POLITECNICO DI MILANO

Facoltá di Ingegneria Industriale
Corso di laurea in
Ingegneria Aeronautica

PROGETTAZIONE DI UN SIMULATORE DI VOLO PER LO STUDIO DELLA PILOT AUGMENTED OSCILLATION

Relatore: Prof. Pierangelo Masarati
Corelatore: Ing. Giuseppe Quaranta

Tesi di laurea di:
Lorenzo TABORELLI Matr. 720684

Anno accademico 2009-2010

Vivi e sinceri ringraziamenti
a coloro che hanno fornito il loro ausilio
durante la stesura di questa tesi di laurea magistrale:
all' Ing. Quaranta e al Prof. Masarati.
Vorrei ringraziare tutti i miei compagni di squadra,
di universitá e di mille disavventure, che non si
sarebbero mai aspettati mi laureassi e tutti coloro che,
invece, hanno sempre sostenuto il contrario.
Per concludere non vorrei dimenticare
i miei genitori che hanno saputo darmi
sempre un sostegno morale, per non dire intellettuale,
anche nei momenti peggiori.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

L' obbiettivo di questa tesi di laurea magistrale é duplice. Da un lato si vuole progettare e realizzare il mockup di un simulatore di volo elicotteristico per prove di vibrazione, rispondendo a esigenze di comfort, ergonomicitá e di crashworthiness, senza dimenticare le richieste vincolanti date dalle MIL-STD considerate.
Dall' altro si vuole generare e verificare un modello numerico ridotto del BO105 che comprenda, sia la struttura dell' elicottero, che il modello di pilota definito da Mayo J.R., cosí da poter fare simulazioni *real-time* interfacciandosi con un simulatore grafico come *FlightGear* e in modo da analizzare fenomeni di *Pilot Augmented Oscillation*.
Infine si é studiato un software di animazione 3D (Blender) in grado di creare animazioni di dettaglio ad alta definizione sfruttando la possibilitá di interfaccia tra il solutore multi-corpo (MBDyn) e il software di rendering grafico.

Parole chiave: socket, real-time, mockup, simulatore di volo, Pilot Augmented Oscillation, Mayo, BO105

Abstract

This master thesis has two objectives. One hand it' s necessary to design and build the mockup of a helicopter flight simulator for vibration tests, satisfying comfort requirements, ergonomics needs and crashworthiness, but without forgetting the binding requests defined by the considered MIL-STD. The other is to generate and verify a reduced numerical model of the BO105 that includes, both the structure of helicopter and the pilot model defined by Mayo J.R., in order to make *real-time* simulations interfacing with graphics simulators such as *FlightGear* and in order to analyze phenomena such as*Pilot Augmented Oscillation*.
Finally, a 3D animation software has been studied (Blender) in order to create high-definition animation exploiting the opportunity to interface the multi-body solver (MBDyn) and the graphics rendering software.

Keywords: socket, real-time, mockup, flight simulator, Pilot Augmented Oscillation, Mayo, BO105

Capitolo 1

Introduzione al lavoro

Nel corso dell'ultimo decennio si é registrato un aumento nel settore della ricerca in materia di cause e rimedi per le vibrazioni o fenomeni di divergenza causati da interazioni negative tra il pilota e il veicolo. L'interazione del pilota con il velivolo, in determinate circostanze, puó comportare un degrado delle caratteristiche della macchina in termini di prestazioni e caratteristiche di manovra; nel peggiore dei casi, il pilota puó anche destabilizzare il sistema. Un aereo e il suo pilota, possono essere visti come due sistemi dinamici collegati in retroazione: il moto del velivolo stimola il pilota, che reagisce iniettando comandi nei controlli di volo attraverso gli strumenti posti nel cockpit. E' noto dalla teoria del controllo che questa interconnessione puó portare ad un sistema instabile, nonostante i due sottosistemi siano perfettamente stabili quando considerati separatamente. In generale gli Aircraft/Rotorcraft-Pilot Couplings sono definiti come, "*oscillazioni del velivolo sostenute e involontarie che sono conseguenza di un' interazione anormale tra il velivolo e il pilota*".

Sono state definite due classi principali di accoppiamenti con il pilota: quelle relative ad un intervento attivo, seppur involontario, del pilota, chiamate *Pilot Induced Oscillation* (PIO), e quelle relative al comportamento d' impedenza passiva del pilota, chiamate *Pilot Augmented Oscillation* (PAO). I due fenomeni sono distinti e differiscono anche per i range di frequenza d' interesse: il comportamento umano é volontario fino a circa 1Hz, mentre é involontario, quindi passivo, al di lá di questo limite. Un limite superiore, convenzionalmente accettato, per la gamma di frequenze che possono essere d' interesse per la dinamica "passiva" é 8Hz. Chiaramente, in questo intervallo di frequenze, il pilota interagisce con le dinamiche strutturali del velivolo, e puó modificare le caratteristiche aeroelastiche dello stesso. Per poter cogliere questo fenomeno aeroelastico si é reso necessario lo sviluppo di modelli dinamici completamente flessi-

bili per il velivolo e di un modello adeguato per la risposta biomeccanica del pilota. Questo problema é stato profondamente indagato per quanto riguarda aeromobili ad ala fissa, come testimoniato da una notevole quantitá di letteratura. Tuttavia, le sue implicazioni sulle dinamiche ed aeroelasticitá di velivoli ad ala rotante non sono state ben comprese. La urgenza di una comprensione ingegneristica e di soluzioni efficaci per il *Pilot Augmented Oscillation* (PAO) deriva da incidenti catastrofici in passato. La comprensione, previsione e prevenzione del fenomeno di PAO é un compito impegnativo che richiede l' analisi e la simulazione di tutto il circuito in anello chiuso.

Sulla base di numerose sperimentazioni in volo nel passato, diversi tipi di A/RPC sono stati osservati in base al loro contenuto in frequenza, nonché in base alla fisica del comportamento umano. A seguito di una classificazione basata principalmente su diversi range caratteristici di frequenze, sono state osservate due classi di A/RPC. La prima classe di frequenze piú basse in prossimitá e al di sotto di 1Hz é caratterizzata da fenomeni di accoppiamento dominato dalle dinamiche del velivolo a bassa frequenza (cioé il settore della meccanica del volo), dal sistema di controllo di volo e da un pilota "attivo" concentrato su come eseguire il suo compito di missione attivamente manipolando i comandi dell' elicottero. La seconda da 2 a 8Hz é invece caratterizzata dalla dinamica a piú alta frequenza del velivolo, come ad esempio i modi elastici della fusoliera e delle pale del rotore principale, da un pilota "passivo" soggetto a vibrazioni con contenuto in frequenza troppo elevato per essere adeguatamente contrastate dalla volontá umana e dalla risposta inerziale dei controlli che interessano i piloti sulle superfici di controllo dell' elicottero.

Questa distinzione é relativamente semplice per gli aerei ad ala fissa, per i velivoli ad ala rotante non si ha una distinzione cosí netta, dal momento che una larga fascia di frequenze dinamiche di volo si va a sovrapporre significativamente alla biodinamica intrinseca degli arti del pilota.

In questo lavoro di tesi vengono presi i modelli di elicottero e di pilota sviluppati finora e vengono rivisti in un' ottica di miglioramento delle prestazioni computazionali, in modo da rendere possibile la creazione di un software di simulazione del volo *real-time* per lo studio dei fenomeni di accoppiamento tra la dinamica dell' elicottero e la biodinamica del pilota. Per primo si verifica la correttezza del nuovo modello ridotto, confrontandolo dal punto di vista inerziale e dinamico con quello originale, dopodiché si verifica il guadagno computazionale ottenuto e infine si accoppiano il modello ottenuto e il solutore multi-corpo con il programma di simulazione grafica scelto, in modo da effettuare le prime prove virtuali di vibrazione.

Capitolo 2

Studio ergonomicitá e comfort

Per lo studio dell' ergonomicitá e del comfort si sono tenuti in considerazione aspetti riguardanti le normative e aspetti di tipo pratico-realizzativo. L' obbiettivo da noi prefissato é quello di poter dare sufficiente libertá di movimento e di abitabilitá a una gamma di individui compresa tra il 5 percentile e il 95 percentile. Questo obbiettivo puó essere raggiunto solo in parte a causa del fatto che si vuole mantenere una elevata semplicitá realizzativa del mockup.

2.1 Antropometria

Prima di procedere alla progettazione degli spazi é stato necessario uno studio delle dimensioni del corpo umano e delle loro escursioni tra individuo e individuo. Dal canto nostro si é scelto di prendere in considerazione il 90% della popolazione umana, limitando cosí le escursioni massime e minime da tenere in considerazione. Per quanto riguarda le dimensioni si é fatto riferimento alla normativa MIL-STD-1472D ([1]), nella quale sono tabulate tutte le dimensioni di nostro interesse. Nella tabella 2.1 di seguito vengono riportate le piú importanti (Ref [2], [3]).

Come si puó notare dalla figura 2.1 le dimensioni variano in modo consistente e ció porta inevitabilmente a una complicazione della gestione degli spazi. Si é consapevoli del fatto che non si potranno soddisfare al meglio le esigenze di tutti gli individui presi in considerazione, ma di sicuro si vuole raggiungere il miglior compromesso.

Tabella 2.1: manichini

	5 *percentile*	95 *percentile*
Peso	$60.4kg$	$96kg$
Statura	$1642mm$	$1877mm$
Altezza torace	$1208mm$	$1385mm$
Altezza vita	$976mm$	$1151mm$
Altezza inguine	$747mm$	$920mm$
Larghezza petto	$295mm$	$385mm$
Larghezza fianchi	$317mm$	$388mm$

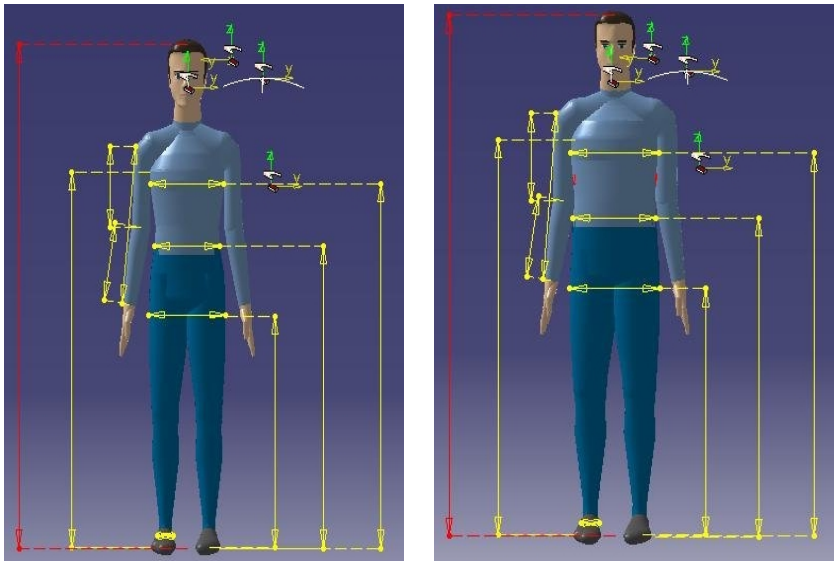
2.2 Progettazione del cockpit

Aspetti generali Per quanto riguarda la progettazione del cockpit si sono dovute seguire delle precise normative atte a massimizzare, al tempo stesso, l' ergonomicitá della seduta e la sicurezza in volo. Un aspetto fondamentale é stato quello di valutare la capacitá di afferrare e muovere i controlli per il tipo di popolazione presa in considerazione.

Le normative MIL-STD 203 ([4]) e MIL-STD 250 ([5]) definiscono tre differenti zone:

ZONA 1 - limitazione con imbracatura bloccata - *raggiungimento funzionale*: questa zona comprende l' area che puó essere funzionalmente raggiunta e azionata da un membro dell' equipaggio, della popolazione definita, quando si trova nella posizione appropriata completamente imbracato ed equipaggiato, senza allungare i muscoli del braccio o della spalla. I controlli posti in questa zona includono quelli piú frequentemente utilizzati durante il funzionamento degli aeromobili nelle fasi di volo che richiedono il massimo controllo. Ció include le fasi di volo come, il decollo, atterraggio, volo a bassa quota e alta velocitá, la consegna di armi e manovre di uscita. Questa zona definisce il limite massimo consentito per il collocamento dei controlli di emergenza e stabilisce il limite operativo a prua dei controlli di volo primari e di propulsione.

ZONA 2 - limitazione con imbracatura bloccata - *massimo raggiungimento funzionale*: questa zona comprende l' area che puó essere funzionalmente raggiunta e azionata da un membro dell' equipaggio, della popolazione definita, quando si trova nella posizione appropriata completamente imbracato e con i muscoli del braccio e della spalla estesi al massimo. Questa zona definisce il limite massimo consenti-



(a) *manichino 5 percentile.* **(b)** *manichino 95 percentile.*

Figura 2.1: manichini

to per il collocamento dei controlli di volo primario elicotteristici e di propulsione e il posizionamento dei controlli di emergenza diversi da quelli per la salvezza.

ZONA 3 - limitazione con imbracatura sbloccata: questa zona comprende l' area che puó essere funzionalmente raggiunta e azionata da un membro dell' equipaggio, della popolazione definita, quando si trova nella posizione appropriata con il sistema di ritenuta della spalla completamente esteso e le braccia estese al massimo.

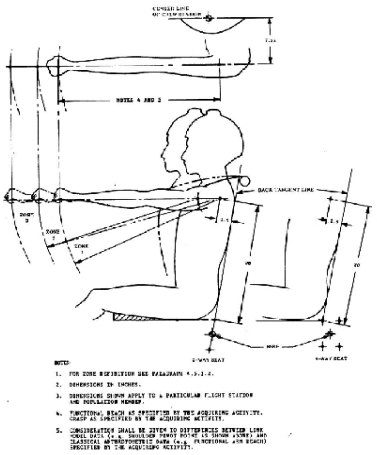


Figura 2.2: zone limite raggiungibili

Geometria della seduta Per definire la geometria della seduta dell' equipaggio bisogna prendere in considerazione tutti gli aspetti di controllo, le esigenze di visualizzazione associate al volare in sicurezza, l' esecuzione della missione e bisogna considerare che siano tutte conformi ai requisiti specificati dalle normative. Nella figura 2.3 si riporta un esempio di seduta, la quale deve rispettare i requisiti per il tipo di popolazione specificata e per l' inclinazione del sedile.

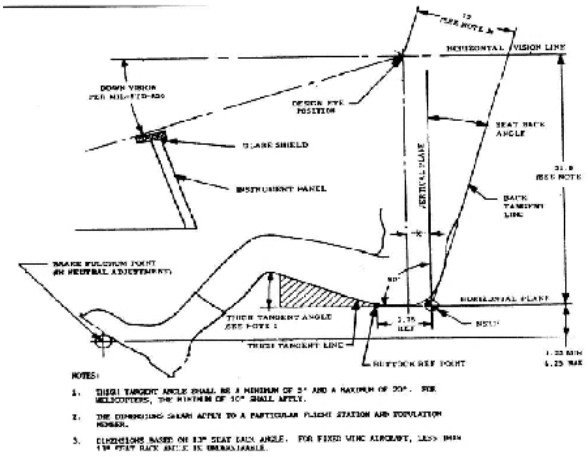


Figura 2.3: geometria seduta

Per quanto riguarda i requisiti di visione applicati agli elicotteri pilotati sia singolarmente che in tandem e sia in posizione arretrata che avanzata

si fa riferimento alla normativa MIL-STD 850b ([6]) la quale stabilisce gli angoli minimi di visibilitá perfetta disponibili al pilota dalla "design eye position":

- A 0" azimuth, almeno 25° verso il basso e 70° verso l' alto
- A 20" azimuth, sia a destra che a sinistra, 25° verso il basso e 70° verso l' alto
- A 30" azimuth, sia a destra che a sinistra, 30° verso il basso e 70° verso l' alto
- A 90" azimuth, sia a destra che a sinistra, 50° verso il basso e 70° verso l' alto
- A 135" azimuth, sia a destra che a sinistra, 34° verso il basso e 70° verso l' alto

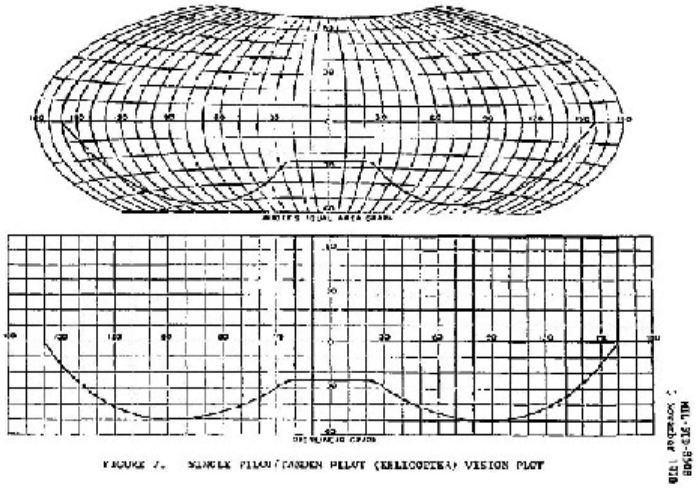


Figura 2.4: inviluppo compo visivo

Geometria passo ciclico La posizione verticale del punto di riferimento dell' impugnatura del controllo deve poter essere raggiunto all' interno della zona 1 per la popolazione considerata. L' intero inviluppo dello stick deve essere nella zona 1. Deve essere mantenuta una distanza minima di 1,5 pollici tra lo stick e tutte le strutture, soprattutto quando é in una

posizione di massima escursione. Particolare attenzione si deve dare all' equipaggiamento personale e di sopravvivenza al momento di stabilire l' inviluppo del comando.

Geometria passo collettivo Per quanto riguarda il passo collettivo esso viene collocato come in figura 2.5.

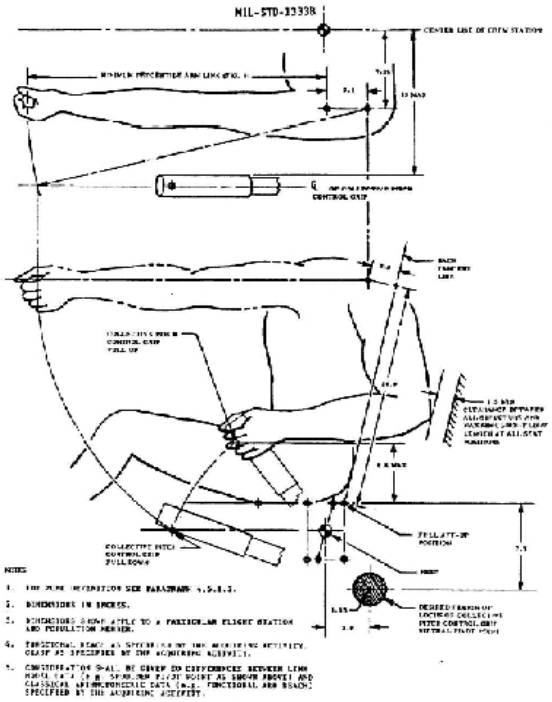


Figura 2.5: geometria passo collettivo

Geometria pedaliera Il controllo d' imbardata si compone di due pedali dalla configurazione conforme alla MIL-STD 8584 ([7]). Anche una frenata differenziale definita come da normativa puó essere fornita da questi pedali. La posizione piú avanzata di regolazione dei comandi d' imbardata si basa sulla lunghezza della gamba da seduto con il sedile completamente indietro e in basso, e i controlli d' imbardata completamente avanti, con il freno premuto a fondo, come mostrato in figura 2.6.

La posizione piú arretrata di regolazione dei controlli d' imbardata si basa sulla minima lunghezza di gamba da seduto con il sedile completamente in avanti e in alto e con i controlli d' imbardata completamente

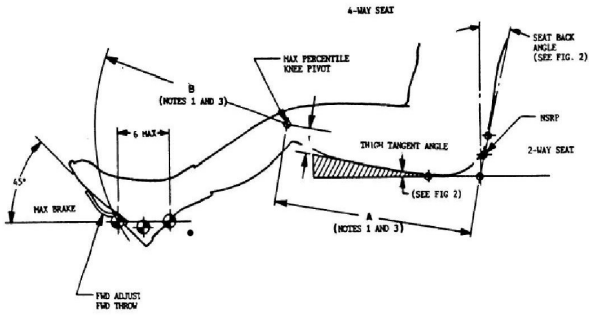


Figura 2.6: geometria pedaliera-posizione avanzata

avanzati, con il freno premuto a fondo come mostrato in figura 2.7. Una distanza minima di 1,5 pollici sopra e 0,75 pollici su entrambi i lati del pedale deve essere mantenuta anche per la massima dimensione del piede compreso lo stivale, per tutta la corsa del pedale compresa l' applicazione dei freni, con il calcagno in corrispondenza del fulcro del pedale.

Con le normali procedure di frenatura deve essere mantenuto uno spazio libero di almeno 1,5 pollici tra le calzature, di dimensioni massime, e tutti gli strumenti e le strutture adiacenti per tutte le geometrie di seduta e per ogni membro della popolazione considerata. Tutta la gamma di regolazioni del controllo d' imbardata, la distanza tra il fulcro del freno e il punto piú vicino al piano devono essere almeno di 4,75 pollici. La lunghezza del pedale deve essere la minima necessaria per soddisfare i requisiti di frenata.

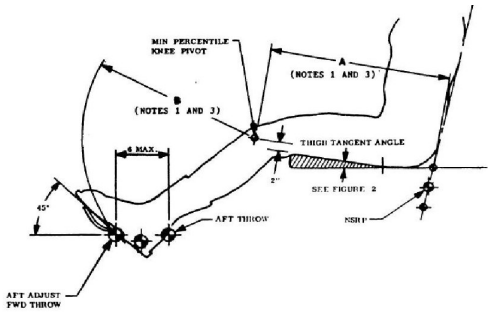


Figura 2.7: geometria pedaliera-posizione arretrata

Disposizione cockpit Dopo aver tenuto conto di tutte le osservazioni fatte in precedenza si é scelta questa disposizione finale del cockpit come mostrato in figura 2.8.

Figura 2.8: disposizione cockpit

In appendice A sono riportati tutti i disegni quotati dei pezzi necessari alla realizzazione del mockup.

Capitolo 3

Dimensionamento piastra di base e scelta dei materiali

3.1 Considerazioni generali

Per il dimensionamento della piastra di base si sono fatte delle considerazioni del tutto qualitative. Cosí facendo, in poco tempo, si sono ottenuti dei valori di massima per i carichi statici e dinamici, che la piastra di base dovrá reggere. I materiali su cui puó ricadere la nostra scelta sono essenzialmente due: il legno e il ferro.

Di conseguenza si sono fatti due dimensionamenti, uno con legno d' abete e l' altro con ferro. Per entrambi si é considerato solamente il carico dovuto al sedile e al 95 percentile, moltiplicato per un fattore di sicurezza pari a 5, concentrato agli appoggi. Inoltre per semplicitá si é considerato un modello a trave perfettamente appoggiato agli estremi.

Per il legno si sono considerate le seguenti caratteristiche:

- $\rho = 500 kg/m^3$
- $E = 9000 MPa$
- $\sigma_R = 85 N/mm^2$
- $\nu = 0.45$

Per quanto riguarda il ferro invece si sono considerati i seguenti valori:

- $\rho = 7800 kg/m^3$
- $E = 210000 MPa$

- $\sigma_R = 340N/mm^2$
- $\nu = 0.33$

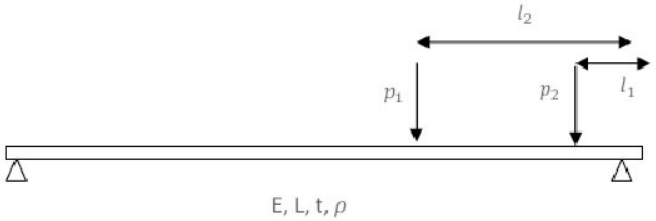


Figura 3.1:

La figura sopra schematizza il modello considerato per il dimensionamento. Di seguito si riportano i valori delle grandezze segnate e lo spessore ottenuto, sia con un legno d' abete di sezione quadrata piena, sia con un ferro di sezione quadrata cava.

- $l_1 = 66mm$
- $l_2 = 504mm$
- $p_1 = p_2 = 2500N$

Per il legno a sezione quadrata vale questa formula (tratta da [8]) per calcolare lo spessore:

$$t = \sqrt{\frac{6M_f}{bt^2}} = 0.746 \approx 1mm \tag{3.1}$$

Invece, per il ferro a sezione rettangolare cava vale la seguente espressione (tratta da [8]) (ipotizzando uno spessore esterno di 0.03m):

$$t = t_1 - \sqrt[3]{\frac{\sigma_R b_1 t_1^3 - 6M_f t_1}{\sigma_R b_1}} = 0.23mm \tag{3.2}$$

Per motivi pratici e per non appesantire troppo il mockup é stato scelto di utilizzare il legno d' abete. Dopo aver scelto il materiale si é verificato inoltre che per tale spessore non si avesse una freccia troppo alta. Per una trave appoggiata ad entrambi i lati caricata da un carico concentrato vale la seguente formula (tratta da [8]):

$$f = \frac{Pc^2c_1^2}{3EJl} = 280.25mm \tag{3.3}$$

dove si sono considerati i seguenti valori,

- $l = 1400$mm
- $P = 5000$N
- $J = \frac{1}{12}bh^3 = 96250mm^4$
- $b = 1155$mm
- $h = 10$mm
- $c = 896$mm
- $c_1 = 504$mm

Come si puó notare il valore di freccia é fin troppo ampio e provocherebbe non pochi problemi in fase di sperimentazione. Per evadere questo problema si é deciso di raddoppiare lo spessore della piastra di base cosí da ottenere una freccia molto piú contenuta, pari a 35mm.

Capitolo 4

Calcolo prima frequenza di vibrare piastra di base

Un' ulteriore importante verifica é quella di controllare che la prima frequenza di vibrare della piastra di base non sia talmente bassa da rientrare nel campo di frequenze di nostro interesse, innescando cosí pericolosi cicli di amplificazione degli spostamenti o al limite portando alla risonanza l' intera struttura. Esistono diversi metodi, sia analitici che sperimentali, per il calcolo approssimato della prima frequenza di vibrare.

Qui di seguito si riportano sia un metodo sperimentale (ESDU) che uno analitico, in piú vengono fatte due tipi di correzione al valore ottenuto per tenere in considerazione, almeno in parte, dell' abbassamento della frequenza dovuto alle masse appoggiate alla piastra.

4.1 Metodo ESDU

La frequenza naturale di una piastra non caricata é ottenuta da questa espressione (tratta da [9]):

$$f = F_b c \frac{t}{b^2} \tag{4.1}$$

Per ogni serie di condizioni e modalitá di bordo i dati sono presentati in termini di coefficienti, un metodo piú conveniente per l' uso di questa formula. Nel nostro caso si é considerato il caso di vincolo di appoggio su tutti e quattro i lati e di seguito si riporta il grafico 4.1 del coefficiente F_b in funzione del rapporto tra i lati $\frac{b}{a}$.

Avendo calcolato un rapporto dei lati $\frac{b}{a}$ pari a 0.825 e avendo considerato

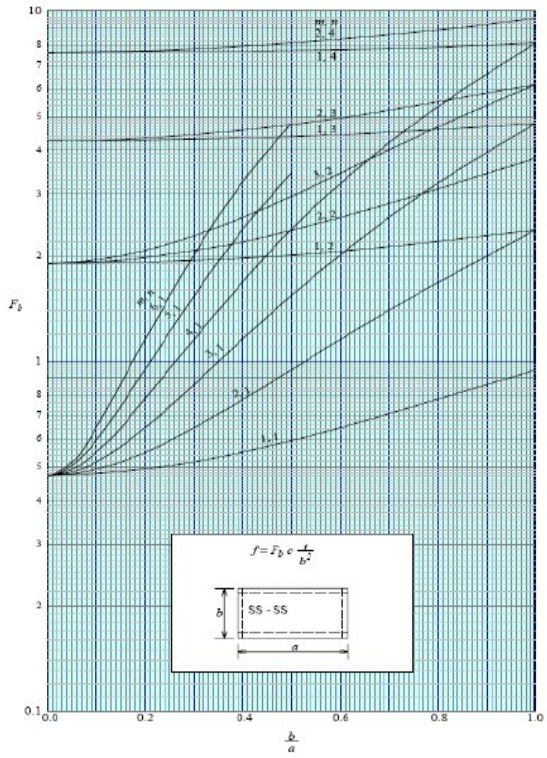


Figura 4.1: diagramma sperimentale ESDU

la prima frequenza di vibrare si é ottenuto, dal grafico, un valore del coefficiente F_b pari a 0.8 e quindi un valore della frequenza intorno ai 50Hz. Da notare che i valori riportati nei grafici sono validi per piastre con rapporti $\frac{b}{t}$ superiori a 10. Nel nostro caso si é verificato di essere abbondantemente al di sopra del rapporto minimo indicato.

4.2 Metodo analitico

Nel caso di piastra omogenea e isotropa rettangolare, avente due lati di lunghezza a e b, appoggiata su tutto il contorno e non soggetta a precarichi, le forme modali possono essere ipotizzate estendendo a due dimensioni il risultato ottenuto per la trave appoggiata (Ref [10]):

$$w(x,y) = \sum_{i=1}^{\infty}\sum_{k=1}^{\infty} \sin\frac{i\pi x}{a}\sin\frac{k\pi y}{b} \tag{4.2}$$

Per la frequenza propria, considerando uno sviluppo arrestato al prim' ordine sia in x che in y, si ottiene (Ref [10]):

$$\omega_{1,1} = \pi^2 \sqrt{\frac{Et^3}{12(1-\nu)m}} \left(\left(\frac{1}{a}\right)^2 + \left(\frac{1}{b}\right)^2 \right) = 48.23Hz \tag{4.3}$$

Risultato in accordo con il precedente sperimentale.

Prima correzione metodo analitico con massa concentrata nel baricentro Il precedente calcolo della prima frequenza di vibrare della piastra non tiene minimamente in considerazione il fatto che, in realtá, la piastra é sottoposta a un carico normale. In modo da non complicare troppo i conti si considera tutto il carico (sedile piú pilota) concentrato nel suo baricentro, come illustrato di seguito.

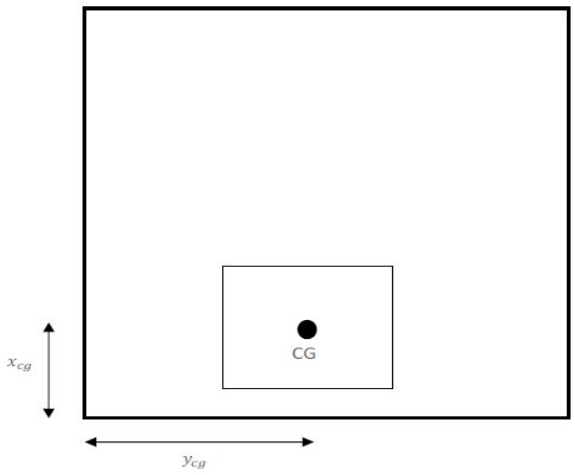


Figura 4.2: prima approssimazione con massa concentrata

A questo punto si scrive l' equazione del moto,

$$\tilde{M}\ddot{q} + \tilde{K}q = 0 \tag{4.4}$$

Si continuano a considerare le stesse funzioni di forma sinusoidale, una lungo x e una lungo y.

$$f(x,y) = \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right) \tag{4.5}$$

Dopodiché si risolvono gli integrali di superficie per calcolare la massa

generalizzata $\tilde{M}$, la rigidezza generalizzata $\tilde{K}$ e il termine di massa concentrata generalizzata $\tilde{m}$ dovuto al carico normale alla piastra.

$$\begin{aligned}\tilde{M} &= \int_S f^T M f \, dS = \iint_0^{a,b} \sin\Big(\frac{\pi x}{a}\Big)^2 \sin\Big(\frac{\pi y}{b}\Big)^2 M \, dx \, dy \\ &= M \int_0^a \sin\Big(\frac{\pi x}{a}\Big)^2 \int_0^b \sin\Big(\frac{\pi y}{b}\Big)^2 dx \, dy \qquad (4.6) \\ &= M \frac{a}{2} \frac{b}{2}\end{aligned}$$

$$\tilde{K} = \int_S f''^T E J f'' \, dS \qquad (4.7)$$

$$\tilde{m} = \sin\Big(\frac{\pi x_{cg}}{a}\Big)^2 \sin\Big(\frac{\pi y_{cg}}{b}\Big)^2 m \qquad (4.8)$$

Infine si riscrive l' equazione del moto aggiungendo il termine di massa concentrata e si ricava il nuovo valore della prima frequenza propria.

$$(\tilde{M} + \tilde{m})\ddot{q} + \tilde{K} q = 0 \qquad (4.9)$$

$$\tilde{\omega_1} = \sqrt{\frac{\omega_1^2}{1 + \frac{\tilde{m}}{\tilde{M}}}} = 14.86 Hz \qquad (4.10)$$

Come si puó notare il valore della prima frequenza si abbassa notevolmente, quasi entrando nella fascia di frequenze di nostro interesse. Sebbene l' approssimazione utilizzata sia molto spartana, si puó comunque giá immaginare la necessitá di un irrigidimento della struttura onde evitare pericolosi problemi di risonanza.

Seconda correzione metodo analitico con 4 masse concentrate agli appoggi Dopo aver fatto una correzione di primo livello (massa concentrata nel baricentro), si é voluta fare una controprova con un' approssimazione di secondo livello, considerando quattro masse concentrate agli appoggi in modo da distribuire in modo piú uniforme l' effetto del carico normale alla piastra, come illustrato nella figura 4.3 alla pagina successiva.

Come fatto precedentemente viene riscritta l' equazione di moto aggiungendo l' effetto delle quattro masse concentrate.

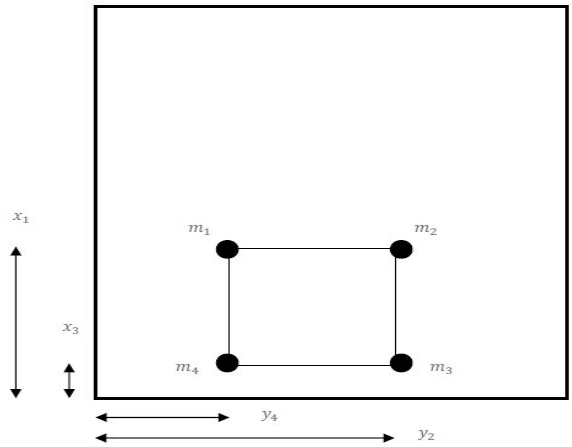


Figura 4.3: seconda approssimazione con masse concentrate

$$\Big(1 + \frac{\tilde{m_1} + \tilde{m_2} + \tilde{m_3} + \tilde{m_4}}{\tilde{M}}\Big)\ddot{q} + \omega_1^2 q = 0 \tag{4.11}$$

Ogni massa concentrata generalizzata viene calcolata come di seguito

$$\tilde{m_i} = \sin\Big(\frac{\pi x_i}{a}\Big)^2 \sin\Big(\frac{\pi y_i}{b}\Big)^2 m_i \qquad \text{con i = 1,2,3,4} \tag{4.12}$$

E infine si ricava il valore della prima frequenza di vibrare.

$$\tilde{\omega_1} = \sqrt{\frac{\omega_1^2}{1 + \frac{\tilde{m_1}+\tilde{m_2}+\tilde{m_3}+\tilde{m_4}}{\tilde{M}}}} = 16.57Hz \tag{4.13}$$

Anche in questo caso si puó notare un notevole abbassamento della frequenza del primo modo, di conseguenza si é ritenuto necessario un irrigidemento della piastra di base attraverso l' innesto di tre traversine metalliche in corrispondenza dei punti d' ingresso del carico posizionate al di sotto della piastra stessa.

Capitolo 5

FlightGear

5.1 Introduzione a FlightGear

FlightGear é un libero simulatore di volo sviluppato cooperativamente su Internet da un gruppo di simulazione di volo e da appassionati di programmazione. Le ragioni per cui, nonostante esistano tanti simulatori di volo, sono state spese migliaia di ore di programmazione e progettazione per sviluppare un simulatore di volo libero sono innumerevoli, ma le piú rilevanti sono le seguenti:

- tutti i simulatori commerciali hanno un grave inconveniente: sono realizzati da un piccolo gruppo di sviluppatori che definiscono le loro proprietá in base a quanto é importante per loro e che forniscono un' interfaccia limitata all' utente finale. Al contrario, FlightGear é progettato dal popolo e per il popolo con piena interfaccia per l' utente.

- i simulatori commerciali sono di solito un compromesso di funzionalitá e usabilitá. La maggior parte degli sviluppatori commerciali vogliono essere in grado di servire un segmento il piú ampio possibile di popolazione, compresi i piloti seri, i principianti e anche i giocatori occasionali. In realtá il risultato é sempre un compromesso a causa di scadenze e finanziamenti. Siccome FlightGear é un software libero ed aperto, non c' é bisogno di un simile compromesso.

- a causa della loro natura "closed-source", i simulatori commerciali tengono solo gli sviluppatori con eccellenti idee e capacitá per contribuire allo sviluppo dei prodotti. Con FlightGear, gli sviluppatori sono di tutti i livelli e le idee hanno la possibiltá di avere un enorme impatto sul progetto. Contribuendo ad un progetto grande e

complesso come FlightGear é molto gratificante e fornisce agli sviluppatori un grande orgoglio il sapere che possano plasmare il futuro di un simulatore cosí grande.

- al di lá di tutto il resto, alla fine é solo puro divertimento! Tutti siamo in grado di comprare un oggetto giá pensato e realizzato, ma c' é qualcosa di speciale nel realizzarlo noi stessi.

I punti di cui sopra costituiscono la base del motivo per cui FlightGear é stato creato. Con queste motivazioni si é stabilito di progettare un simulatore di volo di alta qualitá che mirasse ad essere "civile", "multi-piattaforma", "aperto", "sostenibile" ed "estendibile" dall' utente. Di seguito vengono prese in considerazione individualmente queste caratteristiche:

Civile: il progetto si rivolge principalmente a simulazioni di volo civile. Deve essere approvato per la simulazione di aviazione generale cosí come per quella civile. L' obiettivo a lungo termine é quello di riuscire a far approvare dalla FAA FlightGear come un dispositivo di addestramento al volo. Al momento non é un simulatore di combattimento, tuttavia, tali caratteristiche non sono esplicitamente escluse.

Multi-piattaforma: gli sviluppatori stanno cercando di mantenere il codice il piú possibile indipendente dalla piattaforma. Questo si basa sull' osservazione che le persone interessate a eseguire simulazioni di volo utilizzano una varietá di hardware e sistemi operativi. FlightGear supporta i seguenti sistemi operativi: Linux, Windows NT/2000/XP, Windows 95/98/ME, BSD UNIX, Sun- OS, Mac OS X.

Aperto: il progetto non é limitato a una cerchia ristretta e statica di sviluppatori. Chiunque sia in grado di contribuire é il benvenuto. Il codice (compresa la documentazione) é coperto da copyright nel rispetto dei termini della GNU General Public License (GPL). Il termine GPL é spesso frainteso. In termini semplici si afferma che si puó copiare e distribuire liberamente il programma come da licenza. Tuttavia, quando si distribuisce il software é necessario renderlo disponibile ai destinatari nel codice sorgente e si devono conservare i diritti d' autore originali. In breve: "*Puoi fare qualsiasi cosa con il software tranne che renderlo non-free*"

Estendibile e sostenibile: a differenza di simulatori piú commerciali, lo scenario e i formati degli aeromobili di FlightGear, le variabili

> interne e tutto il resto sono accessibili all' utente e sono documentati sin dal principio. Anche senza documentazione esplicita di sviluppo si puó sempre andare al codice sorgente per vedere come funziona qualcosa. L' obiettivo degli sviluppatori é di costruire un motore di base al quale i progettisti degli scenari, gli ingegneri dei pannelli, gli artisti del suono e tanti altri possano contribuire. La speranza é che il progetto, compresi gli sviluppatori e gli utenti finali, beneficeranno della creativitá e delle idee di centinaia di talenti in tutto il mondo.

Descrizione generale Il motore di simulazione in FlightGear viene chiamato SimGear. Esso é un insieme di librerie open-source progettate per essere usate come "mattoni" per la rapiditá di montaggio di simulazioni 3D, giochi e applicazioni di visualizzazione.

SimGear é un progetto relativamente nuovo, e mentre un po' di codice é stato scritto in collaborazione con il progetto FlightGear, l' interfaccia finale e le modalitá sono ancora in evoluzione.

FlightGear é utilizzato sia come un' applicazione end-user sia in ambienti accademici e di ricerca, per lo sviluppo e la realizzazione di idee di simulazione di volo.

La personalizzabilitá di FlightGear é illustrata dalla vasta gamma di modelli di aerei disponibili, da alianti agli elicotteri, e da aerei a jet da combattimento. Questi modelli di aerei sono stati apportati da molte persone diverse.

Attualmente esiste solo un motore per la realizzazione del terreno, TerraGear. Esso é una collezione di strumenti open-source e di librerie di rendering che puó trasformare dati GIS disponibili pubblicamente in rappresentazioni 3D (vale a dire i modelli 3D o mappe 3D) della terra per l' utilizzo in tempo reale del rendering dei progetti.

Gli aerei di FlightGear utilizzano uno dei tre principali modelli di volo JSBSim, YAsim, o UIUC a partire dalla versione 0.9.10.

Modelli dinamici Storicamente, FlightGear é stato basato su un modello di volo ereditato da LaRCsim. Dato che questo aveva diverse limitazioni, ci sono stati diversi tentativi di sviluppare o includere modelli di volo alternativi. Come risultato, FlightGear supporta diversi modelli di volo diversi, a scelta dalla fase di comando.

Il piú importante é il modello di volo JSB sviluppato da Jon Berndt. In realtá il modello di volo JSB é parte di un progetto autonomo denominato JSBSim.

Per quanto riguarda gli aerei, il modello di volo JSB attualmente fornisce il supporto per il Cessna 172, il Cessna 182 e 310 e per un aeroplano

sperimentale chiamato X15. Jon e il suo gruppo si stanno attrezzando verso un modello molto accurato di volo, e cosí il modello JSB é diventato il modello di volo di default per FlightGear.

Un' alternativa interessante é Christian Mayer che ha sviluppato un modello di volo per una mongolfiera. Inoltre, Curt Olson ha integrato una speciale modalitá per gli UFO, che permette di volare molto velocemente da un punto ad un altro.

Recentemente Andrew Ross ha contribuito con un altro modello di volo chiamato YASim. Allo stato attuale vengono simulati parecchi velivoli come ad esempio DC-3, Boeing 747, Harrier e tanti altri. YASim ha un approccio sostanzialmente diverso in quanto si basa su informazioni geometriche piuttosto che su coefficienti aerodinamici. JSBSim sará esatto per ogni situazione di volo conosciuta e testata, ma potrebbe avere strani comportamenti irrealistici al di fuori del volo normale. YASim sará ragionevole e coerente in quasi tutte le situazioni di volo, ma é probabile che si differenziano per numero di prestazioni.

Come ulteriore alternativa vi é il modello di volo UIUC, sviluppato da un team della University of Illinois a Urbana-Champaign. Questo lavoro é stato inizialmente orientato verso la modellazione di aeromobili in condizioni di ghiaccio con un sistema intelligente per meglio consentire ai piloti di volare in sicurezza anche in condizioni di ghiaccio. Mentre questa ricerca continua, il progetto si é ampliato per includere la modellazione "non lineare" dell' aerodinamica, che si traduce in un maggior realismo in atteggiamenti estremi come lo stallo e il volo con alti angoli d' attacco. Due buoni esempi che illustrano questa funzionalitá sono il deltaplano Airwave Xtreme 150 e il velivolo dei fratelli Wright.

"Property tree" Il cosiddetto "albero delle proprietá" in FlightGear é generalmente considerato il sistema nervoso centrale e uno dei maggiori patrimoni di FlightGear. L' albero delle proprietá di sistema in FlightGear é generalmente utilizzato da quasi tutti i sottosistemi FlightGear che sono fondamentalmente legati insieme da esso, in altre parole: é l' albero di proprietá il meccanismo che consente di "condividere" importanti dati runtime tra diverse componenti di FlightGear.

Anche se questo non necessariamente si applica a tutte le strutture di dati interne, FlightGear lo fa nella maggior parte dei casi e lo applica a quelle variabili che possono avere bisogno di essere modificate in fase di esecuzione, sia a fini di personalizzazione o altri usi. Inoltre, l' albero delle proprietá si é reso molto semplice per pubblicare od esporre nuove variabili da C++.

I concetti e i meccanismi dietro all' "albero delle proprietá" possono non essere immediatamente evidenti per i principianti di FlightGear.

Quando una simulazione é in esecuzione, tutte le variabili come posizione, velocitá, flaps, luci di cabina, e tante altre sono calcolate e manipolate attraverso un albero di proprietá.

L' albero "virtuale" che esegue la simulazione appare molto simile a una directory o un file struttura, come in figura 5.1:

Figura 5.1: property tree

Alcune di queste variabili sono "calcolate" all' interno della simulazione, mentre altre possono essere manipolate o addirittura essere impostate anche dall' interfaccia httpd.

Ció che rende FG potente é che un nuovo aeromobile puó essere facilmente progettato con il suo set unico di proprietá che in qualche modo influiscono sulla simulazione. Il modello di aeromobile dispone di un file .XML di proprietá all' interno della struttura di proprietá.

É per questo che l' albero di proprietá non é coerente con le variabili fisse, esse sono state create dinamicamente.

L' albero di proprietá é leggibile, scrivibile, accessibile e manipolabile in diversi modi, come ad esempio:

- codice interno compilato entro FlightGear - C / C++;
- Nasal script - questo é uno scripting come JavaScript, con lettura / scrittura per l' albero di proprietá. Questo é il modo in cui vengono implementati la maggior parte degli aerei;
- Socket che si trovano in / out / bi - questo consente di inviare o ricevere alla sim FG via socket;

- Interfaccia telnet;
- Interfaccia html.

Per esempio, la modalitá multiplayer si realizza attraverso lo scambio di insiemi di variabili dall' albero (altezza, posizione, velocitá, ecc.).

5.2 YASim, yet another simulator

Come detto precedentemente FlightGear ha la possibilitá d' interfacciarsi a diversi simulatori dinamici, ma per quanto riguarda la dinamica degli elicotteri, senza dubbio, il migliore é YASim. In questo simulatore di volo, a differenza di tutti gli altri, si prendono in considerazione, oltre alle equazioni del corpo rigido, due aspetti fondamentali per la modellazione della dinamica dell' elicottero:

- il calcolo della velocitá di downwash;
- la dinamica della pala.

Per prima cosa dopo aver trovato la massa totale del sistema, tutte le forze agenti su di esso, aerodinamiche, gravitazionali, inerziali ed esterne (ad esempio quelle che nascono da un controllo imposto dal pilota), il programma calcola le accelerazioni lineari nelle tre direzioni secondo questa formula:

$$a_{lin} = \frac{F_{tot}}{M} \tag{5.1}$$

e quelle rotazionali utilizzando la formula seguente:

$$a_{rot} = I^{-1}(C - (\omega \times I\omega)) \tag{5.2}$$

dove I é il tensore d' inerzia ω é la velocitá rotazionale e C é la coppia totale a cui é soggetto il corpo.

$$I = \begin{bmatrix} mz^2 + my^2 & -mxy & -mzx \\ -mxy & mx^2 + mz^2 & -myz \\ -mzx & -myz & mx^2 + my^2 \end{bmatrix}$$

Ricavate le accelerazioni il programma é in grado di aggiornare, per ogni step temporale, le posizioni e gli orientamenti del corpo calcolandosi, attraverso semplici integrazioni, le velocitá e gli spostamenti sia lineari che rotazioni.

Come si puó facilmente intuire il calcolo delle quantitá dinamiche ha una forte rilevanza nel programma di simulazione di volo considerato, ma sarebbe superficiale pensare che si limiti soltanto a ció. Grande importanza hanno anche il calcolo delle prestazioni del motore (in YASim sono previsti differenti tipi di motore tra cui il jet, il motore a pistoni e il turboelica), il calcolo delle forze scambiate dal carrello in fase d' atterraggio e particolari tipi di decollo e atterraggio, come il decollo con barra di lancio e atterraggio con gancio (come sulle portaerei).

Calcolo velocitá di downwash Il parametro fondamentale per determinare la trazione del rotore é la velocitá di downwash. Per poterla calcolare viene, prima di tutto, determinata la normale al disco del rotore in modo corretto considerando sia una possibile variazione dovuta al rollio, sia al beccheggio

$$\vec{n} = \vec{z}\cos\varphi\cos\phi + \vec{x}\sin\varphi + \vec{y}\sin\phi \tag{5.3}$$

dove sono stati definiti φ e ϕ rispettivamente l' angolo di beccheggio e l' angolo di rollio. Dopodiché si calcola la velocitá di downwash sia appena sotto il disco, sia ad una distanza $dist$ da esso, assumendo una distribuzione gaussiana della velocitá con σ proporzionale a $dist^2$. Si assume che σ sia pari a metá del diametro del rotore per $dist = 0$ e pari al diametro stesso per $dist = (2R\sqrt{2})$. Quindi si avrá nel primo caso,

$$\vec{w_1} = 0.8\omega r\sin\alpha_r \tag{5.4}$$

avendo definito

$$\alpha_r = \delta + 2\zeta\frac{r}{2R} - 0.7\zeta \tag{5.5}$$

dove δ é il valore di collettivo considerato, ζ lo svergolamento della pala, r la posizione della sezione considerata rispetto il centro del rotore e R il raggio del rotore. Invece nel secondo caso,

$$\vec{w_2} = \vec{v}_{1bar}\frac{2R}{\sqrt{2\pi}\sigma}e^{\frac{-.5r^2}{\sigma^2}}\frac{2R}{2\sigma} \tag{5.6}$$

dove sono state definite

$$\sigma = R + \frac{dist^2}{8R} \tag{5.7}$$

$$\vec{w}_{1bar} = 0.56R\omega \sin\alpha \tag{5.8}$$

Dopo aver definito i pesi delle due velocitá di downwash attraverso una funzione peso g,

$$g = e^{\frac{-2dist}{2R}} \tag{5.9}$$

si puó ottenere cosí la velocitá di downwash effettiva, diretta nella direzione opposta alla normale la disco

$$w = gw_1 + (1-g)w_2. \tag{5.10}$$

Dinamica della pala La pala viene considerata rigida, libera di flappeggiare e di variare la sua incidenza rispetto il vento asintotico. Per semplicitá viene considerata solo la dinamica della pala attorno alla cerniera di flappeggio, in modo da calcolare l' angolo di flappeggio della pala in condizione di equilibrio tra la forza centripeta e la portanza. Prima del calcolo delle forze aerodinamiche nel sistema di riferimento locale si calcola l' incidenza locale di ogni tratto di pala, considerando l' effetto dello svergolamento e del fattore di Prandtl, il quale prende in considerazione il fatto che all' estremitá aumenta la velocitá indotta del fluido e quindi diminuisce la portanza locale.

$$F = \left(\frac{2}{\pi}\right)\cos^{-1}\left(e^{-\frac{N_b}{2}\left(\frac{1-r}{r\phi}\right)}\right) \tag{5.11}$$

$$\alpha_{loc} = \alpha + \zeta r - \zeta r_\alpha \tag{5.12}$$

dove ζ é lo svergolamento della pala, $r_\alpha = 0.7$ é la posizione adimensionale lungo la pala dove viene misurata l' incidenza e α é l' incidenza dovuta ai controlli del pilota (ciclico e collettivo) diminuita dell' effetto di δ_3.

A questo punto si possono determinare le forze aerodinamiche, resistenza e portanza facendo distinzione tra portanza con e senza comando

di ciclico, e la forza centrifuga

$$D = \frac{1}{2}\rho v^2 S C_D \tag{5.13}$$

$$L = L_{wocyc} + \kappa(L_{withcyc} - L_{wocyc}) \tag{5.14}$$

$$F_\omega = m\omega^2 r \tag{5.15}$$

dove il parametro κ dipende dalla velocitá di rotazione del rotore, dall' eccentricitá e dal contributo di portanza dovuto al comando di collettivo. I contributi della portanza sono calcolati con le seguenti formule:

$$L_{wocyc} = \frac{1}{2}\rho v^2 S C_{L_{wocyc}} \tag{5.16}$$

$$L_{withcyc} = \frac{1}{2}\rho v^2 S C_{L_{withcyc}}. \tag{5.17}$$

Il particolare accorgimento preso per quanto riguarda la portanza nasce dal fatto che il rotore é un sistema in risonanza dove un comando di ciclico é portato ad assumere valori sempre crescenti.

File di input Il modo piú rapido e "user-friendly" che un utente ha per dialogare con YASim é quello di scrivere un file .XML d' input nel quale vengono riportate tutte le caratteristiche principali di un qualsiasi velivolo. La sintassi del codice é molto semplice e ricorda vagamente quella maggiormente nota chiamata HTML. In questo file l' utente deve riportare tutte le caratteristiche aerodinamiche, strutturali, propulsive e accessorie necessarie per la simulazione del volo.

Molti sono i parametri che si possono dare in ingresso al programma scrivendo questo semplice file. Di seguito vengono riportati quelli di maggior interesse per un velivolo a pala rotante:

airplane l' elemento principale del file il quale a sua volta ha come attributo

mass la massa a vuoto in libbre del velivolo.

approach i parametri del velivolo per l' approccio. Il solutore genererá un velivolo che corrisponde a queste impostazioni. Questo parametro puó avere diversi attributi

speed la velocitá in nodi TAS per l' approccio,

aoa l' angolo di attacco per l' approccio in gradi.

cruise come per l' approccio, ma si considerano i parametri per la crociera

speed velocitá di crociera,

alt quota di crociera in piedi sul livello del mare.

control-setting questo tag viene utilizzato per definire una particolare impostazione per un asse di controllo all' interno del <cruise> o <approach> tag

axis nome del controllo in ingresso,

value valore del controllo

rotor il rotore. Utilizzato per la simulazione di elicotteri. Si puó avere uno, due o anche piú rotori. Quando si definisce un rotore non é necessario specificare un' ala o un piano di coda orizzontale. Il rotore genera downwash agente su tutte le superfici aerodinamiche e sulla fusoliera. Per questa tag sono presenti numerosi attributi:

name il nome del rotore,

x,y,z la posizione del centro del rotore,

nx,ny,nz i versori della normale al rotore,

fx,fy,fz i versori che definiscono il vettore che punto verso l' avanti,

diameter diametro del disco in metri,

numblades il numero di pale,

weightperblade il peso in libbre di ogni pala,

chord la corda di ogni pala,

twist lo svergolamento della pala,

taper la rastrematura della pala,

rel-len-where-incidence-is-measured posizione in cui viene misurata l' incidenza della pala in caso di svergolamento,

rel-len-blade-start posizione da cui parte la pala rispetto al centro del rotore,

rpm velocitá di rotazione del rotore in giri al minuto,

phi0 posizione iniziale del rotore,

ccw determina se il rotore gira in senso antiorario od orario,

maxcollective massima incidenza per il collettivo in gradi,

mincollective minima incidenza per il collettivo in gradi,

maxcyclicele massima incidenza per il flappeggio longitudinale in gradi,

mincyclicele minima incidenza per il flappeggio longitudinale in gradi,

maxcyclicail massima incidenza per il flappeggio laterale in gradi,

mincyclicail minima incidenza per il flappeggio laterale in gradi,

incidence-stall-zero-speed incidenza di stallo a velocitá nulla,

incidence-stall-half-sonic-speed incidenza di stallo a Mach = 0.5,

lift-factor-stall fattore che tiene in considerazione la diminuzione di portanza durante lo stallo,

drag-factor-stall fattore che tiene in considerazione l' aumento di resistenza durante lo stallo,

pitch-a angolo d' incidenza del collettivo,

pitch-b angolo d' incidenza del collettivo,

airfoil-lift-coefficient coefficiente aerodinamico,

airfoil-drag-coefficient0 coefficiente aerodinamico,

airfoil-drag-coefficient1 coefficiente aerodinamico,

rotor-correction-factor fattore che corregge l' effetto dei vortici nel calcolo della portanza,

flapmin minimo angolo di flappeggio,

flapmax massimo angolo di flappeggio,

flap0 angolo di flappeggio senza rotazione,

dynamic fattore che modifica la reazione del rotore a un controllo di input,

rellenflaphinge lunghezza relativa dal centro del rotore alla cerniera di flappeggio,

delta3 fattore che considera l' effetto dell' angolo δ_3 il quale provoca una diminuzione dell' incidenza quando il rotore sta flappeggiando. In YASim il valore di questo parametro é pari a $\tan^{-1}(\delta_3)$,

delta un fattore per la costante di smorzamento per il flappeggio,

number-of-parts numero di parti in cui é diviso il rotore,

number-of-segments numero di segmenti in cui é simulato il rotore in ogni direzione,

cyclic-factor la risposta del rotore ad un ingresso di ciclico é difficile da calcolare (é un oscillatore smorzato in risonanza, alcuni parametri hanno un impatto molto grande per la risposta ciclica). Con questo parametro é possibile regolare il simulatore in modo da seguire meglio la realtá,

downwashfactor fattore per il downwash del rotore.

control-input questo elemento gestisce un mapping dalle proprietá del 'property tree' ai valori degli oggetti del velivolo.

axis nome della proprietá da usare come input,

control nome del controllo da associare alla proprietá

invert inverte il valore della proprietá prima di applicarla all' oggetto,

src0/src1/dst0/dst1 rappresenta un mapping lineare dal comando di input al valore di output. Un input compreso in un range tra src0/src1 viene mappato linearmente su un range tra dst0/dst1.

rotorgear questa tag serve per associare ad ogni rotore un motore. A questa tag sono associati i seguenti attributi

max-power-engine la massima potenza del motore in kW,

engine-prop-factor il motore lavora come un regolatore PD. Questo attributo é la larghezza della banda di regolamento o, in altre parole, l' inverso del fattore proporzionale del regolatore,

engine-accel-limit massimo rateo di accelerazione del motore,

max-power-rotor-brake massima potenza del freno motore in kW,

rotorgear-friction potenza persa negli attriti

cockpit questa tag definisce la posizione del cockpit

x,y,z la posizione del "punto di vista" del pilota

fuselage questa tag crea una struttura tubolare simile alla fusoliera. Dal solutore viene associata una massa e una forza aerodinamica distribuita per ogni struttura

ax,ay,az posizione di una fine del tubo,

bx,by,bz posizione della seconda fine del tubo,

width larghezza del tubo in metri,

taper rastremazione del tubo espressa come frazione della larghezza massima,

midpoint posizione di massima larghezza del tubo,

idrag moltiplicatore per la resistenza indotta dall' oggetto.

hstab definisce il piano di coda del velivolo

length lunghezza dell' ala,

chord la corda dell' ala alla base,

taper rastremazione,

sweep svergolamento.

stall questa é una tag che definisce il comportamento allo stallo dell' ala

aoa angolo di stallo relativo all' ala,

width l'arghezza dello stallo in gradi. Un valore alto definisce uno stallo gentile.

peak picco di portanza relativo alla parte post-stallo

vstab definisce il piano di coda verticale. Molto importante da sottolineare che questo tipo di superfici non vengono coinvolte nel calcolo della soluzione dal solutore

gear definisce il carrello di atterraggio. Gli attributi a questa tag sono i seguenti

x,y,z posizione dell' estremitá del carrello totalmente esteso,

compression corsa di schiacciamento del carrello,

sfric coefficiente di attrito statico,

dfric coefficiente di attrito dinamico,

spring moltiplicatore adimensionale per definire la rigidezza del carrello

tank definisce i serbatoi per il carburante

x,y,z posizione del serbatoio,

capacity capacitá del serbatoio in libbre.

ballast definisce un meccanismo per modificare la distribuzione di massa del velivolo. L' ambiente zavorra specifica che una determinata quantitá del peso a vuoto del velivolo deve essere inserito in una determinata posizione. Il peso rimanente non-zavorra sará distribuito "intelligentemente" in tutta la fusoliera e le ali. Da notare che cosí facendo non cambia il peso a vuoto del velivolo.

x,y,z posizione della zavorra,

mass valore della zavorra. Puó essere anche un valore negativo nel caso in cui si volesse alleggerire qualche particolare del velivolo.

weight definisce una massa aggiunta al peso a vuoto del velivolo, come ad esempio passeggeri, munizioni o cargo.

x,y,z posizione della massa aggiunta,

mass-prop il nome della proprietá associata alla massa,

size dimensione aerodinamica in metri dell' oggetto. Questo attributo é importante per quegli oggetti che si trovano all' esterno, poiché causano un aumento di resistenza

thruster definisce un "oggetto motore" molto semplice di sola spinta

thrust massima spinta in libbre,

x,y,z posizione sul corpo dove verrá applicata la spinta,

vx,vy,vz direzione della spinta in assi corpo.

Dopo aver scritto il file di input, YASim é in grado di generare il suo modello di velivolo, che come si puó vedere in figura 5.2 per il caso del BO105 é ridotto alle sue componenti fondamentali.

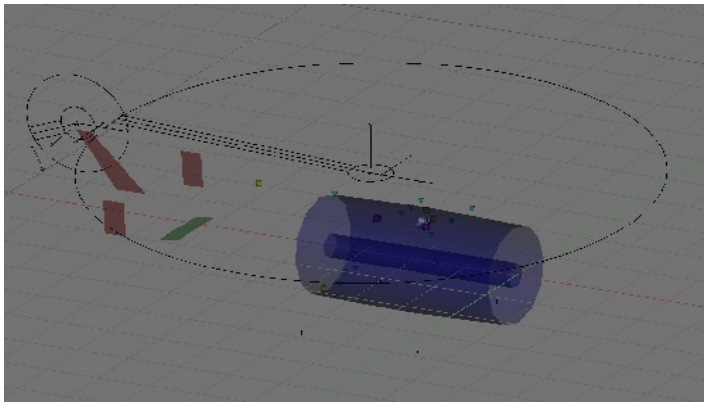


Figura 5.2: Bo105 YASim

Interfaccia YASim-FlightGear Per permettere alla simulazione di procedere in tempo reale, é necessaria una comunicazione reciproca tra i due programmi. Uno scambio reciproco d' informazioni tra il simulatore di volo (YASim) e il software per la simulazione grafica (FlightGear) é reso possibile grazie a un' interfaccia costituita dal "property tree", come descritto precedentemente e da due classi (*FGInterface* definita in src/FDM/flight.hxx, *YASim* definita in src/FDM/YASim/YASim.hxx) che permettono, istante per istante, l' interazione reciproca tra i due programmi.

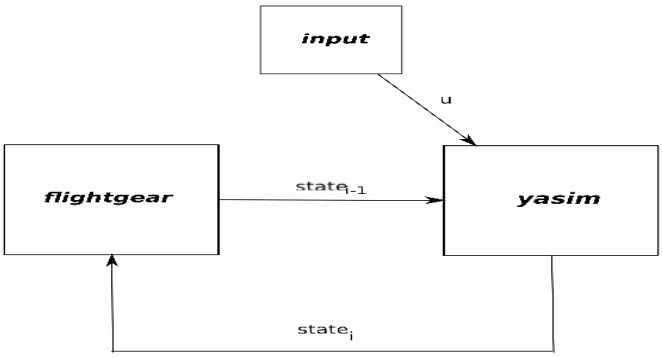


Figura 5.3: interfaccia

Quindi ad ogni istante di tempo FlightGear aggiorna l' albero delle proprietá, mentre YASim aggiorna lo stato, prendendo i valori degli ingressi e dello stato del velivolo al passo precedente, in base alle forze aerodinamiche, gravitazionali, propulsive e inerziali che si é calcolato al passo stesso.

Capitolo 6

Realizzazione dell' interfaccia MBDyn - FlightGear

L' interfaccia con cui un processo applicativo accede ai servizi di comunicazione (ossia in ambiente TCP/IP, l' interfaccia con lo strato di trasporto) é detta genericamente *Application Program Interface*(API).

In teoria ogni sistema elaborativo potrebbe inventarsi le proprie interfacce. In pratica l' interfaccia piú diffusa é quella universalmente nota come *socket*.

6.1 I pacchetti socket

L' interfaccia socket é apparsa la prima volta in UNIX con la versione 4.2 del Berkeley System Distribution (BSD), é stata poi portata sulle versioni commerciali da essa derivate (SunOs, Linux, ecc.) e, senza sostanziali modifiche, anche in altri sistemi operativi (ad esempio Windows).

L' interfaccia socket consiste in una serie di chiamate rese disponibili al programmatore. Il linguaggio di programmazione di riferimento é il C, tuttavia l' interfaccia socket é stata resa disponibile in molti altri linguaggi.

A livello trasporto sono disponibili due scelte: servizio "*connection oriented*" fornito tramite il protocollo TCP (*Transmission Control Protocol*) e servizio "*connectionless*" (o di tipo datagramma) fornito mediante il protocollo UDP (*User Datagram Protocol*).

Entrambi i protocolli poggiano sul sottostante IP (Internet Protocol) che é di tipo datagramma. Ma mentre TCP ne arricchisce le prestazioni in modo da garantire la sequenzialitá dei pacchetti e da evitare la loro perdita o duplicazione, UDP si limita a incapsulare ogni datagramma utente in un datagramma IP, fornendo cosí un servizio intrinsecamente inaffidabile. La scelta di UDP deve pertanto prevedere, a livello applicativo, un insieme di

provvedimenti atti a sopperire agli inconvenienti tipici della comunicazione a datagramma. Questo rende in genere l' uso di UDP piú difficoltoso.

Che cosa sono Un socket é un punto terminale di una comunicazione. Esso puó essere di due tipi:

stream: usa un servizio di trasporto affidabile (*connection oriented*) e fornisce all' utente un flusso di dati bidirezionale e continuo, cioé non strutturato in record.

datagram: usa un trasporto inaffidabile (*connectionless*) e fornisce un flusso bidirezionale in cui é mantenuta la distinzione in record.

Come si utilizzano Il meccanismo dei socket puó essere considerato come un' estensione delle operazioni di I/O. Un applicativo accede alle funzioni di I/O specificando un numero intero, il *file descriptor*, che gli viene assegnato da UNIX mediante una chiamata alla funzione *open()*. UNIX usa il file descriptor per puntare a una struttura di dati interna che contiene la descrizione del file.

Similmente UNIX gestisce, per ogni socket attivo, una struttura di dati interna a cui punta mediante un numero intero detto *socket descriptor*. La tabella dei file descriptor e dei socket descriptor é comune,quindi uno stesso numero non puó essere assegnato a un file e a un socket. Per creare un socket e farsi assegnare un socket descriptor si usa la chiamata di sistema *socket()*:

int socket(domain, type, protocol)

L' argomento *domain* indica l' ambiente di comunicazione che si vuole usare. In ambiente TCP/IP esso vale sempre AF_INET (Address Family Internet).

type: indica il tipo di socket voluto (SOCK_STREAM per socket di tipo stream; SOCK_DGRAM per socket di tipo datagram)

protocol: indica il protocollo che si vuole usare. Normalmente si usa il protocollo di default, che viene indicato con 0.

La chiamata restituisce un *socket descriptor*. In caso di errore restituisce -1 (se ad esempio non si ha il permesso di creare il socket, oppure si é specificato un protocollo incompatibile col tipo di socket). Quindi per creare un socket di tipo stream si usa la sequenza:

int sock;
int sock = (AF_INET, SOCK_STREAM, 0);

Come detto, il socket costituisce il punto terminale di una comunicazione. Tale punto é identificato da un indirizzo IP e da un numero di porta. A un socket possono far capo piú processi, il numero di porta identifica il processo. L' insieme di queste informazioni (indirizzo IP piú numero di porta) costituisce l' indirizzo del punto terminale.

Processi Client e Server La comunicazione fra processi in rete é di solito basata sul modello di interazione Client/Server. Un processo **Cliente** per accedere a un determinato servizio effettua una serie di richieste al processo **Servente**. Il servente per soddisfare ogni richiesta effettua alcune elaborazioni e fornisce una risposta. Quando il cliente ha inoltrato l' ultima richiesta e ricevuta l' ultima risposta l' interazione finisce. Nel caso piú semplice si ha una sola richiesta seguita da una sola risposta.

Come si vede, ció che distingue il cliente dal servente é il fatto che il cliente prende l' iniziativa dell' interazione. Quindi se i due processi usano un protocollo di comunicazione di tipo *connection oriented* il cliente sará sempre il chiamante e il servente il chiamato.

Un processo servente é sempre attivo, in attesa di una richiesta da parte di un cliente. Soddisfatta la richiesta si pone in attesa della prossima richiesta, ponendosi cosí in un loop infinito.

Un cliente TCP, ossia che vuole accedere a un servente tramite TCP, segue tipicamente i seguenti passi:

- specifica del punto terminale remoto
- allocazione di un socket
- formazione della connessione
- comunicazione col servente
- chiusura del socket

Specifica del punto terminale remoto Una comunicazione ha due punti terminali, uno locale e uno remoto. Il punto locale corrisponde al cliente stesso, quello remoto al servente. Specificare il punto terminale del servente significa costruirne l' indirizzo, che comprende indirizzo IP e numero di porta del servente. Si tratta di riservare un' area di memoria per la struttura server e di riempirne gli appositi campi con l' indirizzo IP e il numero di porta.

Allocazione di un socket Questa operazione é giá stata vista e serve per definire un descrittore di socket.

Formazione della connessione A questo punto il cliente puó richiedere la connessione al servente usando la chiamata di sistema *connect()*:

```
int connect(socket, addr, addrlen);
int socket;
struct sockaddr_in *addr;
int addrlen;
```

L' argomento *socket* é il descrittore del socket; *addr* é l' indirizzo della struttura che specifica il punto terminale remoto; *addrlen* é la lunghezza dell' argomento precedente.

La chiamata restituisce 0 in caso di successo, -1 in caso di errore.

Comunicazione col servente Dopo che la connessione é stata effettuata, inizia la fase di interazione fra cliente e servente. Essa dipende fortemente dal tipo di servizio e dal protocollo applicativo, per cui é impossibile definire uno schema di carattere generale.

É definito invece il meccanismo di scambio di dati attraverso il socket di tipo stream. Esso avviene come per un file, utilizzando le funzioni di I/O *write()* e *read()*. Precisamente, il cliente allocherá ad esempio un' area di memoria *buf*, di dimensione utile *size* byte, in cui formare le richieste da mandare al servente, e un' area *buf_from* di *size_from* byte in cui ricevere le risposte.

Le istruzioni per lo scambio di dati potranno allora essere del tipo:

```
char buf_to[SIZETO+1];
char buf_from[SIZEFROM+1];
int n;
write (sock, buf_to, sizeof(buf_to));
read(sock, buf_from, SIZEFROM)
```

L' operazione di invio della richiesta, effettuata mediante la funzione *write()*, non presenta particolaritá. Invece la lettura della risposta deve essere fatta in modo da premunirsi dalla possibilitá di frammentazioni provacate dal protocollo di trasporto.

Chiusura del socket Finita l' interazione cliente/servente il socket puó essere rilasciato dal cliente mediante la chiamata

close(sock)

Invece per quanto riguarda un server TCP, esso usa sempre due tipi di socket:

Master: lo assegna l' applicativo,

Slave: lo assegna il SO

Il primo serve a ricevere le connessioni in arrivo; il secondo a effettuare lo scambio di dati col client. Entrambi vanno dichiarati nel programma.

Esistono due diverse tipologie di server:

sequenziali: riescono a servire un cliente alla volta,

concorrenti: riescono a servire piú utenti contemporaneamente.

Per realizzare un server concorrente esistono due tecniche fondamentali:

1. un unico processo che usa delle chiamate a *listen* non bloccanti, dedica tempo a uno dei clienti attivi quando questo effettua una richiesta e quindi passa ad esaminare il successivo cliente attivo;

2. un processo dedicato a ogni cliente. Ogni nuova chiamata genera un nuovo processo.

6.2 Gestione dell' interfaccia tra FlightGear e MBDyn

In FlightGear le possibilitá di comunicazione con programmi esterni sono molteplici, le piú utilizzate sono:

trasmissione seriale modalitá di comunicazione tra dispositivi digitali nella quale le informazioni sono comunicate una di seguito all' altra e giungono sequenzialmente al ricevente nello stesso ordine in cui le ha trasmesse il mittente,

trasmissione da file modalitá per cui i dati vengono presi da file,

trasmissione via socket particolare modalitá di comunicazione via rete.

Un particolare accorgimento che bisogna prendere in considerazione per ogni tipo di comunicazione che si scelga di mettere in piedi é che FlightGear ha bisogno di un file di protocollo in formato .XML il quale definisce a ogni byte trasferito (in ingresso o in uscita) il nome della proprietá al quale é associato e il formato in cui é inviato. Ci sono tre diversi tipi di formati in cui si possono inviare file da FlightGear:

1. *string*
2. *integer*
3. *float*

Ad ogni diverso formato vengono allocati un numero di byte differenti, ad esempio vengono allocati 4 byte nel caso integer, invece per il float (doppia precisione) ne vengono allocati 8. Da ultimo si puó definire anche se si preferisce creare un protocollo basato su codice ASCII o binario.

Invece per quanto riguarda i dati in ingresso a FlightGear c' é molta meno flessibilitá, infatti l' unico formato che il software accetta é quello *string* basato su codice ASCII.

Tra le varie possibilitá sopra esposte alla fine si é scelto di utilizzare una comunicazione via socket, per permettere una comunicazione piú rapida e agevole tra due diverse console, un protocollo basato su codice binario e un formato float per l'output, un protocollo ASCII e un formato stringa per l'input.

Date queste specifiche, sono state modificate le routine *socket2stream* e *stream2socket* di MBDyn in modo da adattare la comunicazione via socket giá esistente con quella necessaria per comunicare con FlightGear, definta dalle due classi *FGNetFDM* definita in src/Network/net_ctrls.h e *FGNetCtrls* definita in src/Network/net_fdm.h.

Capitolo 7

MBDyn - il modello del Bolkow 105

7.1 Descrizione generale BO-105

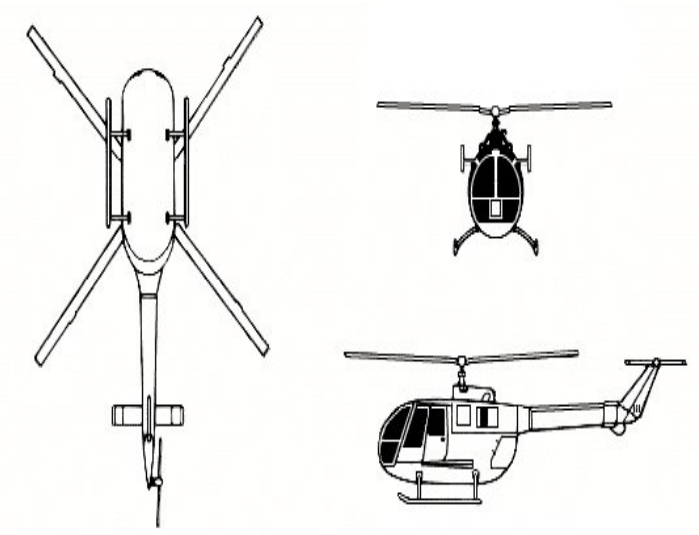


Figura 7.1: Bo105

L' MBB BO 105 é un elicottero leggero multiuso biturbina, progettato dall' azienda tedesca Bölkow con sede a Stoccarda e avviato in produzione dalla Messerschimitt-Bölkow-Blohm alla fine degli anni sessanta. Si tratta di un elicottero relativamente piccolo con un peso a vuoto di circa 1200kg e un peso massimo al decollo di 2300kg. I tipici usi di questo elicottero altamente manovrabile sono i trasporti sulle piattaforme offshore, la poli-

zia e le missioni militari. Il BO105 ha un rotore principale a quattro pale *hingeless* di 4.9 metri di diametro e un rotore di coda a due pale. Le pale costruite in composito del rotore principale fanno in modo che esso abbia una grande eccentricitá e quindi eccellente manovrabilitá lungo il rollio e il beccheggio. La coda si trova sul lato sinistro dell' elicottero e funge da pusher in modo da annullare la coppia torcente generata dal rotore principale. Gli ingressi di comando del pilota sono integrati da due sistemi paralleli servoidraulici.

Il BO 105 era caratterizzato per essere stato il primo al mondo con un rotore rigido "*hingless*" a quattro pale realizzate in materiale composito. Questa configurazione consente una elevata manovrabilitá evidenziata anche da un BO 105CBS (7.2) utilizzato per scopi promozionali dalla Red Bull negli Stati Uniti che nelle esibizioni é in grado di effettuare looping, tonneau, virate Immelmann e altre manovre normalmente considerate prerogativa solo degli aerei acrobatici.

Figura 7.2: Bolkow loop

7.2 Modello multi-corpo del Bo-105

Prima di passare alla descrizione del modello multi-corpo utilizzato per le analisi numeriche, si definisce cosa s' intende per analisi multi-corpo:

L' analisi multi-corpo (multi-body) permette di simulare il comportamento cinematico, dinamico e strutturale di assiemi meccanici composti da parti molteplici in moto reciproco relativo (cinematismi). Dall' analisi multibody é possibile dedurre, ad esempio, le traiettorie e le velocitá di movimento dei membri del cinematismo, cosí come le forze che essi si scambiano in funzione del tempo e (se il meccanismo é schematizzato tramite corpi flessibili) gli stress meccanici, la storia ad essi associata e la loro vita a fatica.

Adesso si passa alla descrizione completa del modello multi-corpo del BO105.

Modellazione strutturale ed aerodinamica La dinamica strutturale dell' elicottero é modellata utilizzando MBDyn, il solutore multi-corpo sviluppato all' interno del Dipartimento di Ingegneria Aerospaziale. L' approccio é abbastanza generico: il solutore puó affrontare direttamente molti aspetti, tra cui anche quello aeroelastico, anche se il modello aerodinamico definito al suo interno si limita alla Teoria dell' elemento di pala e alla Teoria del disco attuatore. In questo caso si sono utilizzati un numero di pannelli aerodinamici pari al numero di travi per ogni pala piú tre ulteriori pannelli associati al piano di coda orizzontale, al piano di coda verticale e alla fusoliera.

Il modello strutturale é costituito dal rotore principale e dalla fusoliera. Il rotore é modellato utilizzando l' approccio multi-corpo: vincoli cinematicamente esatti, garantiti dai moltiplicatori di Lagrange, descrivono il moto relativo tra i corpi rigidi, mentre la dinamica strutturale é affrontata con un approccio agli elementi finiti utilizzando elementi di trave non lineari geometricamente esatti e masse concentrate. Invece per quanto riguarda la fusoliera, questa é stata modellata utilizzando un approccio CMS (Component Mode Synthesis). Essa é connessa al rotore da un *revolute joint* che permette la rotazione relativa tra i due corpi. L' interfaccia tra il modello CMS e il dominio multi-corpo avviene in punti precisi che comprendono il rotore principale, quello di coda, i sedili del pilota e del copilota. Il modello CMS é costituito da forme modali normali (Normal Vibration Mode) selezionate in modo che la loro frequenza di vibrare sia all' interno di un determinato range d' interesse e che le partecipazioni

modali dei sedili dell' equipaggio e dell' attacco del rotore principale siano elevate. In base alle considerazioni precedenti, le forme modali considerate sono le seguenti:

Tabella 7.1: modi

Modo	$frequenza[Hz]$	*Forma modale*
1	5.80	*beccheggio fusoliera*
2	7.68	*laterale fusoliera*
3	11.42	*fusoliera*
4	12.57	*fusoliera*

Gli NVM sono stati calcolati con la massa del rotore concentrata nel nodo di connessione. Infine il rotore di coda é stato modellato come una coppia applicata alla fusoliera, in modo da contrastare la coppia generata dal rotore principale.

7.3 Riduzione del modello

Uno degli obiettivi principali di questo lavoro, che forse non é stato ancora preso in considerazione, é quello di ottenere un modello il piú possibile dettagliato dell' elicottero, ma che al tempo stesso possa permettere un' analisi numerica in *real-time*. Questo perché uno degli scopi fondamentali é quello di avere un modello multi-corpo adatto per simulazioni di volo elicotteristiche.

Cosí com' é il modello non permette assolutamente di avere tempi di calcolo adeguati, di conseguenza si é passati a una riduzione del modello cercando di mantenere il piú possibile le stesse caratteristiche del modello originale.

Per prima cosa si é deciso di eliminare i pannelli aerodinamici della fusoliera e dei piani di coda poiché, in condizioni di volo in hover o comunque con piccoli fattori di avanzamento, risultano poco determinanti nel calcolo della risultante aerodinamica e inoltre rendono la convergenza dell' analisi piú impegnativa, imponendo all' intero sistema un passo temporale molto piccolo.

Dopodiché si é preso in considerazione il rotore principale: esso é composto da quattro pale, ognuna composta da cinque elementi di trave a tre nodi, quattro per modellare la pala e uno per la flexbeam e da quattro pannelli aerodinamici. La scelta finale é stata quella di ridurre gli elementi

di trave a tre, due per la pala e una per la flexbeam, e a due i pannelli aerodinamici. Cosí facendo si perde in accuratezza della soluzione, ma si riesce a garantire sia una buona corrispondenza inerziale che una buona corrispondenza della forma del primo modo di flappeggio della pala.

Per quanto riguarda la fusoliera, inizialmente, si era pensato di ridurre i modi considerati nel modello CMS, ma i margini di guadagno computazionale ridotti hanno fatto sí che la scelta ricadesse sul mantenere l' elemento cosí com' é.

Dopo aver ridotto il sistema come esposto si é riusciti a ottenere rapporti tra il tempo di calcolo e il tempo di simulazione abbastanza vicini all' unitá. Ovviamente bisogna ricordare che i tempi di calcolo vengono molto influenzati anche dalle prestazioni della macchina su cui si lavora. Nel nostro caso, per migliorare ulteriormente le prestazioni si é scelto di mettere in rete due pc: in uno viene portata avanti l' analisi numerica di MBDyn, nell' altro la simulazione grafica di FlightGear.

7.4 Calibrazione dei controlli

I controlli vengono introdotti nel sistema come forze astratte applicate a rigidezze fittizie, dopo aver subito un filtraggio precedente, in modo da simulare un ritardo tra il comando dato dal pilota e il comando applicato alla pala.

Ció che permette una buona controllabilitá del mezzo é una buona calibrazione del sistema di controllo. Per fare una buona calibrazione bisogna identificare un punto di zero e definire una sensibilitá adeguata del controllo in modo da rendere la risposta del rotore né troppo nervosa né troppo rilassata. Come punto di zero si é considerata la condizione di hover, invece per la sensibilitá si é considerato un valore costante adeguato in modo da avere una corrispondenza lineare sul range d' interesse (piccoli spostamenti dalla condizione di equilibrio).

L' hovering é quella condizione di volo per cui le forze risultanti lungo z si annullano. Per poter mettere in evidenza quale valore di passo collettivo permetta l' annullamento della forza peso si é utilizzato l' espediente di vincolare a terra il modello e di monitorare la reazione vincolare lungo z. Dopo qualche iterazione si é ottenuto il punto di zero cercato per il passo collettivo come si puó vedere dalla figura. Questa procedura é stata applicata per ottenere i punti di zero per ogni controllo, ma siccome la dinamica dell' elicottero é fortemente accoppiata tra il moto longitudinale e latero-direzionale, trovare una combinazione di comandi tale da annullare

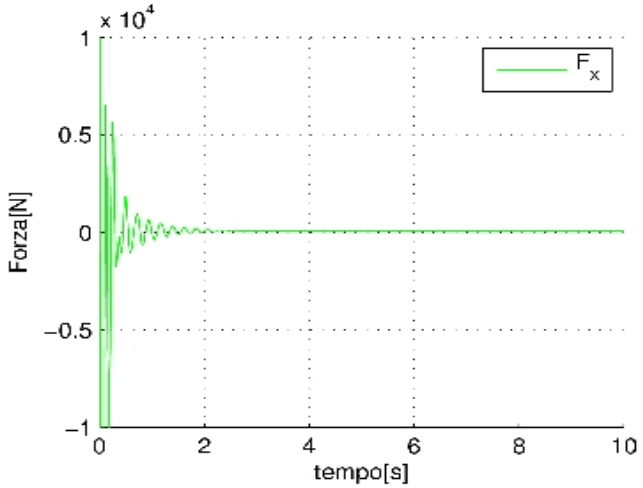


Figura 7.3: forza z hover

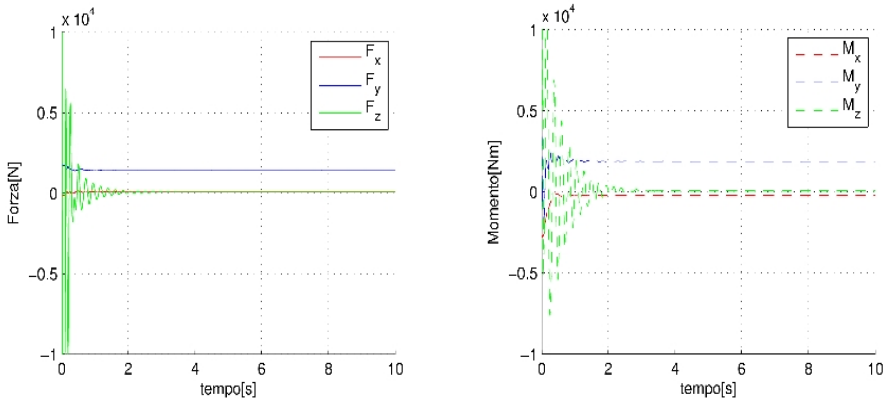
tutte e 3 le risultanti e i 3 momenti non é stato possibile. Di conseguenza si é cercato di mantenere nulle la risultante lungo z delle forze e dei momenti (calibrando la pedaliera) e per le altre si é cercato il migliore compromesso come si puó vedere nella figura 7.4.

I range dei controlli sono stati definiti a posteriori, conoscendo gli angoli massimi e minimi di assetto delle pale per il collettivo, ciclico longitudinale e laterale. Per i nostri scopi comunque non é indispensabile definire

Tabella 7.2: range

	massimo[°]	*minimo*[°]
collettivo	15.8	0.2
ciclico laterale	10.5	−4.7
ciclico longitudinale	5.65	−4.23

con precisione tutto il range poiché si considereranno piccoli spostamenti nell' intorno della condizione di hover.



(a) *forze hover.* **(b)** *momenti hover.*

Figura 7.4: forze momenti hover

7.5 Validazione del modello ridotto - Risposte dinamiche della pala

La figura 7.5 mostra il famplot, un diagramma basato sul database aeroelastico del BO105. Esso mette in relazione i modi aeroelastici in atmosfera standard e in condizioni di vuoto, cioé senza carichi di aria, in funzione della velocitá di rotazione del rotore.

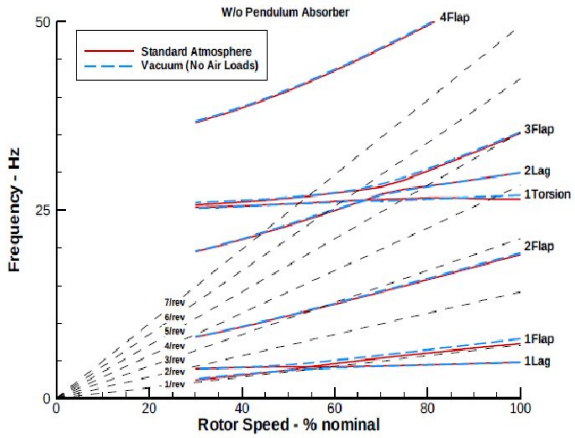


Figura 7.5: fam plot

Per la validazione dinamica del modello di rotore ridotto, si é fatto riferimento al modello completo, la cui dinamica é rappresentata dal famplot. Un elemento fondamentale per una buona corrispondenza di comportamento tra i due modelli numerici é la dinamica delle pale, soprattutto, quella a piú bassa frequenza.

I modi della pala che vengono presi in considerazione per il confronto, poiché si é notato essere quelli che partecipano maggiormente all' instabilitá aeroservoelastica sono:

1. primo modo di ritardo,
2. primo modo di flappeggio,
3. secondo modo di flappeggio.

Inoltre per meglio mettere in luce le dinamiche che caratterizzano la pala, si considera un' eccitazione a doppio scalino dovuta a una forza, in direzione trasversale alla pala sia in apertura che in spessore, posizionata all' estremitá della pala e l' assenza dell' aria per rendere minimi gli smorzamenti aerodinamici.

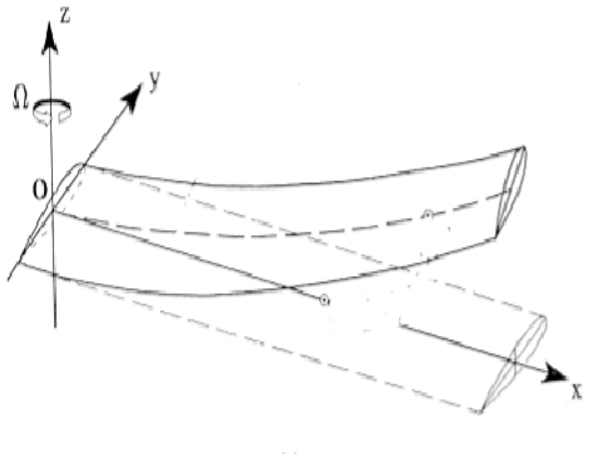
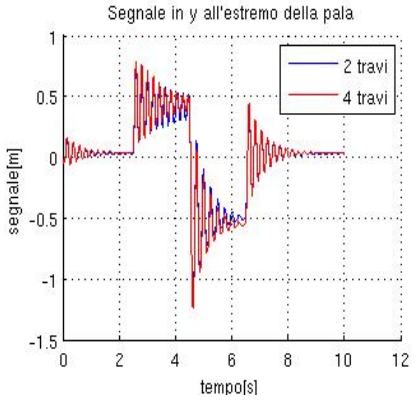
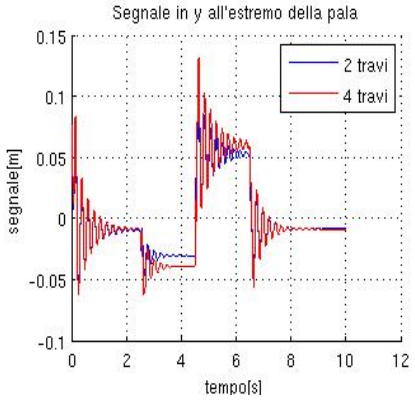


Figura 7.6: sistema di riferimento pala

Differenze tra i due modelli nell' aria Di seguito si riportano i risultati ottenuti dai due modelli, con quattro travi e con 2 travi per pala in atmosfera standard. Per il confronto si sono considerati gli spostamenti in z 7.8(flappeggio), in y 7.7(ritardo) e il contenuto in frequenza in entrambe le direzioni 7.9 7.10(modulo e fase).

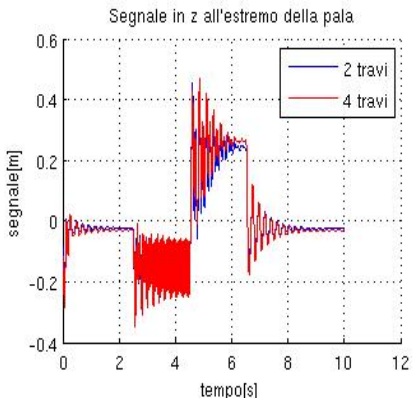


(a) *segnale in y - eccitazione in y - aria.*

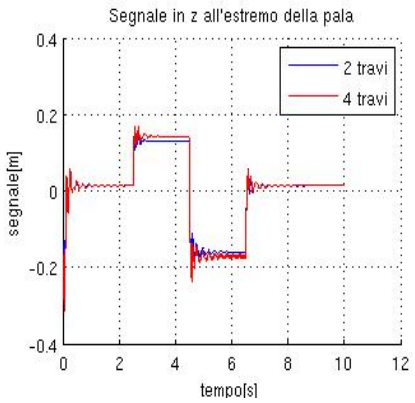


(b) *segnale in y - eccitazione in z - aria.*

Figura 7.7: segnale in y con aria



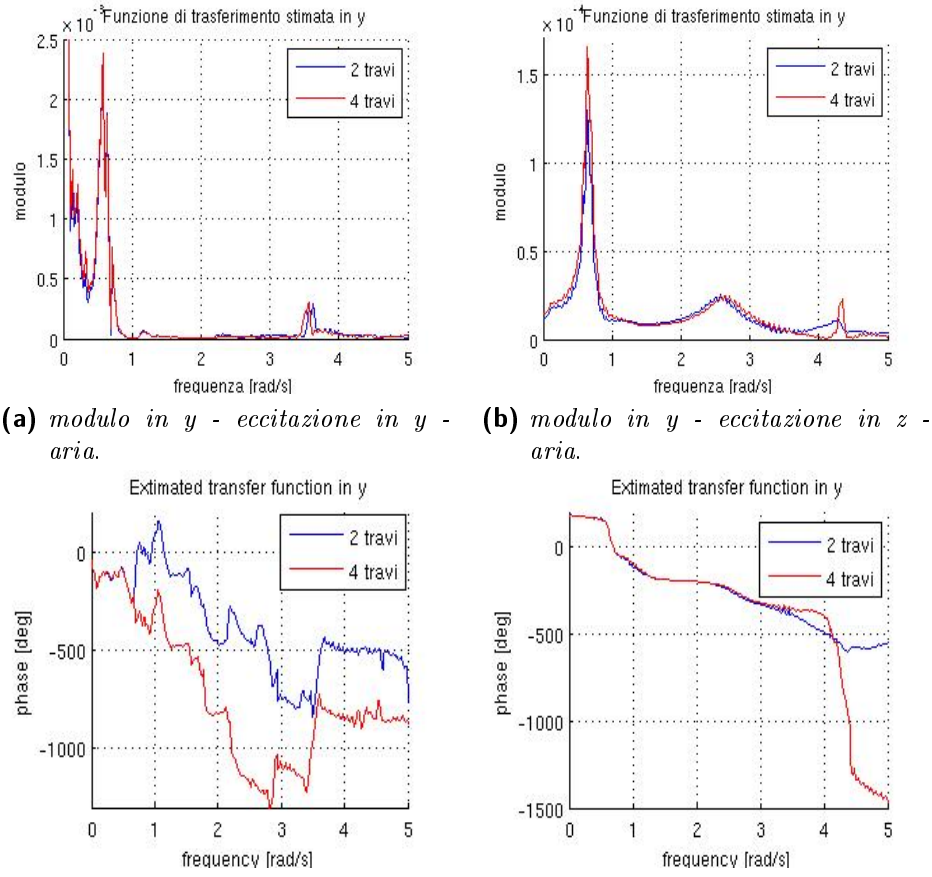
(a) *segnale in z - eccitazione in y - aria.*



(b) *segnale in z - eccitazione in z - aria.*

Figura 7.8: segnale in z con aria

Come si puó vedere all' estremitá la pala si comporta allo stesso modo in entrambi i casi manifestando oscillazioni rapidamente smorzate. Inoltre si puó riscontrare una buona corrispondenza dinamica nelle figure 7.9 7.10. Nella prima figura si puó vedere come lungo y si abbia un forte contributo del primo modo di ritardo a bassa frequenza e del primo modo torsionale a piú alta frequenza, sia con eccitazione in direzione y che in direzione z. Nella seconda figura si vede che in z si ha un forte contributo del primo



(a) *modulo in y - eccitazione in y - aria.*

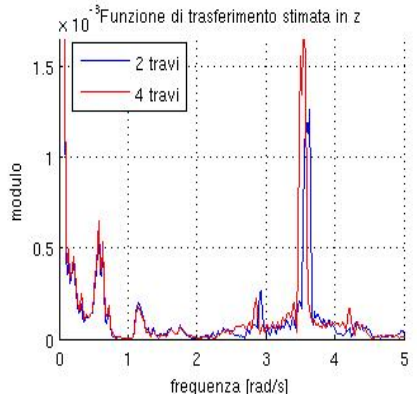
(b) *modulo in y - eccitazione in z - aria.*

(c) *fase in y - eccitazione in y - aria.*

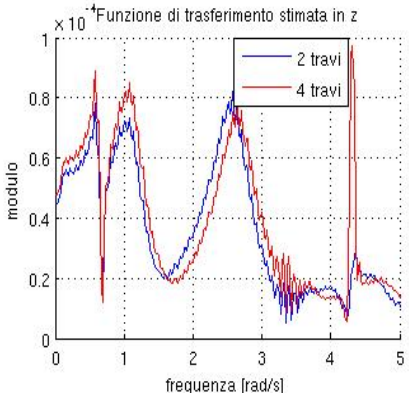
(d) *fase in y - eccitazione in z - aria.*

Figura 7.9: modulo e fase in y con aria

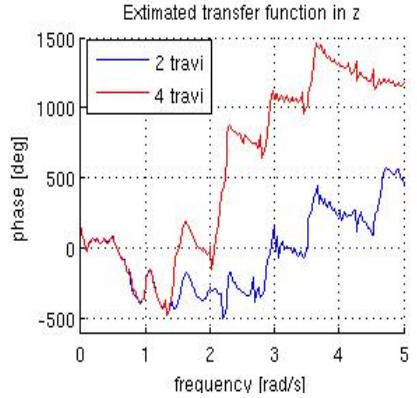
modo di ritardo, dei due modi di flappeggio, del primo modo torsionale e per quanto riguarda la pala a quattro travi si ha anche un contributo da parte del secondo modo di ritardo intorno a 30Hz.



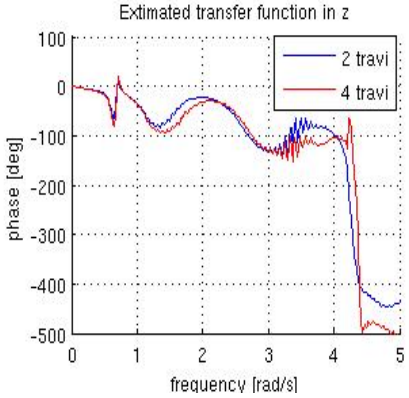
(a) *modulo in z - eccitazione in y - aria.*



(b) *modulo in z - eccitazione in z - aria.*



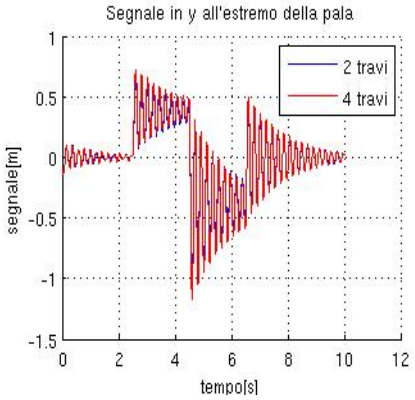
(c) *fase in z - eccitazione in y - aria.*



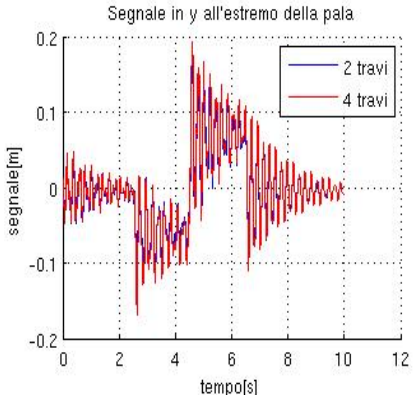
(d) *fase in z - eccitazione in z - aria.*

Figura 7.10: modulo e fase in z con aria

Differenze tra i due modelli nel vuoto Di seguito si riportano i risultati ottenuti dai due modelli, con quattro travi per pala e con 2 travi per pala nel vuoto.

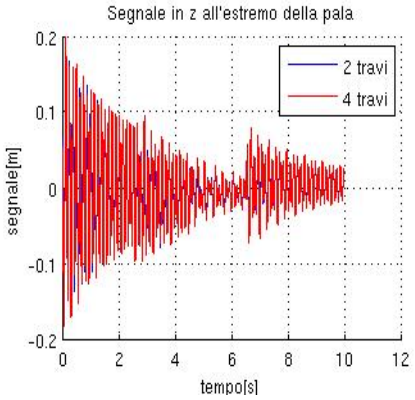


(a) *segnale in y - eccitazione in y - vuoto.*

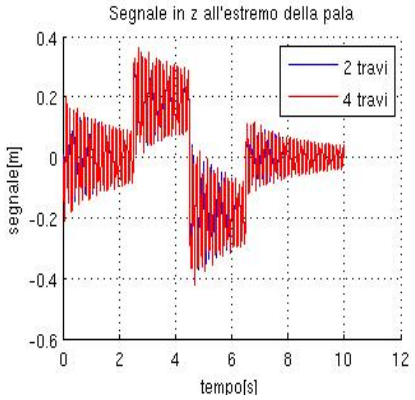


(b) *segnale in y - eccitazione in z - vuoto.*

Figura 7.11: segnale in y nel vuoto



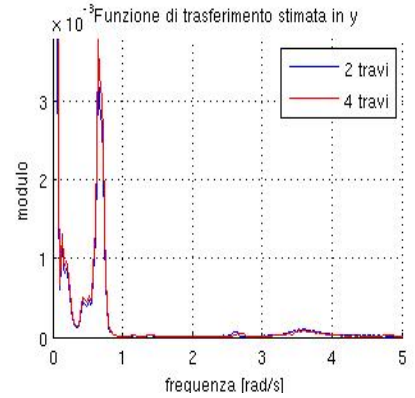
(a) *segnale in z - eccitazione in y - vuoto.*



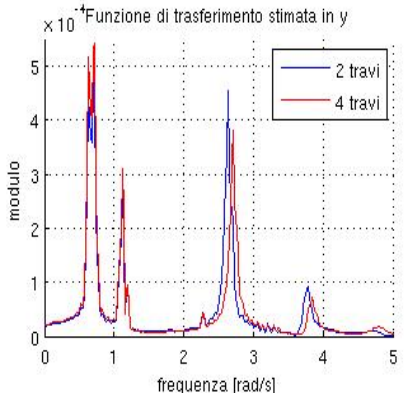
(b) *segnale in z - eccitazione in z - vuoto.*

Figura 7.12: segnale in z nel vuoto

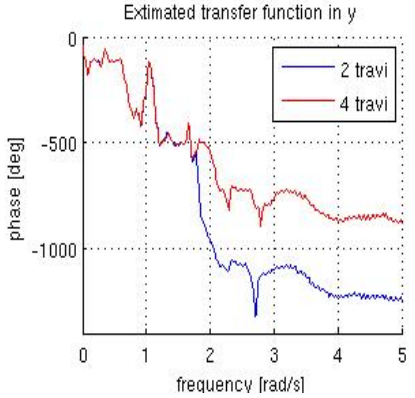
Come si puó vedere all' estremitá la pala si comporta allo stesso modo in entrambi i casi manifestando oscillazioni ben poco smorzate, poiché la causa principale dello smorzamento (l' aerodinamica) é stata spenta.



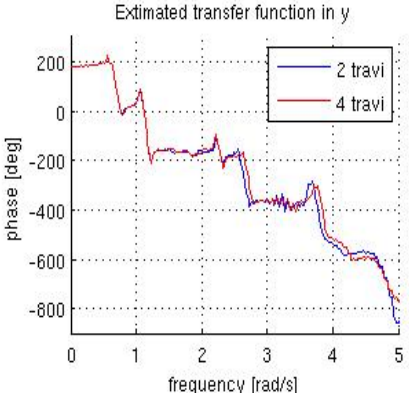
(a) *modulo in y - eccitazione in y - vuoto.*



(b) *modulo in y - eccitazione in z - vuoto.*



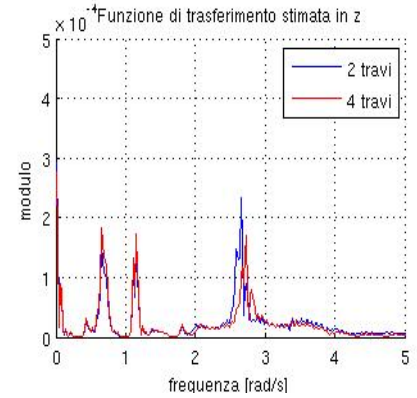
(c) *fase in y - eccitazione in y - vuoto.*



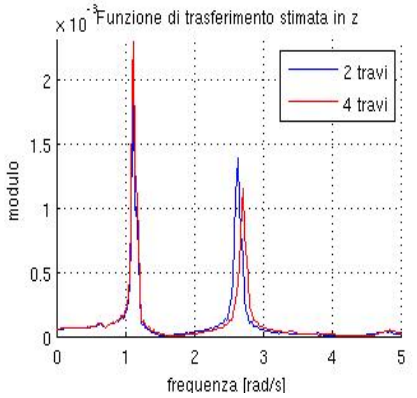
(d) *fase in y - eccitazione in z - vuoto.*

Figura 7.13: modulo in y nel vuoto

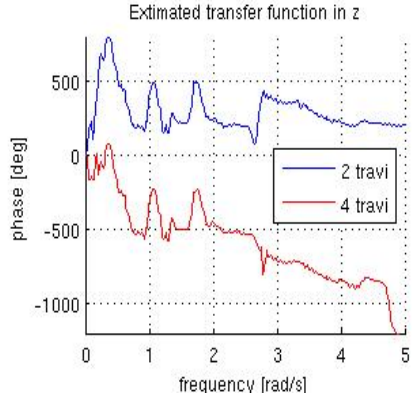
In questo caso non avendo piú uno smorzamento dovuto all' aerodinamica si hanno dei contributi alla dinamica molto definiti dei primi quattro modi della pala.



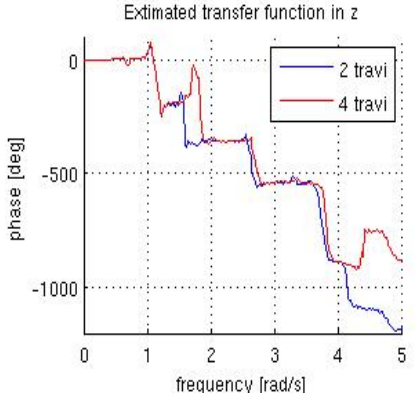
(a) *modulo in z - eccitazione in y - vuoto.*



(b) *modulo in z - eccitazione in z - vuoto.*



(c) *fase in z - eccitazione in y - vuoto.*



(d) *fase in z - eccitazione in z - vuoto.*

Figura 7.14: modulo in z nel vuoto

7.6 Biomeccanica del pilota

Il pilota puó interagire con la dinamica dell' elicottero in diversi modi. Quando il pilota intenzionalmente dá un comando per eseguire qualche compito, ma l' input é alterato da percezioni errate, l' intervento é volontario. Questo tipo di intervento é comunque a banda limitata dalle capacitá degli arti umani.

Il pilota puó anche comandare i controlli in modo non intenzionale, come conseguenza delle eccitazioni provenienti dall' abitacolo. Per esempio, le vibrazioni del sedile o di altre parti del cockpit, possono indurre il movimento non intenzionale dei controlli, precedentemente filtrato dalla dinamica passiva degli arti del pilota. Questo fenomeno conosciuto come "vertical bounce", o "collective bounce" é caratteristico degli elicotteri. Esso consiste in oscillazioni verticali causate da pulsazioni di spinta indot-

te da oscillazioni involontariamente introdotte nel controllo collettivo da parte del pilota.

La biomeccanica passiva del pilota é stata inizialmente presa in considerazione per controlli specifici, ad esempio, il collettivo quando subisce vertical bounce, utilizzando funzioni di trasferimento disponibili dalla letteratura. Mayo ha individuato la funzione di trasferimento tra l' accelerazione verticale del sedile dell' elicottero e l' accelerazione tangenziale del bastone del collettivo, utilizzando un mock-up di cockpit strumentato, soggetto ad un' eccitazione armonica ([11]). Le due funzioni di trasferimento sono presentate qui di seguito,

$$H_{meso} = \frac{4.02s + 555.4}{s^2 + 13.31s + 555.4} \tag{7.1}$$

$$H_{ecto} = \frac{5.19s + 452.3}{s^2 + 13.70s + 452.3} \tag{7.2}$$

Queste sono illustrate nella figura 7.15 e sono rispettivamente associate al pilota "mesomorfo" (piccola taglia) e al pilota "ectomorfo" (grande taglia). Entrambe le funzioni di trasferimento presentano una coppia di poli coniugati a circa 3.5Hz, con uno smorzamento del 30% circa.

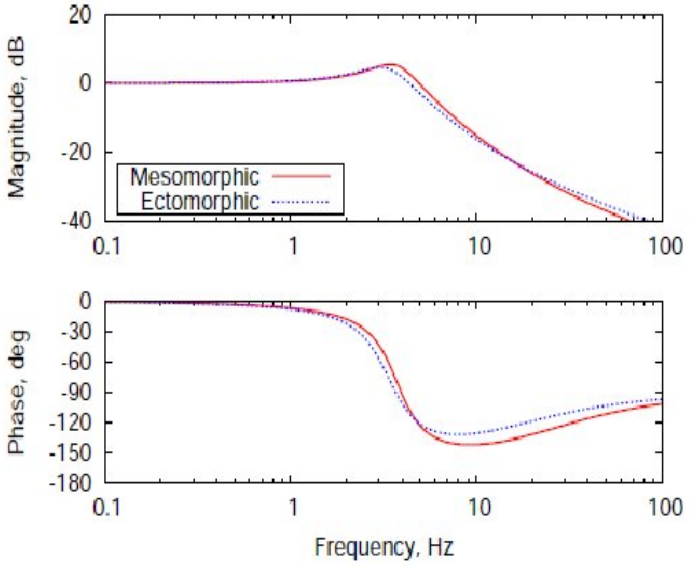


Figura 7.15: Mayo

Modello di pilota In questo caso, il pilota é stato modellato con una rappresentazione agli stati di un sistema Multi-Input Multi-Output (MIMO). Questo riceve in ingresso l' accelerazione assoluta del sedile e in output il movimento dei controlli, come la barra del collettivo e del ciclico. In questo lavoro si é considerato solo un comando di collettivo. La funzione di trasferimento proposta da Mayo prende in considerazione le accelerazioni assolute. A noi interessano le quantitá relative, quindi bisogna trasformare l' ingresso in rotazioni relative della barra del collettivo e l' uscita in accelerazioni relative del sedile. Per quanto riguarda le accelerazioni relative si possono ottenere semplicemente da questa relazione

$$H_{rel}(s) = H_{abs} - 1 \tag{7.3}$$

Invece per le rotazioni relative della barra bisogna dividere l' accelerazione relativa per la lunghezza della barra del collettivo, L, e integrare due volte,

$$\Delta\theta(s) = \frac{1}{s^2}\frac{1}{L}(H_{abs} - 1)a(s) \tag{7.4}$$

Quando viene fatta questa trasformazione alle equazioni 7.6 le funzioni di trasferimento risultano come in figura 7.16. La presenza dei due integratori

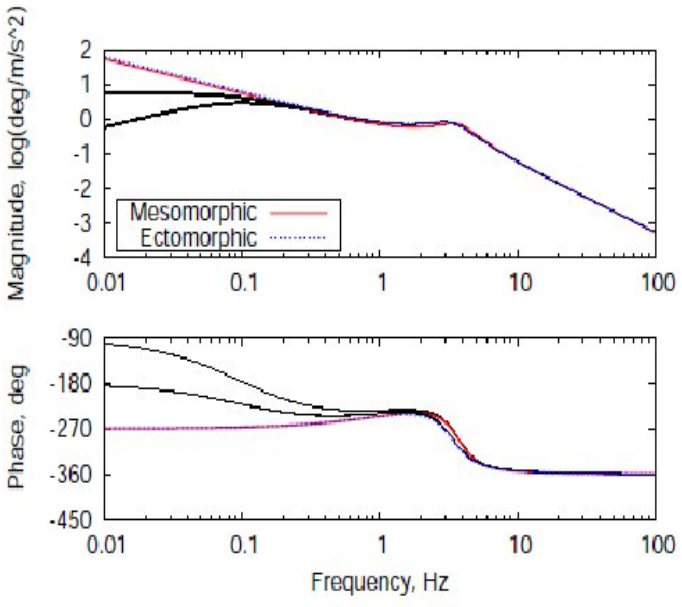


Figura 7.16: Mayo2

nell' equazione precedente produce un comportamento di deriva quando

s$\rightarrow$ 0. Questo non é un comportamento fisico, in quanto ció implicherebbe, per esempio, che il comando di collettivo si riduca indefinitivamente a causa della gravitá. Ció che questa funzione di trasferimento sperimentale non prende in considerazione é che la volontá del pilota compensa ogni cambiamento a bassa frequenza della posizione del collettivo non appena ne sia adeguatamente cosciente. In modo da tenere in considerazione questo fatto, le funzioni sono filtrate da un filtro passa-alto, semplicemente trasformando i poli $\frac{1}{s^2}$ in poli reali vicini allo zero,cioé

$$\Delta\theta(s) = \frac{1}{(s-\alpha_1)(s-\alpha_2)}\frac{1}{L}(H_{abs}-1)a(s) \tag{7.5}$$

La linea tratteggiata in figura 7.16 illustra le correzioni precedenti. Queste corrispondono a trasformare o uno o tutti e due gli integratori in uno o due poli a 0.1Hz. Gli output sono trasformati nei comandi del piatto oscillante dopo un'ulteriore filtraggio, in modo da rappresentare la dinamica del sistema di attuazione. La dinamica degli attuatori che comanda il moto del piatto oscillante é tipicamente espressa utilizzando funzioni di trasferimento del primo o del secondo ordine. Nel nostro caso si é considerata una funzione di trasferimento del prim' ordine con costante di tempo $\tau = 0.4$

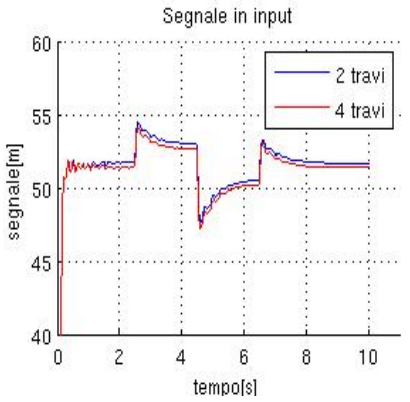
$$y = \frac{1}{1+\tau s}u \tag{7.6}$$

dove u é l' elongazione dell' attuatore comandata dal pilota, mentre y é l' elongazione risultante.

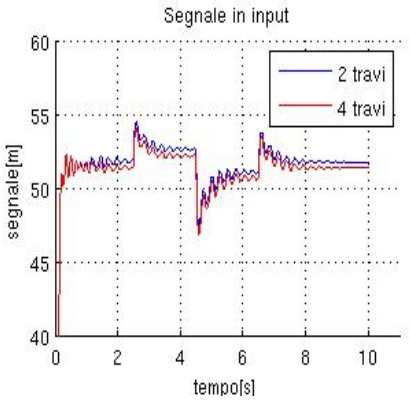
7.7 Accoppiamento pilota-elicottero

I modelli di pilota sono accoppiati col sistema elicottero dal solutore multi-corpo stesso. Esistono gli elementi "general-purpose" (GENELs) i quali permettono di modellare la dinamica del sistema in modo arbitrario. Viene introdotto un "guadagno fittizio" fra la funzione di trasferimento del pilota e quella degli attuatori del piatto oscillante. Occorre sottolineare che questo parametro non é in alcun modo collegato ad alcuna "aggressivitá" nel comportamento del pilota. É piuttosto legato alla determinazione dei parametri di progetto che possono mettere in pericolo la stabilitá quando il pilota é nel loop. Di conseguenza questo guadagno puó comprendere molti aspetti del problema: un cambiamento di ampiezza modale del movimento del sedile, una modifica del tasso di rotazione tra il collettivo e il passo della pala, e cosí via.

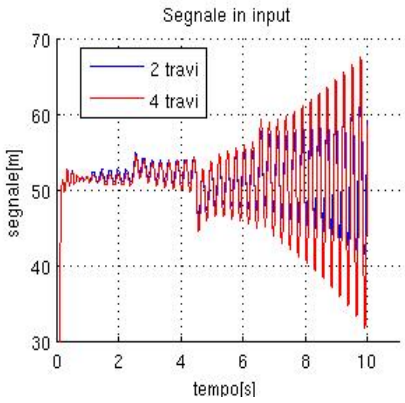
Risultati dei due modelli Di seguito si ripropongono i risultati ottenuti dallo studio dell' interazione pilota-elicottero. Si riportano i grafici relativi al comando di collettivo, allo spostamento dell' estremo della pala in z e in y per diversi valori di guadagno fittizio. Si deve notare che per il caso a quattro travi si hanno oscillazioni con ampiezza maggiore, come se il modello ridotto introducesse dello smorzamento strutturale.



(a) *input - gain=0.5.*

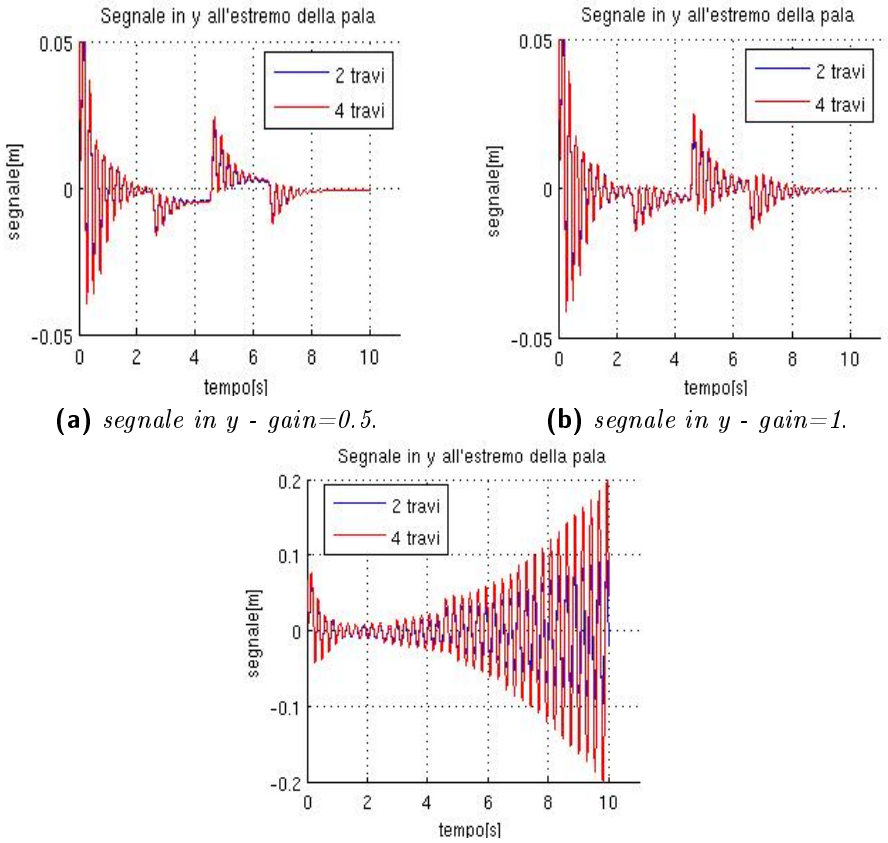


(b) *input - gain=1.*



(c) *input - gain=1.5.*

Figura 7.17: input pilota



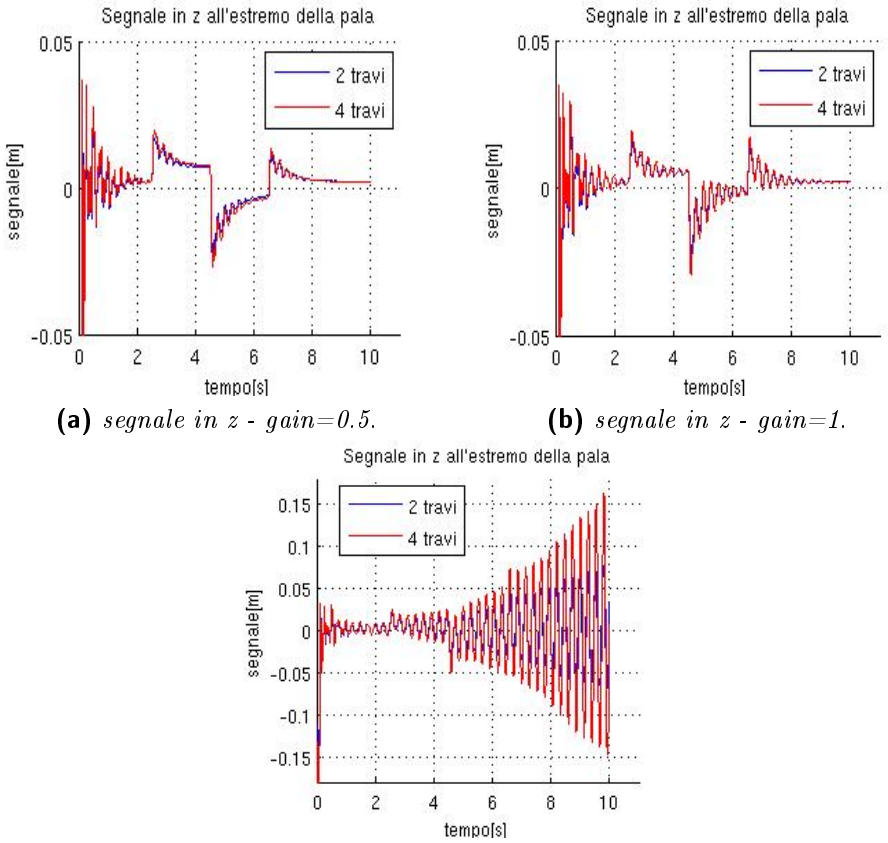
(a) *segnale in y - gain=0.5.*

(b) *segnale in y - gain=1.*

(c) *segnale in y - gain=1.5.*

Figura 7.18: segnale in y pilota

Come si puó notare nelle figure 7.19 7.18 7.17, in entrambi i casi di modellazione, le oscillazioni aumentano all' aumentare del guadagno fino ad un valore di divergenza. L' oscillazione dominante é intorno ai 4Hz e nasce proprio dal fenomeno di "vertical bounce". Questa é la componente responsabile della migliore o peggiore controllabilitá del velivolo.



(a) *segnale in z - gain=0.5.*

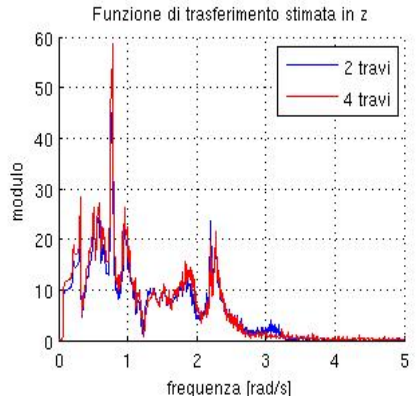
(b) *segnale in z - gain=1.*

(c) *segnale in z - gain=1.5.*

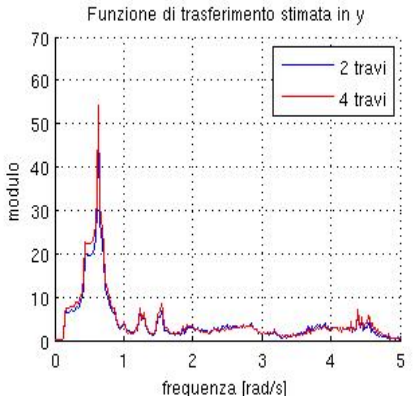
Figura 7.19: segnale in z pilota

Infine si riporta il grafico del contenuto in frequenza dello spostamento in z e in y dell' estremo della pala 7.20, il quale sottolinea a sua volta un forte contributo alla dinamica del sistema intorno ai 4Hz. I valori di guadagno che portano alla condizione di divergenza nei due sistemi sono leggermente diversi a causa di un diverso smorzamento strutturale:

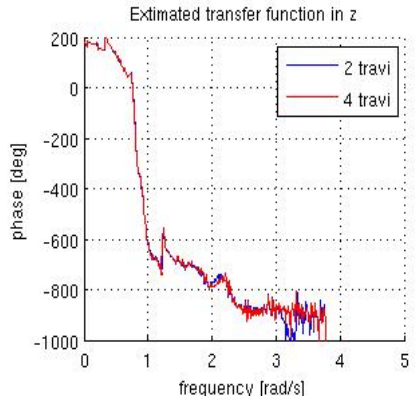
- 1.38 per la modellazione a quattro travi,
- 1.41 per la modellazione a due travi.



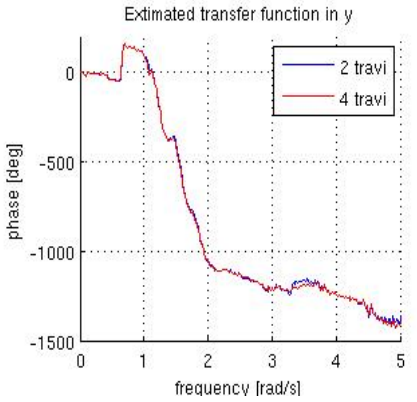
(a) *modulo in z - gain=1.38 - gain=1.41.*



(b) *modulo in y - gain=1.38 - gain=1.41.*



(c) *fase in z - gain=1.38 - gain=1.41.*



(d) *fase in y - gain=1.38 - gain=1.41.*

Figura 7.20: modulo e fase pilota

Capitolo 8

Blender - l' animazione 3D del modello

8.1 In generale

Blender é un programma open source di modellazione, rigging, animazione, compositing e rendering di immagini tridimensionali. Dispone inoltre di funzionalitá per mappature UV, simulazioni di fluidi, di rivestimenti, di particelle, altre simulazioni non lineari e creazione di applicazioni/giochi 3D. É disponibile per vari sistemi operativi: Microsoft Windows, Mac OS X, Linux, IRIX, Solaris, NetBSD, FreeBSD, OpenBSD assieme a porting non ufficiali per BeOS, SkyOS, AmigaOS, MorphOS e Pocket PC. Blender é dotato di un robusto insieme di funzionalitá paragonabili, per caratteristiche e complessitá, ad altri noti programmi per la modellazione 3D come Softimage XSI, Cinema 4D, 3D Studio Max, LightWave 3D e Maya. Tra le funzionalitá di Blender sono incluse l' utilizzo di raytracing e di script (in Python). Tra le sue potenzialitá, si possono ricordare:

- Supporto per una grande varietá di primitive geometriche, incluse le mesh poligonali, le curve di Bezier, le NURBS, le metaball e i font vettoriali,
- Conversione da e verso numerosi formati per applicazione 3D, come Wings 3D, 3D Studio Max, LightWave 3D e altri,
- Strumenti per gestire le animazioni, come la cinematica inversa, le armature (scheletri) e la deformazione lattice, la gestione dei keyframe, le animazioni non lineari, i vincoli, il calcolo pesato dei vertici e la capacitá delle mesh di gestione delle particelle,
- Gestione dell' editing video non lineare,

- Caratteristiche interattive, come la collisione degli ostacoli, il motore dinamico e la programmazione della logica, permettendo la creazione di programmi stand-alone o applicazioni real time come la visione di elementi architettonici o la creazione di videogiochi,
- Motore di rendering interno versatile ed integrazione nativa col motore esterno YafaRay (un raytracer open source),
- Scripting in python per automatizzare e/o controllare numerosi aspetti del programma e della scena.

8.2 Interfaccia Blender-MBDyn

L' aspetto su cui MBDyn é mancante é il *pre/post-processing*, questo perché per sua natura esso é nato solamente come simulatore multi-corpo. In realtá é giá esistente un' interfaccia grafica 2D con EasyAnime, strumento di animazione gratuito sviluppato inizialmente da Olivier Verlinden, presso la Faculté Polytechnique de Mons, per servire come strumento leggero per la visualizzazione di analisi multi-corpo, ma effettivamente la qualitá grafica messa a disposizione é piuttosto scarsa.

Ultimamente é stata messa a punto da Doug Baldwing un' interfaccia con Blender, strumento molto piú potente di rendering e animazione 3D, come giá evidenziato precedentemente. Questa interfaccia permette di utilizzare Blender sia come pre-processing che post-processing in modo da completare MBDyn. Quindi sono due gli obiettivi raggiunti da questa interfaccia: riuscire a scrivere un file di input per MBDyn e gestire i file di output (solamente i file .mov) direttamente da Blender, cosí da utilizzare il solutore solo per le analisi multi-corpo. In questo modo si riesce a rendere accessibile a chiunque (o quasi) la scrittura di un file di input.

L' interfaccia con Blender é costituita da due file scritti in Phyton, *mbdyn.py* e *mbdyn_gui.py*. Nel primo si definiscono quasi tutte le *card* d' ingresso possibili per MBDyn, nel secondo si definisce l' intefaccia grafica utente(GUI) per la scrittura del file d' input e per la gestione dei file di output. L' unica cosa da prendere in considerazione é che, per quanto riguarda la gestione dell' output, bisogna definire le orientazioni attraverso la matrice di orientazione invece che con gli angoli di Cardano (default per MBDyn).

8.3 Realizzazione dell' animazione 3D

Per l' animazione 3D é necessario:

1. definire le texture per creare la grafica dell' oggetto,
2. definire il movimento degli oggetti,
3. associare il movimento a ogni oggetto e creare l' *armatura*,
4. definire il rendering e creare il video.

Il primo punto é necessario per avere un' immagine di partenza su cui imporre il movimento: essa é stata ricavata dai modelli presenti nella "biblioteca" di velivoli di FlightGear. Nella figura 8.1 si riporta una immagine del modello di partenza.

Figura 8.1: Bo105 blender

Il secondo punto é direttamente l' output di MBDyn. In pratica si é deciso di salvare su file solo la dinamica dei nodi strutturali che a noi interessano, quindi i nodi delle pale. Cosí facendo si mantengono contenuti sia i tempi di scrittura, da parte di MBDyn, che i tempi di lettura di Blender (non del tutto trascurabili).

Infine si deve associare il movimento dei nodi alle texture che rappresentano le pale. Questa é la parte piú interessante perché é quella che permette di avere una deformazione elastica della pala continua in apertura e non costante a tratti. Blender usa le *armature* per l' animazione degli oggetti. L' armatura, una volta associata alla mesh del nostro oggetto, é proprio come uno scheletro che ci permette di definire una serie di

Figura 8.2: armatura

pose sul nostro oggetto, lungo la linea temporale della nostra animazione. L' armatura é composta da un numero arbitrario di ossa. Le dimensioni, posizioni e orientamenti di tutte le ossa nell' armatura sono una scelta che spetta all' utente, ovviamente scelte diverse porteranno a comportamenti diversi dell' oggetto.

Nel nostro caso si é scelto di creare un' armatura che avesse un numero di ossa pari al numero di nodi strutturali lungo la pala, e di associare ad ogni osso la posizione e l' orientamento del nodo corrispondendente.

Il rendering é il processo finale del post-processing ed é la fase in cui finalmente viene creata l' immagine corrispondente alla propria scena 3D. Le opzioni che mette a disposizione Blender per l' ottimizzazione delle immagini sono le seguenti:

Shadow serve per calcolare e creare le ombre. Le ombre sono emesse dalle lampade adatte a proiettarle, e sono ricevute dagli oggetti che hanno materiali atti alla ricezione,

Environment map cambia il colore dello sfondo e degli oggetti, a seconda della luce emessa dal' "environment map" del mondo. Serve per avere un rendering piú realistico,

Raytracing é un metodo di calcolo piú preciso del colore di una superficie, in particolare per superfici riflettenti,

SSS o dispersione interna é un meccanismo di trasporto della luce in cui essa penetra la superficie di un oggetto traslucido, viene diffusa dall'

interazione con il materiale ed esce da un punto differente della superficie. La luce generalmente penetra la superficie e viene riflessa un certo numero di volte con angoli differenti all' interno del materiale, prima di passare attraverso ad esso con un angolo diverso da quello che avrebbe avuto se fosse stata direttamente riflessa dalla superficie. La dispersione interna é importante per la grafica 3D poiché é necessaria per un rendering realistico dei materiali,

Motion Blur calculation(MBLUR) questa fa in modo che Blender generi un numero di frame "intermedi" pari al numero di sovracampionamento scelto (5, 8, 11 o 16) e che li metta insieme tutti in un singolo frame. Il *Blur factor* definisce il tempo di otturazione,

Oversampling(OSA) questa fa in modo che ogni immagine accumulata abbia anti-aliasing, una tecnica per ridurre l' effetto aliasing (scalettatura o gradinatura) quando un segnale a bassa risoluzione viene mostrato ad alta risoluzione. L' opzione OSA in genere non é necessaria finché é attivo il MBLUR, ma per avere una lisciatura realistica delle superfici bisogna attivarla,

Radiosity quando la luce colpisce un oggetto, parte dello spettro dei colori della luce viene assorbito e altri colori vengono riflessi ai nostri occhi. La radiositá é quando la luce colpisce anche un oggetto vicino ad esso, dando colore anche ad esso. Alle volte questo effetto viene chiamato "travaso di colore", perché il colore di un oggetto travasa su quello vicino. La radiositá da un maggior fotorealismo

Infine gli elementi necessari per la creazione del video sono una *camera* e un numero sufficiente di *lamp*, di cui sono presenti diversi tipi in base a quale luce si preferisce per la scena. Nel nostro caso il risultato finale é un video con le seguenti caratteristiche:

- formato AVI Jpeg: salva un AVI come una serie di immagini Jpeg con perdita d' informazioni. Genera file piú piccoli ma non tanto quanto si potrebbe fare con un algoritmo di compressione migliore,
- risoluzione 640x480,
- a colori (RGB),
- a 100 frame per secondo.

Capitolo 9

Conclusioni e sviluppi futuri

In accordo con il *vertical bounce*, il modo del pilota (circa a 3.5Hz quando disaccoppiato) si accoppia sia al primo modo della fusoliera (poco meno di 6Hz), sia al primo modo di flappeggio delle pale (circa a 7Hz) generando oscillazioni difficilmente governabili attorno ai 4Hz. Questa eccitazione arriva dall' accelerazione verticale del sedile del pilota, composta da un moto rigido e una deformazione elastica della fusoliera. Come dimostrato nel corpo della tesi le oscillazioni che si innescano sono stabili, ma con margini di guadagno molto modesti, dimostrando ancora una volta come questo tipo di problematica sia un parametro importante da considerare nella progettazione. Per questo motivo che con questa tesi si é cercato di mettere le basi di un simulatore di volo in grado di simulare fenomeni di tipo aeroelastico.

In futuro si spera potranno essere considerati fenomeni aeroelastici giá nel loop di preprogettazione, poiché grazie a questo tipo di simulazioni si possono ottenere rapidamente interessanti stime di comportamento del velivolo ad un costo, sia computazionale che economico, relativamente basso.

Un ulteriore aspetto da considerare é l' interfaccia con Blender. Essa é di estrema importanza per creare semplici modelli dinamici e per vedere rapidamente il loro comportamento. Sviluppi futuri potrebbero permettere di includere nel preprocessing ulteriori card definite nei file sorgenti di MBDyn, in modo da poter generare elementi, vincoli, forze, etc... differenti e nel postprocessing una creazione automatica del' armatura associando le posizioni e le rotazioni di ogni nodo strutturale direttamente a ogni osso dell' armatura stessa, rendendo piú rapida la creazione di un' animazione soddisfacente.

Appendice A

Disegni del mockup

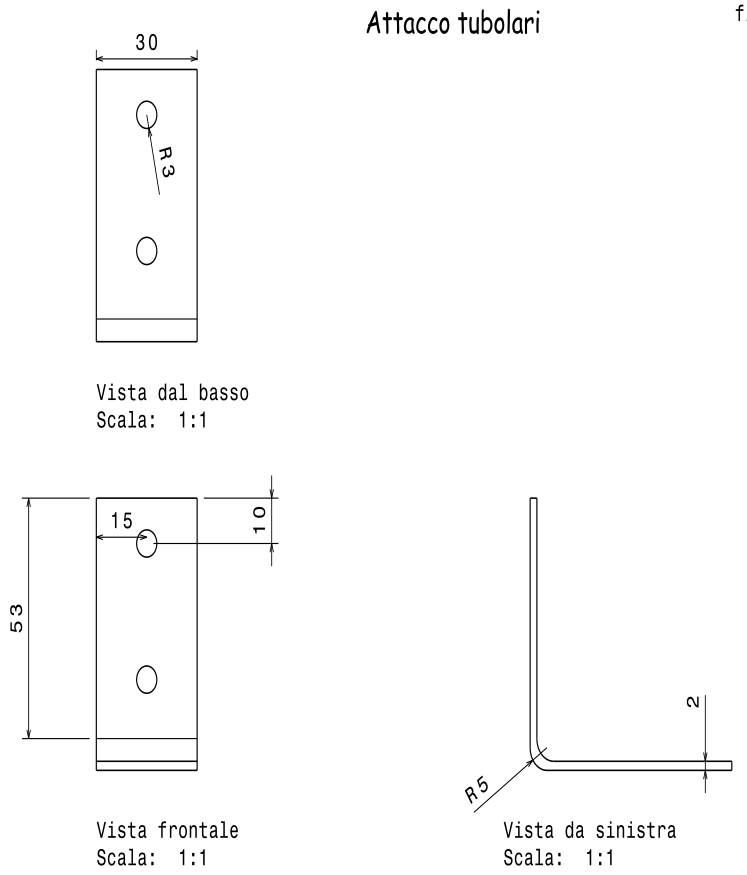


Figura A.1: Attacco tubolari

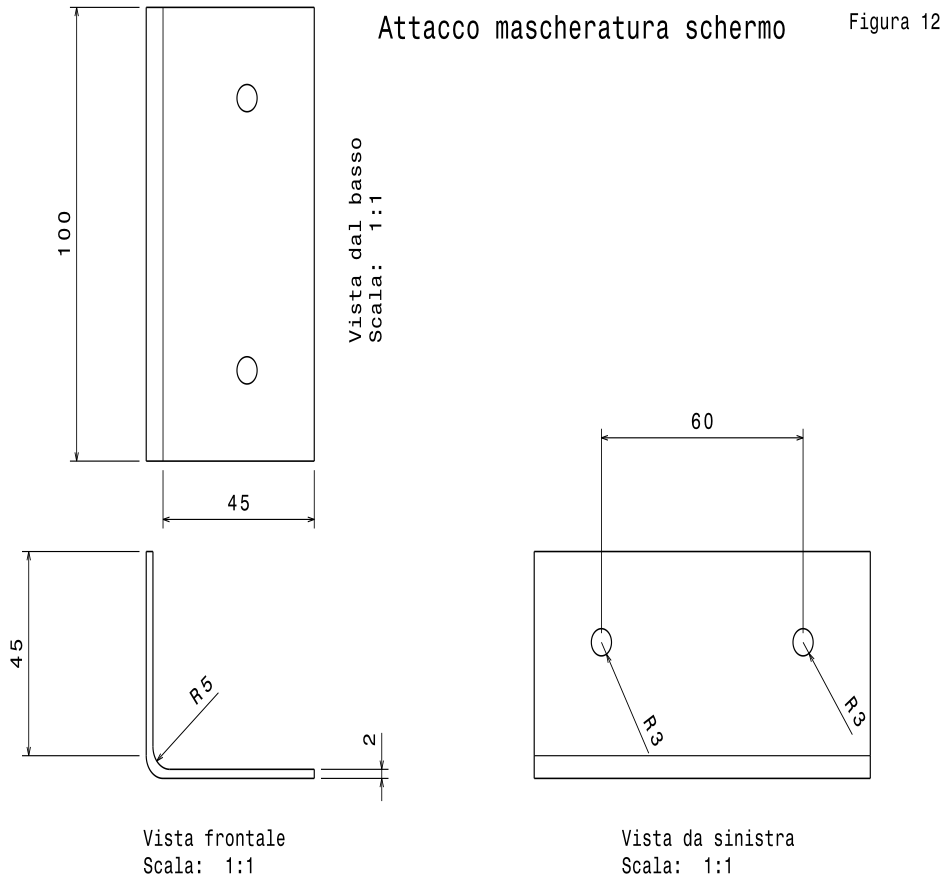


Figura A.2: Attacco mascheratura schermo

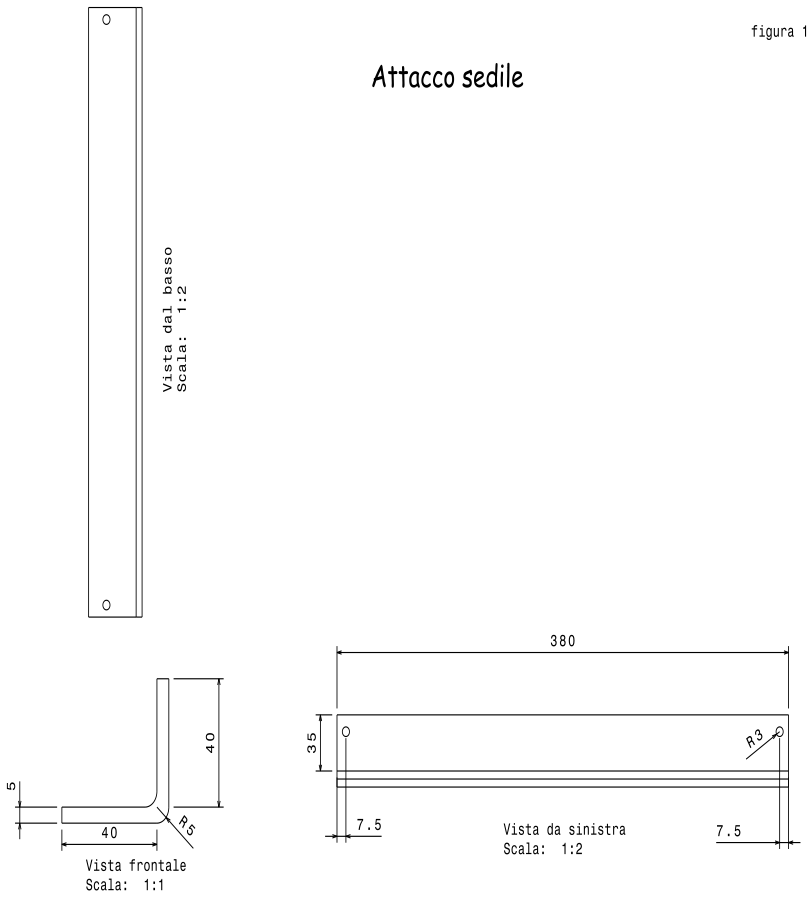


Figura A.3: Attacco sedile

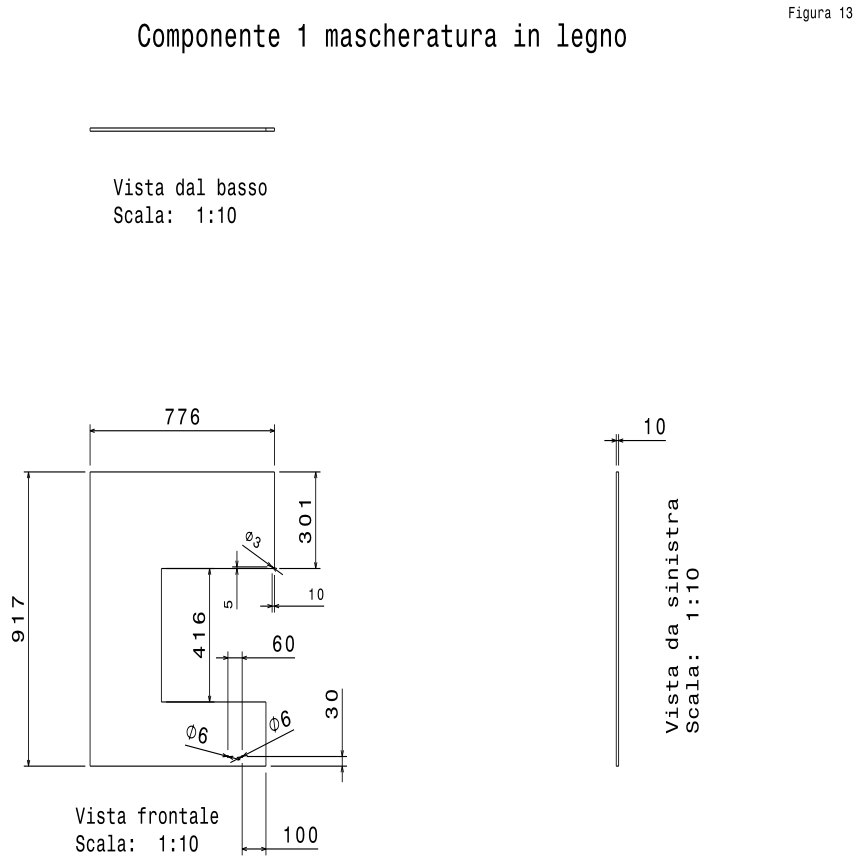


Figura A.4: componente 1 mascheratura in legno

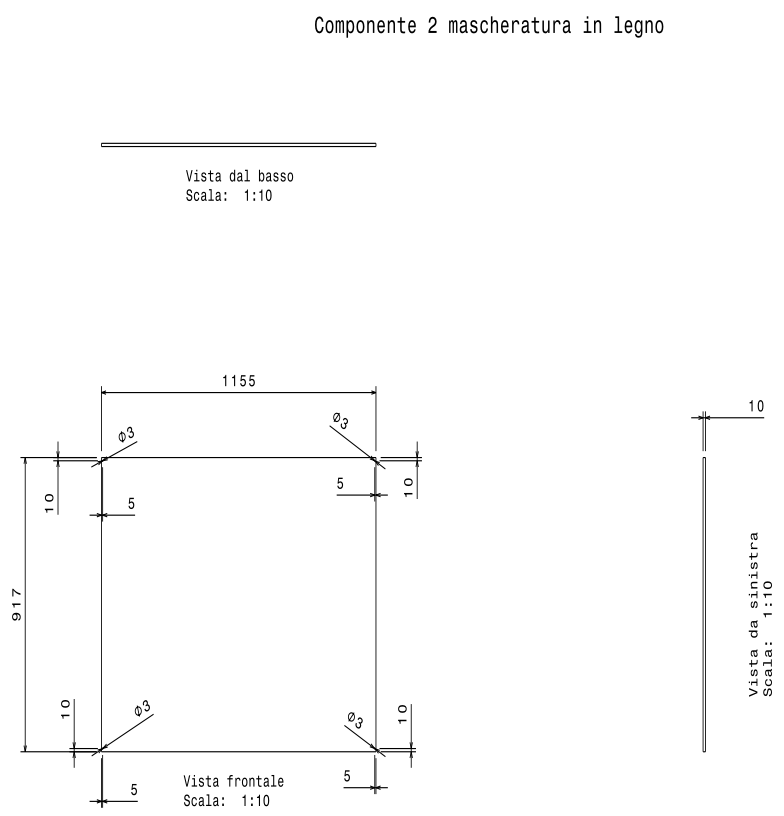


Figura A.5: componente 2 mascheratura in legno

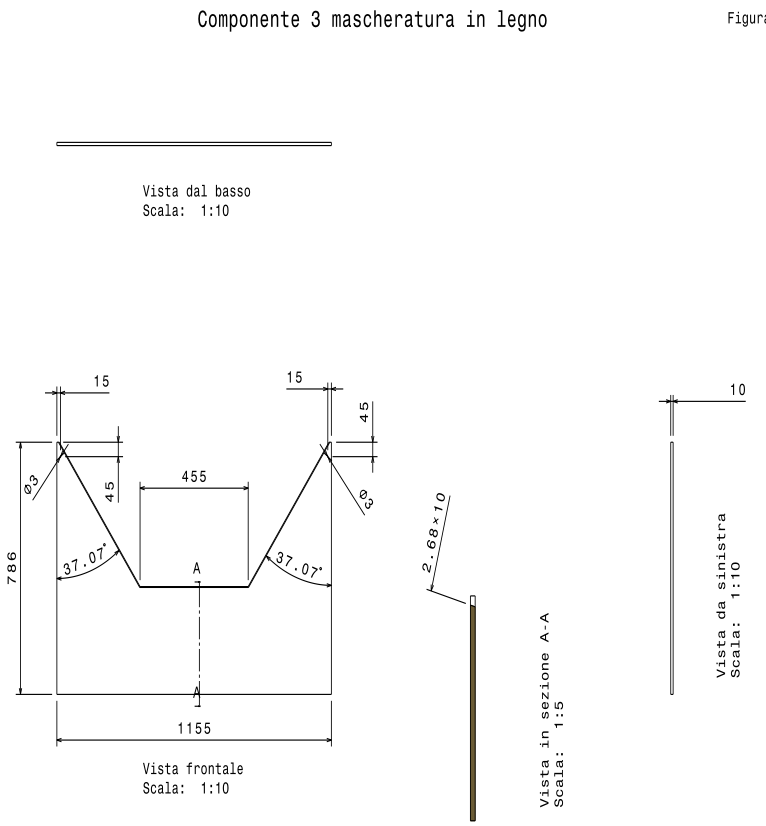


Figura A.6: componente 3 mascheratura in legno

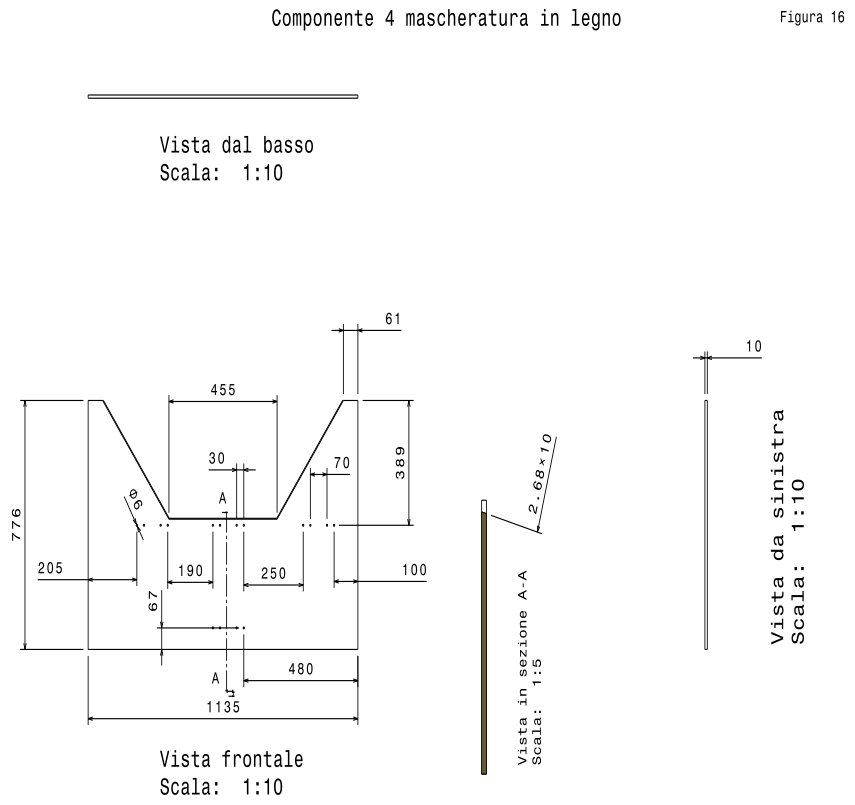


Figura A.7: componente 4 mascheratura in legno

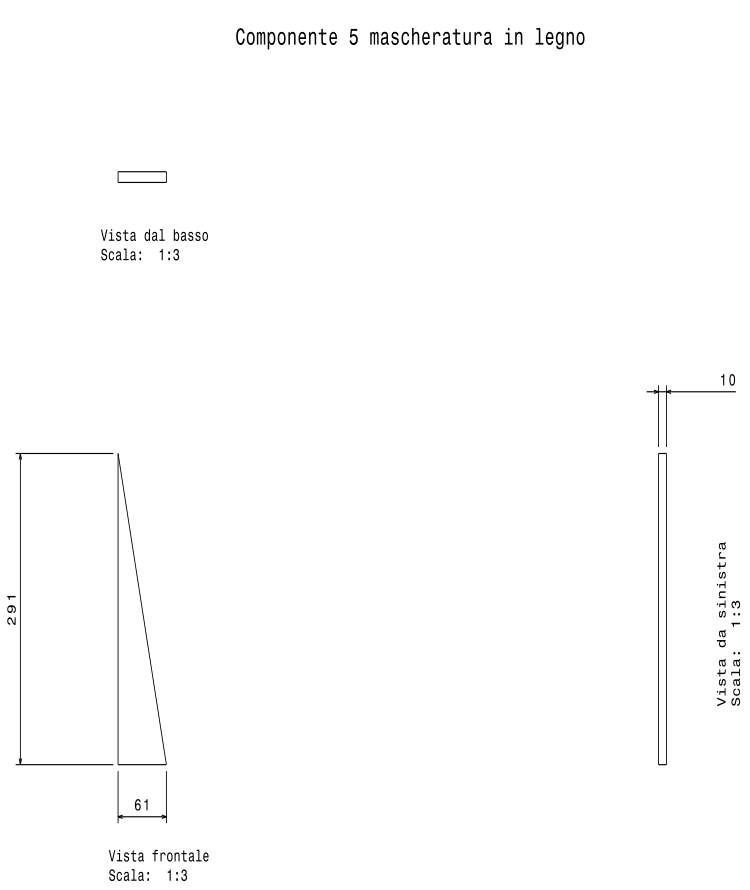


Figura A.8: componente 5 mascheratura in legno

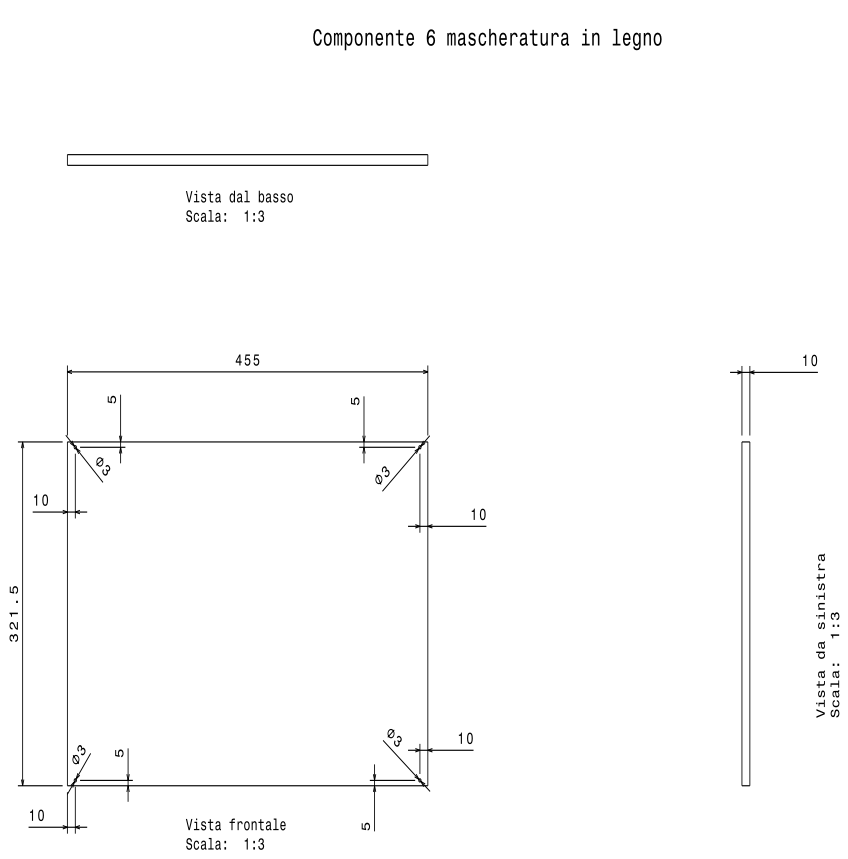


Figura A.9: componente 6 mascheratura in legno

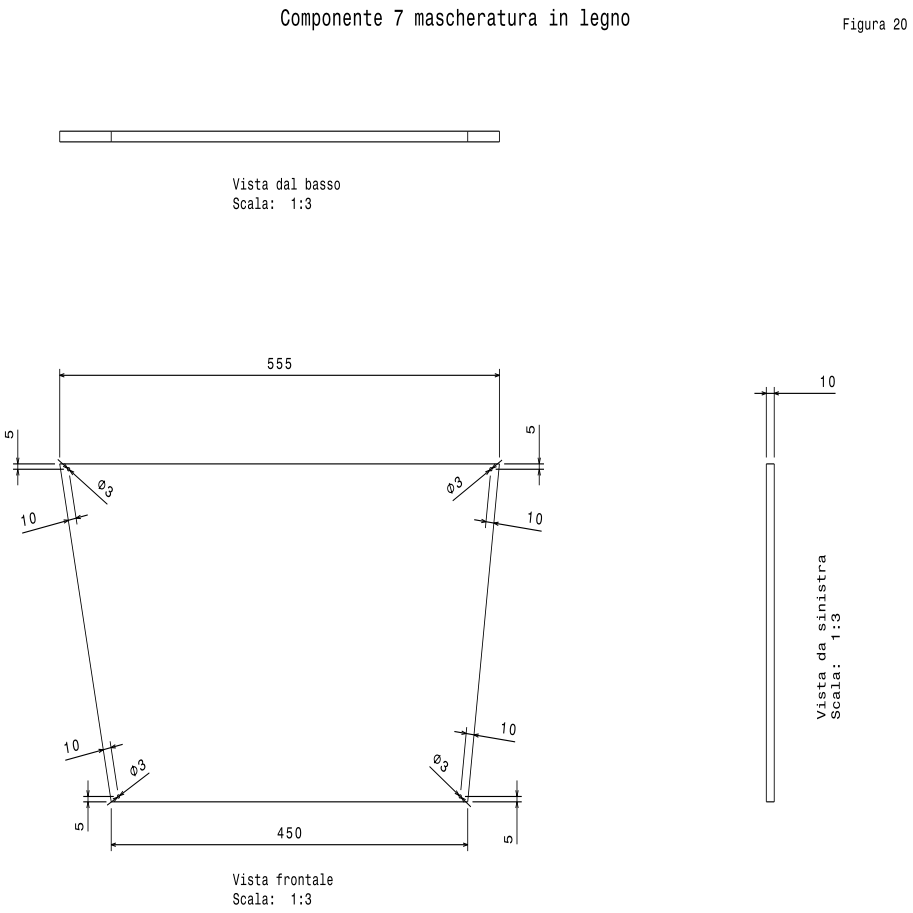


Figura A.10: componente 7 mascheratura in legno

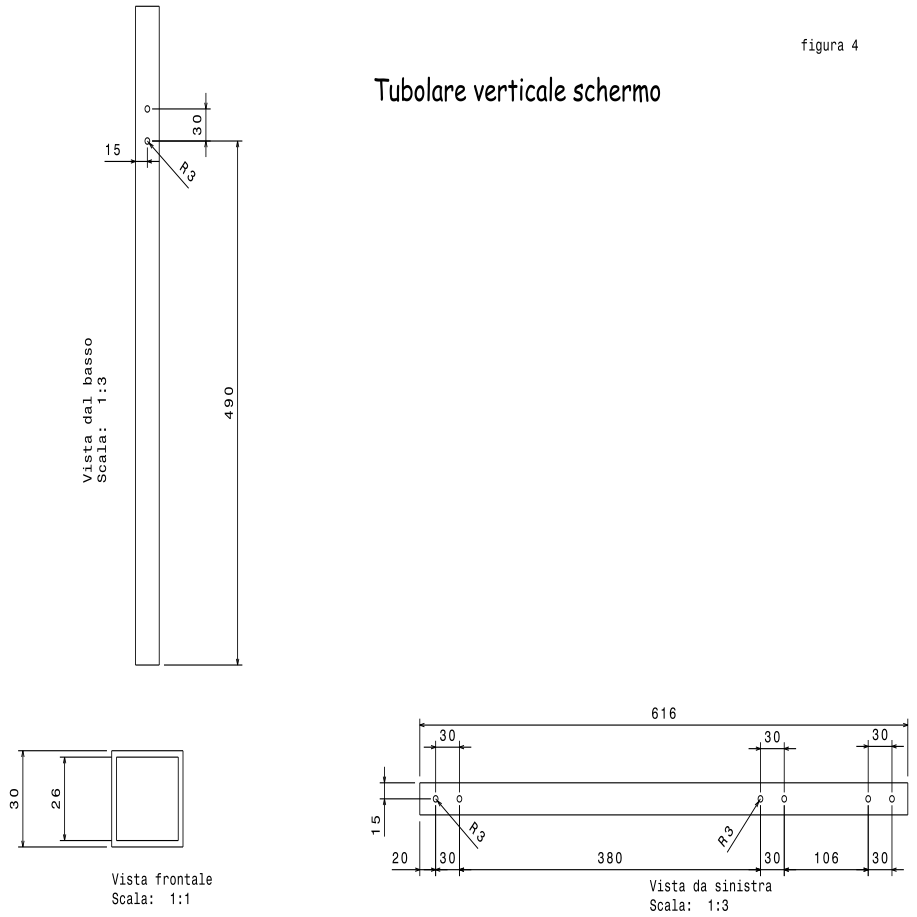


Figura A.11: tubolare verticale schermo

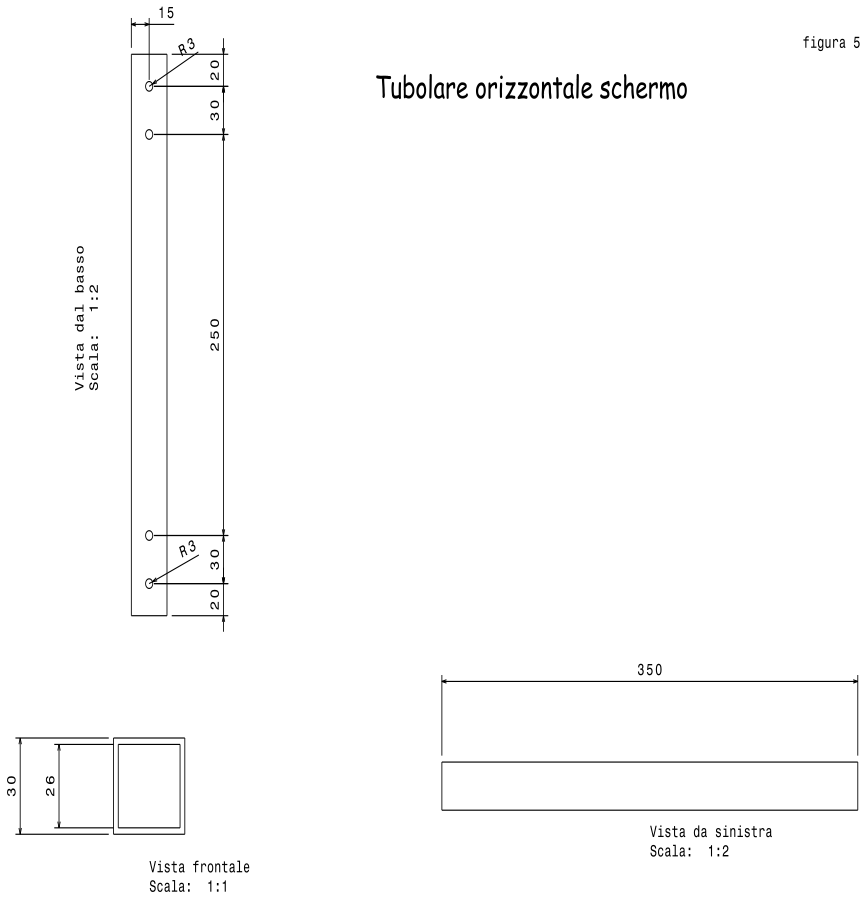


Figura A.12: tubolare orizzontale 1 schermo

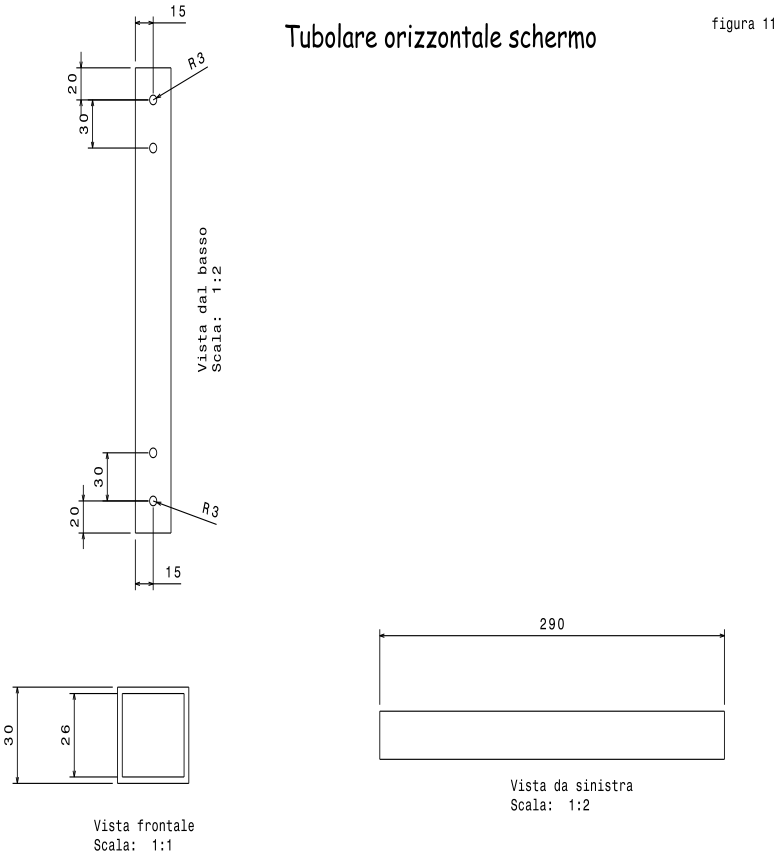


Figura A.13: tubolare orizzontale 2 schermo

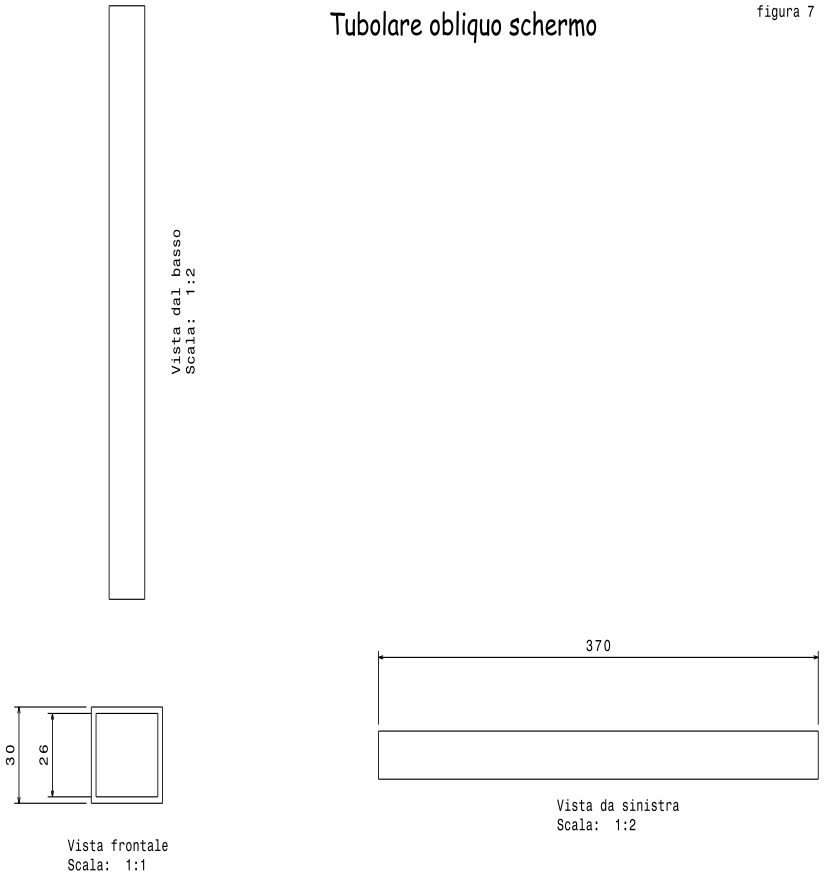


Figura A.14: tubolare obliquo schermo

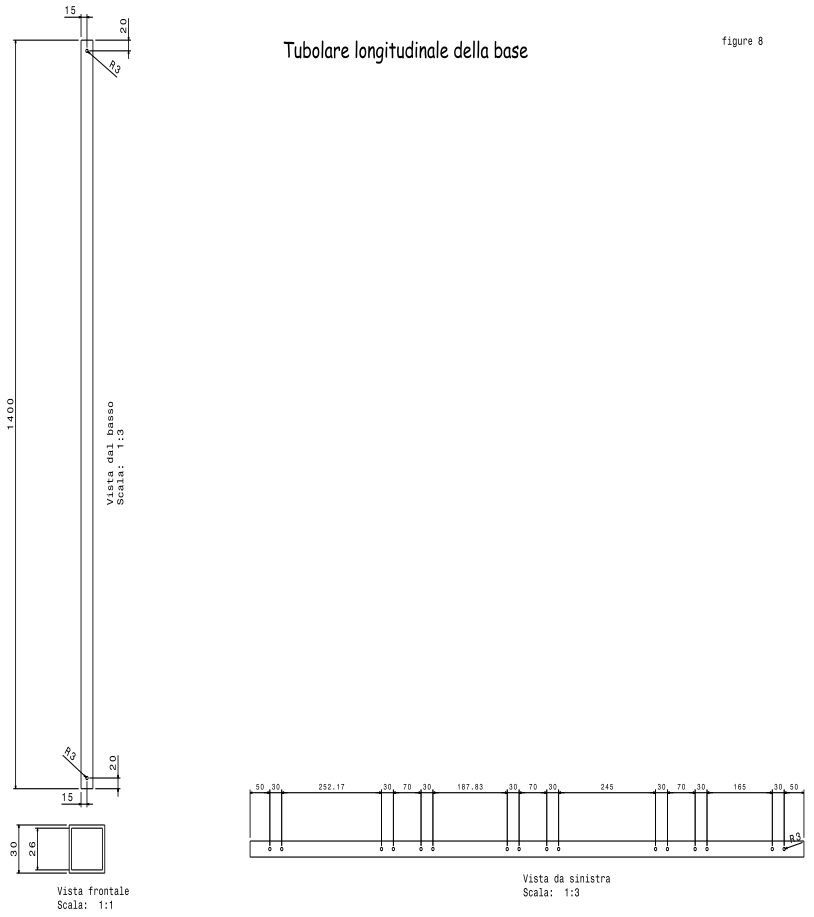


Figura A.15: tubolare longitudinale della base

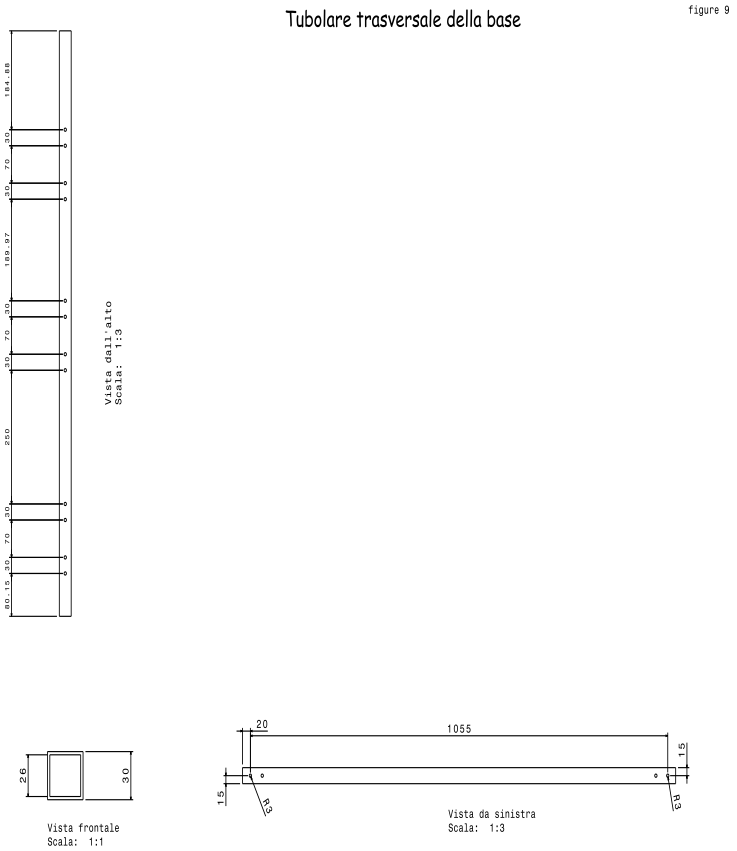


Figura A.16: tubolare trasversale della base

Appendice B

File di input a YASim

```
<!-- official RD + 0.53 m = yasim RD (reference datum) -->
<airplane mass="2868">

<approach speed="47" aoa="16">
  <control-setting axis="/controls/engines/engine[0]/throttle"
  value="0.3"/>
</approach>

<cruise speed="123" alt="0">
  <control-setting axis="/controls/engines/engine[0]/throttle"
  value="1.0"/>
</cruise>



<rotor name="main" x="-2.75" y="0.0" z="1.55" nx="0.05" ny="0"
  nz="1." fx="1" fy="0" fz="0" ccw="1"
  maxcollective="15.8" mincollective="0.2"
  mincyclicele="-4.7" maxcyclicele="10.5"
  mincyclicail="-4.23" maxcyclicail="5.65"
  diameter="9.98" numblades="4" weightperblade="75"
  relbladecenter="0.5"
  dynamic="1" rpm="442" rellenflaphinge="0.18" delta3="0"
  delta=".125"
  pitch-a="10"
  pitch-b="15"
  flapmin="-15"
  flapmax="15"
```

```
    flap0="-5"
    flap0factor="0"
    notorque="0"
    dragfactor="0.30"
    translift-ve="20"
    translift-maxfactor="1.3"
    ground-effect-constant="0.1"
    twist="-8.5"
    taper="1"
    chord="0.27"
    number-of-segments="8"
    number-of-parts="8"
    rel-len-where-incidence-is-measured="0.7"
    rel-len-blade-start="0.076"

    airfoil-lift-coefficient="3.85"
    airfoil-drag-coefficient0="0.0075"
    airfoil-drag-coefficient1="0.25"
    incidence-stall-zero-speed="15"
    incidence-stall-half-sonic-speed="14.5"
    lift-factor-stall="0.18"
    stall-change-over="5.5"
    drag-factor-stall="2.0"
    cyclic-factor="0.8"
    rotor-correction-factor="1.0"
    downwashfactor="1.0"
  >
    <control-input axis="/controls/flight/aileron-trim"
                   control="CYCLICAIL"/>
    <control-input axis="/controls/flight/aileron" control="CYCLICAIL"
                   src0="-1.0" src1="1.0"
                   dst0="-1.0" dst1="1.0"/>
    <control-input axis="/controls/flight/elevator-trim"
                   control="CYCLICELE"/>
    <control-input axis="/controls/flight/elevator" control="CYCLICELE"
                   src0="-1.0" src1="1.0"
                   dst0="-1.0" dst1="1.0"/>
    <control-input axis="/controls/engines/engine[0]/throttle"
                   control="COLLECTIVE"
                   src0="0.0" src1="1.0"
                   dst0="1.0" dst1="-1.0"/>
```

```
</rotor>


<rotor name="tail" x="-8.65" y="0.18" z="1.5" nx="0.07" ny="-1"
  nz="-0.05" fx="1" fy="0" fz="0" ccw="1"
  maxcollective="20" mincollective="-10"
  phi0="110"
  diameter="1.91" numblades="2" weightperblade="2"
  relbladecenter="0.7"
  dynamic="1" rpm="2219" rellenflaphinge="0.0" delta3="1"
  translift="0" delta="0.5"
  sharedflaphinge="1"
  flap0="-0.5"
  pitch-a="10"
  pitch-b="15"
  airfoil-lift-coefficient="6.4"
  airfoil-drag-coefficient0="0.0035"
  airfoil-drag-coefficient1="0.10"
  notorque="0"
  taper="1"
  chord="0.25"
  number-of-segments="5"
  number-of-parts="4"
  rel-len-blade-start="0.33"
  incidence-stall-zero-speed="9"
  incidence-stall-half-sonic-speed="18"
  lift-factor-stall="0.18"
  stall-change-over="5.5"
  drag-factor-stall="8"
  rotor-correction-factor="0.9"
  downwashfactor="1.0"
>
  <control-input axis="/controls/flight/rudder-trim"
                 control="COLLECTIVE" invert="true"/>
  <control-input axis="/controls/flight/rudder"
                 control="COLLECTIVE" invert="true"
                 src0="-1.0" src1="1.0"
                 dst0="-1.0" dst1="1.0"/>
  <control-input axis="/controls/engines/engine[0]/throttle"
                 control="COLLECTIVE"
                 src0="0.1" src1="1.0"
```

```
                    dst0=".0007" dst1="-.00055"/>
  </rotor>


  <rotorgear
    max-power-engine="626"
    max-power-rotor-brake="100"
    rotorgear-friction="1.4"
    engine-prop-factor="0.005"
    engine-accel-limit="4"
    yasimdragfactor="59.5"
    yasimliftfactor="60"
  >
    <control-input axis="/controls/engines/engine[0]/magnetos"
                   control="ROTORGEARENGINEON"/>
    <control-input axis="/controls/rotor/brake" control="ROTORBRAKE"
                   src0="0.0" src1="1.0"
                   dst0="0.0" dst1="1.0"/>
    <control-input axis="/controls/rotor/reltarget"
                   control="ROTORRELTARGET"
                   src0="0.0" src1="2.0"
                   dst0="0.0" dst1="2.0"/>
    <control-input axis="/controls/rotor/maxreltorque"
                   control="ROTORENGINEMAXRELTORQUE"
                   src0="0.0" src1="2.0"
                   dst0="0.0" dst1="2.0"/>
  </rotorgear>


  <cockpit x="-1.88" y=".33" z="0.6"/>

  <fuselage ax="-.25" ay="0" az="-.07" bx="-4.05" by="0" bz="-.07"
            width="1.92" taper="0.95" midpoint="0.5" idrag="0"/>

  <fuselage ax="-4.05" ay="0" az="-.38" bx="-7.60" by="0" bz=".51"
            width="0.47" taper="0.9" midpoint="0.05" idrag="0.3"/>


  <vstab x="-7.55" y="0" z=".41" taper=".5" length="1.37" chord=".59"
    sweep="40">
    <stall aoa="16" width="4" peak="1.5"/>
```

```
</vstab>

<vstab x="-7.12" y="1.27" z=".12" taper="1" length=".65"
  chord=".40" sweep="0">
  <stall aoa="16" width="4" peak="1.5"/>
</vstab>

<vstab x="-7.12" y="-1.27" z=".12" taper="1" length=".65"
  chord=".40" sweep="0">
  <stall aoa="16" width="4" peak="1.5"/>
</vstab>

<hstab x="-6.91" y="0.05" z="-0.44" taper="1.0" effectiveness="1.0
  length="1.20" chord="0.36" sweep="0" incidence="5">
  <stall aoa="0" width="0" peak="1.5"/>
</hstab>

<!-- left skid -->
<gear x="-1.24" y="+1.30" z="-1.22" skid="1" compression="0.10"
spring="0.8" sfric="0.5" dfric="0.4"/>
<gear x="-3.52" y="+1.30" z="-1.22" skid="1" compression="0.10"
spring="0.8" sfric="0.5" dfric="0.4"/>

<!-- right skid -->
<gear x="-1.24" y="-1.30" z="-1.22" skid="1" compression="0.10"
spring="0.8" sfric="0.5" dfric="0.4"/>
<gear x="-3.52" y="-1.30" z="-1.22" skid="1" compression="0.10"
spring="0.8" sfric="0.5" dfric="0.4"/>

<!-- tail skid -->
<gear x="-8.03" y="+0.00" z="+0.34" skid="1" compression="0.01"
spring="0.1" sfric="1" dfric="1"/>


<tank x="-2.69" y="0" z=".68" capacity="1006"/>  <!-- main:
580 l (570 l usable) @ 6.682 ppg (H3EU.pdf) -->
<tank x="-1.74" y="0" z=".68" capacity="163.7"/> <!-- supply:
93 l (usable?) -->

<ballast x="-1.6" y="+0.0" z="0.0" mass="150"/>
<ballast x="-2.7" y="-1.8" z="0.0" mass="200"/>
```

```
<ballast x="-1.7" y="+0.0" z="1.0" mass="600"/>
<ballast x="-2.7" y="+1.8" z="0.0" mass="200"/>
<ballast x="-3.2" y="+0.0" z="0.0" mass="150"/>

<ballast x="-5.0" y="+0.0" z="1.0" mass="300"/>

<weight x="-1.0" y=".33"  z="1" mass-prop="/sim/
  weight[0]/weight-lb"/>          <!-- pilot -->
<weight x="-1.0" y="-.33" z="1" mass-prop="/sim/
  weight[1]/weight-lb"/>          <!-- co-pilot -->
<weight x="-2.0" y=".33"  z="1" mass-prop="/sim/
  weight[2]/weight-lb"/>          <!-- right passenger -->
<weight x="-2.0" y="0"    z="1" mass-prop="/sim/
  weight[3]/weight-lb"/>          <!-- middle passenger -->
<weight x="-2.0" y="-.33" z="1" mass-prop="/sim/
  weight[4]/weight-lb"/>          <!-- left passenger -->
<weight x="-3.5" y="0"    z="1" mass-prop="/sim/
  weight[5]/weight-lb"/>          <!-- patient/load -->

<weight x="-2.93" y="0.4"   z="-0.19" size="0.25"
mass-prop="/sim/model/bo105/weapons/MG[0]/weight-lb"/><!--R-->
<weight x="-2.93" y="-0.4"  z="-0.19" size="0.25"
mass-prop="/sim/model/bo105/weapons/MG[1]/weight-lb"/><!--L-->

<weight x="-2.63" y="-1.58" z="-0.25" size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[0]/weight-lb"/><!--OR-->
<weight x="-2.63" y="1.58"  z="-0.25" size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[1]/weight-lb"/><!--OL-->
<weight x="-2.63" y="-1.39" z="-0.07" size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[2]/weight-lb"/><!--MR-->
<weight x="-2.63" y="1.39"  z="-0.07" size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[3]/weight-lb"/><!--ML-->
<weight x="-2.63" y="-1.21" z="0.08"  size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[4]/weight-lb"/><!--IR-->
<weight x="-2.63" y="1.21"  z="0.08"  size="0.18"
  mass-prop="/sim/model/bo105/weapons/HOT[5]/weight-lb"/><!--IL-->

<thruster x="-2.55" y="0.0" z="1.55" vx="1" vy="0" vz="0"
  thrust="4000">
  <control-input axis="/rotors/main/vibration/longitudinal"
    src0="-1" src1="1" dst0="-1" dst1="1" control="THROTTLE"/>
```

```
</thruster>

<thruster x="-2.55" y="0.0" z="1.55" vx="0" vy="1" vz="0"
   thrust="4000">
   <control-input axis="/rotors/main/vibration/lateral"
      src0="-1" src1="1" dst0="-1" dst1="1" control="THROTTLE"/>
</thruster>

</airplane>}
```

Bibliografia

[1] Department of defense. MIL-STD 1472, 1989.

[2] *Hybrid III 95th Percentile Large Male.* `www.dentonard.com`.

[3] *FAA Hybrid III 50th Percentile Male.* `www.dentonard.com`.

[4] Department of defense. MIL-STD 203, 1991.

[5] Department of defense. MIL-STD 250, 1968.

[6] Department of defense. MIL-STD 850b, 1967.

[7] Department of defense. MIL-B 8584, 1985.

[8] C. Malavasi. Vademecum per l' ingegnere costruttore meccanico. Ulrico Hoepli, xiv edition, 1974.

[9] ESDU 75030. *Natural frequencies of rectangular flat plates with various edge conditions*, November 1975.

[10] Paolo Mantegazza. Dispense del corso di dinamica e controllo di strutture aerospaziali.

[11] M. Gennaretti J. Serafini P. Masarati, G. Quaranta. *Aeroservoelastic Analysis of Rotorcraft-Pilot Coupling: a Parametric Study.*

[12] Jon S. Berndt and the JSBSim development team. *JSBSim An open source, platform-independent, flight dynamics model in C++*, 2009.

[13] Roggero Piero. *Linguaggio di programmazione C++.*

[14] Stuart Buchanan Jon Berndt Bernhard Buckel Cameron Moore Curt Olson Dave Perry Michael Selig Darrell Walisser Michael Basler, Martin Spott. *The FlightGear Manual*, March 2009. `http://wiki.flightgear.org/index.php/Main_Page`.

[15] E. Bruce Jackson. *Manual for a Workstation-based Generic Flight Simulation Program (LaRCsim) Version 1.4.*

[16] P. Masarati G. Quaranta O. Dieterich J. Serafini, M. Gennaretti. *Aeroelastic and Biodynamic Modelling for Stability Analysis of Rotorcraft-Pilot Coupling Phenomena.*

[17] Eric F. Sorton and Sonny Hammaker. *Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear*, 2005. Fairmont, West Virginia.

[18] Ronald P.M. Verhoeven and Antoine J.C. de Reus. *Human Factors Assistance during Prototyping of Cockpit Applications*, 2006. Amsterdam, The Netherlands.

[19] J. Serafini M. Gennaretti P. Masarati, G. Quaranta. *Numerical investigation of aeroservoelastic rotorcraft-pilot coupling*, 2007. in XIX Congresso Nazionale AIDAA, Forlí.

[20] W. Basso R. Bianco-Mengotti C. Monteggia P. Masarati, G. Quaranta. *Biodynamic tests for pilots' characterization on the BA-609 fly-by-wire tiltrotor*, 2009. in XX Congresso Nazionale AIDAA, Milano.

[21] M. Mattaboni A. Fumagalli P. Masarati, G. Quaranta. *Identification of the biomechanical behavior of a rotorcraft pilot arm*, 2009. in XX Congresso Nazionale AIDAA, Milano.

[22] M. Jump P. Masarati-G. Quaranta M. Mattaboni, A. Fumagalli. *Biomechanical pilot properties identification by inverse kinematics/inverse dynamics multibody analysis.* Department of Engineering, The University of Liverpool.

[23] P. Masarati G. Quaranta M. Mattaboni, M. Jump. *Experimental identification of rotorcraft pilots' biodynamic response for investigation of PAO events.* Department of Engineering, The University of Liverpool.

[24] P.P. Valentini L. Vita E. Pennestri, R. Stefanelli. *Virtual musculoskeletal model for the biomechanical analysis of the upper limb*, May 2006.

[25] A. Lazzari. *Comunicazioni fra Processi - l' interfaccia socket*, Giugno 2001.

[26] Mayo J.R. *The Involuntary Participation of a Human Pilot in a Helicopter Collective Control Loop*, September 1989.

[27] `http://wiki.blender.org/index.php/Doc:Manual`.