ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA

SECONDA FACOLTA' DI INGEGNERIA
CON SEDE A CESENA

CORSO DI LAUREA
IN INGEGNERIA AEROSPAZIALE

Sede di Forlì

ELABORATO FINALE DI LAUREA
in Elaborazione Dati per la Navigazione

SVILUPPO DI UN MODELLO DI DRONE DI TIPO FLYING WINGS IN AMBIENTE FLIGHTGEAR INTERFACCIATO CON AUTOPILOTI DI ARDUINO IMPLEMENTATI IN SIMULINK

Candidato:
Matteo Metalli

Relatore:
Prof. Matteo Zanzi

Correlatore:
Ing. Niki Regina

Anno Accademico 2010-2011
Sessione III*a*

Indice

Indice delle figure

RINGRAZIAMENTI

Desidero innanzitutto ringraziare il Professore Matteo Zanzi per i suoi preziosi ed utili insegnamenti fondamentali per lo svolgimento di questa tesi, per la sua continua disponibilità e non da ultimo per la simpatia dimostrata.

In egual modo ringrazio l'Ingegnere Niki Regina che mi ha assistito e supportato in questo periodo di lavoro introducendomi all'utilizzo dei programmi necessari per la realizzazione di questo elaborato.

Un sincero ringraziamento va anche agli Ingegneri Antonio Ghetti e Alessandro Mirri per i loro utili consigli.

Desidero sottolineare la cordialità sempre dimostratami da tutti durante questo periodo di intenso lavoro.

Da ultimo un ringraziamento particolare alla mia famiglia che mi ha sempre sostenuto in questi anni di studio.

Capitolo 1

INTRODUZIONE

L'idea che sta alla base di questa tesi è riuscire a trovare un nuovo modo di interazione tra il simulatore di volo FlightGear e Simulink. Fino ad oggi, infatti, si ricostruiva in Simulink un sistema di equazioni che rappresentasse il "sistema aereo" e poi si sfruttava FlightGear solo esclusivamente come visualizzatore in modo da avere un riscontro più diretto e intuitivo del comportamento dell'aereo.

Qui invece si vuole sfruttare FlightGear nella sua totalità di simulatore, creando un nuovo tipo di collegamento FlightGear – Simulink con il quale sia possibile governare un aereo all'interno di FlightGear direttamente da Simulink.

Si andrà quindi a riprodurre all'interno di FlightGear un piccolo drone, in dotazione all'università, e si userà Simulink esclusivamente per controllarlo attraverso l'implementazione degli autopiloti presenti su di esso.

La tesi è organizzata nel seguente modo.

Nel Capitolo 1 si ha una breve introduzione generale sui due principali programmi usati FlightGear e Matlab – Simulink.

Il Capitolo 2 è dedicato alla realizzazione del collegamento FlightGear – Simulink con particolare attenzione per il Paragrafo 2.3 nel quale si spiega la principale funzione che consente questo tipo di collegamento.

Nel Capitolo 3 si spiega tutto il processo di modellizzazione del drone.

Il Capitolo 4 è dedicato all'implementazione degli autopiloti presenti sul drone in Simulink.

Nel Capitolo 5 vengono riportati i risultati delle simulazioni effettuate.

1.1 FLIGHTGEAR

FlightGear Flight Simulator (o semplicemente FlightGear) è un progetto collaborativo con lo scopo di creare un sofisticato framework di simulatore di volo. È disponibile, sotto licenza GNU GPL (licenza per software libero), per vari sistemi operativi tra cui Microsoft Windows, Linux, Mac OS X, Unix. Può essere utilizzato sia come applicazione per un utente finale sia come ricerca in ambito accademico al fine di provare e sviluppare soluzioni per la simulazione di volo.

La personalizzazione di FlightGear la si può notare nei vari tipi di aerei che sono disponibili, dall'aliante all'elicottero, dall'aereo di linea a quelli da combattimento. Tutti questi modelli sono stati creati da svariate persone. Per creare questi modelli si possono usare diversi FDM (Flight Dynamics Model). Un FDM è un insieme di equazioni matematiche che sono utilizzate per calcolare le forze fisiche che agiscono sull'aereo simulato come la spinta, la portanza e la resistenza. Ogni aereo che viene simulato in FlightGear deve usare uno dei seguenti FDM: LaRCsim, JSBSim, UIUC, YASim.

1.2 MATLAB-SIMULINK

Matlab (abbreviazione per Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato da MathWorks. Matlab consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente e interfacciarsi con altri programmi. È usato da milioni di persone nell'industria e nelle università e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, GNU7Linux e Unix.

Simulink è un software integrato in Matlab ideato per modellare, simulare e analizzare sistemi dinamici di qualsiasi tipo (lineari e non lineari). La sua interfaccia grafica principale è uno strumento grafico di diagrammi a blocchi, raggruppati in librerie, risultando così molto intuitiva. Esso offre una stretta integrazione con il resto dell'ambiente Matlab e può o far funzionare Matlab o essere programmato da esso.

Capitolo 2

COLLEGAMENTO FLIGHTGEAR-SIMULIK

Come idea di base il collegamento tra i due programmi è abbastanza semplice, poiché si dovranno ricevere, in tempo reale, in Simulink i dati di volo del velivolo provenienti da FlightGear e nello stesso tempo inviare quelli che sono necessari per manovrare le superfici di comando. È evidente come si vada a sfruttare FlightGear nella sua totalità poiché è esso che fornisce il modello di equazioni che descrivono il moto dell'aereo in quanto Simulink si limita solo ad agire sui comandi di volo.
Per quanto riguarda il lato pratico il collegamento risulta abbastanza complesso, infatti per realizzarlo si è dovuti procedere per passi in modo da rendere più semplice la costruzione del collegamento nella sua interezza. Si è partiti utilizzando delle funzioni in Matlab per arrivare alla definitiva creazione di un blocco in Simulink che svolgesse il collegamento tra i due programmi.

2.1 PARAMETRI

Prima di procedere a creare il collegamento si sono selezionati i parametri principali di volo che sono necessari per conoscere l'esatta posizione e assetto dell'aereo e quelli che consentono di manovrarlo. Tra i numerosi parametri che FlightGear può fornire, quelli che si è ritenuto più utili allo scopo sono: la latitudine, la longitudine, l'altezza, l'angolo di rollio, l'angolo di beccheggio, la prua e la velocità di volo (sia rispetto all'aria che al suolo) del velivolo che viene simulato. Per il controllo dell'aereo si è scelto di agire sulle principali superfici di comando e sulla potenza del motore andando a controllare i seguenti parametri: l'angolo di alettone, l'angolo di elevatore, l'angolo del timone e la posizione della manetta motore.

Per far in modo che i dati che si ricevono e si immettono in FlightGear siano effettivamente quelli sopra citati è necessario creare un file xml (ardupilot.xml) nel quale si andranno a specificare esattamente i parameri sopra indicati che poi dovranno essere inseriti nella cartella "protocol" di FlightGear insieme agli altri file preesistenti.

2.2 SPECIFICHE COLLEGAMENTO

Il collegamento tra i due programmi avviene attraverso una porta udp sia per i dati in uscita da FlightGear sia per quelli in ingresso. Affinché questo sia possibile si dovrà far avviare FlightGear attraverso un batch file (prova.bat) nel quale viene specificato il tipo di collegamento da usare (in particolare il numero della porta udp di ingresso e uscita di FlightGear), il file protocol che dovrà essere utilizzato da FlightGear durante la simulazione, il tipo di aereo utilizzato e in più può essere anche specificata la sua posizione iniziale (luogo di partenza, altitudine, prua).

2.3 JUDP

Si passa ora a vedere come con Simulink si è in grado di comunicare direttamente con FlightGear utilizzando dei blocchi già esistenti al suo interno. Purtroppo non si sono trovati blocchi utili a questo scopo, in quanto quelli già esistenti che sono in grado di comunicare attraverso una porta udp lo fanno solo se il collegamento avviene ad un altro computer e non in grado quindi di comunicare con un programma presente all'interno dello stesso, come si richiede in questo caso.

Si è deciso quindi di procedere per gradi cercando in un primo momento di ricevere dei dati da FlightGear in Matlab e poi vedere come è possibile creare un blocco in Simulink con le stesse funzioni.

Si è quindi pensato di costruire una function in Matlab utile allo scopo, ma si è visto che la sua realizzazione era molto complessa e laboriosa. Quindi si è deciso di andare a vedere se sul sito www.mathworks.com fosse già presente una function che rispondesse a queste esigenze. Dopo una piccola ricerca si è trovato la function "judp" la quale permette di comunicare con

programmi all'interno dello stesso computer attraverso un collegamento udp, proprio come si cercava. Una volta aperta, dopo averla scaricata, si è visto che, subito all'inizio, viene riportato in verde un commento nel quale si spiega come è possibile utilizzare tale function. Il commento della function judp viene riportato in figura 1.

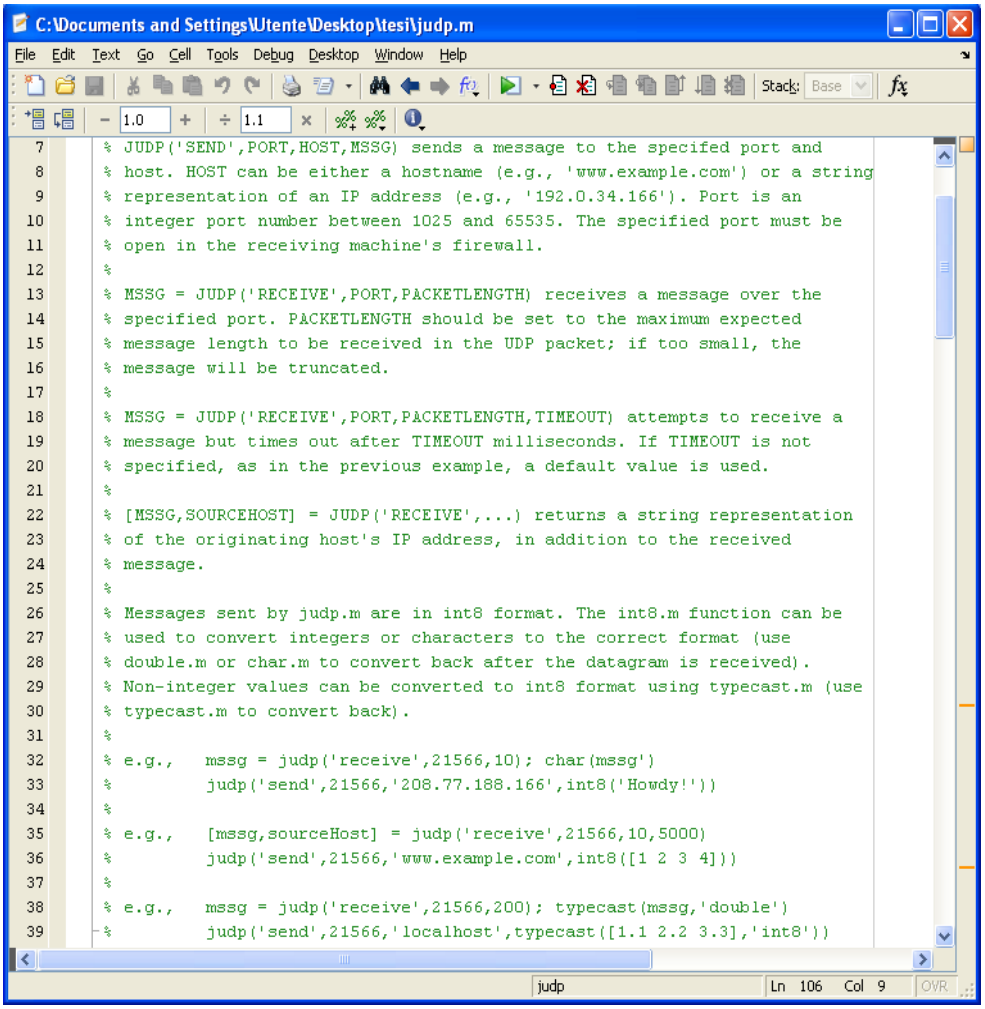

Figura 1

Come prima cosa si è cercato di ricevere dei dati da FlightGear utilizzando il comando, da digitare nella command window di Matlab, relativo al primo esempio.

Prima però di provare il suo funzionamento si deve avviare FlightGear attraverso un batch file che per questa prima prova è "simulazione.bat", nel quale si usa la porta udp di uscita 49005 e come aereo un cessna c172p ad una quota di 2000ft sull'aeroporto di Forlì.

Avviato FlightGear si digita in Matlab “mssg=judp(“receive”,49005,1000)” e, una volta spinto invio, si ricevono dei numeri in colonna. Tali numeri si riferiscono, rispettivamente, ai parametri elencati nel file “ardupilot” che si è creato. In particolare i numeri che si sono ricevuti si riferiscono alla latitudine, longitudine, altitudine, angolo di rollio, angolo di beccheggio, prua e alla velocità. Nella figura 2 sono riportati i numeri che si ricevono.

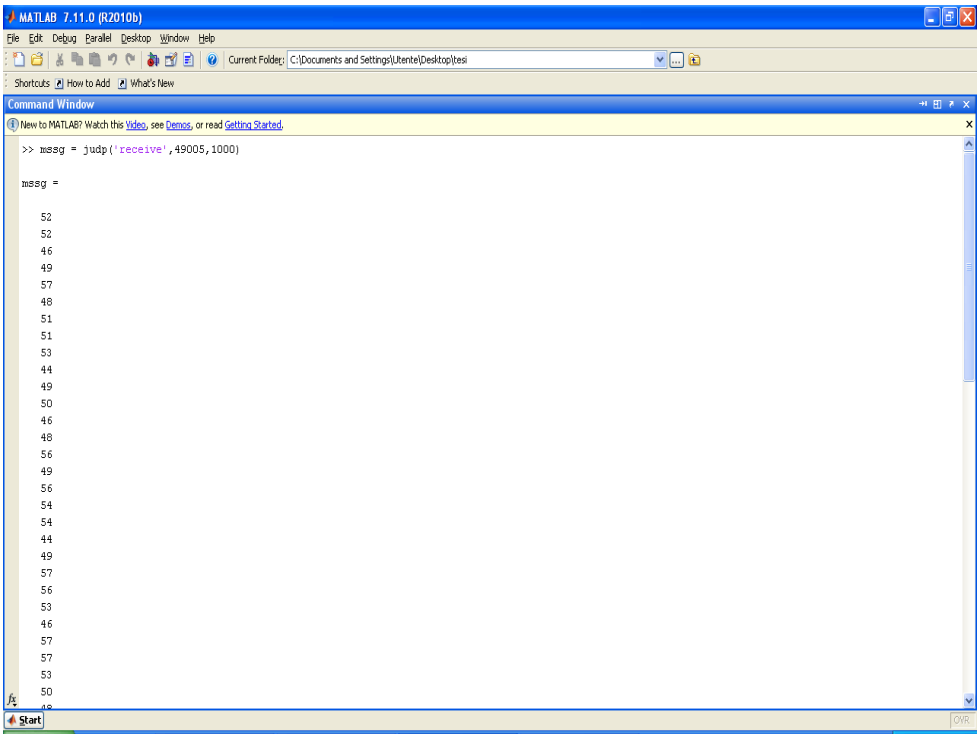

Figura 2

Dopo un’attenta analisi di questi numeri si è visto che essi esprimono i dati ricevuti, nell’istante in cui si è spinto invio, da FlightGear in codice ASCII. Ora il passo successivo è quello di riuscire a trasformare questi dati scritti in codice ASCII in numeri che rappresentano effettivamente i parametri che ci interessano.

Per fare questa operazione si possono sfruttare delle function (già presenti in Matlab) che, usate in una sequenza ben specifica ci restituiscono proprio i numeri, e quindi i valori, delle grandezze elencate nel file ardupilot. Si sono usate le function char, textscan, str2double utilizzate in questa esatta successione. Per leggere direttamente i valori associati alle rispettive grandezze, senza ripetere ogni volta tutti i comandi, si è creato una function

apposita (fg1.m) nella quale sono riportati tutti i passaggi necessari per ottenere il risultato desiderato. Tale function richiede in ingresso il numero della porta udp attraverso la quale avviene la ricezione dei dati. Digitando [lat long h roll pitch heading v]=fg1(49005), una volta dato l'invio, si ottengono i risultati visibili nella figura 3.

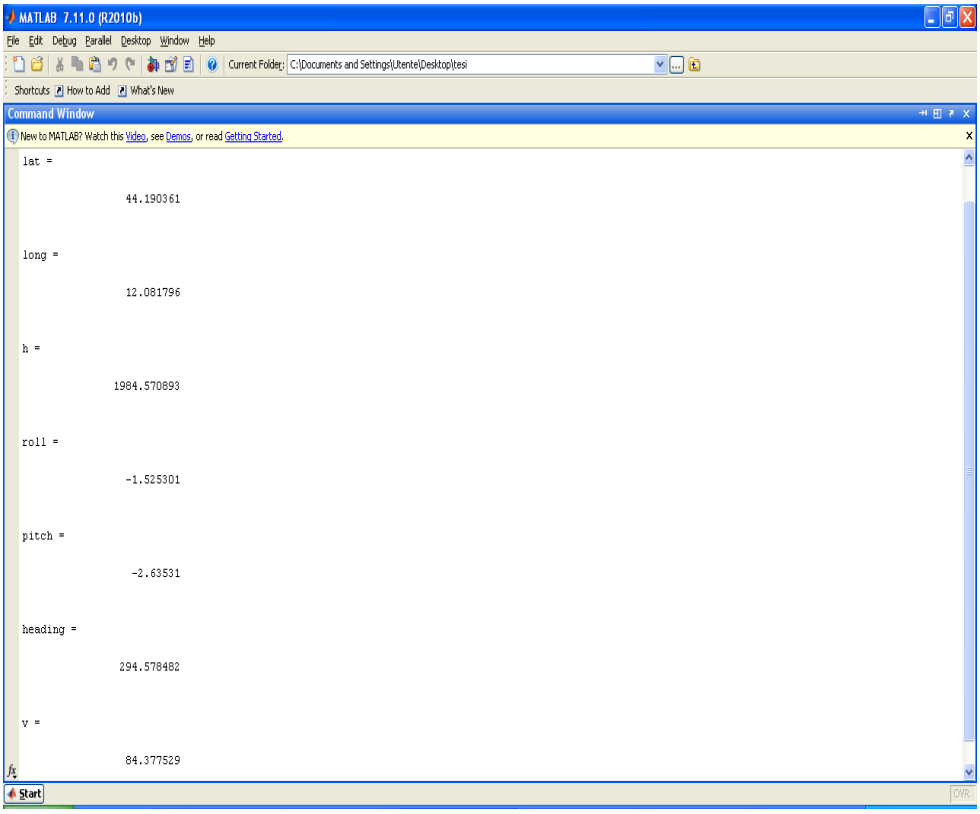

Figura 3

Il primo passo, cioè riuscire a leggere in Matlab i dati provenienti da FlightGear , è stato fatto. Il passo successivo è riuscire a riportare il tutto in Simulink. Chiaramente in Matlab si riescono a leggere i dati di un solo istante, mentre in Simulink il blocco che implementerà la funzione la eseguirà ogni istante della simulazione ricevendo così i dati istante per istante. Per prima cosa si è andati a vedere se in Simulink è già presente un blocco capace di implementare direttamente la function fg1 appena creata. Purtroppo non è possibile associare ad un blocco la fg1 così com'è, ma si deve creare una S-function analoga in modo da riuscire ad usare il blocco Level-2 MATLAB S-function che è l'unica possibilità per riuscire nello

scopo prefissato. Il vantaggio di usare una S-function è che si può riportare lo script della fg1 al suo interno, però purtroppo questa ha una costruzione più complessa che richiede un linguaggio per la costruzione.

2.4 S-FUNCTION

Una Level-2 S-function è una funzione che può essere scritta in vari linguaggi tra cui C, C++ e ovviamente l'M (linguaggio di Matlab). Ha comandi ben precisi che consentono la sua costruzione e permette di creare blocchi in Simulink con ingressi e uscite multipli in grado di gestire qualsiasi tipo di segnale.

2.4.1 STRUTTURA

Questo tipo di S-function ha la struttura formata da varie function:

- Function nome(block)

  Ha lo scopo di dare il nome alla S-function e di conseguenza al blocco ed formata comando setup (block) che avvia la corrispettiva Function.

- Function setup(block)

  Qui si settano tutti i parametri del blocco a cominciare dal numero degli ingressi: block.NumInputPorts = n;

  delle uscite:
  block.NumOutputPorts = m;

  la dimensione delle uscite e degli ingressi:
  block.InputPort (1) .Dimensions = 1;
  …
  block.InputPort (n) .Dimensions = 1;
  block.OutputPort (1) .Dimensions = 1;
  …
  block.OutputPort (m) .Dimensions = 1;

si richiama il block.RegBlockMethod, ne sono presenti più di uno e ad ognuno è associata una diversa function che saranno utili affinché il blocco esegua il compito per cui è stato pensato;

- Vanno poi inserite le varie function associate ad ogni block.RegBlockMethod. Tra le più importanti abbiamo la function InitConditions(block) nella quale si vanno a specificare le condizioni iniziali del blocco e la function Outputs(block) nella quale si andrà a specificare, attraverso una script, quelli che saranno i valori delle uscite.

2.5 BLOCCO LEVEL-2 MATLAB S-FUNCTION

Il blocco Level-2 MATLAB S-function consente di utilizzare una S-function in modo da dare al blocco stesso le proprietà e le funzioni specificate all'interno della S-function.

Quindi ora partendo dalla function fg1 si andrà a creare una S-function che svolga lo stesso compito così da essere accettata da riuscire a creare un blocco in Simulink.

Seguendo quello scritto in precedenza su come si costruisce una S-function si è creati prova.m che si è costruita andando ad inserire all'interno della function Outputs(block) praticamente lo stesso script della fg1.

Si inserisce le S-function prova.m all'interno del blocco Level-2 MATLAB S-function e si ottiene così un blocco, di nome prova, con un ingesso e sette uscite. In ingresso si ha la porta udp dalla quale si ricevono i dati da FlightGear e le uscite sono le grandezze specificate nel file ardupilot. Per completare lo schema in Simulink inseriamo un blocco costante, nel quale si specifica il numero della porta, e dei display per vedere i valori delle uscite. Si ottiene così lo schema a blocchi riportato nella seguente figura 4.

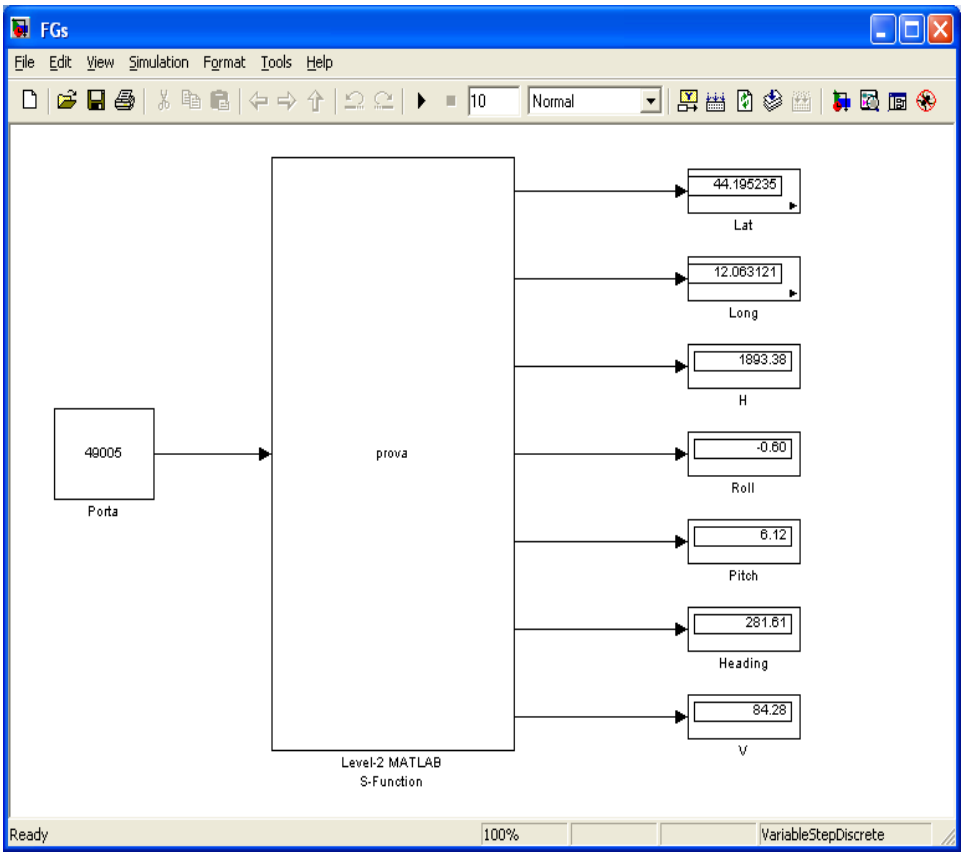

Figura 4

Se si avvia FlightGear e poi si fa partire la simulazione in Simulink cliccando sul tasto play, si vedono effettivamente che i numeri nel display variano continuamente nel tempo a seconda del comportamento del velivolo e della rotta che sta seguendo.

Il passo successivo è di riuscire, oltre che a ricevere, ad inviare dei dati a FlightGear che andranno a controllare le superfici di comando dell'aereo.

2.6 REALIZZAZIONE COLLEGAMENTO

Dal modo in cui può essere usata la function judp si è visto che questa consente l'invio di dati ad un altro programma usando, invece della modalità receive, la send digitando judp('send', 49000, '127.0.0.1', int8([1 2 3 4])). Si è pensato di andare a creare subito una S-function che fosse in grado di inviare e ricevere dati da FlightGear . Una cosa importante da ricordare è che i dati che si andranno ad inviare a FlightGear si riferiranno,

rispettivamente, alle grandezze elencate all'interno del file protocol ardupilot.

Si è pensato che il messaggio che si andrà ad inviare attraverso la judp sarà sintatticamente uguale a quello che si è ricevuto e andrà inserito al posto di int8([1 2 3 4]).

Quindi ora si tratta di trasformare i valori che si vogliono attribuire allo spostamento delle varie superfici di comando in un messaggio, in codice ASCII, simile a quello che si è ricevuto precedentemente da FlightGear .

Per fare ciò possiamo sfruttare delle function già presenti all'interno di Matlab che eseguite in un determinato ordine consentono di ottenere il corretto messaggio da inviare tramite la judp. Tali function sono num2str, double, int8.

Ora si dovrà inserire questi comandi, insieme a quelli precedenti usati per ricevere i dati, all'interno della function Outputs(block) di una nuova S-function iprovau.m che andrà a formare un nuovo blocco in Simulink che ha come ingressi lo spostamento delle tre superfici di comando più la posizione della manetta motore e come uscite quelle già viste precedentemente. Si andrà a completare, per fare una prima prova, lo schema di Simulink inserendo quattro blocchi costanti agli ingressi e mettendo i soliti display alle uscite. In più aggiungiamo il blocco To Workspace ad ogni uscita per salvare i dati nel Workspace di Matlab. L'intero schema è riportato nella figura 5.

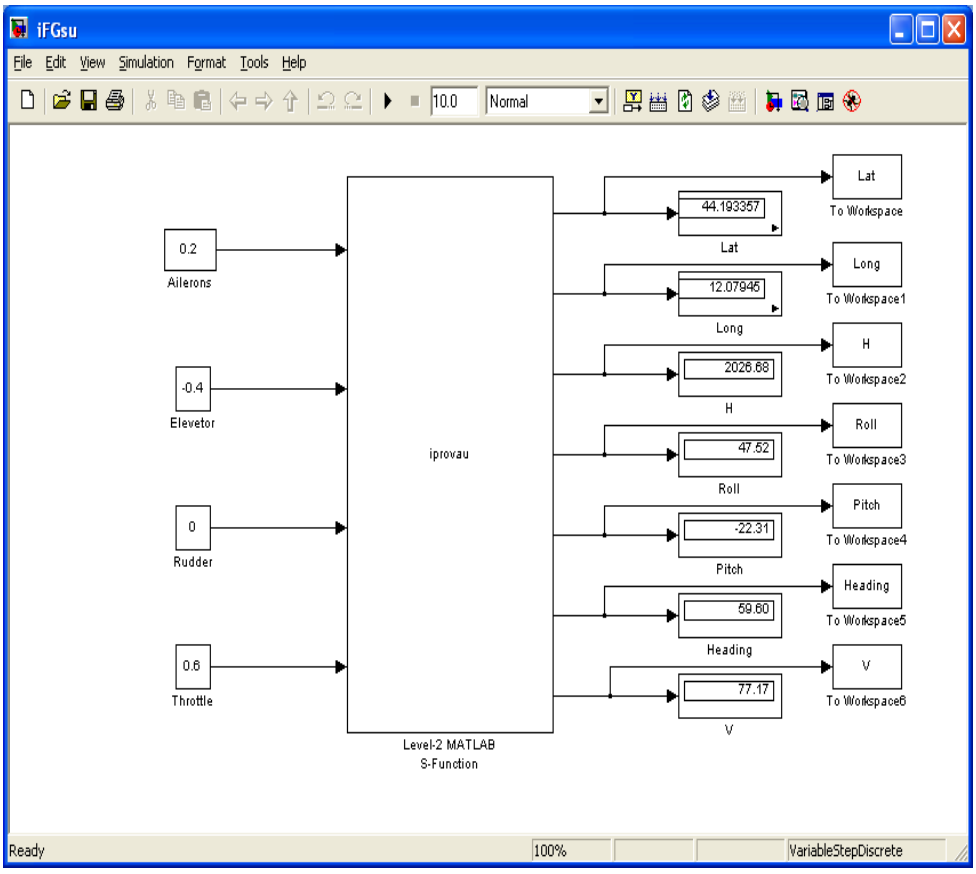

Figura 5

Avviato FlightGear e cliccato play nello schema di Simulink si può vedere che effettivamente l'aereo compie le manovre corrispondenti allo spostamento delle superfici di comando del valore inserito. Si è così riuscito ad ottenere il nuovo tipo di collegamento, tra i due programmi, che si era pensato.

Una cosa importante da sottolineare è che i numeri riferiti agli ingressi devono essere inseriti sapendo che, per la posizione della manetta motore, il valore 1 si riferisce alla massima potenza raggiungibile, mentre il valore 0 alla minima potenza erogata. Nel caso, invece, ci si riferisca agli spostamenti delle superfici di comando i valori non esprimono i gradi di deflessione ma sono riferiti agli spostamenti massimi. In particolare, per angoli negativi, il -1 si ha uno spostamento massimo e, per angoli positivi, si ha uno spostamento massimo con il valore 1. Se invece si da come valore 0 non si ha nessun spostamento angolare. Per fare un esempio pratico, se mettiamo 1 al valore dell'elevatore questi si andrà ad inclinare verso il basso del suo massimo angolo consentito, con 0 è in posizione neutra cioè non si

sposta proprio, mentre con il valore -1 va ad inclinarsi verso l'alto del suo angolo massimo consentito.

Si è così riusciti a realizzare l'idea di base della tesi, ora ci si dedicherà alla riproduzione del piccolo drone in dotazione all'università all'interno di FlightGear .

Capitolo 3

MODELLO DRONE

3.1 UAV

Un aereo a pilotaggio remoto, conosciuto internazionalmente con il nome RPAS (Remotely Piloted Air System) e in precedenza come UAV (Unmanned Aerial Vehicle) che tradotto in italiano significa veicolo aereo senza pilota (che può essere autonomo o con pilotaggio a distanza) è il termine con il quale si definisce quella categoria di veicoli che volano senza l'ausilio di un pilota a bordo. Talvolta vengono impropriamente anche chiamati droni, italianizzando la parola inglese drone, che significa ronzio, a causa del rumore che producono durante il volo. Questi mezzi possono essere completamente autonomi (cioè seguono un profilo di volo pre-programmato) oppure essere telecomandati da terra attraverso una stazione fissa o mobile.

Con il trascorrere del tempo si è arrivati anche alla definizione di small UAV (piccoli UAV) con la quale vengono indicati dei velivoli senza pilota di piccole dimensioni. Tuttavia questa definizione è abbastanza arbitraria, ma solitamente vengono considerati tali i velivoli che sono più piccoli di 6 metri di lunghezza e pesano meno di 25 chili.

All'inizio questi tipi di aereo erano stati creati esclusivamente per scopi militari, ma con il passare del tempo si è pensato di usarli anche per scopi civili (come per esempio il monitoraggio del territorio).

3.2 FLYING WINGS

Un'ala volante (Flying wings) è un aereo costituito solamente dall'ala principale e quindi è privo di fusoliera ed impennaggi. In realtà anche quegli aerei che, pur privi di impennaggi orizzontali e di una fusoliera definita,

sono dotati di piani verticali di dimensioni assai ridotte sono comunemente definiti “ali volanti”.

Questa configurazione, grazie all’eliminazione degli impennaggi e non avendo una fusoliera definita, comporta alcuni vantaggi. In particolare si ha una cospicua diminuzione della resistenza aerodinamica e del peso che, in generale, garantiscono migliori prestazioni rispetto ad altri aerei tradizionali. Purtroppo la soppressione della deriva porta anche degli svantaggi soprattutto si ha una perdita di stabilità direzionale, nonché alcune difficoltà nel volo a velocità ridotte. Il modo migliore per risolvere questo tipo di problemi sarebbe adottare un’ala a delta, ma di solito l’utilizzazione angoli di freccia moderatamente accentuati da dei buoni risultati per la risoluzione del problema.

3.3 DRONE

L’aereo in dotazione all’università è un piccolo UAV di tipo Flying Wings, che chiameremo drone, con una struttura interamente in polistirolo, le superfici di comando sono in compensato molto sottile e tutto questo è ricoperto con dei fogli metallici. Una sua foto e le sue dimensioni principali sono riportate nelle figure 6 e 7:

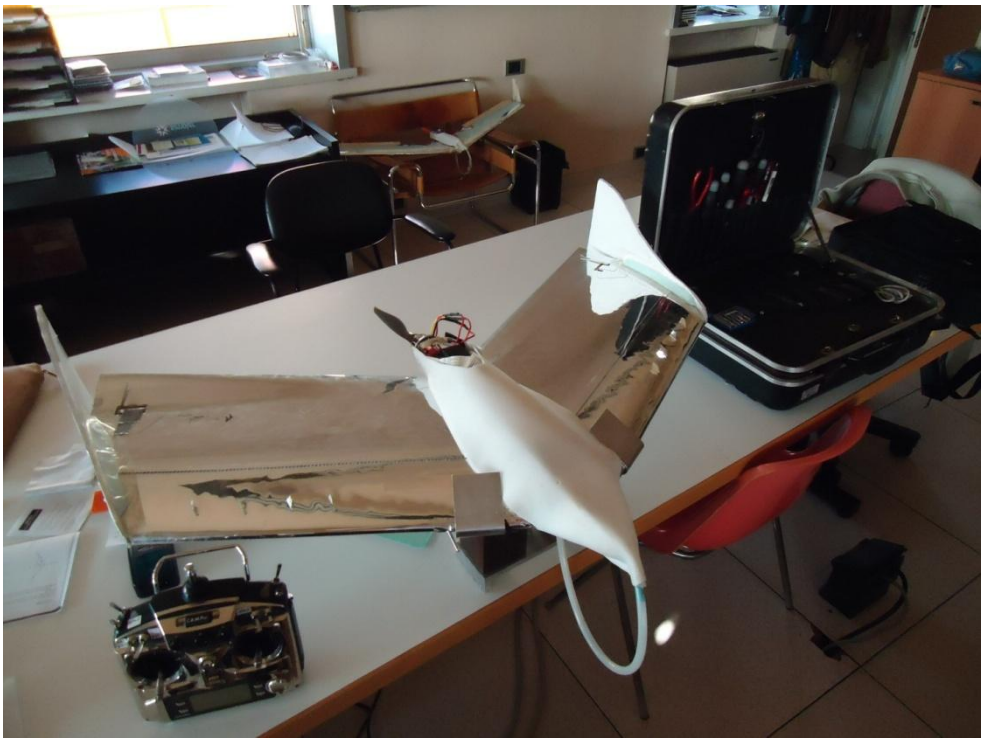

Figura 6

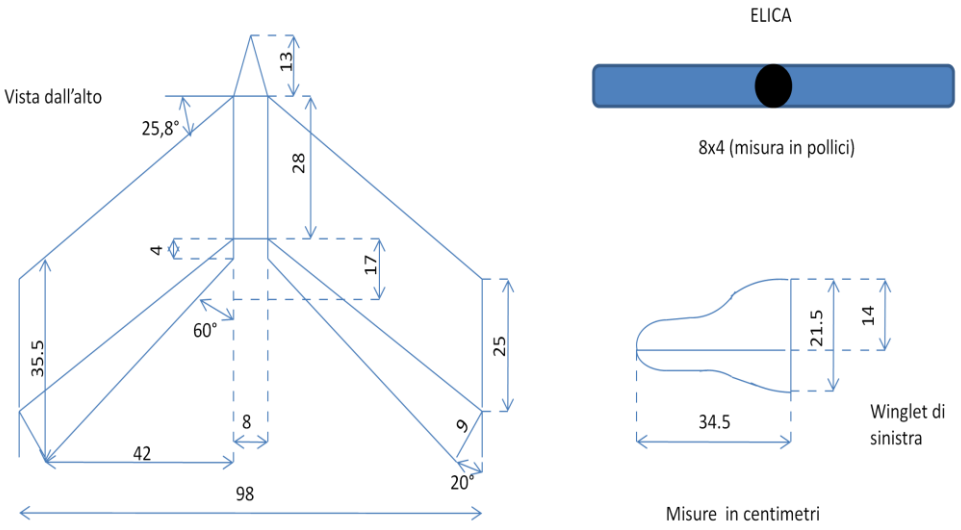

Figura 7

Si sottolinea che le quote dell'aereo, riportate in figura, non sono state fornite dal costruttore ma ricavate direttamente attraverso misurazioni.

Il drone è dotato di diversi sistemi che ne consentono il volo attraverso un radiocomando oppure in modo autonomo. Oltre alla presenza dei servi,

incaricati di muovere le superfici di comando, e del motore, si è installato l'autopilota ArdupilotMega che è composto dalle seguenti parti:

- Ardupilot mega (piattaforma dotata di un microcontrollore ATM 1280 e piattaforma IMU 6 DoF. Questa contiene tre accelerometri, tre giroscopi, una presa di lettura per la tensione di batteria e presa di pressione statica).
- GPS Mediatek (montato a bordo del velivolo consente di sapere sempre la sua esatta posizione).
- AIR Data (modulo dati aria che consente la misurazione della velocità aria del drone).
- Modulo di Telemetria (garantisce la trasmissione dei dati di volo alla ground station).

Tutti questi sistemi consentono all'aereo di avere a bordo degli autopiloti che gli permettono il volo in maniera autonoma una volta prestabilito il percorso.
Ora, presa visione dell'aero e dei suoi sistemi, se ne deve creare un modello che andrà inserito in FlightGear cercando di ottenere, alla fine, delle simulazioni di volo molto vicine alla realtà.

3.4 SCELTA MODELLO AEREO IN FLIGHTGEAR

Valutati gli scopi della tesi si è pensato di partire da un modello di aereo già esistente per FlightGear che abbia caratteristiche simili a quelle del drone e poi andare a modificare i suoi parametri con quelli specifici di quest'ultimo.
Come aereo di partenza si è scelto, tra tutti quelli disponibili sul sito di FlightGear , il Malolo1 poiché come configurazione e dimensione è quello che più si avvicinava alle caratteristiche dell'aereo in dotazione.
Scaricato l'aereo e aperti i suoi file di configurazione si è visto che questi sono sostanzialmente divisi in due aree: una è quella destinata a creare l'aspetto esteriore del velivolo, cioè la figura che si vede sullo schermo una volta avviato FlightGear ; mentre l'altra è quella riferita alla struttura fisica dell'aereo, cioè come si dovrà comportare durante la simulazione. C'è in più un file di settaggio che serve per unire nel modo corretto i file delle due

aree. La cosa importante da notare è che il principale file che riguarda la struttura dell'aereo è stato realizzato utilizzando JSBSim.

3.5 JSBSIM

JSBSim è una raccolta di codice di programma per lo più scritto nel linguaggio di programmazione C++, ma sono incluse alcune routine in linguaggio C. Prende gli ingressi di controllo, calcola le forze e i momenti che derivano da questi e avanza lo stato del velivolo (velocità, orientamento, posizione, ecc) in passi di tempo discreti.

L'architettura di JSBSim è pensata per essere ragionevolmente facile da comprendere, ed è progettata per essere utile agli studenti di ingegneria aerospaziale. Grazie alla facilità con cui può essere configurato, ha anche dimostrato che può essere utile ai professionisti del settore in diversi modi.

3.5.1 STRUTTURA JSBSIM

Questo modello prevede la creazione di diversi file per riprodurre le varie parti dell'aereo e i sistemi che permettono il volo. In oltre serve un ulteriore file di set per far si che le diverse parti create singolarmente (in altri file) si possano unire nel modo corretto per dare origini ad un modello di aereo il più possibile simile a quello reale.

I file più importanti, quelli che poi si andranno a modificare, sono quelli in cui si riporta il modello aerodinamico del velivolo, le sue caratteristiche di peso e la posizione dei vari componenti e quello in cui si riportano le caratteristiche del motore e dell'elica.

Prima di passare alla descrizione della sintassi per la configurazione del file, alcune informazioni di base sui sistemi di riferimento utilizzati per descrivere la posizione degli oggetti sul velivolo.

Structural Frame: questo sistema di riferimento viene usato per i punti del velivolo come il baricentro, la posizione delle ruote del carrello, il punto di vista del pilota, le masse concentrate, l'elica, ecc. Gli assi di riferimento sono orientati con X che va dal naso verso la coda dell'aereo, Y uscente dall'ala destra (guardando in avanti dalla cabina di pilotaggio) e di

conseguenza Z rivolto verso l'alto a formare una terna destrorsa. Di solito, l'origine di questo sistema di riferimento è vicino alla parte anteriore del velivolo (sulla punta del naso o ad una certa distanza poco più avanti ad esso). L'asse X di solito coincide con l'asse spinta. Si noti che l'origine può essere ovunque per un aereo modellato con JSBSim, poiché questo internamente utilizza solo le distanze relative tra il baricentro e i vari oggetti.

Body frame: questo sistema di riferimento, come viene usato in JSBSim, è molto simile al Structural frame, poiché è ruotato di 180 gradi intorno all'asse Y rispetto a questo e l'origine coincide con il baricentro. Questo è il sistema di riferimento in cui le forze, i momenti e le accelerazioni dell'aereo sono integrati per ottenere la velocità.

Stability frame: questo sistema di riferimento è simile al Body frame, eccetto l'asse X coincida con il vettore vento relativo proiettato sul piano XY di simmetria del velivolo. L'asse Y è sempre uscente dall'ala destra e l'asse Z completi la terna destrorsa.

Wind frame: questo sistema di riferimento è simile al Stability frame, eccezion fatta per l'asse X che punta direttamente lungo il vento relativo. L'asse Z è perpendicolare all'asse X e rimane all'interno del piano XZ dell'aereo in assi corpo (chiamato anche piano di riferimento). L'asse Y completa la terna destrorsa.

Una volta capiti i sistemi di riferimento si può passare a vedere come si struttura il file che riguarda l'aerodinamica e il peso.

JSBSim usa, quasi esclusivamente, unità inglesi per i calcoli interni. Tuttavia è possibile inserire alcuni parametri nel file di configurazione utilizzando diverse unità di misura. Per evitare confusione è sempre meglio specificare l'unità di misura utilizzata e lo si fa utilizzando il comando "unit".

Ci sono diversi modi per modellare le forze aerodinamiche e i momenti agenti sull'aereo. JSBSim utilizza il metodo del coefficiente di accumulo che consiste nel sommare vari contributi per determinare una forza, per esempio per determinare la portanza totale si sommano i contributi dati dalle singole superfici portanti. Per ottenere i vari contributi (che non sono altro che forze) che poi si sommeranno tra loro, bisogna introdurre dei

coefficienti che, opportunamente moltiplicati, danno come risultato proprio una forza. Tali coefficienti possono essere ottenuti da test di volo, libri di testo, usando degli appositi software oppure calcolandoli a mano.
Una volta ottenuti devono essere inseriti in apposite tabelle poste all'interno della sezione <aerodynamics>. Questa sezione presenta al suo interno sei sottosezioni, le quali rappresentano i tre assi di riferimento e i tre assi momento (per un totale di sei gradi di libertà).
In JSBSim sono presenti tre set standard di assi tra cui si può scegliere: DRAG-SIDE-LIFT (assi vento), X-Y-Z (assi corpo) e AXIAL-SIDE-NORMAL (assi corpo). Tutti e tre questi sistemi accettano le definizioni di assi ROLL, PITCH e YAW. Una cosa importante è che non si possono mischiare i tre sistemi di riferimento.
Un'altra sezione importante è definita <metrics>. Qui vengono definite le misure caratteristiche geometriche del velivolo e la posizione dei punti chiave (vpr, eyepoint, aerorp). Una particolare attenzione la poniamo al VPR (Visual Reference Poin) al quale FlightGear si riferite per calcolare le grandezze latitudine, longitudine e altitudine.
Anche la sezione <mass_balance> merita attenzione perché è quella che permette di descrivere esattamente la massa dell'aereo e le sue proprietà. In questa sezione sono inclusi il peso a vuoto dell'aereo, i momenti di inerzia, la posizione del baricentro e la posizione delle concentrazioni di masse.
Importante infine è anche la sezione <propulsion> nella quale si specifica il posizionamento del o dei motori e la loro orientazione, la posizione dell'elica, se presente, e quella dei vari serbatoi.

3.6 MODELLIZZAZIONE DRONE

Per riuscire ad elaborare in JSBSim un corretto modello bisogna conoscere tutti i dati che riguardano la geometria, il peso, le caratteristiche aerodinamiche e le prestazioni del motore da poter inserire all'interno dei file di cui si è parlato precedentemente.
Per la creazione della geometria dell'aereo si è usato il software XFLR5 che ne consente la realizzazione in modo abbastanza agevole e con un riscontro visivo diretto di quello che si è creato.

3.7 XFLR5

È un software per l'analisi di ali e aerodine complete sviluppato dal francese Andre Deperrois, distribuito con licenza open source. XFLR5 analizza il comportamento non solo di un profilo isolato (come fanno le "gallerie virtuali" come XFoil) ma anche di un'ala finita tenendo conto del suo allungamento e della sua forma, e di un'aerodina completa. In questo modo è possibile studiare la resistenza indotta dall'ala, costruire la polare dell'ala o del modello, analizzare la distribuzione di portanza lungo l'apertura alare, individuare le condizioni di trim e di stabilità del modello.

3.7.1 GUIDA ALL'USO

Una volta scaricato e installato il programma lo si andrà ad avviare, cliccando sull'apposita icona, ottenendo l'aperture di una finestra completamente nera come mostra la figura 8.

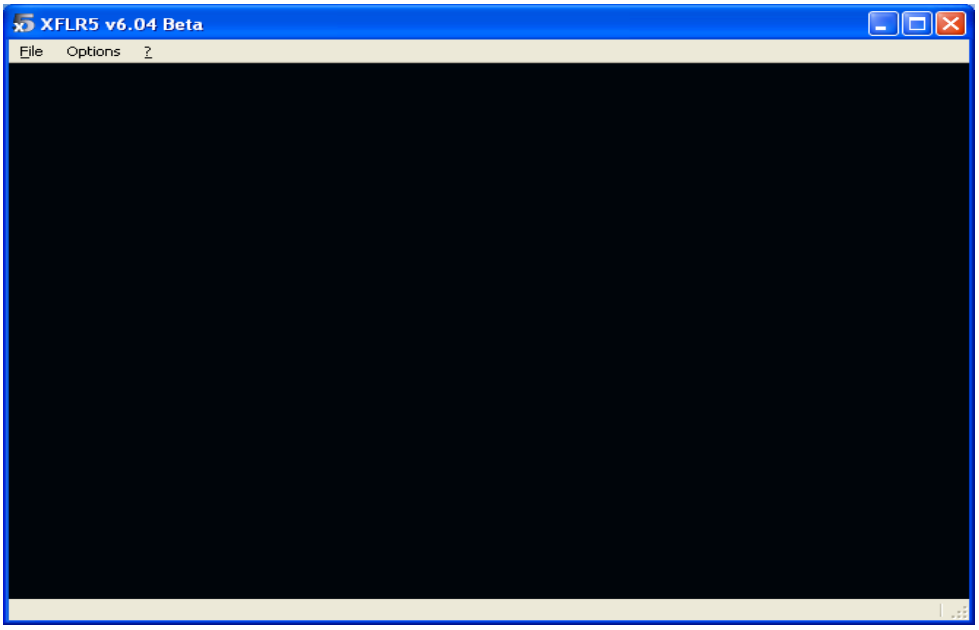

Figura 8

Una cosa importante da fare, prima di iniziare la costruzione vera e propria dell'aereo, è quella di caricare all'interno del programma tutti i profili che si

vorranno usare per le superfici portanti del velivolo ricordando che altrimenti il programma andrà ad usare il profilo NACA 0012.

Per fare questa semplice operazione basta andare su file, cliccare su open e selezionare il profilo desiderato dall'apposita cartella. Al termine dell'operazione otteniamo la schermata riportata nella figura 9.

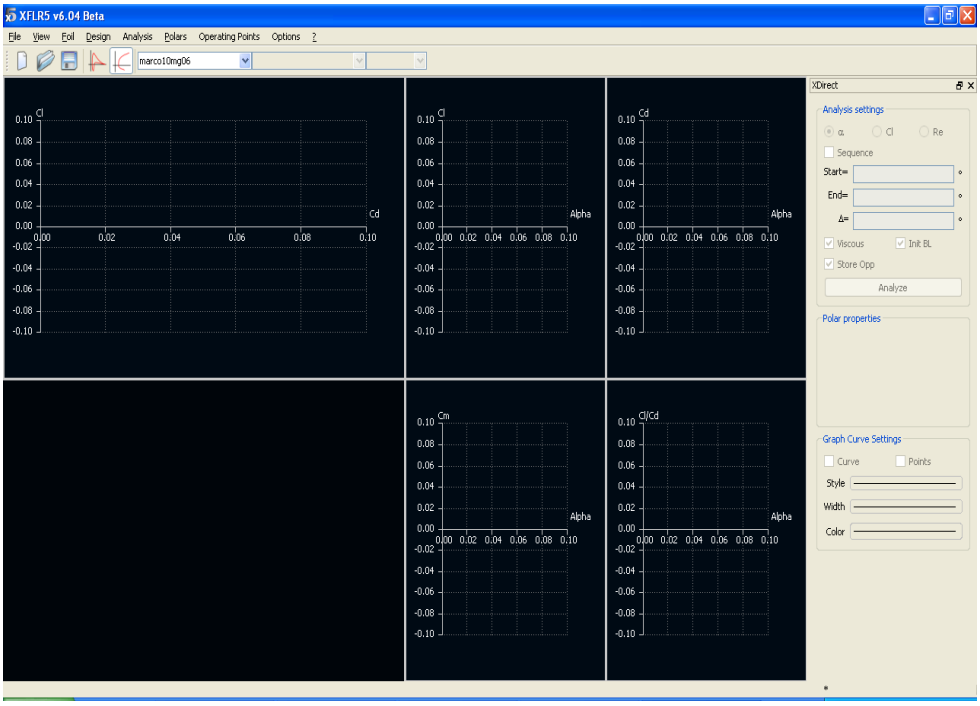

Figura 9

Se si seleziona view e si va a cliccare su OpPoint view (oppure direttamente sull'apposito tasto) si apre una nuova schermata dove viene visualizzata la forma del profilo caricato come in figura 10.

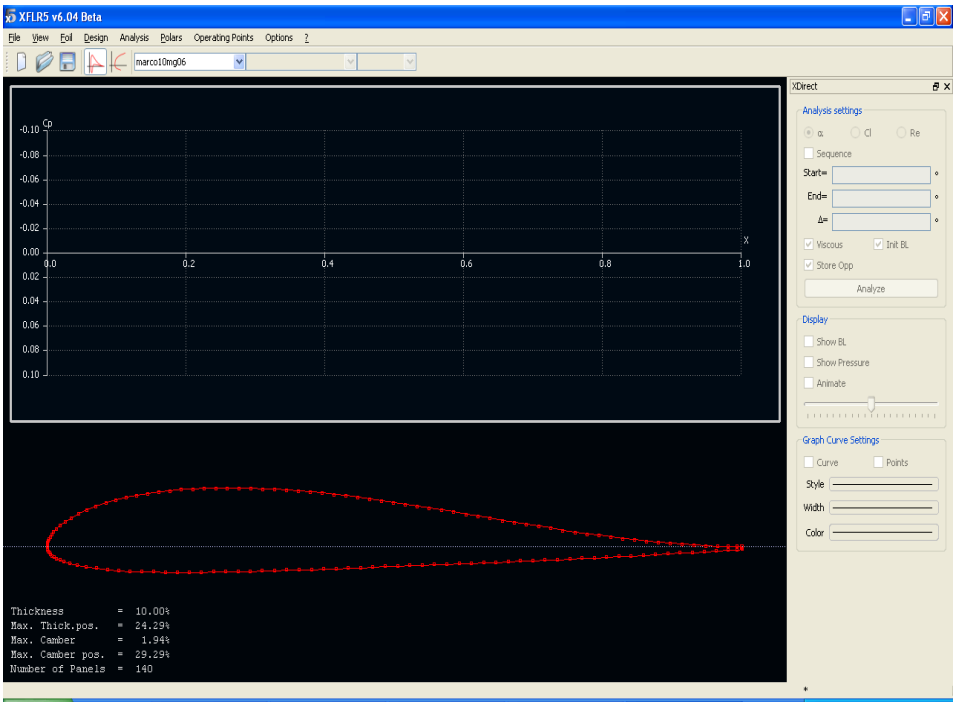

Figura 10

Affinché il profilo possa essere effettivamente usato per le superfici aerodinamiche dell'aereo va prima effettuata su di esso l'analisi. Tale operazione si effettua selezionando Analysis e cliccando su Batch Analysis e subito compare la finestra riportata nella figura 11.

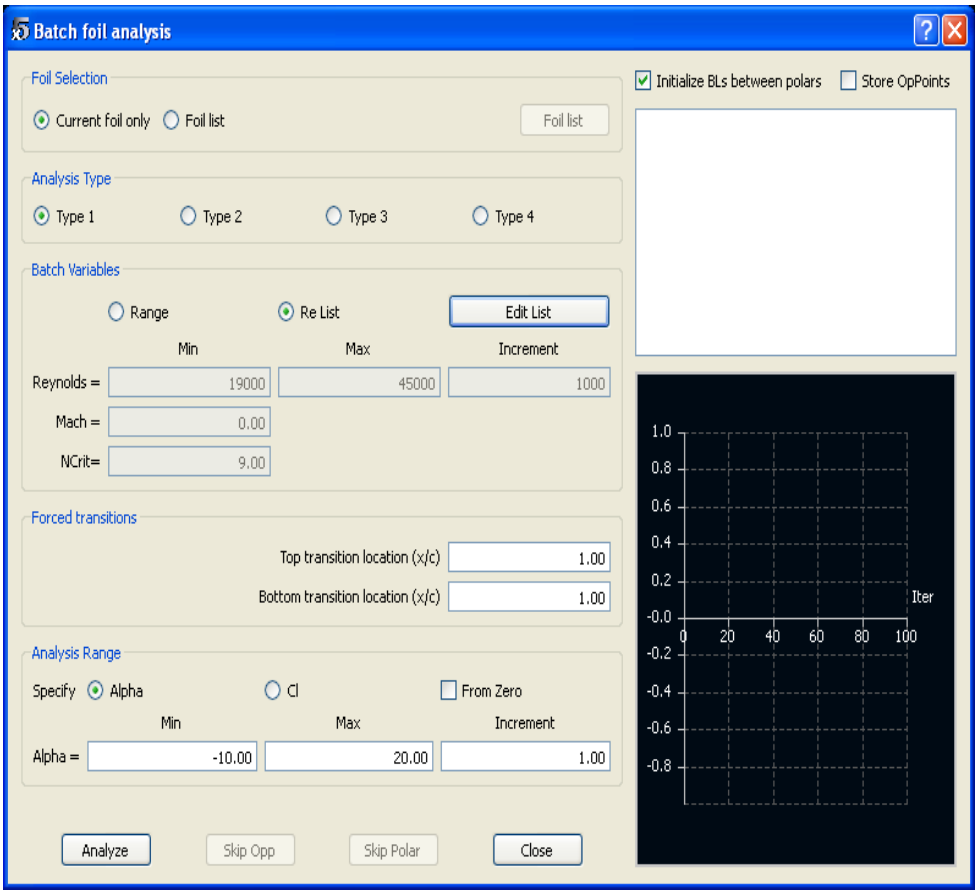

Figura 11

Si selezionano tutti i vari parametri, a seconda delle esigenze, e successivamente si può avviare la simulazione cliccando su Analyze. Una volta terminata l'analisi, si ottengono nella finestra Polar view (figura 9) dei grafici che riguardano le caratteristiche del singolo profilo simili a quelli di figura 12.

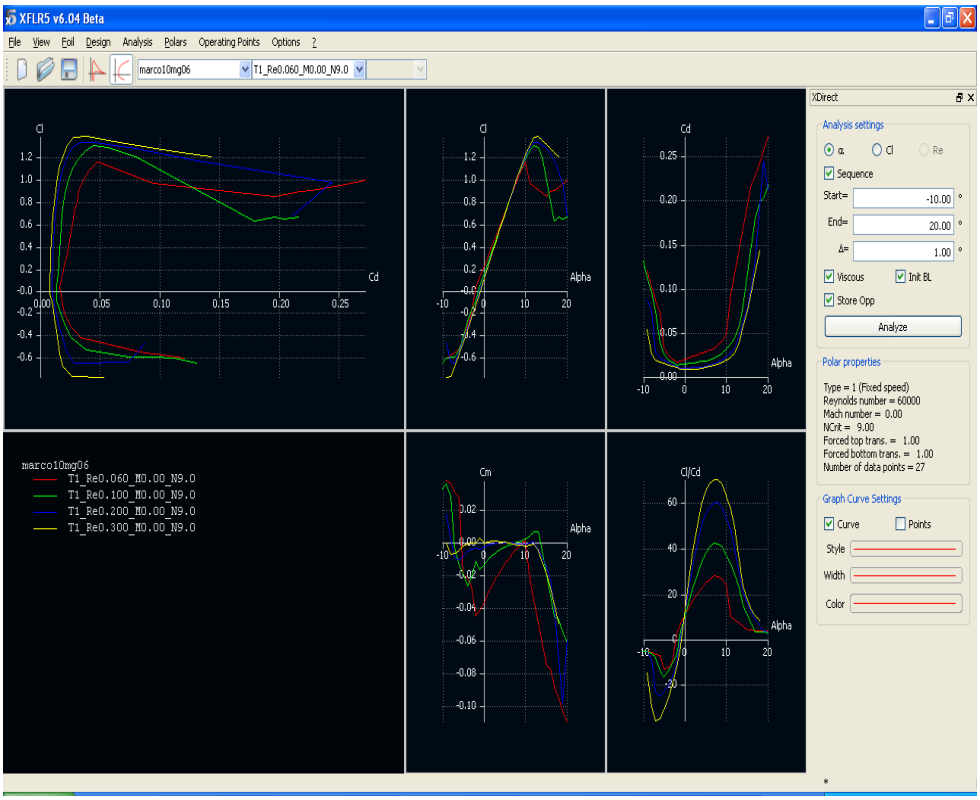

Figura 12

Ogni colore si riferisce ad un numero di Reynolds ben preciso. Ora si andrà a salvare tutto il lavoro fatto cliccando sull'apposito pulsante in cui è raffigurato il floppy disc. Così facendo il profilo è pronto per essere utilizzato sulle superfici aerodinamiche dell'aereo. Cosa importante è che questa operazione va ripetuta per tutti i diversi profili che si desidera utilizzare.

È anche possibile creare dei profili che presentino già uno spostamento delle superfici di comando. Lo si può fare andando nella finestra OpPoint view, dove si visualizza il profilo (figura 10), selezionando Design e cliccando su Set Flap. A questo punto si aprirà una finestra dalla quale è possibile modificare il profilo originale a seconda delle esigenze (figura 13).

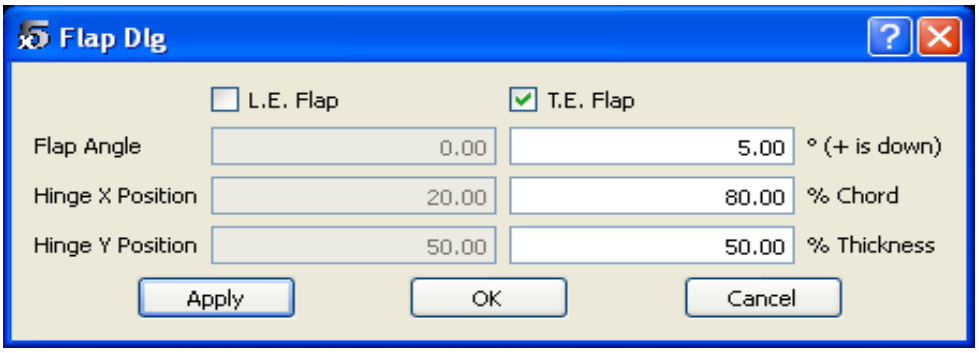

Figura 13

Una volta modificato a piacimento andrà anche salvato, come fatto per il profilo originale, per poi poterlo utilizzare successivamente nel modello dell'aereo. Una cosa importante è che vanno salvati tanti profili per quanti angoli di inclinazione delle superfici di comando si desiderano effettuare le simulazioni.

Terminate queste operazioni si passa alla creazione vera e propria del modello dell'aereo. Come prima cosa si va su file, si seleziona Wing and Plane Design e si aprirà una schermata come quella della figura 14.

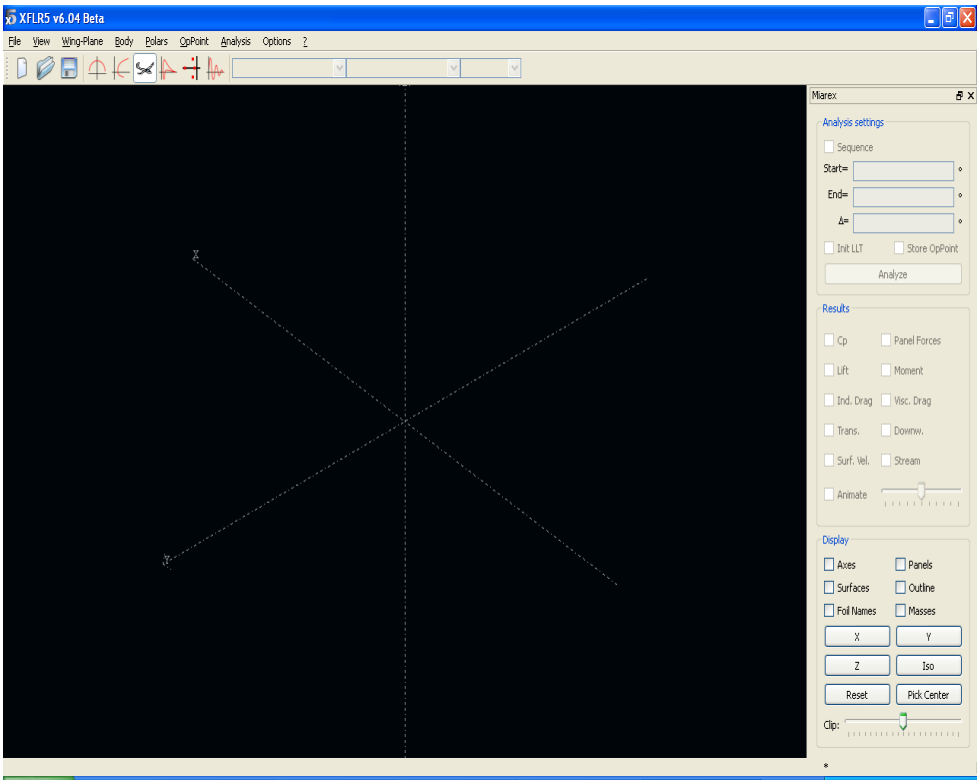

Figura 14

Questa è la schermata in cui, una volta terminata la costruzione di tutto l'aereo, si andrà a visualizzare la sua configurazione definitiva.

Per iniziare la costruzione del modello è più opportuno iniziare con la modellazione della fusoliera, anche perché è quella che può creare più problemi. Si parte andando su Body, cliccando su Define a New Body aprendo così una finestra (figura 15) dalla quale è possibile modellare interamente la fusoliera cercando di crearne una che sia il più possibile simile a quella reale.

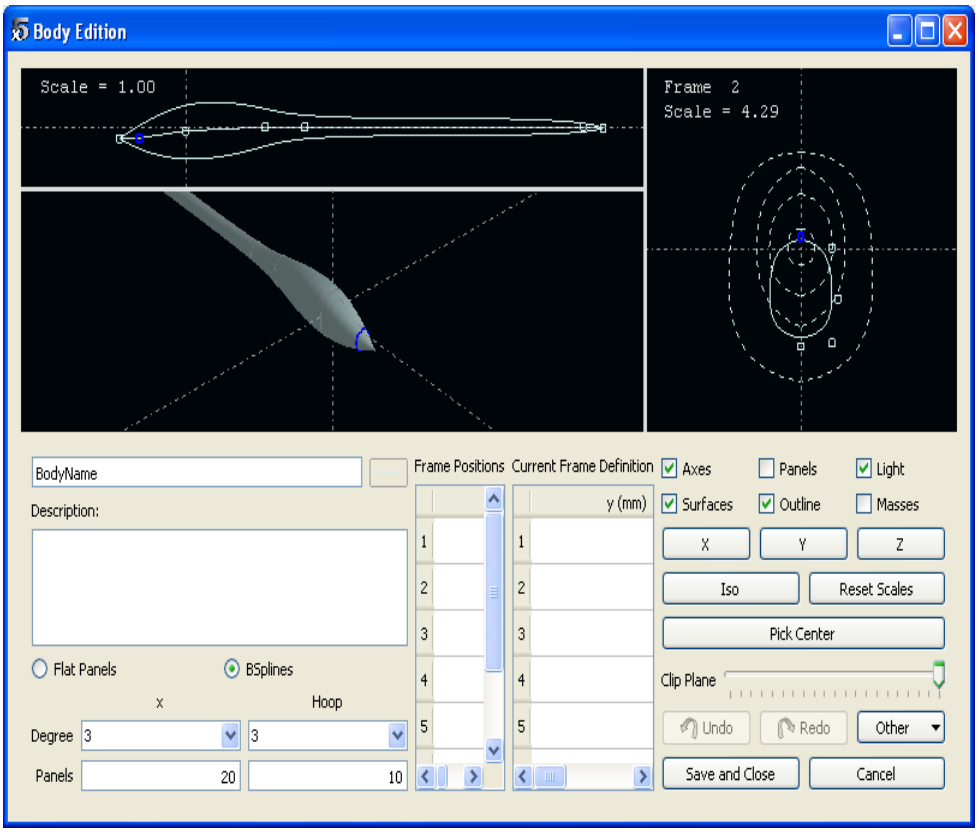

Figura 15

In particolare, oltre a modellare la forma, è possibile anche inserire il peso della singola struttura, il peso di masse concentrate e la loro esatta posizione. Affinché si possano inserire queste informazioni basta andare su Other, cliccare su Inertia e compilare la finestra che si aprirà (figura 16).

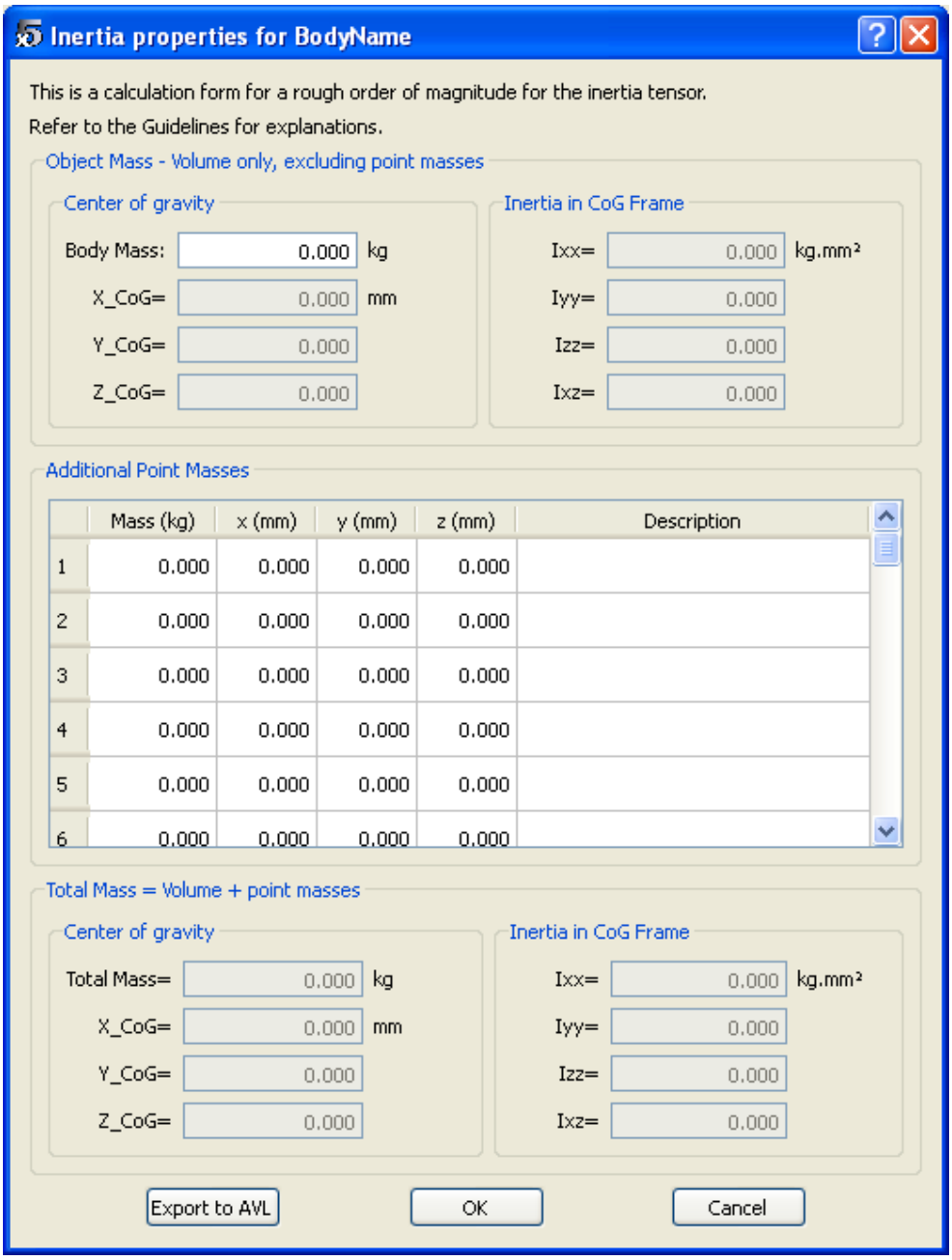

Figura 16

Una volta completata tutta la struttura cliccare su Save and Close nella finestra rappresentata nella figura 15.

Una volta terminata l'intera struttura della fusoliera si può passare alla creazione delle restanti parti dell'aereo per poi unirle tutte insieme. Questa operazione la si può fare andando su Wing-Plane e cliccando su Define a New Plane, si aprirà così una schermata in cui vengono elencate tutte le parti dell'aereo (figura 17).

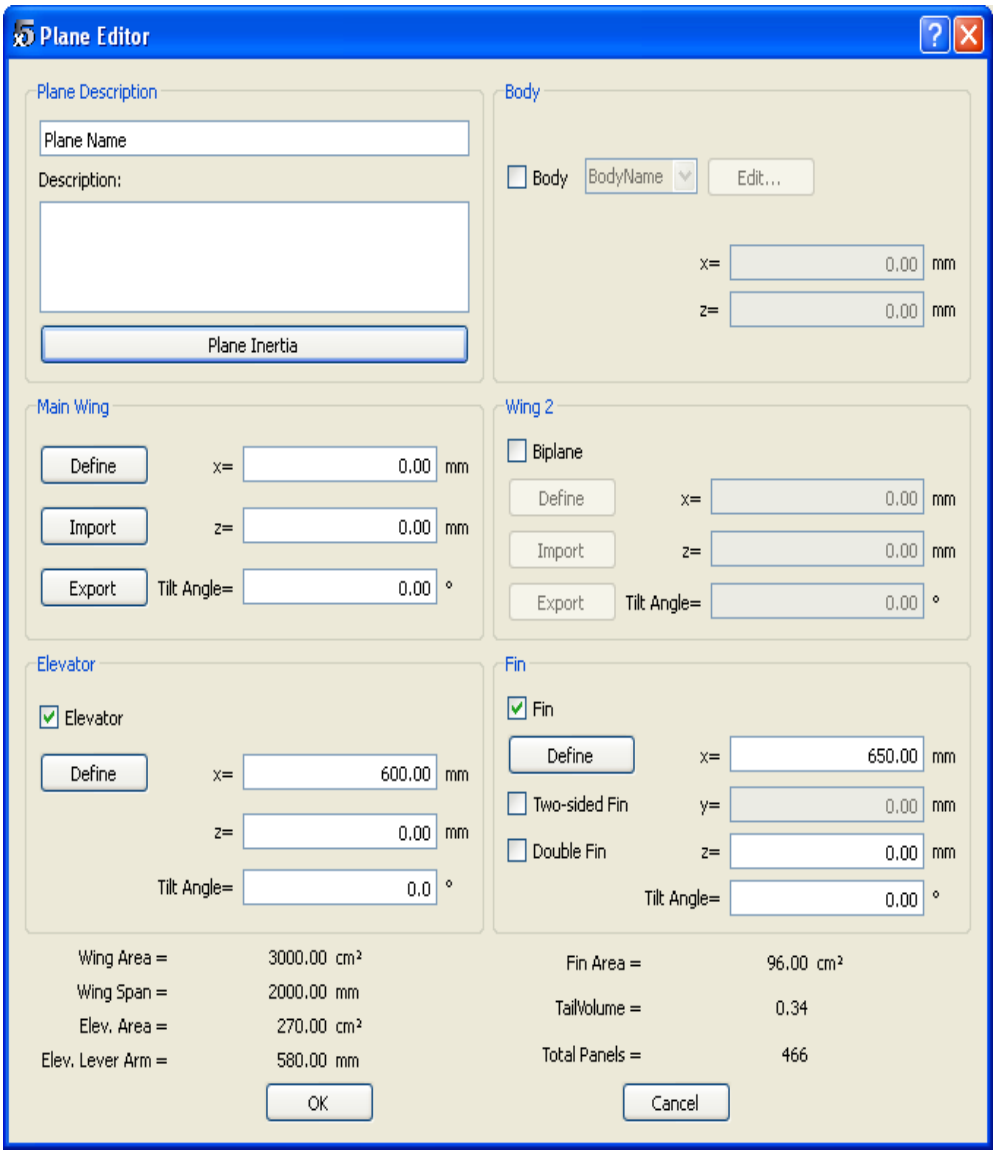

Figura 17

Spuntando gli appositi quadratini si va a selezionare la parte dell'aereo che si vuole includere per ottenere la configurazione desiderata. Ogni singola parte, che è stata selezionata, andrà poi modellata autonomamente e poi unirle insieme.

Per creare o modellare qualsiasi elemento dell'aereo basta cliccare sul Define relativo. In questo modo si accede ad una specifica finestra che permette di settare tutti i vari parametri riguardanti l'elemento selezionato. Per esempio se si vuole modellare l'ala principale si cliccherà sul pulsante Define apposito e successivamente si aprirà una finestra (figura 18).

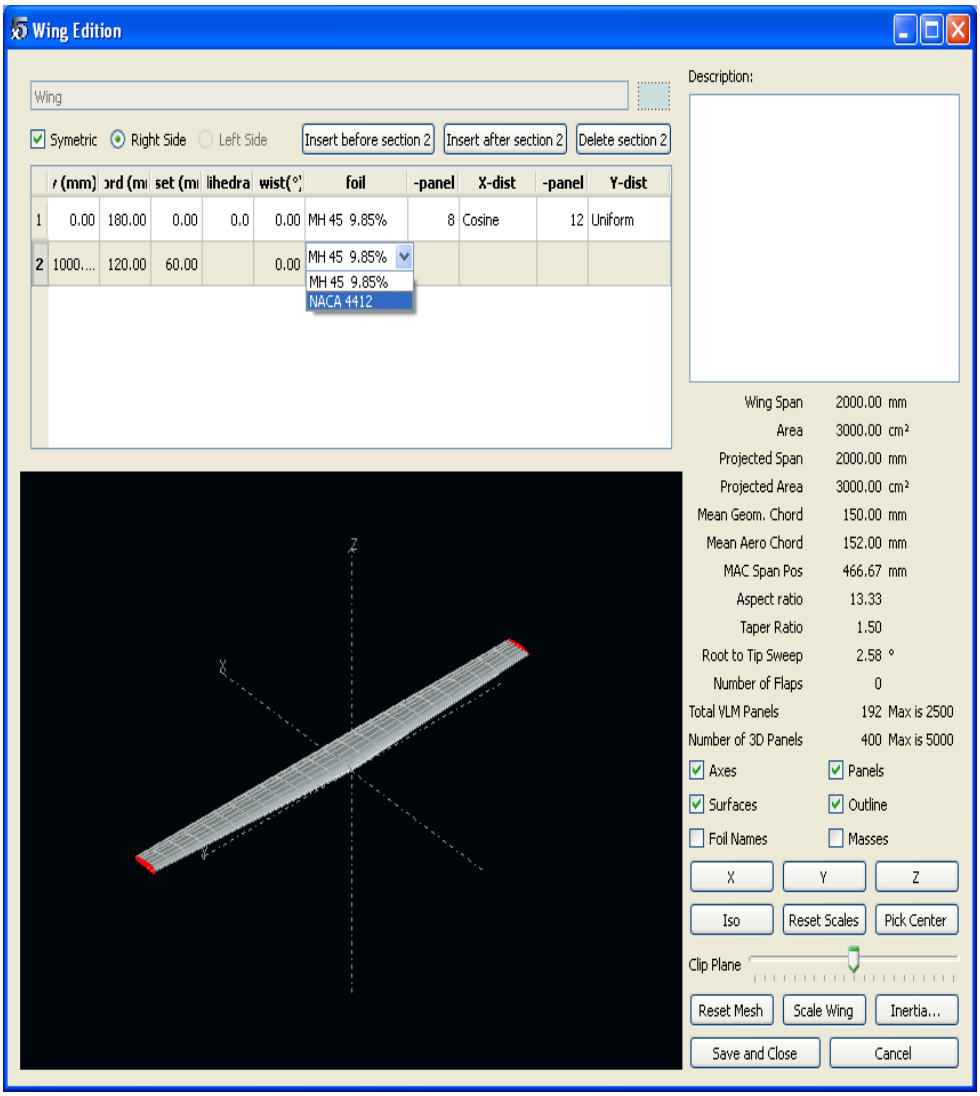

Figura 18

Come si può facilmente vedere dalla figura, la creazione dell'ala è abbastanza semplice e intuitiva. In particolare si nota che in questo caso è possibile variare il numero delle sezioni di cui è formata l'ala e che su ogni sezione può essere selezionato differente profilo (chiaramente tra quelli che sono stati salvati in precedenza). Anche in questo caso, come visto in precedenza per il Body, è possibile indicare il peso dell'ala e le varie concentrazioni di massa che si hanno su di essa andando a cliccare il pulsante Inertia. Per quanto riguarda gli altri elementi da modellare, ad eccezione del Body che si è già visto come si costruisce, la finestra che permette la loro creazione praticamente identica a quella vista per l'ala principale nella figura 18.

Una volta create e modellate tutte le parti del nostro aereo andranno posizionate, una rispetto all'altra, correttamente in modo da riprodurre l'esatta configurazione del nostro aereo. Questa operazione la si può fare settando i parametri (x, y dove presente, z e Tilt Angle), di ogni singolo elemento, presenti nel Plane Editor (figura 17). Una volta cliccato OK si visualizzerà l'aereo nel suo insieme nella finestra principale (figura 14). Spesso e volentieri il risultato che si ottiene non rispecchia quello desiderato, allora si deve tornare nel Plane Editor e modificare i valori dei parametri fino a quando non si è arrivati ad una configurazione soddisfacente.

3.7.2 MODELLO DRONE IN XFLR5

Seguendo le indicazioni appena riportate si è costruita la geometria del drone all'interno di XFLR5 e viene riportata in figura 19.

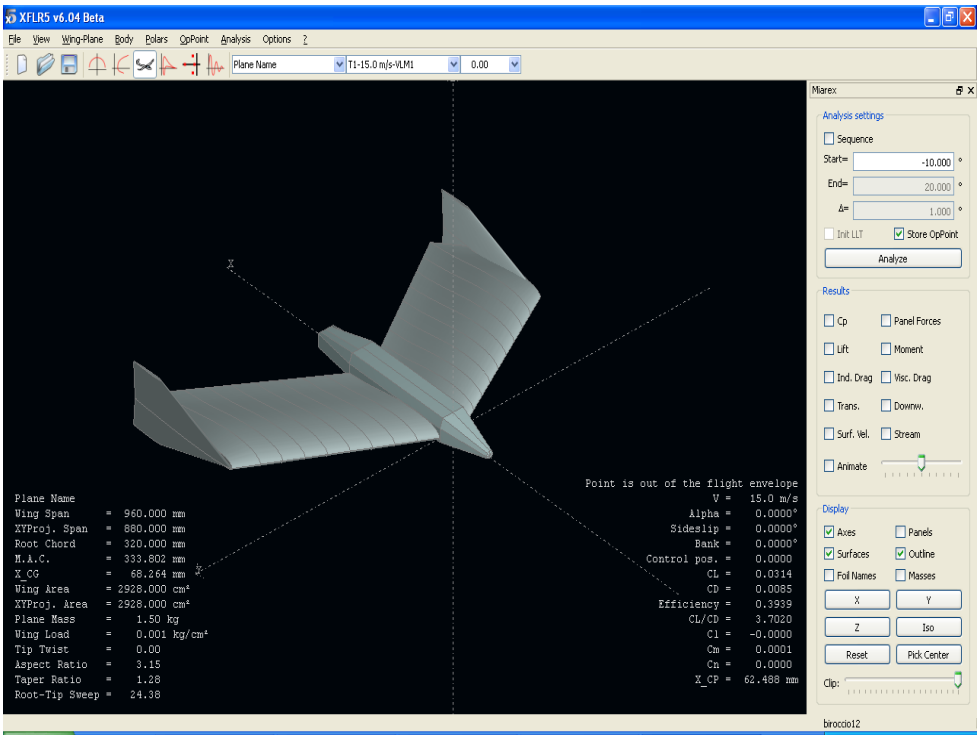

Figura 19

Come si può facilmente vedere il modello rispecchia abbastanza bene l'aereo reale tranne per la forma delle winglet. Queste, purtroppo, nella

realtà hanno una forma piuttosto complessa che è impossibile riprodurre all'interno di XFLR5 e quindi si è dovuto optare per una forma che ci si avvicinasse il più possibile e che ricordasse vagamente quelle reali.
Il profilo utilizzato per le ali è stato fornito direttamente del costruttore attraverso il file di dati marco10mg06.dat.
Ogni qualvolta che si vuole cambiare un qualsiasi parametro all'aereo è possibile farlo andando in Wing-Plane, selezionando Current Plane e infine Edit. Così facendo si va ad aprire direttamente la finestra del Plane Editor (figura 17) dalla quale è possibile modificare qualsiasi parametro dell'aereo.
Creato il modello si dovranno ora andare a calcolare tutti i coefficienti aerodinamici che lo caratterizzano. Per effettuare questi calcoli si è pensato di usare un altro programma, AVL, più specifico per lo scopo.
Questo passaggio lo si può fare abbastanza agevolmente perché XFLR5 consente di esportare il suo modello direttamente in AVL. Per fare questa operazione basta andare su Wing-Plane, selezionare Current Plane e cliccare infine su Export to AVL. In questo modo si è andati a creare un file di AVL nel quale è riportata la geometria di tutte le superfici alari dell'aereo. Purtroppo però viene esclusa la fusoliera. Oltre alla geometria, vanno anche esportate in AVL le proprietà di massa e la distribuzione dei pesi. Questo lo si può fare andando sempre su Wing-Plane, selezionando Current Plane e cliccando su Define Inertia. Fatta tale operazione si apre una finestra come quella riportata nella figura 20.

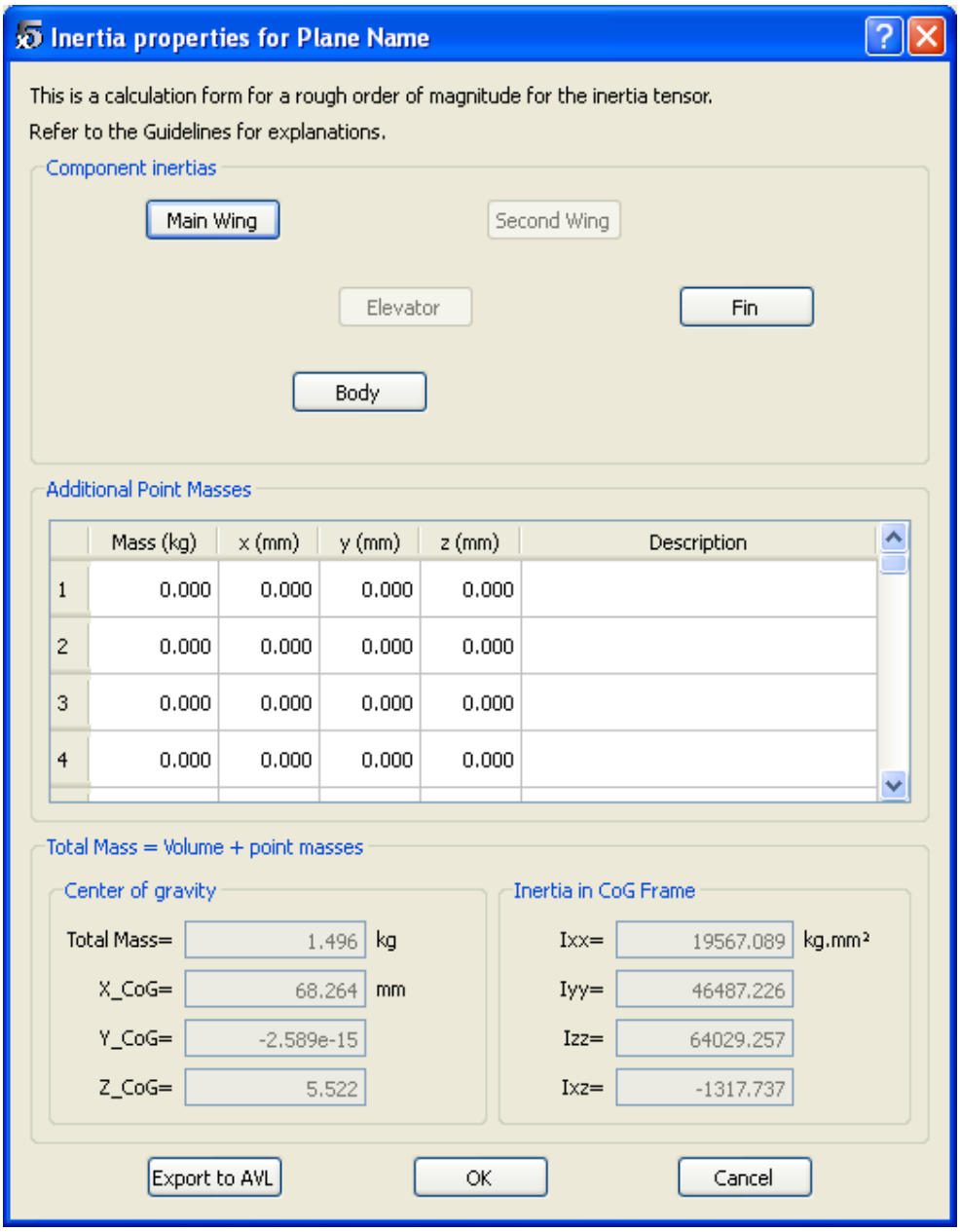

Figura 20

Questa finestra riassume tutte le informazioni sul peso dell'aereo e delle singole parti, la locazione del baricentro e anche il valore dei momenti di inerzia rispetto al baricentro.

Come si può facilmente vedere per esportare tutte questa informazioni in AVL basta semplicemente cliccare su Export to AVL così da ottenere un file di AVL che riguarda la massa dell'aereo e sarà associato al file, riportante la geometria, precedentemente creato.

Da questa finestra è possibile eventualmente modificare le caratteristiche di peso e distribuzione di massa di ogni singola parte andando a cliccare sul pulsante con il nome dell'elemento specifico.

3.8 AVL

AVL è un programma per l'analisi aerodinamica e di dinamica di volo di aerei rigidi di configurazione arbitraria. Impiega un esteso modello di vortex lattice per le superfici portanti, unito ad un modello slender-body per la fusoliera e le gondole motori. L'analisi dinamica del volo unisce una piena linearizzazione del modello aerodinamico di qualsiasi stato del volo con le specificate proprietà di massa.

3.8.1 GUIDA ALL'USO

Per far si che AVL possa compiere l'analisi dei coefficienti aerodinamici bisogna, per prima cosa, creare i due file in cui vengono riportate tutte le caratteristiche del velivolo. Uno è il file.avl dove si riporta l'intera geometria dell'aereo, l'altro è il file.mass dove si riporta il suo peso e la sua esatta distribuzione all'interno della struttura.

Questi due file possono essere creati utilizzando qualsiasi editor di testo, in questo caso si è usato WordPad, seguendo una particolare e precisa sintassi.

Per prima cosa si specifica subito che il sistema di riferimento a cui ci si riferisce per la costruzione della geometria dell'aereo è il seguente: l'asse X è uscente dalla coda dell'aereo, l'asse Y uscente dall'ala destra e infine l'asse Z, a formare una terna destrorsa, è orientato verso l'alto. Il centro della terna cartesiana può essere posizionato dove si preferisce, anche se è consigliabile metterlo o sul muso dell'aereo oppure nel suo baricentro per una più semplice e veloce comprensione.

Ora si può vedere come costruire il primo dei due file, quello riguardante la geometria.

Il file deve iniziare con le seguenti informazioni nelle prime cinque righe senza inserire nessuna linea di commento tra di esse:

```
abc…                    | titolo
0.0                     | Mach
1       0       0.0     | iYsym   iZsym   Zsym
4.0     0.4     0.1     | Sref    Cref    Bref
0.1     0.0     0.0     | Xref    Yref    Zref
0.020                   | CDp (opzionale)
```

Mach = numero di Mach predefinito del flusso libero per la correzione di Prandtl-Glauert.

iYsym = 1 caso simmetrico rispetto a Y=0,

= -1 caso antisimmetrico rispetto a Y=0,

= 0 non si assume nessuna simmetria rispetto a Y.

iZsym = 1 caso simmetrico rispetto a Z=Zsym,

= -1 caso antisimmetrico rispetto a Z=Zsym,

= 0 non si assume nessuna simmetria rispetto a Z (Zsym ignorato).

Zsym = posizione del suolo.

Sref = area di riferimento usata per definire tutti i coefficienti (CL, CD, Cm, etc).

Cref = corda di riferimento usata per definire il momento picchiante (Cm).

Bref = apertura di riferimento per definire i momenti di rollio e imbardata (Cl, Cm).

X,Y,Zref = posizione del baricentro.

CDp = coefficiente di resistenza del profilo dovuto alla geometria.

La creazione vera e propria della geometria dell'aereo inizia adesso.

Si incomincia con il creare le varie superfici portanti. È consigliabile partire dalla costruzione dell'ala principale per avere un approccio più facile.

```
SURFACE     | per riferirsi a superfici a cui si può associare un profilo
Main Wing   | nome della superficie a cui ci si riferisce
12      1.0 | Nchord      Cspace
```

Nchord = numero di vortici, lungo la corda, presenti sulla superficie.

Cspace = parametro di spaziatura del vortice lungo la corda.

INDEX |
3 |
YDUPLICATE |
0.0 | posizione Y del piano X-Z
SCALE | permette l'intera riscalatura di tutta la superficie
1.0 1.0 0.8 | Xscale Yscale Zscale

Xscale, Yscale, Zscale = fattore di scala applicato a tutte le coordinate X,Y,Z.

TRANSLATE | per traslare la superficie senza modificare le coordinate
10.0 0.0 0.5 | dX dY dZ

dX, dY, dZ = offset aggiunto a tutti i valori X, Y, Z della superficie.

ANGLE | permette di cambiare l'angolo d'incidenza di tutta la superficie
2.0 | dAinc

dAinc = offset dell'angolo.

SECTION | indica una sezione
0.0 5.0 0.2 0.5 1.50 5 | Xle Yle Zle Chord Ainc
AFILE | serve per avere la possibilità di richiamare un profilo da un file di dati
Prova.dat | nome del file

Xle, Yle, Zle = posizione del bordo di attacco del profilo.
Chord = lunghezza della corda del profilo.
Ainc = angolo di calettamento del profilo.

Quest'ultimo blocco (da SECTION in giù) andrà ripetuto per tutte le sezioni che si desiderano avere lungo la superficie.

L’intero blocco, invece, andrà ripetuto per tutte le superfici portanti che si vuole creare.
Per quanto riguarda la fusoliera la si può creare in due modi distinti. Il primo è quella di considerarla come una superficie portante e andrà costruita sulla falsa riga di quello visto fino ad ora. Il secondo invece è quella di importarla direttamente da un file di dati e lo si può fare attraverso i seguenti comandi:

BODY | per riferirsi alla fusoliera
BFILE | serve per avere la possibilità di richiamare una fusoliera da un file di dati
Nome.dat | nomo del file

creare Dopo aver creato il file.avl si può passare a vedere come creare il secondo che riguarda la massa e la sua distribuzione nel velivolo.
La prima cosa da fare, all’inizio del file, è impostare le unità di misura di riferimento così da scalare correttamente tutte le misure che si introdurranno in questo file e quelle già introdotte all’interno del file precedente.

Lunit = 0.0254 m
Munit = 0.001 kg
Tunit = 1.0 s
g = 9.81
rho = 1.225

In questo modo si definiscono le unità di misura e le costanti di riferimento.
Ora si può procedere nello specificare l’esatto peso del velivolo e in che modo è distribuita la massa su di esso. Questa operazione è molto facile perché si deve semplicemente compilare una tabella, riportata di seguito, con le caratteristiche di ogni singola parte dell’aereo.

Mass	x	y	z	Ixx	Iyy	Izz	
58.0	3.34	12	1.05	4400	180	4580	!ala destra

58.0	3.34	-12	1.05	4400	180	4580	!ala sinistra
16.0	-5.2	0.0	0.0	0	80	80	!fusoliera

Nella colonna mass va inserito il peso dell'elemento considerato, le colonne x,y,z si riferiscono alla posizione del baricentro del suddetto elemento, mentre nelle restanti colonne (Ixx, Iyy, Izz) vanno inseriti i momenti di inerzia riferiti rispettivamente all'asse x, y e z del singolo elemento.
Più parti si inseriscono più è accurato il modello e più accurati saranno i risultati.
Una volta compilati questi due file andranno salvati e posizionati all'interno della cartella runs di Avl. Fatta questa operazione si può procedere nel vedere come effettuare l'analisi del modello e ricavare così i coefficienti desiderati.
Per prima cosa si apre AVL cliccando sull'apposita icona e comparirà una finestra come quella riportata in figura 21.

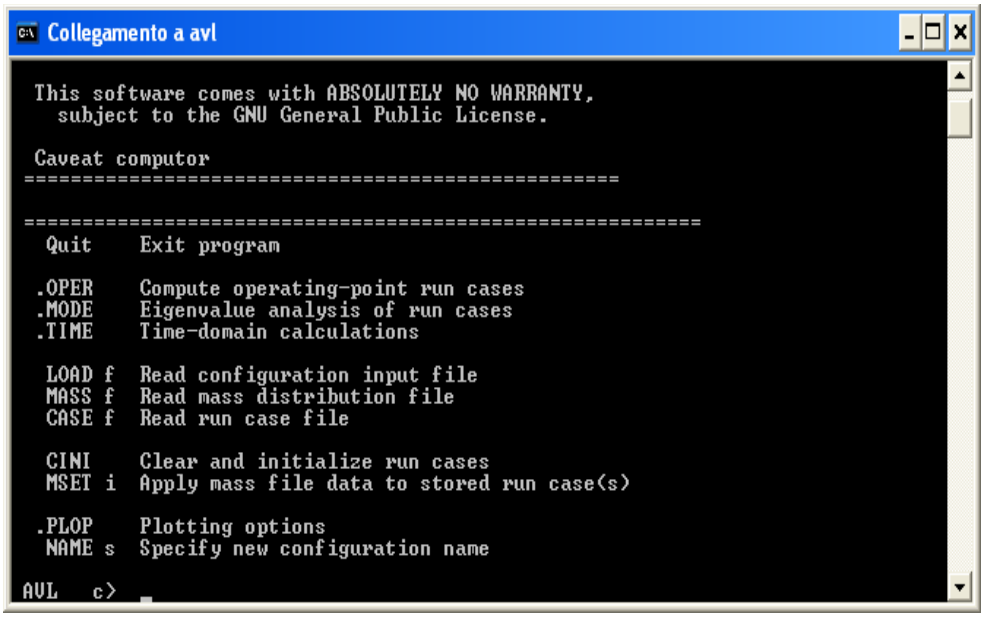

Figura 21

Come si può facilmente vedere dalla figura questo è un programma abbastanza intuitivo da usare in quanto tutti i comandi che possono essere sfruttati, con relativa spiegazione, vengono sempre specificati nelle varie

schermate. Prima di iniziare, per semplificare, si specifica che ad ogni comando digitato segue sempre la premuta del tasto invio.

Ora si può procedere nel vedere come è possibile caricare i due file visti in precedenza all'interno del programma. Si incomincia digitando il comando load. Nella riga successiva andrà inserito il nome del file della geometria dell'aereo come in figura 22.

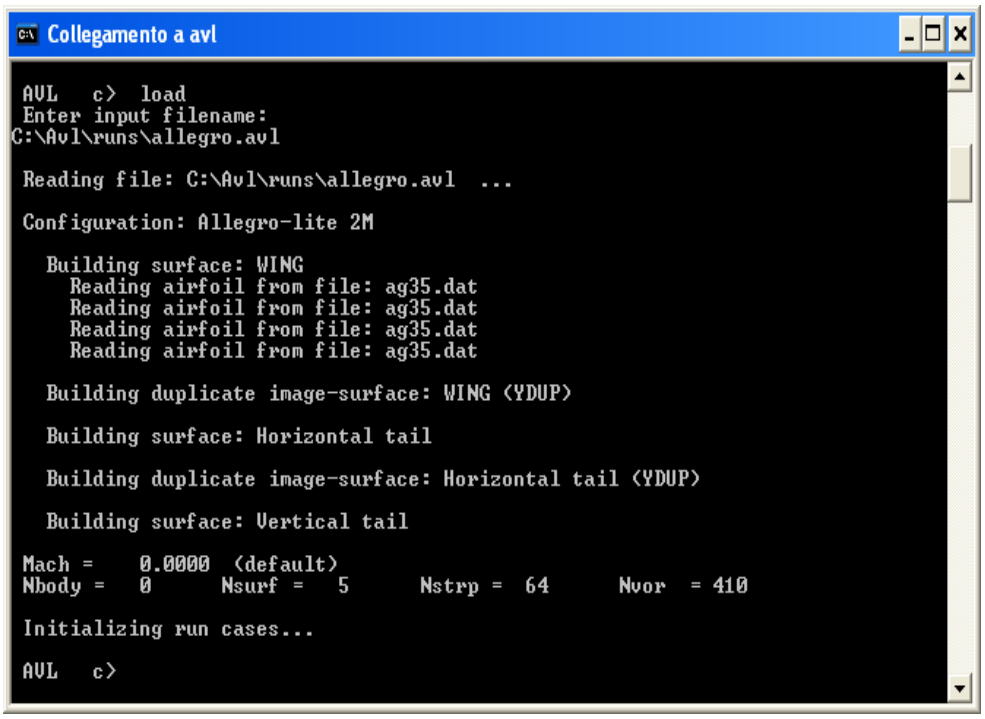

Figura 22

Le scritte che compaiono successivamente, dopo aver spinto invio, fanno capire se il file è stato caricato e/o compilato in modo corretto. Nel caso in cui compaiono delle scritte con error probabilmente non si è compilato il file.avl in modo corretto.

Si digita il comando mass e poi, come fatto in precedenza, si introduce il nome del file in cui è riportata la massa (figura 23).

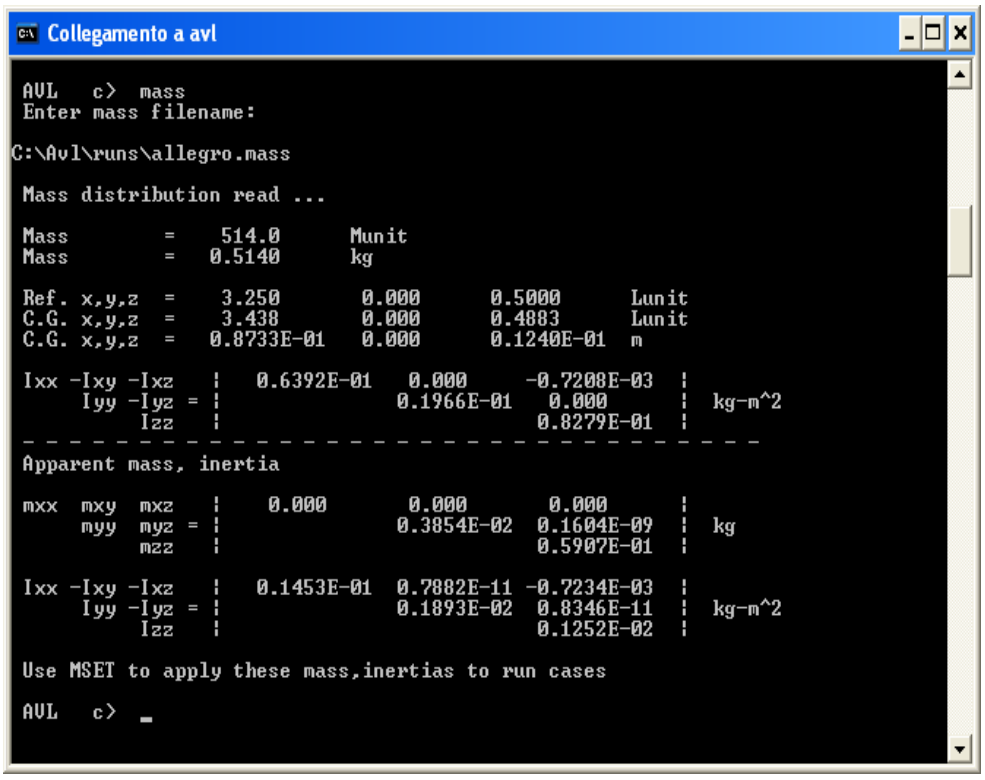

Figura 23

Come mostrato in figura, una volta caricato il file.mass, vengono visualizzati il peso totale, i vari momenti di inerzia e, la cosa più importante, la posizione del baricentro.

Così facendo abbiamo caricato, in modo indipendente, i due file all'interno del programma. Seguendo le istruzioni che vengono suggerite da AVL stesso, si andrà ad associare al primo file, quello riguardante la geometria, il secondo, quello della massa digitando mset e poi 0. Con il comando oper si prepara AVL all'analisi del modello, infatti viene mostrata una finestra con una serie di comandi utili per il calcolo di svariati coefficienti (figura 24).

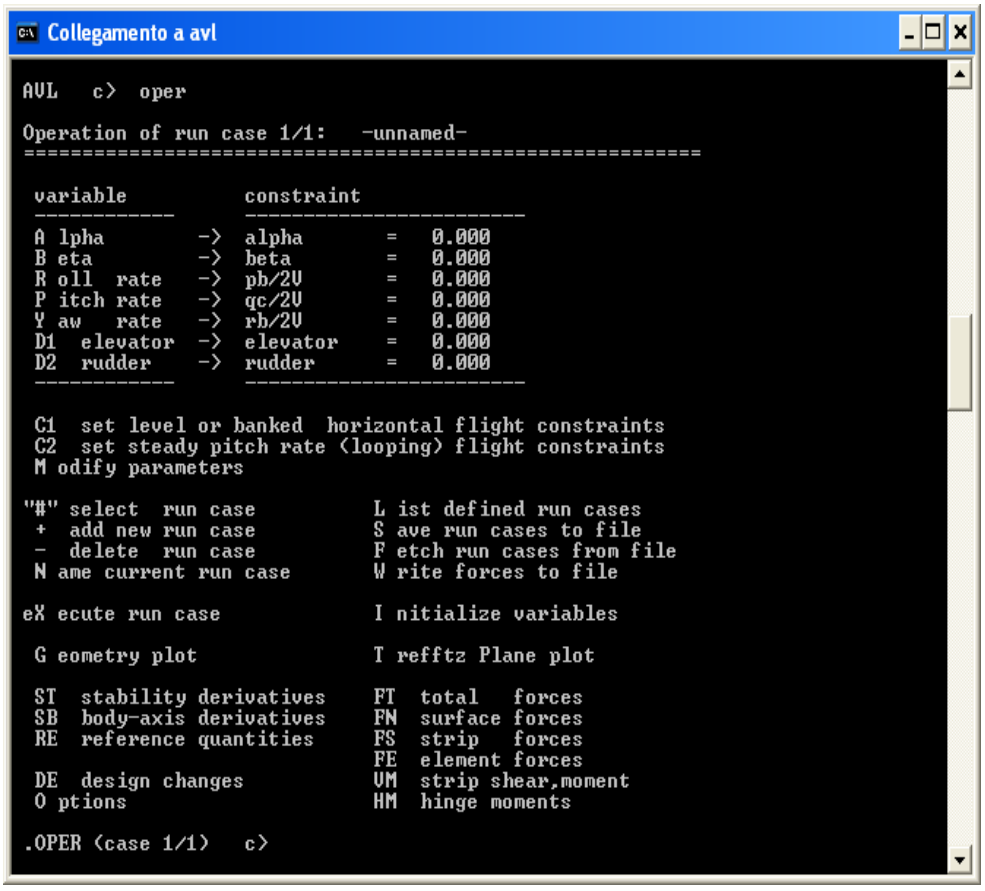

Figura 24

Arrivati a questo punto, se si vuole, è possibile visualizzare la geometria dell'aereo per verificare la correttezza del modello vedendo se assomiglia a quello reale. Digitando, infatti, g e poi k si apre una finestra dove è possibile vedere la forma del velivolo da tutte le angolazioni possibili usando i comandi d (down - giù), u (up - su), r (right - destra), l (left - sinistra).

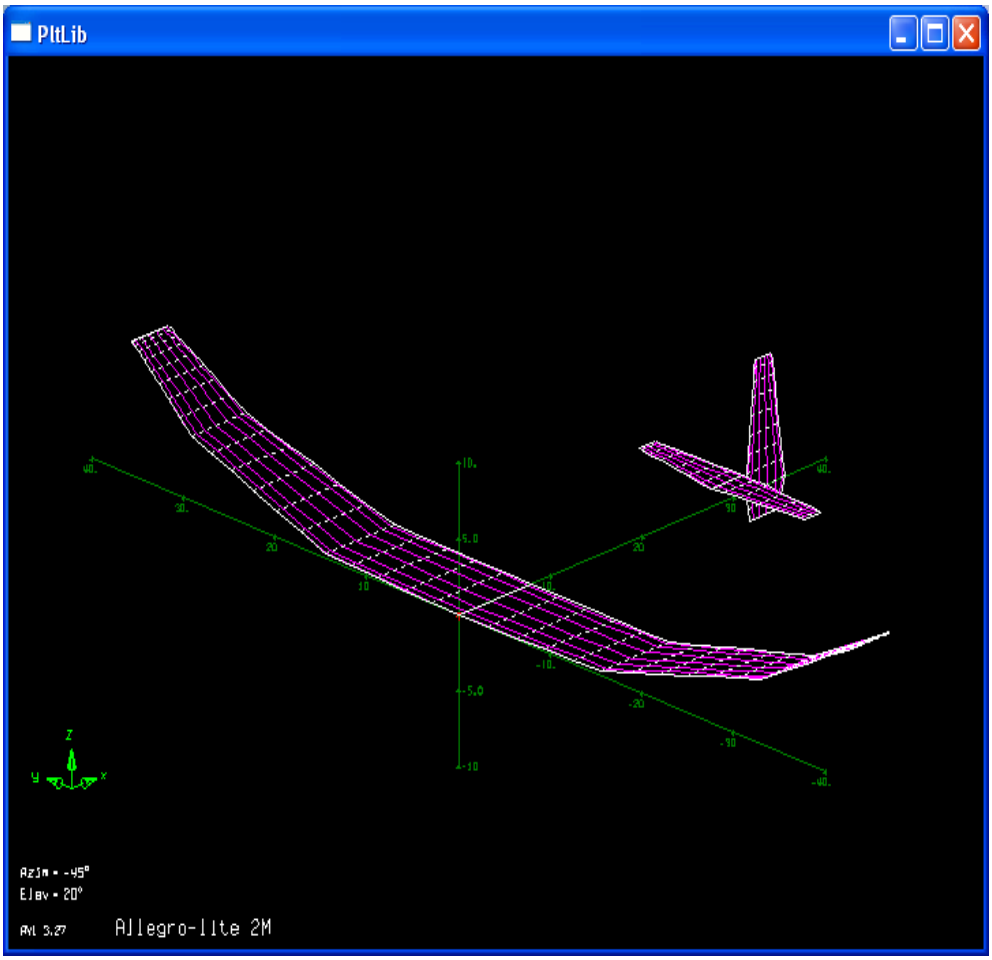

Figura 25

La visualizzazione dell'aereo è utile perché, se per caso questo non rispecchia la realtà, è possibile andare a modificare il file della geometria prima di procedere con l'analisi e trovare così dei coefficienti non corretti, dovendo quindi rifare tutto il procedimento.

Ritornando alla figura 24 si può vedere come siano già presenti tutte le indicazioni per riuscire ad ottenere i coefficienti utili allo scopo desiderato.

3.8.2 COEFFICIENTI DRONE

Dopo aver appreso come utilizzare AVL si prendono i file della geometria (npc.avl) e della massa (npcmassa.mass) del drone, che si sono ottenuti direttamente da XFLR5, e si introducono in esso per il calcolo dei coefficienti.

Prima di procedere nel calcolo si è verificato che i file fossero stati compilati correttamente da XFLR5. Purtroppo si è visto che all'interno del file che riguarda la geometria non è stata riportata la fusoliera, ma solo le superfici aerodinamiche quali ali, elevatore e winglet. Quindi si è dovuto introdurre una parte che riproducesse la fusoliera dell'aereo. Come visto in precedenza ci sono due modi per modellarla. Si è scelto di usare quello che la assimila ad una superficie portante visto che il drone non ha una fusoliera ben definita e in più è il metodo più veloce e non influisce così tanto nel risultato finale.

Fatta questa operazione si possono caricare i due file in AVL e procedere con le simulazioni.

Guardando attentamente la figura 24 si è visto che i coefficienti aerodinamici che dovranno poi essere inseriti in FlightGear sono forniti dal comando "st". Per ottenerli si procede così: si digita x, che fa eseguire l'analisi totale sul modello; poi si digita st, che ti visualizza solo i coefficienti relativi alla stabilità dell'aereo. Questi possono essere anche salvati all'interno di un file digitandone il nome nell'apposita riga come si può vedere in figura 26.

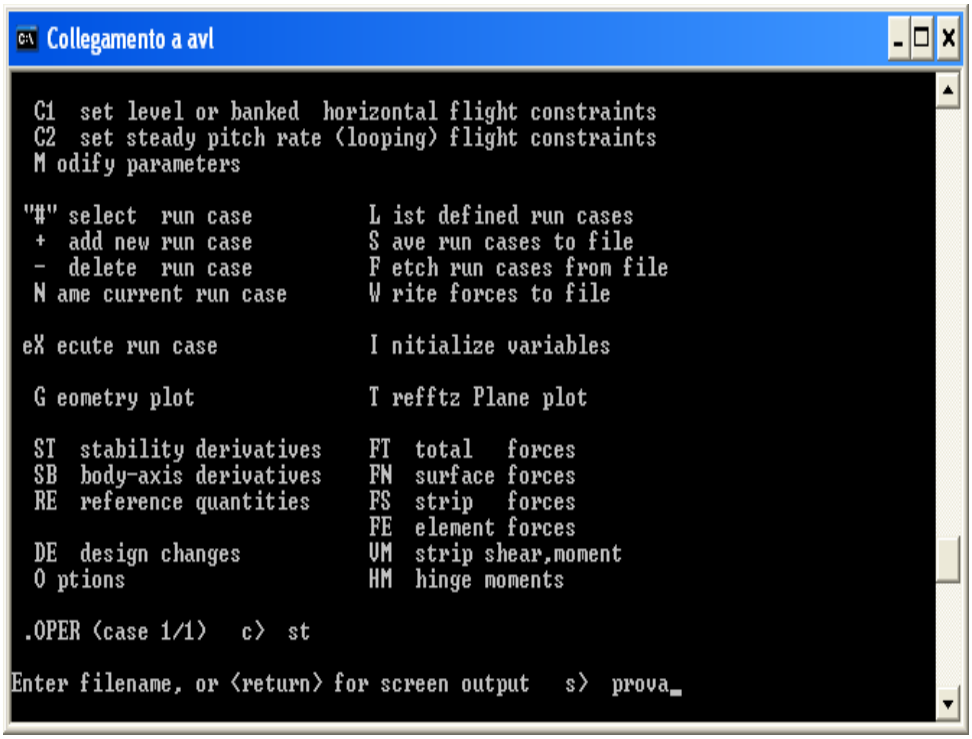

Figura 26

Le simulazioni si è eseguito sono state fatte:

- variando l'angolo alfa di incidenza del velivolo da -10 a 20 gradi;
- l'angolo beta di imbardata da -10 a 10 gradi;
- la posizione degli elevatori da -20 a 20 gradi con angolo alpha di incidenza del velivolo pari a zero;
- la posizione degli alettoni da -20 a 20 gradi sempre ad alpha del velivolo uguale a zero.

Per la prima simulazione si procede variando l'angolo di incidenza direttamente in AVL. Dopo aver digitato oper si inserisce due volte il comando a e infine il valore dell'angolo in gradi (figura 27) poi si procede come descritto in precedenza.

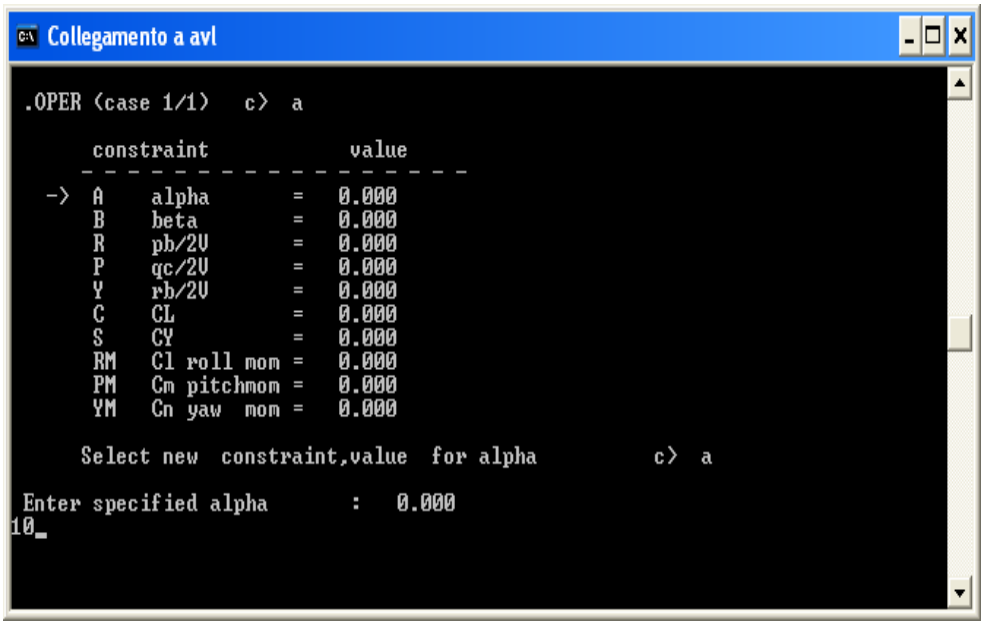

Figura 27

Chiaramente questa operazione va fatta per tutti gli angoli di incidenza della simulazione.

Per la seconda simulazione il procedimento è identico al primo caso solo che invece di digitare a si deve usare il comando b così da riferirsi all'angolo di imbardata beta. Anche in questo caso il procedimento deve essere ripetuto per tutti gli angoli della simulazione.

Per quanto riguarda, invece, gli ultimi due casi non si può procedere come fatto negli altri poiché la deflessione degli elevatori e degli alettoni (che nel nostro aereo coincidono) nella realtà non è omogenea lungo l'apertura alare. Infatti si ha una deflessione minima, di pochi gradi, vicino alla radice dell'ala, mentre la deflessione desiderata si ottiene in corrispondenza dell'estremità della superficie di comando. A causa di ciò si deve creare apposta un file della geometria dell'aereo per ogni angolo di incidenza degli elevatori e degli alettoni per i quali si vogliono eseguire le simulazioni. Purtroppo questa è un'operazione molto laboriosa, anche se abbastanza facile, in quanto ogni volta si deve caricare un file.avl diverso dopo averlo precedentemente creato con XFLR5.

Una volta eseguite tutte le simulazioni si ottengono i dati riportati nelle seguenti tabelle che poi andranno inseriti all'interno del simulatore FlightGear .

Angolo	Alpha (CL)	Beta(CL)	δe (CL)	δa (CL)
-20			-0,2593	0,3046
-19			-0,2248	0,2693
-18			-0,2054	0,2499
-17			-0,1994	0,2458
-16			-0,1931	0,2371
-15			-0,1869	0,2312
-14			-0,1755	0,2197
-13			-0,1611	0,2054
-12			-0,1439	0,1881
-11			-0,1242	0,1685
-10	-0,3469	0,0173	-0,1047	0,1489
-9	-0,3108	0,0182	-0,0842	0,1284
-8	-0,2747	0,0189	-0,0637	0,1079
-7	-0,2383	0,0196	-0,0439	0,0881
-6	-0,2015	0,0202	-0,0265	0,0707
-5	-0,1646	0,0208	-0,0118	0,0559
-4	-0,1275	0,0212	0,0001	0,0441
-3	-0,0902	0,0215	0,0093	0,0347
-2	-0,0529	0,0218	0,0158	0,0282
-1	0,0154	0,0219	0,0199	0,0241
0	0,0219	0,0219	0,0219	0,0219
1	0,0594	0,0219	0,0241	0,0241
2	0,0968	0,0218	0,0282	0,0282
3	0,1341	0,0215	0,0347	0,0347
4	0,1713	0,0212	0,0441	0,0441
5	0,2084	0,0208	0,0559	0,0559
6	0,2452	0,0202	0,0707	0,0707
7	0,2817	0,0196	0,0881	0,0881
8	0,3180	0,0189	0,1079	0,1079
9	0,3540	0,0182	0,1284	0,1284
10	0,3897	0,0173	0,1489	0,1489
11	0,4249		0,1685	0,1685
12	0,4597		0,1881	0,1881
13	0,4941		0,2054	0,2054
14	0,5279		0,2197	0,2197
15	0,5613		0,2312	0,2312
16	0,5941		0,2371	0,2371
17	0,6263		0,2458	0,2458
18	0,6579		0,2499	0,2499
19	0,6888		0,2693	0,2693
20	0,7191		0,3046	0,3046

Angolo	Alpha (CD)	Beta (CD)	δe (CD)	δa (CD)
-20			0,0244	0,0291
-19			0,0211	0,0251
-18			0,0193	0,0231
-17			0,0184	0,0223
-16			0,0177	0,0211
-15			0,0169	0,0203
-14			0,0158	0,0191
-13			0,0146	0,0175
-12			0,0134	0,0161
-11			0,0122	0,0146
-10	0,0313	0,0039	0,0113	0,0133
-9	0,0268	0,0048	0,0105	0,0122
-8	0,0227	0,0056	0,0098	0,0112
-7	0,0191	0,0063	0,0093	0,0104
-6	0,0160	0,0069	0,0091	0,0097
-5	0,0134	0,0074	0,0087	0,0092
-4	0,0113	0,0078	0,0086	0,0089
-3	0,0098	0,0082	0,0085	0,0087
-2	0,0087	0,0084	0,0085	0,0086
-1	0,0083	0,0085	0,0085	0,0085
0	0,0086	0,0086	0,0086	0,0086
1	0,0089	0,0085	0,0085	0,0085
2	0,0100	0,0084	0,0086	0,0086
3	0,0116	0,0082	0,0087	0,0087
4	0,0138	0,0078	0,0089	0,0089
5	0,0165	0,0074	0,0092	0,0092
6	0,0197	0,0069	0,0097	0,0097
7	0,0234	0,0063	0,0104	0,0104
8	0,0276	0,0056	0,0112	0,0112
9	0,0323	0,0048	0,0122	0,0122
10	0,0375	0,0039	0,0133	0,0133
11	0,0431		0,0146	0,0146
12	0,0491		0,0161	0,0161
13	0,0556		0,0175	0,0175
14	0,0624		0,0191	0,0191
15	0,0697		0,0203	0,0203
16	0,0773		0,0211	0,0211
17	0,0852		0,0223	0,0223
18	0,0935		0,0231	0,0231
19	0,1020		0,0251	0,0251
20	0,1108		0,0291	0,0291

3.9 MOTORE ED ELICA DEL DRONE

Il nostro velivolo, per la propulsione, usa un motore elettrico brushless Turnigy Aerodive TR 35-30A 1700kv accoppiato con un elica a due pale di dimensioni 8x4 pollici (8 è il diametro, mentre 4 è il passo).

3.9.1 MOTORI BRUSHLESS

Sono motori elettrici a magneti permanenti. Rispetto ad un tradizionale motore a spazzole, qui non si ha bisogno dei contatti elettrici striscianti sull'albero motore per consentire il funzionamento, poiché la commutazione della corrente circolante negli avvolgimenti avviene elettronicamente e non più per via meccanica (tramite i contatti striscianti). Questo sistema comporta tre grandi vantaggi: minore resistenza meccanica, eliminazione della possibilità di formare scintille al crescere della velocità di rotazione, riduzione della necessità di manutenzione periodica.

3.9.2 SPECIFICHE MOTORE

Il motore in dotazione ha le seguenti caratteristiche tecniche:

tensione = 7.4V ~ 14.8V (2~4S Li-po);
Kv = 1700rpm/V;
dimensioni = 35mm X 30mm;
potenza = 340 W;
corrente = 23;
Rm = 0.075;
no load current = 2A;
peso = 77g;

Per riprodurre le caratteristiche tecniche del motore all'interno di FlightGear lo si può fare compilando due file appositi. In uno vengono riportate le informazioni che riguardano l'elica (il diametro, il numero delle pale, l'angolo di calettamento), il numero dei giri del motore e le prestazioni del motore e dell'elica attraverso due tabelle dove sono riportati,

rispettivamente, il Cp (coefficiente di potenza) e il Ct (coefficiente di spinta) in funzione del rapporta di avanzamento J. Nell'altro file invece si deve inserire esclusivamente la potenza, espressa in watt, del motore.

3.9.3 CALCOLO PRESTAZIONI MOTORE

Per procedere nel calcolo delle prestazioni del motore in accoppiamento all'elica scelta si è ricorsi all'uso di una serie di formule utili a riuscire a calcolare i coefficienti che si dovranno poi inserire nel modello di FlightGear .

La prima cosa da fare è capire a quanti giri al minuto giri il motore sfruttando l'equazione

$$n = Kv * (V – (I * Rm)) \qquad (1)$$

n → numero di giri al minuto a vuoto

V → tensione di alimentazione del motore

I → corrente di alimentazione del motore

Così si trova il numero di giri del motore a vuoto, cioè senza aver montato l'elica. Partendo da questo valore si andrà a vedere a quanti giri, effettivamente, con l'elica 8x4 montata gira al massimo della potenza erogata. Per farlo si può usare al seguente equazione

$$P = Kp * (d^4) * p * (n^3) \qquad (2)$$

P → potenza

Kp → costante riferita al tipo di elica montata. Se non se ne conosce il valore si può usare

tranquillamente il valore 1,25.

d → diametro elica espresso in piedi

p → passo elica espresso in piedi

n → giri al minuto espressi in migliaia (es. 1300 giri inserire 1,3)

Importante notare che questa formula la si può usare solo nel caso di un'elica a due pale, come in questo caso, e che la potenza che si trova è quella necessaria per far girare l'elica al numero di giri inserito.
Com'è facilmente intuibile se si inserisce nella (2) il valore di n ricavato dalla (1) si ottiene una potenza superiore a quella disponibile. Questo vuol dire che il motore, al massimo della potenza girerà ad un numero di giri inferiore a quello a vuoto.
Per trovare il valore esatto si può sfruttare un'altra relazione

$$P = ((V - (I * Rm)) * (I - Io)) \qquad (3)$$

Questa fornisce la potenza massima effettiva disponibile all'albero del motore, che dovrà coincidere con quella che si ottiene dalla (2). Come già detto in precedenza il valore della potenza che si ottiene dalla (2) è superiore a quello ottenuto dalla (3). Questo succede perché si è inserito nella (2) il numero massimo di giri a vuoto, quindi ora partendo da questo valore si andrà a diminuire il numero dei giri fino a quando non si ottiene lo stesso valore di potenza dalle due formule. In valore così ottenuto sarà il numero massimo di giri che il motore compie con quella determinata elica.
Ora è possibile andare a calcolare i valori del coefficiente di potenza Cp e del coefficiente di spinta Ct servendosi della relazione per il primo della relazione (4)

$$P = Cp * \rho * n^3 * d^5 \qquad (4)$$

$P \rightarrow$ potenza teorica = $V * I$
$\rho \rightarrow$ densità dell'aria
$n \rightarrow$ giri al secondo del motore
$d \rightarrow$ diametro elica espresso in metri

e per il secondo della relazione (5)

$$T = Ct * \rho * n^2 * d^4 \qquad (5)$$

T → trazione
n → giri al secondo del motore
d → diametro elica espresso in metri

Per quello che riguarda la relazione (5) non si conosce ancora il valore della trazione T e quindi, di conseguenza, non si possono calcolare i valori del Ct. A questo si può rimediare abbastanza facilmente servendosi di una relazione semi-empirica

$$T = (P^2 * \rho * (d/2)^2 * \pi)^{(1/3)} \qquad (6)$$

d → diametro dell'elica espresso in metri

Questa relazione fornisce un'ottima stima e i valori ottenuti non si discostano molto da quelli reali e quindi si è pensato di usare i risultati di trazione ottenuti in quanto anche se ci fossero dei piccoli errori di calcolo questi non influenzeranno più di tanto il comportamento finale del velivolo.
L'ultimo valore da calcolare è il rapporto di avanzamento J e lo si fa sfruttando la seguente relazione:

$$J = v / (n * d) \qquad (7)$$

v → velocità del velivolo rispetto al suolo = P / T
n → giri al minuto del motore
d → diametro elica espresso in metri

Per ottenere dei calcoli più rapidi si sono riportate queste formule all'interno di un script di Matlab (motore.m) e si sono ottenuti i seguenti risultati:

J	Cp	Ct
0,006955	0,057086	0,136788
0,00695	0,056956	0,136582
0,006946	0,056852	0,136415
0,006943	0,056777	0,136294
0,006941	0,056734	0,136225
0,006941	0,056729	0,136218
0,006943	0,05677	0,136283
0,006946	0,056862	0,136431
0,006953	0,057016	0,136677
0,006962	0,057245	0,137042
0,006975	0,057562	0,137547
0,006992	0,057986	0,138223
0,007014	0,058543	0,139106
0,007043	0,059264	0,140246
0,007079	0,06019	0,141704
0,007126	0,061379	0,143564
0,007184	0,062909	0,145939
0,007259	0,064888	0,148984
0,007354	0,067473	0,152916
0,007477	0,070899	0,158049
0,007636	0,075523	0,164848
0,007845	0,08192	0,17403
0,008127	0,091056	0,18674
0,008513	0,104666	0,204912
0,009059	0,126123	0,232038
0,009862	0,162718	0,274992
0,011112	0,232757	0,349112
0,01324	0,393748	0,495651
0,017515	0,911517	0,867374

Una cosa importante da notare è che non si è potuto tenere conto della presenza di un dispositivo ESC. Questo è un dispositivo regola la velocità di rotazione del motore a seconda della posizione del comando, ed in più ha anche la funzione di prediligere il funzionamento dei servi rispetto a quello del motore allo scaricarsi delle batterie. Purtroppo però non si è in grado di sapere il suo esatto funzionamento, cioè com'è stato programmato, ne si riesce a riprogrammalo in quanto ci è stato fornito senza nessun tipo di informazione da parte del costruttore dell'aereo. Questo purtroppo comporta degli errori nei calcoli che non possono essere corretti in quanto il suo funzionamento non sarà sicuramente lineare come invece si è supposto.

3.10 RIPRODUZIONE DRONE IN FLIGHTGEAR

Si è riusciti a calcolare o a stimare attraverso degli appositi programmi tutti i coefficienti necessari per creare in FlightGear , attraverso JSBSim, una riproduzione del drone per riuscire a fare qualche simulazione. Prima di procedere con la modifica della struttura del Malolo1 si precisa che sia per la complessità sia per la non influenza sugli scopi della tesi non si è andati a modificare il modello visivo del velivolo, cioè si è mantenuto l'aspetto del Malolo1 ma si è cambiato solamente quello che riguarda la sua struttura fisica.

3.10.1 MODIFICA MALOLO1

Arrivati a questo punto non resta che cambiare i valori relativi alle caratteristiche del Malolo1 con quelli calcolati relativi al drone. Purtroppo però non è un'operazione così semplice come sembrerebbe e tra poco si vedrà subito meglio cosa si intende.

Per prima cosa si entra nella cartella Malolo1 e si apre, attraverso WordPad, il file Malolo1. Questo file è quello in cui vengono riportate tutte le caratteristiche geometriche, massiche e aerodinamiche del velivolo. Si procederà ora a modificare il file sezione per sezione partendo da quelle più semplici da modificare.

Si parte dai valori delle caratteristiche geometriche. Nella figura 28 si sono riportati quelli di partenza relativi al Malolo1.

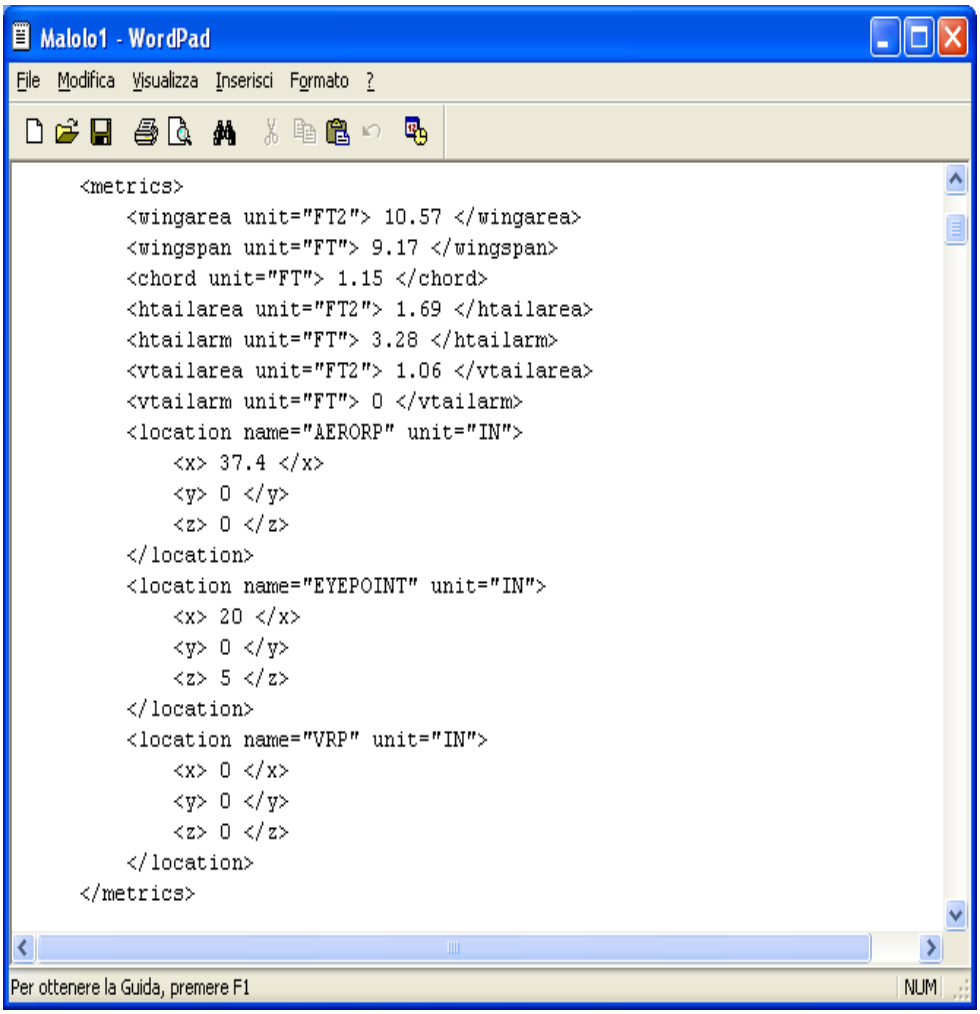

Figura 28

Come si può facilmente notare, pur essendo visualizzato come un aereo tutt'ala (Flying wings), la sua struttura in JSBSim è quella di un aereo tradizionale essendo presenti i piani di coda. Perciò la prima cosa che si è fatto e stata quella di eliminare questo particolare e poi di sostituire gli altri valori con quelli relativi al drone come si può vedere in figura 29.

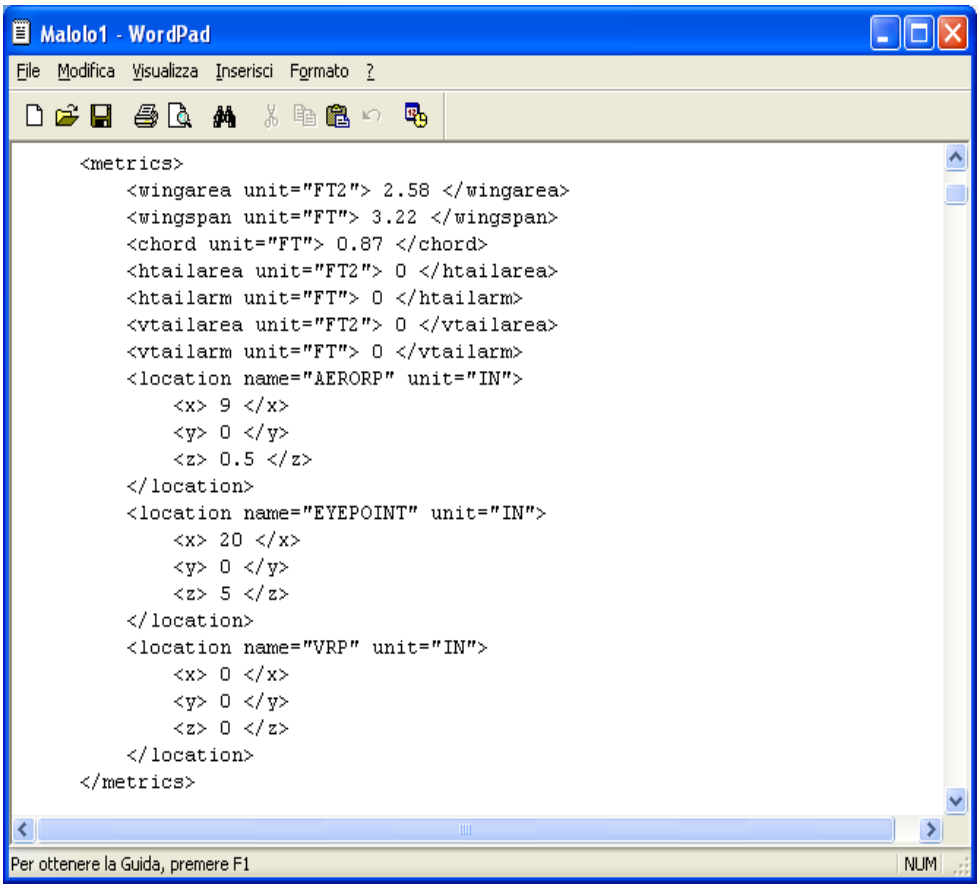

Figura 29

Un'altra cosa interessante che si è subito notata è la presenza dal carrello pur questo non essendo visibile in FlightGear . Si è così subito pensato di eliminarlo poiché anche l'aereo reale ne è privo. Purtroppo questa operazione non è consentita da FlightGear perché questo simulatore non è stato creato appositamente per riprodurre questo tipo di aerei, ma aerei più convenzionali e quindi per ora si dovrà comunque inserire il carrello anche se il drone ne è privo.

Si proseguirà andando a modificare tutti i valori che riguardano il peso, la posizione del baricentro e i momenti di inerzia.

Rimangono da modificare i coefficienti aerodinamici, purtroppo non basta solo sostituire dei numeri, come si è fatto fino ad ora, poiché in alcuni casi si deve proprio modificare la struttura stessa del file. Ma procediamo per gradi.

Per quanto riguarda quei coefficienti aerodinamici in cui è presente una tabella dove si riporta il valore al variare dell'angolo di incidenza, come per

il coefficiente di portanza figura 30, basta tranquillamente sostituire i valori con quelli trovati in precedenza.

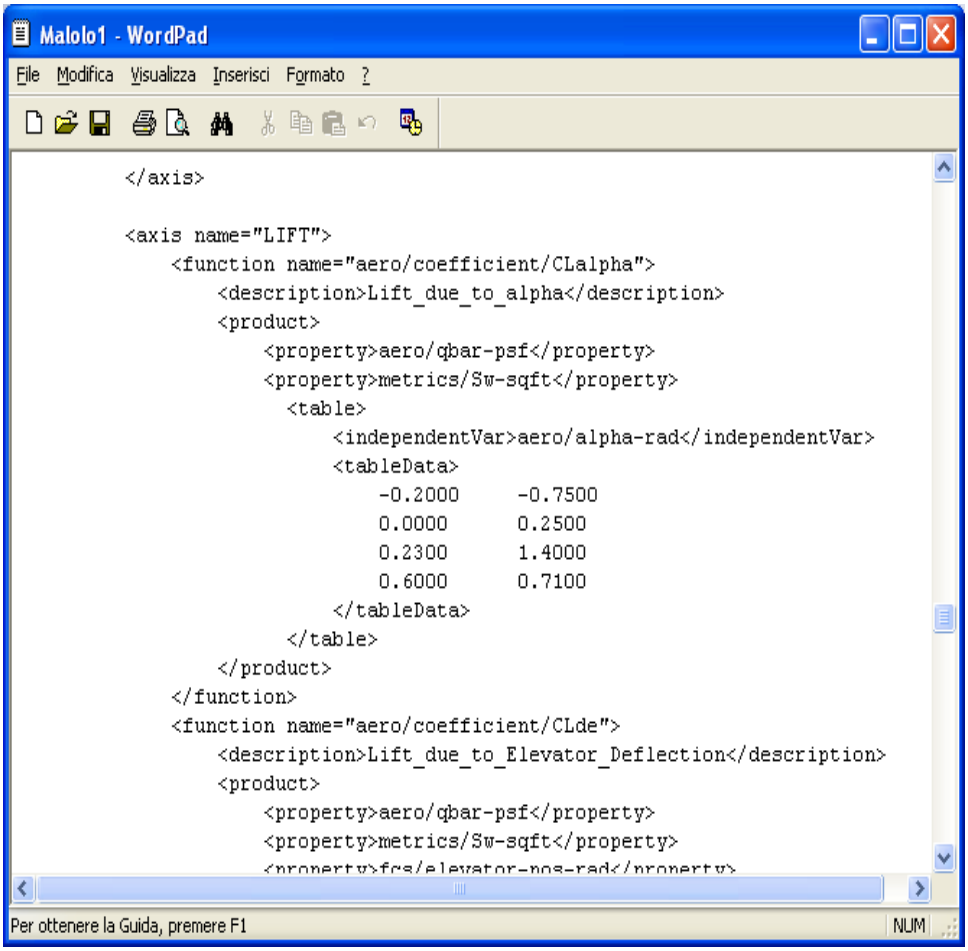

Figura 30

Nel caso in cui i coefficienti siano espressi da un solo valore, come nel caso del coefficiente di portanza al variare dell'angolo di elevatore CLde (figura 31), si dovrà modificare la struttura in modo da poter inserire una tabella analoga a quella del CL come si può vedere nella figura 32.

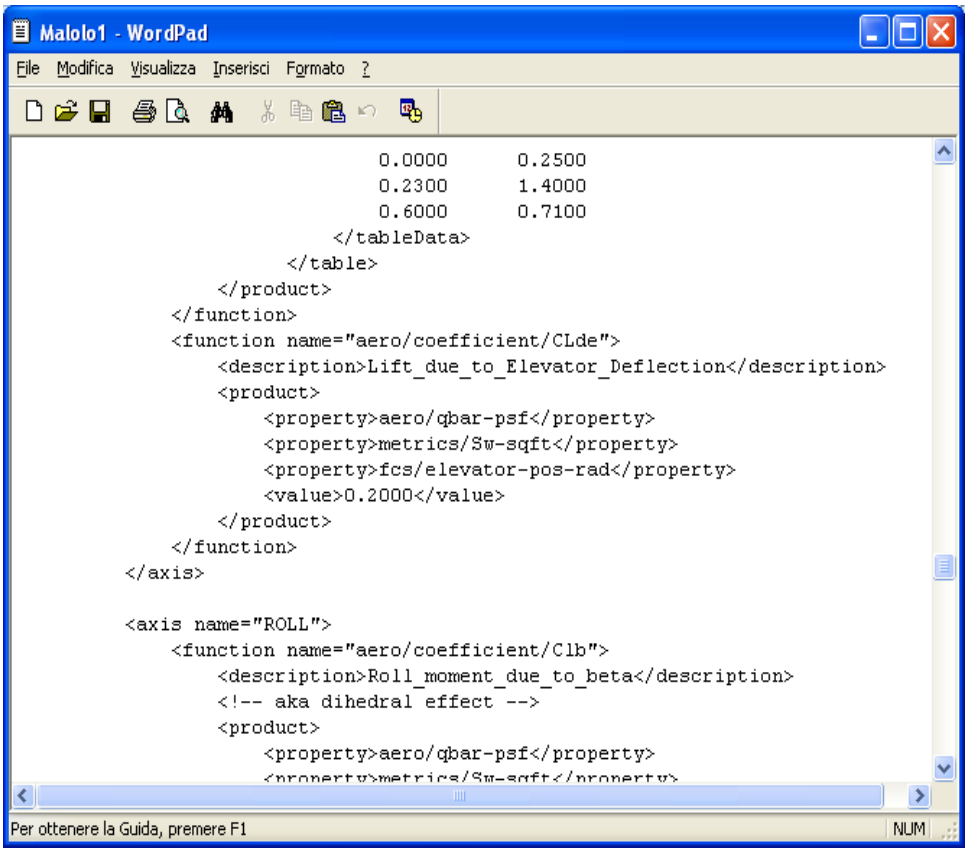

Figura 31

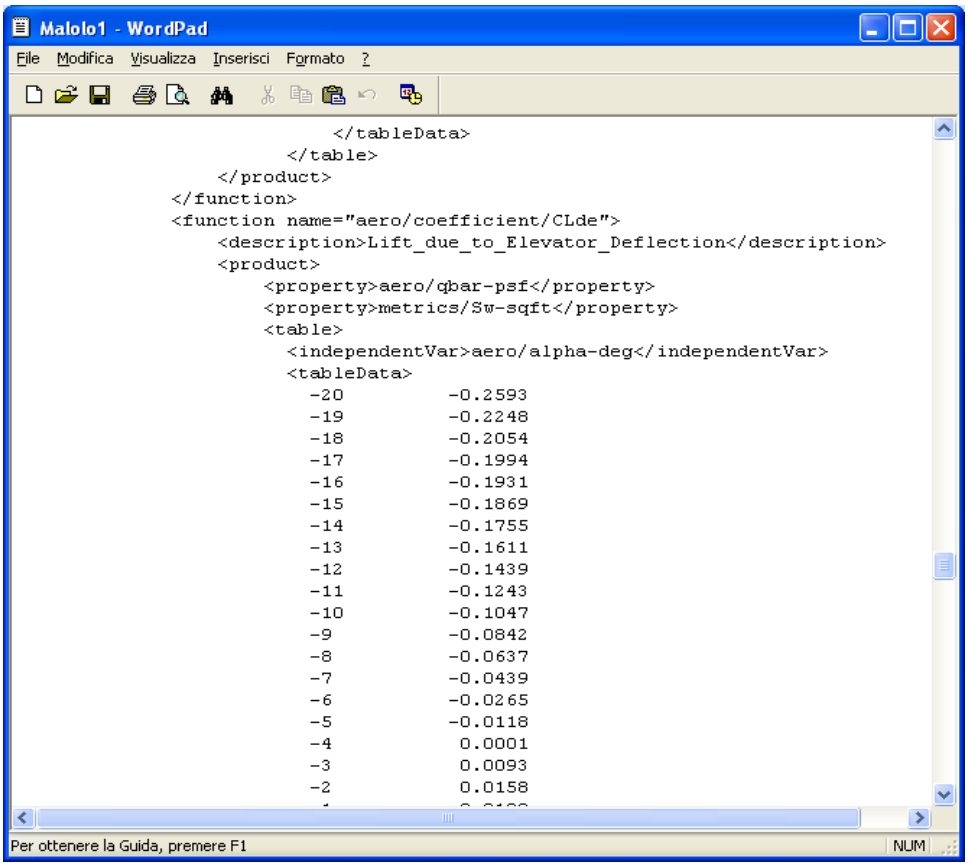

Figura 32

Modificati tutti i coefficienti come descritto si prosegue andando nella cartella Models e aprendo il file Malolo1. Qui si dovrà andare a modificare l'escursione degli alettoni e degli elevatori, che in questo caso coincidono, da quindici a venti gradi. Per fare questa operazione basta sostituire il numero 15 con 20 in tutte le righe in cui compare <factor>15</factor>.

L'ultima modifica da eseguire riguarda la parte motoristica. Si va nella cartella Engines, si apre il file 18x8 e si va ad inserire i valori di Ct e Cp in funzione di J trovati in precedenza e i valori relativi all'elica in dotazione.

Terminata quest'ultima modifica il modello del drone in FlightGear è completato e pronto per la simulazione. Prima di questa però si deve riprodurre, in Simulink, gli autopiloti che sono presenti sul drone.

Capitolo 4

AUTOPILOTI

Solitamente con la parola autopiloti ci si riferisce ad un concetto più ampio rispetto a quello che è realmente. Infatti l'autopilota vero e proprio è un sistema che permette il mantenimento dell'assetto del velivolo, mentre quando si parla comunemente di autopiloti ci si riferisce al sistema di guida, che sfruttando gli autopiloti permette di compiere una missione specifica (es. mantenere la quota).
Gli autopiloti che si trovano sul drone sono riportati all'interno di file di Arduino che una parte di Ardupilot mega.

4.1 ARDUINO

Arduino è una piattaforma open - source per la prototipizzazione di dispositivi elettronici basata su hardware e software flessibili e facili da usare. È destinato ad artisti, designer, appassionati e chiunque sia interessato a creare oggetti e ambienti interattivi.
Può percepire l'ambiente attraverso la ricezione di input proveniente da una varietà di sensori e influenzarlo attraverso controlli di luci, motori e altri attuatori. Il microcontrollore sulla scheda è programmabile usando il linguaggio di programmazione di Arduino (basato su Wiring) e il suo ambiente di sviluppo (basato su Processing). I progetti di Arduino possono essere indipendenti o possono comunicare con software in esecuzione su un computer (es. Flash, Processing, MaxMSP).

4.2 AUTOPILOTI DEL DRONE

Aperti i file di Arduino si è visto che i sistemi di guida, che dovranno essere riprodotti nel file di Simulink di figura 5, sono due:

- mantenimento della quota.
- possibilità di compiere un percorso prestabilito intercettando dei waypoint.

Questi sfruttano tre autopiloti:

- mantenimento angolo di rollio
- mantenimento angolo di beccheggio
- automanetta

Il primo sistema di guida ha due modalità: una è quella di mantenere la quota a cui sta volando l'aereo; l'altra permette al drone di raggiungere una quota prestabilita e di mantenerla nel proseguo del volo. Tutte e due le modalità sfruttano l'accoppiamento degli ultimi due autopiloti.
Il secondo sistema di guida, invece, permette il raggiungimento, in modo autonomo da parte dell'aereo di uno o più waypoint prestabiliti sfruttando solo il primo autopilota.
La struttura di entrambi gli autopiloti prevede la presenza di regolatori PID.

4.2.1 REGOLATORE PID

Nel controllo di molti processi soprattutto industriali, come quelli relativi a impianti chimici e petrolchimici, le caratteristiche dinamiche dei sistemi controllati possono variare entro ampi limiti (ad esempio, un controllo di portata ha in generale una risposta molto più pronta di un controllo di temperatura), mentre d'altra parte risulta economicamente conveniente unificare gli apparati di controllo. Le variabili manipolabili di questi sistemi sono in genere le posizioni di steli di valvole, che vengono variate con attuatori pneumatici o elettropneumatici, collegati ad apparati di controllo standard, ad amplificazione pneumatica od elettronica; tali apparati sono provvisti di opportune manopole di regolazione, agendo sulle quali si possono modificare i valori dei parametri che ne caratterizzano il comportamento, in modo da poterli facilmente adattare alla dinamica del sistema controllato ed ottenere così dal sistema complessivo in retroazione una risposta soddisfacente. In altre parole, si dispone di apparati di controllo standard, ma provvisti di dispositivi di correzione con parametri regolabili

entro ampi limiti, così da poter essere adattati al particolare sistema di regolazione in cui vengono inseriti.

[figura schema a blocchi di un controllo in retroazione pag 228 libro controlli]

Si distinguono i seguenti regolatori di tipo standard:

- Regolatore proporzionale (P)
- Regolatore integrale (I)
- Regolatore proporzionale – integrale (PI)
- Regolatore proporzionale – derivativo (PD)
- Regolatore proporzionale – integrale – derivativo (PID)

Il regolatore standard usato all'interno degli autopiloti presenti sull'NPC sono di tipo PID e la sua funzione di trasferimento è

$$G(S) = K_p \left(1 + T_d S + \frac{1}{T_i S} \right)$$

La costante Kp si dice sensibilità proporzionale, Td costante di tempo dell'azione derivativa, Ti costante di tempo dell'azione integrale.

4.2.2 DATI DI VOLO NECESSARI AGLI AUTOPILOTI

Affinché si possano implementare gli autopiloti in Simulink è necessario riuscire ad estrarre da FlightGear tutti i dati che essi richiedono in ingresso. Per ottenere un loro corretto funzionamento bisogna conoscere latitudine, longitudine, altezza, assetto, prua, velocità dell'aria, velocità a terra e posizione della manetta motore. Si dovrà, quindi, andare a modificare il file protocol Ardupilot di FlightGear e il file prova.m di Matlab per permettere di riuscire ad avere tutte le informazioni necessarie affinché gli autopiloti possano restituire in uscita le posizioni dei comandi (alettoni, elevatore e posizione manetta) e riuscire così a controllare l'aereo.

4.3 REALIZZAZIONE AUTOPILOTI IN SIMULINK

Come si è visto gli autopiloti (da ora in avanti ci si riferisce al significato più comune, cioè l'insieme autopilota più sistema di guida) usano all'interno della loro struttura dei regolatori PID che elaborano il segnale in ingresso, differenza tra un riferimento e il valore istantaneo, e restituiscono un'uscita che permette il raggiungimento del valore desiderato o di riferimento.

Da tenere sempre ben presente che l'uscita del PID, affinché restituisca un valore veritiero, andrà moltiplicata per uno o più opportuni valori di scala.

Poiché lo scopo principale, per questo tipo di aerei, è quello di riuscire a raggiungere dei waypoint prestabiliti in modo autonomo, si è pensato di iniziare a riprodurre in Simulink l'autopilota che permette di compiere questa specifica missione.

Guardando la sua struttura in arduino si vede chiaramente che è composta da due regolatori PID. Il primo ha come ingresso la differenza tra l'angolo che c'è tra la posizione dell'aereo rispetto al waypoint e la prua e restituisce in uscita l'angolo di rollio che serve per riuscire a raggiungere il waypoint. Presa questa uscita e fatta la differenza con l'angolo di rollio dell'aereo si ha l'ingresso del secondo PID che restituisce come uscita la posizione della superficie di comando, in questo caso sono gli alettoni, che consenta all'aereo il raggiungimento del waypoint (figura 33). Da sottolineare che si è impostato come angolo massimo di rollio raggiungibile durante la virata a un valore di 25 gradi.

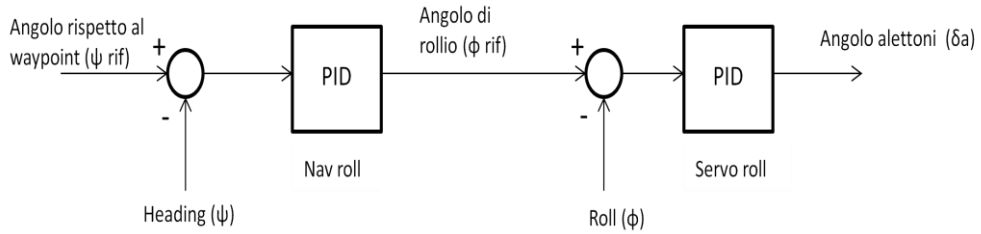

Figura 33

Riprodurre questo schema in Simulink è piuttosto semplice poiché basta usare i blocchi funzione già esistenti in esso. L'unica complicazione sta nel calcolare la differenza tra l'angolo del waypoint rispetto all'aereo e la prua di quest'ultimo in quanto il primo va da -180 a +180 gradi, mentre il

secondo va da 0 a 360, che non è un'operazione così banale. Infatti si è dovuto creare una S-function (angolorif.m) che calcoli l'esatto valore della differenza tra i due angoli a seconda di come sono orientati reciprocamente.
Tale autopilota, formante un unico blocco, consente di andare a intercettare un solo waypoint perché ha come riferimento la posizione di un solo waypoint, poi una volta verificato il corretto funzionamento di tutti gli autopiloti si andrà a creare una funzione che permetta il cambio del waypoint una volta raggiuntone uno.
Terminata la costruzione dell'autopilota si è fatto partire una prima simulazione per vedere se, effettivamente, tutto sia stato riprodotto correttamente e funzioni come dovrebbe. Purtroppo però si è subito visto che il blocco che comunica con FlightGear crea problemi a causa della presenza della retroazione poiché si viene a formare un loop algebrico. Per risolvere questo problema si è pensato di sdoppiare il blocco in due differenti così da non andare ad influire sull'azione dell'autopilota introducendo un blocco delay su ogni uscita che avrebbe portato ad un ritardo di risposta da parte degli autopiloti.
Quindi ora al posto di un solo blocco se ne avranno due che svolgono però la stessa funzione, cioè uno ha solo le uscite (costruito utilizzando la S-function prova2.m) e l'altro presenta solo gli ingressi (costruito utilizzando la S-function prova1.m). Questo sdoppiamento non porta nessuna modifica reale in quanto rimane comunque la retroazione data dagli autopiloti.
Risolto questo problema si può passare a riprodurre il restante autopilota.
Per quello che riguarda il mantenimento della quota si è visto che l'autopilota è formato da due parti: una che agisce sull'elevatore e quindi controlla l'angolo di beccheggio dell'aereo, mentre l'altra agisce sulla manetta del motore e quindi controlla la potenza erogata. L'accoppiamento di questi due controlli consente di riuscire a mantenere una determinata quota di riferimento.
Poiché questo autopilota ha due modalità di funzionamento si è deciso di riprodurre per primo la modalità che fissa la quota di volo che è una delle principali richieste per questo tipo di aereo.
In questo caso solo l'autopilota che controlla il beccheggio usa come regolatore due PID mentre l'automanetta si basa esclusivamente su una

espressione algebrica. Il primo PID prende in ingresso la differenza tra la quota desiderata e quella attuale del velivolo e da in uscita l'angolo di pitch che teoricamente serve per colmare la differenza di quota. Questo angolo andrà sia in ingresso all'espressione algebrica dell'automanetta ottenendo in uscita la posizione della manetta motore sia al secondo PID dell'autopilota che controlla il beccheggio. Chiaramente questo angolo di pitch non entra direttamente nel PID ma viene fatta la differenza con l'angolo di pitch effettivo dell'aereo e in più si va a sommare l'angolo di rollio per compensare il fatto che durante una virata si perde quota. In uscita si avrà finalmente l'angolo di elevatore per cui si riesce a raggiungere la quota impostata (figura 34).

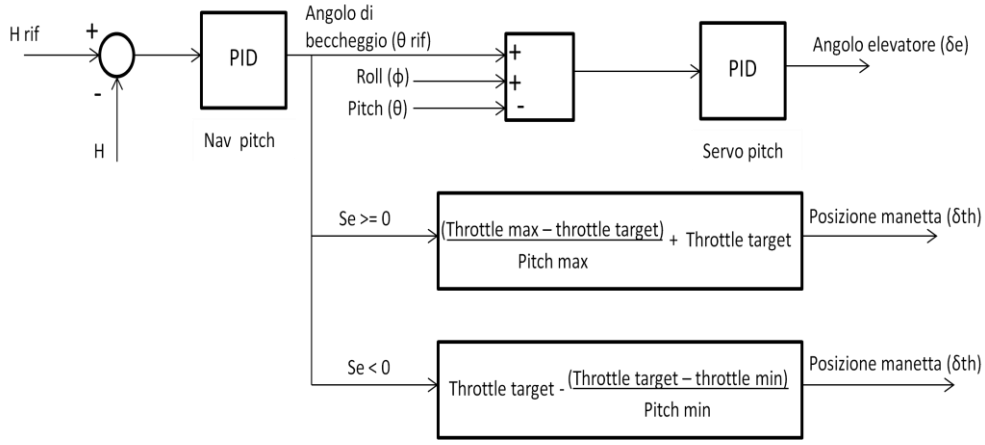

Figura 34

Ora si va a vedere come è strutturata la modalità che mantiene la quota alla quale l'aereo sta volando. La parte che riguarda il controllo degli elevatori prende la differenza tra l'airspeed cruise e la airspeed dell'aereo e passa attraverso un primo PID il quale restituisce un valore dell'angolo di pitch utile a mantenere la quota. Ora come nell'altra modalità questo valore va a fare la differenza con l'angolo di pitch dell'aereo, con l'aggiunta dell'angolo di rollio, e poi entra nel secondo PID che restituisce l'angolo di elevatore che permette il mantenimento della quota. Per quanto riguarda l'automanetta, nella seconda modalità, il suo funzionamento è più complicato, infatti si prende l'airspeed e l'airspeed cruise e vengono inserite in una espressione matematica che poi viene inviata ad un PID, il cui

segnale di uscita aggiunto alla throttle cruise restituisce la posizione della manetta motore (figura 35).

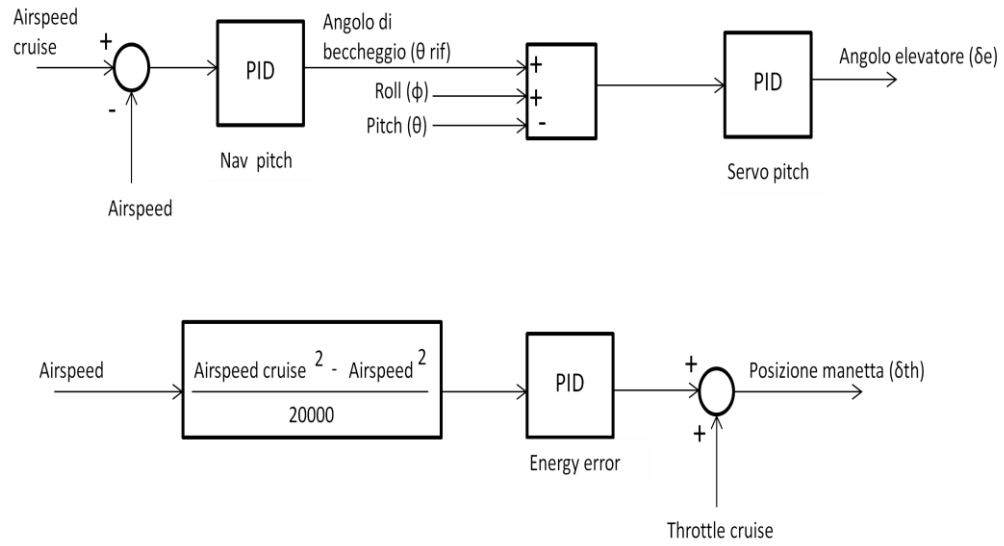

Figura 35

Nello schema in Simulink per poter scegliere tra le due modalità si è inserito due manual switch che chiaramente devono essere cambiati insieme altrimenti si pregiudica il corretto funzionamento degli autopiloti stessi.
Una volta verificato anche il loro funzionamento si è in grado di far fare al velivolo quello richiesto. L'ultima cosa che manca è quella di riuscire a fargli intercettare più di un waypoint.
L'idea di base è che una volta raggiunto un waypoint, cioè che l'aereo sia entro un cerchio di un certo diametro prestabilito, si passi al successivo. Affinché questo si possa verificare bisogna riuscire prima di tutto a calcolare la distanza, in tempo reale, che intercorre tra l'aereo e il waypoint.
Riuscire a calcolare la distanza conoscendo la latitudine e la longitudine, sia dell'aereo che del waypoint, non è affatto semplice perché non si può usare un banale teorema di Pitagora, anche se comunque le distanze in gioco nella simulazione sono piccole, in quanto bisogna tenere conto che la terra non è piatta. Quindi il calcolo della distanza si effettua attraverso una funzione trigonometrica, riportata in arduino, che conoscendo le coordinate di latitudine e longitudine del waypoint e dell'aereo restituisce la loro distanza.

Riusciti a riprodurre questa funzione in Simulink e quindi a calcolare la distanza si può passare a creare un blocco, utilizzando una S-function (target.m), che permetta di cambiare waypoint una volta che la distanza è inferiore a 30 metri e che intercettato anche l'ultimo waypoint fa ricominciare il giro ripartendo dal primo.

Questo blocco come ingressi oltre alle coordinate (latitudine e longitudine) di tutti i waypoint presenta anche la distanza, mentre in uscita si avrà le coordinate del waypoint che l'aereo sta cercando di raggiungere. Poiché l'uscita oltre ad andare in ingresso all'autopilota va anche al blocco destinato al calcolo della distanza si viene a creare, anche in questo caso, un loop algebrico che non permette la simulazione dando errore. Si è risolto facilmente mettendo un delay all'uscita del blocco che calcola la distanza.

La S-function target, e quindi il blocco associato, prende in ingresso quattro waypoint, ma possono essere anche di più, più la distanza e restituisce in uscita il waypoint che l'aereo sta andando ad intercettare.

Capitolo 5

SIMULAZIONI

Terminata la riproduzione del drone e dei suoi sistemi di controllo (gli autopiloti) si può avviare una prima simulazione di prova e vedere così il risultato del lavoro svolto.
In questa prima simulazione si è utilizzato, per la quota, l'autopilota nella modalità in cui si fissa un valore di riferimento che dovrà essere raggiunto e mantenuto durante tutta la simulazione.
Durante tutta la simulazione i dati di volo vengono salvati nel Workspace di Matlab così da essere consultabili una volta terminata.
Ottenuti tutti i dati di questa prima simulazione si vuole vedere, per prima cosa, se effettivamente si sono intercettati tutti i waypoint prestabiliti. Attraverso la funzione plot di Matlab riusciamo a ricostruire il percorso compiuto dall'aereo con l'aggiunta di tutti i waypoint (segnalati con un asterisco) riportato nella figura 36.

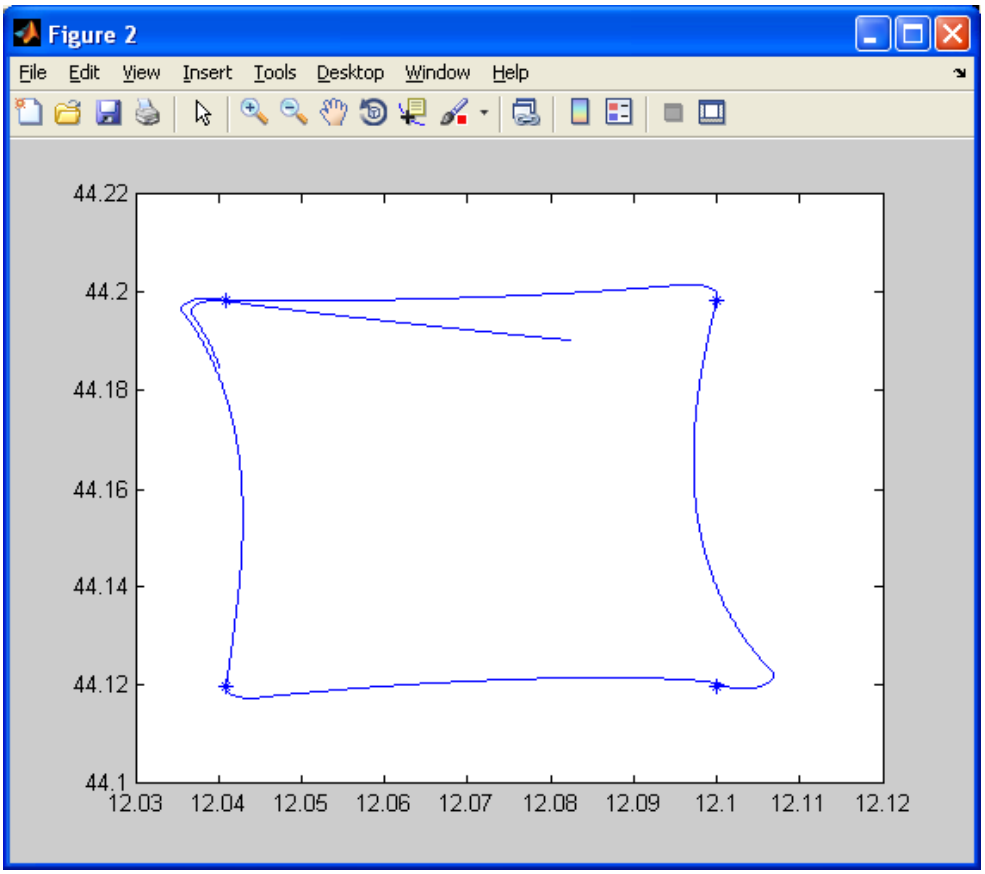

Figura 36

Si nota subito che tutti i waypoint sono stati intercettati e che il percorso seguito dal drone tra un waypoint e l'altro, dopo le virate, non segue una linea retta, come logica vorrebbe, ma prosegue sempre lungo una traiettoria curva.

La seconda cosa importante da verificare è l'effettiva possibilità di riuscire a raggiungere e mantenere una quota prefissata, posta a 350 piedi, per tutta la durata del volo.

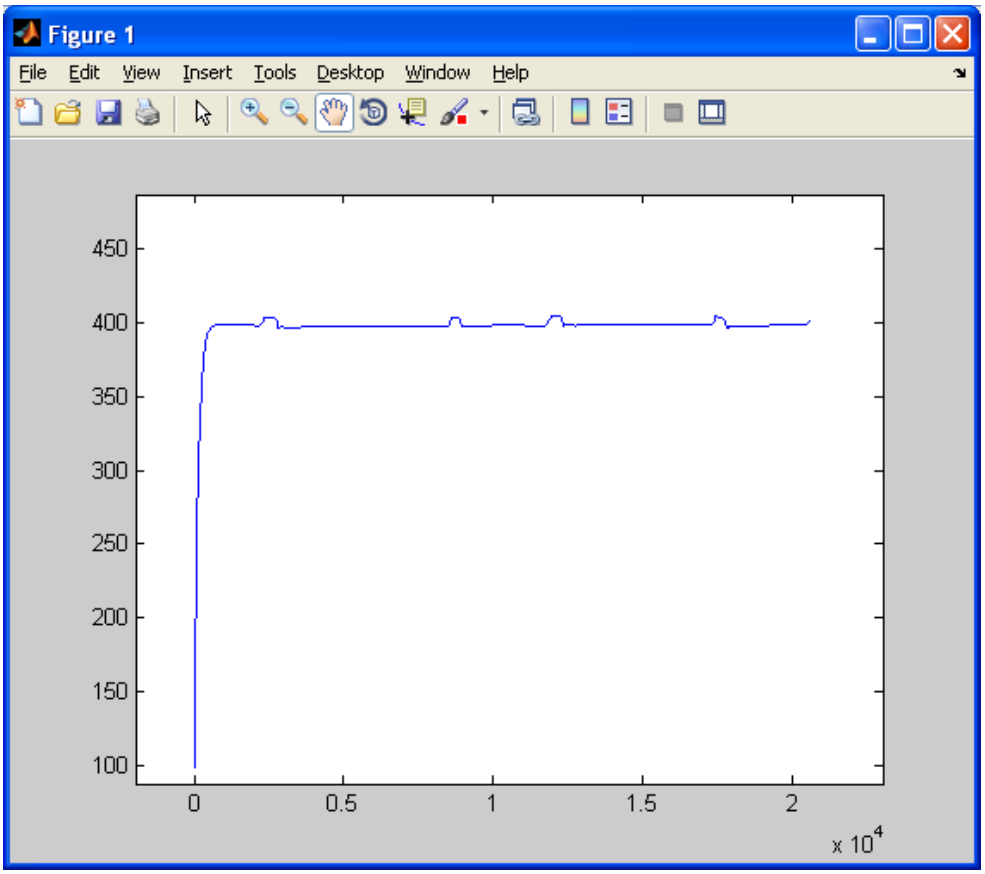

Figura 37

La figura 37 mostra chiaramente che, avendo fissato la quota a 350 piedi, il drone mantenga una quota di poco inferiore ai 400 piedi in modo abbastanza costante, con solo la presenza di qualche oscillazione in corrispondenza delle virate che effettua una volta intercettati i waypoint.

Si riportano di seguito, nelle figure 38-39-40, anche l'andamento dei principali parametri del drone durante il volo che rispettivamente sono: velocità, l'angolo di rollio e l'angolo di beccheggio.

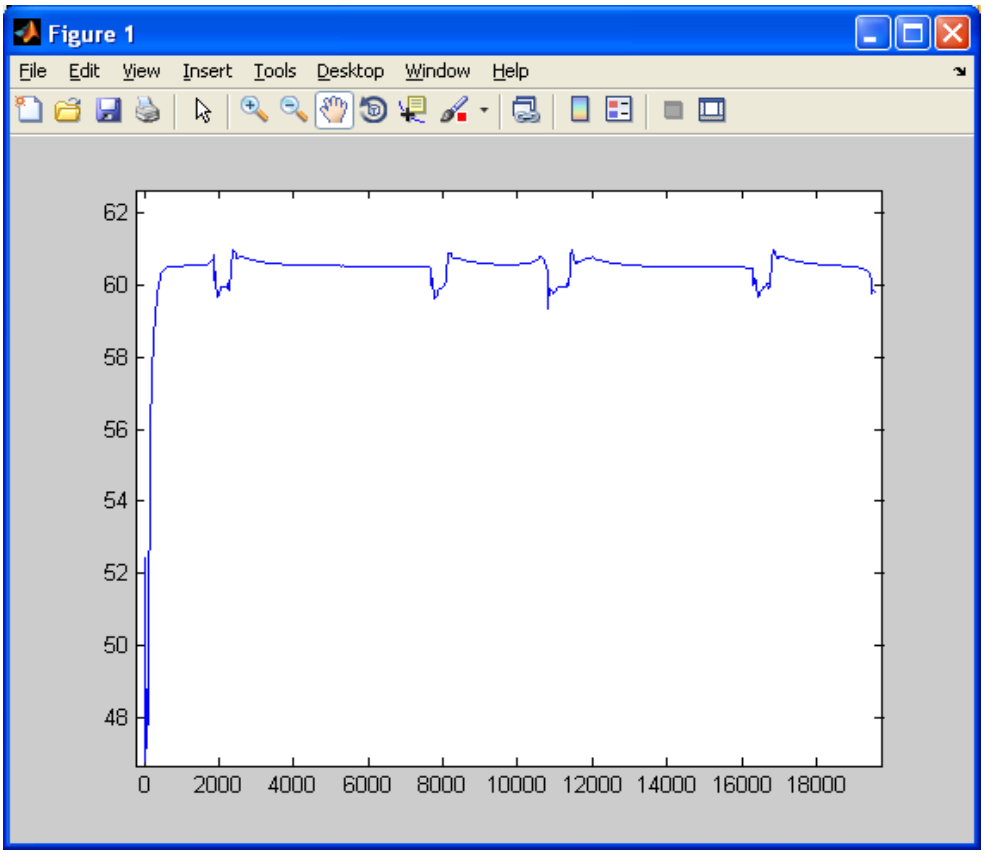

Figura 38

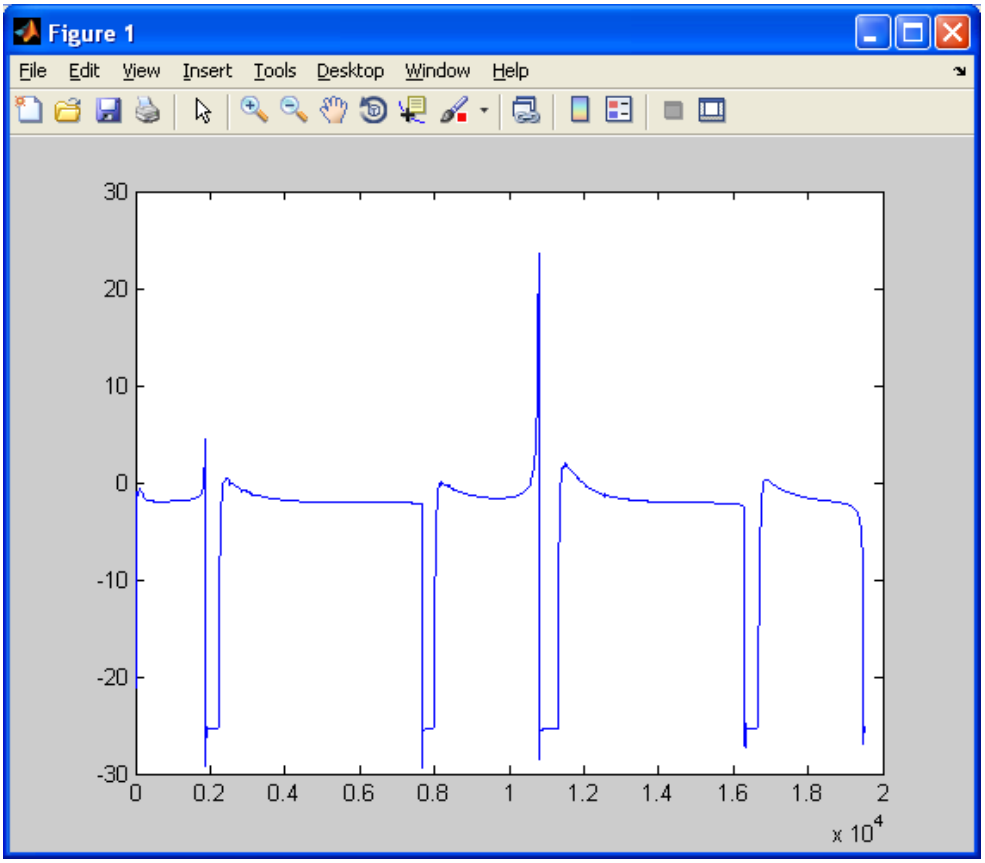

Figura 39

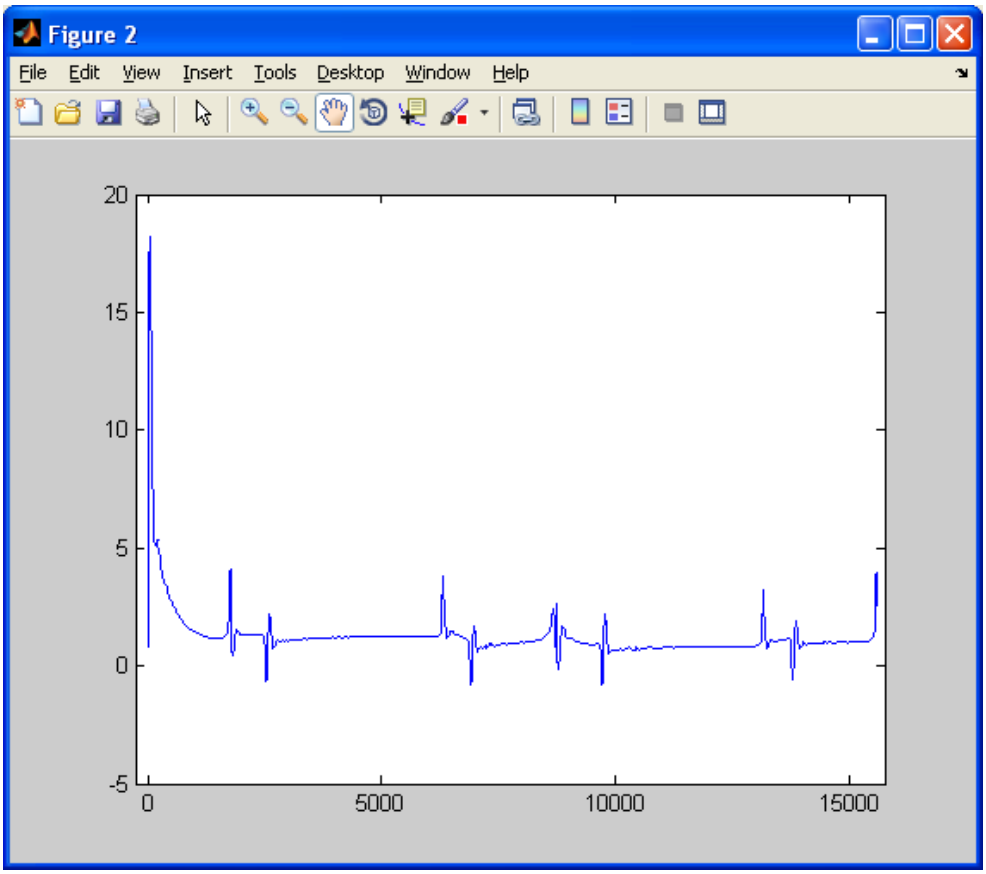

Figura 40

Osservando questa tre figure si può facilmente notare come la velocità si mantenga sempre intorno ad un valore di poco superiore ai 60 nodi, che corrispondono circa a 30 m/s, eccezion fatta in corrispondenza delle virate dove sono presenti variazioni anche repentine dovute alle variazioni di assetto del velivolo durante queste manovre.

Soffermandosi sul grafico che riporta l'angolo di rollio si può vedere meglio quello già osservato dal grafico della traiettoria e cioè che l'aereo non segue mai una traiettoria rettilinea ma è costantemente in virata poiché i valori non sono mai prossimi allo zero. Si nota anche che effettivamente non si va praticamente mai, se non nell'attimo iniziale delle virate, oltre al valore massimo di angolo di rollio fissato nell'autopilota a 25 gradi.

Nell'ultimo grafico, che riporta l'angolo di beccheggio, si vede come l'aereo mantenga, a parte un picco iniziale dovuto al fatto che questo si trovi ad una quota inferiore a quella fissata e a qualche ondeggiamento durante le virate, un angolo con valori nell'ordine di qualche grado.

Ora si può procedere con l'effettuare una seconda simulazione andando ad usare l'altro autopilota di quota, cioè quello che mantiene quella di volo. Purtroppo però, dopo pochi secondi dall'inizio della simulazione, ci si è accorti che la quota continuava ad aumentare costantemente. Si è subito capito la presenza di un errore all'interno dell'autopilota e si è fermata la simulazione anticipatamente. In figura 41 si riportano i valori assunti dalla quota nei pochi istanti di simulazione.

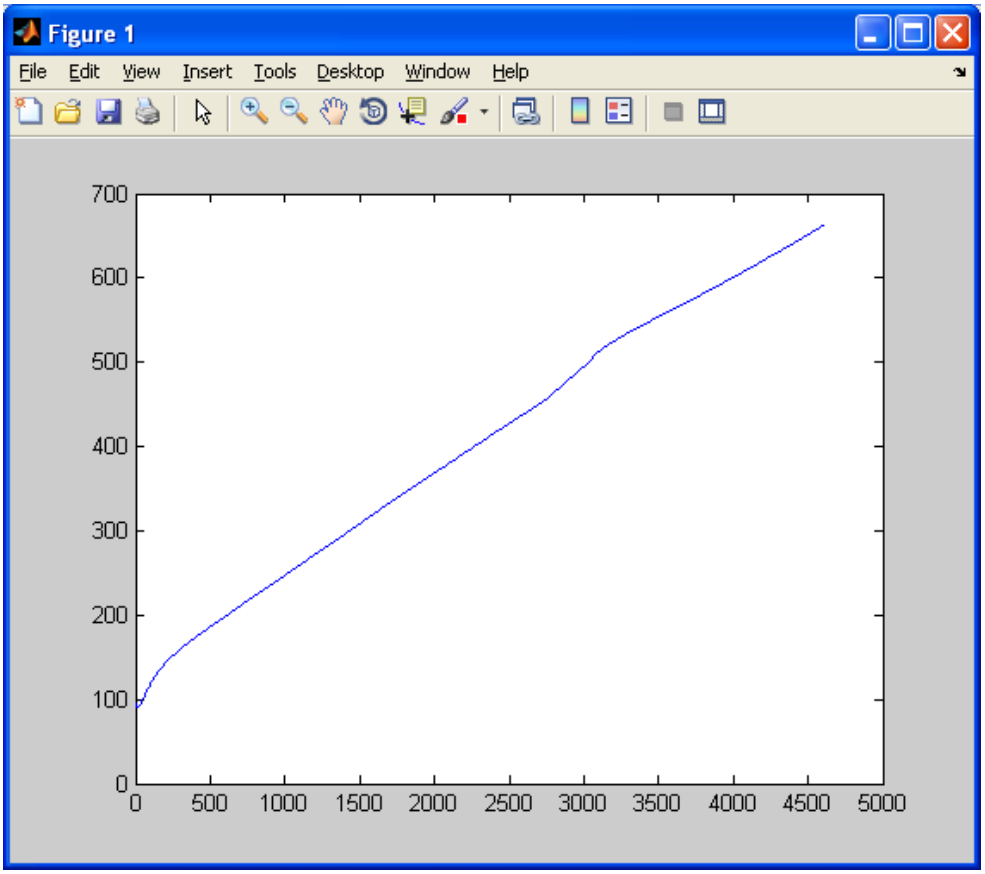

Figura 41

Capitolo 6

CONCLUSIONI

Con questa tesi si è voluto sfruttare FlightGear come un vero e proprio simulatore e non, come prassi comune, fermarsi ad usarlo solo come un semplice visualizzatore.
Per questo motivo si è andati a creare un nuovo tipo di collegamento con Simulink facendo in modo che da quest'ultimo si potesse controllare direttamente un aereo all'interno di FlightGear .
Questo ha permesso, infine, di andare a riprodurre la struttura di un drone, in dotazione all'università, all'interno di FlightGear e usare Simulink solo esclusivamente per il suo controllo implementando gli autopiloti presenti su di esso.

6.1 SVILUPPI FUTURI

Come si può facilmente intuire osservando le figure in cui si riportano i risultati delle simulazioni effettuate ci possono essere possibili sviluppi futuri per questa tesi.
Quello che balza subito all'occhio è sicuramente la possibilità di migliorare l'efficacia degli autopiloti, soprattutto quello riguardante il mantenimento, da parte del drone, della quota di volo. Infatti con uno studio appropriato sulla struttura degli autopiloti e sui valori da dare alle costanti all'interno dei regolatori PID è sicuramente possibile migliorarli così da ottenere dei risultati migliori durante le simulazioni e poterli poi trasferire sull'aereo vero e proprio.
Un'altra cosa importante da poter migliorare è il modello del drone che si è inserito all'interno di FlightGear . Le tre cose più importanti su cui si può effettivamente agire sono:

- Riuscire a conoscere l'effettiva distribuzione del peso e dei momenti di inerzia dell'aereo.
- Riuscire a conoscere il funzionamento esatto dell'ESC che controlla la potenza erogata dal motore.
- Inserire un modello aerodinamico più completo, comprendente tutti i coefficienti e non solo quelli principali di portanza e resistenza.

Così facendo si può riuscire sia ad avere una riproduzione del drone all'interno di FlightGear ancora più realistica e molto simile alla realtà.
Un ultimo importante aspetto da sottolineare è che se si confrontassero i dati di volo ottenuti nelle simulazioni effettuate con FlightGear e quelli ottenuti da voli reali del drone e si vedesse che non sono paragonabili, probabilmente è possibile pensare di poter riuscire nel processo inverso, cioè progettare un drone. Infatti si potrebbe creare un modello di un aereo in FlightGear , vedere il suo comportamento tramite Simulink, verificando così la bontà del progetto, e alla fine pensare di poterlo costruire realmente.

APPENDICE

I file usati in questa tesi sono:

- ardupilot.xml
- prova.bat
- judp.m
- fg1.m
- prova.m
- iprovau.m
- marco10mg06.dat
- npc.avl
- npcmassa.mass
- motore.m
- angolorif.m
- prova1.m
- prova2.m
- target.m

si riportano di seguito il loro contenuto.

ardupilot.xml

```
<?xml version="1.0" ?>
  <PropertyList>
  <generic>
  <output>
  <line_separator>newline</line_separator>
  <var_separator>,</var_separator>
  <!--  Position
  -->
  <chunk>
  <name>latitude-deg</name>
  <type>float</type>
```

<format>%f</format>
<node>/position/latitude-deg</node>
</chunk>
<chunk>
<name>longitude-deg</name>
<type>float</type>
<format>%f</format>
<node>/position/longitude-deg</node>
</chunk>
<chunk>
<name>altitude-ft</name>
<type>float</type>
<format>%f</format>
<node>/position/altitude-ft</node>
</chunk>
<!-- Orientation
-->
<chunk>
<name>roll-deg</name>
<type>float</type>
<format>%f</format>
<node>/orientation/roll-deg</node>
</chunk>
<chunk>
<name>pitch-deg</name>
<type>float</type>
<format>%f</format>
<node>/orientation/pitch-deg</node>
</chunk>
<chunk>
<name>heading-deg</name>
<type>float</type>
<format>%f</format>
<node>/orientation/heading-deg</node>

```
</chunk>
<!-- Velocities
-->
<chunk>
<name>airspeed-kt</name>
<type>float</type>
<format>%f</format>
<node>/velocities/airspeed-kt</node>
</chunk>
<chunk>
<name>groundspeed-kt</name>
<type>float</type>
<format>%f</format>
<node>/velocities/groundspeed-kt</node>
</chunk>
<chunk>
<name>throttle</name>
<type>float</type>
<format>%f</format>
<node>/controls/engines/engine/throttle</node>
</chunk>
</output>
<input>
<line_separator>newline</line_separator>
<var_separator>,</var_separator>
<!-- Controls
-->
<chunk>
<name>aileron</name>
<type>float</type>
<node>/controls/flight/aileron</node>
</chunk>
<chunk>
<name>elevator</name>
```

```
<type>float</type>
<node>/controls/flight/elevator</node>
</chunk>
<chunk>
<name>rudder</name>
<type>float</type>
<node>/controls/flight/rudder</node>
</chunk>
<chunk>
<name>throttle</name>
<type>float</type>
<node>/controls/engines/engine/throttle</node>
</chunk>
</input>
</generic>
</PropertyList>
```

prova.bat

```
C:
cd C:\Programmi\FlightGear
SET FG_ROOT=C:\Programmi\FlightGear\\data
.\\bin\\win32\\fgfs ^
--generic="socket,in,50,127.0.0.1,49000,udp,ArduPilot"
--generic="socket,out,50,127.0.0.1,49005,udp,ArduPilot"
--aircraft="Malolo1"
--in-air
--airport=LIPK
--runway=10L
--altitude="100"
--bpp="16"
--vc="50"
--heading="300"
--timeofday="noon"
```

--model-hz="50"

judp.m

```
SEND = 1;
RECEIVE = 2;
DEFAULT_TIMEOUT = 1000; % [milliseconds]
 % Handle input arguments.
if strcmpi(actionStr,'send')
    action = SEND;
    if nargin < 4
   error([mfilename '.m--SEND mode requires 4 input arguments.']);
    end % if
    port = varargin{1};
    host = varargin{2};
    mssg = varargin{3};
elseif strcmpi(actionStr,'receive')
    action = RECEIVE;
 if nargin < 3
        error([mfilename '.m--RECEIVE mode requires 3 input arguments.']);
    end % if
 port = varargin{1};
    packetLength = varargin{2};
 timeout = DEFAULT_TIMEOUT;
    if nargin > 3
        % Override default timeout if specified.
        timeout = varargin{3};
    end % if
else
    error([mfilename '.m--Unrecognised actionStr "' actionStr '".']);
end % if
% Test validity of input arguments.
if ~isnumeric(port) || rem(port,1)~=0 || port < 1025 || port > 65535
```

```
    error([mfilename '.m--Port number must be an integer between 1025 and
65535.']);
end % if
if action == SEND
    if ~ischar(host)
        error([mfilename '.m--Host name/IP must be a string (e.g.,
"www.examplecom" or "208.77.188.166".).']);
    end % if
    if ~isa(mssg,'int8')
        error([mfilename '.m--Mssg must be int8 format.']);
    end % if
elseif action == RECEIVE
    if ~isnumeric(packetLength)||rem(packetLength,1)~=0||packetLength < 1
        error([mfilename '.m--packetLength must be a positive integer.']);
    end % if
    if ~isnumeric(timeout) || timeout <= 0
        error([mfilename '.m--timeout must be positive.']);
    end % if
end % if
% Code borrowed from O'Reilly Learning Java, edition 2, chapter 12.
import java.io.*
import java.net.DatagramSocket
import java.net.DatagramPacket
import java.net.InetAddress
if action == SEND
    try
        addr = InetAddress.getByName(host);
        packet = DatagramPacket(mssg, length(mssg), addr, port);
        socket = DatagramSocket;
        socket.setReuseAddress(1);
        socket.send(packet);
        socket.close;
    catch sendPacketError
        try
```

```
            socket.close;
        catch closeError
            % do nothing.
        end % try
        error('%s.m--Failed to send UDP packet.\nJava error message
follows:\n%s',mfilename,sendPacketError.message);
    end % try
else
    try
        socket = DatagramSocket(port);
        socket.setSoTimeout(timeout);
        socket.setReuseAddress(1);
        packet = DatagramPacket(zeros(1,packetLength,'int8'),packetLength);
        socket.receive(packet);
        socket.close;
        mssg = packet.getData;
        mssg = mssg(1:packet.getLength);
        inetAddress = packet.getAddress;
        sourceHost = char(inetAddress.getHostAddress);
        varargout{1} = mssg;
        if nargout > 1
            varargout{2} = sourceHost;
        end % if
    catch receiveError
       % Determine whether error occurred because of a timeout.
        if
~isempty(strfind(receiveError.message,'java.net.SocketTimeoutException'))
            errorStr = sprintf('%s.m--Failed to receive UDP packet; connection
timed out.\n',mfilename);
        else
            errorStr = sprintf('%s.m--Failed to receive UDP packet.\nJava error
message follows:\n%s',mfilename,receiveError.message);
        end % if
       try
```

```
            socket.close;
        catch closeError
            % do nothing.
        end % try
       error(errorStr);
    end % try
end % if
```

fg1.m

```
function [lat long h roll pitch heading v]=fg1(port)
mssg = judp('receive',port,1000);
s = char(mssg');
[c] = textscan(s, '%s %s %s %s %s %s %s', 'delimiter', ',');
lat=str2double(c{1});
long=str2double(c{2});
h=str2double(c{3});
roll=str2double(c{4});
pitch=str2double(c{5});
heading=str2double(c{6});
v=str2double(c{7});
```

prova.m

```
function prova(block)
setup(block);

function setup(block)
block.NumInputPorts = 1;
block.NumOutputPorts = 9;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
```

```
block.OutputPort(2).Dimensions = 1;
block.OutputPort(3).Dimensions = 1;
block.OutputPort(4).Dimensions = 1;
block.OutputPort(5).Dimensions = 1;
block.OutputPort(6).Dimensions = 1;
block.OutputPort(7).Dimensions = 1;
block.OutputPort(8).Dimensions = 1;
block.OutputPort(9).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('Outputs', @Outputs);

function SetInputPortSamplingMode(block,idx,fd)
 block.InputPort(idx).SamplingMode = fd;
 block.OutputPort(1).SamplingMode = fd;
 block.OutputPort(2).SamplingMode = fd;
 block.OutputPort(3).SamplingMode = fd;
 block.OutputPort(4).SamplingMode = fd;
 block.OutputPort(5).SamplingMode = fd;
 block.OutputPort(6).SamplingMode = fd;
 block.OutputPort(7).SamplingMode = fd;
 block.OutputPort(8).SamplingMode = fd;
 block.OutputPort(9).SamplingMode = fd;

function Outputs(block)
      mssg = judp('receive',block.InputPort(1).Data,1000);
      s = char(mssg');
      [c] = textscan(s, '%s %s %s %s %s %s %s %s %s', 'delimiter', ',');
      block.OutputPort(1).Data=str2double(c{1});
      block.OutputPort(2).Data=str2double(c{2});
      block.OutputPort(3).Data=str2double(c{3});
      block.OutputPort(4).Data=str2double(c{4});
      block.OutputPort(5).Data=str2double(c{5});
```

```
block.OutputPort(6).Data=str2double(c{6});
block.OutputPort(7).Data=str2double(c{7});
block.OutputPort(8).Data=str2double(c{8});
block.OutputPort(9).Data=str2double(c{9});
pause(0.02);
```

iprovau.m

```
function iprovau(block)
setup(block);

 function setup(block)
block.NumInputPorts = 4;
block.NumOutputPorts = 7;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.InputPort(2).Dimensions = 1;
block.InputPort(3).Dimensions = 1;
block.InputPort(4).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
block.OutputPort(2).Dimensions = 1;
block.OutputPort(3).Dimensions = 1;
block.OutputPort(4).Dimensions = 1;
block.OutputPort(5).Dimensions = 1;
block.OutputPort(6).Dimensions = 1;
block.OutputPort(7).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('Outputs', @Outputs);

function SetInputPortSamplingMode(block,idx,fd)
 block.InputPort(idx).SamplingMode = fd;
```

```
 block.OutputPort(1).SamplingMode = fd;
 block.OutputPort(2).SamplingMode = fd;
 block.OutputPort(3).SamplingMode = fd;
 block.OutputPort(4).SamplingMode = fd;
 block.OutputPort(5).SamplingMode = fd;
 block.OutputPort(6).SamplingMode = fd;
 block.OutputPort(7).SamplingMode = fd;

function Outputs(block)
z=num2str(block.InputPort(1).Data);
x=num2str(block.InputPort(2).Data);
c=num2str(block.InputPort(3).Data);
v=num2str(block.InputPort(4).Data);
A=double(z);
E=double(x);
R=double(c);
T=double(v);
m=double(10);
n=double(44);
w=int8([A n E n R n T m]);
q=w';
judp('send',49000,'127.0.0.1',q);
mssg = judp('receive',49005,1000);
s = char(mssg');
[c] = textscan(s, '%s %s %s %s %s %s %s', 'delimiter', ',');
block.OutputPort(1).Data=str2double(c{1});
block.OutputPort(2).Data=str2double(c{2});
block.OutputPort(3).Data=str2double(c{3});
block.OutputPort(4).Data=str2double(c{4});
block.OutputPort(5).Data=str2double(c{5});
block.OutputPort(6).Data=str2double(c{6});
block.OutputPort(7).Data=str2double(c{7});
pause(0.02);
```

marco10mg06.dat

marco10mg06

0.999998	-0.000888
0.993308	-0.000853
0.980309	-0.000634
0.964940	-0.000169
0.948428	0.000525
0.931503	0.001407
0.914396	0.002445
0.897197	0.003612
0.879951	0.004893
0.862697	0.006276
0.845470	0.007761
0.828291	0.009354
0.811148	0.011065
0.794022	0.012901
0.776875	0.014863
0.759669	0.016943
0.742383	0.019122
0.725020	0.021378
0.707586	0.023690
0.690119	0.026038
0.672630	0.028414
0.655140	0.030814
0.637645	0.033238
0.620134	0.035684
0.602603	0.038145
0.585072	0.040609
0.567583	0.043055
0.550154	0.045463
0.532783	0.047819
0.515468	0.050106
0.498203	0.052311
0.480984	0.054422

0.463801 0.056425
0.446654 0.058313
0.429534 0.060074
0.412432 0.061702
0.395355 0.063190
0.378292 0.064531
0.361253 0.065720
0.344241 0.066745
0.327251 0.067602
0.310287 0.068283
0.293352 0.068782
0.276453 0.069089
0.259602 0.069197
0.242820 0.069094
0.226131 0.068764
0.209561 0.068186
0.193112 0.067338
0.176798 0.066202
0.160650 0.064757
0.144719 0.062982
0.129092 0.060851
0.113828 0.058320
0.098926 0.055345
0.084423 0.051902
0.070450 0.048001
0.057276 0.043683
0.045210 0.039019
0.034608 0.034168
0.025791 0.029376
0.018837 0.024885
0.013550 0.020844
0.009584 0.017284
0.006626 0.014155
0.004384 0.011390

0.002773	0.008882
0.001380	0.006682
0.000515	0.004524
0.000124	0.002377
-0.000072	0.000365
-0.000061	-0.001527
0.000255	-0.003410
0.000757	-0.005302
0.001922	-0.006905
0.003404	-0.008473
0.005087	-0.010156
0.007332	-0.011808
0.010106	-0.013504
0.013676	-0.015217
0.018278	-0.017007
0.024385	-0.018897
0.032572	-0.020908
0.043324	-0.022984
0.056460	-0.024969
0.071197	-0.026702
0.086806	-0.028110
0.102896	-0.029192
0.119373	-0.029986
0.136182	-0.030559
0.153178	-0.030957
0.170269	-0.031200
0.187437	-0.031300
0.204710	-0.031273
0.222091	-0.031147
0.239567	-0.030947
0.257097	-0.030692
0.274649	-0.030389
0.292222	-0.030041
0.309815	-0.029654

0.327432 -0.029232
0.345068 -0.028782
0.362722 -0.028308
0.380385 -0.027815
0.398056 -0.027302
0.415735 -0.026774
0.433422 -0.026230
0.451125 -0.025674
0.468843 -0.025111
0.486570 -0.024544
0.504301 -0.023974
0.522034 -0.023402
0.539772 -0.022829
0.557507 -0.022256
0.575240 -0.021684
0.592966 -0.021109
0.610680 -0.020531
0.628371 -0.019943
0.646054 -0.019343
0.663724 -0.018728
0.681392 -0.018096
0.699064 -0.017445
0.716750 -0.016780
0.734448 -0.016104
0.752152 -0.015420
0.769845 -0.014728
0.787522 -0.014025
0.805171 -0.013307
0.822792 -0.012565
0.840400 -0.011793
0.858022 -0.010990
0.875667 -0.010162
0.893334 -0.009316
0.911018 -0.008457

0.928699 -0.007591

0.946306 -0.006726

0.963597 -0.005881

0.979742 -0.005103

0.993250 -0.004450

1.000003 -0.004114

npc.avl

biroccio12

0.0 | Mach

0 0 0.0 | iYsym iZsym Zsym

292799.987 333.802 960.000 | Sref Cref Bref

68.264 -0.000 5.522 | Xref Yref Zref

0.0084 | CDp (optional)

#==

SURFACE | (keyword)

Plane Name_Wing

#Nchord Cspace [Nspan Sspace]

11 1.0

INDEX | (keyword)

1842 | Lsurf

YDUPLICATE

0.0

SCALE

1.0 1.0 1.0

TRANSLATE

0.0 0.0 0.0

ANGLE

0.000 | dAinc

#_____

SECTION | (keyword)

0.0000 40.0000 0.0000 320.0000 0.000 3 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

25.3750 87.5000 0.0000 324.3750 0.000 2 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

50.7500 135.0000 0.0000 328.7500 0.000 2 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

76.1250 182.5000 0.0000 333.1250 0.000 2 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

101.5000 230.0000 0.0000 337.5000 0.000 2 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

126.8750 277.5000 0.0000 341.8750 0.000 2 0 | Xle Yle Zle Chord Ainc [Nspan Sspace]

AFIL 0.0 1.0

marco10mg06.dat

#_____

SECTION | (keyword)

```
 152.2500  325.0000    0.0000  346.2500   0.000    2    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
marco10mg06.dat
#_____________
SECTION                                    |  (keyword)
 177.6250  372.5000    0.0000  350.6250   0.000    2    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
marco10mg06.dat
#_____________
SECTION                                    |  (keyword)
 203.0000  420.0000    0.0000  355.0000   0.000    2    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
marco10mg06.dat
#_____________
SECTION                                    |  (keyword)
 235.0000  480.0000    0.0000  250.0000   0.000    2    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
marco10mg06.dat
#-----------------------------------------------
SURFACE
Fuselage
#Nchordwise  Cspace   Nspanwise  Sspace
32          0.6
INDEX
1
YDUPLICATE
0.0
SCALE
1.0   1.0  1.0
SECTION
```

```
#Xle       Yle    Zle     Chord   Ainc  Nspanwise  Sspace
 -130.0    0.0    0.0     580.0    0.    1          0.
SECTION
#Xle    Yle    Zle     Chord      Ainc  Nspanwise  Sspace
 0.0       40.0   0.0     450.0       0.    1          0.
#-------------------------------------------------
SURFACE
Fuselage V Top
#Nchordwise  Cspace   Nspanwise  Sspace
24           1.0
SECTION
#Xle      Yle    Zle      Chord   Ainc  Nspanwise  Sspace
 -130.0   0.0    0.0      580.0    0.    1          0.
SECTION
#Xle     Yle    Zle      Chord   Ainc  Nspanwise  Sspace
 -110.0   0.0    6.0      400.0   0.    1          0.
SECTION
#Xle      Yle    Zle      Chord   Ainc  Nspanwise  Sspace
 -90.0    0.0    12.0     300.0   0.    1          0.
SECTION
#Xle   Yle    Zle       Chord   Ainc  Nspanwise  Sspace
 -70.0   0.0    18.0       200.0    0.    1          0.
SECTION
#Xle    Yle    Zle      Chord   Ainc  Nspanwise  Sspace
 -50.0   0.0    24.0      100.0    0.    1          0.
SECTION
#Xle   Yle    Zle       Chord   Ainc  Nspanwise  Sspace
-30.0   0.0    30.0       10.0    0.    1          0.
#================================================
SURFACE                  | (keyword)
Plane Name_Fin
#Nchord    Cspace   [ Nspan Sspace ]
3          1.0
INDEX                    | (keyword)
```

```
1845                    | Lsurf
YDUPLICATE
  1.0000
SCALE
1.0  1.0  1.0
TRANSLATE
0.0  0.0  0.0
ANGLE
  0.000                 | dAinc
#_______________
SECTION                                 |  (keyword)
 485.0000  480.0000  -60.0000  100.0000   0.000    2    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
NACA0006.dat
#_______________
SECTION                                 |  (keyword)
 235.0000  480.0000   0.0000  345.0000   0.000    1    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
NACA0006.dat
#_______________
SECTION                                 |  (keyword)
 485.0000  480.0000   80.0000  100.0000   0.000    1    0   | Xle Yle Zle
Chord Ainc   [ Nspan Sspace ]
AFIL 0.0 1.0
NACA0006.dat
```

npcmassa.mass

```
#--------------------------------------------
#
#   Plane Name
#
```

```
#  Dimensional unit and parameter data.
#  Mass & Inertia breakdown.
#-------------------------------------------------
#  Names and scalings for units to be used for trim and eigenmode
calculations.
#  The Lunit and Munit values scale the mass, xyz, and inertia table data
below.
#  Lunit value will also scale all lengths and areas in the AVL input file.
#
Lunit =     0.0010 m
Munit =     1.0000 kg
Tunit = 1.0 s
#-------------------------
#  Gravity and density to be used as default values in trim setup (saves
runtime typing).
#  Must be in the unit names given above (i.e. m,kg,s).
g   = 9.81
rho = 1.225
#-------------------------
#  Mass & Inertia breakdown.
#  x y z  is location of item's own CG.
#  Ixx... are item's inertias about item's own CG.
#
#  x,y,z system here must be exactly the same one used in the .avl input file
#    (same orientation, same origin location, same length units)
#
#    mass         x         y         z       Ixx       Iyy       Izz       Ixy       Ixz
Iyz
    0.196       245  -2.18e-14       4.45   1.63e+04   1.96e+04   1.82e+04
0       17.7         0 ! Plane Name_Wing
    0.01       429   3.98e-14       6.67   2.31e+03       61.9   2.36e+03         0
-1.81         0 ! Plane Name_Fin
    0.39       175         0       13.6       3.82   9.21e+03    9.2e+03         0
73.4         0 ! Body's inertia
```

```
    0.444      -70       0       10     0.000     0.000     0.000 ! battery
     0.16      -40       0       15     0.000     0.000     0.000 ! peso foto
     0.06     -130       0     -120     0.000     0.000     0.000 ! peso
aggiuntivo
    0.169      -10       0       10     0.000     0.000     0.000 ! strumenti
    0.045      440       0       10     0.000     0.000     0.000 ! motore
    0.022      400       0       10     0.000     0.000     0.000 !
```

motore.m

```
rho=1.225;
p=4;
d=8;
Kv=1700;
Vin=14.8;
Iin=23;
Io=2;
Rm=0.075;
 n=Kv*(Vin-(Iin*Rm));
 n1=n/1000;
d1=d*0.0833;
p1=p*0.0833;
Kp=1.25;
Pout=Kp*(d1^4)*p1*(n1^3);
Pout1=(Vin-(Iin*Rm))*(Iin-Io);
 if Pout1 > Pout+10
    disp('elica errata');
end
 while Pout1 <= Pout
    n=n-10;
    n1=n/1000;
    d1=d*0.0833;
    p1=p*0.0833;
    Pout=Kp*(d1^4)*p1*(n1^3);
```

```
end
g=(n:-500:0)';
g1=g./1000;
Pout2=Kp.*(d1.^4).*p1.*(g1.^3);
Iin2=(((Pout2.*Kv))./g)+Io;
Vin2=(g./Kv)+((((Pout2.*Kv)./g)+Io)*Rm);
Pin2=Vin2.*Iin2;
g2=g./60;
dm=d*0.0254;
Cp=Pin2./(rho.*(g2.^3).*(dm^5));
T=(((Pin2.^2).*rho.*(((dm/2)^2)*pi)).^(1/3));
Ct=T./((rho.*(g2.^2)).*(dm.^4));
v=Pin2./T;
J=v./(g.*dm);
```

angolorif.m

```
function angolorif(block)
setup(block);

function setup(block)
block.NumInputPorts = 2;
block.NumOutputPorts = 1;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.InputPort(2).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('Outputs', @Outputs);

function SetInputPortSamplingMode(block,idx,fd)
```

```
 block.InputPort(idx).SamplingMode = fd;
 block.OutputPort(1).SamplingMode = fd;

function Outputs(block)
b = block.InputPort(1).Data - block.InputPort(2).Data;
B = abs(b);
c = 36000 - B;
if c > B
   u = b;
end
if (c < B) && (block.InputPort(1).Data > 18000)
   u = -c;
end
if (c < B) && (block.InputPort(1).Data < 18000)
   u = c;
end
if c == B
   u = b;
end
block.OutputPort(1).Data = u;
```

prova1.m

```
function prova1(block)
setup(block);

function setup(block)
block.NumInputPorts = 4;
block.NumOutputPorts = 1;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.InputPort(2).Dimensions = 1;
block.InputPort(3).Dimensions = 1;
```

```
block.InputPort(4).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('Outputs', @Outputs);

function SetInputPortSamplingMode(block,idx,fd)
 block.InputPort(idx).SamplingMode = fd;
 block.OutputPort(1).SamplingMode = fd;

function Outputs(block)
z=num2str(block.InputPort(1).Data);
x=num2str(block.InputPort(2).Data);
c=num2str(block.InputPort(3).Data);
v=num2str(block.InputPort(4).Data);
A=double(z);
E=double(x);
R=double(c);
T=double(v);
m=double(10);
n=double(44);
w=int8([A n E n R n T m]);
q=w';
judp('send',49000,'127.0.0.1',q);
block.OutputPort(1).Data=10;
```

prova2.m

```
function prova2(block)
setup(block);

function setup(block)
block.NumInputPorts = 1;
```

```
block.NumOutputPorts = 9;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
block.OutputPort(2).Dimensions = 1;
block.OutputPort(3).Dimensions = 1;
block.OutputPort(4).Dimensions = 1;
block.OutputPort(5).Dimensions = 1;
block.OutputPort(6).Dimensions = 1;
block.OutputPort(7).Dimensions = 1;
block.OutputPort(8).Dimensions = 1;
block.OutputPort(9).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('Outputs', @Outputs);

function SetInputPortSamplingMode(block,idx,fd)
 block.InputPort(idx).SamplingMode = fd;
 block.OutputPort(1).SamplingMode = fd;
 block.OutputPort(2).SamplingMode = fd;
 block.OutputPort(3).SamplingMode = fd;
 block.OutputPort(4).SamplingMode = fd;
 block.OutputPort(5).SamplingMode = fd;
 block.OutputPort(6).SamplingMode = fd;
 block.OutputPort(7).SamplingMode = fd;
 block.OutputPort(8).SamplingMode = fd;
 block.OutputPort(9).SamplingMode = fd;

function Outputs(block)
      mssg = judp('receive',block.InputPort(1).Data,10000);
      s = char(mssg');
      [c] = textscan(s, '%s %s %s %s %s %s %s %s %s', 'delimiter', ',');
```

```
block.OutputPort(1).Data=str2double(c{1});
block.OutputPort(2).Data=str2double(c{2});
block.OutputPort(3).Data=str2double(c{3});
block.OutputPort(4).Data=str2double(c{4});
block.OutputPort(5).Data=str2double(c{5});
block.OutputPort(6).Data=str2double(c{6});
block.OutputPort(7).Data=str2double(c{7});
block.OutputPort(8).Data=str2double(c{8});
block.OutputPort(9).Data=str2double(c{9});
pause(0.02);
```

target.m

```
function target(block)
setup(block);

function setup(block)
block.NumInputPorts = 9;
block.NumOutputPorts = 2;
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
block.InputPort(1).Dimensions = 1;
block.InputPort(2).Dimensions = 1;
block.InputPort(3).Dimensions = 1;
block.InputPort(4).Dimensions = 1;
block.InputPort(5).Dimensions = 1;
block.InputPort(6).Dimensions = 1;
block.InputPort(7).Dimensions = 1;
block.InputPort(8).Dimensions = 1;
block.InputPort(9).Dimensions = 1;
block.OutputPort(1).Dimensions = 1;
block.OutputPort(2).Dimensions = 1;
block.SampleTimes = [-1 0];
block.RegBlockMethod('PostPropagationSetup',@DoPostPropSetup);
```

```
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSampli
ngMode);
block.RegBlockMethod('InitializeConditions',@InitConditions);
block.RegBlockMethod('Outputs', @Outputs);

function DoPostPropSetup(block)
block.NumDworks = 3;
block.Dwork(1).Name = 'n';
block.Dwork(1).Dimensions = 1;
block.Dwork(1).DatatypeID      = 0;
block.Dwork(1).Complexity      = 'Real';
block.Dwork(1).UsedAsDiscState = true;
block.Dwork(2).Name = 'm';
block.Dwork(2).Dimensions = 1;
block.Dwork(2).DatatypeID      = 0;
block.Dwork(2).Complexity      = 'Real';
block.Dwork(2).UsedAsDiscState = true;
block.Dwork(3).Name = 'f';
block.Dwork(3).Dimensions = 1;
block.Dwork(3).DatatypeID      = 0;
block.Dwork(3).Complexity      = 'Real';
block.Dwork(3).UsedAsDiscState = true;

function InitConditions(block)
block.Dwork(1).Data = 2;
block.Dwork(2).Data = 3;
block.Dwork(3).Data = 1;
block.OutputPort(1).Data = 44.198477;
block.OutputPort(2).Data = 12.040955;

function SetInputPortSamplingMode(block,idx,fd)
 block.InputPort(idx).SamplingMode = fd;
 block.OutputPort(1).SamplingMode = fd;
 block.OutputPort(2).SamplingMode = fd;
```

```
function Outputs(block)
if (block.Dwork(3).Data == 0) && (block.InputPort(1).Data < 30)
   block.Dwork(1).Data = block.Dwork(1).Data +2;
   block.Dwork(2).Data = block.Dwork(2).Data +2;
   if block.Dwork(1).Data > 8
      block.Dwork(1).Data = 2;
      block.Dwork(2).Data = 3;
   end
   block.Dwork(3).Data = 1;
end
if (block.InputPort(1).Data > 100)
   block.Dwork(3).Data = 0;
end
block.OutputPort(1).Data = block.InputPort(block.Dwork(1).Data).Data;
block.OutputPort(2).Data = block.InputPort(block.Dwork(2).Data).Data;
```

Bibliografia

[1] http://www.flightgear.org

[2] http://www.mathworks.com/matlabcentral/fileexchange/24525-a-simple-udp-communications-application

[3] http://wiki.flightgear.org

[4] http://www.helifly.org/Index_Lezioni.asp

[5] http://jsbsim.sourceforge.net

[6] http://web.mit.edu/drela/Public/web/avl

[7] http://www.xflr5.com/xflr5.htm

[8] G. Marro, *Controlli Automatici* - Quinta Edizione, Zanichelli, Bologna, 2006