← q̄∞ × ...
    <property>metrics/Sw-sqft</property> ← S × ...
    <property>aero/alphadot-rad_sec</property> ← α̇B × ...
    <property>aero/ci2vel</property> ← c̄/(2V) × ...
    <value>1.7000</value> ← C_Lα̇ in 1/rad
  </product>
</function>
...
</axis>
```

corrispondente all'espressione:

$$\Delta L\big|\dot{\alpha}_B = \bar{q}_\infty \, S \, C_{L_{\dot{\alpha}}} \frac{\dot{\alpha}_B \bar{c}}{2V} \tag{15.9}$$

dove $C_{L_{\dot{\alpha}}}$ è il ben noto coefficiente aerodinamico 'instazionario'. Si noti come il valore a run-time in s^{-1} del rapporto $\bar{c}/(2V)$ sia prelevato dalla variabile predefinita `aero/ci2vel`.

Infine, il contributo $\Delta L\big|q$ dovuto al moto di beccheggio viene codificato attraverso la `function`:

```
<axis name="LIFT">
...
<function name="aero/coefficient/CLq"> ← si veda ΔL|q nella (15.5)
  <description>Lift_due_to_pitch_rate</description>
  <product> ← ΔL|q = al prodotto di ...      — si veda la formula (15.10)
    <property>aero/qbar-psf</property> ← q̄∞ × ...
    <property>metrics/Sw-sqft</property> ← S × ...
    <property>velocities/q-aero-rad_sec</property> ← q × ...
    <property>aero/ci2vel</property> ← c̄/(2V) × ...
    <value>3.9000</value> ← C_Lq in 1/rad
  </product>
</function>
...
</axis>
```

corrispondente all'espressione:

$$\Delta L\big|q = \bar{q}_\infty \, S \, C_{L_q} \frac{q \bar{c}}{2V} \tag{15.10}$$

dove C_{L_q} è il ben noto coefficiente aerodinamico collegato al pitch.

Si lascia per esercizio al lettore l'interpretazione della parte rimanente del blocco `aerodynamics` riportata qui di seguito.

```
<aerodynamics>
  <alphalimits unit="RAD"> ← αB,min e αB,max nel database
```

```
    <min>-0.087</min>
    <max>0.28</max>
  </alphalimits>

  <hysteresis_limits unit="RAD"> ← intervallo di isteresi [α_B,min , α_B,max]_hyst
    <min>0.09</min>
    <max>0.36</max>
  </hysteresis_limits>

...    ← qui le definizioni di K_CD,ge , K_CL,ge

<function name="aero/function/velocity-induced-fps">
  <description> ← induzione dell'elica; incremento di velocità sull'impennaggio orizzontale
    velocity including the propulsion induced velocity.
  </description>
  <sum> ← u_ind = somma di ... (in assi aerodinamici)
    <property>velocities/u-aero-fps</property> ← (V_wind − V) · i_A + ...
    <property>  ← 2Δu_prop
      propulsion/engine/prop-induced-velocity_fps
    </property>
    <property>
      propulsion/engine/prop-induced-velocity_fps
    </property>
  </sum>
</function>

<function name="aero/function/qbar-induced-psf">
  <description> ← incremento di pressione dinamica in coda per l'induzione dell'elica
    q bar including the propulsion induced velocity.
  </description>
  <product> ← q̄_ind = ½ ρ u²_ind
    <property>aero/function/velocity-induced-fps</property>
    <property>aero/function/velocity-induced-fps</property>
    <property>atmosphere/rho-slugs_ft3</property>
    <value>0.5</value>
  </product>
</function>

<axis name="DRAG"> ← si veda la formulazione (15.11)
  <function name="aero/coefficient/CDo">
    <description>Drag_at_zero_lift</description>
      <product> ← D_0 = ...
        <property>aero/qbar-psf</property> ← q̄_∞ ×
        <property>metrics/Sw-sqft</property> ← S ×
        <value>0.027</value> ← C_D0
      </product>
    </function>
    <function name="aero/coefficient/CDDf">
      <description>Delta_drag_due_to_flap_deflection</description>
      <product> ← Δ⋆D|δ_f = al prodotto di ...
        <property>aero/qbar-psf</property> ← q̄_∞ ×
        <property>metrics/Sw-sqft</property> ← S ×
        <property>aero/function/kCDge</property> ← K_CD,ge ×
        <table> ← Δ⋆C_D(δ_f) , funzione tabulare di ...
```

```
          <independentVar>
              fcs/flap-pos-deg ← δf
          </independentVar>
          <tableData>
             0.0000   0.0000
            10.0000   0.0070
            20.0000   0.0120
            30.0000   0.0180
        </tableData>
      </table>
      </product>
    </function>
    <function name="aero/coefficient/CDwbh">
      <description>Drag_due_to_alpha</description>
      <product> ← ΔD|(αB, δf) = al prodotto di ...
        <property>aero/qbar-psf</property> ← q̄∞×
        <property>metrics/Sw-sqft</property> ← S×
        <property>aero/function/kCDge</property> ← K_CD,ge×
        <table> ← ΔCD(αB, δf) , funzione tabulare di ...
          <independentVar lookup="row">
            aero/alpha-rad  ← αB e di ...
          </independentVar>
          <independentVar lookup="column">
            fcs/flap-pos-deg  ← δf
          </independentVar>
          <tableData> ← si veda la figura 15.24
          0.0     10.0     20.0     30.0
-0.0873   0.0041   0.0000   0.0005   0.0014
-0.0698   0.0013   0.0004   0.0025   0.0041
-0.0524   0.0001   0.0023   0.0059   0.0084
-0.0349   0.0003   0.0057   0.0108   0.0141
-0.0175   0.0020   0.0105   0.0172   0.0212
 0.0000   0.0052   0.0168   0.0251   0.0299
 0.0175   0.0099   0.0248   0.0346   0.0402
 0.0349   0.0162   0.0342   0.0457   0.0521
 0.0524   0.0240   0.0452   0.0583   0.0655
 0.0698   0.0334   0.0577   0.0724   0.0804
 0.0873   0.0442   0.0718   0.0881   0.0968
 0.1047   0.0566   0.0874   0.1053   0.1148
 0.1222   0.0706   0.1045   0.1240   0.1343
 0.1396   0.0860   0.1232   0.1442   0.1554
 0.1571   0.0962   0.1353   0.1573   0.1690
 0.1745   0.1069   0.1479   0.1708   0.1830
 0.1920   0.1180   0.1610   0.1849   0.1975
 0.2094   0.1298   0.1746   0.1995   0.2126
 0.2269   0.1424   0.1892   0.2151   0.2286
 0.2443   0.1565   0.2054   0.2323   0.2464
 0.2618   0.1727   0.2240   0.2521   0.2667
 0.2793   0.1782   0.2302   0.2587   0.2735
 0.2967   0.1716   0.2227   0.2507   0.2653
 0.3142   0.1618   0.2115   0.2388   0.2531
 0.3316   0.1475   0.1951   0.2214   0.2351
 0.3491   0.1097   0.1512   0.1744   0.1866
          </tableData>
```

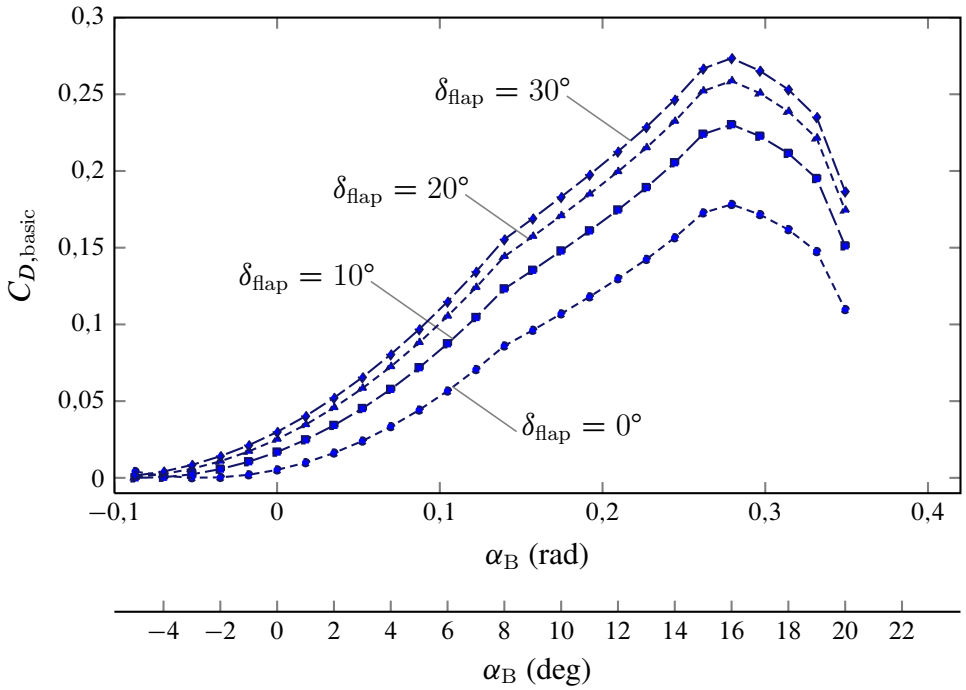


Figura 15.24 La funzione tabulare di due variabili $C_{D,\text{basic}}(\alpha_{\text{B}}, \delta_{\text{f}})$ corrispondente alla `function` denominata `aero/coefficient/CDwbh` nel modello aerodinamico del `c172p`.

```
        </table>
      </product>
    </function>

    <function name="aero/coefficient/CDDe">
      <description>Drag_due_to_Elevator_Deflection</description>
      <product> ← ΔD|δe = al prodotto di ...
        <property>aero/qbar-psf</property> ← q̄∞×
        <property>metrics/Sw-sqft</property> ← S×
        <property>fcs/mag-elevator-pos-rad</property> ← |δe|×
        <value>0.0000</value> ← C_{D δe}
      </product>
    </function>

  <function name="aero/coefficient/CDbeta">
    <description>Drag_due_to_sideslip</description>
    <product> ← ΔD|β = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>aero/mag-beta-rad</property> ← |β|×
      <value>0.1700</value> ← C_{D β}
    </product>
  </function>
</axis>
... (continua)
```

Per quanto riguarda il build-up della resistenza aerodinamica, si riconosca nel listato

precedente la formulazione:

$$\begin{aligned} D &= D_0 + \Delta^\star D\big|\delta_\mathrm{f} + \Delta D\big|(\alpha_\mathrm{B}, \delta_\mathrm{f}) + \Delta D\big|\delta_\mathrm{e} + \Delta D\big|\beta \\ &= \bar{q}_\infty S \Big\{ C_{D0} + K_{C_L,\mathrm{ge}}\Big[\Delta^\star C_D(\delta_\mathrm{f}) + \Delta C_D(\alpha_\mathrm{B}, \delta_\mathrm{f})\Big] \\ &\qquad + C_{D_{\delta_\mathrm{e}}}|\delta_\mathrm{e}| + C_{D_\beta}|\beta| \Big\} \end{aligned} \tag{15.11}$$

con C_{D0}, C_{D_β} e $C_{D_{\delta_\mathrm{e}}}$ i ben noti coefficienti aerodinamici e $\Delta^\star C_D(\delta_\mathrm{f})$ e $\Delta C_D(\alpha_\mathrm{B}, \delta_\mathrm{f})$ due funzioni tabulari che tengono conto dell'effetto combinato di α_B e δ_f sulla variazione del C_D del velivolo. La seconda funzione è rappresentata nella figura 15.24.

```
... (continua dal listato precedente)
<axis name="SIDE"> ← si veda la formulazione (15.12)
  <function name="aero/coefficient/CYb">
    <description>Side_force_due_to_beta</description>
    <product> ← ΔY|β = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <table> ← ΔC_Y(β, δf) , funzione tabulare di ...
        <independentVar lookup="row">
          aero/beta-rad  ← β e di ...
        </independentVar>
        <independentVar lookup="column">
          fcs/flap-pos-deg  ← δf
        </independentVar>
        <tableData>
                        0.0000   30.0000
         -0.3490        0.1370    0.1060
          0.0000        0.0000    0.0000
          0.3490       -0.1370   -0.1060
        </tableData>
      </table>
    </product>
  </function>

  <function name="aero/coefficient/CYda">
    <description>Side_force_due_to_aileron</description>
    <product> ← ΔY|δa = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>fcs/left-aileron-pos-rad</property> ← δa,left×
      <value>0.0000</value> ← C_Yδa in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/CYdr">
    <description>Side_force_due_to_rudder</description>
    <product> ← ΔY|δr = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>fcs/rudder-pos-rad</property> ← δr×
      <value>0.1870</value> ← C_Yδr in 1/rad
```

```
    </product>
  </function>

  <function name="aero/coefficient/CYp">
    <description>Side_force_due_to_roll_rate</description>
    <product> ← ΔY|p = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>aero/bi2vel</property> ← b/(2V)×
      <property>velocities/p-aero-rad_sec</property> ← p×
      <table> ← CYp(αB, δf) in 1/rad, funzione tabulare di ...
        <independentVar lookup="row">
          aero/alpha-rad  ← αB e di ...
        </independentVar>
        <independentVar lookup="column">
          fcs/flap-pos-deg  ← δf
        </independentVar>
        <tableData>
                      0.0       30.0
          0.0000    -0.0750    -0.1610
          0.0940    -0.1450    -0.2310
        </tableData>
      </table>
    </product>
  </function>

  <function name="aero/coefficient/CYr">
    <description>Side_force_due_to_yaw_rate</description>
    <product> ← ΔY|r = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>aero/bi2vel</property> ← b/(2V)×
      <property>velocities/r-aero-rad_sec</property> ← r×
      <table> ← CYr(αB, δf) in 1/rad, funzione tabulare di ...
        <independentVar lookup="row">
          aero/alpha-rad  ← αB e di ...
        </independentVar>
        <independentVar lookup="column">
          fcs/flap-pos-deg  ← δf
        </independentVar>
        <tableData>
                     0.0      30.0
          0.0000    0.2140    0.1620
          0.0940    0.2670    0.2150
        </tableData>
      </table>
    </product>
  </function>
</axis>

<axis name="LIFT">
...    ← qui la definizione di L secondo la formulazione (15.5)
</axis>
... (continua)
```

Il build-up della forza aerodinamica laterale corrisponde alla formulazione:

$$\begin{aligned} Y &= \Delta Y|_{\beta} + \Delta Y|_{\delta_a} + \Delta Y|_{\delta_r} + \Delta Y|_{p} + \Delta Y|_{r} \\ &= \bar{q}_\infty S \left[\Delta C_Y(\beta, \delta_f) + C_{Y_{\delta_a}} \delta_{a,\text{left}} + C_{Y_{\delta_r}} \delta_r \right. \\ &\qquad \left. + C_{Y_p}(\alpha_B, \delta_f) \frac{pb}{2V} + C_{Y_r}(\alpha_B, \delta_f) \frac{rb}{2V} \right] \end{aligned} \tag{15.12}$$

dove sono ben noti i significati dei coefficienti aerodinamici $C_{Y_{\delta_a}}$, $C_{Y_{\delta_r}}$, C_{Y_p} e C_{Y_r}. Gli ultimi due sono delle funzioni tabulari di α_B e δ_f. Un'analoga dipendenza in forma tabulare si ha per l'incremento $\Delta C_Y(\beta, \delta_f)$, che per piccoli angoli di derapata e deflessione nulla dei flap si esprimerebbe come $C_{Y_\beta}\beta$.

```
... (continua dal listato precedente)
<axis name="ROLL"> ← si veda la formulazione (15.13)
  <function name="aero/coefficient/Clb">
    <description>Roll_moment_due_to_beta</description>
    <product> ← ΔL|β = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <table> ← ΔC_L(β), funzione tabulare di ...
        <independentVar>aero/beta-rad</independentVar> ← β
        <tableData>
          -0.3490    0.0322
           0.0000    0.0000
           0.3490   -0.0322
        </tableData>
      </table>
    </product>
  </function>

  <function name="aero/coefficient/Clp">
    <description>
      Roll_moment_due_to_roll_rate_(roll_damping)
    </description>
    <product> ← ΔL|p = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>aero/bi2vel</property> ← b/(2V)×
      <property>velocities/p-aero-rad_sec</property> ← p×
      <value>-0.4840</value> ← C_Lp in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Clr">
    <description>Roll_moment_due_to_yaw_rate</description>
    <product> ← ΔL|r = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>aero/bi2vel</property> ← b/(2V)×
```

```
      <property>velocities/r-aero-rad_sec</property> ← r×
      <table> ← C_ℒr(α_B, δ_f) in 1/rad, funzione tabulare di ...
        <independentVar lookup="row">
          aero/alpha-rad  ← α_B e di ...
        </independentVar>
        <independentVar lookup="column">
          fcs/flap-pos-deg  ← δ_f
        </independentVar>
        <tableData>
                    0.0     30.0
          0.0000   0.0798   0.1246
          0.0940   0.1869   0.2317
        </tableData>
      </table>
    </product>
  </function>

  <function name="aero/coefficient/ClDa">
    <description>Roll_moment_due_to_aileron</description>
    <product> ← Δℒ|δ_a = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄_∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>fcs/left-aileron-pos-rad</property> ← δ_a,left×
      <value>0.2290</value> ← C_ℒδa in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cldr">
    <description>Roll_moment_due_to_rudder</description>
    <product> ← Δℒ|δ_r = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄_∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>fcs/rudder-pos-rad</property> ← δ_r×
      <value>0.0147</value> ← C_ℒδr in 1/rad
    </product>
  </function>
</axis>
... (continua)
```

Il build-up del momento di rollio (intorno all'asse aerodinamico x_A) corrisponde alla formulazione:

$$\begin{aligned}\mathcal{L} &= \Delta\mathcal{L}|\beta + \Delta\mathcal{L}|\delta_a + \Delta\mathcal{L}|\delta_r + \Delta\mathcal{L}|p + \Delta\mathcal{L}|r \\ &= \bar{q}_\infty\, S\, b \left[\Delta C_{\mathcal{L}}(\beta) + C_{\mathcal{L}_{\delta_a}} \delta_{a,\text{left}} + C_{\mathcal{L}_{\delta_r}} \delta_r + C_{\mathcal{L}_p} \frac{pb}{2V} + C_{\mathcal{L}_r} \frac{rb}{2V} \right]\end{aligned} \tag{15.13}$$

con il ben noto il significato dei coefficienti aerodinamici $C_{\mathcal{L}_{\delta_a}}$, $C_{\mathcal{L}_{\delta_r}}$, $C_{\mathcal{L}_p}$ e $C_{\mathcal{L}_r}$.

```
... (continua dal listato precedente)
<axis name="PITCH"> ← si veda la formulazione (15.14)
  <function name="aero/coefficient/Cmo">
    <description>Pitching_moment_at_zero_alpha</description>
```

```
    <product> ← ΔM0 = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <value>0.1000</value> ← CM0
    </product>
  </function>

  <function name="aero/coefficient/Cmalpha">
    <description>Pitch_moment_due_to_alpha</description>
    <product> ← ΔM|αB = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <sin> ← sin αB×
        <property>aero/alpha-rad</property>
      </sin>
      <value>-1.8000</value> ← CMα in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cmq">
    <description>Pitch_moment_due_to_pitch_rate</description>
    <product> ← ΔM|q = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <property>aero/ci2vel</property> ← c̄/(2V)×
      <property>velocities/q-aero-rad_sec</property> ← q×
      <value>-12.4000</value> ← CMq in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cmadot">
    <description>Pitch_moment_due_to_alpha_rate</description>
    <product> ← ΔM|α̇B = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <property>aero/ci2vel</property> ← c̄/(2V)×
      <property>aero/alphadot-rad_sec</property> ← α̇B×
      <value>-7.2700</value> ← CMα̇ in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cmde">
    <description>
      Pitch_moment_due_to_elevator_deflection
    </description>
    <product> ← ΔM|δe = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <property>fcs/elevator-pos-rad</property> ← δe×
```

```
      <value>-1.1220</value> ← C_{M_δe} in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cmdf">
    <description>
      Delta_pitching_moment_due_to_flap_deflection
    </description>
    <product> ← ΔM|δf = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/cbarw-ft</property> ← c̄×
      <table> ← ΔCM(δf), funzione tabulare di ...
        <independentVar>fcs/flap-pos-deg</independentVar> ← δf
        <tableData>
           0.0000    0.0000
          10.0000   -0.0654
          20.0000   -0.0981
          30.0000   -0.1140
        </tableData>
      </table>
    </product>
  </function>
</axis>
... (continua)
```

Il build-up del momento di beccheggio corrisponde alla formulazione:

$$\begin{aligned}\mathcal{M} &= \mathcal{M}_0 + \Delta\mathcal{M}\big|\alpha_\mathrm{B} + \Delta\mathcal{M}\big|\delta_\mathrm{f} + \Delta\mathcal{M}\big|\delta_\mathrm{e} + \Delta\mathcal{M}\big|q + \Delta\mathcal{M}\big|\dot{\alpha}_\mathrm{B} \\ &= \bar{q}_\infty\, S\, \bar{c} \left[C_{\mathcal{M}0} + C_{\mathcal{M}_\alpha} \sin\alpha_\mathrm{B} + \Delta C_\mathcal{M}\big(\delta_\mathrm{f}\big) + C_{\mathcal{M}_{\delta_\mathrm{e}}}\delta_\mathrm{e} \right. \\ &\qquad \left. + C_{\mathcal{M}_q}\frac{q\bar{c}}{2V} + C_{\mathcal{M}_{\dot{\alpha}_\mathrm{B}}}\frac{\dot{\alpha}_\mathrm{B}\bar{c}}{2V} \right]\end{aligned} \tag{15.14}$$

con il ben noto significato dei coefficienti aerodinamici $C_{\mathcal{M}0}$, $C_{\mathcal{M}_{\delta_\mathrm{e}}}$, $C_{\mathcal{M}_q}$ e $C_{\mathcal{M}_{\dot{\alpha}}}$. Il coefficiente $\Delta C_\mathcal{M}(\delta_\mathrm{f})$ è l'incremento di momento di beccheggio in presenza di ipersostentazione ed è una funzione tabulare di δ_f. Si noti che in questo modello l'effetto non lineare del discostamento verticale del baricentro dal polo dei momenti è rappresentato dalla proporzionalità con il $\sin\alpha_\mathrm{B}$ del contributo $\Delta\mathcal{M}\big|\alpha_\mathrm{B}$.

```
... (continua dal listato precedente)
<axis name="YAW"> ← si veda la formulazione (15.15)
  <function name="aero/coefficient/Cnb">
    <description>Yaw_moment_due_to_beta</description>
    <product> ← ΔN|β = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <table> ← ΔCN(β), funzione tabulare di ...
        <independentVar>aero/beta-rad</independentVar> ← β
        <tableData>
          -0.3490   -0.0205
           0.0000    0.0000
```

```
          0.3490    0.0205
        </tableData>
      </table>
    </product>
  </function>

  <function name="aero/coefficient/Cnp">
    <description>Yaw_moment_due_to_roll_rate</description>
    <product> ← Δ𝒩|p = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>aero/bi2vel</property> ← b/(2V)×
      <property>velocities/p-aero-rad_sec</property> ← p×
      <value>-0.0278</value> ← C_{𝒩_p} in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cnr">
    <description>Yaw_moment_due_to_yaw_rate</description>
    <product> ← Δ𝒩|r = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>aero/bi2vel</property> ← b/(2V)×
      <property>velocities/r-aero-rad_sec</property> ← r×
      <value>-0.0937</value> ← C_{𝒩_r} in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cnda">
    <description>Yaw_moment_due_to_aileron</description>
    <product> ← Δ𝒩|δa = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>fcs/left-aileron-pos-rad</property> ← δ_{a,left}×
      <value>-0.0053</value> ← C_{𝒩_{δa}} in 1/rad
    </product>
  </function>

  <function name="aero/coefficient/Cndr">
    <description>Yaw_moment_due_to_rudder</description>
    <product> ← Δ𝒩|δr = al prodotto di ...
      <property>aero/qbar-psf</property> ← q̄∞×
      <property>metrics/Sw-sqft</property> ← S×
      <property>metrics/bw-ft</property> ← b×
      <property>fcs/rudder-pos-rad</property> ← δr×
      <value>-0.0430</value> ← C_{𝒩_{δr}} in 1/rad
    </product>
  </function>
</axis>
</aerodynamics>
...
```

Il build-up del momento di imbardata (intorno all'asse aerodinamico z_A) corrisponde alla formulazione:

$$\begin{aligned}\mathcal{N} &= \Delta\mathcal{N}\big|_{\beta} + \Delta\mathcal{N}\big|_{\delta_a} + \Delta\mathcal{N}\big|_{\delta_r} + \Delta\mathcal{N}\big|_{p} + \Delta\mathcal{N}\big|_{r} \\ &= \bar{q}_\infty\, S\, b \left[\Delta C_{\mathcal{N}}(\beta) + C_{\mathcal{N}_{\delta_a}} \delta_{a,\text{left}} + C_{\mathcal{N}_{\delta_r}} \delta_r + C_{\mathcal{N}_p} \frac{pb}{2V} + C_{\mathcal{N}_r} \frac{rb}{2V} \right]\end{aligned} \tag{15.15}$$

con il ben noto il significato dei coefficienti aerodinamici $C_{\mathcal{N}_{\delta_a}}$, $C_{\mathcal{N}_{\delta_r}}$, $C_{\mathcal{N}_p}$ e $C_{\mathcal{N}_r}$.

Il tag <output>

Esistono diversi modi in cui il programma JSBSim può fornire in output i risultati delle simulazioni. Un dato, prodotto in un certo istante della simulazione, può essere concettualmente inviato a un determinato "canale d'uscita", che può essere un semplice file o una porta di comunicazione con un'altra applicazione. Per ogni esecuzione è possibile aprire canali di output multipli. La scelta dei canali e dei dati da inviare ad essi viene configurata attraverso il tag <output>.

Un blocco output può essere presente nel file di configurazione principale di un FDM oppure può comparire in un file separato. Nel secondo caso il file con le direttive di output può essere passato a JSBSim attraverso la riga di comando con l'opzione --outputlogfile. In ogni caso, il contenuto della sezione output fornirà a JSBSim le direttive necessarie a mandare su un determinato numero di canali d'uscita le storie temporali desiderate.

Passando ad un esempio pratico, si supponga di voler effettuare una simulazione utilizzando il modello di velivolo c172p e di voler configurare JSBSim in modo che salvi un file con le storie temporali dei sei gradi di libertà di corpo rigido. Si scriverà:

```
<output
    type="CSV"  ← il formato di scrittura dei dati
    name="JSBout_c172p_6DOF.csv"  ← il nome del file in cui scrivere i dati
    rate="60">  ← frequenza di scrittura in Hz
  <property> position/h-sl-ft </property>  ← quota h sul livello del mare in ft
  <property> position/lat-gc-rad </property>  ← latitud. geocentrica λ_Geoc in rad
  <property> position/long-gc-rad </property>  ← long. geocentrica μ_Geoc in rad
  <property> attitude/phi-rad </property>  ← φ in rad
  <property> attitude/theta-rad </property>  ← θ in rad
  <property> attitude/psi-rad </property>  ← ψ in rad
</output>
```

Con la direttiva type="CSV" si chiede di aprire un canale d'uscita corrispondente ad un file in formato CSV (*Comma Separated Values*). L'attributo name permette di specificare il nome del file di log, in questo caso JSBout_c172p_6DOF.csv. L'attributo rate permette di impostare la frequenza di scrittura sul canale, in questo caso pari a 60 Hz.

☛ Si osservi che il passo temporale Δt usato per integrare le equazioni del moto in JSBSim è configurabile dall'utente e che tipicamente esso corrisponde a una frequenza di 120 Hz. Si vedano gli esempi nella cartella ⟨*JSBSim root*⟩/scripts/. La frequenza di scrittura può essere diversa dalla frequenza di aggiornamento dello stato del velivolo. Spesso è conveniente inviare i dati sui canali d'uscita con frequenze molto minori di 120 Hz, ad esempio per non avere file di log di dimensioni eccessive in simulazioni di lunga durata.

La scelta delle specifiche variabili da scrivere sul canale di output può essere fatta attraverso il marcatore <property>. Così come si è visto per le function, questo tag torna utile ogni volta che si vuole utilizzare una variabile della simulazione — sia essa

predefinita oppure definita dall'utente per un particolare modello di velivolo. Si ricordi che nel gergo degli sviluppatori di JSBSim e di FlightGear le variabili del catalogo di un determinato FDM (si veda il paragrafo 15.3.1) vengono dette "properties". Da ciò deriva il nome del marcatore. L'esempio precedente non ha bisogno di particolari commenti e risulta chiaramente l'insieme delle variabili che verranno effettivamente stampate nel file.

La prima riga del file di output conterrà una lista di stringhe alfanumeriche separate da virgole. Le stringhe corrisponderanno ai nomi delle variabili scelte. Di default la prima di queste stringhe è sempre `Time`; ciò significa che, prima della lista delle variabili scelte, `JSBSim` scriverà sul canale il valore della variabile indipendente t.

L'esempio precedente permette di creare un file del tipo:

```
Time, h-sl-ft, lat-gc-deg, long-gc-deg, phi-rad, theta-rad, psi-rad
0.0000000, ⟨h @ t1⟩, ⟨λGeoc @ t1⟩, ⟨μGeoc @ t1⟩, ⟨ϕ @ t1⟩, ⟨θ @ t1⟩, ⟨ψ @ t1⟩
0.0166666, ⟨h @ t2⟩, ⟨λGeoc @ t2⟩, ⟨μGeoc @ t2⟩, ⟨ϕ @ t2⟩, ⟨θ @ t2⟩, ⟨ψ @ t2⟩
0.0333332, ⟨h @ t3⟩, ⟨λGeoc @ t3⟩, ⟨μGeoc @ t3⟩, ⟨ϕ @ t3⟩, ⟨θ @ t3⟩, ⟨ψ @ t3⟩
0.0499998, ⟨h @ t4⟩, ⟨λGeoc @ t4⟩, ⟨μGeoc @ t4⟩, ⟨ϕ @ t4⟩, ⟨θ @ t4⟩, ⟨ψ @ t4⟩
...
```

Nella generica riga $(i+1)$-ma compariranno, nell'ordine, il tempo corrente t_i (corrispondente alla frequenza di scrittura desiderata), la quota $h(t_i)$ in ft, la latitudine geocentrica $\lambda_{\text{Geoc}}(t_i)$ in rad, la longitudine geocentrica $\mu_{\text{Geoc}}(t_i)$ in rad, i tre angoli di Eulero $\phi(t_i)$, $\theta(t_i)$ e $\psi(t_i)$ in rad. I valori numerici contenuti nelle righe del file di log successive alla prima saranno agevolmente manipolabili da applicazioni come Matlab o Gnuplot per essere poi rappresentati graficamente.

Vista la semplicità con cui si può configurare la scrittura su file dei dati della simulazione, sarà possibile creare più file di output 'tematici', ad esempio uno contenente le componenti di forza e momento aerodinamici, uno con le deflessioni delle superfici di governo, uno con gli angoli e così via. Oppure, si potrà scegliere di collocare in un unico file una ben determinata lista di variabili che si intende utilizzare per l'analisi del volo simulato. A questo scopo, una volta scelto il velivolo, è consigliabile creare un file di testo con il catalogo di tutte le variabili utilizzate dal FDM per poter scegliere tra esse quelle più opportune. Questo perché, a seconda del tipo di velivolo, ad esempio se propulso ad elica o a jet, se monomotore o plurimotore, le variabili del catalogo presentano spesso delle significative differenze da un modello all'altro. Si rimanda al manuale d'uso di JSBSim per approfondimenti sulle possibilità di configurazione avanzata dei canali d'uscita.

15.4.3 Organizzazione delle simulazioni

Per utilizzare produttivamente JSBSim è necessario un primo passo importante che è quello di individuare un'opportuna cartella di lavoro. Ci si deve assicurare che in essa si trovi l'eseguibile `JSBSim(.exe)` oltre l'albero di cartelle riportato nella figura 15.13 a pagina 17. Naturalmente, un cartella di lavoro plausibile è la stessa ⟨*JSBSim root*⟩, dopo avervi copiato l'eseguibile ottenuto in ⟨*JSBSim root*⟩`/src/` a seguito della compilazione del software.

Nella cartella di lavoro, che chiameremo genericamente ⟨*Work*⟩, si archivieranno tutti i file di cui si avrà bisogno per eseguire le simulazioni. Si ricorda che devono essere presenti i file con il FDM del velivolo e delle condizioni iniziali in ⟨*Work*⟩`/`⟨*nome velivolo*⟩`/`, i file con i dati del motore e del propulsore in ⟨*Work*⟩`/engine/` ed eventuali file di utilità nella cartella metaWork`/systems/`. In particolare, converrà avere la cartella

⟨*Work*⟩`/scripts/` in cui salvare i file di script per le simulazioni batch. I file di log creati attraverso le direttive `<output>` verranno creati al livello della cartella ⟨*Work*⟩.

Gli scripts: programmare il volo

Con i file di configurazione posizionati correttamente sarà possibile effettuare una simulazione batch con `JSBSim` creando un opportuno file di script. Negli script è possibile dare comandi accettati dal JSBSim-ML accedendo a tutte le variabili presenti nel catalogo del velivolo.

Il tipo e l'ordine dei comandi all'interno dei file di script finisce per seguire lo stesso schema ricorrente. Dopo le intestazioni iniziali, tipiche dei file XML, si specifica il tipo di velivolo da utilizzare, il tempo di inizio e di fine simulazione e l'intervallo di tempo tra un passo e l'altro della simulazione.

Un esempio di file di script ispirato al file ⟨*JSBSim root*⟩`/scripts/c1723.xml` della distribuzione ufficiale è il seguente:

```
<?xml version="1.0"?>
<?xml-stylesheet
    type="text/xsl"
    href="http://jsbsim.sourceforge.net/JSBSimScript.xsl"?>
<runscript   ← tag di radice dello script
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://jsbsim.sf.net/JSBSimScript
   .xsd"
    name="C172-01A takeoff run">
  <!--
    This run is for testing the C172 altitude hold autopilot
  -->
  <use
    aircraft="c172x"        ← sceglie il velivolo ⟨Work⟩/c72x/c172.xml
    initialize="reset00"    ← condiz. iniziali in ⟨Work⟩/c72x/reset00.xml
  />
  <run
    start="0.0"                      ← tempo iniziale, in s
    end="200"                        ← tempo finale, in s
    dt="0.00833333333333333333"      ← Δt, in s
  >

    ...   ← lista degli eventi della simulazione

  </run>
</runscript>
```

Con questo script si intende eseguire una simulazione di decollo utilizzando il modello `c172x` dotato di autopilota. Dopo i due tag d'intestazione (di importanza minore e comunque sempre uguali per tutti gli script) compare il tag principale: `<runscript>`. All'interno del blocco `runscript` compaiono i due tag `<use>` e `<run>`. Dall'esame del listato precedente risulta immediato comprendere che queste due direttive, per mezzo degli attributi `aircraft`, `initialize`, `start`, `end` e `dt`, fissano il modello di velivolo, le condizioni iniziali e i tempi della simulazione. In particolare, il tag `<use ... />` è un esempio di marcatore XML senza omologo marcatore di chiusura; esso viene chiuso

anteponendo il carattere / alla seconda parentesi angolare e le informazioni necessarie sono contenute nelle assegnazioni degli attributi.

All'interno della sezione run si articolano i comandi impartiti al software di simulazione. Ciascuna azione corrisponderà ad un "evento", individuato dal un'istanza del marcatore <event>. Ecco un esempio del contenuto della sezione run:

```
...
<use aircraft="c172x" initialize="reset00"/>
<run start="0.0" end="200" dt="0.0083333333333333333">
  <!-- Initialize with brakes on --> ← inizializzazioni prima di qualsiasi evento
  <property> simulation/notify-time-trigger </property>
  <property value="1"> simulation/run_id </property>
  <property value="1"> fcs/left-brake-cmd-norm </property>
  <property value="1"> fcs/right-brake-cmd-norm </property>
  <property value="1"> fcs/center-brake-cmd-norm </property>
  <property value="3.49"> guidance/specified-heading-rad </property>
  <property value="1"> guidance/heading-selector-switch </property>
  <property value="2498"> simulation/randomseed </property>

  <event name="engine start">
    <description>
      Start the engine and set roll control to heading hold
    </description>
    <condition>
      simulation/sim-time-sec >= 0.25 ← non appena t ≥ 0,25 s
    </condition>
    <set
      name="fcs/throttle-cmd-norm"
      value="1.0"
      action="FG_RAMP"
      tc ="0.5"
    />
    <set name="propulsion/magneto_cmd" value="3"/>
    <set name="propulsion/starter_cmd" value="1"/>
    <set name="ap/roll-attitude-mode" value="1"/>
    <set name="ap/autopilot-roll-on" value="1"/>
    <notify> ← stampa a video (notifica)
      <property caption="Sim Run ID: ">simulation/run_id</property>
      <property>accelerations/a-pilot-x-ft_sec2</property>
      <property>accelerations/a-pilot-y-ft_sec2</property>
      <property>accelerations/a-pilot-z-ft_sec2</property>
      ...  ← qui le rimanenti variabili da stampare all'accadere dell'evento
    </notify>
  </event>

  <event name="Begin roll">
    <description>
       Release brakes and get rolling with flaps at 10 degrees.
    </description>
    <condition> simulation/sim-time-sec >= 3.0 </condition>
    <set name="fcs/left-brake-cmd-norm" value="0"/>
    <set name="fcs/right-brake-cmd-norm" value="0"/>
    <set name="fcs/center-brake-cmd-norm" value="0"/>
    <set name="fcs/flap-cmd-norm" value="0.33"/>
```

```
    <set name="fcs/elevator-cmd-norm" value="0.00"/>
    <notify>
      <property
        caption="Cal. airspeed (kts):  ">
        velocities/vc-kts</property>
      <property
        caption="Altitude (AGL, ft):   ">
        position/h-agl-ft</property>
      <property
        caption="Latitude (geod, deg): ">
        position/lat-geod-deg</property>
      <property
        caption="Altitude (geod, ft):  ">
        position/geod-alt-ft</property>
      <property
        caption="Throttle pos:         ">
        fcs/throttle-pos-norm[0] </property>
      <property
        caption="Mixture pos:          ">
        fcs/mixture-pos-norm[0] </property>
      <property
        caption="Weight (lbs):         ">
        inertia/weight-lbs </property>
      <property
        caption="Propeller RPM:        ">
        propulsion/engine/propeller-rpm </property>
    </notify>
  </event>

  <event name="Rotate">
    <description
      Set Autopilot for 1000 ft and rotate at 51 keas.
    </description>
    <condition> velocities/vc-kts >= 51 </condition>
    <set name="ap/altitude_setpoint" value="1000.0"/>
    <set name="ap/altitude_hold" value="1"/>
    <notify>
      ...   ← qui le variabili da stampare all'accadere dell'evento
    </notify>
  </event>

  <event name="Set autopilot for 6000 ft.">
    <description>
      Set Autopilot for 6000 ft after a five second delay.
    </description>
    <condition>
      velocities/vc-kts >= 51
    </condition>
    <delay>10.0</delay>
    <set name="ap/altitude_setpoint" value="6000.0"/>
    <notify>
      ...   ← qui le variabili da stampare all'accadere dell'evento
    </notify>
  </event>
```

```
  <event name="Adjust throttle/flaps">
    <description>
      Remove flaps at 1000 ft and set heading to 100;
      acquire heading.
    </description>
    <condition>
      position/h-agl-ft >= 1000
    </condition>
    <set name="fcs/flap-cmd-norm" value="0"/>
    <set name="guidance/specified-heading-rad" value="1.75"/>
    <set name="ap/roll-attitude-mode" value="1"/>
  </event>

  <event
    name="Time Notify"
    persistent="true">
    <description>
      Output message at 1 minute intervals
    </description>
    <condition> ← t ≥ trigger time
      simulation/sim-time-sec >= simulation/notify-time-trigger
    </condition>
    <set
      name="simulation/notify-time-trigger"
      value="60"
      type="FG_DELTA"/> ← riassegna il trigger time spostandolo in avanti di 60 s
    <notify>
      ...   ← qui le variabili da stampare all'accadere dell'evento
    </notify>
  </event>

</run>
</runscript>
```

Dall'esame del listato precedente si vede che gli eventi della simulazione sono definiti dal marcatore `<event>` e al loro interno dal marcatore `<condition>`. Tipicamente la condizione di "trig" dell'evento riguarda la variabile temporale, definita nel catalogo come `simulation/sim-time-sec`. Tuttavia una condizione può riguardare qualsiasi altra variabile del catalogo. Nell'esempio precedente si può notare l'evento nominato `"Set autopilot for 6000 ft"` che è associato alla condizione in cui la velocità calibrata V_c supera i 51 kts oppure l'evento `"Adjust throttle/flaps"` che si verifica quando la quota supera i 1000 ft.

Allo scoccare di un determinato evento, cioè al verificarsi di una certa condizione, alcuni valori delle variabili della simulazione vengono impostati attraverso il marcatore `<set ... />`. Per mezzo degli attributi `name` e `value` si può richiedere che in certe condizioni venga assegnato istantaneamente un dato valore ad una determinata variabile. In altri casi, con gli attributi `type` e `tc` si può, ad esempio, richiedere che non appena si verifica l'evento la variabile debba subire una variazione temporale "a rampa" (`"FG_RAMP"`) con una data costante di tempo. In altre occasioni ancora si può richiedere un incremento della variabile (`"FG_DELTA"`), circostanza in cui potrebbe non essere necessario conoscerne il valore corrente né quello finale.

Un'attenta riflessione sulle possibilità offerte dal linguaggio di scripting di JSBSim non può che far concludere che con gli script gli utenti hanno a disposizione una modalità, allo stesso tempo, semplice, intuitiva e potente per poter impostare simulazioni batch. Si rimanda al manuale di riferimento e agli esempi contenuti nella cartella ⟨*JSBSim root*⟩`/scripts/` della distribuzione ufficiale per ulteriori approfondimenti sulle simulazioni batch.

File di inizializzazione

Un file di inizializzazione di JSBSim contiene i valori da assegnare ad alcun variabili della simulazione al tempo iniziale. Ovviamente, anche questo file è di tipo XML e contiene dei tag chiave accettati in lettura dalla procedura di inilizializzazione di JSBSim.

Ecco come si presenta un file del genere quando si vuole assegnare una condizione di partenza con il velivolo in prossimità del suolo, in una certa posizione geografica:

```
<?xml version="1.0"?>
<initialize name="reset00">   ← tag di radice del file di inizializzazione
  <!--
    This file sets up the aircraft to start off
    from the runway in preparation for takeoff.
  -->
  <ubody unit="FT/SEC">    0.0  </ubody>
  <vbody unit="FT/SEC">    0.0  </vbody>
  <wbody unit="FT/SEC">    0.0  </wbody>
  <longitude unit="DEG"> -95.163839  </longitude>
  <latitude unit="DEG">   29.593978  </latitude>
  <phi unit="DEG">         0.0  </phi>
  <theta unit="DEG">       0.0  </theta>
  <psi unit="DEG">       200.0  </psi>
  <altitude unit="FT">     4.6  </altitude>
  <elevation unit="FT">    0.0  </elevation>
  <hwind> 0.0 </hwind>
</initialize>
```

Il marcatore fondamentale, che racchiude tutti gli altri, è `<initialize>`. Si riconoscono le assegnazioni delle componenti di velocità in assi body, delle coordinate geografiche, della quota e degli angoli di Eulero iniziali.

In questo caso particolare, si noti che la quota iniziale è dell'ordine di grandezza della distanza verticale del baricentro dai punti di contatto del carrello principale. D'altra parte, è del tutto plausibile pensare di assegnare una posizione ed un orientamento iniziali qualsiasi. Ad esempio, si può assegnare una quota di 3000 ft, una velocità nulla, angoli $\psi(t_0) = 200\,\text{deg}$, $\theta(t_0) = -85\,\text{deg}$ (positzione picchiata) e $\phi(t_0) = 0\,\text{deg}$ (ali livelate). In tal caso, se la velocità iniziale è nulla il velivolo effettuerà un'affondata; l'esito della storia di volo dipenderà dall'intervento del pilota e dall'eventuale autopilota (se soggetto ad impostazioni particolari).

★ ★ ★

Esercizio 15.1: *Eseguire lo script* `c1723.xml` *della distribuzione ufficiale*

Individuare il file di script ⟨*JSBSim root*⟩`/scripts/c1723.xml` e interpretarne i comandi alla luce di quanto visto sopra.

Verificare l'esistenza nella distribuzione ufficiale di JSBSim del file di configurazione `c172x.xml` nella cartella ⟨*JSBSim root*⟩`/aircraft/c172x/`. Controllare le impostazioni del marcatore `<output>` ed eventualmente modificarle per ottenere un output personalizzato. Verificare l'esistenza, nella stessa cartella in cui risiede `c172x.xml`, del file `reset00.xml` con i valori iniziali delle variabili di stato.

Eseguire una simulazione batch con `JSBSim` utilizzando lo script `c1723.xml` e interpretare l'output a video e su file. Scegliere se eseguire il programma da una cartella di lavoro personalizzata oppure da ⟨*JSBSim root*⟩.
Verificare che il comando:

```
$ JSBSim --script=scripts/c1723.xml ⟨invio⟩
```

ha il seguente output a video:

```
     JSBSim Flight Dynamics Model v1.0 Mar 26 2012 11:48:36
            [JSBSim-ML v2.0]

JSBSim startup beginning ...

Aircraft Configuration File Cessna C-172 Skyhawk II
Version: 2.00
This aircraft model is BETA release!!!
This aircraft model probably will not fly as expected.
Use this model for development purposes ONLY!!!
  Description:   Models a 1982 Cessna 172P.
  Model Author:  Tony Peden
  Creation Date: 1999-01-01
...
  Output data set: 0    Output parameters read inline
    Log output goes to file: JSBout172B.csv in CSV format output
                             at rate 20.0000 Hz
    Simulation parameters logged
    Aerosurface parameters logged
    Rate parameters logged
    Velocity parameters logged
    Force parameters logged
    Moments parameters logged
    Atmosphere parameters logged
    Mass parameters logged
    Propagate parameters logged
    Ground parameters logged
    FCS parameters logged
    Propulsion parameters logged
    Properties logged:
      - vrp-gc-latitude_deg
      - vrp-longitude_deg
      - vrp-radius-ft
...
End of vehicle configuration loading.
------------------------------------------------------------------------------
Script: "C172-01A takeoff run"
  begins at 0.0 seconds and runs to 200.0 seconds
  with dt = 0.008333 (120.000000 Hz)

Local property: simulation/notify-time-trigger = 0.000000
Local property: simulation/run_id = 1.000000
Local property: fcs/left-brake-cmd-norm = 1.000000
Local property: fcs/right-brake-cmd-norm = 1.000000
Local property: fcs/center-brake-cmd-norm = 1.000000
Local property: guidance/specified-heading-rad = 3.490000
Local property: guidance/heading-selector-switch = 1.000000
Local property: simulation/randomseed = 0.000000

Event 0 (engine start):
  When first triggered, executes once if all of the following are true:
    sim-time-sec >= 0.250000
  Actions taken:
    {
      set fcs/throttle-cmd-norm to 1.000000 (constant via ramp
          with time constant 0.500000)
      set propulsion/magneto_cmd to 3.000000 (constant via step)
      set propulsion/starter_cmd to 1.000000 (constant via step)
      set ap/roll-attitude-mode to 1.000000 (constant via step)
      set ap/autopilot-roll-on to 1.000000 (constant via step)
    }
  Notifications:
    {
```

```
      simulation/run_id
      ap/hdg-roll-err-c1
      accelerations/a-pilot-x-ft_sec2
      accelerations/a-pilot-y-ft_sec2
      accelerations/a-pilot-z-ft_sec2
      ...
    }

Event 1 (Begin roll):
  When first triggered, executes once if all of the following are true:
    sim-time-sec >= 3.000000
  Actions taken:
    {
      set fcs/left-brake-cmd-norm to 0.000000 (constant via step)
      set fcs/right-brake-cmd-norm to 0.000000 (constant via step)
      set fcs/center-brake-cmd-norm to 0.000000 (constant via step)
      set fcs/flap-cmd-norm to 0.330000 (constant via step)
      set fcs/elevator-cmd-norm to 0.000000 (constant via step)
    }
  Notifications:
    {
      velocities/vc-kts
      position/h-agl-ft
      position/lat-geod-deg
      position/geod-alt-ft
      fcs/throttle-pos-norm
      fcs/mixture-pos-norm
      inertia/weight-lbs
      propulsion/engine/propeller-rpm
    }

Event 2 (Rotate):
  When first triggered, executes once if all of the following are true:
    vc-kts >= 51.000000
  Actions taken:
    {
      set ap/altitude_setpoint to 1000.000000 (constant via step)
      set ap/altitude_hold to 1.000000 (constant via step)
    }
  Notifications:
    {
      velocities/vc-kts
      position/h-agl-ft
      position/lat-geod-deg
      position/geod-alt-ft
      fcs/throttle-pos-norm
      fcs/mixture-pos-norm
      inertia/weight-lbs
      propulsion/engine/propeller-rpm
    }

Event 3 (Set autopilot for 6000 ft.):
  When first triggered, executes once if all of the following are true:
    vc-kts >= 51.000000
  Actions taken (after a delay of 10.000000 secs):
    {
      set ap/altitude_setpoint to 6000.000000 (constant via step)
    }
  Notifications:
    {
      velocities/vc-kts
      position/h-agl-ft
      position/lat-geod-deg
      position/geod-alt-ft
      fcs/throttle-pos-norm
      fcs/mixture-pos-norm
      inertia/weight-lbs
      propulsion/engine/propeller-rpm
    }

Event 4 (Adjust throttle/flaps):
  When first triggered, executes once if all of the following are true:
    h-agl-ft >= 1000.000000
  Actions taken:
    {
      set fcs/flap-cmd-norm to 0.000000 (constant via step)
      set guidance/specified-heading-rad to 1.750000 (constant via step)
      set ap/roll-attitude-mode to 1.000000 (constant via step)
    }

Event 5 (Time Notify):
  Whenever triggered, executes once if all of the following are true:
    sim-time-sec >= notify-time-trigger
  Actions taken:
    {
```

```
      set simulation/notify-time-trigger to 60.000000 (delta via step)
    }
  Notifications:
    {
      velocities/vc-kts
      position/h-agl-ft
      position/lat-geod-deg
      position/geod-alt-ft
      fcs/throttle-pos-norm
      fcs/mixture-pos-norm
      inertia/weight-lbs
      propulsion/engine/propeller-rpm
    }

------------------------------------------------------------------
State Report at sim time: 0.000000 seconds
  Position
    ECI:   -1637698.664775363, -18121987.2251045, 10334144.4994273 (x,y,z, in ft)
    ECEF:  -1637698.664775 , -18121987.225104 , 10334144.499427 (x,y,z, in ft)
    Local: 29.593978, -95.163839, 4.600000 (lat, lon, alt in deg and ft)

  Orientation
    ECI:   148.9431571455538, 54.7962082214304, 228.4457634111151 (phi, theta, psi in deg)
    Local: -1.693223314408871e-14, 6.265051503247011e-15, 200 (phi, theta, psi in deg)

  Velocity
    ECI:   1321.476148739929, -119.422869988884, 0 (x,y,z in ft/s)
    ECEF:  0, 0, 0 (x,y,z in ft/s)
    Local: 0.000000 , 0.000000 , 0.000000 (n,e,d in ft/sec)
    Body:  -0.000000 , 0.000000 , 0.000000 (u,v,w in ft/sec)

  Body Rates (relative to given frame, expressed in body frame)
    ECI:   -0.003413932570522442, 0.001242569837462072, -0.002063343901886427 (p,q,r in deg/s)
    ECEF:  0, 0, 0 (p,q,r in deg/s)

---- JSBSim Execution beginning ... ------------------------------------------


  Event 5 (Time Notify) executed at time: 0.008333
    Cal. airspeed (kts):   = 0.000000
    Altitude (AGL, ft):    = 4.600000
    Latitude (geod, deg):  = 29.759390
    Altitude (geod, ft):   = 17184.610116
    Throttle pos:          = 0.000000
    Mixture pos:           = 0.999465
    Weight (lbs):          = 2400.000000
    Propeller RPM:         = 0.000000

Start: Monday March 26 2012 12:45:47 (HH:MM:SS)
0: GEAR_CONTACT: Nose Gear 1
1: GEAR_CONTACT: Left Main Gear 1
2: GEAR_CONTACT: Right Main Gear 1

  Event 0 (engine start) executed at time: 0.258333
    Sim Run ID:  = 1.000000
    ap/hdg-roll-err-c1 = 0.500000
    accelerations/a-pilot-x-ft_sec2 = 0.832031
    accelerations/a-pilot-y-ft_sec2 = 0.018330
    accelerations/a-pilot-z-ft_sec2 = -32.135820
    ...

  Event 1 (Begin roll) executed at time: 3.008333
    Cal. airspeed (kts):   = 0.108162
    Altitude (AGL, ft):    = 4.320328
    Latitude (geod, deg):  = 29.759389
    Altitude (geod, ft):   = 17184.330239
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.999475
    Weight (lbs):          = 2399.983460
    Propeller RPM:         = 2166.794064

  Event 2 (Rotate) executed at time: 19.550000
    Cal. airspeed (kts):   = 51.009739
    Altitude (AGL, ft):    = 4.431240
    Latitude (geod, deg):  = 29.757512
    Altitude (geod, ft):   = 17182.468212
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.999471
    Weight (lbs):          = 2399.635256
    Propeller RPM:         = 2559.781083

3: GEAR_CONTACT: Nose Gear 0
4: GEAR_CONTACT: Left Main Gear 0
5: GEAR_CONTACT: Right Main Gear 0
```

```
Takeoff report for Nose Gear (Liftoff at time: 25.316667 seconds)
  Distance traveled:                827.730235 ft,     252.292176 meters
  Distance traveled (over 50'):     1273.151421 ft,     388.056553 meters
  [Altitude (ASL): 53.143085 ft. / 16.198012 m  | Temperature: 58.810854 F / 14.894919 C]
  [Velocity (KCAS): 55.778993]

Takeoff report for Left Main Gear (Liftoff at time: 25.391667 seconds)
  Distance traveled:                886.066545 ft,     270.073083 meters
  Distance traveled (over 50'):     1280.061982 ft,     390.162892 meters
  [Altitude (ASL): 54.683206 ft. / 16.667441 m  | Temperature: 58.805373 F / 14.891874 C]
  [Velocity (KCAS): 55.638099]

Takeoff report for Right Main Gear (Liftoff at time: 25.391667 seconds)
  Distance traveled:                891.606864 ft,     271.761772 meters
  Distance traveled (over 50'):     1280.061982 ft,     390.162892 meters
  [Altitude (ASL): 54.683206 ft. / 16.667441 m  | Temperature: 58.805373 F / 14.891874 C]
  [Velocity (KCAS): 55.638099]

  Event 3 (Set autopilot for 6000 ft.) executed at time: 29.558333
    Cal. airspeed (kts):   = 49.264031
    Altitude (AGL, ft):    = 129.235030
    Latitude (geod, deg):  = 29.755149
    Altitude (geod, ft):   = 17304.789667
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.994976
    Weight (lbs):          = 2399.416158
    Propeller RPM:         = 2559.588059

  Event 5 (Time Notify) executed at time: 60.008333
    Cal. airspeed (kts):   = 48.201746
    Altitude (AGL, ft):    = 444.211216
    Latitude (geod, deg):  = 29.748295
    Altitude (geod, ft):   = 17612.565866
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.983693
    Weight (lbs):          = 2398.758955
    Propeller RPM:         = 2552.388519

  Event 4 (Adjust throttle/flaps) executed at time: 117.308333

  Event 5 (Time Notify) executed at time: 120.000000
    Cal. airspeed (kts):   = 48.863781
    Altitude (AGL, ft):    = 1032.760745
    Latitude (geod, deg):  = 29.734860
    Altitude (geod, ft):   = 18187.006461
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.962886
    Weight (lbs):          = 2397.483634
    Propeller RPM:         = 2553.442572

  Event 5 (Time Notify) executed at time: 180.000000
    Cal. airspeed (kts):   = 60.004984
    Altitude (AGL, ft):    = 1690.757735
    Latitude (geod, deg):  = 29.730467
    Altitude (geod, ft):   = 18840.392488
    Throttle pos:          = 1.000000
    Mixture pos:           = 0.940052
    Weight (lbs):          = 2396.212546
    Propeller RPM:         = 2627.150236

End: Monday March 26 2012 12:45:49 (HH:MM:SS)
```

❖

Esercizio 15.2: *Uno script per la simulazione di affondata*

Si assegni un velivolo alla quota iniziale di 2000 m, con orientamento predisposto per un'affondata e velocità iniziale nulla. Modificare lo script `c1723.xml` della distribuzione ufficiale comandando l'autopilota, per $t = t_0 + 2$ s, in modo che raggiunga la quota di 5000 m mantenendo le ali livellate.

❖

15.5 Leggere e visualizzare i dati di output

La distribuzione ufficiale di JSBSim comprende la cartella

⟨*JSBSim root*⟩`/src/utilities/`

in cui si trova il file `prep_plot.cpp`. Questo è il sorgente di un programma di utilità capace di leggere i file di output di JSBSim e di generare file di script per Gnuplot utili a confezionare i grafici delle storie di volo. Il lettore interessato all'uso di questa utilità troverà nelle prime righe di commento del sorgente le semplici istruzioni di compilazione del programma `prep_plot(.exe)`. Esso potrà essere copiato nella cartella ⟨*Work*⟩ e utilizzato attraverso la bash shell.

Lavorando con una bash shell sarà possibile automatizzare la generazione di un file PostScript o PDF (*Portable Document Format*) contenente i grafici corrispondenti alle impostazioni dei marcatori `<output>`. Per fare ciò ci si deve assicurare di aver installato correttamente il programma Gnuplot (http://www.gnuplot.info/) e il programma GhostScript (http://it.wikipedia.org/wiki/Ghostscript). Si rimanda alla lettura del manuale di riferimento di JSBSim e dei commenti del file `prep_plot.cpp` per approfondimenti sulle utilità di disegno offerte da JSBSim.

Un'alternativa al metodo precedente è l'uso dell'applicazione Matlab. Qui di seguito si proporrà uno schema di lavoro basato sulla configurazione di canali di output multipli e su uno script in linguaggio Matlab per il disegno delle storie di volo. Il materiale potrà essere reperito scaricando l'archivio wpage.unina.it/agodemar/DSV-DQV/JSBSim_Plot_Script.rar.

Nel file di configurazione del velivolo scelto occorrerà creare delle sezioni `output` come quelle seguenti:

```
<output name="c172_fcs.csv" rate="60" type="CSV">
  <property> fcs/aileron-cmd-norm </property>        <!--  2 -->
  <property> fcs/elevator-cmd-norm </property>       <!--  3 -->
  <property> fcs/rudder-cmd-norm </property>         <!--  4 -->
  <property> fcs/flap-cmd-norm </property>           <!--  5 -->
  <property> fcs/elevator-pos-rad </property>        <!--  6 -->
  <property> fcs/elevator-pos-deg </property>        <!--  7 -->
  <property> fcs/elevator-pos-norm </property>       <!--  8 -->
  <property> fcs/rudder-pos-rad </property>          <!--  9 -->
  <property> fcs/rudder-pos-deg </property>          <!-- 10 -->
  <property> fcs/rudder-pos-norm </property>         <!-- 11 -->
  <property> fcs/flap-pos-rad </property>            <!-- 12 -->
  <property> fcs/flap-pos-deg </property>            <!-- 13 -->
  <property> fcs/flap-pos-norm </property>           <!-- 14 -->
  <property> fcs/left-aileron-pos-rad </property>    <!-- 15 -->
  <property> fcs/left-aileron-pos-deg </property>    <!-- 16 -->
  <property> fcs/left-aileron-pos-norm </property>   <!-- 17 -->
  <property> fcs/right-aileron-pos-rad </property>   <!-- 18 -->
  <property> fcs/right-aileron-pos-deg </property>   <!-- 19 -->
  <property> fcs/right-aileron-pos-norm </property>  <!-- 20 -->
  <property> fcs/throttle-cmd-norm</property>        <!-- 21 -->
  <property> fcs/throttle-pos-norm</property>        <!-- 22 -->
</output>

<output name="c172_forces.csv" rate="60" type="CSV">
  <property> forces/fbx-aero-lbs </property>    <!--  2 -->
```

```
  <property> forces/fby-aero-lbs </property>    <!--  3 -->
  <property> forces/fbz-aero-lbs </property>    <!--  4 -->
  <property> forces/fwx-aero-lbs </property>    <!--  5 -->
  <property> forces/fwy-aero-lbs </property>    <!--  6 -->
  <property> forces/fwz-aero-lbs </property>    <!--  7 -->
  <property> forces/fbx-gear-lbs </property>    <!--  8 -->
  <property> forces/fby-gear-lbs </property>    <!--  9 -->
  <property> forces/fbz-gear-lbs </property>    <!-- 10 -->
  <property> forces/fbx-total-lbs </property>   <!-- 11 -->
  <property> forces/fby-total-lbs </property>   <!-- 12 -->
  <property> forces/fbz-total-lbs </property>   <!-- 13 -->
  <property> forces/load-factor </property>     <!-- 14 -->
  <property> moments/l-aero-lbsft </property>   <!-- 15 -->
  <property> moments/m-aero-lbsft </property>   <!-- 16 -->
  <property> moments/n-aero-lbsft </property>   <!-- 17 -->
  <property> moments/l-gear-lbsft </property>   <!-- 18 -->
  <property> moments/m-gear-lbsft </property>   <!-- 19 -->
  <property> moments/n-gear-lbsft </property>   <!-- 20 -->
  <property> moments/l-total-lbsft </property>  <!-- 21 -->
  <property> moments/m-total-lbsft </property>  <!-- 22 -->
  <property> moments/n-total-lbsft </property>  <!-- 23 -->
</output>

<output name="c172_aero.csv" rate="60" type="CSV">
  <property> aero/alpha-rad </property>             <!--  2 -->
  <property> aero/beta-rad </property>              <!--  3 -->
  <property> aero/alpha-deg </property>             <!--  4 -->
  <property> aero/beta-deg </property>              <!--  5 -->
  <property> aero/Re </property>                    <!--  6 -->
  <property> aero/qbar-psf </property>              <!--  7 -->
  <property> aero/alphadot-rad_sec </property>      <!--  8 -->
  <property> aero/betadot-rad_sec </property>       <!--  9 -->
  <property> aero/alphadot-deg_sec </property>      <!-- 10 -->
  <property> aero/betadot-deg_sec </property>       <!-- 11 -->
  <property> aero/coefficient/CDo </property>       <!-- 12 -->
  <property> aero/coefficient/CDDf </property>      <!-- 13 -->
  <property> aero/coefficient/CDwbh </property>     <!-- 14 -->
  <property> aero/coefficient/CDDe </property>      <!-- 15 -->
  <property> aero/coefficient/CDbeta </property>    <!-- 16 -->
  <property> aero/coefficient/CYb </property>       <!-- 17 -->
  <property> aero/coefficient/CYda </property>      <!-- 18 -->
  <property> aero/coefficient/CYdr </property>      <!-- 19 -->
  <property> aero/coefficient/CYp </property>       <!-- 20 -->
  <property> aero/coefficient/CYr </property>       <!-- 21 -->
  <property> aero/coefficient/CLwbh </property>     <!-- 22 -->
  <property> aero/coefficient/CLDf </property>      <!-- 23 -->
  <property> aero/coefficient/CLDe </property>      <!-- 24 -->
  <property> aero/coefficient/CLadot </property>    <!-- 25 -->
  <property> aero/coefficient/CLq </property>       <!-- 26 -->
  <property> aero/coefficient/Clb </property>       <!-- 27 -->
  <property> aero/coefficient/Clp </property>       <!-- 28 -->
  <property> aero/coefficient/Clr </property>       <!-- 29 -->
  <property> aero/coefficient/Clda </property>      <!-- 30 -->
  <property> aero/coefficient/Cldr </property>      <!-- 31 -->
```

```
  <property> aero/coefficient/Cmo </property>       <!-- 32 -->
  <property> aero/coefficient/Cmalpha </property>   <!-- 33 -->
  <property> aero/coefficient/CmDf </property>      <!-- 34 -->
  <property> aero/coefficient/Cmq </property>       <!-- 35 -->
  <property> aero/coefficient/Cmadot </property>    <!-- 36 -->
  <property> aero/coefficient/Cmde </property>      <!-- 37 -->
  <property> aero/coefficient/Cnb </property>       <!-- 38 -->
  <property> aero/coefficient/Cnp </property>       <!-- 39 -->
  <property> aero/coefficient/Cnr </property>       <!-- 40 -->
  <property> aero/coefficient/Cnda </property>      <!-- 41 -->
  <property> aero/coefficient/Cndr </property>      <!-- 42 -->
</output>

<output name="c172_position.csv" rate="60" type="CSV">
  <property> position/h-sl-ft </property>           <!--  2 -->
  <property> position/h-sl-meters </property>       <!--  3 -->
  <property> position/h-agl-ft </property>          <!--  4 -->
  <property> position/lat-geod-rad </property>      <!--  5 -->
  <property> position/lat-geod-deg </property>      <!--  6 -->
  <property> position/geod-alt-ft </property>       <!--  7 -->
  <property> position/lat-gc-deg </property>        <!--  8 -->
  <property> position/long-gc-deg </property>       <!--  9 -->
  <property>
    position/terrain-elevation-asl-ft </property>   <!-- 10 -->
  <property>
    position/distance-from-start-lon-mt </property> <!-- 11 -->
  <property>
    position/distance-from-start-lat-mt </property> <!-- 12 -->
  <property>
    position/distance-from-start-mag-mt </property> <!-- 13 -->
  <position> ON </position>
</output>

<output name="c172_velocities.csv" rate="60" type="CSV">
  <property> velocities/u-fps </property>           <!--  2 -->
  <property> velocities/v-fps </property>           <!--  3 -->
  <property> velocities/w-fps </property>           <!--  4 -->
  <property> velocities/p-rad_sec </property>       <!--  5 -->
  <property> velocities/q-rad_sec </property>       <!--  6 -->
  <property> velocities/r-rad_sec </property>       <!--  7 -->
  <property> velocities/p-aero-rad_sec </property>  <!--  8 -->
  <property> velocities/q-aero-rad_sec </property>  <!--  9 -->
  <property> velocities/r-aero-rad_sec </property>  <!-- 10 -->
  <property> velocities/phidot-rad_sec </property>  <!-- 11 -->
  <property> velocities/thetadot-rad_sec </property> <!-- 12 -->
  <property> velocities/psidot-rad_sec </property>  <!-- 13 -->
  <property> velocities/u-aero-fps </property>      <!-- 14 -->
  <property> velocities/v-aero-fps </property>      <!-- 15 -->
  <property> velocities/w-aero-fps </property>      <!-- 16 -->
  <property> velocities/vc-fps </property>          <!-- 17 -->
  <property> velocities/vt-fps </property>          <!-- 18 -->
  <property> velocities/mach </property>            <!-- 19 -->
</output>
```

```
<output name="c172_accelerations.csv" rate="60" type="CSV">
  <property> accelerations/pdot-rad_sec2 </property>  <!--  2 -->
  <property> accelerations/qdot-rad_sec2 </property>  <!--  3 -->
  <property> accelerations/rdot-rad_sec2 </property>  <!--  4 -->
  <property> accelerations/udot-ft_sec2 </property>   <!--  5 -->
  <property> accelerations/vdot-ft_sec2 </property>   <!--  6 -->
  <property> accelerations/wdot-ft_sec2 </property>   <!--  7 -->
  <property>
    accelerations/a-pilot-x-ft_sec2 </property>       <!--  8 -->
  <property>
    accelerations/a-pilot-y-ft_sec2 </property>       <!--  9 -->
  <property>
    accelerations/a-pilot-z-ft_sec2 </property>       <!-- 10 -->
  <property> accelerations/n-pilot-x-norm </property> <!-- 11 -->
  <property> accelerations/n-pilot-y-norm </property> <!-- 12 -->
  <property> accelerations/n-pilot-z-norm </property> <!-- 13 -->
</output>

<output name="c172_attitude.csv" rate="60" type="CSV">
  <property> attitude/phi-rad </property>           <!--  2 -->
  <property> attitude/theta-rad </property>         <!--  3 -->
  <property> attitude/psi-rad </property>           <!--  4 -->
  <property> attitude/roll-rad </property>          <!--  5 -->
  <property> attitude/pitch-rad </property>         <!--  6 -->
  <property> attitude/heading-true-rad </property>  <!--  7 -->
</output>

<!--
<output name="c172_animation.csv" rate="60" type="CSV">
  <property> position/lat-gc-deg </property>        <!--  2 -->
  <property> position/long-gc-deg </property>       <!--  3 -->
  <property> position/h-agl-ft </property>          <!--  4 -->
  <property> attitude/Phi </property>               <!--  5 -->
  <property> attitude/Theta </property>             <!--  6 -->
  <property> attitude/Psi </property>               <!--  7 -->
  <property> fcs/aileron-cmd-norm </property>       <!--  8 -->
  <property> fcs/elevator-cmd-norm </property>      <!--  9 -->
  <property> fcs/rudder-cmd-norm </property>        <!-- 10 -->
</output>
-->

<output name="c172_propulsion.csv" rate="60" type="CSV">
  <property> propulsion/tank/contents-lbs </property>  <!--  2 -->
  <property>
    propulsion/tank[1]/contents-lbs </property>       <!--  3 -->
  <property> propulsion/starter_cmd </property>       <!--  4 -->
  <property> propulsion/magneto_cmd </property>       <!--  5 -->
  <property> propulsion/active_engine </property>     <!--  6 -->
  <property> propulsion/total-fuel-lbs </property>    <!--  7 -->
  <property>
    propulsion/engine/pitch-angle-rad </property>     <!--  8 -->
  <property>
    propulsion/engine/yaw-angle-rad </property>       <!--  9 -->
  <property>
```

```
    propulsion/engine/advance-ratio </property>         <!-- 10 -->
  <property>
    propulsion/engine/blade-angle </property>           <!-- 11 -->
  <property>
    propulsion/engine/thrust-coefficient </property>    <!-- 12 -->
  <property>
    propulsion/engine/propeller-rpm </property>         <!-- 13 -->
  <property>
    propulsion/engine/helical-tip-Mach </property>      <!-- 14 -->
  <property>
    propulsion/engine/constant-speed-mode </property>   <!-- 15 -->
  <property>
    propulsion/engine/prop-induced-velocity_fps
      </property>                                       <!-- 16 -->
  <property>
    propulsion/engine/thrust-lbs </property>            <!-- 17 -->
  <property>
    propulsion/engine/fuel-flow-rate-pps  </property>   <!-- 18 -->
  <property>
    propulsion/engine/power-hp </property>              <!-- 19 -->
  <property>
    propulsion/engine/bsfc-lbs_hphr </property>         <!-- 20 -->
  <property>
    propulsion/engine/volumetric-efficiency
      </property>                                       <!-- 21 -->
  <property>
    propulsion/engine/map-pa </property>                <!-- 22 -->
  <property>
    propulsion/engine/map-inhg </property>              <!-- 23 -->
  <property>
    propulsion/engine/air-intake-impedance-factor
      </property>                                       <!-- 24 -->
  <property>
    propulsion/engine/ram-air-factor </property>        <!-- 25 -->
  <property>
    propulsion/engine/boost-speed </property>           <!-- 26 -->
  <property>
    fcs/throttle-cmd-norm </property>                   <!-- 27 -->
  <property>
    fcs/throttle-pos-norm </property>                   <!-- 28 -->
</output>

<output name="c172_standard.csv" rate="60" type="CSV">
  <position> ON </position>
  <velocities> ON </velocities>
  <rates> OFF </rates>
  <fcs>ON</fcs>
</output>
```

Con le configurazioni precedenti si potrà aprire Matlab e mandare in esecuzione il seguente programma di importazione dei dati e disegno delle storie temporali:

```
%---------------------------------------------------------------------
% Dinamica e Simulazione di Volo - Coiro, De Marco
%---------------------------------------------------------------------
% Plot JSBSim output data
% Authors: A. De Marco, Jary D'Auria
```

```
%-------------------------------------------------------------------------
% myplot.m importa i dati da i file .CSV, output della simulazione del
% velivolo C172x effettuata con JSBSim.
%-------------------------------------------------------------------------
% Assicurarsi di avere nella stessa cartella di questo script anche la
% funzione importJSBSimOutTSV.m
%-------------------------------------------------------------------------

% Begin script: cleaning stuffs
% tic
clear all; close all; clc;

%-------------------------------------------------------------------------
% Plot Menu: flaggare con 1 i plot desiderati

fcs_flag =              1; % Storia temporale dei comandi
forces_flag =           1; % Forze e momenti
aero_flag =             1; % Coefficienti Aerodinamici
position_flag =         1; % Dati Posizione
velocities_flag =       1; % Dati Velocità
accelerations_flag =    1; % Dati Accelerazioni
attitude_flag =         1; % Dati Orientamento
propulsion_flag =       1; % Dati Propulsione
summary_flag =          1; % Sommario Dati Importanti

%-------------------------------------------------------------------------
% Definizioni Costanti Aereo: C172
Sw = 174.0; % ft^2
c = 4.9;    % ft
b = 35.8;   % ft

%-------------------------------------------------------------------------
% Importazione dati fcs
d1 = importJSBSimOutTSV('c172_fcs.csv');

time = d1.data(:,1); % s
da_cmd_norm = d1.data(:,2);
de_cmd_norm = d1.data(:,3);
dr_cmd_norm = d1.data(:,4);
df_cmd_norm = d1.data(:,5);

de_pos_rad = d1.data(:,6); % rad
de_pos_deg = d1.data(:,7); % deg
de_pos_norm = d1.data(:,8);

dr_pos_rad = d1.data(:,9);  % rad
dr_pos_deg = d1.data(:,10); % deg
dr_pos_norm = d1.data(:,11);

df_pos_rad = d1.data(:,12); % rad
df_pos_deg = d1.data(:,13); % deg
df_pos_norm = d1.data(:,14);

da_l_pos_rad = d1.data(:,15); % rad
da_l_pos_deg = d1.data(:,16); % deg
da_l_pos_norm = d1.data(:,17);

da_r_pos_rad = d1.data(:,18); % rad
da_r_pos_deg = d1.data(:,19);
da_r_pos_norm = d1.data(:,20); % deg

da_pos_rad = (da_r_pos_rad+da_l_pos_rad)./2; % rad
da_pos_deg = (da_r_pos_deg+da_l_pos_deg)/2; % deg
da_pos_norm = (da_r_pos_norm+da_l_pos_norm)./2;

dt_cmd_norm = d1.data(:,21);
dt_pos_norm = d1.data(:,22);

%*************************************************************************
% Plot dati fcs
if fcs_flag == 1
```

```
    figure(1); clf;
    subplot(2,1,1);
    plot( ...
        time, da_cmd_norm, ...
        time, de_cmd_norm, ...
        time, dr_cmd_norm, ...
        time, df_cmd_norm, ...
        time, dt_cmd_norm)
    xlabel('time (s)'); ylabel('Surfaces Commands Normalized');
    title('Commands Histories','FontSize',14');
    grid on;
    legend ('\delta_a_{ cmd}', '\delta_e_{ cmd}', '\delta_r_{ cmd}', ...
    '\delta_f_{ cmd}', '\delta_t_{ cmd}');
    %ylim([0 10000]);

    subplot(2,1,2);
    plot( ...
        time, de_pos_deg, ...
        time, dr_pos_deg, ...
        time, df_pos_deg, ...
        time, da_pos_deg)
    xlabel('time (s)'); ylabel('(deg)');
    grid on;
    legend ('\delta_e', '\delta_r', '\delta_f', '\delta_a');
    %ylim([-50 1250]);

end
%--------------------------------------------------------------------------
% Importazione dati forces
% ...
```

Questo programma è stato chiamato per semplicità `myplot.m`. Si lascia al lettore il compito di visionare il resto dei comandi contenuti in questo file. Tra gli altri, questo script produce i grafici riportati nelle figure 15.25, 15.26, 15.27, e 15.28.

Esercizio 15.3: *Storie temporali di una simulazione batch*

Effettuare una la simulazione batch come programmata in `c1723.xml` e tracciare i diagrammi di tutte le storie temporali prodotte dallo script `myplot.m`. Si ricordi di impostare i canali di output multipli come spiegato sopra.

❖

Esercizio 15.4: *Storie temporali di una simulazione batch*

Con riferimento all'esercizio 15.2 tracciare i diagrammi di tutte le storie temporali prodotte dallo script `myplot.m`.

❖

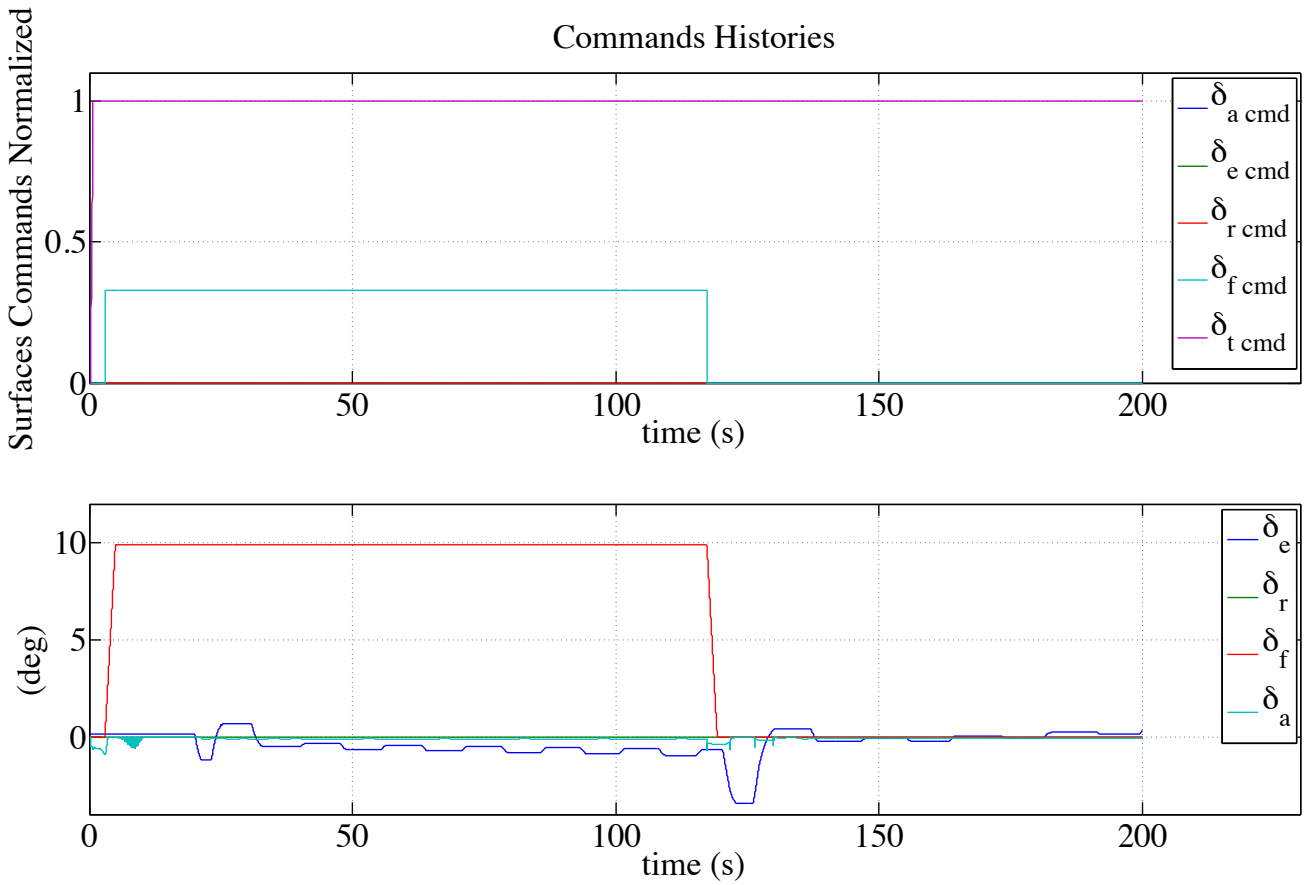


Figura 15.25 Storie temporali dei comandi.

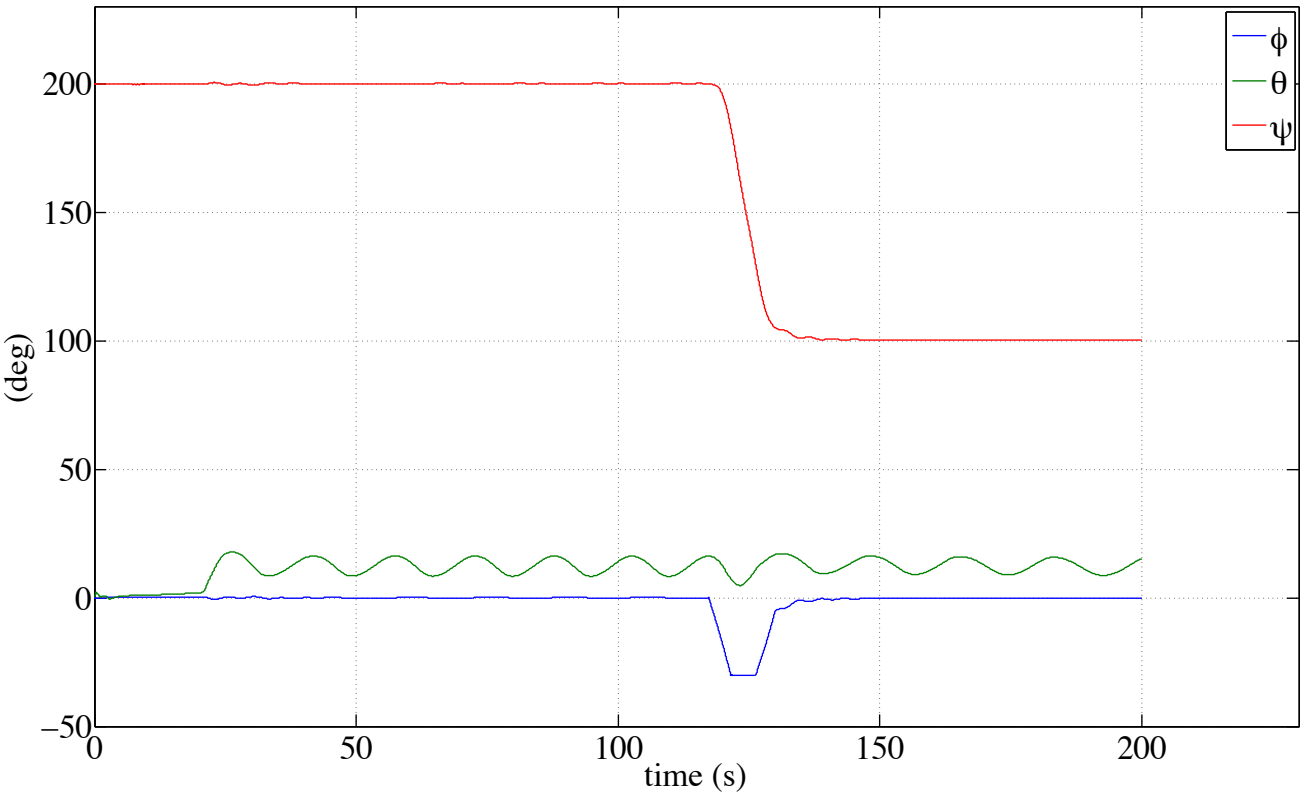


Figura 15.26 Storie temporali degli angoli di Eulero.

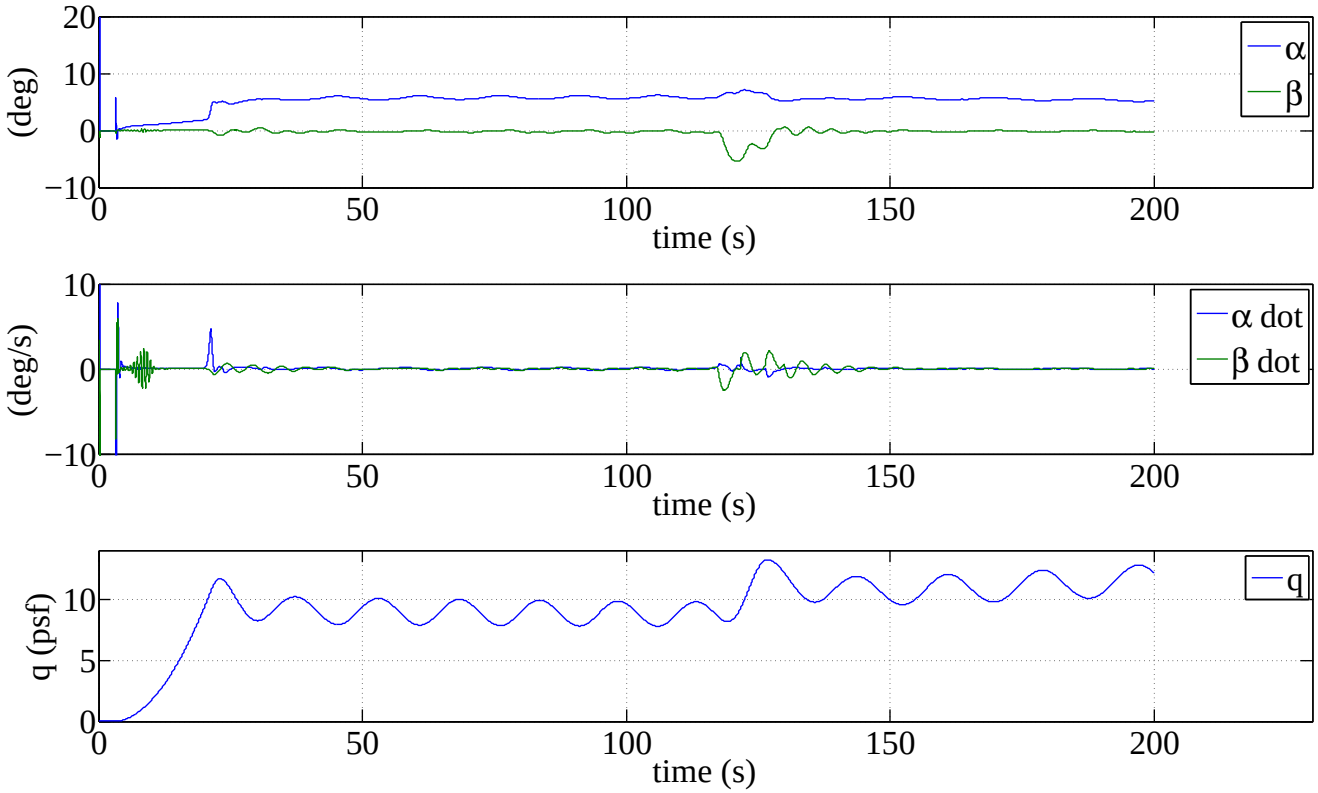


Figura 15.27 Storie temporali degli angoli aerodinamici.

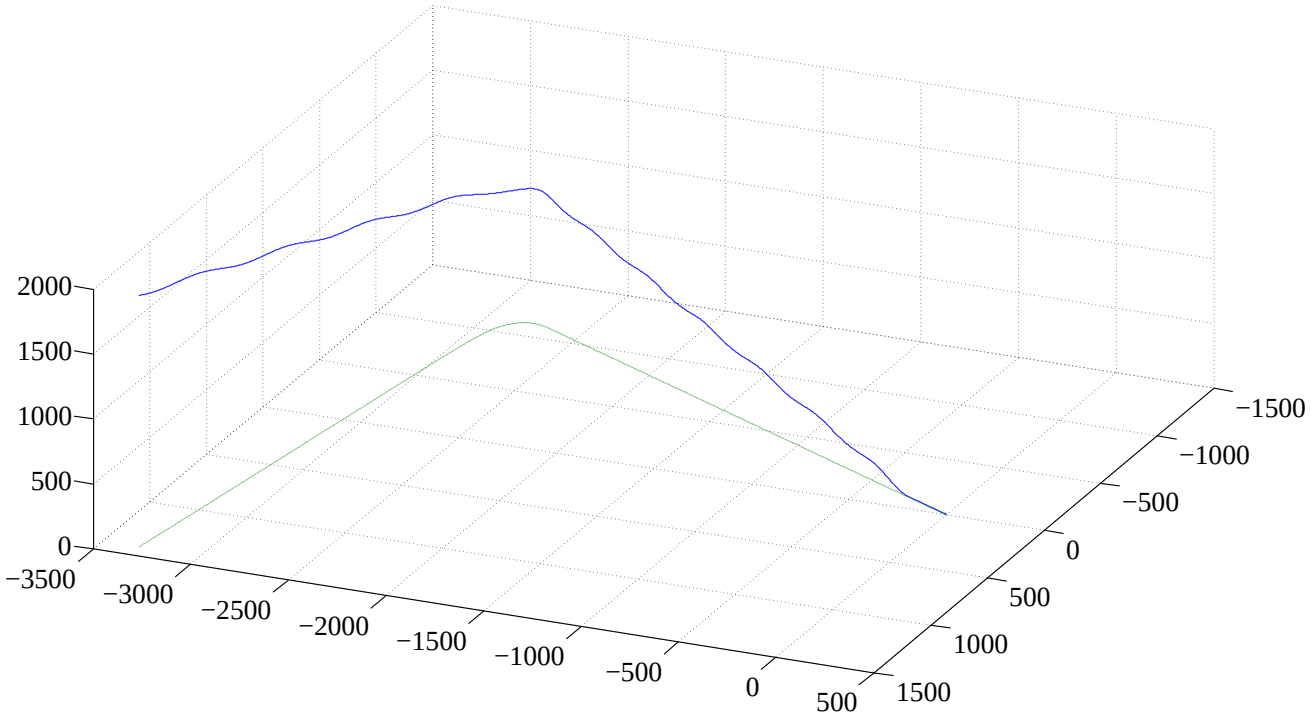


Figura 15.28 Traiettoria e *ground track*.

Bibliografia

[1] W. R. Hamilton, *Lectures on Quaternions*, Hodeges & Smith, 1853.

[2] O. Rodrigues, "Des lois géometriques qui régissent les désplacements d'un système solide dans l'espace, et de la variation des coordonnée provenant de ses désplacements considerées indépendamment des causes qui peuvent les produire", *Journal des Mathématiques Pures et Appliquées*, vol. 5, 1840.

[3] E. Salamin, "Application of Quaternions to Computation with Rotations", Working paper, Stanford AI Lab, 1979.

[4] A. P. Yefremov, "Quaternions: Algebra, Geometry and Physical Theories", *Hypercomplex Numbers in Geometry and Physics*, vol. 1, 2004.

[5] Schwab A. L., "Quaternions, Finite Rotations and Euler Parameters", Course notes on Applied Multibody Dynamics, Delft University of Technology, Laboratory for Engineering Mechanics, 2003.
http://tam.cornell.edu/~{}als93/quaternion.pdf.

[6] AIAA/ANSI, *Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*. R-004-1992, 1992.

[7] G. H. Bryan, *Stability in Aviation: An Introduction to Dynamical Stability as Applied to the Motions of Aeroplanes*. Macmillan and Co., Limited, London, 1911.

[8] D. J. Diston, *Computational Modelling of the Aircraft and the Environment. Volume 1, Platform Kinematics and Synthetic Environment*. John Wiley & Sons, Inc., 2009.

[9] W. F. Phillips, *Mechanics of Flight*. John Wiley & Sons, Inc., 2004.

[10] W. F. Phillips, "Phugoid Approximation for Conventional Airplanes", Journal of Aircraft, Vol. 37, No. 1, January-February 2000.

[11] W. F. Phillips, "Improved Closed-Form Approximation for Dutch-Roll", Journal of Aircraft, Vol. 37, No. 1, May-June 2000.

[12] R. Stengel, *Flight Dynamics*. Princeton University Press, Princeton, 2004.

[13] M. R. Napolitano, *Aircraft Dynamics: From Modeling to Simulation*. John Wiley, 2012.

[14] D. K. Schmidt, *Modern Flight Dynamics*. McGraw-Hill, 2010.

[15] B. Stevens, F. Lewis, *Aircraft Control and Simulation*. John Wiley & Sons, Inc., 1992.

[16] D. Stinton, *The Anatomy of the Airplane* (2nd edition). American Institute of Aeronautics and Astronautics, 1998.

[17] B. Etkin, *Dynamics of Flight, Stability and Control*. John Wiley & Sons, New York, 1982.

[18] M. Calcara, *Elementi di dinamica del velivolo*. Edizioni CUEN, Napoli, 1988.

[19] L. V. Schmidt, *Introduction to Aircraft Flight Dynamics*. AIAA Education Series, 1998.

[20] W. J. Duncan, *Control and Stability of Aircraft*. Cambridge University Press, Cambridge, 1952.

[21] R. Jategaonkar, *Flight Vehicle System Identification: A Time Domain Methodology*. Progress in Astronautics and Aeronautics Series, 2006.

[22] C. D. Perkins, R. E. Hage, *Aircraft Performance, Stability and Control*. John Wiley & Sons, New York, 1949.

[23] J. R. Wright, J.. E. Cooper, *Introduction to Aircraft Aeroelasticity and Loads*. John Wiley & Sons, Inc., 2007.

[24] V. Losito, *Fondamenti di Aeronautica Generale*. Accademia Aeronautica, Napoli, 1994.

[25] E. Torenbeek, H. Wittenberg, *Flight Physics*. Springer, Heidelberg, 2009.

[26] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition. AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA. 2007.

[27] J. D. Mattingly, *Elements of Propulsion: Gas Turbines and Rockets*. AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA. 2006.

[28] K. Hünecke, *Jet Engines. Fundamentals of Theory, Design and Operation*. Motorbooks International, 1997.

[29] A. Linke-Diesinger, *Systems of Commercial Turbofan Engines*. Springer-Verlag, Berlin Heidelberg, 2008.

[30] F. R. Garza, E. A. Morelli, "A Collection of Nonlinear Aircraft Simulations with MATLAB". NASA-TM-2003-212145, January 2003.

[31] Voce WGS84 su *Wikipedia*:
http://en.wikipedia.org/wiki/World_Geodetic_System

[32] Anonimo, *Department of Defense World Geodetic System 1984. Its Definition and Relationship with Local Geodetic Systems*. NIMA TR8350.2, Third Edition, Amendment 2. National Imagery and Mapping Agency, US Department of Defense, 2004.

[33] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls*. DARcorporation, 2001.

[34] H. T. Schlichting, E. A. Truckenbrodt, *Aerodynamics of the Aeroplane*. McGraw Hill Higher Education, 2nd edition, 1979.

[35] M. M. Munk, "The aerodynamic forces on airship hulls". NACA-TR-184, 1924.

[36] A. Silverstein, S. Katzoff, "Aerodynamic characteristics of horizontal tail surfaces". NACA-TR-688, 1940.

[37] R. I. Sears, "Wind-tunnel data on the aerodynamic characteristics of airplane control surfaces". NACA-WR-L-663, 1943.

[38] E. Garner, "Wind-tunnel investigation of control-surface characteristics XX: plain and balanced flaps on an NACA 0009 rectangular semispan tail surface". NACA-WR-L-186, 1944.

[39] J. D. Brewer, M. J. Queijo, "Wind-tunnel investigation of the effect of tab balance on tab and control-surface characteristics". NACA-TN-1403, 1947.

[40] S. M. Crandall, H. E. Murray, "Analysis of available data on the effects of tabs on control-surface hinge moments". NACA-TN-1049, 1946.

[41] B. W. McCormick, *Aerodynamics, Aeronautics, and Flight Mechanics*. John Wiley & Sons, 1979.

[42] B. N. Pamadi, *Performance, Stability, Dynamics and Control of Airplanes*. AIAA Education Series, 1998.

[43] A. Tewari, *Atmospheric and Space Flight Dynamics. Modelling and Simulation with Matlab and Simulink*. Birkhäuser, Berlin, 2007.

[44] D. Howe, *Aircraft Loading and Structural Layout*. AIAA Education Series, 2004.

[45] P. Morelli, *Static Stability and Control of Sailplanes*. Levrotto & Bella, Torino, 1976.

[46] L. Prandtl, O. G. Tietjens, *Fundamentals of Hydro and Aeromechanics*. Dover, 1957.

[47] R. K. Heffley, W. F. Jewell, "Aircraft Handling Qualities Data". NASA-CR-2144, December 1972.

[48] H. P. Stough III, J. M. Patton Jr, S. M. SliWa, "Flight Investigation of the Effect of Tail Configuration on Stall, Spin, and Recovery Characteristics of a Low-Wing General Aviation Research Airplane". NASA-TP-1987-2644, February 1987.

[49] J. D. Anderson, *Fundamentals of Aerodynamics*. McGraw-Hill, 3rd edition, New York, 2001.

[50] J. J. Bertin, *Aerodynamics for Engineers*. Prentice-Hall, 4th edition, Upper Saddle River, NJ, 2002.

[51] J. Katz, A. Plotkin, *Low-Speed Aerodynamics*. Cambridge University Press, 2nd edition, Cambridge, England, U.K., 2001.

[52] D. E. Hoak, *et al.*, "The USAF Stability and Control Datcom". Air Force Wright Aeronautical Laboratories, TR-83-3048, 1960 (Revised 1978).

[53] R. T. Jones, "A Note on the Stability and Control of Tailless Airplanes". NACA Report 837, 1941.

[54] D. P. Coiro, F. Nicolosi, A. De Marco, N. Genito, S. Figliolia, "Design of a Low Cost Easy-to-Fly STOL Ultralight Aircraft in Composite Material". *Acta Polytecnica*, Vol. 45 no. 4, 2005, pp. 73-80; ISSN 1210-2709.

[55] F. Nicolosi, A. De Marco, P. Della Vecchia, "Flight Tests, Performances and Flight Certification of a Twin-Engine Light Aircraft". *Journal of Aircraft*, Vol 48, No. 1, January-February 2011.

[56] F. Nicolosi, A. De Marco, P. Della Vecchia, "Parameter Estimation and Flying Qualities of a Twin-Engine CS23/FAR23 Certified Light Aircraft". AIAA-2010-7947, AIAA Atmospheric Flight Mechanics Conference, Toronto, 2010.

[57] B. Etkin, *Dynamics of Atmospheric Flight*, Dover Publications, 2005.

[58] L. Mangiacasale, *Flight Mechanics of a μ-Airplane*, Edizioni Libreria CLUP, Milano, 1998.

[59] G. Mengali, *Elementi di Dinamica del Volo con Matlab*, Edizioni ETS, Pisa, 2001.

[60] R. Nelson, *Flight Stability and Automatic Control*, McGraw-Hill, 1989.

[61] Y. Li, M. Nahon, "Modeling and simulations of airship dynamics", *Journal of Guidance, Controls and Dynamics*, Vol 30, No. 6, November-December 2007.

[62] Y. Fan, F. H. Lutze, E. M. Cliff, "Time-Optimal Lateral Maneuvers of an Aircraft", *Journal of Guidance, Controls and Dynamics*, Vol 18, No. 5, September-October 1995.

[63] J. N. Nielsen, *Missile Aerodynamics*, AIAA, Cambridge, MA, 1988.

[64] T. I. Fossen, *Guidance and Control of Ocean's Vehicles*, Whiley, New York, 1998.

[65] J. N. Newman, *Marine Hydrodynamics*, MIT Press, Cambridge, MA, 1977.

[66] E. L. Duke, R. F. Antoniewicz, K. D. Krambeer, "Derivation and Definition of a Linear Aircraft Model". Technical Report NASA Reference Publication RP-1207, Research Engineering, NASA Ames Research Center and NASA Dryden Flight Research Facility, 1988.

[67] G. A. Stagg, *An Unsteady Aerodynamic Model for Use in the High Angle of Attack Regime*. MS thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1998.

[68] Y. Fan, *Identification of an Unsteady Aerodynamic Model up to High Angle of Attack Regime*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1997.

[69] *MATLAB Users' Guide*. The Mathworks, 2003 ed edizioni successive.
http://www.mathworks.com/
http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html

[70] V. Comincioli, *Analisi numerica: metodi, modelli, applicazioni*. McGraw-Hill, 1990, seconda edizione 1995.

[71] E. Kreyszig, *Advanced Engineering Mathematics*. John Wiley & Sons, seventh edition, 1993.

[72] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag, 1978.

[73] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*. Cambridge University Press, 1992.

[74] G. Dahlquist, A. Bjorck, *Numerical Methods. Volume I: Fundamentals of Numerical Discretization*. John Wiley & Sons, 1988.

[75] R. D. Richtmyer, K. W. Morton, *Difference Methods for Initial Value Problems*. Wiley-Interscience, 1967.

[76] C. Hirsch, *Numerical Computation of Internal and External Flows*. John Wiley & Sons, 1994.

[77] R. D. Finck, "USAF Stability and Control Datcom". AFWAL-TR-83-3048, October 1960, Revised 1978.

[78] S. R. Vukelich, J. E. Williams, "The USAF Stability and Control Digital Datcom". AFFDL-TR-79-3032, Volume I, April 1979, Updated by Public Domain Aeronautical Software 1999.

[79] W. B. Blake, "Prediction of Fighter Aircraft Dynamic Derivatives Using Digital Datcom". AIAA-85-4070, AIAA Applied Aerodynamics Conference, Colorado Springs, Colorado, 1985.

[80] Autori Vari, Distribuzione ufficiale di Digital Datcom,
sito internet:
http://wpage.unina.it/agodemar/DSV-DQV/Digital-Datcom-Package.zip

[81] B. Galbraith, "Digital Datcom+", Holy Cows, Inc.,
sito internet: http://www.holycows.net/datcom/