POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in
Ingegneria Aeronautica

Generazione automatica della geometria del velivolo mediante Catia

Relatore: Prof. Sergio RICCI

Tesi di Laurea di:

Luca PARPINELLI Matr. 733978

Anno Accademico 2011 – 2012

Indice

Elenco delle figure

Sommario

Come aggiunta alle funzionalità di CEASIOM ci si propone con questa tesi di fornire la possibilità di ottenere un modello CAD del velivolo completo fornendo al programma solo alcuni parametri geometrici. L'obiettivo è quindi risparmiare tempo nella modellazione CAD e ottenere una rappresentazione grafica immediata migliore di quella fornita da AcBuilder. Verrà proposta una metodologia di modellazione sfruttando il software CATIA V5, illustrando tutti i passaggi necessari per generare il modello generico e i comandi necessari per permetterne la modifica automatica. I risultati verranno poi confrontati con la rappresentazione fornita da AcBuilder per verificare che non vi siano incongruenze.

Parole chiave: modellazione automatica,

Abstract

The aim of this master thesis is to produce an extension of the CEASIOM's functionality, to have the possibility of obtaining a CAD model of the whole airplane starting from few geometric parameters, saving time in the CAD modeling phase and achieving a better instant representation if compared to the AcBuilder's one. A model metodology using CATIA V5 will be shown, highlighting all the steps to create a generic model and all the necessary commands to obtain the automatic configuration's change. The result of AcBuilder and of this procedure will be compared, to verify the presence of any differences.

Key words: automatic modeling

Introduzione

CEASIOM è l'acronimo di Computerised Environment for Aircraft Synthesis and Integrated Optimisation Methods. Si tratta di un pacchetto formato da diversi moduli che, sviluppato all'interno del progetto SimSAC, acronimo di Simulate Stability And Control, ha lo scopo di progettare un velivolo senza dover fare riferimento a programmi che, partendo da metodi da manuale, operano degli adattamenti in base ad esperienze pregresse. L'obiettivo è ottenere un programma che calcoli le informazioni necessarie da principi primi, così da permettere lo studio anche di configurazioni innovative. Particolare attenzione viene posta alla possibilità di giungere il prima possibile ad una stima accurata del comportamento dinamico del velivolo, per poterne progettare al meglio il sistema di controllo. L'intento è quello di ottenere soluzioni nell'ottica del "First-Time-Right", per evitare i costi e i rischi elevati di eventuali correzioni durante la fase di verifica in volo. Unendo differenti pacchetti è possibile analizzare diversi aspetti del progetto del velivolo in tempi ristretti, riducendo quindi i costi relativi al progetto preliminare, che rappresentano l'80% del life-cycle cost.

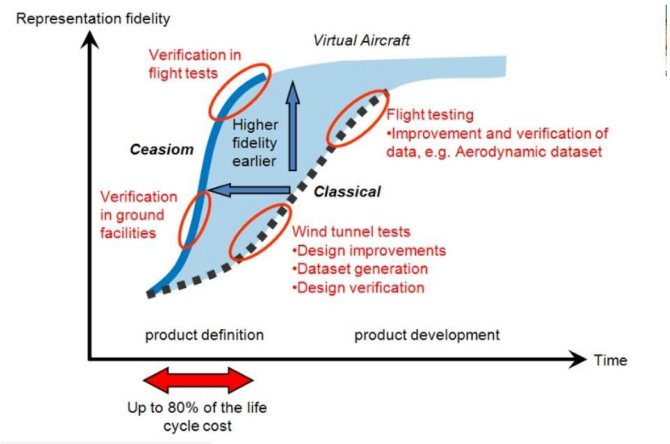

Figura 1 Progetto virtuale e classico del velivolo

CEASIOM ha quindi lo scopo di assistere gli ingegneri nella fase di progetto concettuale, attribuendo particolare enfasi alla possibilità di prevedere il

comportamento dinamico del velivolo, grazie all'elevata affidabilità degli attuali metodi di calcolo. Inoltre, racchiude in un'unica applicazione i diversi aspetti che maggiormente influenzano le prestazioni del velivolo: aerodinamico, strutturale e dinamica di volo. Si può quindi parlare di un progetto *aeroservoelastico* del velivolo.
CEASIOM, tuttavia, non provvede all'intero progetto concettuale, richiedendo, in ingresso, una configurazione di base. Di seguito è proposto uno schema riassuntivo di come è strutturato il software CEASIOM, mostrando aspetti delle sue funzionalità e il flusso logico di come avviene l'elaborazione dei dati.

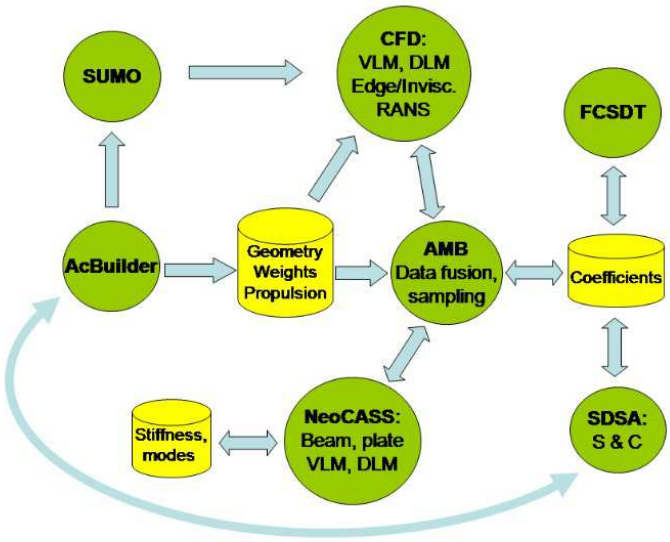

Figura 2 Mappa della struttura di CEASIOM

Molti aspetti sono sviluppati in moduli integrati:

- AcBuilder
  Fornisce un modello 3D del velivolo, sia interno che esterno, basato su alcuni parametri caratteristici.

- Geometry, Weights
  Integrato all'interno di AcBuilder, è utilizzato per calcolare i parametri di riferimento delle ali, così come pesi e baricentri.

- Propulsion
  Fornisce la spinta necessaria e prodotta, basandosi sull'altitudine e sui parametri del velivolo.

- SUMO
  Genera la mesh data la geometria del velivolo, necessario per alcuni solutori di CFD.

- AMB
  Aerodinamic Model Builder è il GUI per i solutori CFD, che sono 4 in totale.

- SDSA
  Simulation and Dynamic Stability Analysis fornisce sia la stabilità, statica e dinamica, che simulazioni delle condizioni di volo per valutare la manovrabilità.

- FCSDT
  Flight Control System Design Toolkit permette di valutare vari aspetti del sistema di controllo.

- NeoCASS
  Next generation Conceptual Aero-Structural Sizing è un'applicazione Matlab che permette di effettuare un'analisi aeroelastica.

1 Modulo geometrico AcBuilder

AcBuilder è il modulo principalmente preposto alla definizione della geometria e alla creazione del file XML, e può, quindi, essere considerato come un punto di partenza per il software CEASIOM, l'esecuzione avviene sia direttamente all'interno di CEASIOM che mediante Matlab.
La schermata iniziale del programma presenta la visualizzazione di un velivolo accompagnata da una tabella, in cui è possibile variare i parametri che caratterizzano la geometria. La modifica può essere effettuata manualmente, oppure tramite importazione di un file XML con la geometria già definita.
La visualizzazione java della geometria del velivolo viene aggiornata ogni volta che viene modificato un parametro. Alcuni di questi, come il profilo alare o la sezione di fusoliera, richiamano delle finestre secondarie, che permettono di visualizzare in dettaglio le differenti configurazioni.

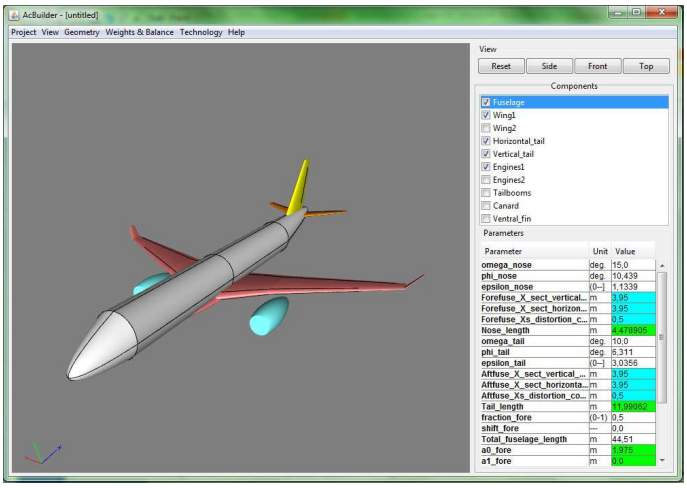

Figura 3 Schermata iniziale AcBuilder

Dal menu geometry è possibile passare alla sezione fuel, da cui si può impostare la posizione e le dimensioni dei serbatoi e definire le caratteristiche del cassone alare, che saranno utili per l'analisi strutturale.

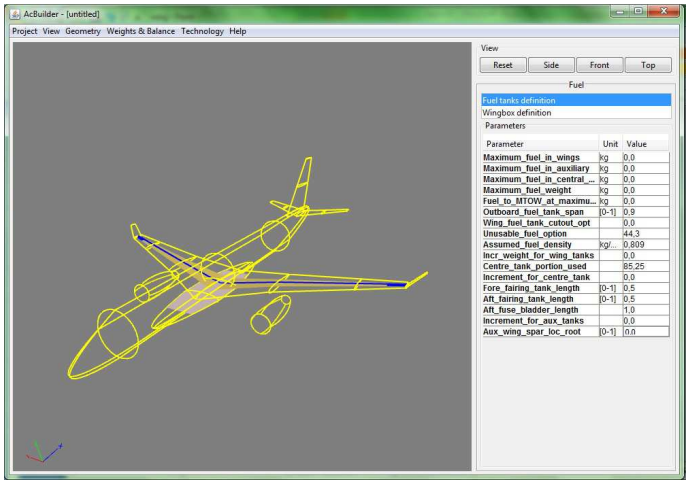

Figura 4 Schermata AcBuilder fuel

Dal menu Weights & Balance è possibile definire la geometria interna della fusoliera, la disposizione dei passeggeri e della stiva. In questa sezione si possono modificare le posizioni di impianti e strutture, definendone anche i pesi. Tutto ciò con lo scopo di determinare la posizione del baricentro e i momenti d'inerzia.

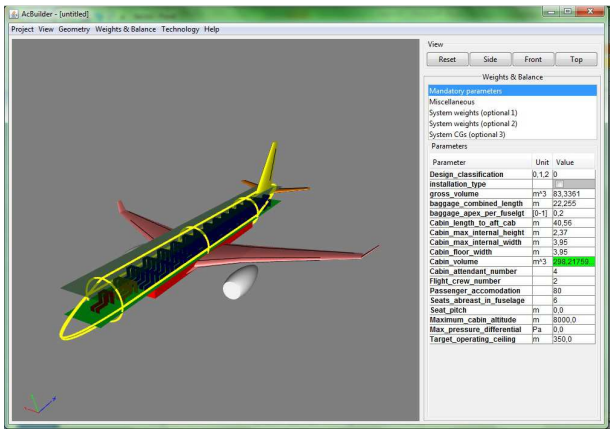

Figura 5 Schermata AcBuilder Weights & Balance

Dal menu Technology si possono impostare i parametri per il modulo NeoCASS: geometria approssimata e proprietà, carichi e tipo di analisi. Questi parametri possono anche essere importati da un file XML.

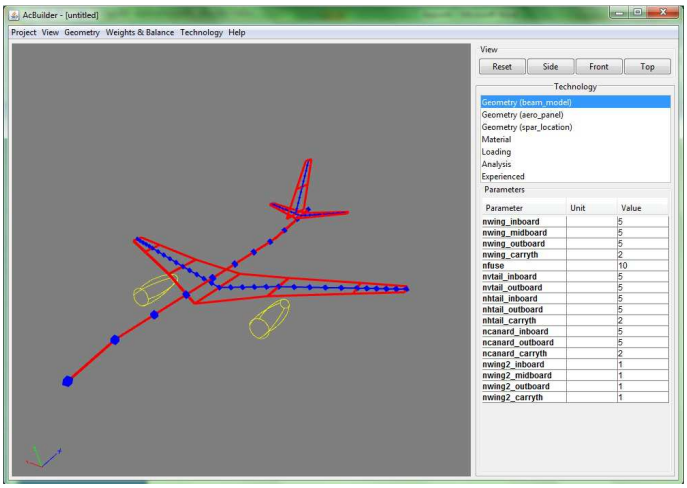

Figura 6 Schermata AcBuilder Technology

Definiti tutti i parametri necessari è possibile esportare un file in formato XML e procedere con il resto delle funzionalità di CEASIOM.

2 Gestione file XML e trasferimento dati in CATIA

Nel presente capitolo si esaminerà la modalità di composizione di un file XML di interesse e la procedura necessaria per giungere ad una struttura di dati in un formato utilizzabile con CATIA.

2.1 Composizione del file XML

I dati geometrici contenuti nel file di output di AcBuilder hanno una struttura ad albero. All'aereo sono associati diversi sottoelementi, che a loro volta sono il riferimento per ulteriori sottoelementi oppure sono il riferimento per un attributo, che viene rappresentato con un *array* di tipo *char* (*character array*). Nel diagramma che segue, fig. 7, è visualizzato il percorso che permette di accedere ai parametri geometrici dell'ala. Selezionata l'ala nel primo livello, è possibile accedere al secondo livello, da cui si possono ottenere i valori assegnati, per esempio ad "Area", oppure accedere al terzo livello, a titolo esemplificativo selezionando "Flap", e così procedendo fino al termine della struttura.

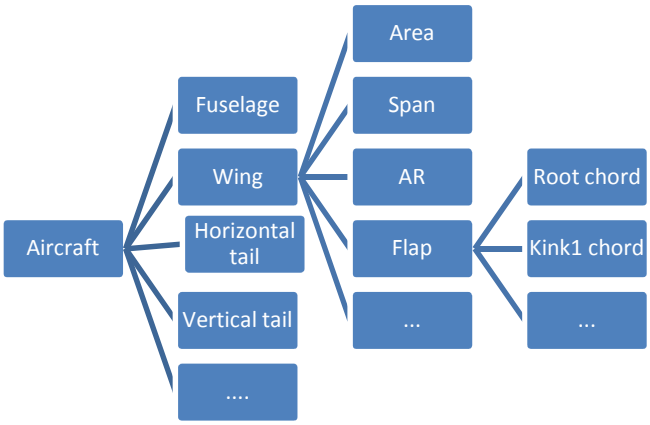

Figura 7 Struttura XML

2.2 Lettura del file XML in Matlab e creazione tabelle exel

Il file XML, esportato da AcBuilder, viene importato e letto da Matlab mediante i seguenti comandi:

```
xmlstr = fileread('prova.xml');
V = xml_parseany(xmlstr);
```

A “V” è associata la struttura descritta in precedenza, alla quale si accede con una sintassi come la seguente:

```
V.Fuselage{1,1}.omega_nose{1,1}.CONTENT
```

Il risultato che si ottiene è il valore associato ad omega_nose espresso come char, per ottenere il valore numerico corrispondente bisogna utilizzare il seguente comando:

```
omega_nose = str2double
(V.Fuselage{1,1}.omega_nose{1,1}.CONTENT);
```

Se si desidera ottenere la stringa corrispondente, come nel caso della sigla relativa al profilo alare, è necessario invece utilizzare il seguente comando:

```
airfoilRoot = num2str (V.Wing1{1,1}.airfoilRoot{1,1}.CONTENT);
```

Ottenuti tutti i parametri necessari nella forma desiderata, è possibile eseguire le operazioni che, come verrà illustrato in seguito, condurranno alla rappresentazione della geometria. Le informazioni da inviare a CATIA vengono ordinate in un vettore, denominato “command”. A questo dovrà esserne affiancato un altro, che servirà come indice per le associazioni ai vincoli. Affiancando i due vettori si ottiene una matrice, che viene salvata in un file exel. Tutto questo viene svolto dai seguenti comandi:

```
command(n) = … ;
n = n+1;
N = length(command);
x = 1:1:N;
q = [x; command];
q = q';
xlswrite('fusoliera',q);
```

2.3 Tabella di progetto in CATIA

In ambiente CATIA è possibile importare i dati contenuti nel file Exel appena creato tramite il comando “tabella di progetto”; dovranno essere spuntate, come si desume in fig. 8, le opzioni per selezionare un file esistente e per definire la direzione di lettura orizzontale, al fine di assecondare l’impostazione del file di origine. Le voci “Nome” e “Commento” servono per definire l’identificativo ed un’eventuale nota.

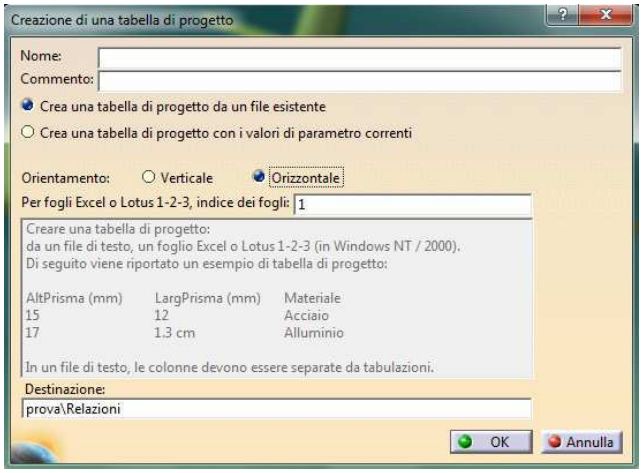

Figura 8 Definizione pannello di controllo

Dopo aver selezionato il file di origine, si ottiene la schermata di fig. 9, la quale costituisce il cuore dell’automazione del lavoro. In tale ambito, infatti, è possibile attribuire ad ogni vincolo presente nel progetto un valore contenuto nel file selezionato. Nella colonna di sinistra si seleziona il vincolo, in quella centrale l’identificativo della riga a cui corrisponde il valore desiderato e a destra c’è una tabella riassuntiva delle associazioni già stabilite.

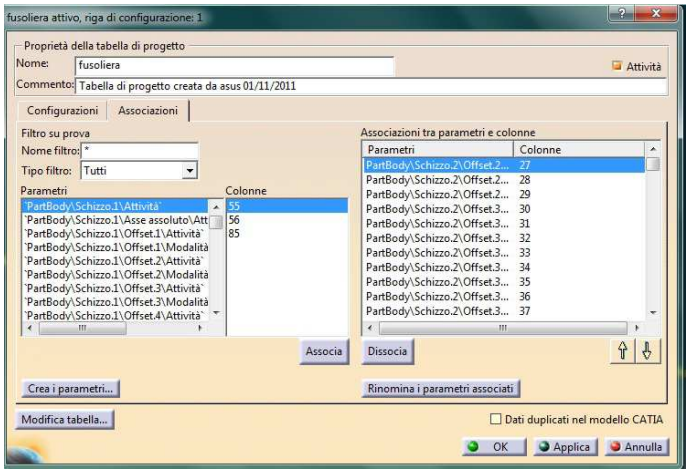

Figura 9 Pannello associazioni

Terminate le associazioni e atteso che il programma recepisca i cambiamenti, evento che viene evidenziato dalla colorazione rossa della parte interessata e mediante l'apertura di una finestra di notifica, con il comando "aggiorna" CATIA rigenera la geometria con i nuovi vincoli. Qualora si verificassero delle incongruenze o non fosse più possibile eseguire alcuni comandi, l'aggiornamento si interromperebbe e sarebbe visualizzata una finestra che, oltre ad identificare il tipo di errore, permetterebbe di modificare direttamente il comando interessato.

Per la parametrizzazione del velivolo verranno create diverse tabelle di progetto, una per ogni componente, in modo da rendere la procedura più semplice ed intuitiva, sia in fase di realizzazione che di eventuale modifica. Nella figura che segue viene rappresentato un esempio di albero logico in cui dal ramo "Relazioni" si accede alle diverse tabelle di progetto.

Figura 10 Albero logico

2.4 Esecuzione Automatica

La procedura automatica prevista per la generazione del modello CATIA consiste nell'esecuzione del file Matlab velivolo_da_XML.m, che richiede di selezionare il file di estensione XML con i dati che si vogliono rappresentare e che fornisce le tabelle exel richieste da CATIA. Per ottenere la visualizzazione in CATIA sarà sufficiente aprire il file Valivolo_modello.CATShape ed eseguire il comando "aggiorna".

3 Modello di velivolo base in CATIA

In questo capitolo verrà illustrato il procedimento seguito per ottenere un modello CATIA generico di un velivolo. Per la definizione di ogni componente si farà ricorso alle stesse approssimazioni e formule implementate per creare la visualizzazione in AcBuilder. Nei paragrafi che seguono saranno presentate nel dettaglio, accompagnate dai passaggi necessari ad applicarle in ambiente CATIA.

3.1 Fusoliera

3.1.1 Sezioni fusoliera

Le sezioni della fusoliera sono definite deformando una circonferenza, mediante lo spostamento del centro lungo l'asse verticale. Per quantificare tale spostamento occorre utilizzare il coefficiente di distorsione, che è definito come il rapporto tra la distanza intercorrente fra il punto inferiore della circonferenza e la posizione del centro $(\xi_x * dv)$ e l'asse verticale (dv), in modo che $0 < \xi_x < 1$.

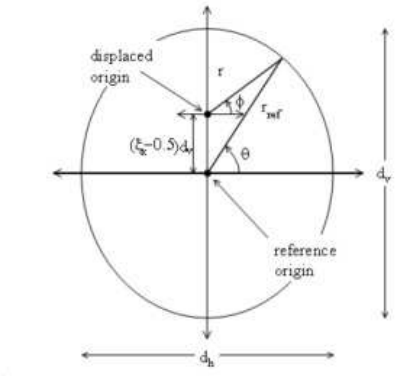

Figura 11 Sezione di fusoliera

Come emerge dalla figura 11, definita la posizione del centro, la geometria della sezione viene determinata sfruttando la seguente espressione:

$$r(\phi) = a_0 + a_1 \sin\phi + b_1 \cos 2\phi$$

I coefficienti di tale equazione vengono calcolati per via numerica.
La sezione è stata ricavata utilizzando una *spline* che interpola 13 punti, ottenuti calcolando il valore di r(φ) ogni $\frac{\pi}{12}$ nell'intervallo compreso tra 270° e 90°.
Le coordinate dei punti del contorno della fusoliera si determinano da:

$$\begin{cases} y_f = r(\phi)\cos\phi \\ z_f = \sin\phi + dv(\xi - 0.5) + 10 \end{cases}$$

Nella definizione di z_f si è tenuto conto della variazione dell'origine e si è aggiunto un *offset* di 10m, in modo da avere uniformità di segno tra i punti della sezione ed evitare, così, problemi in CATIA .
Per completare la sezione, viene sfruttata la simmetria rispetto al piano xz.
Nel modello si possono definire tre sezioni, anteriore, di mezzeria e posteriore, e un offset verticale.

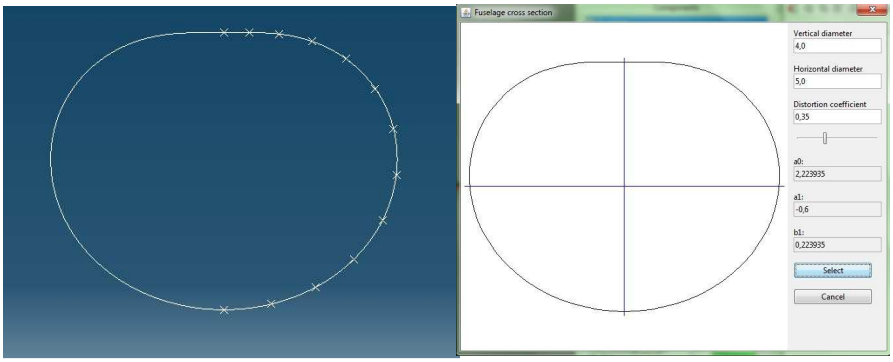

Figura 12 Sezioni fusoliera in CATIA e AcBuilder

3.1.2 Naso e coda

Per la definizione del naso e della coda si farà riferimento alla figura seguente:

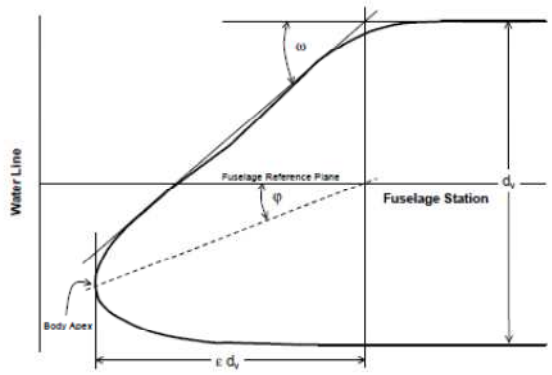

Figura 13 Convenzioni per la sezione del naso

Importati ε, ϕ, ω dal *file* .XML, si può determinare β mediante la formula empirica

$\beta = 0.54 + 0.1 \tan(\omega - \varphi)$.

Le curve che definiscono il profilo nel piano xz sono:

$$z(x) = \begin{cases} -\dfrac{d_v^{(1-\beta)}}{2}\left(\dfrac{x}{\varepsilon}\right)^{\beta} - (\varepsilon d_v - x)\tan\varphi + 2d_v(\xi - 0.5) + offset_{down} \\ \dfrac{d_v^{(1-\beta)}}{2}\left(\dfrac{x}{\varepsilon}\right)^{\beta} - (\varepsilon d_v - x)\tan\varphi + 2d_v(\xi - 0.5) + offset_{up} \end{cases}$$

Con gli *offset* definiti da:

$$offset_{down} = 10 + x\frac{\left(z_f(270°) - z_d(l_{nose})\right)}{l_{nose}} + shift_{fore} * dv$$

$$offset_{up} = 10 + x\frac{\left(z_f(90°) - z_u(l_{nose})\right)}{l_{nose}} + shift_{fore} * dv$$

Con $shift_{fore}$ si intende lo sfasamento della sezione anteriore rispetto a quella posteriore.
Come per la sezione di fusoliera, vengono create delle *spline*, una superiore ed una inferiore. Questo, però, solo dopo aver ridotto dell'1% la z dell'estremità inferiore ed aumentato dell'1% quella dell'estremità superiore, perché in CATIA, se i punti coincidono, non è possibile generare la superficie. Per ottenere una buona rappresentazione delle curve, ma non appesantire inutilmente il disegno, sono stati utilizzati sette punti per ogni *spline*. Tali punti sono stati congiunti mediante una circonferenza, che verrà utilizzata per creare la superficie del naso con il comando "*superficie multi sezione*". Con lo stesso procedimento è possibile ottenere la coda.

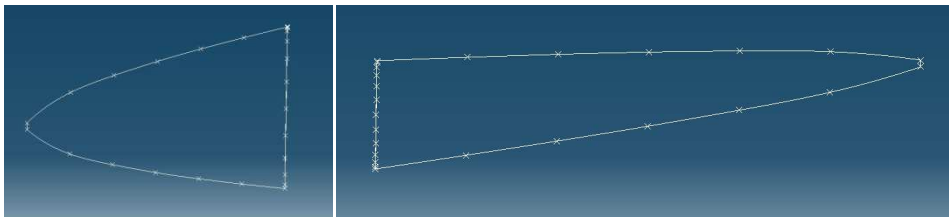

Figura 14 Sezioni del naso e della coda

La superficie centrale della fusoliera viene realizzata utilizzando il comando "*connessione*" per collegare le tre sezioni caratteristiche.
Per raccordare le superfici dei diversi segmenti da cui è composta la fusoliera sono stati eseguiti dei tagli, utilizzando dei piani paralleli al piano yz. Per ciascuna sezione sono stati utilizzati 2 piani simmetrici rispetto alla sezione di interesse. Tanto per il taglio del naso quanto per quello della coda i piani sono fatti passare per il primo punto utilizzato per la *spline*; per la sezione centrale si è ipotizzata una distanza di 1,2 m.

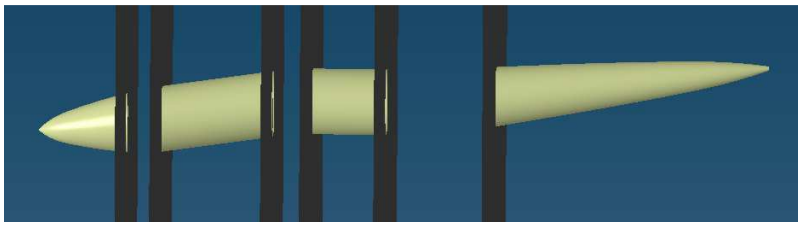

Figura 15 Piani di sezione della fusoliera

Di seguito si può vedere un confronto della visualizzazione della sola fusoliera fornita da AcBuilder con quella ottenuta con CATIA utilizzando gli stessi parametri.

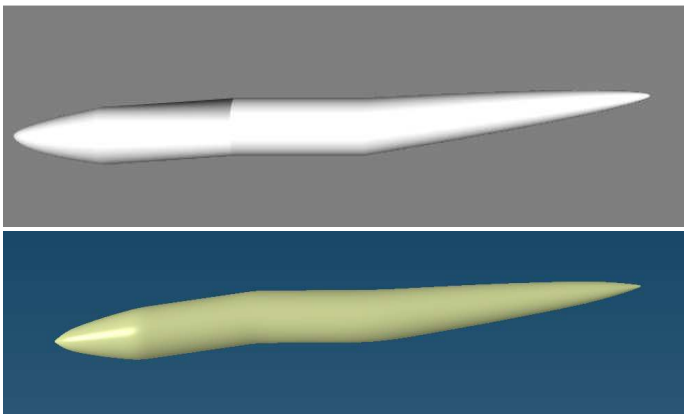

Figura 16 Confronto della geometria della fusoliera

3.2 Ali

Per la definizione delle ali ci si riferirà solo all'ala sinistra, in quanto sarà sfruttata la simmetria rispetto al piano xz.
Come prima cosa vengono definiti i sistemi di riferimento posti in corrispondenza del bordo d'attacco di ogni sezione caratteristica dell'ala. La posizione del sistema di riferimento della sezione di radice è fornita tramite il file XML, generato un punto alle coordinate assegnate, viene definito il sistema di riferimento avente come origine tale punto. Ora è possibile definire l'angolo di diedro, del primo settore dell'ala, e l'angolo di calettamento, assegnando una rotazione opportuna degli assi. In CATIA questa si esprime mediante l'assegnazione dei rapporti: $\frac{x}{x'}$ $\frac{y}{x'}$ .... , cioè i valori del coseno e seno dell'angolo di rotazione.
A questo punto si utilizza uno "*schizzo*", definito nel piano xy del nuovo sistema di riferimento, per tracciare un segmento a cui vengono assegnate la lunghezza e l'angolo, così da definire il bordo d'attacco del primo settore e imporre l'angolo di freccia; nella definizione della lunghezza va considerato l'effetto dell'angolo

diedro, considerando cioè l'apertura come la distanza dal piano di simmetria del velivolo.
Con lo stesso procedimento vengono rappresentati i rimanenti settori e la winglet, il risultato è visualizzato nella figura seguente.

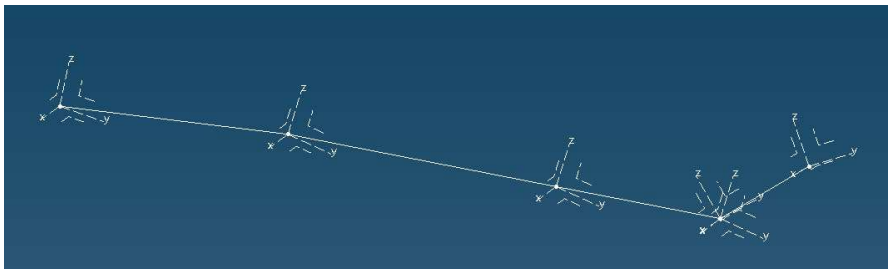

Figura 17 Sistemi di riferimento e bordo d'attacco dell'ala

Creando uno "*schizzo*" per ogni piano xz dei sistemi di riferimento di ciascun settore è possibile passare alla modellazione dei profili alari; questi vengono approssimati con 24 punti, ricavati, partendo dai file .DAT disponibili nel database di AcBuilder, mediante la funzione MatLab *coord_prof.m,* che interpola i punti del file, restituendo due vettori con le coordinate dei punti di interesse. Lo "*schizzo*" è costituito da 24 punti posti nel quadrante con z>0 e x<0, questo perché CATIA permette di modificare il valore assoluto del vincolo e non il segno. Si può procedere quindi all'assegnazione delle coordinate, facendo attenzione che abbiano valori >0, per questo in z vengono maggiorate di 0,4m. I punti ottenuti vengono utilizzati per creare una spline che approssimi il contorno del profilo, questa non viene chiusa per evitare problemi al momento di generare la superficie alare, visto che il bordo di uscita presenta in genere una cuspide.
La curva creata viene infine traslata, in modo da riportare il bordo d'attacco a z=0, e scalata così da ottenere un profilo dalla corda desiderata.

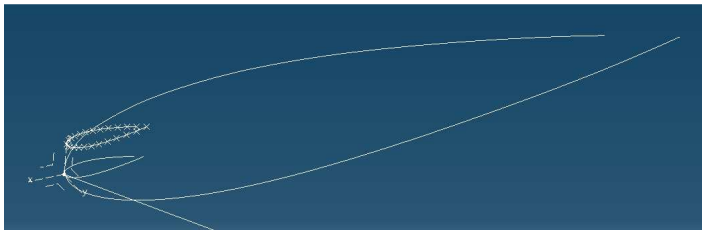

Figura 18 Generazione del profilo alare

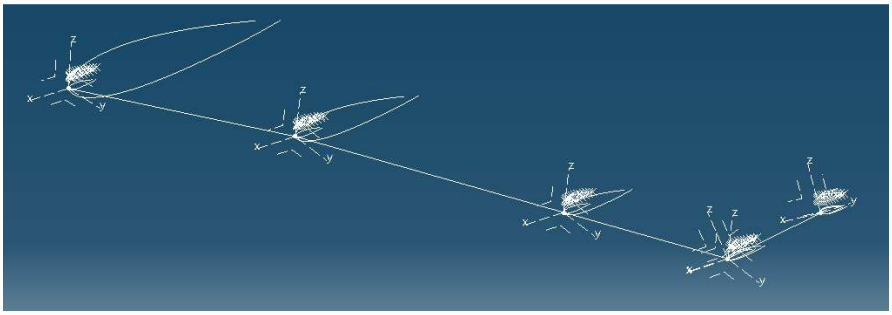

Figura 19 Generazione dei profili alari per ogni sezione

Il profilo di estremità dell'ala viene ruotato in modo da essere in asse con la winglet, questo accorgimento nasce dalla difficoltà di gestire gli spigoli nella creazione delle superfici, un raccordo più accurato può essere eseguito manualmente alla fine del processo di modellazione automatica.
Le superfici vengono generate con il comando "*connessione*", utilizzando come estremi le *spline* scalate, il risultato è una superficie aperta, per chiudere la quale è sufficiente utilizzare nuovamente il comando "*connessione*", ponendo come estremi i bordi della superficie aperta ed imponendo il vincolo di tangenza nell'estremità sul dorso del profilo.
A questo punto si possono definire le superfici mobili mediante dei tagli applicati alla superficie alare. Questi sono definiti mediante uno "*schizzo*" nel piano xy del settore alare, in cui è tracciato il contorno da cui sarà estrusa la superficie che verrà usata per effettuare il taglio. Nella fig. 20 il contorno voluto è evidenziato in verde e si possono vedere le 5 condizioni di vincolo imposte.

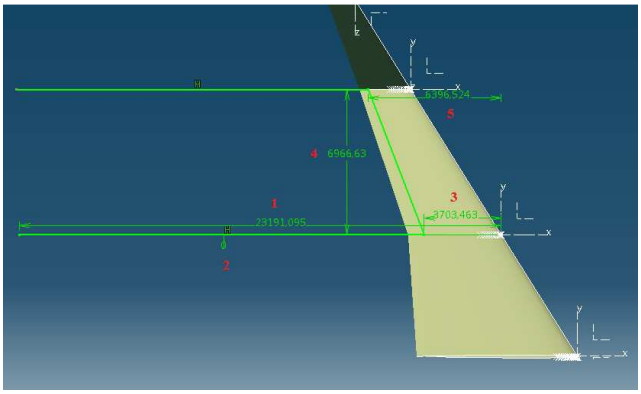

Figura 20 Generazione superficie mobile

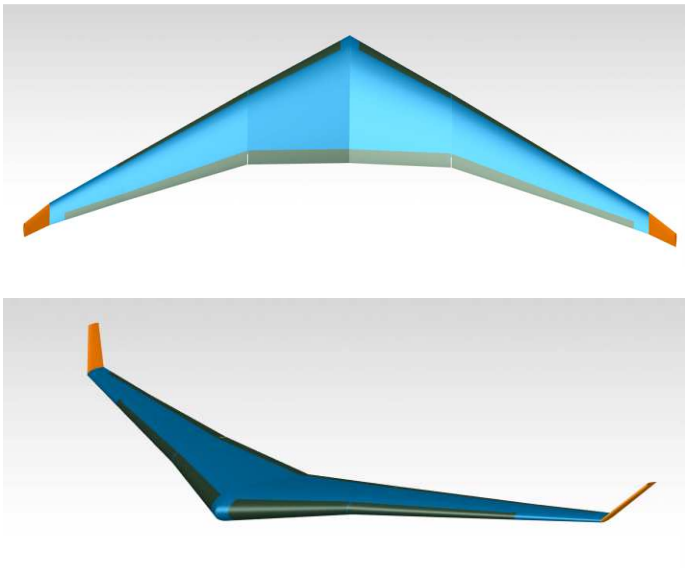

Figura 21 Due visualizzazioni delle ali

3.3 Gondole motore

Tramite le coordinate XYZ è possibile posizionare un punto corrispondente al centro della sezione frontale della gondola motore, utilizzando questo come origine viene definito un sistema di riferimento a cui vengono applicate le rotazioni rispetto all'asse Z e all'asse Y, che costituiscono rispettivamente gli angoli di toe e pitch.
Per creare la curva che sarà usata per la definizione della superficie della gondola si utilizzerà lo stesso metodo illustrato in precedenza, questa volta i punti utilizzati saranno 12, che sono sufficienti per una buona rappresentazione. Le coordinate dei punti sono ottenute mediante le seguenti funzioni:

$$x = \zeta_{lgt}\, l_{eng}\, e^{\frac{\pi}{2}t} \sin\left(\frac{\pi}{2}t\right)$$

$$z = \zeta_{dia}\, d_{eng} \left(1 + e^{\frac{\pi}{2}t}\, cos\left(\frac{\pi}{2}t\right)\right)$$

Dove il parametro t= i/n per i=0,1,2… serve a fornire la separazione tra le coppie di coordinate, l_{eng} e d_{eng} sono la lunghezza del motore e il diametro

massimo. I fattori di scala sono ricavati dalle proprietà delle funzioni trigonometriche e sono espressi da:

$$\zeta_{lgt} = e^{-\frac{\pi}{2}}$$

$$\zeta_{dia} = \frac{1}{2\left(1 + \frac{e^{\frac{\pi}{4}}\sqrt{2}}{2}\right)}$$

La superficie viene realizzata con il comando "*rivoluzione*".

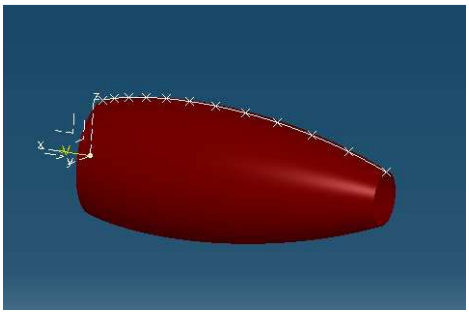

Figura 22 Gondola motore con evidenziato il profilo

3.4 Fairing

Per la modellazione di questo componente ci si discosta dall'approssimazione grossolana presente in AcBuilder, rifacendosi a quanto presentato in [1], in modo da ottenere una rappresentazione più veritiera e flessibile.
Il fairing viene suddiviso in parte anteriore e posteriore, di queste sono individuate 8 sezioni. La parte centrale della carenatura nascerà dal collegamento delle due estremità.
Al centro di ogni sezione è definito un sistema di riferimento nel cui piano yz vengono definiti 7 punti da cui si ricaverà la spline per il contorno della sezione.

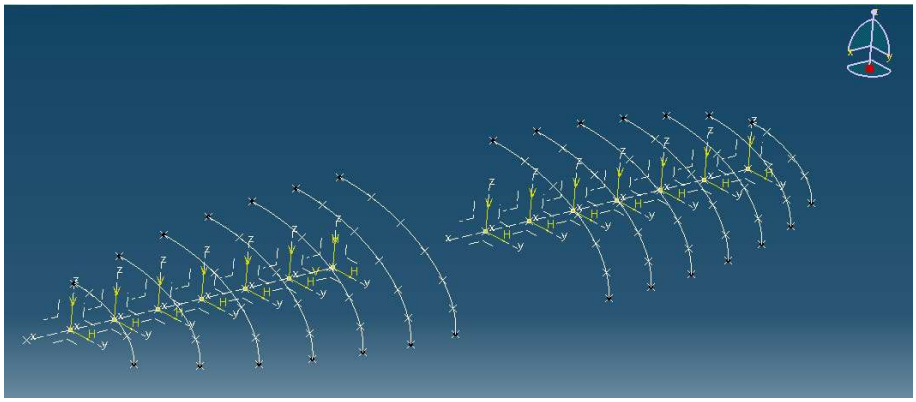

Figura 23 Sezioni fairing

Le coordinate dei sette punti dai quali verrà estrapolata la spline sono ricavate dalle seguenti equazioni:

$$z = |\sin\vartheta|^{\frac{2}{n}} \frac{h}{2} \, sign(\sin\vartheta)$$

$$y = \frac{w}{2}\left(1 - \left(\frac{x}{l_f}\right)^{nx} - \left(\frac{z}{h/_2}\right)^{n}\right)^{\frac{1}{n}}$$

La coordinata Z dei punti sul piano xz è ottenuta da :

$$z = \frac{h}{2}\left(1 - \left(\frac{x}{l_f}\right)^{nx}\right)^{\frac{1}{n}}$$

Modificando gli esponenti “n” e “nx” si può modificare la forma del profilo del super-ellissoide.
Collegando le varie sezioni mediante il comando “connessione” e sfruttando la simmetria dell’elemento si ottiene la superficie totale.
Le due superfici alle estremità sono interpolate, con il vincolo di passare per il vertice e di rispettare la tangenza con le superfici confinanti.

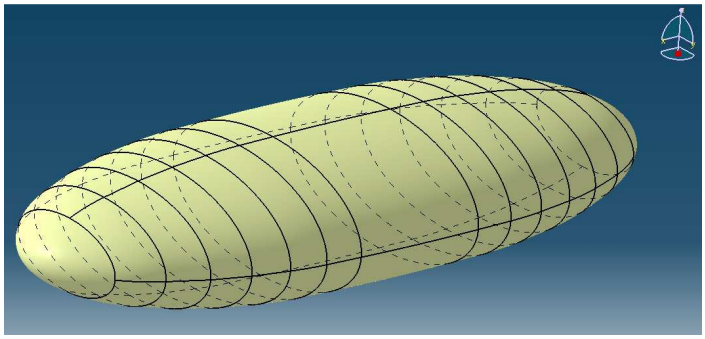

Figura 24 Fairing con n=2 e nx=2

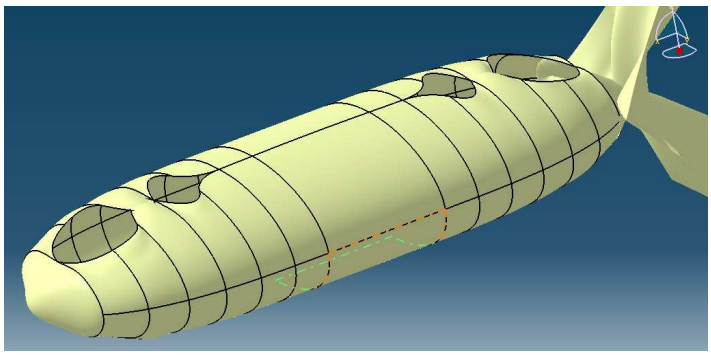

Figura 25 Fairing con n=2 e nx=3

L'immagine in figura 25 mostra che nel passare da nx=2 a nx=3 si modifica il profilo e nella superficie appaiono delle aperture. Questo fenomeno è dovuto al fatto che le spline sono eseguite interpolando solo i punti con coordinate Y reali, quelle immaginarie sono poste uguali a 0. Nella tabella che segue si può vedere che già in corrispondenza della sezione 3 si riscontra un valore di y immaginario che viene quindi posto uguale a 0. L'effetto di questa approssimazione è che nella definizione della spline non verranno utilizzati tutti e 7 i punti, ma solo quelli precedenti il secondo con y nulla.

	Y1	Y2	Y3	Y4	Y5	Y6	Y7
sezione1	1	0,965926	0,866025	0,707107	0,5	0,258819	0
sezione2	0,989743	0,955303	0,854161	0,692526	0,479157	0,215822	0
sezione3	0,958315	0,922703	0,817537	0,646813	0,410326	0	0
sezione4	0,903508	0,865644	0,752547	0,562429	0,257539	0	0
sezione5	0,820652	0,77877	0,650745	0,416497	0	0	0
sezione6	0,699854	0,650237	0,48969	0	0	0	0
sezione7	0,515079	0,44533	0,123718	0	0	0	0

E’ necessario quindi modificare manualmente le splines interessate, aggiungendo, o rimuovendo, i punti in più, o in meno, rispetto alla configurazione precedente. Nella fig. 26 viene mostrata la finestra che permette con poche operazioni di modificare la definizione della spline.

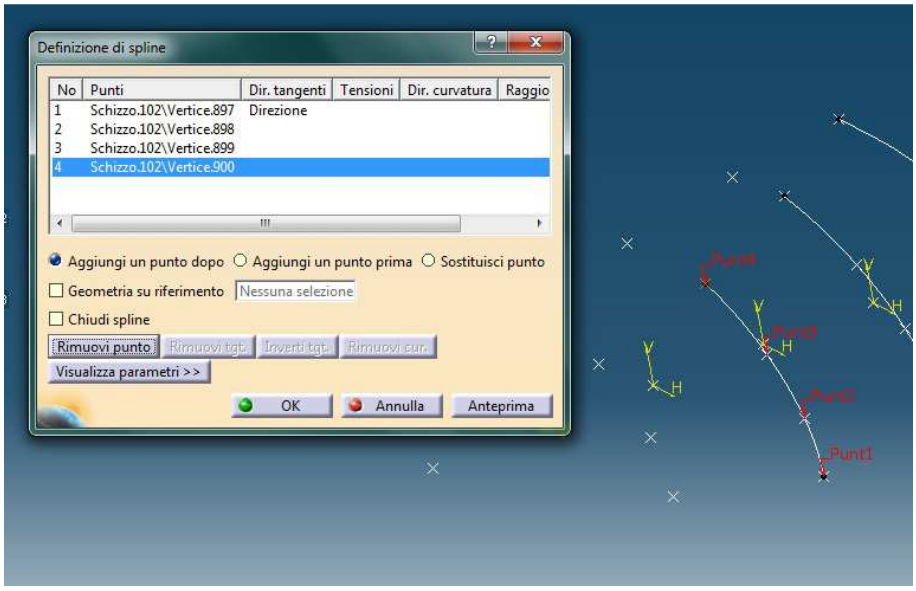

Figura 26 Definizione spline

Può essere necessario modificare le condizioni di tangenza della spline e delle superfici ad essa legate per evitare risultati anomali, come esemplificato in fig. 27.

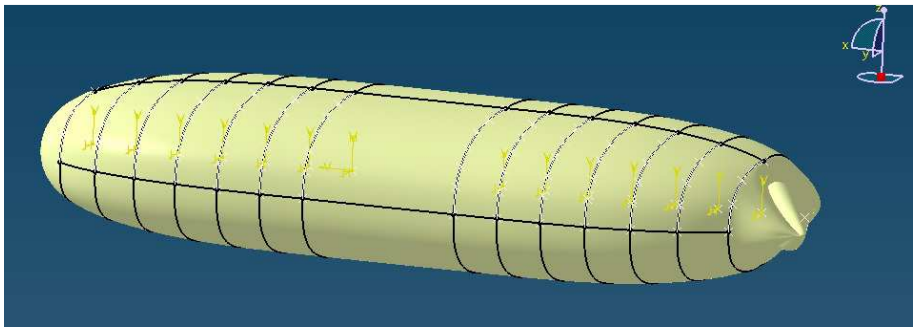

Figura 27 Fairing con n=2.5 e nx =3

Passando da una configurazione all'altra, si possono verificare delle anomalie nella generazione delle superfici, che andranno corrette manualmente in ambiente CATIA, andando a verificare la correttezza delle condizioni di tangenza tra le superfici.

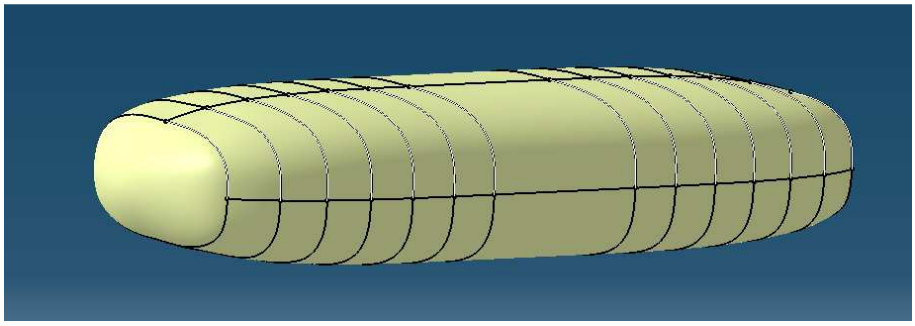

Figura 28 Fairing con n=3 e nx=3

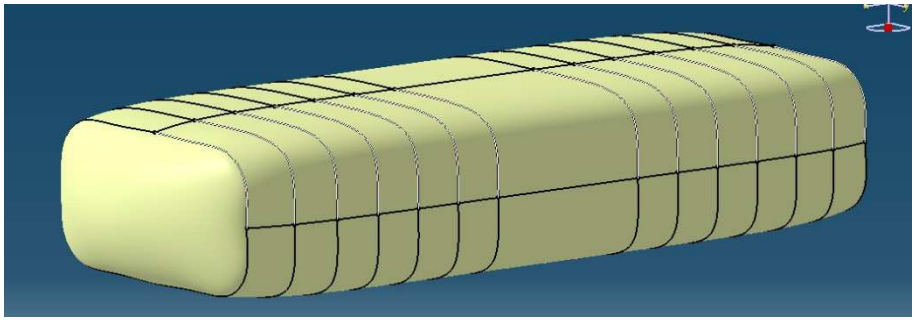

Figura 29 Fairing con n=5 e nx=5

3.5 Impennaggio di coda orizzontale

Il piano orizzontale è realizzato utilizzando lo stesso procedimento visto per le ali, limitando però lo sviluppo a soli due settori. L'equilibratore è ricavato con un unico taglio, visto che usualmente è costituito da un'unica superficie mobile.

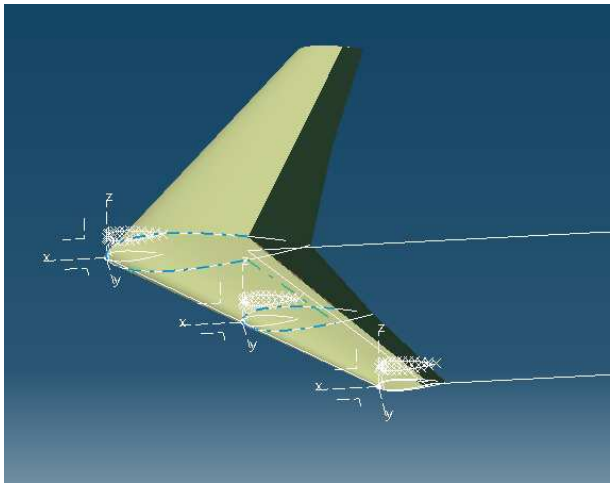

Figura 30 Impennaggio di coda orizzontale

3.6 Impennaggio di coda verticale

La realizzazione del piano verticale avviene con un procedimento analogo a quello seguito per le ali. Per quanto riguarda la definizione del profilo alare in questo caso avendo a che fare con un profilo simmetrico, ci si limiterà a creare la metà dei punti per poi sfruttare la simmetria. L'origine del sistema di riferimento della sezione di radice viene definita da:

$x = L_{fus} - apex_{locale} * L_{fus}$

Dove x rappresenta la distanza dalla coda, la coordinata Z, dato che il riferimento è l'estremità di coda, è posta uguale a 0.

L'imposizione delle dimensioni del timone rispecchia la procedura vista per gli ipersostentatori e come per l'equilibratore è realizzato con un unico taglio. In fig. 31 è possibile vedere gli "schizzi" utilizzati e la superficie risultante.

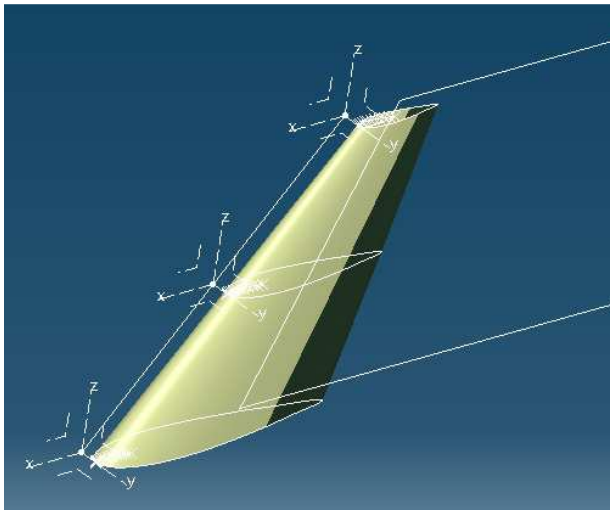

Figura 31 Impennaggio di coda verticale

3.7 Ventral fin

I ventral fin vengono realizzati mediante il comando “connessione” una volta creati i segmenti che rappresentano le corde in radice, mezzeria e estremità. Gli angoli di cant e freccia vengono definiti imponendo gli angoli di rotazione degli assi dei sistemi di riferimento posti on corrispondenza del segmento di radice e di mezzeria. La posizione viene imposta assegnando le coordinate X e Z dell’origine del sistema di riferimento in cui si crea il segmento di radice.

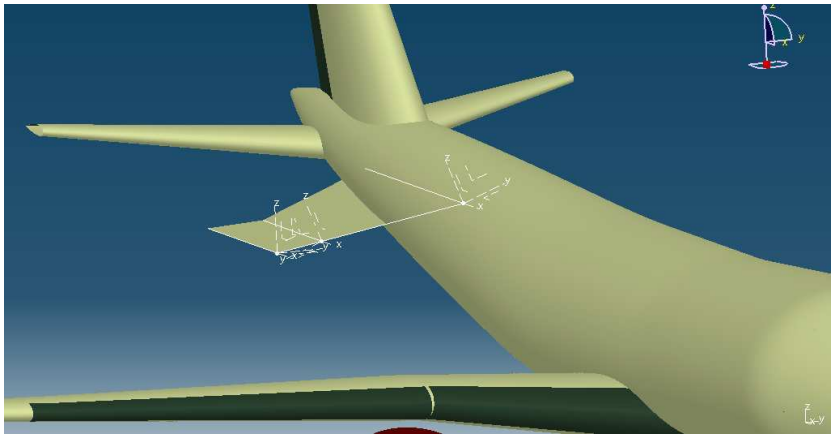

Figura 32 Ventral fin

3.8 Canard

L’ala canard viene realizzata sfruttando gli stessi comandi utilizzati per il piano di coda orizzontale, si può vedere infatti nella figura come siano stati utilizzati gli stessi elementi costruttivi.

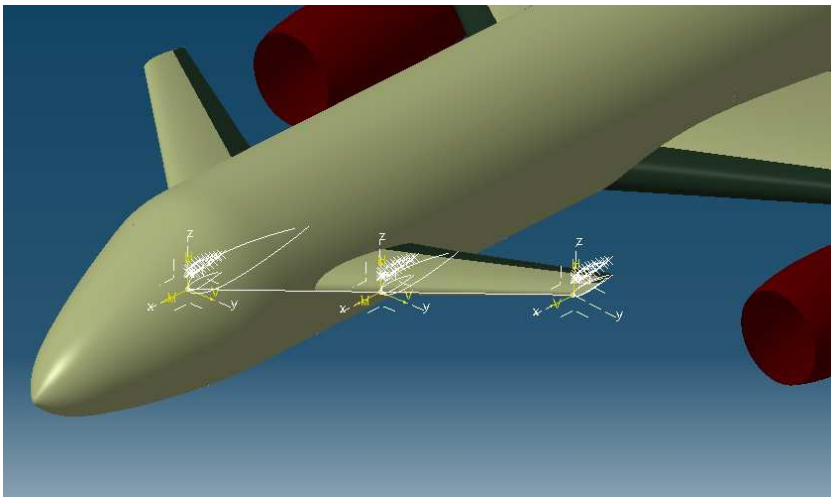

Figura 33 Canard

3.9 Cassone alare

Per completare il modello vengono realizzati anche i longheroni, che permettono di definire la geometria del cassone alare. Dal file XML vengono importate le posizioni dell’asse elastico e le semiampiezze del cassone, in corrispondenza delle sezioni in cui è divisa l’ala. Creati i punti in corrispondenza dell’intersezione tra il piano della sezione alare e l’asse elastico, è possibile definire i punti sul piano di sezione in corrispondenza dei longheroni conoscendo la semiampiezza del cassone. Collegando questi punti si ottiene l’asse del longherone che viene completato mediante estrusione. Ritenendo i due longheroni equidistanti rispetto all’asse elastico è immediato ottenere il longherone posteriore sfruttando la simmetria. Facendo terminare l’estrusione in

corrispondenza della superficie dell'ala si ottiene un'approssimazione del cassone alare.

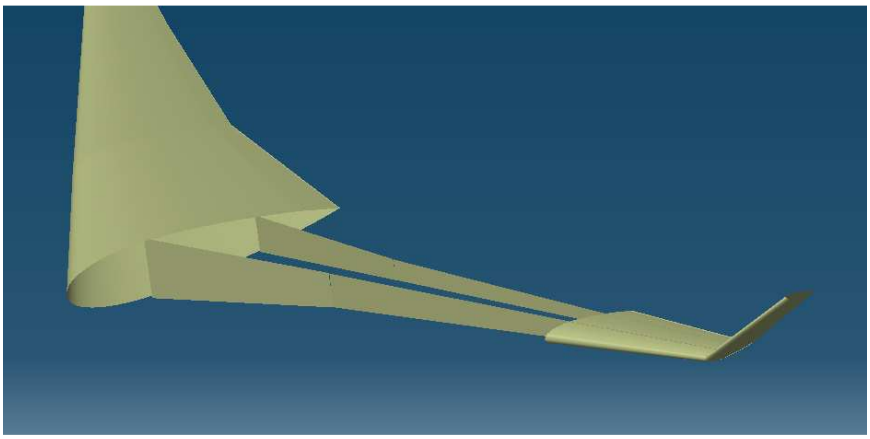

Figura 34 Longheroni

3.10 Struttura del modello

Dato che si vuole rendere il modello il più flessibile possibile, le componenti appena descritte costituiscono ognuna un "body" differente, in modo da poter essere manipolate separatamente. L'aver organizzato in questo modo il modello permette di accedere facilmente al componente di interesse e mediante l'utilizzo del comando "nascondi/visualizza" di visualizzarlo o meno.
Dal file XML è possibile importare anche l'informazione riguardante la presenza o meno di un componente che in CATIA è possibile tradurre mediante l'imposizione del parametro booleano "attività". In questo modo il modello di partenza rimane immutato e non vengono cancellate le componenti non presenti nel modello da rappresentare, ma vengono solo disattivate, quindi non visualizzate. Questa caratteristica permette di poter rappresentare configurazioni notevolmente diverse partendo da un unico modello CATIA.
Per evitare eventuali problemi con l'aggiornamento della configurazione si suggerisce di partire sempre dal modello standard e poi eseguire il programma con la configurazione desiderata.

Figura 35 Albero logico

3.11 Esempi di configurazione

Vengono ora presentate delle immagini relative ai modelli tridimensionali ottenuti in Catia al termine della procedura illustrata nei paragrafi precedenti. Con questi esempi si vuole rappresentare la versatilità della procedura illustrata in questa tesi.

Figura 36 Esempio 1

Figura 37 Esempio 2

Figura 38 Esempio 3

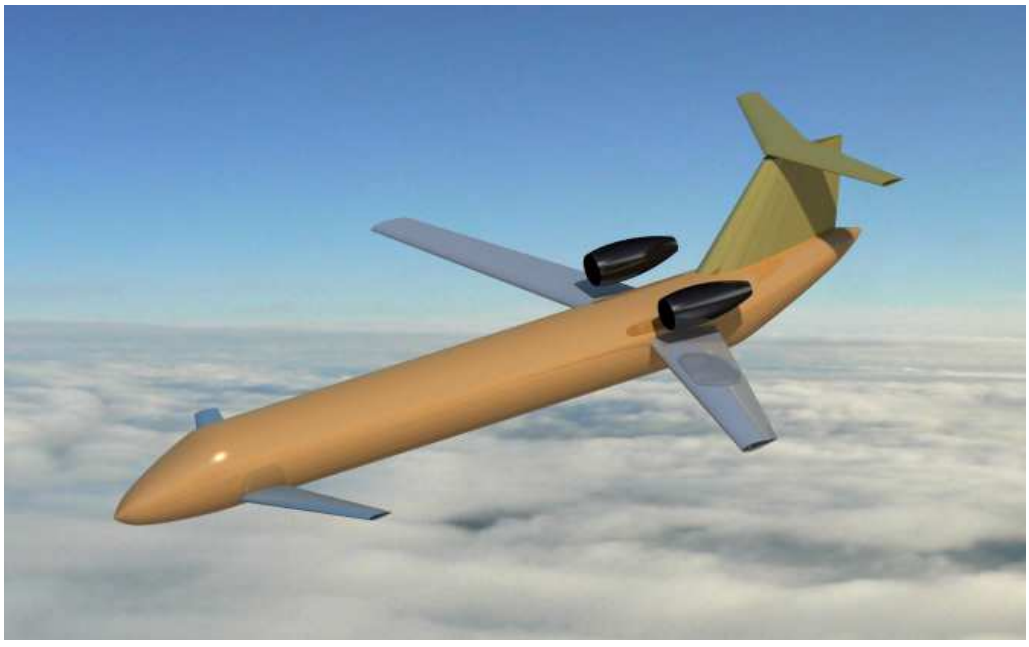

Figura 39 Esempio 4

4 Confronto

In questo capitolo verranno messi a confronto i risultati ottenenti con la procedura sopra descritta con la rappresentazione grafica fornita da AcBuilder. Facendo ciò, oltre ad evidenziare le migliori qualità estetiche del modello CAD, è possibile verificare che durante la realizzazione non siano state omesse delle componenti.

Il procedimento euristico che si intende seguire è l'accostamento di immagini riportanti la stessa configurazione e ricercare eventuali anomalie.

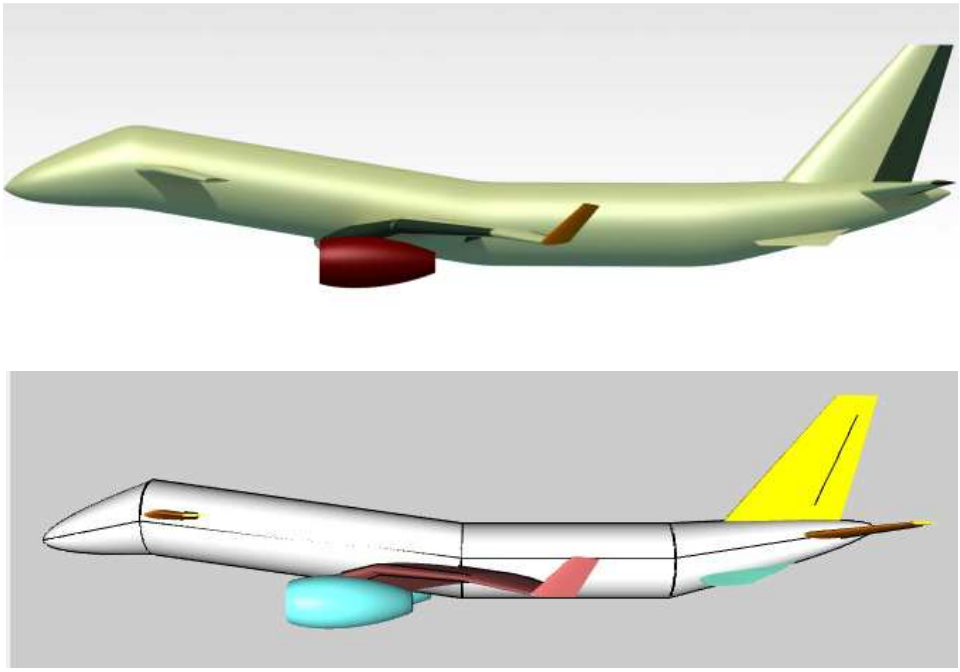

Figura 40 Confronto vista laterale

Dalla fig. 36 come il modello CAD rispecchi la geometria prevista, si può notare in più come si sia migliorata la qualità del modello avendo eliminato le discontinuità in corrispondenza dei cambi di sezione della fusoliera. Si fa notare che la versione di AcBuilder sia visualizzata in maniera prospettica, presentando quindi inclinazioni differenti delle superfici.

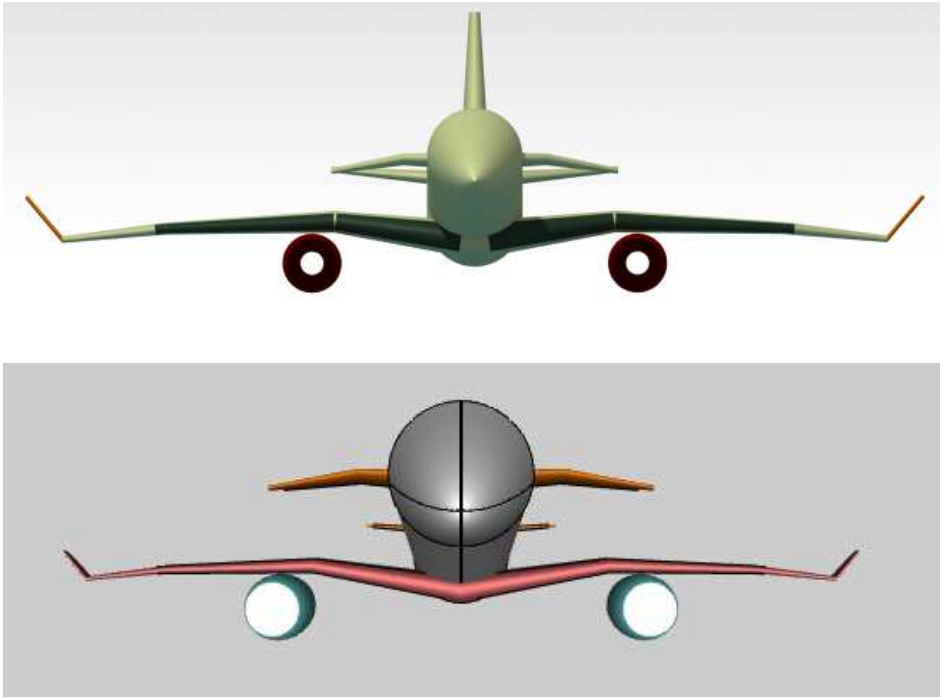

Figura 41 Confronto vista frontale

Nella vista frontale la distorsione prospettica è maggiore di quella vista in figura 36, però permette di verificare la correttezza degli angoli di diedro dei diversi segmenti in cui sono suddivise le ali e il canard.

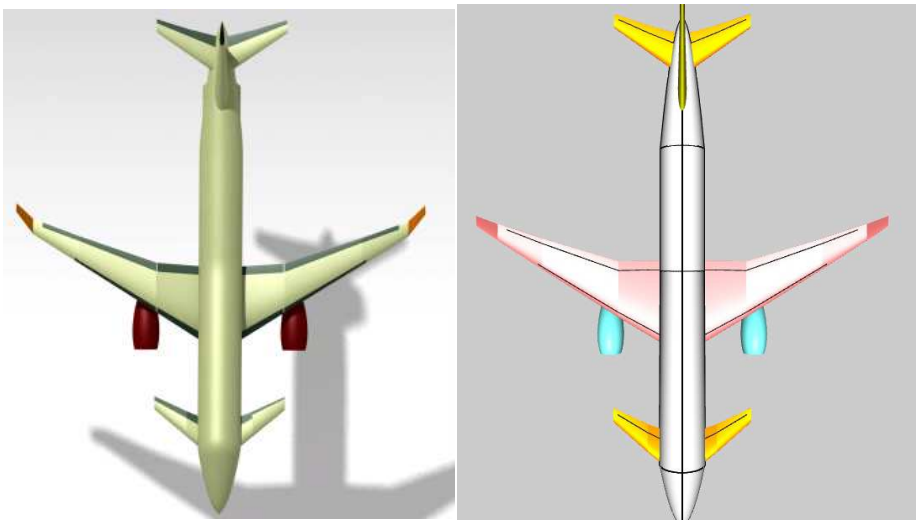

Figura 42 Confronto vista dall'alto

La fig. 38 completa il quadro delle viste di confronto, permettendo di verificare che le geometrie delle superfici alari siano corrette. Superato questo esame si ritiene di poter affermare che la modellazione proposta sia in grado di fornire una corretta rappresentazione a superfici del velivolo ottimizzato nel corso delle analisi dei diversi pacchetti contenuti in CEASIOM.

5 Conclusioni

Al termine di questo lavoro si è ottenuta una procedura in grado di fornire un modello CAD di un velivolo di configurazione generica mediante l'esecuzione di pochi comandi. Il modello è costituito dalle superfici del velivolo, che vengono predisposte per un successivo perfezionamento manuale, per esempio fornendo i tagli in corrispondenza delle superfici mobili.

Come sviluppi futuri è possibile proporre una migliore integrazione del fairing con il resto del modello, visto che per la modellazione attuale si è fatto riferimento a possibili migliorie del programma AcBuilder, non ancora implementate nella versione aggiornata al 22/10/2010.

Appendice A: parametri d’ingresso del file XML

Fuselage
- Forefuse_X_sect_vertical_diameter
- Forefuse_Xs_distortion_coefficient
- Forefuse_X_sect_horizontal_diameter
- omega_nose
- phi_nose
- epsilon_nose
- shift_fore
- fraction_fore
- Total_fuselage_length
- Aftfuse_X_sect_vertical_diameter
- Aftfuse_Xs_distortion_coefficient
- Aftfuse_X_sect_horizontal_diameter
- omega_tail
- phi_tail
- epsilon_tail

Wing *(both Wing1 and Wing2)*
- configuration
- placement
- apex_locale
- area
- AR
- Span
- spanwise_kink1
- spanwise_kink2
- taper_kink1
- taper_kink2
- taper_tip
- root_incidence
- kink1_incidence
- kink2_incidence
- tip_incidence
- quarter_chord_sweep_inboard
- quarter_chord_sweep_midboard
- quarter_chord_sweep_outboard
- LE_sweep_inboard
- LE_sweep_midboard
- LE_sweep_outboard
- dihedral_inboard
- dihedral_midboard
- dihedral_outboard
- thickness_root
- thickness_kink1

- thickness_kink2
- thickness_tip
- winglet
  - Span
  - taper_ratio
  - LE_sweep
  - Cant_angle
  - root_incidence
  - tip_incidence
- aileron
  - position
  - chord
  - Span
- flap
  - root_chord
  - kink1_chord
  - kink2_chord
- slat
  - chord
  - root_position
  - tip_position
- root_airfoil
- kink1_airfoil
- kink2_airfoil
- tip_airfoil

Fairing *(both Fairing 1 and Fairing 2)*
- Fore_length
- Aft_length
- Mid_length
- Width
- Height
- n_exp
- nx_exp

Horizontal_tail
- empennage_layout
- area
- AR
- Span
- spanwise_kink
- taper_kink
- taper_tip
- root_incidence
- kink_incidence
- tip_incidence
- quarter_chord_sweep_inboard
- quarter_chord_sweep_outboard
- LE_sweep_inboard
- LE_sweep_outboard
- dihedral_inboard

- dihedral_outboard
- vertical_locale
- apex_locale
- thickness_root
- thickness_kink
- thickness_tip
- Elevator
  - chord
  - Span
- root_airfoil
- kink_airfoil
- tip_airfoil

Vertical_tail
- area
- AR
- Span
- spanwise_kink
- taper_kink
- taper_tip
- quarter_chord_sweep_inboard
- quarter_chord_sweep_outboard
- LE_sweep_inboard
- LE_sweep_outboard
- vertical_locale
- apex_locale
- thickness_root
- thickness_kink
- thickness_tip
- dihedral_inboard
- dihedral_outboard
- root_incidence
- kink_incidence
- tip_incidence
- Rudder
  - chord
  - Span
- root_airfoil
- kink_airfoil
- tip_airfoil

Ventral_fin
- chord_fraction_at_midfuse
- Span
- spanwise_kink
- taper_kink
- taper_tip
- LE_sweep_inboard
- LE_sweep_outboard
- cant_inbord
- cant_outboard

X_locale
Z_locale

Engines *(both Engines1 and Engines2)*
Number_of_engines
Layout_and_config
Propulsion_type
Y_locale
X_locale
Z_locale
toe_in
pitch
Nacelle_body_type
fineness_ratio
d_max
Propeller_diameter
Max_thrust
Bypass_ratio_to_emulate
Thrust_reverser_effectivness
Thrust_to_weight_ratio

fuel
box1_ea_loc_root
box1_ea_loc_kink1
box1_ea_loc_kink2
box1_ea_loc_tip
box1_semispan_root
box1_semispan_kink1
box1_semispan_kink2
box1_semispan_tip
Wing1_fuel_tank_cutout_opt
Outboard_fuel_tank1_span
Unusable_fuel_option
Assumed_fuel_density
Centre_tank1_portion_used
box2_ea_loc_root
box2_ea_loc_kink1
box2_ea_loc_kink2
box2_ea_loc_tip
box2_semispan_root
box2_semispan_kink1
box2_semispan_kink2
box2_semispan_tip
Wing2_fuel_tank_cutout_opt
Outboard_fuel_tank2_span
Centre_tank_portion_used
Fore_fairing1_tank_length
Aft_fairing1_tank_length
Fore_fairing2_tank_length
Aft_fairing2_tank_length
Aft_fuse_bladder_length

Baggage
- installation_type
- gross_volume
- Baggage_combined_length
- Baggage_apex_per_fuselgt

cabin
- Cabin_length_to_aft_cab
- Cabin_max_internal_height
- Cabin_max_internal_width
- Cabin_floor_width
- Cabin_volume
- Passenger_accomodation
- Seats_abreast_in_fuselage
- Maximum_cabin_altitude
- Cabin_attendant_number
- Flight_crew_number

miscellaneous
- Design_classification
- Target_operating_ceiling
- Spoiler_effectivity
- Undercarriage_layout
- main_landing_gear_x_cg
- main_landing_gear_y_cg
- main_landing_gear_z_cg
- aux_landing_gear_x_cg
- aux_landing_gear_z_cg

weight_balance
- Ramp_increment
- Weight_cont_allow_perc_of_MEW
- Manufacturer_weights_tolerance

Appendice B: Estratto dal listato Matlab

Funzione per il calcolo delle coordinate dei punti necessari per ottenere il profilo alare.

```
function [Y,X] = coord_prof (profilo)
 nome_file = [profilo];
 K = load (nome_file);
 n = length (K(:,1));
 a = [pi/2:-pi/25:0];
 a1 = [0:pi/25:pi/2];
 xi = 1 - cos (a);
 xi2 = 1 - sin (a1);
  Q = size(K,1);
  x = K(:,1);
  y = K(:,2);
   if mod(Q,2) == 0
        x_dorso=x(1:ceil(Q/2));
        y_dorso=y(1:ceil(Q/2));
        x_ventre=x(ceil((Q/2)+1):end);
        y_ventre=y(ceil((Q/2)+1):end);
   else
        x_dorso=x(1:ceil(Q/2));
        y_dorso=y(1:ceil(Q/2));
        x_ventre=x(ceil((Q/2)):end);
        y_ventre=y(ceil((Q/2)):end);
   end
 Y1 = interp1(x_ventre,y_ventre,xi);
 Y2 = interp1(x_dorso,y_dorso,xi2(2:end-1));
 Y=[];
   for i=1:length(Y1)
       Y(i)=Y1(i);
   end
 g=1;
   for j = length(Y1)+1:2*(length(Y1))-2
       Y(j)=Y2(g);
       g=g+1;
   end
 X=[];
   for i=1:length(Y1)
       X(i)=xi(i);
   end
 h=1;
   for j = length(Y1)+1:2*(length(Y1))-2
       X(j)=xi2(h+1);
       h=h+1;
   end
```

Programma per ottenere la generazione del velivolo in Catia V5.

```
% IMPORTAZIONE PARAMETRI

clear all; close all;
filename = uigetfile('*.xml');
xmlstr = fileread(filename);
V = xml_parseany(xmlstr);

%% FUSOLIERA

omega_nose = str2double (V.Fuselage{1,1}.omega_nose{1,1}.CONTENT);
phi_nose = str2double (V.Fuselage{1,1}.phi_nose{1,1}.CONTENT);
epsilon_nose = str2double (V.Fuselage{1,1}.epsilon_nose{1,1}.CONTENT);
Forefuse_X_sect_vertical_diameter = str2double
(V.Fuselage{1,1}.Forefuse_X_sect_vertical_diameter{1,1}.CONTENT);
Forefuse_X_sect_horizontal_diameter = str2double
(V.Fuselage{1,1}.Forefuse_X_sect_horizontal_diameter{1,1}.CONTENT);
Forefuse_Xs_distortion_coefficient = str2double
(V.Fuselage{1,1}.Forefuse_Xs_distortion_coefficient{1,1}.CONTENT);
Nose_length = str2double (V.Fuselage{1,1}.Nose_length{1,1}.CONTENT);
fraction_fore = str2double
(V.Fuselage{1,1}.fraction_fore{1,1}.CONTENT);
shift_fore = str2double (V.Fuselage{1,1}.shift_fore{1,1}.CONTENT);
omega_tail = str2double (V.Fuselage{1,1}.omega_tail{1,1}.CONTENT);
phi_tail = str2double (V.Fuselage{1,1}.phi_tail{1,1}.CONTENT);
epsilon_tail = str2double (V.Fuselage{1,1}.epsilon_tail{1,1}.CONTENT);
Aftfuse_X_sect_vertical_diameter = str2double
(V.Fuselage{1,1}.Aftfuse_X_sect_vertical_diameter{1,1}.CONTENT);
Aftfuse_X_sect_horizontal_diameter = str2double
(V.Fuselage{1,1}.Aftfuse_X_sect_horizontal_diameter{1,1}.CONTENT);
Aftfuse_Xs_distortion_coefficient = str2double
(V.Fuselage{1,1}.Aftfuse_Xs_distortion_coefficient{1,1}.CONTENT);
Tail_length = str2double (V.Fuselage{1,1}.Tail_length{1,1}.CONTENT);
Total_fuselage_length = str2double
(V.Fuselage{1,1}.Total_fuselage_length{1,1}.CONTENT);
a0_fore = str2double (V.Fuselage{1,1}.a0_fore{1,1}.CONTENT);
a1_fore = str2double (V.Fuselage{1,1}.a1_fore{1,1}.CONTENT);
b1_fore = str2double (V.Fuselage{1,1}.b1_fore{1,1}.CONTENT);
a0_aft = str2double (V.Fuselage{1,1}.a0_aft{1,1}.CONTENT);
a1_aft = str2double (V.Fuselage{1,1}.a1_aft{1,1}.CONTENT);
b1_aft = str2double (V.Fuselage{1,1}.b1_aft{1,1}.CONTENT);

asse_orizzontale = Forefuse_X_sect_horizontal_diameter;
asse_verticale = Forefuse_X_sect_vertical_diameter;
distorsion_coeff = Forefuse_Xs_distortion_coefficient;
a_0 = a0_fore;  a_1 = a1_fore;  b_1 = b1_fore;
asse_orizzontale_2 = asse_orizzontale;
asse_verticale_2 = asse_verticale;
distorsion_coeff_2 = distorsion_coeff;
asse_orizzontale_3 = Aftfuse_X_sect_horizontal_diameter ;
asse_verticale_3 = Aftfuse_X_sect_vertical_diameter;
distorsion_coeff_3 = Aftfuse_Xs_distortion_coefficient;
```

```
phi = [0:pi/12:2*pi];
for i =1:length(phi)
    r(i) = a_0+a_1*sin(phi(i))+b_1*cos(2*phi(i));
    y(i) = r(i)*cos(phi(i));
    z(i) = r(i)*sin(phi(i))-
asse_verticale/2+distorsion_coeff*asse_verticale+10+shift_fore*Aftfuse_
X_sect_vertical_diameter;
end
    a_0_3 = a0_aft;    a_1_3 = a1_aft;    b_1_3 = b1_aft;
    delta_3 = -shift_fore*Aftfuse_X_sect_vertical_diameter;
    phi_3 = [0:pi/12:2*pi];
        for i =1:length(phi_3)
            r_3(i) = a_0_3+a_1_3*sin(phi_3(i))+b_1_3*cos(2*phi_3(i));
            y_3(i) = r_3(i)*cos(phi_3(i));
            z_3(i) = r_3(i)*sin(phi_3(i))-
asse_verticale_3/2+distorsion_coeff_3*asse_verticale_3+10;
        end

command = [];
g = 1; h = 1;
for j = 1:2:13
    command(j) = y(g);
    g=g+1;
end
for k = 2:2:14
    command(k) = z(h);
    h=h+1;
end
g = 19; h = 19;
for j = 15:2:25
    command(j) = y(g);
    g=g+1;
end
for k = 16:2:26
    command(k) = z(h);
    h=h+1;
end


%NASO

D_v =2*((-asse_verticale/2)+distorsion_coeff*asse_verticale);
epsilon = epsilon_nose;
l_nose = Nose_length;
psi = phi_nose; psi_rad = (pi*psi)/180;
z_nose =
l_nose*tan(psi_rad)+shift_fore*Aftfuse_X_sect_vertical_diameter;
omega = omega_nose; omega_rad = (pi*omega)/180;
beta = 0.54+0.1*tan(omega_rad-psi_rad);
Z_1 = z(7);
Z_2 = z(19);
Y_1 = -1*(-0.5*(asse_verticale^(1-beta))*((l_nose/epsilon)^beta))-
(epsilon*asse_verticale-
l_nose)*tan(psi_rad)+D_v+10+shift_fore*asse_verticale;
Y_2 = -1*(0.5*(asse_verticale^(1-beta))*((l_nose/epsilon)^beta))-
(epsilon*asse_verticale-
l_nose)*tan(psi_rad)+D_v+10+shift_fore*asse_verticale;
```

```
Dvup = Z_1-Y_1;
Dvdown= Z_2-Y_2;
y_down = [0:l_nose/6:l_nose];
for i=1:length(y_down)
    z_down(i) = (-1*(0.5*(asse_verticale^(1-
beta))*((y_down(i)/epsilon)^beta))-(epsilon*asse_verticale-
y_down(i))*tan(psi_rad)+D_v+10+y_down(i)*Dvdown/l_nose)+shift_fore*asse
_verticale;
    z_up(i) = (-1*(-0.5*(asse_verticale^(1-
beta))*((y_down(i)/epsilon)^beta))-(epsilon*asse_verticale-
y_down(i))*tan(psi_rad)+D_v+10+y_down(i)*Dvup/l_nose)+shift_fore*asse_v
erticale;
end
z_down(1) = z_down(1)*0.99;
z_up(1) = z_up(1)*1.01;
r_nose = (z_up(1)-z_down(1))/1.95;
g=6+1; h=1;
for i = 27:2:39
    command(i) = y_down(g);
    g=g-1;
end
for i = 28:2:40
    command(i) = z_down(h);
    h=h+1;
end
g=6+1; h=1;
for i = 41:2:53
    command(i) = y_down(g);
    g=g-1;
end
for i = 42:2:54
    command(i) = z_up(h);
    h=h+1;
end
command(55) = r_nose;

%CORPO CENTRALE

l_main = Total_fuselage_length-Nose_length-Tail_length;
command(56) = l_main;

%CODA

D_v_c =2*((-asse_verticale_3/2)+distorsion_coeff_3*asse_verticale_3);
epsilon_c = epsilon_tail;
l_coda = Tail_length;
omega_c = omega_tail; omega_c_rad = pi*omega_c/180;
psi_c = phi_tail; psi_c_rad = pi*psi_c/180;
beta_c = 0.54+0.1*tan(omega_c_rad-psi_c_rad);
Z_1_c = z_3(7);
Z_2_c = z_3(19);
Y_1_c = -1*(-0.5*(asse_verticale_3^(1-
beta_c))*((l_coda/epsilon_c)^beta_c)-(epsilon_c*asse_verticale_3-
l_coda)*tan(psi_c_rad))+D_v_c+10;
```

```
Y_2_c = -1*(0.5*(asse_verticale_3^(1-
beta_c))*((l_coda/epsilon_c)^beta_c)-(epsilon_c*asse_verticale_3-
l_coda)*tan(psi_c_rad))+D_v_c+10;
Dvup_c = Z_1_c-Y_1_c;
Dvdown_c= Z_2_c-Y_2_c;
y_c = [0:l_coda/6:l_coda];
for i=1:length(y_c)
    z_down_c(i) = (-1*(0.5*(asse_verticale_3^(1-
beta_c))*((y_c(i)/epsilon_c)^beta_c)-(epsilon_c*asse_verticale_3-
y_c(i))*tan(psi_c_rad))+D_v_c+10+y_c(i)*Dvdown_c/l_coda);
    z_up_c(i) = (-1*(-0.5*(asse_verticale_3^(1-
beta_c))*((y_c(i)/epsilon_c)^beta_c)-(epsilon_c*asse_verticale_3-
y_c(i))*tan(psi_c_rad))+D_v_c+10+y_c(i)*Dvup_c/l_coda);
end
z_down_c(1) = z_down_c(1)*0.99;
z_up_c(1) = z_up_c(1)*1.01;
g=6+1; h=1;
for i = 57:2:69
    command(i) = y_c(g)+l_main;
    g=g-1;
end
for i = 58:2:70
    command(i) = z_down_c(h);
    h=h+1;
end
g=6+1; h=1;
for i = 71:2:83
    command(i) = y_c(g)+l_main;
    g=g-1;
end
for i = 72:2:84
    command(i) = z_up_c(h);
    h=h+1;
end
r_c = (z_up_c(1)-z_down_c(1))/1.95;
command(85) = r_c;
command(86) = -2*y_down(2);
command(87) = 2*y_c(2);

%SECONDA SEZIONE FUSOLIERA

y_2 = y_3;
z_2 = z_3;
g = 1; h = 1;
   k=length(command)+1;
        for n = k:2:k+10
            command(n) = y_2(g);
            g=g+1;
        end
        for n = k+1:2:k+11
            command(n) = z_2(h);
            h=h+1;
        end
   k=length(command)+1;
   g = 19; h = 19;
        for n = k:2:k+10
            command(n) = y_2(g);
            g=g+1;
```

```
        end
        for k = k+1:2:k+11
            command(k) = z_2(h);
            h=h+1;
        end
   n=length(command)+1;
   command(n)=y_2(7);n=n+1;
   command(n)=z_2(7);

%TERZA SEZIONE FUSOLIERA

   g = 1; h = 1;
   k=length(command)+1;
        for n = k:2:k+10
            command(n) = y_3(g);
            g=g+1;
        end
        for n = k+1:2:k+11
            command(n) = z_3(h);
            h=h+1;
        end
   k=length(command)+1;
   g = 19; h = 19;
        for n = k:2:k+10
            command(n) = y_3(g);
            g=g+1;
        end
        for k = k+1:2:k+11
            command(k) = z_3(h);
            h=h+1;
        end
   n=length(command)+1;
   command(n)=y_3(7);n=n+1;
   command(n)=z_3(7); n=n+1;
   command(n) = -l_main; n = n+1;
   command(n) = -fraction_fore*l_main;

 clear fusoliera.xls
 N = length(command);
 x = 1:1:N;
 q = [x; command];
 q = q';
 xlswrite('fusoliera',q);


%% ALA

area = str2double (V.Wing1{1,1}.area{1,1}.CONTENT);
Span = str2double (V.Wing1{1,1}.Span{1,1}.CONTENT);
AR = str2double (V.Wing1{1,1}.AR{1,1}.CONTENT);
spanwise_kink1 = str2double (V.Wing1{1,1}.spanwise_kink1{1,1}.CONTENT);
spanwise_kink2 = str2double (V.Wing1{1,1}.spanwise_kink2{1,1}.CONTENT);
taper_kink1 = str2double (V.Wing1{1,1}.taper_kink1{1,1}.CONTENT);
taper_kink2 = str2double (V.Wing1{1,1}.taper_kink2{1,1}.CONTENT);
taper_tip = str2double (V.Wing1{1,1}.taper_tip{1,1}.CONTENT);
root_incidence = str2double (V.Wing1{1,1}.root_incidence{1,1}.CONTENT);
```

```
kink1_incidence = str2double
(V.Wing1{1,1}.kink1_incidence{1,1}.CONTENT);
kink2_incidence = str2double
(V.Wing1{1,1}.kink2_incidence{1,1}.CONTENT);
tip_incidence = str2double (V.Wing1{1,1}.tip_incidence{1,1}.CONTENT);
LE_sweep_inboard = str2double
(V.Wing1{1,1}.LE_sweep_inboard{1,1}.CONTENT);
LE_sweep_midboard = str2double
(V.Wing1{1,1}.LE_sweep_midboard{1,1}.CONTENT);
LE_sweep_outboard = str2double
(V.Wing1{1,1}.LE_sweep_outboard{1,1}.CONTENT);
dihedral_inboard = str2double
(V.Wing1{1,1}.dihedral_inboard{1,1}.CONTENT);
dihedral_midboard = str2double
(V.Wing1{1,1}.dihedral_midboard{1,1}.CONTENT);
dihedral_outboard = str2double
(V.Wing1{1,1}.dihedral_outboard{1,1}.CONTENT);
airfoilRoot = num2str (V.Wing1{1,1}.airfoilRoot{1,1}.CONTENT);
airfoilKink1 = num2str (V.Wing1{1,1}.airfoilKink1{1,1}.CONTENT);
airfoilKink2 = num2str (V.Wing1{1,1}.airfoilKink2{1,1}.CONTENT);
airfoilTip = num2str (V.Wing1{1,1}.airfoilTip{1,1}.CONTENT);
thickness_root = str2double (V.Wing1{1,1}.thickness_root{1,1}.CONTENT);
thickness_kink1 = str2double
(V.Wing1{1,1}.thickness_kink1{1,1}.CONTENT);
thickness_kink2 = str2double
(V.Wing1{1,1}.thickness_kink2{1,1}.CONTENT);
thickness_tip = str2double (V.Wing1{1,1}.thickness_tip{1,1}.CONTENT);
reference_convention = str2double
(V.Wing1{1,1}.reference_convention{1,1}.CONTENT);
configuration = str2double (V.Wing1{1,1}.configuration{1,1}.CONTENT);
winglet_Span = str2double
(V.Wing1{1,1}.winglet{1,1}.Span{1,1}.CONTENT);
winglet_taper_ratio = str2double
(V.Wing1{1,1}.winglet{1,1}.taper_ratio{1,1}.CONTENT);
winglet_LE_sweep = str2double
(V.Wing1{1,1}.winglet{1,1}.LE_sweep{1,1}.CONTENT);
winglet_Cant_angle = str2double
(V.Wing1{1,1}.winglet{1,1}.Cant_angle{1,1}.CONTENT);
winglet_root_incidence = str2double
(V.Wing1{1,1}.winglet{1,1}.root_incidence{1,1}.CONTENT);
winglet_tip_incidence = str2double
(V.Wing1{1,1}.winglet{1,1}.tip_incidence{1,1}.CONTENT);
flap_root_chord = str2double
(V.Wing1{1,1}.flap{1,1}.root_chord{1,1}.CONTENT);
flap_kink1_chord = str2double
(V.Wing1{1,1}.flap{1,1}.kink1_chord{1,1}.CONTENT);
flap_kink2_chord = str2double
(V.Wing1{1,1}.flap{1,1}.kink2_chord{1,1}.CONTENT);
aileron_chord = str2double
(V.Wing1{1,1}.aileron{1,1}.chord{1,1}.CONTENT);
aileron_Span = str2double
(V.Wing1{1,1}.aileron{1,1}.Span{1,1}.CONTENT);
aileron_position = str2double
(V.Wing1{1,1}.aileron{1,1}.position{1,1}.CONTENT);
aileron_limit_deflection_up = str2double
(V.Wing1{1,1}.aileron{1,1}.limit_deflection_up{1,1}.CONTENT);
aileron_limit_deflection_down = str2double
(V.Wing1{1,1}.aileron{1,1}.limit_deflection_down{1,1}.CONTENT);
```

```
slat_chord = str2double (V.Wing1{1,1}.slat{1,1}.chord{1,1}.CONTENT);
slat_root_position = str2double
(V.Wing1{1,1}.slat{1,1}.root_position{1,1}.CONTENT);
slat_tip_position = str2double
(V.Wing1{1,1}.slat{1,1}.tip_position{1,1}.CONTENT);
placement = str2double (V.Wing1{1,1}.placement{1,1}.CONTENT);
apex_locale = str2double (V.Wing1{1,1}.apex_locale{1,1}.CONTENT);
x_acb = str2double (V.Wing1{1,1}.x{1,1}.CONTENT);
y_acb = str2double (V.Wing1{1,1}.y{1,1}.CONTENT);
z_acb = str2double (V.Wing1{1,1}.z{1,1}.CONTENT);


         [Y0,X0] = coord_prof (airfoilRoot);
        N = length(Y0);
        command=[];
        h=1;        g=1;        n=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X0(h);
                h=h+1;
            else
                command(n)=Y0(g)+0.4; %traslato di 0.4 per evitare di
avere valori negativi
                g=g+1;
            end
        end
        n=n+1;
        alfa_0 = root_incidence*pi/180;
        command(n) = cos(alfa_0);        n=n+1;
        command(n) = sin(alfa_0);        n=n+1;
        [Y1,X1] = coord_prof (airfoilKink1);
        h=1;        g=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X1(h);
                h=h+1;
            else
                command(n)=Y1(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;
        alfa_1 = kink1_incidence*pi/180;
        command(n)= cos(alfa_1);        n=n+1;
        command(n)= sin(alfa_1);        n=n+1;
        [Y2,X2] = coord_prof (airfoilKink2);
        h=1;        g=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X2(h);
                h=h+1;
            else
                command(n)=Y2(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;
        alfa_2 = kink2_incidence*pi/180;
         command(n) = cos(alfa_2);        n=n+1;
```

```
        command(n) = sin(alfa_2);        n=n+1;
        [Y3,X3] = coord_prof (airfoilTip);
        h=1;        g=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X3(h);
                h=h+1;
            else
                command(n)=Y3(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;
        alfa_3 = tip_incidence*pi/180;
        command(n) = cos(alfa_3);        n=n+1;
        command(n) = sin(alfa_3);
        trasla_profilo_0 = -command(25);        n=n+1;
        command(n) = trasla_profilo_0;        n=n+1;
        CR = (2/(1+taper_tip))*(area/AR)^0.5; %corda_0;
        command(n) = CR;        n=n+1;
        command(n) = z_acb +10;%z_ala aumentata di 10 in modo da essere
sempre con z positiva
        n=n+1;
        command(n) = -(x_acb - l_nose); %x_ala
        trasla_profilo_1 = -command(75);        n=n+1;
        command(n) = trasla_profilo_1;        n=n+1;
        command(n) = CR*taper_kink1;%corda_1;
        trasla_profilo_2 = -command(125);        n=n+1;
        command(n) = trasla_profilo_2;        n=n+1;
        command(n) = CR*taper_kink2;%corda_2;
        trasla_profilo_3 = -command(175);        n=n+1;
        command(n) = trasla_profilo_3;        n=n+1;
        command(n) = CR*taper_tip;%corda_3;
        n=n+1;        command(n) = (-(90-
LE_sweep_inboard)*pi)/180;%freccia_0
        n=n+1;        command(n) = (-(90-
LE_sweep_midboard)*pi)/180;%freccia_1
        n=n+1;        command(n) = (-(90-
LE_sweep_outboard)*pi)/180;%freccia_2
        n=n+1;        command(n) =
cos((dihedral_inboard*pi)/180);%cos((diedro_0*pi)/180);
        n=n+1;        command(n) =
sin((dihedral_inboard*pi)/180);%sin((diedro_0*pi)/180);
        n=n+1;        command(n) =
cos((dihedral_midboard*pi)/180);%cos((diedro_1*pi)/180);
        n=n+1;        command(n) =
sin((dihedral_midboard*pi)/180);%sin((diedro_1*pi)/180);
        n=n+1;        command(n) =
cos((dihedral_outboard*pi)/180);%cos((diedro_2*pi)/180);
        n=n+1;        command(n) =
sin((dihedral_outboard*pi)/180);%sin((diedro_2*pi)/180);
         n=n+1;
        [Y4,X4] = coord_prof (airfoilTip);
        h=1;        g=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=Y4(g)+0.4;
                g=g+1;
```

```
            else
                command(n)=X4(h);
                h=h+1;
            end
        end
        n=n+1;
        trasla_profilo_4 = -command(222);
        command(n) = trasla_profilo_4;        n=n+1;
        command(n) = winglet_taper_ratio*CR*taper_tip;%corda_winglet;
        n=n+1;        command(n) = 3*CR;%V1_1;
        n=n+1;        command(n) = 0;%V2_1;
        n=n+1;        command(n) = CR*(1-flap_root_chord);%V3_1;
        n=n+1;        command(n) = spanwise_kink1*(Span*0.5);%V4_1;
        n=n+1;        command(n) = 3*CR;%V1_2;
        n=n+1;        command(n) = 0;%V2_2;
        n=n+1;        command(n) = CR*taper_kink1*(1-
flap_kink1_chord);%V3_2;
        n=n+1;        command(n) = spanwise_kink2*(Span*0.5)-
spanwise_kink1*(Span*0.5);%V4_2;
        n=n+1;        command(n) = CR*3;%V1_3;
        n=n+1;
        if aileron_position == 0
            a = 0;%V2_3;
        elseif aileron_position == 1
            a = 0.5*(aileron_Span*(1-spanwise_kink2)*Span*0.5);%V2_3;
        else
            a = aileron_Span*(1-spanwise_kink2)*Span*0.5;%V2_3;
        end
        command(n) = a;%V2_3;
        n=n+1;
        Cy = CR*taper_kink2-((CR*taper_kink2-taper_tip*CR)/(Span*0.5-
spanwise_kink2*Span*0.5))*a;
        command(n) = Cy*(1-
aileron_chord)+a*tan(LE_sweep_outboard*pi/180);%V3_3;
        n=n+1;        command(n) = aileron_Span*(1-
spanwise_kink2)*(Span*0.5);%V4_3;
        n=n+1;        command(n) = 3*CR;%V1_BA1;
        n=n+1;        command(n) =
slat_root_position*spanwise_kink1*(Span*0.5);%V2_BA1;
        n=n+1;
        Cyslat = CR-((CR-
taper_kink1*CR)/(spanwise_kink1*Span*0.5))*slat_root_position*spanwise_
kink1*(Span*0.5);
        command(n) =
Cyslat*slat_chord+slat_root_position*spanwise_kink1*(Span*0.5)*tan(LE_s
weep_inboard*pi/180);%V3_BA1;
        n=n+1;        command(n) = spanwise_kink1*Span*0.5;%V4_BA1;
        n=n+1;        command(n) = 3*CR;%V1_BA2;
        n=n+1;        command(n) = 0;%V2_BA2;
        n=n+1;        command(n) = taper_kink1*CR*slat_chord;%V3_BA2;
        n=n+1;        command(n) = spanwise_kink2*(Span*0.5)-
spanwise_kink1*Span*0.5;%V4_BA2;
        n=n+1;        command(n) = 3*CR;%V1_BA3;
        n=n+1;        command(n) = 0;%V2_BA3;
        n=n+1;        command(n) = taper_kink2*CR*slat_chord;%V3_BA3;
        n=n+1;        command(n) = (Span*0.5-
spanwise_kink2*(Span*0.5))*slat_tip_position;%V4_BA3;
```

```
        n=n+1;        command(n) =
spanwise_kink1*(Span*0.5)/(cos((dihedral_inboard*pi)/180));%apertura1;
        n=n+1;        command(n) = (spanwise_kink2*(Span*0.5)-
spanwise_kink1*(Span*0.5))/(cos((dihedral_midboard*pi)/180));%apertura2
;
        n=n+1;        command(n) = (Span*0.5-
spanwise_kink2*(Span*0.5))/(cos((dihedral_outboard*pi)/180));%apertura3
;
        n=n+1;        command(n) =
winglet_Span*((cos(winglet_Cant_angle))^-1)*CR*taper_tip;%apertura
winglet
        n=n+1;        command(n) =
cos((winglet_Cant_angle*pi)/180);%cant_w
        n=n+1;        command(n) =
sin((winglet_Cant_angle*pi)/180);%cant_w
        n=n+1;        command(n) =
(90+winglet_LE_sweep)*pi/180;%sweep_w
        n=n+1;        command(n) =
cos((winglet_root_incidence*pi)/180);%inc_r_w
        n=n+1;        command(n) =
sin((winglet_root_incidence*pi)/180);%inc_r_w
        n=n+1;        command(n) =
cos((winglet_tip_incidence*pi)/180);%inc_t_w
        n=n+1;        command(n) =
sin((winglet_tip_incidence*pi)/180);%inc_t_w
        n=n+1;        command(n) =
taper_kink1*CR*slat_chord+spanwise_kink1*(Span*0.5)*tan((LE_sweep_inboa
rd*pi)/180);%V5_BA1
        n=n+1;        command(n) =
taper_kink2*CR*slat_chord+(spanwise_kink2*(Span*0.5)-
spanwise_kink1*(Span*0.5))*tan((LE_sweep_midboard*pi)/180);%V5_BA2
        Cyslat_tip = CR-((CR-CR*taper_tip)/(Span*0.5))*((Span*0.5-
spanwise_kink2*(Span*0.5))*slat_tip_position+spanwise_kink2*(Span*0.5))
;
        n=n+1;        command(n) = Cyslat_tip*slat_chord+(Span*0.5-
spanwise_kink2*(Span*0.5))*slat_tip_position*tan((LE_sweep_outboard*pi)
/180);%V5_BA3
        n=n+1;        command(n) = CR*taper_kink1*(1-
flap_kink1_chord)+spanwise_kink1*(Span*0.5)*tan((LE_sweep_inboard*pi)/1
80);%V5_1
        n=n+1;        command(n) = CR*taper_kink2*(1-
flap_kink2_chord)+(spanwise_kink2*(Span*0.5)-
spanwise_kink1*(Span*0.5))*tan((LE_sweep_midboard*pi)/180);%V5_2

clear ala.xls
K = length(command);
x = 1:1:K;
q = [x; command];
q = q';
xlswrite('ala',q)

%% HORIZONTAL & VERTICAL TAIL

area_H = str2double (V.Horizontal_tail{1,1}.area{1,1}.CONTENT);
Span_H = str2double (V.Horizontal_tail{1,1}.Span{1,1}.CONTENT);
AR_H = str2double (V.Horizontal_tail{1,1}.AR{1,1}.CONTENT);
```

```
spanwise_kink_H = str2double
(V.Horizontal_tail{1,1}.spanwise_kink{1,1}.CONTENT);
taper_kink_H = str2double
(V.Horizontal_tail{1,1}.taper_kink{1,1}.CONTENT);
taper_tip_H = str2double
(V.Horizontal_tail{1,1}.taper_tip{1,1}.CONTENT);
root_incidence_H = str2double
(V.Horizontal_tail{1,1}.root_incidence{1,1}.CONTENT);
kink_incidence_H = str2double
(V.Horizontal_tail{1,1}.kink_incidence{1,1}.CONTENT);
tip_incidence_H = str2double
(V.Horizontal_tail{1,1}.tip_incidence{1,1}.CONTENT);
LE_sweep_inboard_H = str2double
(V.Horizontal_tail{1,1}.LE_sweep_inboard{1,1}.CONTENT);
LE_sweep_outboard_H = str2double
(V.Horizontal_tail{1,1}.LE_sweep_outboard{1,1}.CONTENT);
dihedral_inboard_H = str2double
(V.Horizontal_tail{1,1}.dihedral_inboard{1,1}.CONTENT);
dihedral_outboard_H = str2double
(V.Horizontal_tail{1,1}.dihedral_outboard{1,1}.CONTENT);
airfoilRoot_H = num2str
(V.Horizontal_tail{1,1}.airfoilRoot{1,1}.CONTENT);
airfoilKink_H = num2str
(V.Horizontal_tail{1,1}.airfoilKink{1,1}.CONTENT);
airfoilTip_H = num2str
(V.Horizontal_tail{1,1}.airfoilTip{1,1}.CONTENT);
thickness_root_H = str2double
(V.Horizontal_tail{1,1}.thickness_root{1,1}.CONTENT);
thickness_kink_H = str2double
(V.Horizontal_tail{1,1}.thickness_kink{1,1}.CONTENT);
thickness_tip_H = str2double
(V.Horizontal_tail{1,1}.thickness_tip{1,1}.CONTENT);
empennage_layout_H = str2double
(V.Horizontal_tail{1,1}.empennage_layout{1,1}.CONTENT);
limit_tailplane_deflection_up_H = str2double
(V.Horizontal_tail{1,1}.limit_tailplane_deflection_up{1,1}.CONTENT);
limit_tailplane_deflection_down_H = str2double
(V.Horizontal_tail{1,1}.limit_tailplane_deflection_down{1,1}.CONTENT);
vertical_locale_H = str2double
(V.Horizontal_tail{1,1}.vertical_locale{1,1}.CONTENT);
apex_locale_H = str2double
(V.Horizontal_tail{1,1}.apex_locale{1,1}.CONTENT);
E_chord_H = str2double
(V.Horizontal_tail{1,1}.Elevator{1,1}.chord{1,1}.CONTENT);
E_Span_H = str2double
(V.Horizontal_tail{1,1}.Elevator{1,1}.Span{1,1}.CONTENT);

area_V = str2double (V.Vertical_tail{1,1}.area{1,1}.CONTENT);
Span_V = str2double (V.Vertical_tail{1,1}.Span{1,1}.CONTENT);
AR_V = str2double (V.Vertical_tail{1,1}.AR{1,1}.CONTENT);
spanwise_kink_V = str2double
(V.Vertical_tail{1,1}.spanwise_kink{1,1}.CONTENT);
taper_kink_V = str2double
(V.Vertical_tail{1,1}.taper_kink{1,1}.CONTENT);
taper_tip_V = str2double (V.Vertical_tail{1,1}.taper_tip{1,1}.CONTENT);
root_incidence_V = str2double
(V.Vertical_tail{1,1}.root_incidence{1,1}.CONTENT);
```

```
kink_incidence_V = str2double
(V.Vertical_tail{1,1}.kink_incidence{1,1}.CONTENT);
tip_incidence_V = str2double
(V.Vertical_tail{1,1}.tip_incidence{1,1}.CONTENT);
LE_sweep_inboard_V = str2double
(V.Vertical_tail{1,1}.LE_sweep_inboard{1,1}.CONTENT);
LE_sweep_outboard_V = str2double
(V.Vertical_tail{1,1}.LE_sweep_outboard{1,1}.CONTENT);
dihedral_inboard_V = str2double
(V.Vertical_tail{1,1}.dihedral_inboard{1,1}.CONTENT);
dihedral_outboard_V = str2double
(V.Vertical_tail{1,1}.dihedral_outboard{1,1}.CONTENT);
airfoilRoot_V = num2str
(V.Vertical_tail{1,1}.airfoilRoot{1,1}.CONTENT);
airfoilKink_V = num2str
(V.Vertical_tail{1,1}.airfoilKink{1,1}.CONTENT);
airfoilTip_V = num2str (V.Vertical_tail{1,1}.airfoilTip{1,1}.CONTENT);
R_chord_V = str2double
(V.Vertical_tail{1,1}.Rudder{1,1}.chord{1,1}.CONTENT);
R_Span_V = str2double
(V.Vertical_tail{1,1}.Rudder{1,1}.Span{1,1}.CONTENT);
vertical_locale_V = str2double
(V.Vertical_tail{1,1}.vertical_locale{1,1}.CONTENT);
apex_locale_V = str2double
(V.Vertical_tail{1,1}.apex_locale{1,1}.CONTENT);
```

```
        [Y0,X0] = coord_prof (airfoilRoot_H);
        N = length(Y0);
        command=[];
        h=1;        g=1;        n=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X0(h);
                h=h+1;
            else
                command(n)=Y0(g)+0.4; %traslato di 0.4 per evitare di
avere valori negativi
                g=g+1;
            end
        end
        [Y1,X1] = coord_prof (airfoilKink_H);
        h=1;        g=1;        n=n+1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X1(h);
                h=h+1;
            else
                command(n)=Y1(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;        command(n) = -command(25);%trasla0
        n=n+1;        command(n) = -command(73);%trasla1
        CR_H = (2/(1+taper_tip_H))*(area_H/AR_H)^0.5;
        n=n+1;        command(n) = CR_H;%corda0
        n=n+1;        command(n) = CR_H*taper_kink_H;%corda1
```

```
        n=n+1;        command(n) =
spanwise_kink_H*Span_H*0.5;%apertura1
        n=n+1;        command(n) = -
(apex_locale_H*Total_fuselage_length - l_nose);%distanzaX
        n=n+1;        command(n) =
vertical_locale_H*Aftfuse_X_sect_vertical_diameter+10;%distanza_Z
        n=n+1;        command(n) = -(90-
LE_sweep_inboard_H)*pi/180;%freccia0
        n=n+1;        command(n) =
cos((dihedral_inboard_H*pi)/180);%diedro0
        n=n+1;        command(n) = sin((dihedral_inboard_H*pi)/180);
        n=n+1;        command(n) =
cos((root_incidence_H*pi)/180);%incidenza0
        n=n+1;        command(n) = sin((root_incidence_H*pi)/180);
        n=n+1;        command(n) =
cos((kink_incidence_H*pi)/180);%incidenza1
        n=n+1;        command(n) = sin((kink_incidence_H*pi)/180);

        [Y2,X2] = coord_prof (airfoilRoot_V);
        h=1;        g=1;        n=n+1;
        for n=n:n+N-1
            if mod(n,2) == 0
                command(n)=X2(h);
                h=h+1;
            else
                command(n)=Y2(g)+0.5;
                g=g+1;
            end
        end
        [Y3,X3] = coord_prof (airfoilKink_V);
        h=1;        g=1;        n=n+1;
        for n=n:n+N-1
            if mod(n,2) == 0
                command(n)=X3(h);
                h=h+1;
            else
                command(n)=Y3(g)+0.5;
                g=g+1;
            end
        end
        n=n+1;        command(n) = LE_sweep_inboard_V*pi/180;%sweep0
        n=n+1;        command(n) =
spanwise_kink_V*Span_V*(cos(LE_sweep_inboard_V*pi/180))^-1;%apertura1
        CR_V = (2/(1+taper_tip_V))*(area_V/AR_V)^0.5;
        n=n+1;        command(n) = CR_V;%corda0
        n=n+1;        command(n) = taper_kink_V*CR_V;%corda1
        n=n+1;        command(n) = Total_fuselage_length-
apex_locale_V*Total_fuselage_length;%distanzaX
        n=n+1;        command(n) = 0;%distanza_Z
        %
        %timone

        n=n+1;        command(n) = 0;%dist_apert;
        n=n+1;        command(n) = CR_V-R_chord_V*CR_V;%dist_lung;
        n=n+1;        command(n) = Span_V;%altezza;
        n=n+1;        command(n) = 3*CR_V;%margine_post;
        %
        %equilibratore
```

```
        n=n+1;        command(n) = 0;%dist_apert;
        n=n+1;        command(n) = CR_H-E_chord_H*CR_H;%dist_lung;
        n=n+1;        command(n) = Span_H*0.5;%altezza;
        n=n+1;        command(n) = 3*CR_H;%margine_post;
        n=n+1;        command(n) = CR_H*taper_tip_H -
E_chord_H*CR_H*taper_tip_H +
spanwise_kink_H*Span_H*0.5*tan(LE_sweep_inboard_H*pi/180) +
(Span_H*0.5-
spanwise_kink_H*Span_H*0.5)*tan(LE_sweep_outboard_H*pi/180);

        n=n+1;        command(n) =
(90+LE_sweep_outboard_H)*pi/180;%sweep1_H
        [Y4,X4] = coord_prof (airfoilTip_H);
        h=1;        g=1;        n=n+1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=X4(h);
                h=h+1;
            else
                command(n)=Y4(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;   command(n) = -command(n-24);%trasla_profilo_tip_H
        n=n+1;   command(n) = CR_H*taper_tip_H;%cordatip_H
        n=       command(n) = (Span_H*0.5-
spanwise_kink_H*Span_H*0.5)*(cos(LE_sweep_outboard_H*pi/180))^-
1;%aperturatip_H
        n=n+1;   command(n) = LE_sweep_outboard_V*pi/180;%sweep1_V
        [Y5,X5] = coord_prof (airfoilTip_V);
        h=1;        g=1;        n=n+1;
        for n=n:n+N-1
            if mod(n,2) == 0
                command(n)=Y5(h)+0.5;
                h=h+1;
            else
                command(n)=X5(g);
                g=g+1;
            end
        end
        n=n+1;  command(n) = CR_V*taper_tip_V;%cordatip_V
        n=n+1;  command(n) = (Span_V-
spanwise_kink_V*Span_V)*(cos(LE_sweep_outboard_V*pi/180))^-
1;%aperturatip_V
        n=n+1;  command(n) = CR_V*taper_tip_V -
R_chord_V*CR_V*taper_tip_V +
spanwise_kink_V*Span_V*tan(LE_sweep_inboard_V*pi/180) + (Span_V-
spanwise_kink_V*Span_V)*tan(LE_sweep_outboard_V*pi/180);
        n=n+1;  command(n) = cos((dihedral_outboard_H*pi)/180);%diedro1
        n=n+1;  command(n) = sin((dihedral_outboard_H*pi)/180);

clear coda.xls
K = length(command);
X = 1:1:K;
q = [X; command];
q = q';
xlswrite('coda',q)
```

```
%% ENGINE

Y_E = str2double (V.Engines1{1,1}.Y_locale{1,1}.CONTENT);
X_E = str2double (V.Engines1{1,1}.X_locale{1,1}.CONTENT);
Z_E = str2double (V.Engines1{1,1}.Z_locale{1,1}.CONTENT);
fineness_ratio = str2double
(V.Engines1{1,1}.fineness_ratio{1,1}.CONTENT);
d_max = str2double (V.Engines1{1,1}.d_max{1,1}.CONTENT);
toe_in = str2double (V.Engines1{1,1}.toe_in{1,1}.CONTENT);
pitch = str2double (V.Engines1{1,1}.pitch{1,1}.CONTENT);
Thrust_to_weight_ratio = str2double
(V.Engines1{1,1}.Thrust_to_weight_ratio{1,1}.CONTENT);
Propeller_diameter = str2double
(V.Engines1{1,1}.Propeller_diameter{1,1}.CONTENT);
Max_thrust = str2double (V.Engines1{1,1}.Max_thrust{1,1}.CONTENT);
Bypass_ratio_to_emulate = str2double
(V.Engines1{1,1}.Bypass_ratio_to_emulate{1,1}.CONTENT);
Thrust_reverser_effectivness = str2double
(V.Engines1{1,1}.Thrust_reverser_effectivness{1,1}.CONTENT);
Fan_cowl_length_ratio = str2double
(V.Engines1{1,1}.Fan_cowl_length_ratio{1,1}.CONTENT);
longitudinal_location = str2double
(V.Engines1{1,1}.Nacelle1{1,1}.longitudinal_location{1,1}.CONTENT);
vertical_location = str2double
(V.Engines1{1,1}.Nacelle1{1,1}.vertical_location{1,1}.CONTENT);

present2 = str2double (V.Engines2{1,1}.present{1,1}.CONTENT);
if present2 ==1
    Y_E2 = str2double (V.Engines2{1,1}.Y_locale{1,1}.CONTENT);
    X_E2 = str2double (V.Engines2{1,1}.X_locale{1,1}.CONTENT);
    Z_E2 = str2double (V.Engines2{1,1}.Z_locale{1,1}.CONTENT);
    fineness_ratio2 = str2double
(V.Engines2{1,1}.fineness_ratio{1,1}.CONTENT);
    d_max2 = str2double (V.Engines2{1,1}.d_max{1,1}.CONTENT);
    toe_in2 = str2double (V.Engines2{1,1}.toe_in{1,1}.CONTENT);
    pitch2 = str2double (V.Engines2{1,1}.pitch{1,1}.CONTENT);
    longitudinal_location2 = str2double
(V.Engines2{1,1}.Nacelle3{1,1}.longitudinal_location{1,1}.CONTENT);
    vertical_location2 = str2double
(V.Engines2{1,1}.Nacelle3{1,1}.vertical_location{1,1}.CONTENT);
else
    Y_E2 = 0.5;
    X_E2 = 0.5;
    Z_E2 = 0.5;
    fineness_ratio2 = str2double
(V.Engines1{1,1}.fineness_ratio{1,1}.CONTENT);
    d_max2 = str2double (V.Engines1{1,1}.d_max{1,1}.CONTENT);
    toe_in2 = str2double (V.Engines1{1,1}.toe_in{1,1}.CONTENT);
    pitch2 = str2double (V.Engines1{1,1}.pitch{1,1}.CONTENT);
    longitudinal_location2 = str2double
(V.Engines1{1,1}.Nacelle1{1,1}.longitudinal_location{1,1}.CONTENT);
    vertical_location2 = str2double
(V.Engines1{1,1}.Nacelle1{1,1}.vertical_location{1,1}.CONTENT);
end


        chi_lg = 0.2079;
```

```
chi_dia = 0.2028;
n = 12;
command = [];
l_nac = d_max*fineness_ratio;
d_nac = d_max;
toe = toe_in*pi/180;
pitch = pitch*pi/180;
t = [1/n:1/n:1];
h=1;        g=1;
for i=1:n
    xE(i) = chi_lg*l_nac*exp(t(i)*pi/2)*sin(t(i)*pi/2);
    zE(i) = chi_dia*d_nac*(1+exp(t(i)*pi/2)*cos(t(i)*pi/2));
end
for i=1:2*n
    if mod(i,2) == 0
        command(i)= xE(h);
        h=h+1;
    else
        command(i)= zE(g);
        g=g+1;
    end
end
q=2*n+1;    command(q) = -(longitudinal_location-l_nose);%posX
q=q+1;      command(q) = Y_E*Span*0.5;%posY
q=q+1;      command(q) = vertical_location+10;%posZ
q=q+1;      command(q) = cos(toe);%toe
q=q+1;      command(q) = sin(toe);
q=q+1;      command(q) = cos(pitch);%pitch
q=q+1;      command(q) = sin(pitch);

l_nac2 = d_max2*fineness_ratio2;
d_nac2 = d_max2;
toe2 = toe_in2*pi/180;
pitch2 = pitch2*pi/180;
t = [1/n:1/n:1];
h=1;        g=1;
for i=1:n
    xE2(i) = chi_lg*l_nac2*exp(t(i)*pi/2)*sin(t(i)*pi/2);
    zE2(i) = chi_dia*d_nac2*(1+exp(t(i)*pi/2)*cos(t(i)*pi/2));
end
for i=q+1:q+2*n
    if mod(i,2) == 1
        command(i)= xE2(h);
        h=h+1;
    else
        command(i)= zE2(g);
        g=g+1;
    end
end
q=length(command)+1;
command(q) = -(longitudinal_location2-l_nose);%posX
q=q+1;        command(q) = Y_E2*Span*0.5;%posY
q=q+1;        command(q) = vertical_location2+10;%posZ
q=q+1;        command(q) = cos(toe2);%toe
q=q+1;        command(q) = sin(toe2);
q=q+1;        command(q) = cos(pitch2);%pitch
q=q+1;        command(q) = sin(pitch2);
```

```
clear motore.xls
K = length(command);
X = 1:1:K;
q = [X; command];
q = q';
xlswrite('motore',q)

%% VENTRAL FIN

present_f = str2double (V.Ventral_fin{1,1}.present{1,1}.CONTENT);
chord_fraction_at_midfuse = str2double
(V.Ventral_fin{1,1}.chord_fraction_at_midfuse{1,1}.CONTENT);
Span_f = str2double (V.Ventral_fin{1,1}.Span{1,1}.CONTENT);
spanwise_kink_f = str2double
(V.Ventral_fin{1,1}.spanwise_kink{1,1}.CONTENT);
taper_kink_f = str2double (V.Ventral_fin{1,1}.taper_kink{1,1}.CONTENT);
taper_tip_f = str2double (V.Ventral_fin{1,1}.taper_tip{1,1}.CONTENT);
LE_sweep_inboard_f = str2double
(V.Ventral_fin{1,1}.LE_sweep_inboard{1,1}.CONTENT);
LE_sweep_outboard_f = str2double
(V.Ventral_fin{1,1}.LE_sweep_outboard{1,1}.CONTENT);
cant_inboard_f = str2double
(V.Ventral_fin{1,1}.cant_inbord{1,1}.CONTENT);
cant_outboard_f = str2double
(V.Ventral_fin{1,1}.cant_outboard{1,1}.CONTENT);
X_locale_f = str2double (V.Ventral_fin{1,1}.X_locale{1,1}.CONTENT);
Z_locale_f = str2double (V.Ventral_fin{1,1}.Z_locale{1,1}.CONTENT);

        command = []; n = 1;
        command(n) = chord_fraction_at_midfuse*Total_fuselage_length;
n=n+1;
        command(n) = -LE_sweep_inboard_f*pi/180; n=n+1;
        apert_f = Span_f*Span;
        command(n) =
spanwise_kink_f*0.5*apert_f*(cos(LE_sweep_inboard_f*pi/180))^-1; n=n+1;
        command(n) = LE_sweep_outboard_f*pi/180; n=n+1;
        command(n) =
taper_kink_f*chord_fraction_at_midfuse*Total_fuselage_length; n=n+1;
        command(n) =
taper_tip_f*chord_fraction_at_midfuse*Total_fuselage_length*(cos(LE_swe
ep_outboard_f*pi/180))^-1;
       n=n+1;     command(n) = cos((cant_inboard_f*pi)/180);%cant0
       n=n+1;     command(n) = -sin((cant_inboard_f*pi)/180);
       n=n+1;     command(n) = -cos((cant_outboard_f*pi)/180);%cant1
       n=n+1;     command(n) = sin((cant_outboard_f*pi)/180);
       n=n+1;     command(n) = X_locale_f*Total_fuselage_length-l_nose;
        if present_f ==1
            n=n+1;        command(n) =
Z_locale_f*Aftfuse_X_sect_vertical_diameter+10-
0.5*Aftfuse_X_sect_vertical_diameter;
        else
            n=n+1;        command(n) =
Z_locale_f*Aftfuse_X_sect_vertical_diameter-
0.5*Aftfuse_X_sect_vertical_diameter;
        end

clear ventral_fin.xls
K = length(command);
```

```
X = 1:1:K;
q = [X; command];
q = q';
xlswrite('ventral_fin',q)

%% FAIRING

command = [];
l_f = 2;  l_m = 1;  l_aft = 2;  w = 2;  h = 1;  nx = 5;  n = 5;
n_f = 7;
x_f = [0:l_f/n_f:l_f];
x_f_a = [l_m:l_aft/n_f:l_m+l_aft];
theta = [0:pi/12:pi/2];
for i=1:length(theta)
    z_theta(i) = (abs(sin(theta(i)))^(2/n))*(h/2)*sign(sin(theta(i)));
end
for i=1:length(x_f)
    z_theta_mid(i) = h*0.5*(1-(x_f(i)/l_f)^nx)^(1/n);
end

%FORE

m=1;
for i=1:8
    command(m)= x_f(i);
    m=m+1;
end
for i=1:8
    command(m) = -x_f_a(i);
    m=m+1;
end
for i=1:length(x_f)
    for j=1:length(theta)
        y_f(i,j)=w*0.5*(1-((x_f(i)/l_f)^nx)-
((z_theta(j)/(h*0.5))^n))^(1/n);
    end
end
for g = 1:7
    for i = 1:7
        k = isreal(y_f(g,i));
            if k == 0
                y_f(g,i)=0;
                command(m)=y_f(g,i);
                m=m+1;
                command(m)=z_theta_mid(g);
                m=m+1;
            else
                command(m)=y_f(g,i);
                m=m+1;
                command(m)=z_theta(i);
                m=m+1;
            end
    end
end

%AFT

for i=1:length(x_f_a)
```

```
    z_theta_mid_a(i) = h*0.5*(1-((x_f_a(i)-l_m)/l_aft)^nx)^(1/n);
end
for i=1:length(x_f_a)
    for j=1:length(theta)
        y_f_a(i,j)=w*0.5*(1-((((x_f_a(i)-l_m)/l_aft))^nx)-
((z_theta(j)/(h*0.5))^n))^(1/n);
    end
end
for g = 1:7
    for i = 1:7
        k = isreal(y_f_a(g,i));
            if k == 0
                y_f_a(g,i)=0;
                command(m)=y_f_a(g,i);
                m=m+1;
                command(m)=z_theta_mid_a(g);
                m=m+1;
            else
                command(m)=y_f_a(g,i);
                m=m+1;
                command(m)=z_theta(i);
                m=m+1;
            end
    end
end

clear fairing.xls

K = length(command);
X = 1:1:K;
q = [X; command];
q = q';
xlswrite('fairing',q)

%% CANARD

present = str2double (V.Canard{1,1}.present{1,1}.CONTENT);
AR_ca = str2double (V.Canard{1,1}.AR{1,1}.CONTENT);
area_ca = str2double (V.Canard{1,1}.area{1,1}.CONTENT);
Span_ca = str2double (V.Canard{1,1}.Span{1,1}.CONTENT);
spanwise_kink_ca = str2double
(V.Canard{1,1}.spanwise_kink{1,1}.CONTENT);
taper_kink_ca = str2double (V.Canard{1,1}.taper_kink{1,1}.CONTENT);
taper_tip_ca = str2double (V.Canard{1,1}.taper_tip{1,1}.CONTENT);
root_incidence_ca = str2double
(V.Canard{1,1}.root_incidence{1,1}.CONTENT);
kink_incidence_ca = str2double
(V.Canard{1,1}.kink_incidence{1,1}.CONTENT);
tip_incidence_ca = str2double
(V.Canard{1,1}.tip_incidence{1,1}.CONTENT);
LE_sweep_inboard_ca = str2double
(V.Canard{1,1}.LE_sweep_inboard{1,1}.CONTENT);
LE_sweep_outboard_ca = str2double
(V.Canard{1,1}.LE_sweep_outboard{1,1}.CONTENT);
dihedral_inboard_ca = str2double
(V.Canard{1,1}.dihedral_inboard{1,1}.CONTENT);
dihedral_outboard_ca = str2double
(V.Canard{1,1}.dihedral_outboard{1,1}.CONTENT);
```

```
airfoilRoot_ca = num2str (V.Canard{1,1}.airfoilRoot{1,1}.CONTENT);
airfoilKink_ca = num2str (V.Canard{1,1}.airfoilKink{1,1}.CONTENT);
airfoilTip_ca = num2str (V.Canard{1,1}.airfoilTip{1,1}.CONTENT);
Elevator_chord_ca = str2double
(V.Canard{1,1}.Elevator{1,1}.chord{1,1}.CONTENT);
Elevator_Span_ca = str2double
(V.Canard{1,1}.Elevator{1,1}.Span{1,1}.CONTENT);
z_ca = str2double (V.Canard{1,1}.z{1,1}.CONTENT);
x_ca = str2double (V.Canard{1,1}.x{1,1}.CONTENT);

command=[];
[YC0,XC0] = coord_prof (airfoilRoot_ca);
        h=1;        g=1;        n=1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=YC0(g)+0.4;
                g=g+1;
            else
                command(n)=XC0(h);
                h=h+1;
            end
        end
[YC1,XC1] = coord_prof (airfoilKink_ca);
        h=1;        g=1;        n=n+1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=XC1(h);
                h=h+1;
            else
                command(n)=YC1(g)+0.4;
                g=g+1;
            end
        end
[YC2,XC2] = coord_prof (airfoilTip_ca);
        h=1;        g=1;        n=n+1;
        for n=n:n+2*N-1
            if mod(n,2) == 0
                command(n)=XC2(h);
                h=h+1;
            else
                command(n)=YC2(g)+0.4;
                g=g+1;
            end
        end
        n=n+1;        command(n) = -(x_ca-l_nose);%distanzaX
        if present == 1
            n=n+1;        command(n) = z_ca+10;%distanza_Z
        else
            n=n+1;        command(n) = z_ca;%distanza_Z
        end
    n=n+1;     command(n) = -command(26);%trasla0
    n=n+1;     command(n) = -command(73);%trasla1
    n=n+1;     command(n) = -command(121);%trasla2
    CR_ca = (2/(1+taper_tip_ca))*(area_ca/AR_ca)^0.5;
    n=n+1;     command(n) = CR_ca;%corda0
    n=n+1;     command(n) = CR_ca*taper_kink_ca;%corda1
    n=n+1;     command(n) = CR_ca*taper_tip_ca;%corda2
    n=n+1;     command(n) = spanwise_kink_ca*Span_ca*0.5;%apertura1
```

```
    n=n+1;     command(n) = Span_ca*0.5-
spanwise_kink_ca*Span_ca*0.5;%apertura2
    n=n+1;     command(n) = -(90-LE_sweep_inboard_ca)*pi/180;%freccia0
    n=n+1;     command(n) = -(90-LE_sweep_outboard_ca)*pi/180;%freccia1
    n=n+1;     command(n) = cos((dihedral_inboard_ca*pi)/180);%diedro0
    n=n+1;     command(n) = sin((dihedral_inboard_ca*pi)/180);
    n=n+1;     command(n) = cos((dihedral_outboard_ca*pi)/180);%diedro1
    n=n+1;     command(n) = sin((dihedral_outboard_ca*pi)/180);
    n=n+1;     command(n) = cos((root_incidence_ca*pi)/180);%incidenza0
    n=n+1;     command(n) = sin((root_incidence_ca*pi)/180);
    n=n+1;     command(n) = cos((kink_incidence_ca*pi)/180);%incidenza1
    n=n+1;     command(n) = sin((kink_incidence_ca*pi)/180);
    n=n+1;     command(n) = cos((tip_incidence_ca*pi)/180);%incidenza2
    n=n+1;     command(n) = sin((tip_incidence_ca*pi)/180);
    n=n+1;     command(n) = 3*CR_H;%V1
    n=n+1;     command(n) = 0;%V2
    n=n+1;     command(n) = CR_ca-Elevator_chord_ca*CR_ca;%V3
    n=n+1;     command(n) = Span_ca*0.5;%V4
    n=n+1;        command(n) = CR_ca*taper_tip_ca*(1-
Elevator_chord_ca)+spanwise_kink_ca*Span_ca*0.5*tan(LE_sweep_inboard_ca
*pi/180)+(Span_ca*0.5-
spanwise_kink_ca*Span_ca*0.5)*tan(LE_sweep_outboard_ca*pi/180);%V5

clear canard.xls

K = length(command);
X = 1:1:K;
q = [X; command];
q = q';
xlswrite('canard',q)
```

Bibliografia

[1] Pierre Saquet - Further development of the AcBuilder tool for constructing geometrical models of aircraft , 02/09/2011 KTH Royal Institute of Technology, Stockholm Sweden

[2] Andrés Puelles – Further Development and Application of CEASIOM to the Transonic Cruiser Design, February 2009 Royal Institute of Technology, Stockholm Sweden

Ringraziamenti

Colgo l’occasione per ringraziare il prof. Sergio Ricci per aver accettato per la seconda volta di farmi da relatore.

Un ringraziamento particolare va a mia madre Maria Gemma e a mio padre Lorenzo, che si sono fatti carico di tutte le, ingenti, spese, sostenendomi ed incoraggiandomi in ogni occasione.

Un grazie enorme va alla mia ragazza Emanuela, che mi ha accompagnato in questa avventura non facendomi mai mancare il suo affetto, anche e soprattutto nei momenti di difficoltà.

Non posso non ringraziare i miei colleghi per aver collaborato con me in numerosi progetti. In particolare Michele, i cui consigli sono stati preziosi per lo svolgimento di questa tesi.

Ringrazio infine tutti coloro che mi hanno sostenuto ed appoggiato durante il mio lungo cammino universitario.