Introduzione al problema e inizializzazione dei parametri

Nel seguente progetto di Controllo dei Sistemi Incerti ci proponiamo lo studio tramite alcuni strumenti di sintesi di un controllore, del sistema costituito da un convertiplano. In particolare, utilizzeremo il controllore LQG, Mix-Sensitivity, H-Infinity-norm e come ultimo un controllore μ-Synthesis con la DK-Iteration. Andremo poi a studiare tali controllori sfruttando lo strumento della μ-Analysis. Per simulare e testare i controllori precedentemente esposti ci siamo avvalsi dell'ambiente Matlab Simulink®.

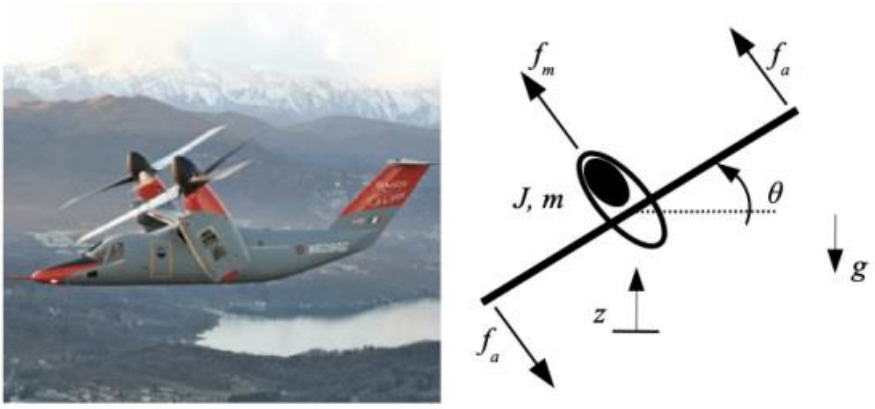

Figura 1.1 – Sistema Convertiplano

1.1 Parametri Fisici

Per impostare i parametri fisici nominali del sistema abbiamo tenuto conto della ragionevolezza dei valori per il sistema fisico reale in studio. Di seguito riportiamo l'elenco di questi con le relative unità di misura nel Sistema Internazionale:

Convertiplano

- Massa della struttura: ..... $M = 3000\ Kg$
- Momento di inerzia associato: ..... $J = 5000\ \mathrm{Kg}m^2$
- Coefficiente di attrito viscoso di **traslazione** con l'aria: ..... $b = 150\ Ns/m$
- Coefficiente di attrito viscoso di **rotazione** con l'aria: ..... $\beta = 15\ Nms/Rad$
- Apertura alare del convertiplano: ..... $l = 10\ m$
- Accelerazione di gravità: ..... $g = 9.81\ \mathrm{m}/s^2$

Attuatori (modellati come funzioni di trasferimento)

- Guadagno statico motori: ..... $K_{m1} = 500\ N$
  ..... $K_{m2} = 100\ N$
- Costante di tempo poli: ..... $T_{m1} = 5\ s$
  ..... $T_{m2} = 5\ s$
- Costante di ritardo: ..... $T_1 = 0.5\ s$
  ..... $T_2 = 0.5\ s$

1.2 Modello dinamico

Le equazioni differenziali sotto riportate rappresentano il modello dinamico del sistema fisico, ovvero descrivono l'evoluzione degli stati del convertiplano così definiti:

- Altitudine del convertiplano: $\boldsymbol{z}$
- Posizione angolare rispetto alla direzione longitudinale: $\boldsymbol{\vartheta}$

Con la seguente dinamica:

$$M\ddot{z} + b\dot{z} = f_m \cdot \cos\theta - Mg$$

$$J\ddot{\theta} + \beta\dot{\theta} = 2lf_a$$

In queste equazioni notiamo la presenza dei parametri $\boldsymbol{f_m}$ e $\boldsymbol{f_a}$ che rappresentano le forze che vengono impresse dagli attuatori del convertiplano, nonchè il nostro ingresso di controllo.

Invece le funzioni di trasferimento che modellano i motori sono date da:

$$G_{m1} = \frac{K_{m1}}{(T_{m1}s + 1)} \cdot e^{-T_1 s} \qquad G_{m2} = \frac{K_{m2}}{(T_{m2}s + 1)} \cdot e^{-T_2 s}$$

1.3 Modellazione e analisi sistema

Linearizzando il sistema intorno al punto di equilibrio $z = \theta = 0$ abbiamo verificato che fosse **completamente raggiungibile**:

$$A = \begin{bmatrix} -b/M & 0 & 0 & 0 \\ 0 & -\beta/J & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 2l \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Da cui:

$$\dot{x} = Ax + Bf$$
$$y = Cx$$

Dove lo stato x è dato da:

$$x = \begin{bmatrix} \dot{z} \\ \dot{\theta} \\ z \\ \theta \end{bmatrix} \qquad y = \begin{bmatrix} z \\ \theta \end{bmatrix} \qquad f = \begin{bmatrix} f_m \\ f_a \end{bmatrix}$$

Quindi abbiamo scelto l'uscita del sistema come gli stati z e θ, così da rendere il sistema linearizzato anche **completamente osservabile**:

$$R = \begin{bmatrix} 1 & 0 & {}^{-b}/_{M} & 0 \\ 0 & 2l & 0 & {}^{-2\beta l}/_{J} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2l \end{bmatrix} \qquad O = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Perciò possiamo effettuare un **controllo a zero** del nostro sistema, cioè fare in modo che il convertiplano rimanga ad una altitudine costante ($z = 0$) e con inclinazione costante ($\theta = 0$).

Progettazione e modellazione su MATLAB Simulink®

La dinamica del sistema è stata modellata su MATLAB Simulink® a partire dalle equazioni proposte, tramite schemi a blocchi e Matlab Function. I dati temporali relativi alla simulazione sono i seguenti:

- $t_{simulazione} \in [40, 400]\ s$

Per la modellazione su Simulink® abbiamo diviso il sistema fisico in sotto-sistemi (subsystems), ciascuno rappresentante parti specifiche della simulazione, come possiamo vedere di seguito:

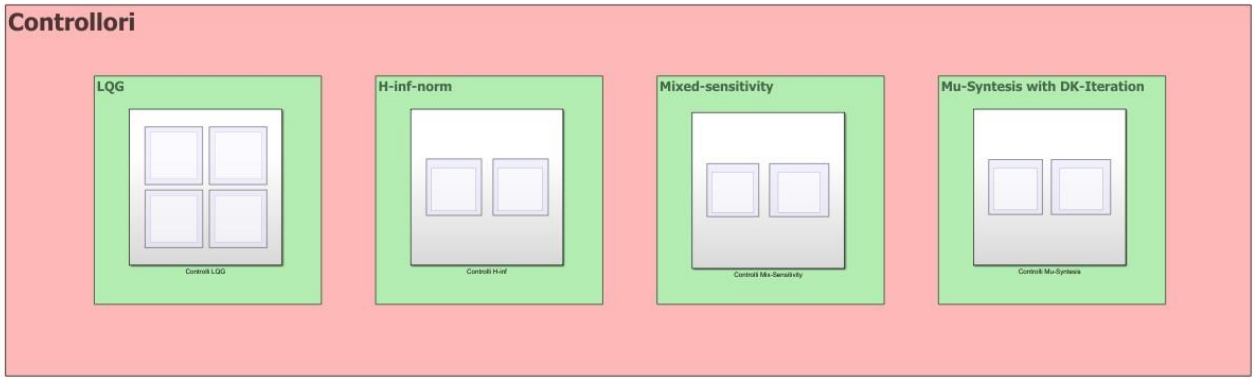

Figura 2.1 - Schema a blocchi del modello Simulink®

Di seguito approfondiremo ciascun blocco, osservando la costruzione e le relative osservazioni/conclusioni di ciascuno.

2.1 Controllore LQG

Il primo passo effettuato per andare a ricavare il controllore LQG, è stato **linearizzare** il sistema intorno al **punto di equilibrio** su cui vogliamo andare a **controllarlo** (come già visto precedentemente nell'analisi di Raggiungibilità e Osservabilità).

Fatto ciò, abbiamo unito gli attuatori al sistema dinamico del convertiplano così da avere un sistema unico:

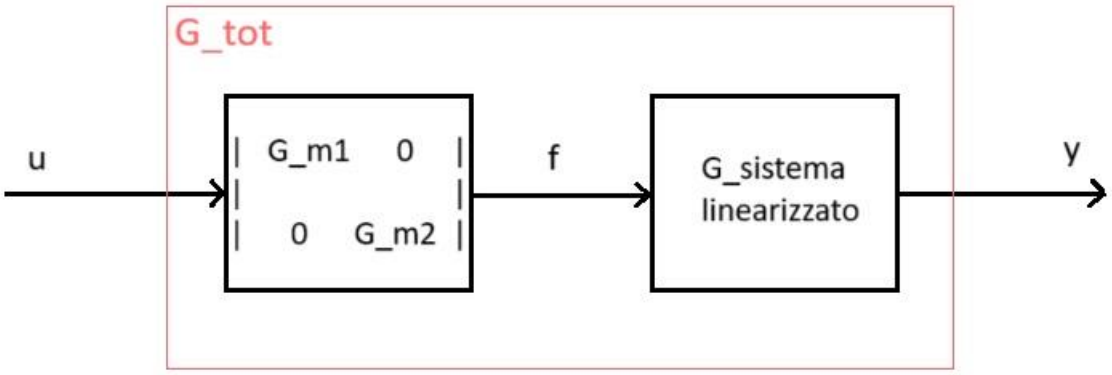

Figura 2.2 – Composizione sistema dinamico con sistema degli attuatori (composizione serie)

Quindi abbiamo cercato il sistema G_{tot} sfruttando la proprietà di composizione di sistemi in serie nella loro forma di stato, così da capire quali stati dover pesare per poi applicare il comando *lqg* su MatLab.

Dunque, il codice usato per ricavare G_{tot} in forma di stato è il seguente:

```
% Creazione sistemi attuatori in forma di stato
G1 = tf([Km1], [Tm1 1]) * exp(-s*T1);
G2 = tf([Km2], [Tm2 1]) * exp(-s*T2);
G_att = blkdiag(G1, G2);

G_att = ss(G_att);

A_att = G_att.A;
B_att = G_att.B;
C_att = G_att.C;
D_att = G_att.D;

% Composizione in serie del sistema dinamico e degli attuatori (forma di
% stato)
A_tot = [      A_att,         zeros(2,4);
           B_nom*C_att,          A_nom    ];

B_tot = [B_att;  B_nom*D_att];

C_tot = [D_nom*C_att,   C_nom];

D_tot = [0, 0;
         0, 0];

G = ss(A_tot, B_tot, C_tot, D_tot);
```

Figura 2.3 – Codice per la composizione in serie di sistemi

Fatto ciò, abbiamo definito la **matrice di peso** (LQR) e la **matrice di covarianza** dei rumori bianchi (KF) così da usarle per il codice *lqg()* che ci restituisce i guadagni per l'LQR e per il KF:

```
% Definisco le matrici di covarianza dei rumori in ingresso e dei sensori
dev_std_fm = 200;
dev_std_fa = 40;
dev_std_z = 10;
dev_std_theta = 0.3;

% Matrice covarianza rumori in inngresso (gli errori di processo sono
% riferiti solo alle equazioni di z: e theta:)
Q = blkdiag(0, 0, dev_std_fm ^ 2, dev_std_fa ^ 2, 0, 0);

% Matrice covarianza errori di misura (y1 = z; y2 = theta)
R = blkdiag(dev_std_z ^ 2, dev_std_theta ^ 2);

% Big matricione di covarianza
QWV = blkdiag(Q, R);

% Matrice di peso per stati e ingressi, dove pesiamo solo gli ingressi e
% gli stati z e theta
Q_pesi = blkdiag(zeros(4), eye(4));

% lqg(sistema in forma di stato,
%     pesi per stato e ingressi,
%     matrici di covarianza)
[reg, info] = lqg(G, Q_pesi, QWV);

% Guadagno LQR %
Kc = - info.Kx;

% Guadagno filtro di Kalman %
Kf = info.L;
```

Figura 2.4 – Codice per calcolo guadagni costanti con lqg()

Infine, per fare un **controllore LQG** con **azione integrale** per disturbi costanti a regime, abbiamo semplicemente usato il comando *lqi()* ridefinendo le matrici di peso per trovare il **nuovo guadagno** per la parte **LQR**:

```
% Matrici di peso per stati (stati + stati da integrare) e ingressi
% Vado a pesare solo z, theta (2 volte sia per stato stimato che stato integrato)e i 2 ingressi
Q_z = blkdiag(zeros(4), blkdiag(1,1,1,0.0001));
R_u = eye(2);

% Calcolo del guadagno del controllore con integratori
Kc_int = -lqi(G, Q_z, R_u);
```

Figura 2.5 – Guadagno per controllo LQG integrale (con azione integrale su z e θ)

3.1 Controllori Mixed-Sensitivity e H-inf

Il primo passo effettuato per andare a ricavare il controllore **Mixed-Sensitivity e H-inf** è stato **linearizzare** il sistema intorno al **punto di equilibrio** su cui vogliamo andare a **controllarlo** (come già visto precedentemente nell'analisi di Raggiungibilità e Osservabilità).

Quello di seguito è lo schema a blocchi usato per trovare il controllore Mixed-Sensitivity e quello H-inf:

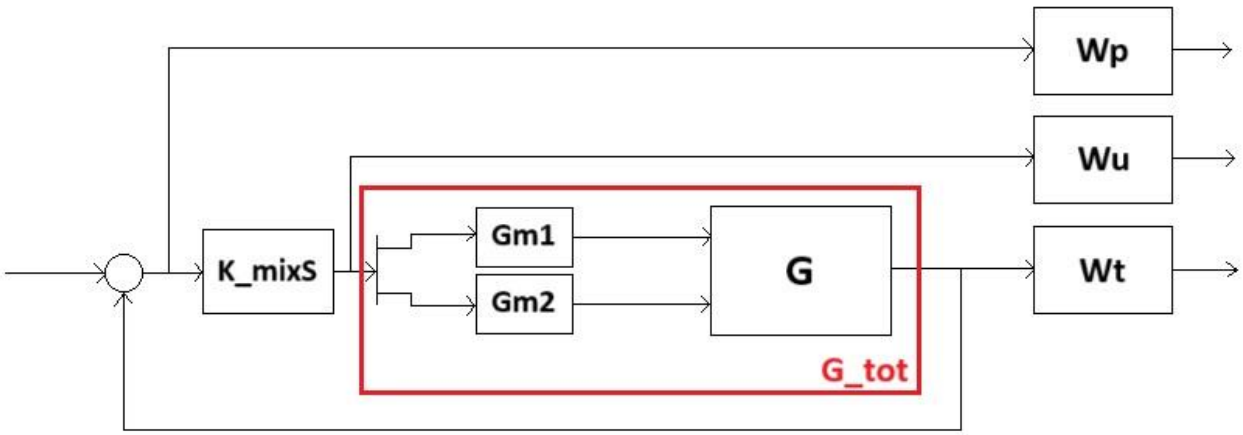

Figura 3.1 – Schema a blocchi per Mixed-Sensitivity

Per trovare il sistema totale (la matrice P nella forma di Doyle senza i pesi), che vediamo in Figura 3.1 come il sistema G_tot, abbiamo eseguito il seguente codice:

```
% Definisco la matrice di trasferimento nominale del convertiplano %
G_n = ss(A_nom,B_nom,C_nom,D_nom);
G_n = zpk(G_n);

% Definisco le fdt degli attuatori nominali (trascuro tempo di ritardo)
Gm1_n = Km1/(Tm1*s+1);
Gm2_n = Km2/(Tm2*s+1);

G_attuators_n = [Gm1_n, 0; 0, Gm2_n];

% Calcolo Matrice di trasferimento del sistema + attuatori (G_tot nominale)
G_tot_n = G_n*G_attuators_n;
```

Figura 3.2 – Codice per calcolo G_tot nominale

Il prossimo passo per la sintesi di un controllo è quello di definire i pesi di **Prestazione**(WP), **Robustezza**(WT) e **sforzo di controllo**(WU): abbiamo scelto come peso WU una costante,

invece per la scelta di WP ci siamo basati sulla Sensitività del controllo LQG; di conseguenza la scelta di WT è automatica, cioè complementare a WP.

Quindi abbiamo scelto WP utilizzando il controllore LQG già sintetizzato:

```
%% Calcolo pesi per gli altri controllori sulla base del K_LQG

% K_LQG = [A_tot + B_tot*Kc + Kf*C_tot,    Kf;
%                     Kc,                zeros(2)];

K_LQG = ss(A_tot + B_tot*Kc + Kf*C_tot, Kf, Kc, zeros(2));
K_LQG = zpk(K_LQG);

G = zpk(G);

% Parametri funzioni peso Wp %
A1=1e-4;
M1=2;
wB1=0.1;
A2=1e-4;
M2=2;
wB2=0.1;

wP1=makeweight(1/A1,wB1,1/M1);
wP2=makeweight(1/A2,wB2,1/M2);

% Confronto S e funzioni peso Wp %
S = 1/(eye(2) + G*K_LQG);

S_Wp = bodeplot(S(1,1), 1/wP1);
setoptions(S_Wp, 'PhaseVisible','off');
legend('S(1,1)', '1/wp1');

S_Wp = bodeplot(S(2,2), 1/wP2);
setoptions(S_Wp, 'PhaseVisible','off');
legend('S(2,2)', '1/wp2');
```

Figura 3.3 – Codice per ricavare il peso WP

Che con i parametri per i pesi WP scelti in *Figura 3.3*, troviamo:

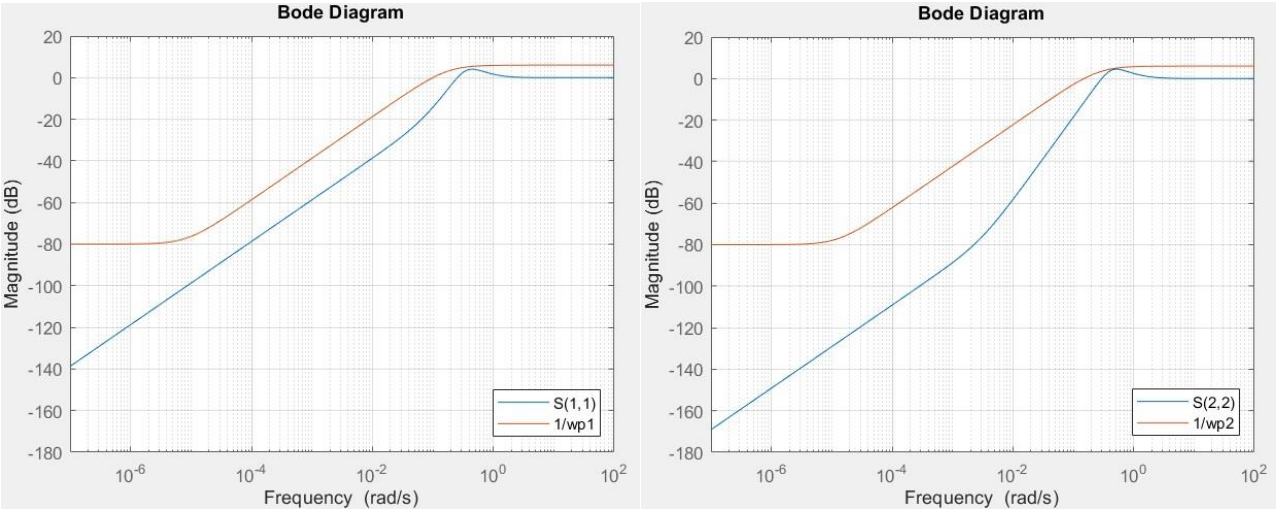

Figura 3.4 – Scelta del peso WP

A questo punto, definiti tutti i pesi, possiamo sintetizzare il controllore con il comando ***mixsyn()***:

```
%% Calcolo controllore Mix sensitivity

% Modello i pesi WU, WP, WT %

wU=tf(1);
WU=blkdiag(wU,wU);

A1=1e-4;
M1=2;
wB1=0.1;
A2=1e-4;
M2=2;
wB2=0.1;

wP1=makeweight(1/A1,wB1,1/M1);
wP2=makeweight(1/A2,wB2,1/M2);
WP=blkdiag(wP1,wP2);

wT1 = makeweight(1/M1,wB1,1/A1);
wT2 = makeweight(1/M2,wB2,1/A2);
WT = [wT1 0;0 wT2];

% Calcolo Controllore Mix-Syn %
[K_ms,CL_ms,GAM_ms,~] = mixsyn(G_tot_n,WP,WU,WT);

% Riduco ordine del controllore
K_ms = minreal(zpk(tf(K_ms)),1e-1);
% Tolgo termini non diagonali (guadagni dell'ordine 10e-10!)
K_ms = [K_ms(1,1) 0; 0 K_ms(2,2)];
```

Figura 3.5 – Calcolo Pesi e calcolo controllore approssimato

Invece per sintetizzare un controllore H-inf bisogna usare il comando ***hinfsyn()***, per cui servono i pesi usati nel Mixed-Sensitivity (WP, WT e WU) e mettere il sistema nella forma P-K usando il comando ***connect()***. Infatti, abbiamo definito ingressi e uscite per ciascun blocco nel seguente modo:

```
%% Calcolo controllore H-inf

% Definisco ingressi e uscite blocchi sistema per connect %
G_tot_n.u = {'uG1','uG2'};
G_tot_n.y = {'yG1','yG2'};

WP.u = {'e1','e2'};
WP.y = {'zp1','zp2'};

WT.u = {'yG1','yG2'};
WT.y = {'zt1','zt2'};

WU.u = {'uG1','uG2'};
WU.y = {'zu1','zu2'};

Sum1 = sumblk ('e1 = yG1 + w1');
Sum2 = sumblk ('e2 = yG2 + w2');
Sum3 = sumblk ('ek1 = -e1');
Sum4 = sumblk ('ek2 = -e2');

% Calcolo della matrice P %
P_hinf = connect(G_tot_n,WP,WT,WU,Sum1,Sum2,Sum3,Sum4,{'w1','w2','uG1','uG2'},{'zp1','zp2','zt1','zt2','zu1','zu2','ek1','ek2'});

% Calcolo controllore H-Inf %
[K_hinf,CL_hinf,GAM_hinf] = hinfsyn(P_hinf,2,2);
```

Figura 3.6 – Configurazione ingressi e uscite e calcolo della matrice P

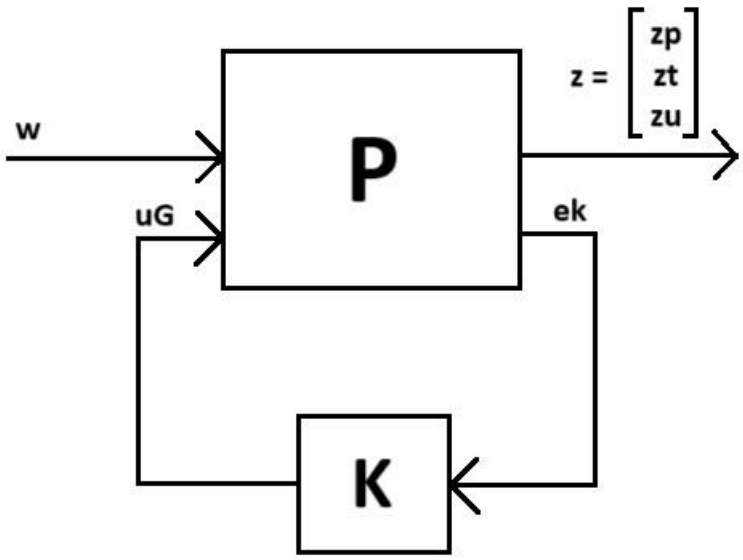

Figura 3.7 – Schema a blocchi per sintesi controllore H-inf

Infine, abbiamo calcolato il controllore con il comando sopra menzionato:

```
% Calcolo controllore H-Inf %
[K_hinf,CL_hinf,GAM_hinf] = hinfsyn(P_hinf,2,2);

% Riduco ordine del controllore
K_hinf = minreal(zpk(tf(K_hinf)),1e-1);
% Tolgo termini non diagonali (guadagni dell'ordine 10e-10!)
K_hinf = [K_hinf(1,1) 0; 0 K_hinf(2,2)];
```

Figura 3.8 – Codice sintesi controllore H-inf approssimato

4.1 Controllore μ-Synthesis con DK-iteration

Il primo passo effettuato per andare a ricavare il controllore ***μ*-Synthesis** è stato **linearizzare** il sistema intorno al **punto di equilibrio** su cui vogliamo andare a **controllarlo** (come già visto precedentemente nell'analisi di Raggiungibilità e Osservabilità).

Linearizzato il sistema in forma di stato, lo scriviamo come matrice di trasferimento G e relativa matrice di incertezza Δ_G:

```
% Trovo matrice di trasferimento del linearizzato
G = ss(A_nom,B_nom,C_nom,D_nom);

[G,Delta_G,Blkstruct]=lftdata(G); % Trovo matrice Delta_G associata ai parametri incerti

G = zpk(G);
```

*Figura 4.1 – Uso del comando **lftdata()** per trovare Δ_G e G*

Ora serve calcolare i pesi di incertezza degli attuatori W_1 e W_2, utilizzando l'incertezza moltiplicativa (concentrata), dovuta dal fatto che nel modello nominale abbiamo trascurato il ritardo:

```
%% Definisco le fdt degli attuatori nominali (senza tempo di ritardo) e quelle perturbate
% Calcolo dei pesi con l'utilizzo del sistema nominale e perturbato
Gm1_n = Km1/(Tm1_i*s+1);
Gm1_p = Km1/(Tm1_i*s+1)*exp(-T1_i*s);
Gm2_n = Km2/(Tm2_i*s+1);
Gm2_p = Km2/(Tm2_i*s+1)*exp(-T2_i*s);

reldiff1 = (Gm1_p-Gm1_n)/Gm1_n;
reldiff2 = (Gm2_p-Gm2_n)/Gm2_n;

% Ricavo i pesi W degli attuatori per ispezione dei reldiff
W1 = makeweight(10e-4, 1, 2.5);
W2 = makeweight(10e-4, 1, 2.5);
```

Figura 4.2 – Calcolo dei pesi W_1 e W_2 per ispezione

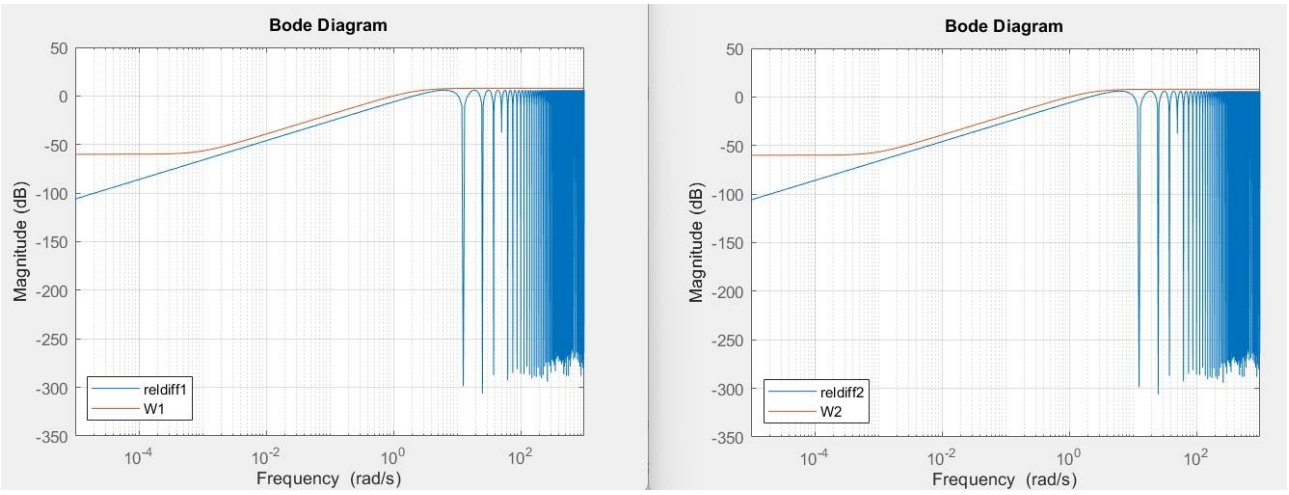

Figura 4.3 – Pesi W_1 e W_2

A questo punto siamo arrivati ad avere il seguente schema a blocchi (considerando che il WP è uguale a quello per la Mixed-Sensitivity):

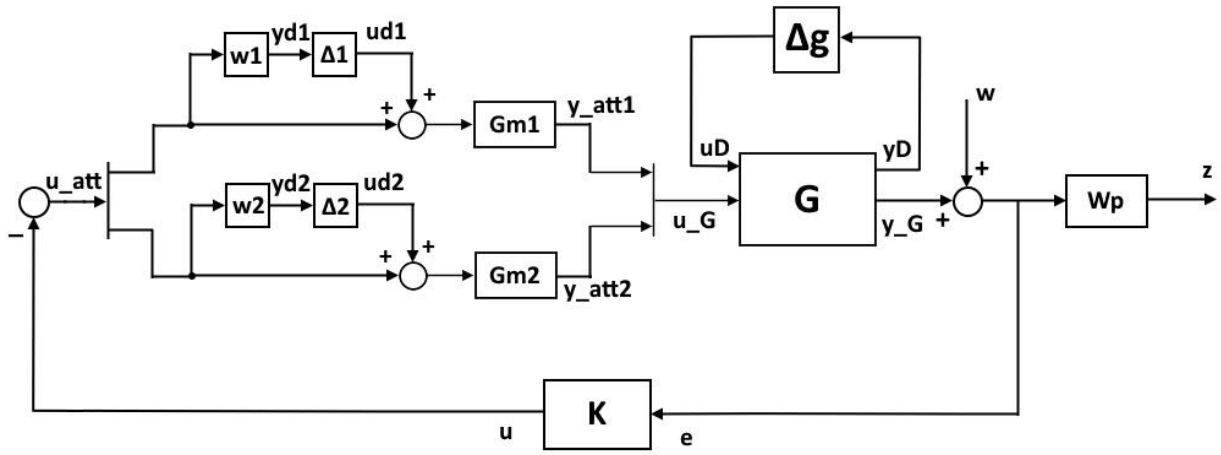

Figura 4.4 – Schema a blocchi per la sintesi del controllo con μ-Synthesis

Ora cerco di tirare fuori un blocco di incertezza strutturato degli attuatori, così da avere uno schema più semplice da gestire con cui usare il comando ***connect()***; lo schema risultante e il codice che lo realizza sono i seguenti:

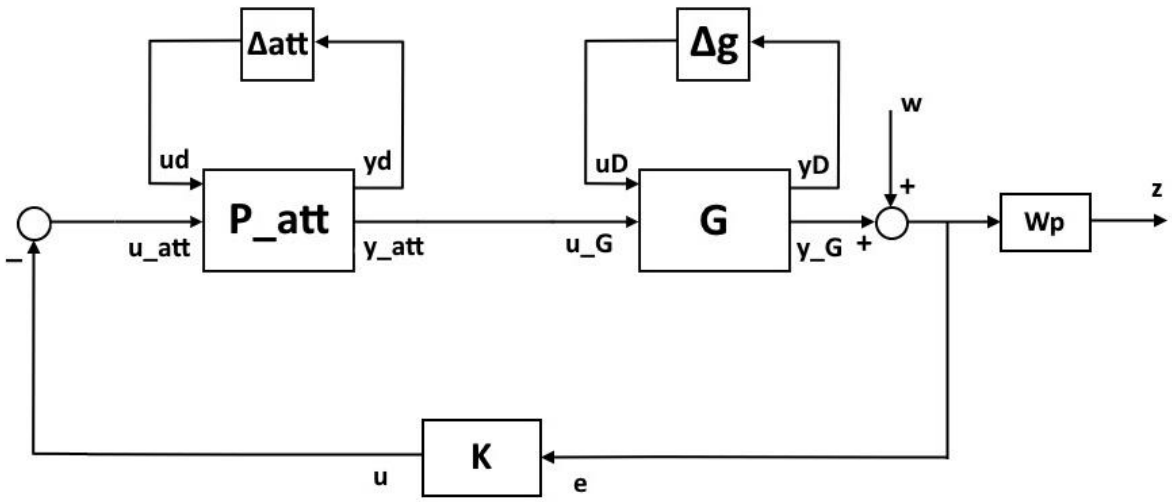

Figura 4.5 – Schema a blocchi dopo aver tirato fuori $\Delta_{att} = blkdiag(\Delta_1, \Delta_2)$

```
% Ricavo la P_att (con il metodo visto a lezione)
P_att = [0      0       W1*Gm1_n       0;
         0      0          0       W2*Gm2_n;
         1      0        Gm1_n         0;
         0      1          0         Gm2_n];

% Definisco la matrice di incertezza strutturata degli attuatori
Delta_att1 = ultidyn('Delta_att1',[1 1]);
Delta_att2 = ultidyn('Delta_att2',[1 1]);
Delta_att = [Delta_att1 0; 0 Delta_att2];
```

Figura 4.6 – Codice per ricavare P_{att} e Δ_{att}

In questo modo, ora possiamo mettere tutto insieme per trovare la forma di Doyle Δ-P-K, dove dentro la matrice P ci sono P_{att}, G e W_p; invece la matrice Δ è fatta come segue:

$$\Delta = \begin{bmatrix} \Delta_{att} & 0 \\ 0 & \Delta_G \end{bmatrix}$$

Il codice che realizza ciò che abbiamo appena spiegato:

```
%% Creo il sistema P con connect

G.u = {'uD1','uD2','u_G1','u_G2'};
G.y = {'yD1','yD2','y_G1','y_G2'};

P_att.u = {'ud1','ud2','u_att1','u_att2'};
P_att.y = {'yd1','yd2','y_att1','y_att2'};

WP.u = {'e1','e2'};
WP.y = {'z1','z2'};

Sum1 = sumblk ('u_att1 = -u1');
Sum2 = sumblk ('u_att2 = -u2');
Sum3 = sumblk ('e1 = w1  + y_G1');
Sum4 = sumblk ('e2 = w2  + y_G2');
Sum5 = sumblk ('u_G1 = y_att1');
Sum6 = sumblk ('u_G2 = y_att2');


P = connect (G,P_att,WP,Sum1,Sum2,Sum3,Sum4,Sum5,Sum6,{'ud1','ud2','uD1','uD2','w1','w2','u1','u2'},{'yd1','yd2','yD1','yD2','z1','z2','e1','e2'});

Delta = [Delta_att,  zeros(2) ; zeros(2) , Delta_G];

P_delta = lft(Delta,P);
```

Figura 4.7 – Codice per ricavare P e costruzione di Δ

Da cui segue la sintesi del controllore con il comando ***musyn()***:

```
% Calcolo Controllore Mu-Synthesis %
[K_DK,CLperf,info] = musyn(P_delta,2,2);

% Riduco ordine del controllore
K_DK = minreal(zpk(tf(K_DK)),0.5);
% Tolgo termini non diagonali (guadagni dell'ordine 10e-10!)
K_DK = [K_DK(1,1) 0; 0 K_DK(2,2)];
```

Figura 4.8 – Codice per sintesi del controllore approssimato

5.1 Risultati LQG

Abbiamo fatto le seguenti prove con il controllore LQG:

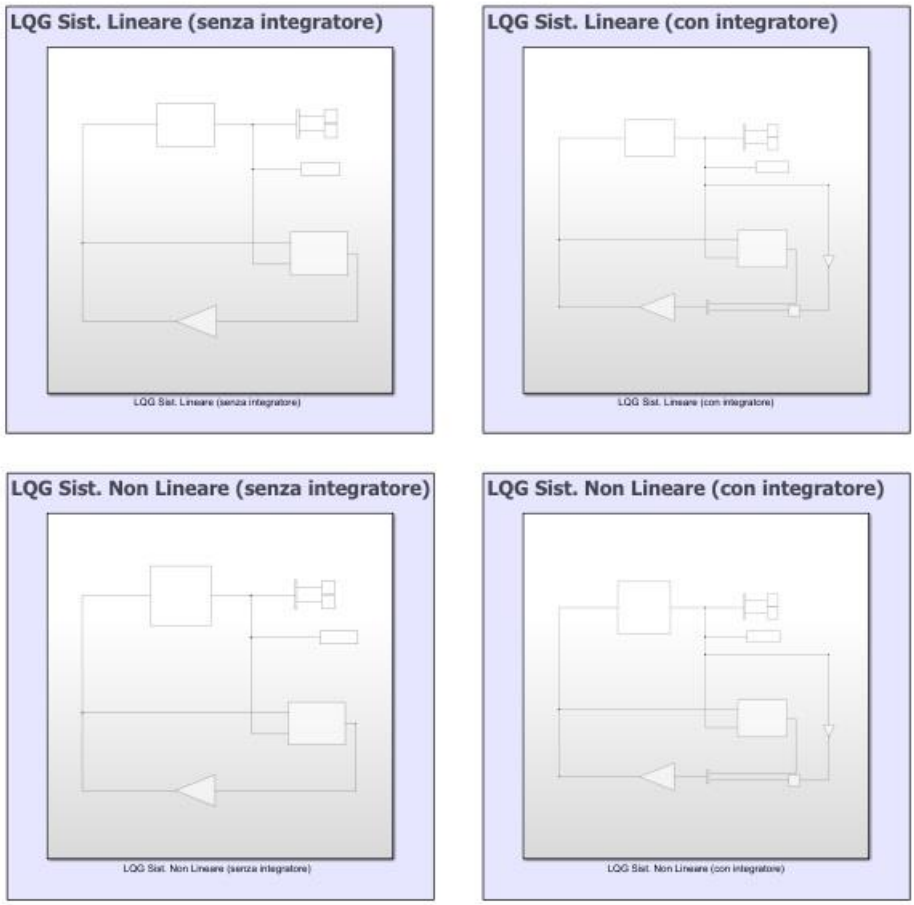

Figura 6.1 – Prove LQG

Abbiamo fatto diverse prove con le varie configurazione mostrate qui sopra, andando a cambiare le condizioni iniziali così da trovare l'esistenza di una **RAS** che fa convergere il sistema nel punto di equilibrio:

- **Prova 1**: condizioni iniziali nulle (punto di equilibrio)
- **Prova 2**: $\boldsymbol{z} = 200\ m,\ \boldsymbol{\theta} = -\pi,\ \dot{\boldsymbol{z}} = 0,\ \dot{\boldsymbol{\theta}} = 0$
- **Prova 3**: $\boldsymbol{z} = -2000\ m,\ \boldsymbol{\theta} = -\pi,\ \dot{\boldsymbol{z}} = 20\ {}^{m}/_{s},\ \dot{\boldsymbol{\theta}} = \pi\ {}^{Rad}/_{s}$

Il risultato finale è che tutte le configurazioni hanno l'equilibrio scelto **Globalmente Asintoticamente Stabile**, a meno del sistema Non Lineare senza integratore che è solo Marginalmente stabile per lo stato z:

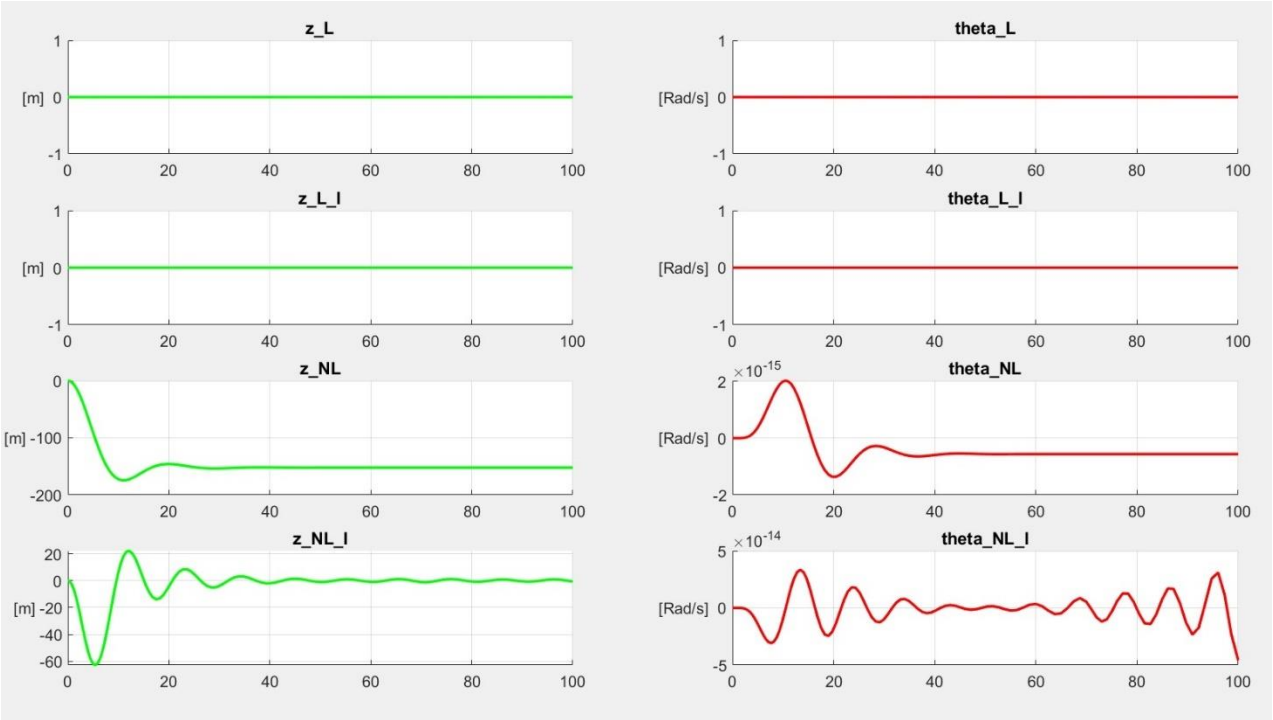

*Figura 5.2 – **Prova 1***

Si può constatare infatti che la z_{NL} non converge a zero.

Questo risultato ce lo aspettiamo perché nel caso **Non Lineare Senza Integratore** abbiamo che non riusciamo a reiettare la costante $-Mg$ nella prima equazione dinamica che caratterizza l'evoluzione dello stato z: non avendo un integratore, l'errore a regime non potrà convergere quindi a zero.

Abbiamo aumentato il tempo di simulazione fino a $1000\ s$ per capire se l'andamento di θ nel caso Non Lineare divergesse:

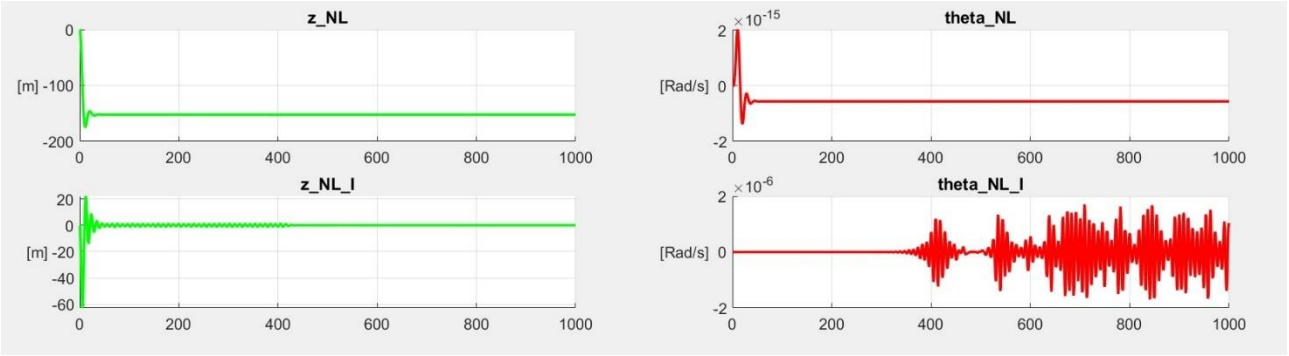

*Figura 5.3 – **Prova 1** (osservazione dello stato θ nel caso Non Lineare con Integratore)*

Ora vediamo le altre prove:

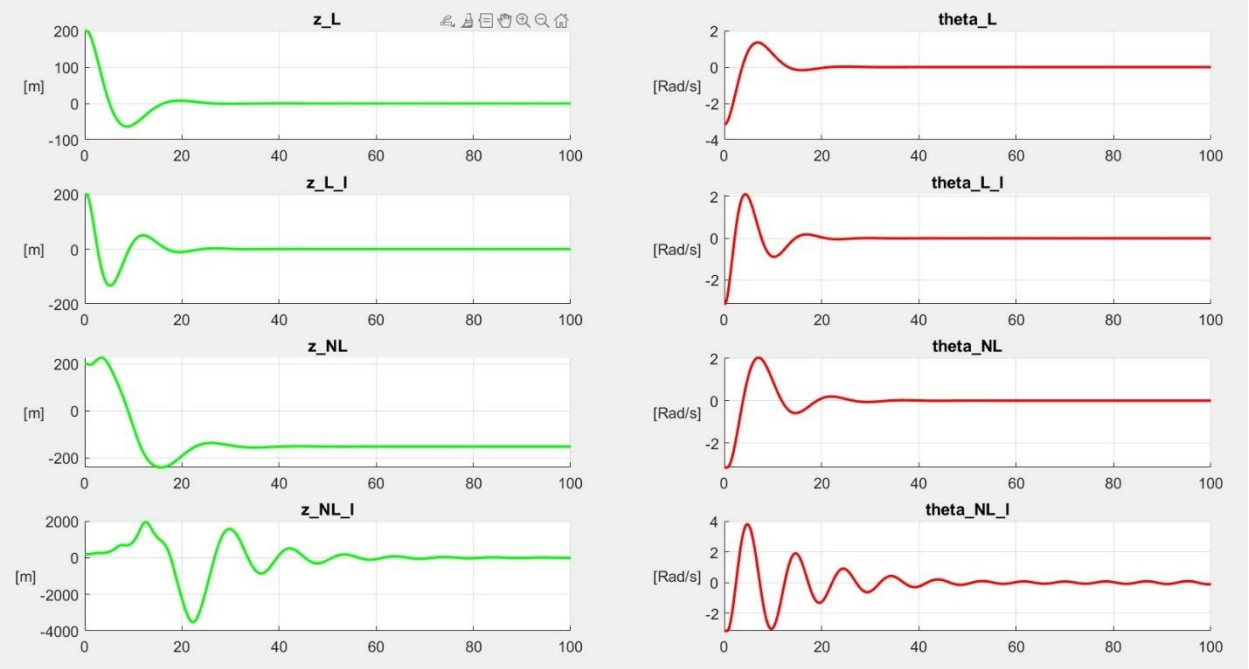

Figura 5.4 – ***Prova 2*** *(T_{aL}~20 s; T_{aNL}~20 ÷ 40 s)*

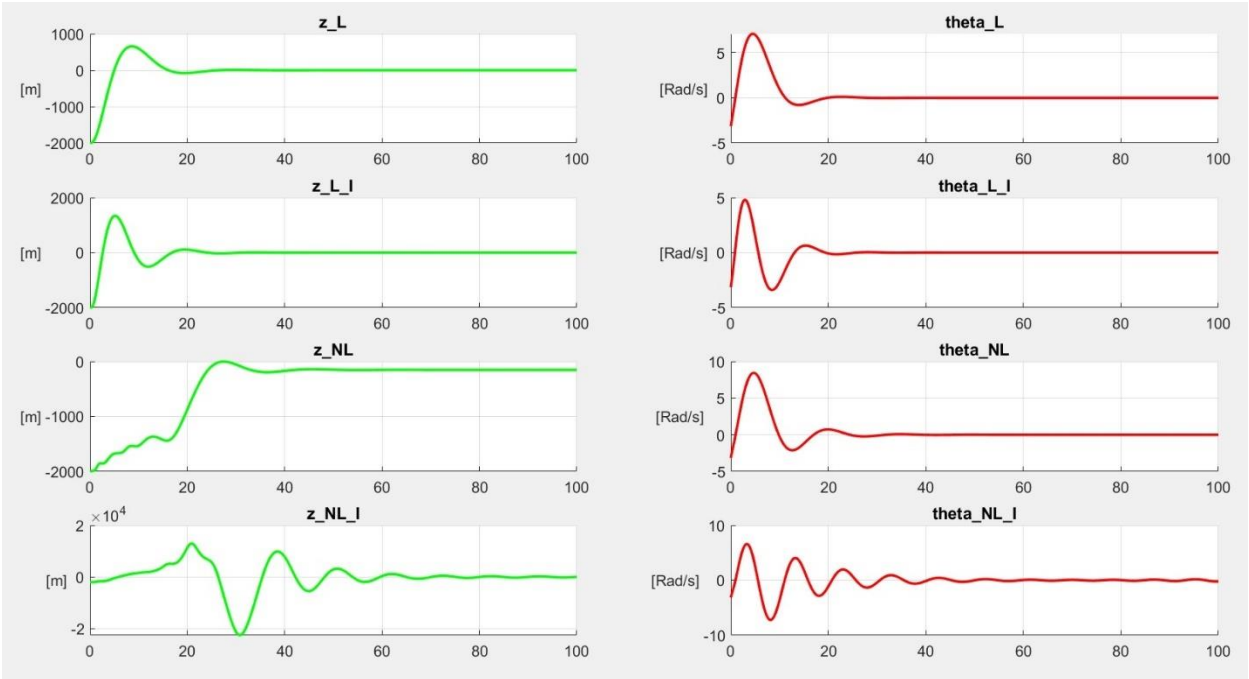

Figura 5.5 – ***Prova 3*** *(T_{aL}~20 s; T_{aNL}~30 ÷ 60 s)*

Quindi, dato che nel caso **Non Lineare Senza Integratore** abbiamo che comunque lo stato θ converge asintoticamente a zero, non serve applicare un integratore per quello stato, anche perché va solo a peggiorare le performance (vedi *Figura 5.4* e *Figura 5.5*); perciò abbiamo ridotto quasi a zero il peso sull'azione integrale sul singolo stato θ così da avere:

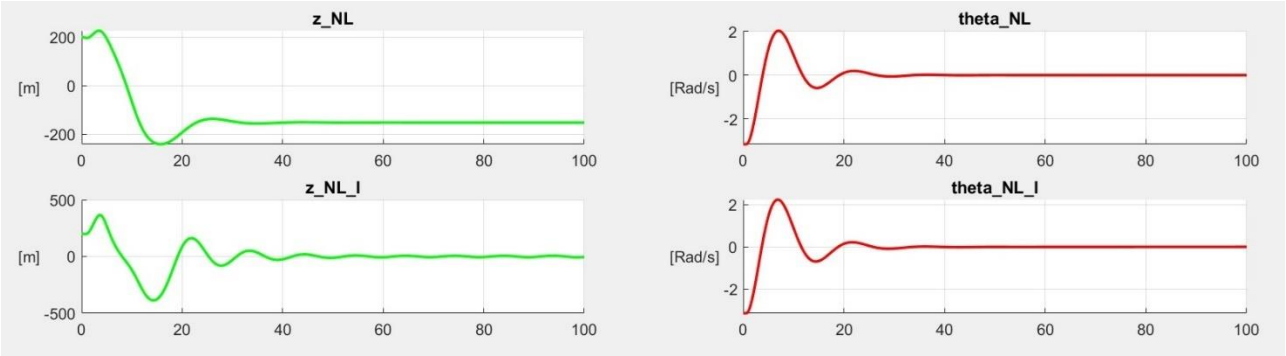

*Figura 5.6 – **Prova 2** togliendo azione integrale a θ*

E' da notare che anche le performance dello stato z sono migliorate.

Abbiamo poi cercato di migliorare la stabilità (reiezione overshoot) del controllore LQG cercando di andare a pesare maggiormente (scelto un fattore 100) gli ingressi di controllo f_m e f_a rispetto agli stati z e θ (mi aspetto di convergere al punto di equilibrio più lentamente):

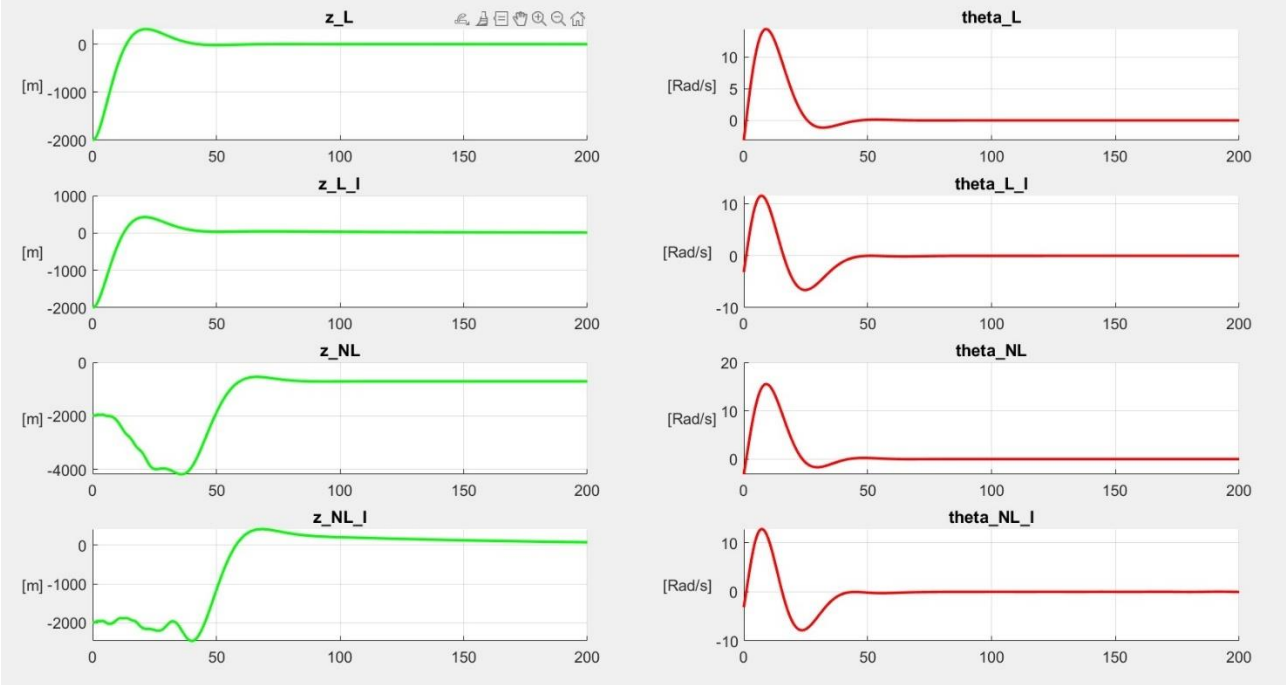

*Figura 5.7 – **Prova 3** abbassando azione integrale a θ e z (T_{aL}~$40\ s$; T_{aNL}~$40 \div 100\ s$)*

Possiamo quindi vedere un andamento più smooth per tutte e 4 le configurazioni: abbiamo guadagnato in stabilità di convergenza, ma abbiamo perso nelle prestazioni di tracking.

Infatti, se prima avevamo un **tempo di assestamento** di $\sim 20 \div 60\ s$, ora invece è aumentato in un intervallo di $\sim 40 \div 100\ s$.

6.1 Risultati Mixed-Sensitivity e H-inf

7.1 Risultati μ-Synthesis con DK-iteration