ALMA MATER STUDIORUM UNIVERSITA' DI BOLOGNA

SCUOLA DI INGEGNERIA

Sede di Forlì

Corso di Laurea Triennale in

INGEGNERIA AEROSPAZIALE

Corso 9234

ELABORATO FINALE DI LAUREA

In

AERODINAMICA DEGLI AEROMODILI

PROGETTO PRELIMINARE DI UN DRONE VTOL PER APPLICAZIONI AD ELEVATA AUTONOMIA CHILOMETRICA ED ORARIA

ALLEGATO

CANDIDATO

Giacomo Pierini

AZIENDA OSPITANTE

Siralab Robotics S.r.l.

RELATORE

Alessandro Talamelli

Anno Accademico

2022/2023

INDICE

1 MANUALE D'ISTRUZIONI DI XFLR5

Il manuale d'istruzioni di XFLR5 è servito da guida per ogni simulazione. In esso sono spiegati i metodi computazionali utilizzati, per cui esso ha permesso di comprendere nel migliore dei modi le mancate convergenze dei metodi. Inoltre, esso consiglia qual è il metodo opportuno per ogni caso sottolineando i limiti di ognuno in modo da mettere in guardia l'utilizzatore dei possibili risultati non accurati.

2 CARATTERISTICHE AERODINAMICHE

2.1 ALA PRINCIPALE

Il file in questione è servito a ricavare i coefficienti relativi alla portanza della sola ala principale e della configurazione ala principale-fusoliera. Per farlo sono stati simulati i due casi in XFLR5 e poi i risultati ottenuti sono stati esportati in Matlab per essere interpolati. Ipotizzando un andamento rettilineo, sono stati ottenuti due valori per ogni caso: le pendenze delle curve di portanza e il valore di portanza per angolo d'attacco nullo, a partire dalla corda. Confrontando le pendenze è stato possibile dimostrare l'interferenza della fusoliera sull'ala principale e ricavarne il coefficiente associato. Per facilitare la considerazione di quest'ultimo coefficiente nella trattazione quantitativa, è stato considerato come un corpo unico l'unione tra l'ala principale e fusoliera. L'ultimo risultato ricavato è stato il coefficiente di momento di beccheggio ad incidenza nulla (rispetto alla corda) della configurazione ala principale-fusoliera poiché servirà nel calcolo dei momenti di beccheggio del drone completo. Per far ciò, è stata simulata la configurazione ala principale-fusoliera considerando il baricentro coincidente con il punto neutro. Una volta esportati i risultati di XFLR5 in Matlab, è stato necessario solo salvare il dato d'interesse poiché era esplicito.

2.2 PIANO DI CODA

Il file in questione è servito a ricavare i coefficienti relativi alla portanza del solo piano di coda e della configurazione piano di coda-fusoliera-timone. Per farlo sono stati simulati i due casi in XFLR5 e poi i risultati ottenuti sono stati esportati in Matlab per essere interpolati. Ipotizzando un andamento rettilineo, sono stati ottenuti due valori per ogni caso: le pendenze delle curve di portanza e il valore di portanza per angolo d'attacco nullo, a partire dalla corda. Confrontando le pendenza è stato possibile dimostrare che la fusoliera interferisce in maniera trascurabile sul piano di coda. Perciò gli output considerati sono solamente i due coefficienti della retta di portanza relativi al solo piano di coda. Oltre a ciò, sono stati interpolati i dati relativi alla resistenza del solo piano di coda perché in fase di crociera è stato verificato se il momento generato dallo sfasamento verticale delle due ali è trascurabile o meno.

2.3 DRONE COMPLETO

Il file in questione è servito a ricavare tutti i coefficienti aerodinamici del drone completo. Per farlo è stato simulato l'intero velivolo considerando il peso totale e la distribuzione di massa effettiva durante la fase operativa. I risultati ottenuti sono poi stati esportati in Matlab per essere interpolati. È stata ricavata la pendenza della curva di portanza, la pendenza della curva di momento, il coefficiente di momento a portanza nulla e i 3 coefficienti che definiscono l'andamento parabolico del coefficiente di resistenza. Inoltre, sono stati ricavati indirettamente i coefficienti di downwash. L'ultimo passaggio è stato quello di ricavare il contributo ai momenti di beccheggio e alla portanza del flap anteriore e dell'equilibratore. Per farlo è stato simulato il drone completo in 4 condizioni di deflessioni: -5, -2, 2, 5 gradi. Le stesse deflessioni sono state considerate sia per il flap che per l'equilibratore, quindi per un totale di 8 simulazioni. Tutti i dati

derivanti da esse sono stati interpolanti con un polinomio di primo grado nel caso dei coefficienti di portanza e di momento di beccheggio, invece con un polinomio di secondo grado per la resistenza. I 3 coefficienti finali relativi alla resistenza del drone completo sono stati ricavati come media di quelli trovati per la condizione di equilibratore deflesso e non in modo da non sottostimare il coefficiente di resistenza aerodinamica in crociera.

2.4 VELOCITA' DI STALLO

Partendo dalle condizioni di stallo dell'ala principale ricavate con XFLR5, è stato risolto il sistema algebrico usato per l'equilibrio in crociera riadatto a due diverse incognite: coefficiente di portanza totale e deflessione dell'equilibratore. Poiché il piano di coda, per effetto della scia dell'ala principale, vedrà un incidenza inferiore, è sensato dire che il drone sarà in stallo quando lo sarà l'ala principale. In ogni caso, nell'allegato 8 è stato calcolato l'effettivo angolo visto dal piano di coda, considerando anche la deflessione dell'equilibratore, per ogni condizione di crociera ed è stato verificato che esso sia inferiore del rispetto angolo di stallo.

3 ELICA

La spinta e la potenza resistente generata dall'elica è stata modellata nella trattazione quantitativa attraverso i coefficienti di spinta e di potenza al variare del rapporto di avanzamento. Nel file in questione è riportato un modello empirico per ricavare i coefficienti d'interesse e dei dati sperimentali ricavati in galleria del vento inerenti alla pala d'elica presa in considerazione per questa tesi. Nel primo caso, il software Matlab è servito solo alla creazione dei grafici, invece nel secondo caso ha anche svolto prima un'interpolazione. Per ragioni di accuratezza, è stato scelto il modello sperimentale per le analisi successive, perciò sono state create due funzioni relative al coefficiente di spinta e di potenza che sono state richiamate in seguito.

4 MOTORE

Sfruttando i dati forniti dal costruttore sono state ricavate le curve di coppia ed efficienza del motore scelto al variare della corrente attraverso un interpolazione in Matlab. inoltre, sono stati interpolati i dati forniti dal costruttore dello stesso modello di motore con diversi KV per confrontare l'andamento della coppia. per finire, è stato realizzato un grafico che rappresenta il numero di giri a regimi al variare della manetta. Ovviamente, tutte i plot realizzati sono approssimati perché derivanti da soli tre dati forniti dal costruttore.

5 SALITA

Questo file è servito a ricavare tutte le variabili d'interesse durante la salita: le leggi orarie, la velocità angolare dell'elica e la potenza necessaria. Per prima cosa è stata calcolata la velocità angolare dell'elica tramite il metodo delle tangenti di Newton. È stata cercata quella velocità angolare tale per cui il drone sarebbe stato in equilibrio alla velocità verticale voluta. Successivamente, è stata risolta l'equazione della dinamica della salita discretizzando il problema poiché non risolvibile analiticamente. Ad ogni step è stata calcolata: la posizione, la velocità, il coefficiente di spinta e di potenza resistente dell'elica, la potenza

necessaria, la coppia resistente e la coppia motrice. Dopo di che è stato creato un grafico per le leggi orarie, la coppia motrice, la coppia resistente e la potenza necessaria. Quest'ultima è stata anche interpolata per rendere possibile l'integrazione a seguire che è servita a ricavare l'energia totale spesa in rotazione.

6 METODO DELLE TAGENTI DI NEWTON

Questo metodo serve a trovare uno zero di una generica funzione. In generale, esso si usa per equazioni trascendenti rispetto alla variabile d'interesse, ma in questa tesi è stato usato per evitare errori di calcolo. Nel caso l'equazione in input fosse un polinomio di grado n, esso troverebbe un'unica soluzione, per questo il metodo di Newton richiede in ingresso anche il valore di primo tentativo in modo da trovare una soluzione vicina a quella prevista. Per questo motivo è sempre bene controllare che la soluzione trovata abbia senso nel contesto in esame perché è possibile trovare soluzioni irrealistiche ingegneristicamente parlando.

7 ROTAZIONE

7.1 ANALISI DELLE GRANDEZZE D'INTERESSE

In questo file sono state calcolate le grandezze d'interesse durante ogni istante della rotazione come: variabili di controllo per l'equilibrio (assetto, equilibratore, flap), velocità angolare e potenza necessaria. La prima cosa fatta è il confronto tra due modalità per svolgere la manovra: con il solo equilibratore, con quest'ultimo e un flap anteriore. Successivamente sono stati confrontati i dati ricavati tramite XFLR5 usando anche il flap con quelli ricavati analiticamente in Matlab. Per finire è stata calcolata in ogni istante della rotazione, attraverso un ciclo iterativo, la velocità angolare tale da garantire l'equilibrio tangenziale, la potenza necessaria richiesta dalla batteria, la coppia motrice e la coppia resistente dell'elica. Le prime due sono state riportante in un grafico e poi sono state anche interpolate poiché la prima servirà per programmare l'ESC e la seconda per calcolare l'energia totale spesa durante la rotazione. Le ultime due servivano a verificare il bilanciamento dei momenti agenti sull'elica.

7.2 METODOLOGIA RISOLUTIVA IN XFLR5

Il file in questione è composto da un sistema algebrico rispetto alle variazione di deflessioni di equilibratore e flap. Esso è servito a ricavare la soluzione target con XFLR5 più rapidamente e ad avere due valori di primo tentativo sensati. Inoltre, è stato utilizzato per ricavare le deflessioni tale da innescare e frenare la manovra di rotazione, la deflessione d'equilibratore tale da equilibrare il drone durante la salita. Il metodo converge, ma la limitata accuratezza angolare nelle deflessioni delle superfici di controllo non permette il perfetto equilibrio in termini di momento di beccheggio.

7.3 STIMA DEGLI ERRORI

Come detto nell'allegato 7.2, non è possibile ricavare delle soluzioni in fase di rotazione che garantiscono un perfetto equilibrio in termini di momento. Perciò è stato stimato l'errore in termini di assetto e velocità angolare dovuti a questi errori. Per farlo sono state implementate le equazioni della cinematica inerenti alle rotazioni e da esse, attraverso un ciclo iterativo, sono stati calcolati gli errori totali. È stato supposto che tra ogni step l'errore sui momenti fosse costante, per questo il computo finale sarà conservativo.

8 CROCIERA

La parte iniziale di questo file è servita a motivare la modalità di crociera scelta. Attraverso l'esportazione dei dati derivanti dalle simulazioni in XFLR5, è stato possibile verificare che è conveniente utilizzare il solo equilibratore in questa fase poiché genera minor resistenza aerodinamica a parità di condizione di volo. La parte successiva serve ad analizzare le diverse condizione di crociera, in particolare: l'assetto e la deflessione dell'equilibratore per l'equilibrio, la velocità angolare dell'elica richiesta, la potenza necessaria, la coppia motrice e la coppia resistente. Le prime tre variabili serviranno a programmare il controllore di volo per mantenere in equilibrio il drone, la terza è servita a ricercare il minimo e a verificarne la fattibilità, le ultime due sono servite a verificare il bilanciamento dei momenti agenti sull'elica. L'ultimo computo svolto, è stato calcolare il coefficiente di momento derivante dallo sfasamento verticali del piano di coda rispetto al baricentro.

9 AUTONOMIA CHILOMETRICA ED ORARIA

Il codice scritto in questo file è un modello di scarica della batteria ricavato sperimentalmente. Esso prende in input la curva di potenza necessaria ricavata nell'allegato 8, i parametri della batteria scelta, i coefficienti sperimentali associati a quest'utltima e computa in output l'autonomia chilometrica e oraria in ogni condizione di volo.

10 ORDINE DI ESECUZIONE FILE MATLAB

Fig.1 File Matlab in ordine di esecuzione.

ALLEGATO: 2 – 9

ALLEGATO 2

ALLEGATO 2.1

```
B=readtable('ala_principale_flap_0.txt');
ala_P_f_0=table2array(B);
   % input: dati ricavati dalla simulazione di ala principale con XFLR5
B=readtable('ala_principale_fusoliera_flap_0.txt');
ala_P_fusoliera_flap_0=table2array(B);
   % input: dati ricavati dalla simulazione di fusoliera piÃ¹ ala
principale con XFLR5

alpha=(-3*(pi/180)):0.01:(10*(pi/180));

ala_P_f_0_interp=fit((ala_P_f_0(:,1).*(pi/180)),ala_P_f_0(:,3),'poly1');
c_ala_P_f_0=coeffvalues(ala_P_f_0_interp);
CL_ala_P_f_0=c_ala_P_f_0(1).*alpha + c_ala_P_f_0(2);
ala_P_fusoliera_flap_0_interp=fit((ala_P_fusoliera_flap_0(:,1).*(pi/180))
,ala_P_fusoliera_flap_0(:,3),'poly1');
c_ala_P_fusoliera_flap_0=coeffvalues(ala_P_fusoliera_flap_0_interp);
CL_ala_P_fusoliera_flap_0=c_ala_P_fusoliera_flap_0(1).*alpha +
c_ala_P_fusoliera_flap_0(2);

Cl_al1=c_ala_P_f_0(1)
   % pendenza curva coefficiente di portanza della sola ala principale
Cl_al1_fuso=c_ala_P_fusoliera_flap_0(1)
   % pendenza curva du portanza di al principale piÃ¹ fusoliera.
   % serve a ricavare il coefficiente di interferenza aerodinamica della
   % fusoliera.
Cl01b=c_ala_P_fusoliera_flap_0(2)
   % coefficiente di portanza ad incidenza nulla considerando anche la
   % presenza della fusoliera
dBeta_dAlpha=-(Cl_al1_fuso/Cl_al1 - 1)
   % coefficiente di interferenza aerodinamuca della fusoliera


figure (1)
plot(alpha,CL_ala_P_f_0,'r',ala_P_f_0(:,1).*(pi/180),ala_P_f_0(:,3),'r*')
hold on
plot(alpha,CL_ala_P_fusoliera_flap_0,'b',ala_P_fusoliera_flap_0(:,1).*(pi
/180),ala_P_fusoliera_flap_0(:,3),'b*')
xlabel('incidenza ala principale')
ylabel('coefficiente di portanza')
title('ALA PRINCIPALE :interpolazione dati ricavati con XFLR5 ')
legend('deflessione flap di 0 deg','','deflessione flap 0 deg con
fusoliera')

B=readtable('ala_principale_fusoliera_flap_0_Cm01B.txt');
ala_P_fusoliera_flap_0_Cm01B=table2array(B);
   % input: dati ricavati dalla simulazione di ala principale piÃ¹
fusoliera
   % considerando il baricentro concidente con il centro aerodinamico
```

```
Cm01B=mean(ala_P_fusoliera_flap_0_Cm01B(:,9))
   % coefficiente di momento aerodinamico della
   % cofigurazione ala principale-fusoliera. Anche se il profilo Ã¨
   % simmetrico la presenza del corpo centrale permette il discostamento
   % tra centro delle pressioni e centro aerodinamico, per questo la
   % configurazione in questione possiede un momento intrinseco
   % indipendente dal'angolo d'attacco. Esso, dopo verifica con XFLR5,
   % risulta non trascurabile.
```

NOTA 1 Script Matlab per ricavare i coeffcienti aerodinamici d'interesse relativi all'ala principale

ALLEGATO 2.2

```
B=readtable('piano_coda_equi_0.txt');
piano_C_equi_0=table2array(B);
   % input: dati ricavati dalla simulazione di piano di coda con XFLR5
B=readtable('piano_coda_fusoliera_timone_equi_0.txt');
piano_C_fusoliera_timone_equi_0=table2array(B);
   % input: dati ricavati dalla simulazione di fusoliera piÃ¹ piano di
coda con XFLR5

alpha=(-3*(pi/180)):0.01:(10*(pi/180));

piano_C_equi_0_interp=fit((piano_C_equi_0(:,1).*(pi/180)),piano_C_equi_0(
:,3),'poly1');
c_piano_C_equi_0=coeffvalues(piano_C_equi_0_interp);
CL_piano_C_equi_0=c_piano_C_equi_0(1).*alpha + c_piano_C_equi_0(2);
piano_C_fusoliera_timone_equi_0_interp=fit((piano_C_fusoliera_timone_equi
_0(:,1).*(pi/180)),piano_C_fusoliera_timone_equi_0(:,3),'poly1');
c_piano_C_fusoliera_timone_equi_0=coeffvalues(piano_C_fusoliera_timone_eq
ui_0_interp);
CL_piano_C_fusoliera_timone_equi_0=c_piano_C_fusoliera_timone_equi_0(1).*
alpha + c_piano_C_fusoliera_timone_equi_0(2);


Cl_al2=c_piano_C_equi_0(1)
    % pendenza curva coefficiente di portanza del solo piano di coda
Cl02=c_piano_C_equi_0(2)
    % coefficiente di portanza ad incidenza nulla del solo piano di coda
Cl_al2_fuso_timo=c_piano_C_fusoliera_timone_equi_0(1)
    % pendenza curva di portanza di piano di coda fusoliera e timone.
    % serve per calcolare l'interferenza di timone e fusoliera sul piano
di
    % coda.
dBeta_dAlpha_coda=-(Cl_al2_fuso_timo/Cl_al2 - 1)
    % coefficiente di interferenza aerodinamica della fusoliera sul piano
    % di coda


figure (2)
plot(alpha,CL_piano_C_equi_0,'b',piano_C_equi_0(:,1).*(pi/180),piano_C_eq
ui_0(:,3),'b*')
hold on
```

```
plot(alpha,CL_piano_C_fusoliera_timone_equi_0,'g',piano_C_fusoliera_timon
e_equi_0(:,1).*(pi/180),piano_C_fusoliera_timone_equi_0(:,3),'g*')
xlabel('incidenza piano di coda')
ylabel('coefficiente di portanza piano di coda')
title('PIANO DI CODA :interpolazione dati ricavati con XFLR5 ')
legend('deflessione equilibratore di 0 deg','','deflessione equilibratore
di 0 deg con fusoliera e timone')

% RESISTENZA AERODINAMICA PIANO DI CODA
piano_C_equi_0_CD_interp=fit((piano_C_equi_0(:,1).*(pi/180)),piano_C_equi
_0(:,6),'poly2');
c_piano_C_equi_0_CD=coeffvalues(piano_C_equi_0_CD_interp);
CD_piano_C_equi_0=c_piano_C_equi_0_CD(1).*alpha.^2 +
c_piano_C_equi_0_CD(2).*alpha + c_piano_C_equi_0_CD(3);
CD_piano_C_equi_0_f=@(x) c_piano_C_equi_0_CD(1).*x.^2 +
c_piano_C_equi_0_CD(2).*x + c_piano_C_equi_0_CD(3);

figure(3)
plot(alpha,CD_piano_C_equi_0,'g',piano_C_equi_0(:,1).*(pi/180),piano_C_eq
ui_0(:,6),'g*')
xlabel('incidenza piano di coda')
ylabel('coefficiente di resistenza piano di coda')
title('PIANO DI CODA :interpolazione dati ricavati con XFLR5 ')
```

NOTA 2 Script Matlab per ricavare i coeffcienti aerodinamici d'interesse relativi al piano di coda

ALLEGATO 2.3

```
B=readtable('drone_f_0_e_-5.txt');
f_0_e_n5=table2array(B);
B=readtable('drone_f_0_e_5.txt');
f_0_e_5=table2array(B);
B=readtable('drone_f_0_e_-2.txt');
f_0_e_n2=table2array(B);
B=readtable('drone_f_0_e_2.txt');
f_0_e_2=table2array(B);
B=readtable('drone_f_0_e_0.txt');
f_0_e_0=table2array(B);
B=readtable('drone_f_0_e_0_LPort.txt');
f_0_e_0_LP=table2array(B);
    % input: dati ricavati dalla simulazione del drone completo con XFLR5
    % f = FLAP ANTERIORE; numero = DEFLESSIONE (n sta per meno)
    % e = EQUILIBRATORE; numero = DEFLESSIONE (n sta per meno)
    % LP = LINEA PORTANTE

% PORTANZA e MOMENTO AERODINAMICI di BECCHEGGIO
alpha=(-3*(pi/180)):0.01:(10*(pi/180));

f_0_e_n5_interp=fit((f_0_e_n5(:,1).*(pi/180)),f_0_e_n5(:,3),'poly1');
c_0_n5=coeffvalues(f_0_e_n5_interp);
CL_f_0_e_n5=c_0_n5(1).*alpha + c_0_n5(2);
f_0_e_5_interp=fit((f_0_e_5(:,1).*(pi/180)),f_0_e_5(:,3),'poly1');
c_0_5=coeffvalues(f_0_e_5_interp);
```

```
CL_f_0_e_5=c_0_5(1).*alpha + c_0_5(2);
f_0_e_n2_interp=fit((f_0_e_n2(:,1).*(pi/180)),f_0_e_n2(:,3),'poly1');
c_0_n2=coeffvalues(f_0_e_n2_interp);
CL_f_0_e_n2=c_0_n2(1).*alpha + c_0_n2(2);
f_0_e_2_interp=fit((f_0_e_2(:,1).*(pi/180)),f_0_e_2(:,3),'poly1');
c_0_2=coeffvalues(f_0_e_2_interp);
CL_f_0_e_2=c_0_2(1).*alpha + c_0_2(2);
f_0_e_0_interp=fit((f_0_e_0(:,1).*(pi/180)),f_0_e_0(:,3),'poly1');
c_0_0=coeffvalues(f_0_e_0_interp);
CL_f_0_e_0=c_0_0(1).*alpha + c_0_0(2);

f_0_e_0_LP_interp=fit((f_0_e_0_LP(:,1).*(pi/180)),f_0_e_0_LP(:,3),'poly1'
);
c_0_0_LP=coeffvalues(f_0_e_0_LP_interp);
CL_f_0_e_0_LP=c_0_0_LP(1).*alpha + c_0_0_LP(2);
    % confronto metodo della LINEA PORTANTE

Cl_alphaTOT=c_0_0(1)
    % pendenza curva del coefficiente di portanza del intero velivolo nel
caso della crociera dove non si utilizza il flap
    % anteriore perchÃ¨ genera maggiore resistenza aerodinamica
pendenze_CL=[c_0_n5(1) c_0_5(1) c_0_n2(1)  c_0_2(1)];
err_Cl_alphaTOT=abs(c_0_0(1) - pendenze_CL)
    % VERIFICA se la variazione della pendenza della curva del
    % coefficiente di portanza Ã¨ trascurabile per effetto della
deflessione
    % dell'equilibratore

Cl0=c_0_0(2)
Cl0_LP=c_0_0_LP(2)
    % confronto con il metodo della LINEA PORTANTE

Cl_al_equi=mean([(c_0_0(2) - c_0_n2(2))*0.52*180/(0.07*0.6*2*pi)
(c_0_0(2) - c_0_n5(2))*0.52*180/(0.07*0.6*5*pi)   (c_0_5(2) -
c_0_n2(2))*0.52*180/(0.07*0.6*7*pi)   (c_0_5(2) -
c_0_2(2))*0.52*180/(0.07*0.6*3*pi)   (c_0_5(2) -
c_0_0(2))*0.52*180/(0.07*0.6*5*pi)   (c_0_2(2) -
c_0_0(2))*0.52*180/(0.07*0.6*2*pi)])
    % pendenza della curva di portanza del solo equilibratore
considerando
    % anche la variazione di It

figure (4)
plot(alpha,CL_f_0_e_n5,'r',f_0_e_n5(:,1).*(pi/180),f_0_e_n5(:,3),'r*')
hold on
plot(alpha,CL_f_0_e_5,'b',f_0_e_5(:,1).*(pi/180),f_0_e_5(:,3),'b*')
hold on
plot(alpha,CL_f_0_e_n2,'g',f_0_e_n2(:,1).*(pi/180),f_0_e_n2(:,3),'g*')
hold on
plot(alpha,CL_f_0_e_2,'c',f_0_e_2(:,1).*(pi/180),f_0_e_2(:,3),'c*')
hold on
plot(alpha,CL_f_0_e_0,'y',f_0_e_0(:,1).*(pi/180),f_0_e_0(:,3),'y*')
xlabel('incidenza velivolo completo')
ylabel('coefficiente di portanza totale')
title('DRONE COMPLETO :interpolazione dati ricavati con XFLR5 ')
legend('deflessione equilibratore di -5 deg','','deflessione
equilibratore di 5 deg','','deflessione equilibratore di -2
```

```
deg','','deflessione equilibratore di 2 deg','','deflessione
equilibratore di 0 deg')

f_0_e_n5_interp=fit((f_0_e_n5(:,1).*(pi/180)),f_0_e_n5(:,9),'poly1');
c_0_n5m=coeffvalues(f_0_e_n5_interp);
Cm_f_0_e_n5=c_0_n5m(1).*alpha + c_0_n5m(2);
f_0_e_5_interp=fit((f_0_e_5(:,1).*(pi/180)),f_0_e_5(:,9),'poly1');
c_0_5m=coeffvalues(f_0_e_5_interp);
Cm_f_0_e_5=c_0_5m(1).*alpha + c_0_5m(2);
f_0_e_n2_interp=fit((f_0_e_n2(:,1).*(pi/180)),f_0_e_n2(:,9),'poly1');
c_0_n2m=coeffvalues(f_0_e_n2_interp);
Cm_f_0_e_n2=c_0_n2m(1).*alpha + c_0_n2m(2);
f_0_e_2_interp=fit((f_0_e_2(:,1).*(pi/180)),f_0_e_2(:,9),'poly1');
c_0_2m=coeffvalues(f_0_e_2_interp);
Cm_f_0_e_2=c_0_2m(1).*alpha + c_0_2m(2);
f_0_e_0_interp=fit((f_0_e_0(:,1).*(pi/180)),f_0_e_0(:,9),'poly1');
c_0_0m=coeffvalues(f_0_e_0_interp);
Cm_f_0_e_0=c_0_0m(1).*alpha + c_0_0m(2);

Cm_alphaTOT=c_0_0m(1)
    % pendenza curva del coefficiente di momento di beccheggio
dell'intero velivolo in fase di crociera dove
    % si sfrutta il solo equilibratore per ridurre la resitenza
pendenze_Cm=[c_0_n5m(1)  c_0_5m(1)  c_0_n2m(1)  c_0_2m(1)];
err_Cm_alphaTOT=abs(c_0_0m(1) - pendenze_Cm)
    % VERIFICA se la variazione della pendenza della curva del
    % coefficiente di momento Ã¨ trascurabile per effetto della
deflessione
    % dell'equilibratore

Cm0=c_0_0m(2)
    % coefficiente di momento a portanza nulla. Rappresenta il momento di
    % beccheggio indipendente dall'angolo d'attacco applicato nel centro
    % aerodinamico dell'intera configurazione, ossia il PUNTO NEUTRO


Cl_EQUI_fitt=-mean([ (c_0_n5m(2) -
c_0_n2m(2))*(0.52*0.4*180)/((0.07*0.6)*(3*pi)*0.889)   (c_0_n2m(2) -
c_0_0m(2))*(0.52*0.4*180)/((0.07*0.6)*(2*pi)*0.889)    (c_0_n5m(2) -
c_0_0m(2))*(0.52*0.4*180)/((0.07*0.6)*(5*pi)*0.889)   (c_0_n5m(2) -
c_0_5m(2))*(0.52*0.4*180)/((0.07*0.6)*(10*pi)*0.889)   (c_0_n2m(2) -
c_0_2m(2))*(0.52*0.4*180)/((0.07*0.6)*(4*pi)*0.889)    (c_0_2m(2) -
c_0_5m(2))*(0.52*0.4*180)/((0.07*0.6)*(3*pi)*0.889)   (c_0_0m(2) -
c_0_2m(2))*(0.52*0.4*180)/((0.07*0.6)*(2*pi)*0.889)])
   % tiene conto anche della variazione di Cmo dovuta alla deflessione
   % dell'equilibratore (il centro delle pressioni non coincide piÃ¹ con
il centro aerodinamico perchÃ¨ il profilo Ã¨ curvato)

figure (5)
plot(alpha,Cm_f_0_e_n5,'r',f_0_e_n5(:,1).*(pi/180),f_0_e_n5(:,9),'r*')
hold on
plot(alpha,Cm_f_0_e_5,'b',f_0_e_5(:,1).*(pi/180),f_0_e_5(:,9),'b*')
hold on
plot(alpha,Cm_f_0_e_n2,'g',f_0_e_n2(:,1).*(pi/180),f_0_e_n2(:,9),'g*')
hold on
plot(alpha,Cm_f_0_e_2,'c',f_0_e_2(:,1).*(pi/180),f_0_e_2(:,9),'c*')
hold on
```

```
plot(alpha,Cm_f_0_e_0,'y',f_0_e_0(:,1).*(pi/180),f_0_e_0(:,9),'y*')
xlabel('incidenza velivolo completo')
ylabel('coefficiente di momento totale')
title('DRONE COMPLETO :interpolazione dati ricavati con XFLR5 ')
legend('deflessione equilibratore di -5 deg','','deflessione
equilibratore di 5 deg','','deflessione equilibratore di -2
deg','','deflessione equilibratore di 2 deg','','deflessione
equilibratore di 0 deg')

% coefficienti di DOWNWASH
de_dal= 1 - ( 0.107 )*Cl_alphaTOT/ (0.4*0.209*Cl_al2)
   % 0.107 distanza tra il centro aerodinamico ala_fusoliera e il punto
   % neutro; 0.4 corda media aerodinamica dell'ala principale; 0.209
volume
   % di coda (calcolato con le posizioni dei centri aerodinamici ricavate
da XFLR5 considerando anche l'effetto della fusoliera);
   % Cl_al2 pendenza curva di portanza del piano di coda
e0= (Cm0 - Cm01B)/(0.209*Cl_al2*(1 - (0.042/0.52)*(Cl_al2/Cl_alphaTOT)*(1
- de_dal)))
   % 0.042/0.52 rapporto tra la superficie alare principale e la
superficie
   % alare del piano di coda;


% RESISTENZA AERODINAMICA
f_parabola=fittype(@(c1,c2,c3,x) c1.*x.^2 + c2.*x +c3);

CL_inter=0:0.01:2;
syms x

e_n5_interp_resi=fit(f_0_e_n5(:,3),f_0_e_n5(:,6),f_parabola,'Startpoint',
[0.01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_n5_resi=coeffvalues(e_n5_interp_resi);
CD_e_n5=c_0_n5_resi(1).*CL_inter.^2 + c_0_n5_resi(2).*CL_inter +
c_0_n5_resi(3);
f_err_Cin_e_n5=@(x) c_0_n5_resi(1).*x.^2 + c_0_n5_resi(2).*x +
c_0_n5_resi(3)
err_Cin_e_n5=max( abs( f_err_Cin_e_n5(f_0_e_n5(:,3)) - f_0_e_n5(:,6) ) )

e_5_interp_resi=fit(f_0_e_5(:,3),f_0_e_5(:,6),f_parabola,'Startpoint',[0.
01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_5_resi=coeffvalues(e_5_interp_resi);
CD_e_5=c_0_5_resi(1).*CL_inter.^2 + c_0_5_resi(2).*CL_inter +
c_0_5_resi(3);
f_err_Cin_e_5=@(x) c_0_5_resi(1).*x.^2 + c_0_5_resi(2).*x + c_0_5_resi(3)
err_Cin_e_5=max( abs( f_err_Cin_e_5(f_0_e_5(:,3)) - f_0_e_5(:,6) ) )

e_n2_interp_resi=fit(f_0_e_n2(:,3),f_0_e_n2(:,6),f_parabola,'Startpoint',
[0.01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_n2_resi=coeffvalues(e_n2_interp_resi);
CD_e_n2=c_0_n2_resi(1).*CL_inter.^2 + c_0_n2_resi(2).*CL_inter +
c_0_n2_resi(3);
f_err_Cin_e_n2=@(x) c_0_n2_resi(1).*x.^2 + c_0_n2_resi(2).*x +
c_0_n2_resi(3)
err_Cin_e_n2=max( abs( f_err_Cin_e_n2(f_0_e_n2(:,3)) - f_0_e_n2(:,6) ) )
```

```
e_2_interp_resi=fit(f_0_e_2(:,3),f_0_e_2(:,6),f_parabola,'Startpoint',[0.
01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_2_resi=coeffvalues(e_2_interp_resi);
CD_e_2=c_0_2_resi(1).*CL_inter.^2 + c_0_2_resi(2).*CL_inter +
c_0_2_resi(3);
f_err_Cin_e_2=@(x) c_0_2_resi(1).*x.^2 + c_0_2_resi(2).*x + c_0_2_resi(3)
err_Cin_e_2=max( abs( f_err_Cin_e_2(f_0_e_2(:,3)) - f_0_e_2(:,6) ) )

e_0_interp_resi=fit(f_0_e_0(:,3),f_0_e_0(:,6),f_parabola,'Startpoint',[0.
01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_0_resi=coeffvalues(e_0_interp_resi);
CD_e_0=c_0_0_resi(1).*CL_inter.^2 + c_0_0_resi(2).*CL_inter +
c_0_0_resi(3);
f_err_Cin_e_0=@(x) c_0_0_resi(1).*x.^2 + c_0_0_resi(2).*x + c_0_0_resi(3)
err_Cin_e_0=max( abs( f_err_Cin_e_0(f_0_e_0(:,3)) - f_0_e_0(:,6) ) )

e_0_LP_interp_resi=fit(f_0_e_0_LP(:,3),f_0_e_0_LP(:,6),f_parabola,'Startp
oint',[0.01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_0_0_LP_resi=coeffvalues(e_0_LP_interp_resi);
CD_e_0_LP=c_0_0_LP_resi(1).*CL_inter.^2 + c_0_0_LP_resi(2).*CL_inter +
c_0_0_LP_resi(3);
f_err_Cin_e_0_LP=@(x) c_0_0_LP_resi(1).*x.^2 + c_0_0_LP_resi(2).*x +
c_0_0_LP_resi(3)
err_Cin_e_0_LP=max( abs( f_err_Cin_e_0_LP(f_0_e_0_LP(:,3)) -
f_0_e_0_LP(:,6) ) )


k=mean([c_0_n5_resi(1)  c_0_5_resi(1)  c_0_n2_resi(1)  c_0_2_resi(1)
c_0_0_resi(1)])
    % coefficiente di resitenza indotta dell'intero velivolo in fase di
    % crociera
k1=mean([c_0_n5_resi(2)  c_0_5_resi(2)  c_0_n2_resi(2)  c_0_2_resi(2)
c_0_0_resi(2)])
    % coefficiente di resistenza derivante dal modello matematico. Se lo
si
    % trascura, l'interpolazione si discosta di un valore massimo non
    % trascurabile
Cd0=mean([c_0_n5_resi(3)  c_0_5_resi(3)  c_0_n2_resi(3)  c_0_2_resi(3)
c_0_0_resi(3)])
    % coefficiente di resistenza di forma e di attrito

k_LP=c_0_0_LP_resi(1)
Cd0_LP=c_0_0_LP_resi(3)
    % confronto metodo della LINEA PORTANTE

figure (6)
plot(CL_inter,CD_e_n5,'b',f_0_e_n5(:,3),f_0_e_n5(:,6),'b*')
hold on
plot(CL_inter,CD_e_5,'r',f_0_e_5(:,3),f_0_e_5(:,6),'r*')
hold on
plot(CL_inter,CD_e_n2,'y',f_0_e_n2(:,3),f_0_e_n2(:,6),'y*')
hold on
plot(CL_inter,CD_e_2,'c',f_0_e_2(:,3),f_0_e_2(:,6),'c*')
hold on
plot(CL_inter,CD_e_0,'g',f_0_e_0(:,3),f_0_e_0(:,6),'g*')
xlabel('coefficiente di portanza totale')
ylabel('coefficiente di resistenza totale')
```

```
title('DRONE COMPLETO :interpolazione dati ricavati con XFLR5 ')
legend('deflessione equilibratore di -5 gradi','','deflessione
equilibratore di 5 gradi','','deflessione equilibratore di -2
gradi','','deflessione equilibratore di 2 gradi','','deflessione
equilibratore di 0 gradi')

figure(7)
plot(CL_inter,CD_e_0,'g',f_0_e_0(:,3),f_0_e_0(:,6),'g*')
hold on
plot(CL_inter,CD_e_0_LP,'b',f_0_e_0_LP(:,3),f_0_e_0_LP(:,6),'b*')
xlabel('coefficiente di portanza totale')
ylabel('coefficiente di resistenza totale')
title('DRONE COMPLETO :interpolazione dati ricavati con XFLR5, confronto
tra metodo Vortex Lattice e metodo della Linea Portante ')
legend('Vortex Lattice','','Linea Portante')


% FLAP ANTERIORE (necessario per la rotazione)
B=readtable('drone_f_5_e_0.txt');
f_5_e_0=table2array(B);
B=readtable('drone_f_2_e_0.txt');
f_2_e_0=table2array(B);
B=readtable('drone_f_-5_e_0.txt');
f_n5_e_0=table2array(B);
B=readtable('drone_f_-2_e_0.txt');
f_n2_e_0=table2array(B);
    % input: dati ricavati dalla simulazione del velivolo completo con
flap
    % anteriore deflesso

f_5_e_0_interp=fit((f_5_e_0(:,1).*(pi/180)),f_5_e_0(:,3),'poly1');
c_5_0=coeffvalues(f_5_e_0_interp);
CL_f_5_e_0=c_5_0(1).*alpha + c_5_0(2);
f_2_e_0_interp=fit((f_2_e_0(:,1).*(pi/180)),f_2_e_0(:,3),'poly1');
c_2_0=coeffvalues(f_2_e_0_interp);
CL_f_2_e_0=c_2_0(1).*alpha + c_2_0(2);
f_n5_e_0_interp=fit((f_n5_e_0(:,1).*(pi/180)),f_n5_e_0(:,3),'poly1');
c_n5_0=coeffvalues(f_n5_e_0_interp);
CL_f_n5_e_0=c_n5_0(1).*alpha + c_n5_0(2);
f_n2_e_0_interp=fit((f_n2_e_0(:,1).*(pi/180)),f_n2_e_0(:,3),'poly1');
c_n2_0=coeffvalues(f_n2_e_0_interp);
CL_f_n2_e_0=c_n2_0(1).*alpha + c_n2_0(2);

Cl_alphaTOT2=mean([c_2_0(1)  c_5_0(1)  c_n2_0(1)  c_n5_0(1)])
    % verifica della variazione di Cl_alphaTOT dovuta alla variazione di
    % de_dalpha e Cl_al1 (la media di Cl_al1 varia per effetto della
separazione ad alte incidenze, stesso vale per Cl_al2. Entrambe le
pendenze decrescono all'aumentare della deflessione delle superfici di
controllo)

figure (8)
plot(alpha,CL_f_5_e_0,'r',f_5_e_0(:,1).*(pi/180),f_5_e_0(:,3),'r*')
hold on
plot(alpha,CL_f_2_e_0,'b',f_2_e_0(:,1).*(pi/180),f_2_e_0(:,3),'b*')
hold on
plot(alpha,CL_f_n5_e_0,'c',f_n5_e_0(:,1).*(pi/180),f_n5_e_0(:,3),'c*')
hold on
```

```
plot(alpha,CL_f_n2_e_0,'y',f_n2_e_0(:,1).*(pi/180),f_n2_e_0(:,3),'y*')
hold on
plot(alpha,CL_f_0_e_0,'g',f_0_e_0(:,1).*(pi/180),f_0_e_0(:,3),'g*')
xlabel('incidenza velivolo completo')
ylabel('coefficiente di portanza')
legend('deflessione flap: 5 deg','','deflessione flap: 2
deg','','deflessione flap: -5 deg','','deflessione flap: -2
deg','','deflessione flap : 0 deg')


Cl_FLAP=mean([(c_5_0(2) - Cl0)/(5*(pi/180))   (c_2_0(2) -
Cl0)/(2*(pi/180))   (c_5_0(2) - c_2_0(2))/(3*(pi/180))  (c_2_0(2) -
c_n5_0(2))/(7*(pi/180))  (c_5_0(2) - c_n5_0(2))/(10*(pi/180))  (c_5_0(2)
- c_n2_0(2))/(7*(pi/180))  (c_0_0(2) - c_n5_0(2))/(5*(pi/180))])
    % tiene conto anche della variazione di Cl0 dovuta alla variazione di
    % e0 e It

f_5_e_0_interp=fit((f_5_e_0(:,1).*(pi/180)),f_5_e_0(:,9),'poly1');
c_5_0m=coeffvalues(f_5_e_0_interp);
Cm_f_5_e_0=c_5_0m(1).*alpha + c_5_0m(2);
f_2_e_0_interp=fit((f_2_e_0(:,1).*(pi/180)),f_2_e_0(:,9),'poly1');
c_2_0m=coeffvalues(f_2_e_0_interp);
Cm_f_2_e_0=c_2_0m(1).*alpha + c_2_0m(2);
f_n5_e_0_interp=fit((f_n5_e_0(:,1).*(pi/180)),f_n5_e_0(:,9),'poly1');
c_n5_0m=coeffvalues(f_n5_e_0_interp);
Cm_f_n5_e_0=c_n5_0m(1).*alpha + c_n5_0m(2);
f_n2_e_0_interp=fit((f_n2_e_0(:,1).*(pi/180)),f_n2_e_0(:,9),'poly1');
c_n2_0m=coeffvalues(f_n2_e_0_interp);
Cm_f_n2_e_0=c_n2_0m(1).*alpha + c_n2_0m(2);

Cm_alphaTOT2=mean([c_2_0m(1)  c_5_0m(1)  c_n2_0m(1)   c_n5_0m(1)
c_0_0m(1)])
    % verifica della pendenza del coefficiente di momento perchÃ¨ la
    % deflessione del flap varia la posizione del centro aerodinamico di
    % ala-fusoliera e la scia dell'ala principale (d_e_dal)

figure (9)
plot(alpha,Cm_f_5_e_0,'r',f_5_e_0(:,1).*(pi/180),f_5_e_0(:,9),'r*')
hold on
plot(alpha,Cm_f_2_e_0,'b',f_2_e_0(:,1).*(pi/180),f_2_e_0(:,9),'b*')
hold on
plot(alpha,Cm_f_n5_e_0,'c',f_n5_e_0(:,1).*(pi/180),f_n5_e_0(:,9),'c*')
hold on
plot(alpha,Cm_f_n2_e_0,'y',f_n2_e_0(:,1).*(pi/180),f_n2_e_0(:,9),'y*')
hold on
plot(alpha,Cm_f_0_e_0,'g',f_0_e_0(:,1).*(pi/180),f_0_e_0(:,9),'g*')
xlabel('incidenza velivolo completo')
ylabel('coefficiente di momento')
legend('deflessione flap: 5 deg','','deflessione flap: 2
deg','','deflessione flap: -5 deg','','deflessione flap: -2
deg','','deflessione flap: 0 deg')


Cl_FLAP_fitt=mean([(c_5_0m(2) - Cm0)*0.4/(0.0616*5*(pi/180))  (c_2_0m(2)
- Cm0)*0.4/(0.0616*2*(pi/180))  (c_5_0m(2) -
c_2_0m(2))*0.4/(0.0616*3*(pi/180))   (Cm0 -
c_n5_0m(2))*0.4/(0.0616*5*(pi/180))   (Cm0 -
```

```
c_n2_0m(2))*0.4/(0.0616*2*(pi/180))   (c_2_0m(2) -
c_n5_0m(2))*0.4/(0.0616*7*(pi/180))   (c_2_0m(2) -
c_n2_0m(2))*0.4/(0.0616*4*(pi/180))])
   % tiene conto che la deflessione del flap anteriore varia anche
   % Cm0, oltre al momento generato dal flap stesso

% Verifica variazione RESISTENZA AERODINAMICA con flap anteriore deflesso
f_n5_interp_resi=fit(f_n5_e_0(:,3),f_n5_e_0(:,6),f_parabola,'Startpoint',
[0.01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_n5_0_resi=coeffvalues(f_n5_interp_resi);
CD_f_n5=c_n5_0_resi(1).*CL_inter.^2 + c_n5_0_resi(2).*CL_inter +
c_n5_0_resi(3);
f_err_Cin_f_n5=@(x) c_n5_0_resi(1).*x.^2 + c_n5_0_resi(2).*x +
c_n5_0_resi(3)
err_Cin_f_n5=max( abs( f_err_Cin_f_n5(f_n5_e_0(:,3)) - f_n5_e_0(:,6) ) )


f_5_interp_resi=fit(f_5_e_0(:,3),f_5_e_0(:,6),f_parabola,'Startpoint',[0.
01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_5_0_resi=coeffvalues(f_5_interp_resi);
CD_f_5=c_5_0_resi(1).*CL_inter.^2 + c_5_0_resi(2).*CL_inter +
c_5_0_resi(3);
f_err_Cin_f_5=@(x) c_5_0_resi(1).*x.^2 + c_5_0_resi(2).*x + c_5_0_resi(3)
err_Cin_f_5=max( abs( f_err_Cin_f_5(f_5_e_0(:,3)) - f_5_e_0(:,6) ) )


f_n2_interp_resi=fit(f_n2_e_0(:,3),f_n2_e_0(:,6),f_parabola,'Startpoint',
[0.01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_n2_0_resi=coeffvalues(f_n2_interp_resi);
CD_f_n2=c_n2_0_resi(1).*CL_inter.^2 + c_n2_0_resi(2).*CL_inter +
c_n2_0_resi(3);
f_err_Cin_f_n2=@(x) c_n2_0_resi(1).*x.^2 + c_n2_0_resi(2).*x +
c_n2_0_resi(3)
err_Cin_f_n2=max( abs( f_err_Cin_f_n2(f_n2_e_0(:,3)) - f_n2_e_0(:,6) ) )


f_2_interp_resi=fit(f_2_e_0(:,3),f_2_e_0(:,6),f_parabola,'Startpoint',[0.
01  0  0],'Lower',[0.001  0  0.0001],'Upper',[0.3  0.1  0.01]);
c_2_0_resi=coeffvalues(f_2_interp_resi);
CD_f_2=c_2_0_resi(1).*CL_inter.^2 + c_2_0_resi(2).*CL_inter +
c_2_0_resi(3);
f_err_Cin_f_2=@(x) c_2_0_resi(1).*x.^2 + c_2_0_resi(2).*x + c_2_0_resi(3)
err_Cin_f_2=max( abs( f_err_Cin_f_2(f_2_e_0(:,3)) - f_2_e_0(:,6) ) )


k_flap=mean([c_0_0_resi(1)  c_2_0_resi(1)  c_n2_0_resi(1)  c_5_0_resi(1)
c_n5_0_resi(1)]);
k1_flap=mean([c_0_0_resi(2)  c_2_0_resi(2)  c_n2_0_resi(2)  c_5_0_resi(2)
c_n5_0_resi(2)]);
Cd0_flap=mean([c_0_0_resi(3)  c_2_0_resi(3)  c_n2_0_resi(3)
c_5_0_resi(3)  c_n5_0_resi(3)]);
```

NOTA 3 Script Matlab per ricavare i coefficienti aerodinamici d'interesse relativi al drone completo

ALLEGATO 2.4

```
alpha_max=(12.5 + 0.06)*(pi/180);
  % somma dell'angolo di stallo dell'ala principale e della differenza
tra
  % incidenza del drone completo e dell'ala principale

A_CLmax_trim=[-1 , (S2/S1)*Cl_al_equi ; 0 ,
(S2/S1)*Cl_EQUI_fitt*(d2/corda1) ];
b_CLmax=[-Cl_alphaTOT*alpha_max   -Cm_alphaTOT*alpha_max];
sol_CLmax=inv(A_CLmax_trim)*b_CLmax'
CL_max=sol_CLmax(1,1);

TAS_stallo=sqrt((2*m_tot*g)/(S1*rho*CL_max));
 % il CL_max Ã¨ stato ricavato indirettamente perchÃ¨
 % XFLR5 non permette di usare il metodo lineare per computare la
 % configurazione completa. Il metodo usato Ã¨ stato spiegato nel
capitolo 7
```

NOTA 4 Script Matlab per ricavare il coefficiente di portanza massimo del drone completo

ALLEGATO 3

```
P=4*2.56*0.01;
R_pala=3*2.56*0.01;
  % rispettivamente passo geometrico e diametro della pala dell'elica
  % scelta.Dopo una prima iterazione Ã¨ stato scelto il motore adatto il
  % quale Ã¨ stato testato con un'elica con le sovrastanti
caratteristiche.

r_ava_inter=0:0.01:1;

% MODELLO EMPIRICO
beta=P/R_pala;
  % parametro da cui dipende il modello empirico
J0t=0.4559*beta + 0.1574;
dCt_max=0.0205*beta - 0.0073;
Jmt=0.2275*beta + 0.0792;
Ct0=-0.034*beta^2 + 0.13306*beta - 0.00287;
bT=inv([2*Jmt 1 0; Jmt^2 Jmt 0; 0 0 1])*[(-Ct0/J0t) (dCt_max-
Ct0*(Jmt/J0t)) Ct0]';
bT2=bT(1);
bT1=bT(2);
bT0=bT(3);

J0p=0.3839*beta + 0.3553;
dCp_max=0.0249*beta - 0.0117;
Jmp=0.267*beta + 0.1192;
Cp0=0.017716*beta^2 + 0.0064*beta + 0.014084;
bP=inv([J0p^3 J0p^2 J0p 1; Jmp^3 Jmp^2 Jmp 0; 3*Jmp^2 2*Jmp 1 0; 0 0 0
1])*[0 (dCp_max-(Cp0/J0p)*Jmp) -(Cp0/J0p) Cp0]';
bP3=bP(1);
bP2=bP(2);
bP1=bP(3);
```

```
bP0=bP(4);

CT_empirico=bT2*r_ava_inter.^2 + bT1*r_ava_inter + bT0;
CP_empirico=bP3*r_ava_inter.^3 + bP2*r_ava_inter.^2 + bP1*r_ava_inter +
bP0;


% DATI DERIVANTI DA UN TEST IN GALLERIA DEL VENTO (BART: Basic
Aerodynamics Research Tunnel)
BART_CT=[0.12 0.082 0.07 0.054 0.042 0.03 0.015 0.018 0.0075];
BART_rAva_CT=[0 0.24 0.4 0.43 0.48 0.52 0.605 0.625 0.665];
  % dati relativi al coefficiente di spinta
BART_CP=[ 0.0082 0.0081 0.0075 0.0066 0.006 0.0052 0.0047 0.0042 0.0038
0.003]*2*pi;
BART_rAva_CP=[ 0.24 0.315 0.43 0.48 0.52 0.605 0.625 0.625 0.665 0.75];
  % dati relativi al coeffciente di potenza resistente

BART_int_CT=fit(BART_rAva_CT',BART_CT','poly2');
c_BART_CT=coeffvalues(BART_int_CT);
CT_BART=c_BART_CT(1)*r_ava_inter.^2 + c_BART_CT(2)*r_ava_inter +
c_BART_CT(3);
BART_int_CP=fit(BART_rAva_CP',BART_CP','poly2');
c_BART_CP=coeffvalues(BART_int_CP);
CP_BART=c_BART_CP(1)*r_ava_inter.^2 + c_BART_CP(2)*r_ava_inter +
c_BART_CP(3);
  % interpolazione dati BART

CT_elica=@(x) c_BART_CT(1)*x.^2 + c_BART_CT(2)*x + c_BART_CT(3);
CP_elica=@(x) c_BART_CP(1)*x.^2 + c_BART_CP(2)*x + c_BART_CP(3);
  % modello scelto per lo studio oggetto di questa tesi. PoichÃ¨ Ã¨ stata
  % testata dal BART un'elica con le stesse caratteristiche geometriche,
  % sono stati ritenuti piÃ¹ accurati quest'ultimi

zero=ones(1,length(r_ava_inter))*0;
figure(10)
plot(r_ava_inter,CT_empirico,'b',r_ava_inter,CP_empirico,'r')
minimo_empirico=find(min(abs(CT_empirico)) == abs(CT_empirico));
EFFI_empirica=(CT_empirico(1:minimo_empirico)./CP_empirico(1:minimo_empir
ico)).*r_ava_inter(1:minimo_empirico);
hold on
plot(r_ava_inter(1:minimo_empirico),EFFI_empirica,'g',r_ava_inter,zero,'b
lack')
xlabel('rapporto di avanzamento V/(n*D0)')
legend('coefficiente di spinta','coefficiente di potenza
resistente','efficienza')
title('MODELLO EMPIRICO')

figure(11)
plot(r_ava_inter,CT_BART,'c',BART_rAva_CT(2:end),BART_CT(2:end),'c*',r_av
a_inter,CP_BART,'m',BART_rAva_CP(2:end),BART_CP(2:end),'m*',BART_rAva_CT(
1),BART_CT(1),'r*')
minimo_BART=find(min(abs(CT_BART)) == abs(CT_BART));
EFFI_BART=(CT_BART(1:minimo_BART)./CP_BART(1:minimo_BART)).*r_ava_inter(1
:minimo_BART);
hold on
plot(r_ava_inter(1:minimo_BART),EFFI_BART,'y',r_ava_inter,zero,'black')
xlabel('rapporto di avanzamento V/(n*D0)')
```

```
legend('intepolazione coefficiente di spinta','dati BART sul coefficiente
di spinta','interpolazione coefficiente di potenza resistente','dati BART
sul coefficiente di potenza','dato derivante dal test del motore
scelto','efficienza')
title('DATI SPERIMENTALI DEL BART')
```

NOTA 5 Script Matlab per ricavare i coefficienti d'interesse relativi all'elica

ALLEGATO 4

```
format short

dati_MOTORE_effi=[0.85  0.87  0.85  0.69];
dati_MOTORE_CORRENTE_effi=[1.5  4.3  7  23.3];

dati_MOTORE_COPPIA=[0  0.0796  0.2106];
dati_MOTORE_CORRENTE_coppia=[0  4.3 23.3];

dati_MOTORE_POTENZA_input=[531.24  98.04];
dati_MOTORE_POTENZA_output=[368  85.3];
dati_MOTORE_POTENZA_input_CORRENTE=[23.3  4.3 ];
dati_MOTORE_V_ANGOLARE=[0  277.58  170.57 ]; % [giri/s]


tensione_MOTORE_test=24;
KV_MOTORE=1350;
R_interna_MOTORE=0.09; % [ohm]
  % dati MOTORE 1350 KV

dati_MOTORE_COPPIA_1850=[0  0.099  0.254];
dati_MOTORE_CORRENTE_coppia_1850=[0  7.5  35.5];
  % dati MOTORE 1850 KV

  % I DUE MOTORI SOVRASTANTI SONO LO STESSO MODELLO

I=0:0.01:35.5;


% COPPIA
  % sono stati usati i dati derivante dal test e i coefficienti ricavati
dal
  % BART per ricavare la coppia nelle due condizioni testate.
  % E' stato scelto un modello lineare ispirandosi ad una tesi.

f_coppia=fittype(@(c1,x) c1.*x.^0.575);
   % in letteratura Ã¨ stato trovato un modello matematico lineare della
   % coppia di un motore brushless in funzione della corrente. Purtroppo
   % questo modello non soddisfava i dati fornitori dal costruttore, per
   % questo Ã¨ stato cambiato il modello analitico.
coppia_MOTORE0_inter=fit(dati_MOTORE_CORRENTE_coppia',dati_MOTORE_COPPIA'
,f_coppia,'Startpoint',0.5,'Lower',0,'Upper',5);
c_MOTORE_COPPIA=coeffvalues(coppia_MOTORE0_inter);
C_motore_=c_MOTORE_COPPIA(1)*I.^0.575 ;
```

```
C_motore=@(x) c_MOTORE_COPPIA(1)*x.^0.575 ;
   % coppia motore al variare della corrente 1350KV

f_coppia2=fittype(@(c1,x) c1.*x.^0.6);
   % in letteratura Ã¨ stato trovato un modello matematico lineare della
   % coppia di un motore brushless in funzione della corrente. Purtroppo
   % questo modello non soddisfava i dati fornitori dal costruttore, per
   % questo Ã¨ stato cambiato il modello analitico.
coppia_MOTORE2_inter=fit(dati_MOTORE_CORRENTE_coppia_1850',dati_MOTORE_CO
PPIA_1850',f_coppia2,'Startpoint',0.5,'Lower',0,'Upper',5);
c_MOTORE_COPPIA2=coeffvalues(coppia_MOTORE2_inter);
C_motore_2=c_MOTORE_COPPIA2(1)*I.^0.6 ;
C_motore2=@(x) c_MOTORE_COPPIA2(1)*x.^0.6 ;



effi_motore(1)=0;
C_effi(1)=0;
n_effi(1)=0;
for i=430:length(I)

    C_effi(i-428)=C_motore(I(i));
    n_effi(i-428)=sqrt( C_effi(i-428) *
1/(3*rho*(2*R_pala)^5*(CP_BART(1)/(2*pi))) );
    effi_motore(i-428)=(C_effi(i-428)*2*pi*n_effi(i-
428))/(I(i)*tensione_MOTORE_test*effi_ESC_cavi);
      % l'efficienza del motore Ã¨ stata calcolata tramite il rapporto
tra
      % la potenza meccanica in uscita e la potenza elettrica in ingresso

end



% MANETTA
%   SarÃ  utile conoscere l'andamento del numero di giri a
%   regime al variare della manetta.La scelta di un polinomio di secondo
grado deriva dalla dipendenza
%   quadratica della coppia resistente dalla velocitÃ  angolare.

MANETTA=0:0.1:100;
N_MANETTA_inter=fit([0  50  100]',[0  170.57  277.58]','poly2');
c_N_MANETTA=coeffvalues(N_MANETTA_inter);
N_MANETTA_par=c_N_MANETTA(1)*MANETTA.^2 + c_N_MANETTA(2)*MANETTA +
c_N_MANETTA(3);
N_MANETTA=@(x) c_N_MANETTA(1)*x.^2 + c_N_MANETTA(2)*x + c_N_MANETTA(3);
figure(39)
plot(MANETTA,N_MANETTA_par,'b',[0  50  100],[0  170.57  277.58],'b*')



effi_MOTORE_inter=fit(I([1 430:end])',effi_motore','poly7');
c_effi_MOTORE=coeffvalues(effi_MOTORE_inter);
EFFI_MOTORE_=c_effi_MOTORE(1)*I.^7 + c_effi_MOTORE(2)*I.^6 +
c_effi_MOTORE(3)*I.^5 + c_effi_MOTORE(4)*I.^4 + c_effi_MOTORE(5)*I.^3 +
c_effi_MOTORE(6)*I.^2 + c_effi_MOTORE(7)*I + c_effi_MOTORE(8);;
```

```
EFFI_MOTORE=@(x) c_effi_MOTORE(1)*x.^7 + c_effi_MOTORE(2)*x.^6 +
c_effi_MOTORE(3)*x.^5 + c_effi_MOTORE(4)*x.^4 + c_effi_MOTORE(5)*x.^3 +
c_effi_MOTORE(6)*x.^2 + c_effi_MOTORE(7)*x + c_effi_MOTORE(8);
   % efficienza motore al variare della corrente

figure(40)
plot(I,EFFI_MOTORE_,'r',I([1  430:end]),effi_motore,'r*',I([1 430
2331]),[0  0.87  0.69],'bo')
xlabel('corrente')
ylabel('efficienza')
legend('interpolazione','dati ricavati analiticamente','dati forniti dal
costruttore')

figure(41)
plot(I,C_motore_,'b',dati_MOTORE_CORRENTE_coppia,dati_MOTORE_COPPIA,'b*')
hold on
plot(I,C_motore_2,'r',dati_MOTORE_CORRENTE_coppia_1850,dati_MOTORE_COPPIA
_1850,'r*')
xlabel('corrente')
ylabel('coppia')
legend('interpolazione 1350KV','dati forniti dal costruttore
1350KV','interpolazione 1850KV','dati forniti dal costruttore 1850KV')
```

NOTA 6 Script Matlab per ricavare le curve caratteristiche del motore scelto in condizioni statiche

ALLEGATO 5

```
% TAKE OFF
k_tas=1.1;
d_Ti=0;

T=0.5*rho*S1*(Cd0)*(TAS_stallo*k_tas)^2 + m_tot*g; % in salita non serve
portanza (CL=0)
    % fase di fine salita: SPINTA_ELICA (T) = F_PESO + R_AERO;
    % valore di partenza per la spinta necessaria

 % spinta elica con coefficiente di spinta (CT)
R0=0.75*R_pala;
f_omega_TakeOff=@(x) (T+d_Ti)/n_pale - (rho* (R_pala*2)^4 * (x/(2*pi))^2
* (c_BART_CT(1)*((TAS_stallo*k_tas*pi)/(x*R0))^2 +
c_BART_CT(2)*((TAS_stallo*k_tas*pi)/(x*R0)) + c_BART_CT(3)));
omega_elica=Newton_tg(N_MANETTA(100)*2*pi,f_omega_TakeOff);
     % velocitÃ  angolare necessaria per generare T,quindi permette
     % l'equilibrio al raggiungimento della velocitÃ  di fine salita
scelta
if double(subs(f_omega_TakeOff,omega_elica)) > 10^-3
fprintf('errore, metodo di Newton non ha trovato una soluzione accetabile
al passo %.f',i)
end
if image(omega_elica) == 0
    disp('velocitÃ  angolare trovata Ã¨ un numero REALE')
end
   % verifiche della soluzione trovata
```

```
v_to(1)=0;
z_to(1)=0;
CT(1)=double(subs(CT_elica,(v_to(1)*pi)/(omega_elica*R0)));
CP(1)=double(subs(CP_elica,(v_to(1)*pi)/(omega_elica*R0)));
    % le 4 condizioni iniziali. L' equazione della dinamica completa non
    % si puÃ² risolvere in maniera diretta, per questo Ã¨ stato
    % utilizzato un metodo discreto

delta_t=0.01;
   % intervallo di discretizzazione
t=0:delta_t:30;

for i=2:length(t)
    % soluzione equazione differenziale ricavata da frwiki.wiki (caduta
con
    % resistenza dell'aria) e riadatta al caso della salita
r_Avanzamento(i-1)=(v_to(i-1)*pi)/(omega_elica*R0);
G=rho*(omega_elica/(2*pi))^2* CT(i-1)* (R_pala*2)^4* n_pale/m_tot - g ;
V0=sqrt((2*G*m_tot)/(rho*S1*Cd0));
v_to(i)=V0*tanh(delta_t*(G/V0)) + v_to(i-1);
z_to(i)=(V0^2/G)*log(cosh(delta_t*(G/V0))) + z_to(i-1) + v_to(i-
1)*delta_t;

CT(i)=double(subs(CT_elica,(v_to(i)*pi)/(omega_elica*R0)));
CP(i)=double(subs(CP_elica,(v_to(i)*pi)/(omega_elica*R0)));

T_pala_elica(i-1)=rho*((omega_elica/(2*pi))^2)* CT(i-1)* (R_pala*2)^4 ;
P_resistente(i-1)=rho*(omega_elica/(2*pi))^3* (R_pala*2)^5* CP(i-1)
*(n_pale/n_motori);
C_resistente(i-1)=P_resistente(i-1)/omega_elica;


i_I_salita(i-1)=find(min(abs(C_motore_ - C_resistente(i-1))) ==
abs(C_motore_ - C_resistente(i-1)));
I_salita(i-1)=I(i_I_salita(i-1));
   % dai dati fornito dal costruttore Ã¨ stato ricavato l'andamento della
   % coppia la variare della corrente in input. Da esso Ã¨ possibile
   % ricavare la corrente minima in uscita dalla batteria per contrastare
   % la coppia resistente dell'elica


P_salita(i-1)=P_resistente(i-1)* (1/(0.745*effi_ESC_cavi));
    % dai dati forniti dal costruttore Ã¨ stata calcolata l'efficienza
per
    % il 100% della manetta (0.69). PoichÃ¨ la spinta iniziale in fase di
    % salita Ã¨ simile a quella fornita dal costruttore per la massima
    % manetta, Ã¨ stato ipotizzato lo stesso rendimento. Tuttvia il
    % costruttore ha testato il motore a 24 Volts quindi Ã¨ correto
    % moltiplicare il rendimento trovato per (24/22.2) per tener conto
    % della differente potenza in ingresso
 C_motore_salita(i-1)=C_motore(P_salita(i-1)/tensione0);
    % PoichÃ¨ la tensione all'uscita della batteria Ã¨ costante,
significa
    % che per fornire l'intera potenza necessaria dovrÃ  variare la
    % corrente. PoichÃ¨ la coppia, in prima approssimazione, dipende solo
dalla corrente
```

```
    % significa che essa dipenderÃ  dalla potenza necessaria in
    % quell'istante



if abs(T_pala_elica(i-1)*n_pale -m_tot*g -0.5*rho*S1*Cd0*(v_to(i-1))^2) <
9*10^-4

    regime=i-1;
    fprintf("il drone Ã¨ andato a regime in %.3f secondi\n",t(i-1))
    break
    % per verificare quando la forza trainante Ã¨ uguale a quella
    % resistente. Il modello usato sovrastima la velocitÃ  di equilibrio
    % perchÃ¨ in G non Ã¨ contenuta la resistenza aerodinamica.
else
    regime=i;
end

end


if regime==length(t)
    fprintf('in %f secondi, il drone non arriva a regime \n',t(end))
end

d_TAS_min=find(abs(TAS_stallo*k_tas - v_to)==min(abs(TAS_stallo*k_tas -
v_to)));
t_end_salita=t(d_TAS_min);
h_end_salita=z_to(d_TAS_min);
KV=(omega_elica*60)/(2*pi*tensione0);
    %rpm / volt, parametro importante per la scelta del motore elettrico

fprintf('la potenza massima richiesta alla batteria, per motore, Ã¨ %.3f,
la media Ã¨ %.3f e serve KV pari a %.3f considerando una batteria LIPO 6S
\n',max(P_salita),mean(P_salita),KV);
fprintf('il drone arriva alla velocitÃ  scelta in %.3f secondi e ad una
quota pari a %.3f metri',t_end_salita,h_end_salita)

figure (13)
plot(t(1:(regime+1)),v_to,'r',t(regime+1),v_to(regime+1),'r*')
xlabel('tempo','FontSize',10)
ylabel('velocitÃ ','FontSize',10)
figure (14)
plot(t(1:(regime+1)),z_to,'b',t(regime+1),z_to(regime+1),'b*')
xlabel('tempo','FontSize',10)
ylabel('spazio','FontSize',10)
figure (15)
plot(t(1:d_TAS_min),P_salita(1:d_TAS_min),'*')
xlabel('tempo','FontSize',10)
ylabel('potenza necessaria','FontSize',10)



Psalita_interp=fit(t(1:d_TAS_min)',P_salita(1:d_TAS_min)','poly6');
coeff_Psalita_sesto=coeffvalues(Psalita_interp);
Psalita_sesto=t(1:d_TAS_min).^6 * coeff_Psalita_sesto(1) +
t(1:d_TAS_min).^5 * coeff_Psalita_sesto(2) +
```

```
coeff_Psalita_sesto(3)*t(1:d_TAS_min).^4 +
coeff_Psalita_sesto(4)*t(1:d_TAS_min).^3 +
coeff_Psalita_sesto(5)*t(1:d_TAS_min).^2 +
coeff_Psalita_sesto(6)*t(1:d_TAS_min) + coeff_Psalita_sesto(7);
f_sesta_Psalita=@(t) t.^6 .* coeff_Psalita_sesto(1) + t.^5 .*
coeff_Psalita_sesto(2) + coeff_Psalita_sesto(3)*t.^4 +
coeff_Psalita_sesto(4).*t.^3 + coeff_Psalita_sesto(5).*t.^2+
coeff_Psalita_sesto(6)*t + coeff_Psalita_sesto(7);
hold on
plot(t(1:d_TAS_min),Psalita_sesto)
legend('potenza reale','interpolazione')

Etot_salita=integral(f_sesta_Psalita,0,t(d_TAS_min)) ;
  % energia spesa in fase di salita


figure(16)
plot(t(1:d_TAS_min),C_resistente(1:d_TAS_min),'r',t(1:d_TAS_min),C_motore
_salita(1:d_TAS_min),'bo')
xlabel('tempo','FontSize',10)
ylabel('Coppia','FontSize',10)
legend('Coppia resistente elica','Coppia motore output')
  % confronto tra la coppia gnerata dal motore e la coppia resistente
  % dell'elica in goni istante della salita.
```

NOTA 7 Script Matlab per ricavare le variabili d'interesse durante il take-off

ALLEGATO 6

```
function y=Newton_tg(x0,s)
syms x
ds=diff(s,x,1);
s0(1)=double(subs(s,x0));
ds0(1)=double(subs(ds,x0));
k(1)=x0;
i=1;
error=1;
while abs(error) > 10^-4 & i<60  & abs(real(double(subs(s,k(i))))) > 10^-
3
    k(i+1)=k(i)-s0(i)/ds0(i);
    s0(i+1)=double(subs(s,k(i+1)));
    ds0(i+1)=double(subs(ds,k(i+1)));
    i=i+1;
    error=k(i)-k(i-1);
    if ds0(i)==0
        break
    end

end
y=k(i);
```

NOTA 8 Script Matlab rappresentante il metodo per trovare gli zeri di Newton (metodo delle tangenti)

ALLEGATO 7

ALLEGATO 7.1

```
% ROTAZIONE
omega=(TAS_stallo*k_tas)/R_manovra; % ipotesi di loop perfetto

gamma=pi/2 :0.01:pi;
    % gamma Ã¨ l'angolo tra asse longitudinale del drone e piano
orizzontale
    % locale
for i=1:length(gamma)
    % asse trasversale positivo verso destra,asse longitudinale verso
poppa
    % e asse verticale verso l'alto.
    % Sistema di riferimeto assi corpo centrato nel baricentro
    % E' stata considerata una manovra verso sinistra
b1_rota=(S2/S1)*V2_V^2*Cl_al2*(omega*d2)/(V2_V*TAS_stallo*k_tas) +
Cl_al1_fuso*(omega*d1)/(TAS_stallo*k_tas) + (m_tot*(-
(TAS_stallo*k_tas)^2/R_manovra -
g*cos(gamma(i))))/(0.5*rho*S1*(TAS_stallo*k_tas)^2);
b2_rota=-Cm0 -
(S2/S1)*V2_V^2*Cl_al2*(omega*d2)/(V2_V*TAS_stallo*k_tas)*(d2/corda1) -
Cl_al1_fuso*(omega*d1)/(TAS_stallo*k_tas)*(d1/corda1);

A_rotazione= A_trim;
sol_i_rotazione(i,[1 2])=inv(A_rotazione)*[b1_rota b2_rota]';
    % soluzioni per l'equilibrio non considerando il flap anteriore

A_rotazione_FLAP=[Cl_FLAP ,  (S2/S1)*Cl_al_equi ;
Cl_FLAP_fitt*(d1/corda1)  ,  (S2/S1)*Cl_EQUI_fitt*(d2/corda1) ];
sol_i_rotazione_FLAP(i,[1 2])=inv(A_rotazione_FLAP)*[b1_rota b2_rota]';
   % soluzioni per l'equilibrio considerando il flap
end

alpha_rota_inter=fit(gamma',sol_i_rotazione(:,1),'poly2');
coeff_alpha_rota=coeffvalues(alpha_rota_inter);
alpha_rota=coeff_alpha_rota(1).*gamma.^2 + coeff_alpha_rota(2).*gamma +
coeff_alpha_rota(3);

deltaE_rota_inter=fit(gamma',sol_i_rotazione(:,2),'poly2');
coeff_deltaE_rota=coeffvalues(deltaE_rota_inter);
deltaE_rota=coeff_deltaE_rota(1).*gamma.^2 + coeff_deltaE_rota(2).*gamma
+ coeff_deltaE_rota(3);

deltaE_F_rota_inter=fit(gamma',sol_i_rotazione_FLAP(:,2),'poly2');
coeff_deltaE_F_rota=coeffvalues(deltaE_F_rota_inter);
deltaE_F_rota=coeff_deltaE_F_rota(1).*gamma.^2 +
coeff_deltaE_F_rota(2).*gamma + coeff_deltaE_F_rota(3);

deltaF_rota_inter=fit(gamma',sol_i_rotazione_FLAP(:,1),'poly2');
coeff_deltaF_rota=coeffvalues(deltaF_rota_inter);
deltaF_rota=coeff_deltaF_rota(1).*gamma.^2 + coeff_deltaF_rota(2).*gamma
+ coeff_deltaF_rota(3);

gamma_XFLR5=[90  100  110  145.5  150  165  180]*(pi/180);
```

```
sol_deltaE_XFLR5=[14.895   12.435     9.855  2.865    2.205 0.75
     0.165]*(pi/180);
sol_deltaF_XFLR5=[-12.795  -10.41     -7.62  0.3 1.05  2.715
     3.39]*(pi/180);

deltaE_F_rotaXFLR5_inter=fit(gamma_XFLR5',sol_deltaE_XFLR5','poly3');
coeff_deltaE_F_rotaXFLR5=coeffvalues(deltaE_F_rotaXFLR5_inter);
deltaE_F_rotaXFLR5=coeff_deltaE_F_rotaXFLR5(1).*gamma.^3 +
coeff_deltaE_F_rotaXFLR5(2).*gamma.^2 +
coeff_deltaE_F_rotaXFLR5(3).*gamma + coeff_deltaE_F_rotaXFLR5(4);

deltaF_rotaXFLR5_inter=fit(gamma_XFLR5',sol_deltaF_XFLR5','poly3');
coeff_deltaF_rotaXFLR5=coeffvalues(deltaF_rotaXFLR5_inter);
deltaF_rotaXFLR5=coeff_deltaF_rotaXFLR5(1).*gamma.^3 +
coeff_deltaF_rotaXFLR5(2).*gamma.^2 + coeff_deltaF_rotaXFLR5(3).*gamma +
coeff_deltaF_rotaXFLR5(4);

   % interpolazioni soluzioni delle due modalitÃ : con e senza flap
   % interpolazione dati ricavati con XFLR5


figure (17)
plot(gamma,sol_i_rotazione(:,1),'red*',gamma,alpha_rota,'red')
xlabel('gamma','FontSize',10)
ylabel('incidenza rispetto alla linea di portanza nulla del velivolo
completo','FontSize',10)
legend('dati ricavati analiticamente','interpolazione')
title('modalitÃ  con solo equilibratore')

figure (18)
plot(gamma,sol_i_rotazione(:,2),'blue*',gamma,deltaE_rota,'blue')
xlabel('gamma','FontSize',10)
ylabel('deflessione equilibratore','FontSize',10)
legend('dati ricavati analiticamente','interpolazione')
title('modalitÃ  con solo equilibratore')

figure(19)
plot(gamma,sol_i_rotazione_FLAP(:,2),'g*',gamma,deltaE_F_rota,'g')
hold on
plot(gamma_XFLR5,sol_deltaE_XFLR5,'black*',gamma,deltaE_F_rotaXFLR5,'blac
k')
xlabel('gamma','FontSize',10)
ylabel('deflessione equilibratore','FontSize',10)
legend('dati ricavati analiticamente','interpolazione','dati ricavati con
XFLR5','interpolazione')
title('modalitÃ  con flap anteriore ed equilibratore')

figure(20)
plot(gamma,sol_i_rotazione_FLAP(:,1),'y*',gamma,deltaF_rota,'y')
hold on
plot(gamma_XFLR5,sol_deltaF_XFLR5,'black*',gamma,deltaF_rotaXFLR5,'black'
)
xlabel('gamma','FontSize',10)
ylabel('deflessione flap anteriore','FontSize',10)
legend('dati ricavati analiticamente','interpolazione','dati ricavati con
XFLR5','interpolazione')
title('modalitÃ  con flap anteriore ed equilibratore')
```

```
for i=1:length(gamma)
CL(i)=(m_tot*(-(TAS_stallo*k_tas)^2/R_manovra -
g*cos(gamma(i))))/(0.5*rho*S1*(TAS_stallo*k_tas)^2);
CD(i)=Cd0 + k1*CL(i) + CL(i)^2*k;
T_rotazione(i)=0.5*rho*S1*CD(i)*(TAS_stallo*k_tas)^2 +
m_tot*g*sin(gamma(i));
f_omega_rota=@(x) T_rotazione(i)/n_pale - rho*(x/(2*pi))^2*(R_pala*2)^4*
( ((TAS_stallo*k_tas*pi)/(x*R0))^2 * c_BART_CT(1) +
c_BART_CT(2)*((TAS_stallo*k_tas*pi)/(x*R0)) + c_BART_CT(3) )  ;
omega_elica_rota(i)=Newton_tg(omega_elica,f_omega_rota);
z=0;
    if abs(double(subs(f_omega_rota,omega_elica_rota(i)))) > 10^-4 ||
imag(double(subs(f_omega_rota,omega_elica_rota(i)))) ~= 0
        while abs(double(subs(f_omega_rota,omega_elica_rota(i)))) > 10^-4
&&  z <= 1500
            z=z+500;
            omega_elica_rota(i)=Newton_tg(1000 + z,f_omega_rota);
        end
    end
    if abs(double(subs(f_omega_rota,omega_elica_rota(i)))) > 10^-3 ||
imag(double(subs(f_omega_rota,omega_elica_rota(i)))) ~= 0
        disp('errore')
        disp(i)
    end

CP_rota(i)=double(subs(CP_elica,(TAS_stallo*k_tas*pi)/(omega_elica_rota(i
)*R0)));
P_resistente_rotazione(i)=rho*(omega_elica_rota(i)/(2*pi))^3*(R_pala*2)^5
*CP_rota(i) *(n_pale/n_motori);
C_resistente_rota(i)=P_resistente_rotazione(i)/omega_elica_rota(i);

i_I_rota(i)=find(min(abs(C_motore_ - C_resistente_rota(i))) ==
abs(C_motore_ - C_resistente_rota(i)));
I_rota(i)=I(i_I_rota(i));
   % dai dati fornito dal costruttore Ã¨ stato ricavato l'andamento della
   % coppia la variare della corrente in input. Da esso Ã¨ possibile
   % ricavare la corrente minima in uscita dalla batteria per contrastare
   % la coppia resistente dell'elica


P_tot_elica_rota(i)=P_resistente_rotazione(i)*(1/(effi_ESC_cavi*0.8));
   % 0.8 rappresenta il rendimento del motore. E' stato scelto un valore
   % medio perchÃ¨ in questa fase varierÃ  in maniera significativa sia
la
   % velocitÃ  angolare che la corrente in ingresso al motore, quindi
   % lavorerÃ  in regime non stazionario. La scelta di questo valore
   % deriva dal valore dell'effcienza in salita. Durante la rotazione
   % l'efficienza aumenterÃ  perchÃ¨ diminuiranno sicuramente le perdite
per
   % effetto Joule, per attrito e di ferro. Quindi Ã¨ stato scelto un
valore
   % maggiore di quello in salita, ma minore di quello massimo
C_motore_rota(i)=C_motore(P_tot_elica_rota(i)/tensione0);
```

```
end


t_gamma=(gamma - pi/2)/omega;
figure (21)
plot(gamma,omega_elica_rota,'r',gamma,P_tot_elica_rota,'b')
xlabel('gamma','FontSize',10)
legend('velocitÃ  angolare elica','potenza necessaria')
figure (22)
plot(t_gamma,P_tot_elica_rota,'b',t_gamma,omega_elica_rota,'r')
xlabel('tempo rotazione','FontSize',10)



t_rota=pi/2 * 1/omega;
Prota_inter=fit(t_gamma',P_tot_elica_rota','poly3');
coef_Prota=coeffvalues(Prota_inter);
Prota_cubi=coef_Prota(1).*t_gamma.^3 + coef_Prota(2).*t_gamma.^2 +
coef_Prota(3).*t_gamma + coef_Prota(4);
  % interpolazione potenza
hold on
plot(t_gamma,Prota_cubi,'b*')
legend('potenza necessaria reale','velocitÃ  angolare
elica','interpolazione potenza reale')

f_cubi_Prota=@(t) coef_Prota(1).*t.^3 + coef_Prota(2).*t.^2 +
coef_Prota(3).*t + coef_Prota(4);
Etot_rota=integral(f_cubi_Prota,t_gamma(1),t_gamma(end));
  % energia totale spesa durante la rotazione sotto l'ipotesi di un
  % efficienza pari a 0.8
effi_salita=(m_tot*g*R_manovra)/Etot_rota;


figure(23)
plot(t_gamma,C_resistente_rota,'r',t_gamma,C_motore_rota,'bo')
xlabel('tempo','FontSize',10)
legend('coppia resistente elica','coppia generata dal motore')
```

NOTA 9 Script Matlab per calcolare le variabili d'interesse durante la rotazione

ALLEGATO 7.2

```
  % partendo da una condizione arbitraria, si può calcolare la variazione
di deflessione rispetto la
  % suddetta condizione del FLAP e dell'equilibratore. E' stato
utilizzato
  % insieme al programma XFLR5. Tutti i coefficienti a seguire sono stati
  % calcolati dal programma in questione. Il metodo converge sempre, ma
il
  % numero di iterazioni non Ã¨ costante. Il criterio di arresto Ã¨ un CL
  % corretto fino alla seconda cifra decimale e un coefficiente di
momento
  % dell'ordine di 10^-5.
```

```
dCL=-0.002;
dCm=0.00049;
A_rota_XFLR5=[Cl_FLAP      ,  Cl_al_equi*(S2/S1) ;
Cl_FLAP_fitt*(d1/corda1)    , Cl_EQUI_fitt*(S2/S1)*(d2/corda1)];

b_rota_XFLR5=[dCL , dCm];
delta_FLAP_EQUI=inv(A_rota_XFLR5)*b_rota_XFLR5'
```

NOTA 10 Script Matlab per calcolare le variazioni di deflessione delle superfici di controllo tali da generare la variazione di coefficiente di momento di beccheggio e di portanza volute

ALLEGATO 7.3

```
Iyy=0.146;
  % momento d'inerzia del drone completo rispetto ad Y (calcolato co
XFLR5)

Cm_err_XFLR5=[0.00009 -0.00002   0.00003     0.00004   0.00005     0    -
0.00007];
delta_Alpha(1)=0;
delta_omega(1)=0;
for i=2:(length(Cm_err_XFLR5))

    M(i-1)=0.5*rho*S1*corda1*(TAS_stallo*k_tas)^2*Cm_err_XFLR5(i-1);
    omega_punto_beccheggio(i-1)=M(i-1)/(Iyy);
    delta_t(i-1)=(gamma_XFLR5(i)-gamma_XFLR5(i-1))/omega;
    delta_Alpha(i)=0.5*delta_t(i-1)^2*omega_punto_beccheggio(i-1) +
delta_omega(i-1)*delta_t(i-1) + delta_Alpha(i-1);
    delta_omega(i)=omega_punto_beccheggio(i-1)*delta_t(i-1) +
delta_omega(i-1);

end
fprintf('variazione di incidenza a fine rotazione pari a %f e variaizone
di velocitÃ  angolare pari a
%f',delta_Alpha(end)*180/pi,delta_omega(end)*180/pi)
```

NOTA 11 Script Matlab per calcolare l'errore in termini di velocità angolare e assetto dovuti al non perfetto equilibrio, in termini di momento di beccheggio, durante la rotazione

ALLEGATO 8

```
% CROCIERA
effi_motore_max=0.87*(24/22.2);% dato fornito dal costruttore
  % il fattore moltiplicativo correttivo deriva dal fatto che il tes Ã¨
  % stato fatto a 24 Volts invece il drone opererÃ  a 22.2 Volts

% MODALITA' CROCIERA
 % confronto di due condizioni di volo livellato (13 m/s ; 20 m/s)
 % con e senza flap. In particolare, si vuole analizzare la resistenza
 % aerodinamica
```

```
B=readtable('CROCIERA_V_20_SI_FLAP.txt');
CRUISE_SiFLAP_20=table2array(B);
B=readtable('CROCIERA_V_20_NO_FLAP.txt');
CRUISE_NoFLAP_20=table2array(B);
B=readtable('CROCIERA_V_13_SI_FLAP.txt');
CRUISE_SiFLAP_13=table2array(B);
B=readtable('CROCIERA_V_13_NO_FLAP.txt');
CRUISE_NoFLAP_13=table2array(B);


CRUISE_SiFLAP_20_resi=fit(CRUISE_SiFLAP_20(:,3),CRUISE_SiFLAP_20(:,6),'po
ly2');
CRUISE_SiFLAP_20_c_resi=coeffvalues(CRUISE_SiFLAP_20_resi);
CD_CRUISE_SiFLAP_20=CRUISE_SiFLAP_20_c_resi(1).*CL_inter.^2 +
CRUISE_SiFLAP_20_c_resi(2).*CL_inter + CRUISE_SiFLAP_20_c_resi(3);
f_err_CD_CRUISE_SiFLAP_20=@(x) CRUISE_SiFLAP_20_c_resi(1).*x.^2 +
CRUISE_SiFLAP_20_c_resi(2).*x + CRUISE_SiFLAP_20_c_resi(3)
err_CRUISE_SiFLAP_20=max( abs(
f_err_CD_CRUISE_SiFLAP_20(CRUISE_SiFLAP_20(:,3)) - CRUISE_SiFLAP_20(:,6)
) )

CRUISE_NoFLAP_20_resi=fit(CRUISE_NoFLAP_20(:,3),CRUISE_NoFLAP_20(:,6),'po
ly2');
CRUISE_NoFLAP_20_c_resi=coeffvalues(CRUISE_NoFLAP_20_resi);
CD_CRUISE_NoFLAP_20=CRUISE_NoFLAP_20_c_resi(1).*CL_inter.^2 +
CRUISE_NoFLAP_20_c_resi(2).*CL_inter + CRUISE_NoFLAP_20_c_resi(3);
f_err_CD_CRUISE_NoFLAP_20=@(x) CRUISE_NoFLAP_20_c_resi(1).*x.^2 +
CRUISE_NoFLAP_20_c_resi(2).*x + CRUISE_NoFLAP_20_c_resi(3)
err_CRUISE_NoFLAP_20=max( abs(
f_err_CD_CRUISE_NoFLAP_20(CRUISE_NoFLAP_20(:,3)) - CRUISE_NoFLAP_20(:,6)
) )

CRUISE_SiFLAP_13_resi=fit(CRUISE_SiFLAP_13(:,3),CRUISE_SiFLAP_13(:,6),'po
ly2');
CRUISE_SiFLAP_13_c_resi=coeffvalues(CRUISE_SiFLAP_13_resi);
CD_CRUISE_SiFLAP_13=CRUISE_SiFLAP_13_c_resi(1).*CL_inter.^2 +
CRUISE_SiFLAP_13_c_resi(2).*CL_inter + CRUISE_SiFLAP_13_c_resi(3);
f_err_CD_CRUISE_SiFLAP_13=@(x) CRUISE_SiFLAP_13_c_resi(1).*x.^2 +
CRUISE_SiFLAP_13_c_resi(2).*x + CRUISE_SiFLAP_13_c_resi(3)
err_CRUISE_SiFLAP_13=max( abs(
f_err_CD_CRUISE_SiFLAP_13(CRUISE_SiFLAP_13(:,3)) - CRUISE_SiFLAP_13(:,6)
) )

CRUISE_NoFLAP_13_resi=fit(CRUISE_NoFLAP_13(:,3),CRUISE_NoFLAP_13(:,6),'po
ly2');
CRUISE_NoFLAP_13_c_resi=coeffvalues(CRUISE_NoFLAP_13_resi);
CD_CRUISE_NoFLAP_13=CRUISE_NoFLAP_13_c_resi(1).*CL_inter.^2 +
CRUISE_NoFLAP_13_c_resi(2).*CL_inter + CRUISE_NoFLAP_13_c_resi(3);
f_err_CD_CRUISE_NoFLAP_13=@(x) CRUISE_NoFLAP_13_c_resi(1).*x.^2 +
CRUISE_NoFLAP_13_c_resi(2).*x + CRUISE_NoFLAP_13_c_resi(3)
err_CRUISE_NoFLAP_13=max( abs(
f_err_CD_CRUISE_NoFLAP_13(CRUISE_NoFLAP_13(:,3)) - CRUISE_NoFLAP_13(:,6)
) )

figure (29)
plot(CL_inter,CD_CRUISE_SiFLAP_20,'y',CRUISE_SiFLAP_20(:,3),CRUISE_SiFLAP
_20(:,6),'y*')
```

```
hold on
plot(CL_inter,CD_CRUISE_NoFLAP_20,'b',CRUISE_NoFLAP_20(:,3),CRUISE_NoFLAP
_20(:,6),'b*')
xlabel('coefficiente di portanza totale')
ylabel('coefficiente di resistenza')
legend('flap ed equilibratore','interpolazione','solo
equilibratore','interpolazione')
title('CONDIZIONE DI CROCIERA PER UNA VELOCITA PARI A 20m/s')
figure (30)
plot(CL_inter,CD_CRUISE_SiFLAP_13,'y',CRUISE_SiFLAP_13(:,3),CRUISE_SiFLAP
_13(:,6),'y*')
hold on
plot(CL_inter,CD_CRUISE_NoFLAP_13,'b',CRUISE_NoFLAP_13(:,3),CRUISE_NoFLAP
_13(:,6),'b*')
xlabel('coefficiente di portanza totale')
ylabel('coefficiente di resistenza')
legend('flap ed equilibratore','interpolazione','solo
equilibratore','interpolazione')
title('CONDIZIONE DI CROCIERA PER UNA VELOCITA PARI A 13m/s')


A_cruise=A_trim;
% INTERVALLO OPERATIVO DI VELOCITA':  TAS_stallo(10.96) : [37.5 - 17];
   % velocitÃ  minima imposta dallo stallo; velocitÃ  massima imposta dal
   % vincolo di rimanere ben al di fuori dal regime turbolento pari circa
a 1000000.
   % 37.5 Ã¨ la velocitÃ  per cui si ha un numero di Reynolds pari a
1000000
V_iter=5:0.01:(37.5 - 17);
for i=1:length(V_iter)

CL_cruise(i)=(m_tot*g)/(0.5*rho*S1*(V_iter(i))^2);
b2_cruise=-Cm0;
sol_cruise([1 2],i)=inv(A_cruise) * [CL_cruise(i) b2_cruise]';

incidenza_P_Coda(i)=sol_cruise(1,i)*(1 - de_dal) - e0 -(-
sol_cruise(2,i));
Cm_sbalzoCoda(i)=CD_piano_C_equi_0_f(incidenza_P_Coda(i))*cos(sol_cruise(
1,i))*(0.042/0.52)*(0.1/0.4) -
CD_piano_C_equi_0_f(incidenza_P_Coda(i))*sin(sol_cruise(1,i))*(0.042/0.52
)*(0.89/0.4);
if ( incidenza_P_Coda(i) ) > 9.5*(pi/180)  % limite imposto dallo stallo
del piano di coda
    % la formula sovrastante rappresenta l'incidenza vista dal piano di
    % coda, tenendo in considerazione la scia dell'ala principale e il
    % calettamento (pari alla deflessione dell'equilibratore cambiata di
segno)
    % In realtÃ , la variazione del calettamento, pari alla variazione
nella
    % direzione di portanza nulla del piano di coda, Ã¨ diversa dalla
deflessione dell'equilibratore, ma dopo verifica
    % con XFLR5 risulta circa uguale. In particolare, Ã¨ stato calcolata
una
    % variazione nella direzione di portanza nulla pari a 4.6 gradi per
una
    % deflessione dell'equilibratore di 5 gradi. Quindi il criterio di
    % verifica sovrastante Ã¨ conservativo ed allora valido.
```

```
    fprintf('il piano di coda stalla per una velocitÃ  pari a
%f',V_iter(i))
end


CD_cruise(i)=CL_cruise(i)^2*k + k1*CL_cruise(i) + Cd0;
T_cruise(i)=0.5*rho*S1*CD_cruise(i)*(V_iter(i))^2;
f_omega_cruise=@(x) -T_cruise(i)/n_pale +
rho*((x/(2*pi))^2)*(R_pala*2)^4* ( ((V_iter(i)*pi)/(x*R0))^2 *
c_BART_CT(1) + c_BART_CT(2)*((V_iter(i)*pi)/(x*R0)) + c_BART_CT(3) )  ;


omega_elica_cruise(i)=Newton_tg(omega_elica_rota(end),f_omega_cruise);


if double(subs(f_omega_cruise,omega_elica_cruise(i))) > 10^-4

fprintf('errore, metodo di Newton non ha trovato una soluzione accetabile
al passo %.f',i)

end

CP_cruise(i)=double(subs(CP_elica,(V_iter(i))*pi/(omega_elica_cruise(i)*R
0)));
CT_cruise(i)=double(subs(CT_elica,(V_iter(i)*pi)/(omega_elica_cruise(i)*R
0)));
P_resistente_cruise(i)=rho*(omega_elica_cruise(i)/(2*pi))^3*(R_pala*2)^5*
CP_cruise(i) *(n_pale/n_motori);
C_resistente_cruise(i)=P_resistente_cruise(i)/(omega_elica_cruise(i));

i_I_cruise(i)=find(min(abs(C_motore_ - C_resistente_cruise(i))) ==
abs(C_motore_ - C_resistente_cruise(i)));
I_cruise(i)=I(i_I_cruise(i));
   % dai dati fornito dal costruttore Ã¨ stato ricavato l'andamento della
   % coppia la variare della corrente in input. Da esso Ã¨ possibile
   % ricavare la corrente minima in uscita dalla batteria per contrastare
   % la coppia resistente dell'elica

effi_elica_cruise(i)=(CT_cruise(i))/(CP_cruise(i)) *
(V_iter(i)*pi)/(omega_elica_cruise(i)*R0);
effi_tot_cruise(i)=effi_motore_max* effi_ESC_cavi *effi_elica_cruise(i);


P_tot_elica_cruise(i)=(T_cruise(i)*V_iter(i))/effi_tot_cruise(i);

C_motore_cruise(i)=C_motore(P_tot_elica_cruise(i)/tensione0);
end


% condizione di Potenza necessaria minima
CL_Pn_min=0.5*(k1/k) + sqrt( (k1^2/4) + 3*k*Cd0 )/( k );
V_Pn_min=sqrt( (2*m_tot*g)/(rho*S1*CL_Pn_min) );
 min1=find( min( abs(CL_cruise - CL_Pn_min) ) == abs(CL_cruise -
CL_Pn_min) );
effi_Pn_min=effi_tot_cruise(min1);
sol_cruise_Pn_min=inv(A_cruise) * [CL_Pn_min  b2_cruise]';
Pn_min_cruise=P_tot_elica_cruise(min1)
```

```
Pn_min_cruise_2=min(P_tot_elica_cruise)
 min2=find(Pn_min_cruise_2==P_tot_elica_cruise);
effi_Pn_min_2=effi_tot_cruise(min2)
alpha_Pn_min_2=sol_cruise(1,min2)
deltaE_Pn_min_2=sol_cruise(2,min2)

figure(24)
plot(V_iter,sol_cruise(1,:),'g',V_iter,sol_cruise(2,:),'r',TAS_stallo*one
s(1,101),-
1:0.02:1,'black',V_iter(min2),alpha_Pn_min_2,'b*',V_iter(min2),deltaE_Pn_
min_2,'b*',V_iter(min1),sol_cruise_Pn_min(1,1),'r*',V_iter(min1),sol_crui
se_Pn_min(2,1),'r*')
xlabel('VelocitÃ  di crociera di trim')
legend('incidenza velivolo completo di trim','deflessione equilibratore
di trim','limite imposto dallo stallo Ala Principale','condizione potenza
minima reale','','condizione potenza minima approssimata')
figure(25)
plot(V_iter,P_tot_elica_cruise,'b',TAS_stallo*ones(1,101),0:1:100,'black'
,V_iter(min2),Pn_min_cruise_2,'b*',V_iter(min1),Pn_min_cruise,'r*')
xlabel('VelocitÃ  di crociera di trim')
legend('Potenza necessaria in condizioni di trim','limite imposto dallo
stallo Ala Principale','condizione potenza minima reale','condizione
potenza minima approssimata')
figure(26)
plot(V_iter,effi_tot_cruise,'b',TAS_stallo*ones(1,101),0:0.01:1,'black',V
_iter(min2),effi_Pn_min_2,'b*',V_iter(min1),effi_Pn_min,'r*')
xlabel('VelocitÃ  di crociera di trim')
legend('efficienza totale in condizione di trim','limite imposto dallo
stallo Ala Principale','condizione potenza minima reale','condizione
potenza minima approssimata')
figure(27)
plot(V_iter,C_resistente_cruise,'m',V_iter,C_motore_cruise,'y',TAS_stallo
*ones(1,101),0:0.01:1,'black')
xlabel('VelocitÃ  di crociera di trim')
legend('Coppia resistente elica','Coppia motore','limite imposto dallo
stallo Ala Principale')
figure(28)
plot(V_iter,omega_elica_cruise,'b',TAS_stallo*ones(1,1001),0:1:1000,'blac
k')
xlabel('VelocitÃ  di crociera di trim')
legend('velocitÃ  angolare','limite imposto dallo stallo Ala Principale')
```

NOTA 12 Script Matlab utilizzato per l'analisi della crociera

ALLEGATO 9

```
% modello sperimentale di scarica della batteria a potenza costante

C0=3;
C_start_cruise=C0 - (Etot_rota + Etot_salita)/(3600*tensione0);
epsilon=-1.009;
delta=24.96;
```

```
beta_battery=0.9664;
C=C_start_cruise*0.8;    % capacitÃ  della batteria che si intende usare
(0.8)
delta_t=delta* P_tot_elica_cruise.^epsilon .*C^beta_battery; % autonomia
oraria (ore)
delta_s=delta_t*3600.*V_iter./1000; % autonomia chilometrica (km)

max_dt=find( delta_t==max(delta_t) );
max_ds=find( delta_s==max(delta_s) );


figure(31)
plot(V_iter,delta_t,'b',TAS_stallo*ones(1,16),0:0.1:1.5,'black',V_iter(ma
x_dt),delta_t(max_dt),'ro')
xlabel('velocitÃ ')
ylabel('autonomia oraria [h]')
legend('','limite imposto dallo stallo')
figure(32)
plot(V_iter,delta_s,'b',TAS_stallo*ones(1,501),0:0.1:50,'black',V_iter(ma
x_ds),delta_s(max_ds),'ro')
xlabel('velocitÃ ')
ylabel('autonomia chilometrica [km]')
legend('','limite imposto dallo stallo')
```

NOTA 13 Script Matlab per il calcolo dell'autonomia chilometrica ed oraria