Progetto CSI

Jacopo Conti Matteo Gafforio

Anno Accademico 2023/2024

Indice

1 Introduzione

Il presente progetto si concentra sullo studio e lo sviluppo di controllori per un convertiplano, un sistema aerodinamico in grado di transizionare tra configurazioni di volo ad ala fissa e rotore. Il sistema risulta visibile in figura 1. La complessità intrinseca di tale veicolo richiede l'implementa-

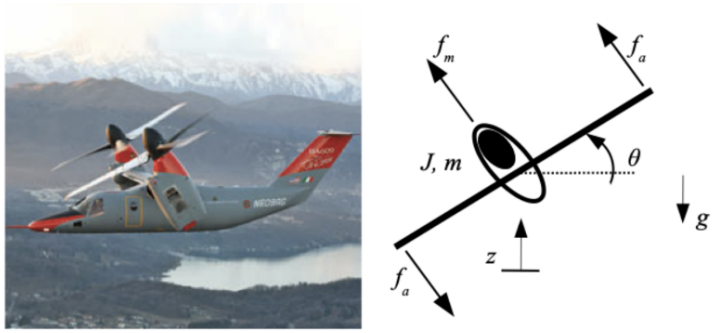

Figura 1: schema di controllo per controllore mix-sensitivity

zione di strategie di controllo efficaci per garantire una stabilità e una performance ottimali durante le diverse fasi di volo. L'obiettivo principale è implementare diverse metodologie di controllo e condurre un'approfondita valutazione delle loro prestazioni in termini di stabilità, risposta dinamica e capacità di adattamento alle variazioni nelle condizioni operative. Nel corso della relazione, esploreremo dettagliatamente i passi intrapresi l'implementazione dei controllori, analizzando le scelte di progetto, i metodi utilizzati e i risultati ottenuti. I capitoli successivi si concentreranno sulle metodologie specifiche utilizzate per la progettazione dei controllori, le simulazioni condotte per valutare le performance, e una discussione dei risultati ottenuti.

2 Analisi del sistema

2.1 Modello dinamico

Le equazioni dinamiche del sistema sono le seguenti:

$$M\ddot{z} + b\dot{z} = f_m \cos\theta - Mg \tag{1}$$

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

In queste equazioni notiamo la presenza dei parametri f_m e f_a che rappresentano le forze che vengono impresse dagli attuatori del convertiplano, queste sono quindi i nostri ingressi di controllo. Abbiamo anche le funzioni di trasferimento degli attuatori, che risultano:

$$G_{m1} = \frac{K_{m1}}{(T_{m1}s + 1)} e^{-T_1 s} \tag{3}$$

$$G_{m2} = \frac{K_{m2}}{(T_{m2}s + 1)} e^{-T_2 s} \tag{4}$$

In uscita al modello, implementato in ambiente simulink, abbiamo inoltre inserito dei rumori bianchi per modellare eventuali errori di misura di sensori.

2.2 Scelta degli stati, uscite e linearizzazione

Lo stato è il seguente vettore:

$$[z; \dot{z}; \theta; \dot{\theta}] \tag{5}$$

mentre come uscite del sistema abbiamo scelto:

$$[z; \theta] \tag{6}$$

la linearizzazione del sistema intorno al punto di equilibrio risulta:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -b/m & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -beta/J \end{bmatrix} \tag{7}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1/m & 0 \\ 0 & 0 \\ 0 & 2l/J \end{bmatrix} \tag{8}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{10}$$

2.3 Controllo raggiungibilità e osservabilità del sistema

In questa sezione, approfondiremo l'analisi della raggiungibilità e dell'osservabilità del sistema, focalizzandoci sulle implicazioni pratiche per la progettazione del controllore. Calcoliamo le matrici di raggiungibilità e di osservabilità del sistema con le scelte degli stati e delle uscite effettuate:

$$R = \begin{bmatrix} 1 & 0 & -b/m & 0 \\ 0 & 2l & 0 & 2\beta/J \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2l \end{bmatrix} \tag{11}$$

$$O = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

Le due matrici hanno rango pieno, quindi il sistema è completamente raggiungibile ed osservabile, in questo modo possiamo effettuare un controllo a zero.

3 Controllore LQG

In questo capitolo esploreremo la sintesi di controllori per sistemi dinamici mediante il metodo Linear Quadratic Gaussian (LQG). Questo approccio ottimale, che unisce la teoria del controllo quadratico lineare (LQ) con la stima di stato gaussiana (G), si focalizza sulla minimizzazione di un funzionale di costo quadratico.

Per effettuare un controllo su sistema abbiamo eseguito i seguenti passaggi:

1. **Definizione del Sistema Totale:** Il primo passo della nostra metodologia di sintesi del controllore è stato definire le matrici del sistema totale, che comprende anche gli attuatori, utilizzando le regole di composizione per sistemi in serie. Questo processo ha consentito di ottenere una rappresentazione completa e accurata del sistema dinamico considerato, integrando gli effetti degli attuatori nell'analisi.

2. **Definizione di Deviazioni Standard e Matrici Q e R:** Successivamente, abbiamo proceduto definendo le deviazioni standard per gli ingressi e i sensori. Utilizzando queste informazioni, abbiamo composto le matrici di covarianza Q e R, le quali riflettono la variazione degli ingressi e delle misurazioni del sistema. Queste matrici sono poi state combinate in una singola matrice QWV, incorporando così le informazioni sulla variabilità nel nostro modello.

3. **Definizione dei Pesi per il Funzionale di Costo:** Per adattare il controllore alle nostre specifiche esigenze, abbiamo definito i pesi per il funzionale di costo. Questi pesi riflettono l'importanza relativa degli obiettivi di controllo e influenzano il comportamento del sistema controllato. La corretta scelta dei pesi è cruciale per ottenere un controllore ottimale in base alle specifiche del progetto.

4. **Calcolo del Guadagno per LQG e del Filtro di Kalman:** Infine, abbiamo utilizzato il comando LQG per calcolare il guadagno ottimale per il nostro sistema. Il guadagno risultante è stato progettato per minimizzare il funzionale di costo ponderato secondo i pesi specificati. Inoltre, abbiamo calcolato il filtro di Kalman associato, che ottimizza la stima dello stato del sistema basandosi sulle misurazioni disponibili.

Listing 1: Codice implementazione LQG

```
%% Controllo LQG %%
clear
close all
clc

%% paramtri sistema
% Definizione parametri convertiplano %
J = 5000;
m = 2000;
b = 150;
beta = 15;
l = 10;

g = 9.81;

params_plano = [J, m, b, beta, l, g];

% Definizione parametri attuatori %
Km1 = 500;
Km2 = 100;
T1 = 0.5;
T2 = 0.5;
Tm1 = 5;
Tm2 = 5;

params_attuatori = [Km1, Km2, T1, T2, Tm1, Tm2];


% Condizioni iniziali %
F_0 = [0; 0];
```

```
31
32 dq_0 = [0; 0];
33 q_0 = [2; -pi/18];
34
35 %%
36 s = tf('s');
37
38 % Linearizzazione sistema convertiplano intorno al punto z = theta = 0 %
39 A_nom = [0 1 0 0; ...
40          0 -b/m 0 0; ...
41          0 0 0 1; ...
42          0 0 0 -beta/J];
43
44 B_nom = [0      0    ; ...
45          1/m    0    ; ...
46          0      0    ; ...
47          0    2*l/J];
48
49 C_nom = [1 0 0 0; ...
50          0 0 1 0];
51
52 D_nom = zeros(2);
53
54 % Creazione sistemi attuatori in forma di stato
55 G1 = tf([Km1], [Tm1 1]) * exp(-s*T1);
56 G2 = tf([Km2], [Tm2 1]) * exp(-s*T2);
57 G_att = blkdiag(G1, G2);
58
59 G_att = ss(G_att);
60
61 A_att = G_att.A;
62 B_att = G_att.B;
63 C_att = G_att.C;
64 D_att = G_att.D;
65
66 % Composizione in serie del sistema dinamico e degli attuatori (forma di
67 % stato)
68 A_tot = [      A_att,          zeros(2,4);
69             B_nom*C_att,          A_nom    ];
70
71 B_tot = [B_att;  B_nom*D_att];
72
73 C_tot = [D_nom*C_att,    C_nom];
74
75 D_tot = [0, 0;
76          0, 0];
77
78 G = ss(A_tot, B_tot, C_tot, D_tot);
79
80 % Definisco le matrici di covarianza dei rumori in ingresso e dei sensori
81 dev_std_fm = 100;
82 dev_std_fa = 10;
83 dev_std_z = 1;
84 dev_std_theta = 0.3;
85
86 % Matrice covarianza rumori in ingresso (gli errori di processo sono
87 % riferiti solo alle equazioni di z: e theta:)
88 Q = blkdiag(0, 0, 0, dev_std_fm ^ 2, 0, dev_std_fa ^ 2);
89
90 % Matrice covarianza errori di misura (y1 = z; y2 = theta)
91 R = blkdiag(dev_std_z ^ 2, dev_std_theta ^ 2);
92
93 % Big matricione di covarianza
94 QWV = blkdiag(Q, R);
95
96 % Matrice di peso per stati e ingressi, dove pesiamo solo gli ingressi e
97 % gli stati z e theta
98 Q_pesi = blkdiag(zeros(2), 1, 0, 1, 0, eye(2));
99
100 % 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;
```

Al fine di affinare ulteriormente le prestazioni del nostro controllore, abbiamo esteso il metodo LQG includendo un termine integratore. Questo approccio è stato implementato attraverso la sintesi del guadagno per LQG con integratore, il quale ha dimostrato di essere particolarmente efficace nel garantire una risposta di sistema robusta e priva di errore a regime. I passaggi fatti sono:

1. **Definizione dei Pesi per le Uscite** Abbiamo introdotto due pesi specifici per le uscite del sistema con lo scopo di minimizzare l'errore a regime[1].
2. **Calcolo del Guadagno per LQG con Integratore** Utilizzando il comando LQI, abbiamo proceduto al calcolo del guadagno ottimale per il sistema, tenendo conto degli effetti dell'integratore. Questo passo è stato fondamentale per ottenere un controllore che potesse gestire efficacemente sia gli aspetti dinamici che quelli statici del sistema.

Listing 2: Codice implementazione LQG con integratore

```
% 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(2), 1, 0, 1, 0, 1, 0.001);
R_u = eye(2);

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

Determinati i valori del nostro controllore, abbiamo esteso il nostro approccio implementando il sistema su Simulink. L'utilizzo di Simulink ci ha permesso di tradurre in pratica le specifiche del nostro controllore e verificare il suo funzionamento in un contesto simulato.

Abbiamo ampliato la nostra analisi delle prestazioni del controllore introducendo quattro configurazioni distintive (2). Queste configurazioni sono state progettate per esplorare il comportamento del sistema in diverse condizioni e fornire una valutazione dettagliata delle capacità di ciascun controllore. Le quattro configurazioni includono:

1. **Modello Lineare con Controllore LQG:** Questa configurazione preserva l'assunzione di linearità del sistema, consentendoci di valutare l'efficacia del controllore LQG in un contesto ideale e agevolmente analizzabile.
2. **Modello Non Lineare con Controllore LQG:** In questa configurazione, abbiamo considerato il sistema nella sua completa non linearità. Questo approccio ci permette di esaminare come il controllore LQG gestisce le variazioni dinamiche e non lineari del sistema, offrendo una prospettiva più realistica delle sue prestazioni.
3. **Modello Lineare con Controllore LQG con Integratore:** Introducendo un termine integratore, abbiamo esplorato come il controllore LQG affronta la gestione degli errori a regime. Questa configurazione è stata progettata per migliorare la precisione del sistema e valutare la sua capacità di mantenere una risposta accurata nel tempo.
4. **Modello Non Lineare con Controllore LQG con Integratore:** La combinazione di non linearità del sistema e l'introduzione del termine integratore ci permette di valutare come il controllore LQG con integratore gestisce scenari più complessi e dinamici.

[1] Dai risultati abbiamo notato che l'uscita θ se pesata peggiorava il controllore, quindi è stato dato un valore più piccolo possibile in modo da dargli poco peso.

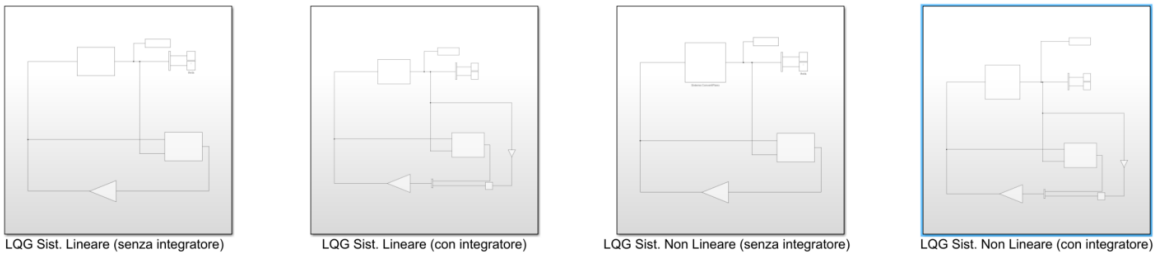

Figura 2: Configurazioni LQG su Simulink

3.1 Risultati

Vediamo i risultati ottenuti con questo controllore nel caso di parametri certi e incerti: intanto osserviamo, in figura 3, il comportamento del sistema nominale con il controllo senza integratore. [!h] Vediamo che abbiamo un tempo di assestamento di circa 15 secondi per entrambe le uscite,

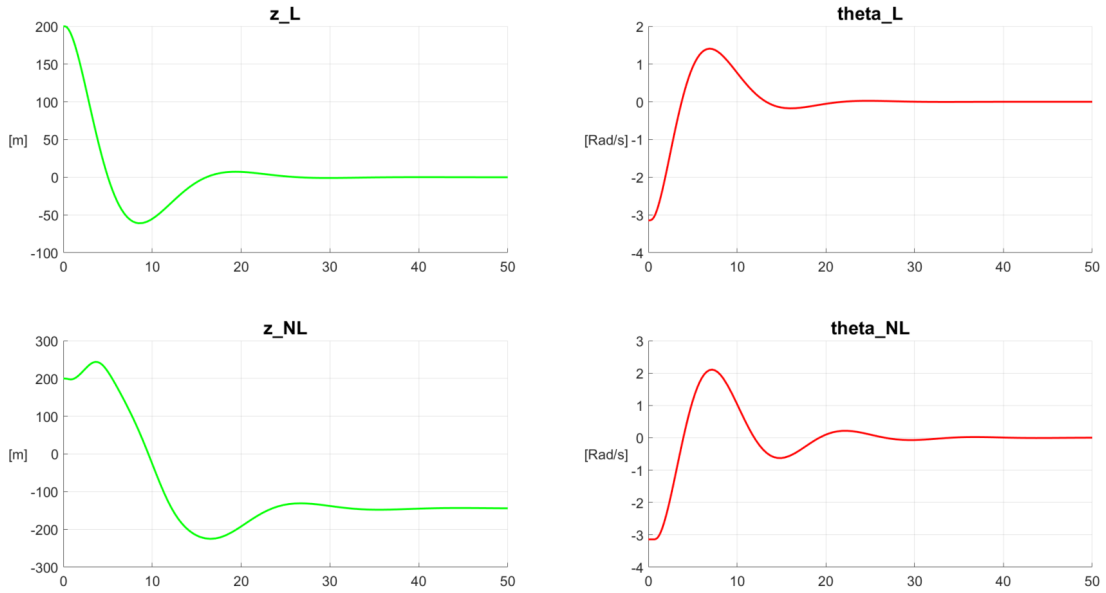

Figura 3: risposte LQG senza integratore, modello nominale

sia per il sistema lineare, sia per quello non lineare, tuttavia z risulta solo marginalmente stabile nel modello non lineare, infatti si assesta ad un valore diverso da zero. Questo problema possiamo aspettarci che venga risolto inserendo la componente integrale che dovrebbe andare ad azzerare l'errore a regime, i risultati di questa prova sono visibili in figura 4. Esattamente come ci aspettavamo, anche l'errore a regime di z per il modello non lineare scompare, il "costo" di ciò è l'aver aumentato il tempo di assestamento[2] delle altre variabili che passa da circa 15 a circa 20 secondi. Abbiamo fatto molte prove variando i parametri che abbiamo scelto come incerti, ovvero b e β, inseriamo in figura 5 i risultati con i parametri raddoppiati rispetto al valore nominale. Da questi risultati osserviamo che il sistema non degrada in modo significativo le sue prestazioni, possiamo quindi concludere che il sistema con controllo LQG rispetta RP.

[2]considerando una fascia di più o meno il dieci per cento rispetto al valore iniziale

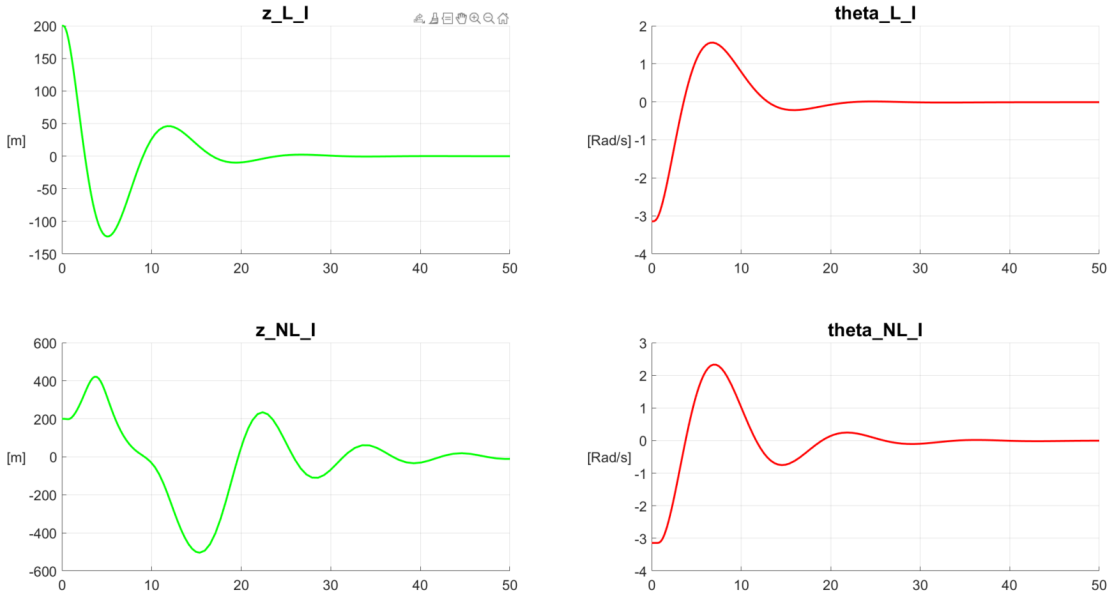

Figura 4: risposte LQG con integratore, modello nominale

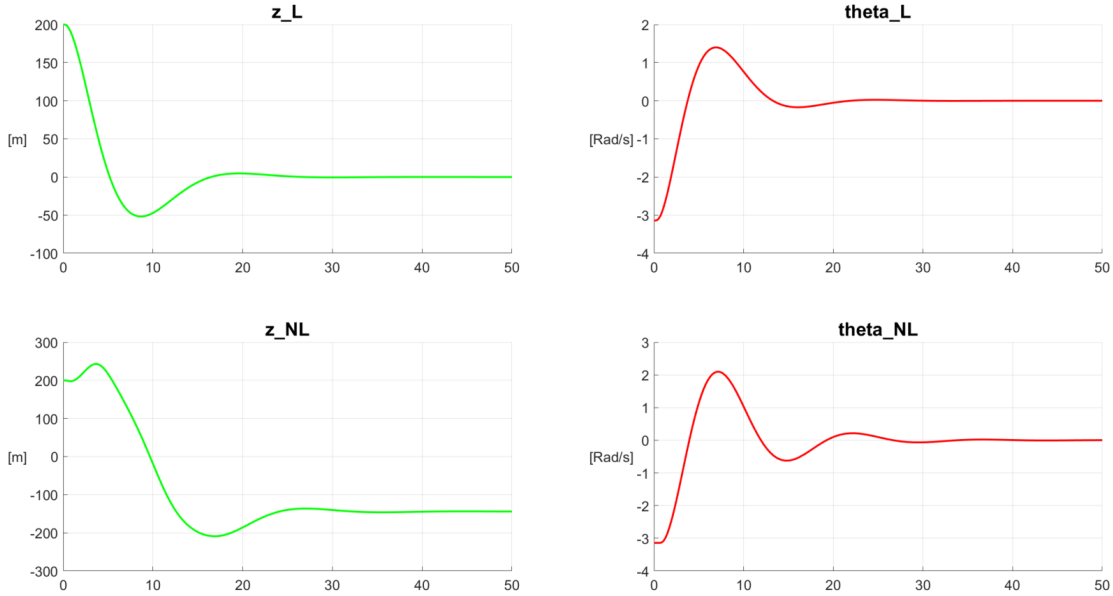

Figura 5: risposte LQG senza integratore, modello incerto

4 Controllori Mixed-sensitivity e H-inf

Nel contesto della progettazione del controllore, adotteremo l'approccio della mix sensitivity, attribuendo particolare importanza alla sensitività, alla sensitività complementare e al termine k·sensitività. Questo approccio offre un quadro teorico robusto per la progettazione di controllori, permettendoci di considerare simultaneamente la reattività del sistema alle perturbazioni, la sua capacità di sopprimere gli effetti indesiderati e l'entità degli sforzi di controllo necessari. Inoltre da notare come non verranno prese in considerazione eventuali incertezze di modello per questo approccio. In figura 6 possiamo vedere lo schema del sistema ed i pesi utilizzati per sintetizzare il controllore.

Nel corso di questo capitolo, oltre alla progettazione del controllore basato sulla mix sensitivity,

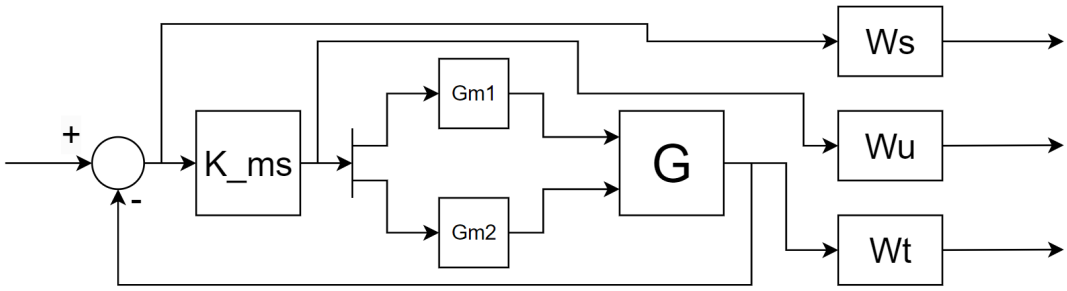

Figura 6: schema di controllo per controllore mix-sensitivity

esploreremo anche l'approccio H_∞ strutturato. La progettazione del controllore H_∞ strutturato verrà condotta in parallelo con quella basata sulla mix sensitivity, e i risultati saranno attentamente analizzati. Verificheremo in particolare se le soluzioni ottenute attraverso entrambi gli approcci mostrano somiglianze significative, garantendo così una validazione incrociata delle scelte di progetto effettuate.

Listing 3: Codice implementazione mix-sensitivity

```
1
2       %% Mix sensitivity
3  clear
4  close all
5  clc
6
7  %% paramtri sistema
8  % Definizione parametri convertiplano %
9  J = 5000;
10 m = 2000;
11 b = 150;
12 beta = 15;
13 l = 10;
14
15 g = 9.81;
16
17 params_plano = [J, m, b, beta, l, g];
18
19 % Definizione parametri attuatori %
20 Km1 = 500;
21 Km2 = 100;
22 T1 = 0.5;
23 T2 = 0.5;
24 Tm1 = 5;
25 Tm2 = 5;
26
27 params_attuatori = [Km1, Km2, T1, T2, Tm1, Tm2];
28
29
30 %% Definisco il sistema linearizzato nel PE
31 % PE = (0,0,0,0)   %(x,x_d,theta,theta_d)
32
33 A = [0  1  0  0;  ...
34       0 -b/m 0  0;  ...
35       0  0  0  1;  ...
36       0  0  0 -beta/J];
37
38 B = [0  0;  ...
39       1  0;  ...
40       0  0;  ...
41       0  2*l];
42
43 C = [1  0  0  0;  ...
```

```
      0 0 1 0];

D = zeros(2);

% trovo fdt nominale del linearizzato

sys_n = ss(A,B,C,D);

G_n = zpk(sys_n);

%% Definisco le fdt degli attuatori nominali (senza tempo di ritardo)
s = tf('s');

Gm1_n = Km1/(Tm1*s+1);
Gm2_n = Km2/(Tm2*s+1);

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

%% Trovo fdt totale del sistema

G_tot_n = G_n*G_attuators_n;
%bodemag (G_tot_n)

%% Trovo controllore Mix sensitivity


wU=tf(1);
WU=blkdiag(wU,wU); %matrice peso ks
A1=1e-4;
M1=1.5;
wB1=0.3;
A2=1e-4;
M2=1.5;
wB2=0.5;
wP1=makeweight(1/A1,wB1,1/M2);
wP2=makeweight(1/A2,wB2,1/M2);
WP=blkdiag(wP1,wP2); %matrice peso s
wT = (s+1)/2/(0.001*s+1);
WT = [wT 0;0 wT];

[K_ms,CL_ms,GAM_ms,~] = mixsyn(G_tot_n,WP,WU,WT);

K_ms = minreal(zpk(tf(K_ms)),1e-1);

K_ms = [K_ms(1,1) 0; 0 K_ms(2,2)];

S = inv(eye(2) + G_tot_n*K_ms);

KS =K_ms*S;

T = eye(2) - S;


step(T(1,1))

%% controllo H-inf

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');

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'});

[K_hinf,CL_hinf,GAM_hinf] = hinfsyn(P_hinf,2,2);

K_hinf = minreal(zpk(tf(K_hinf)),1e-1);

K_hinf = [K_hinf(1,1) 0; 0 K_hinf(2,2)];
```

Nello specifico, abbiamo seguito il seguente procedimento:

1. Definizione e calcolo della linearizzazione del sistema nel punto di equilibrio.
2. Calcolo della matrice di trasferimento utilizzando il comando `zpk`.
3. Definizione delle funzioni di trasferimento degli attuatori e incorporazione in una matrice di trasferimento diagonale 2×2.
4. Calcolo della matrice totale del sistema, includendo gli effetti degli attuatori.
5. Definizione dei pesi per la mix-sensitivity sulla base della sensitività e sensitività complementare del sistema con controllore LQG. In particolare le funzioni peso sono state scelte con il comando `makeweight`, come upper bound proprio di S e T(figura 7).
6. Utilizzo del comando `mixsyn` per ottenere il controllore basato sulla mix-sensitivity.
7. Applicazione del comando `minreal` per ridurre il grado del controllore.

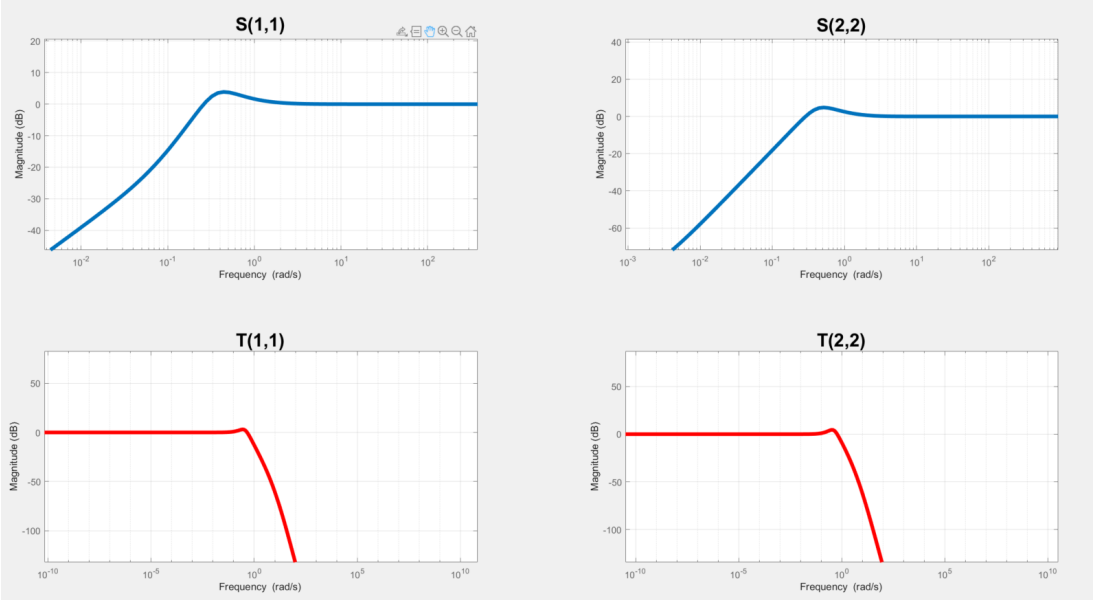

Figura 7: diagrammi di modulo delle funzioni S e T per sistema ed LQG

Come riprova, abbiamo provato a trovare un controllore con il comando `Hinfsyn` , se il procedimento effettuato è corretto dovremmo trovare un controllore molto simile al precedente.

1. Creazione manuale del sistema, considerando la struttura automaticamente generata dal comando `mixsyn`.
2. Utilizzo del comando `Hinfsyn` per ottenere il controllore H_∞ strutturato.

3. Applicazione del comando `minreal` per ridurre il grado del controllore H_∞ strutturato.

In questo caso i due controllori trovati sono esattamente gli stessi, questo rispecchia le nostre aspettative ed è un indizio della buona riuscita della sintesi.

Per verificare il funzionamento abbiamo creato, come per LQG, due modelli su simulink, uno con modello lineare e uno non lineare.

4.1 μ analisi

Una volta trovato il controllore per analizzarlo abbiamo implementato il codice per eseguire la μ analisi. Il μ-analysis valuta la performance e la stabilità di un sistema di controllo considerando perturbazioni specifiche nel modello del sistema. Per far questo abbiamo definito il sistema con connect andando prima a ricavare le incertezze degli attuatori avendo trascurato la dinamica data dal tempo di ritardo. Nel codice sottostante vediamo come abbiamo implementato il problema.

```
1      %% paramtri sistema
2  % Definizione parametri convertiplano %
3  J = 5000;
4  m = 2000;
5  b = ureal('b',150,'Percentage',[-5,5]);
6  beta=ureal('beta',15,'Percentage',[-5,5]);
7  l = 10;
8
9  g = 9.81;
10
11 %params_plano = [J, m, b, beta, l, g];
12
13 % Definizione parametri attuatori %
14 Km1 = 500;
15 Km2 = 100;
16 T1 = 0.5;
17 T2 = 0.5;
18 Tm1 = 5;
19 Tm2 = 5;
20
21 %params_attuatori = [Km1, Km2, T1, T2, Tm1, Tm2];
22
23 %% Definisco il sistema linearizzato nel PE
24 % PE = (0,0,0,0)  %(x, x_d, theta, theta_d)
25
26 A = [0    1   0     0     ;...
27      0 -b/m 0     0     ;...
28      0    0   0     1     ;...
29      0    0   0 -beta/J];
30
31 B = [ 0     0    ;...
32       1/m   0    ;...
33        0    0    ;...
34        0   2*l/J];
35
36 C = [1 0 0 0 ;...
37      0 0 1 0];
38
39 D = zeros(2);
40
41 % trovo fdt nominale del linearizzato
42
43 sys_n = ss(A,B,C,D);
44
45 [P,Delta_G,Blkstruct]=lftdata(sys_n);
46
47 G_P = zpk(P);
48
49
50 %% Definisco le fdt degli attuatori nominali (senza tempo di ritardo) e
       perturbate
51 s = tf('s');
52
```

```
Gm1_n = Km1/(Tm1*s+1);
Gm1_p = Km1/(Tm1*s+1)*exp(-T1*s);
Gm2_n = Km2/(Tm2*s+1);
Gm2_p = Km2/(Tm2*s+1)*exp(-T2*s);

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

% ricavo i pesi Wi degli attuatori
Wi1 = makeweight(10^-4,1,2.5);
Wi2 = makeweight(10^-4,1,2.5);

% creo la upper lft degli attuatori
P_att = [0 0 Wi1*Gm1_n 0; 0 0 0 Wi2*Gm2_n;1 0 Gm1_n 0; 0 1 0 Gm2_n];

Delta_att1 = ultidyn('Delta_att1',[1 1]);
Delta_att2 = ultidyn('Delta_att2',[1 1]);
Delta_att = [Delta_att1 0; 0 Delta_att2];

%% definisco il peso Wp di prestazione
A1=1e-4;
M1=2;
wB1=0.1;
A2=1e-4;
M2=2;
wB2=0.13;
wP1=makeweight(1/A1,wB1,1/M2);
wP2=makeweight(1/A2,wB2,1/M2);
WP=blkdiag(wP1,wP2); %matrice peso s

%% creo il sistema con connect

G_P.u = {'uD1','uD2','u_G_P1','u_G_P2'};
G_P.y = {'yD1','yD2','y_G_P1','y_G_P2'};

P_att.u = {'ud1','ud2','u_P_att1','u_P_att2'};
P_att.y = {'yd1','yd2','y_P_att1','y_P_att2'};

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

Sum1 = sumblk ('u_P_att1 = -u1');
Sum2 = sumblk ('u_P_att2 = -u2');
Sum3 = sumblk ('e1 = w1  + y_G_P1');
Sum4 = sumblk ('e2 = w2  + y_G_P2');
Sum5 = sumblk ('u_G_P1 = y_P_att1');
Sum6 = sumblk ('u_G_P2 = y_P_att2');


P = connect (G_P,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'});

% Creo incertezza di performance per RP
Delta_perf = ultidyn('Delta_perf', [2 2]);


% Struttura matrice Delta_RP
Delta_RP = [ Delta_att,  zeros(2), zeros(2) ;
             zeros(2) , Delta_G, zeros(2);
             zeros(2), zeros(2), Delta_perf];
blk_RP = [-1 0; -1 0; -1 0; -1 0; 2 2];

% Struttura matrice Delta
Delta = [Delta_att,  zeros(2) ; zeros(2) , Delta_G];
blk_RS = [-1 0; -1 0; -1 0; -1 0];

N = lft(P, K_ms);
N_zpk = zpk(N);

```

```
%NS
N22_ms = N_zpk(5:6, 5:6);

%RS e RP
N11 = N(1:4, 1:4);

mubnds_mix = mussv(N, blk_RP);
muRP_mix = mubnds_mix(:,1);
[muRPinf_mix, MuRPw_mix] = norm(muRP_mix, inf);

mubnds_mix = mussv(N11, blk_RS);
muRS_mix = mubnds_mix(:,1);
[muRSinf_mix, MuRSw_mix] = norm(muRS_mix, inf);

RP_mix = lft(Delta_RP,N);
[stabmargRP_mix, wcuRP_mix] = robstab(RP_mix);

RS_mix = lft(Delta,N11);
[stabmargRS_mix, wcuRS_mix] = robstab(RS_mix);
```

4.2 Risultati

Per analizzare il funzionamento del controllore Mix-sensitivity e poi andare ad ottenere il controllore con migliori performance, mantenendo comunque una buona robustezza alle prestazioni, siamo partiti dal definire i pesi sulla base della funzione sensibilità dell'LQG (figura 8).

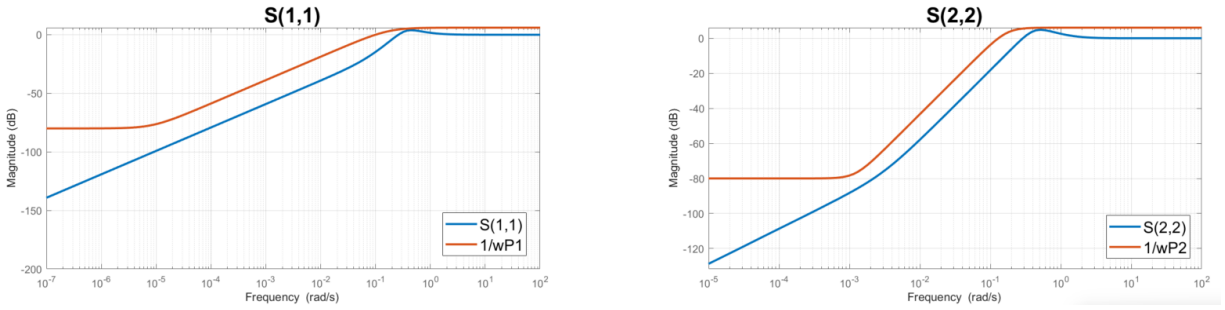

Figura 8: Scelta dei pesi basata su sensibilità LQG

I valori del peso ricavato sono:

$$Per\ Z:\ A = 10^{-}4\ M = 2\ wB = 0.15 \tag{13}$$

$$Per\ \theta:\ A = 10^{-}4\ M = 2\ wB = 0.15 \tag{14}$$

Il controllore che abbiamo ricavato, semplificato tramite il comando `minreal`, lasciando la tolleranza di default[3] è:

$$K11 = \frac{39.016(s + 1.092e-06)(s + 0.075)(s + 0.2)}{(s + 35.41)(s + 1.299e-05)(s^2 + 0.995s + 0.4672)} \tag{15}$$

$$K22 = \frac{1.7776(s + 0.2)(s + 0.003)(s + 4.555e-06)}{(s + 1.68)(s + 1.299e-05)(s^2 + 1.12s + 0.7161)} \tag{16}$$

Sintetizzato poi il controllore, ne abbiamo svolto la μ analisi. Analizzando i risultati ottenuti abbiamo visto se c'era la possibilità di aumentare le performance, aumentando la banda dei pesi, oppure diminuirla, andando a ridurre la banda.

Il primo test fatto ha dato i seguenti risultati:

- La prima cosa da verificare era la nominale stabilità del sistema, tramite il diagramma di Nyquist. Dai grafici è facile verificare la NS.

[3] Abbiamo considerato solo la diagonale, dato che i guadagni sulla antidiagonale erano molto piccoli

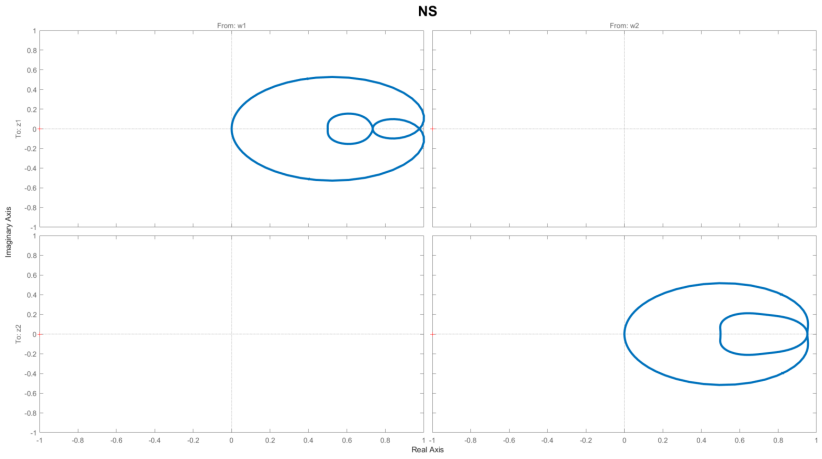

Figura 9: Nyquist primo test per Mix sensitivity

- Successivamente siamo andati a testare la stabilità robusta, graficando i valori singolari strutturati per ogni frequenza. Abbiamo verificato così di avere molto stabilità robusta.

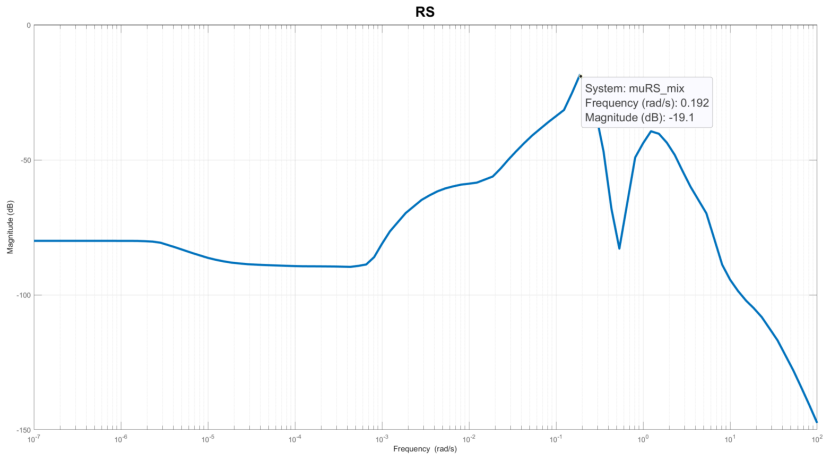

Figura 10: RS primo test per Mix sensitivity

- Ultimo passo è stato calcolare il valore singolare strutturato massimo per la robusta prestazione. Con il solito procedimento usato per la RS ma con il blocco δ_P aggiunto al sistema abbiamo trovato: In questo caso il valore è appena sopra 1, quindi siamo già al limite delle prestazioni.

- A questo punto siamo andati a verificare che i tempi di assestamento fossero simili a quelli dell'LQG date condizioni iniziali identiche. Come possiamo notare, anche se la dinamica è diversa il tempo di assestamento è simile.

A questo punto abbiamo provato a alzare la banda del sistema per tentare di aumentare le prestazioni, i risultati sono visibili in figura 14. Nello specifico, abbiamo aumentato w_b a 0.5, in questo modo siamo riusciti a diminuire il tempo di assestamento.

Infine abbiamo eseguito un esperimento significativo per valutare l'impatto della banda delle funzioni peso sulle prestazioni del sistema. Questo test ha come scopo la verifica sperimentale di ciò che ci aspettiamo dalla teoria. I risultati, in accordo con le nostre aspettative, sono visibili in figura 15.

Abbiamo ripetuto il primo ed il terzo test per il sistema non lineare,

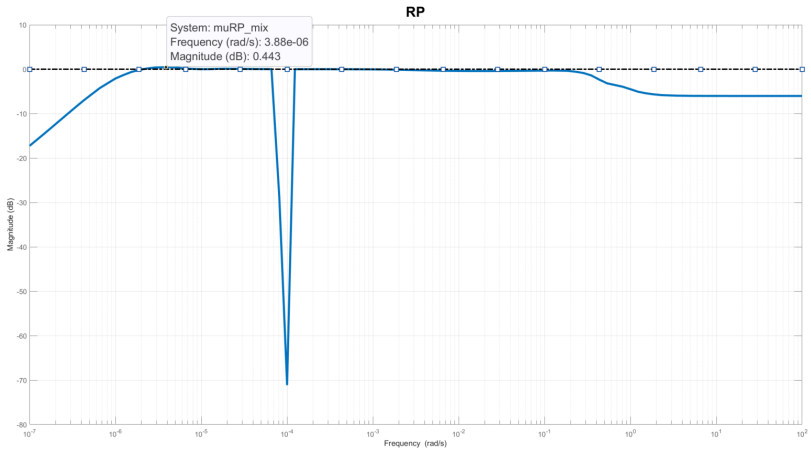

Figura 11: RP primo test per Mix sensitivity

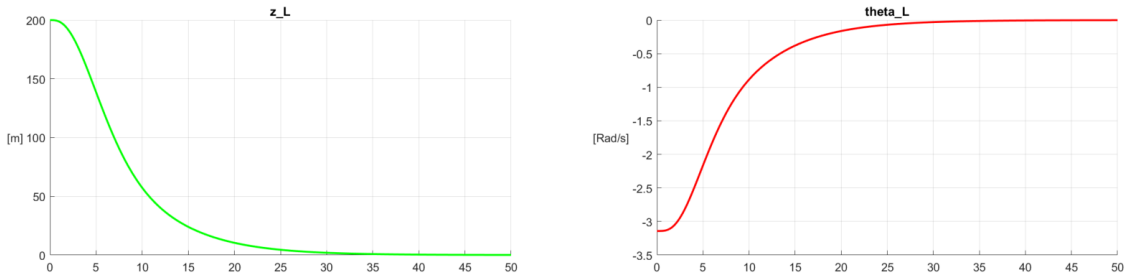

Figura 12: tempo assestamento primo test per Mix Sensitivity

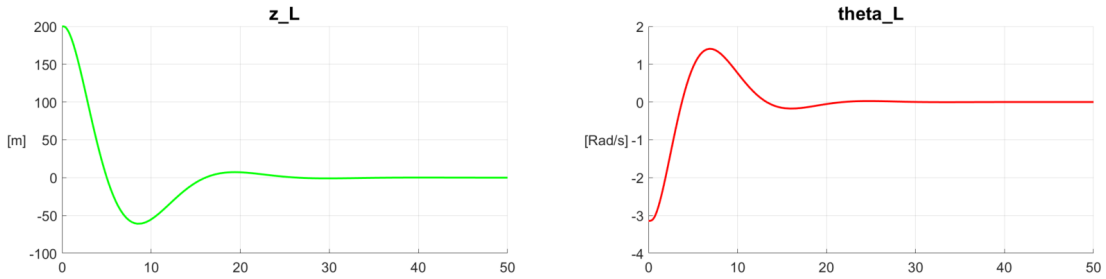

Figura 13: tempo assestamento primo test per mix sensitivity

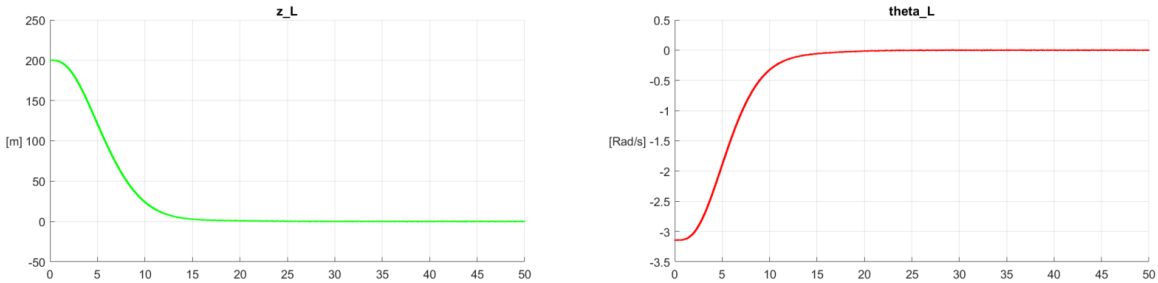

Figura 14: tempo assestamento con aumento banda per mix sensitivity

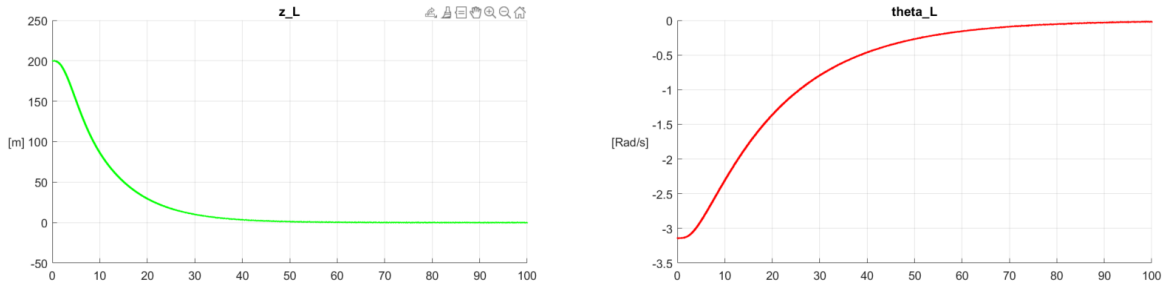

Figura 15: tempo assestamento con diminuzione banda per mix sensitivity

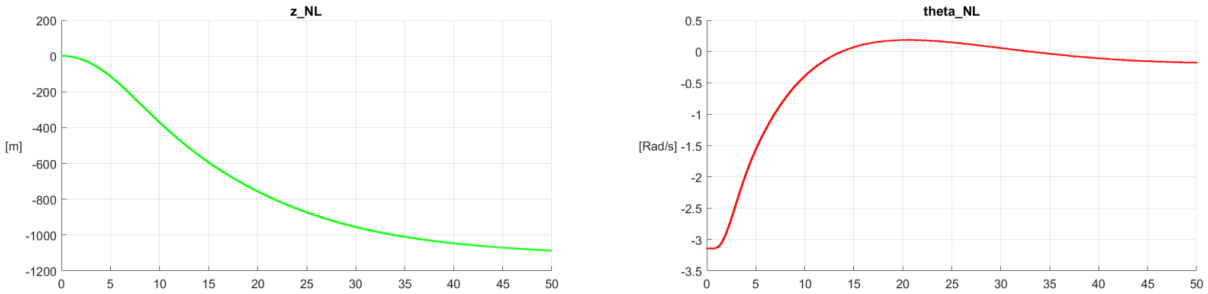

Figura 16: tempo assestamento con diminuzione banda per mix sensitivity

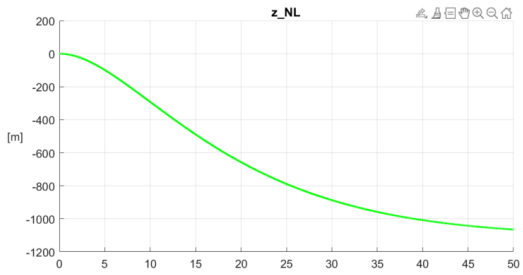

Figura 17: tempo assestamento con diminuzione banda per mix sensitivity

5 Controllore con μ synthesis

Nel capitolo precedente, abbiamo esplorato la sintesi del controllore utilizzando la tecnica di Mix-Sensitivity, una metodologia consolidata che ha dimostrato la sua efficacia in vari contesti di controllo. Ora, ci concentriamo su un approccio altrettanto rilevante e potente: la DK iteration. Quest'approccio, una delle tecniche di controllo lineare, si distingue per la sua capacità di gestire le incertezze del sistema in modo dinamico, al contrario del Mix-Sensitivity che si basa sul modello nominale del sistema.

Vediamo ora i procedimenti che abbiamo eseguito per la sintesi del controllore:

1. **Definizione dei Parametri di Incertezza:** Inizialmente, abbiamo definito i parametri di incertezza del sistema utilizzando il comando `ureal`, stabilendo le variazioni possibili nei parametri del sistema.

2. **Determinazione delle Matrici di Trasferimento in Configurazione G-Δ:** Con il comando `lftdata`, abbiamo ottenuto le matrici di trasferimento del sistema nella configurazione G-Δ, considerando le incertezze definite precedentemente.

3. **Specificazione dei Pesi di Prestazione:** Abbiamo definito i pesi di prestazione, mantenendo la coerenza con quelli utilizzati nella tecnica di mix-sensitivity.

4. **Ricavo delle Incertezze degli Attuatori:** Per determinare le incertezze degli attuatori, abbiamo calcolato l'involuzione dell'errore relativo tra la funzione di trasferimento perturbata e quella nominale.

5. **Creazione del Sistema con il Comando Connect:** Utilizzando il comando `connect` (dello schema in figura 18), abbiamo costruito il sistema integrando le matrici di trasferimento G-Δ, i pesi di prestazione e le incertezze degli attuatori.

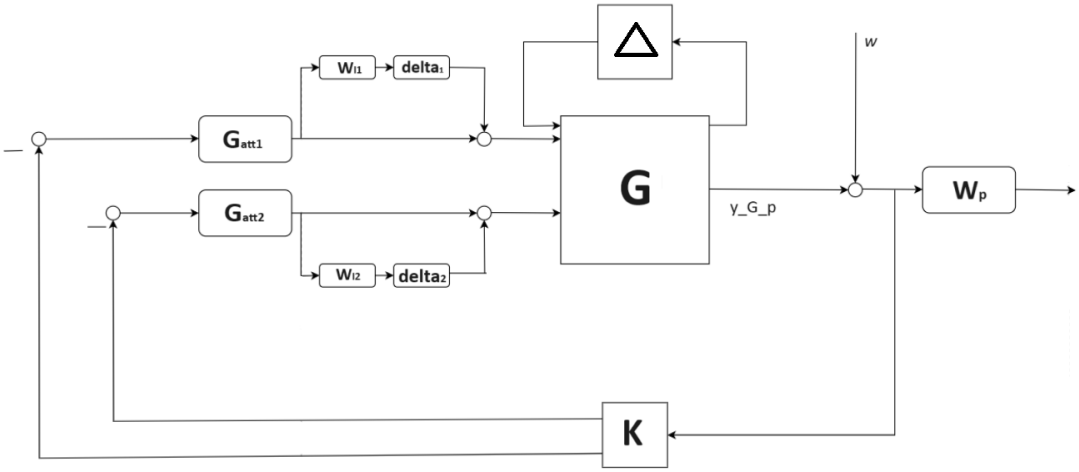

Figura 18: schema di controllo per controllore DK iteration

6. **Definizione della Matrice di Incertezza Δ Totale:** Abbiamo definito la matrice di incertezza totale, considerando tutte le fonti di incertezza precedentemente identificate.

7. **Sintesi del Controllore con il Comando Musyn:** Infine, abbiamo utilizzato il comando `musyn` per trovare il controllore ottimale, ottimizzando le performance del sistema considerando sia la dinamica del sistema che le incertezze.

Questi passaggi sono stati implementati tramite il codice sottostante.

Listing 4: Codice implementazione DK iteration

```
%% mu-synthesis
 clear
 close all
 clc

%% paramtri sistema
% Definizione parametri convertiplano %
J = 5000;
m = 2000;
b = ureal('b',150,'Percentage',[-5,5]);
beta=ureal('beta',15,'Percentage',[-5,5]);
l = 10;

g = 9.81;

%params_plano = [J, m, b, beta, l, g];

% Definizione parametri attuatori %
Km1 = 500;
Km2 = 100;
T1 = 0.5;
%params_attuatori.T1=ureal('T1',0.1,'Percentage',[-5,5]);
T2 = 0.5;
%params_attuatori.T2=ureal('T2',0.2,'Percentage',[-5,5]);
```

```
Tm1 = 5;
Tm2 = 5;

%params_attuatori = [Km1, Km2, T1, T2, Tm1, Tm2];

%% Definisco il sistema linearizzato nel PE
% PE = (0,0,0,0)  %(x,x_d,theta,theta_d)

A = [0 1 0 0; 0 -b/m 0 0; 0 0 0 1; 0 0 0 -beta/J];

B = [0 0; 1 0; 0 0; 0 2*l];

C = [1 0 0 0; 0 0 1 0];

D = zeros(2);

% trovo fdt nominale del linearizzato

sys_n = ss(A,B,C,D);

[P,Delta_G,Blkstruct]=lftdata(sys_n);

G_P = zpk(P);


%% Definisco le fdt degli attuatori nominali (senza tempo di ritardo) e
    perturbate
s = tf('s');

Gm1_n = Km1/(Tm1*s+1);
Gm1_p = Km1/(Tm1*s+1)*exp(-T1*s);
Gm2_n = Km2/(Tm2*s+1);
Gm2_p = Km2/(Tm2*s+1)*exp(-T2*s);

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

% ricavo i pesi Wi degli attuatori
Wi1 = makeweight(10^-4,1,2.5);
Wi2 = makeweight(10^-4,1,2.5);
%Wi1 = 0.003*5*10^4*(s+0.03)/(s+15);
%Wi2 = 0.012*10^4*(s+0.03)/(s+15)^2;

figure(1)
hold on
grid on
bodemag(reldiff1,Wi1)
hold off

figure(2)
hold on
grid on
bodemag(reldiff2,Wi2)
hold off

% creo la upper lft degli attuatori

P_att = [0 0 Wi1*Gm1_n 0; 0 0 0 Wi2*Gm2_n;1 0 Gm1_n 0; 0 1 0 Gm2_n];

Delta_att1 = ultidyn('Delta_att1',[1 1]);
Delta_att2 = ultidyn('Delta_att2',[1 1]);
Delta_att = [Delta_att1 0; 0 Delta_att2];

%% definisco il peso Wp di prestazione
A1=1e-4;
M1=2;
wB1=0.3;
A2=1e-4;
M2=2;
wB2=0.5;
```

```
wP1=makeweight(1/A1,wB1,1/M2);
wP2=makeweight(1/A2,wB2,1/M2);
WP=blkdiag(wP1,wP2); %matrice peso s

%% creo il sistema con connect

G_P.u = {'uD1','uD2','u_G_P1','u_G_P2'};
G_P.y = {'yD1','yD2','y_G_P1','y_G_P2'};

P_att.u = {'ud1','ud2','u_P_att1','u_P_att2'};
P_att.y = {'yd1','yd2','y_P_att1','y_P_att2'};

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

Sum1 = sumblk ('u_P_att1 = -u1');
Sum2 = sumblk ('u_P_att2 = -u2');
Sum3 = sumblk ('e1 = w1  + y_G_P1');
Sum4 = sumblk ('e2 = w2  + y_G_P2');
Sum5 = sumblk ('u_G_P1 = y_P_att1');
Sum6 = sumblk ('u_G_P2 = y_P_att2');


P = connect (G_P,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_dist = lft(Delta,P);

[K_DK,CLperf,info] = musyn(P_dist,2,2);

K_DK = minreal(zpk(tf(K_DK)),0.5);

K_DK = [K_DK(1,1) 0; 0 K_DK(2,2)];

%bodemag(K_DK)
```

Anche in questo caso l'ultimo passaggio è stato quello di generare su Simulink dei modelli di verifica.

5.1 μ analisi

Come per il mix sensitivity abbiamo usato il solito sistema generato con connect cambiando solamente il controllore all'interno del sistema.

```
    %% Mu-Synthesis

N = lft(P, K_DK);
N_zpk = zpk(N);

%NS
N22_DK = N_zpk(5:6, 5:6);

%RS e RP
N11 = N(1:4, 1:4);

mubnds_mu = mussv(N, blk_RP);
muRP_mu = mubnds_mu(:,1);
[muRPinf_mu, MuRPw_mu] = norm(muRP_mu,inf);

mubnds_mu = mussv(N11, blk_RS);
muRS_mu = mubnds_mu(:,1);
[muRSinf_mu, MuRSw_mu] = norm(muRS_mu,inf);

RP_mu = lft(Delta_RP,N);
[stabmargRP_mu, wcuRP_mu] = robstab(RP_mu);

```

```
RS_mu = lft(Delta,N11);
[stabmargRS_mu, wcuRS_mu] = robstab(RS_mu);
```

5.2 Risultati

In questo caso, invece di partire dai risultati ottenuti con LQG, siamo partiti dalla funzione Wp del Mix sensitivity. Sulla base di questa abbiamo iniziato la nostra analisi, facendo le stesse prove del Mix sensitivity. Una volta verificata la nominale stabilità, abbiamo trovato $\mu_{max} = 0.89$.

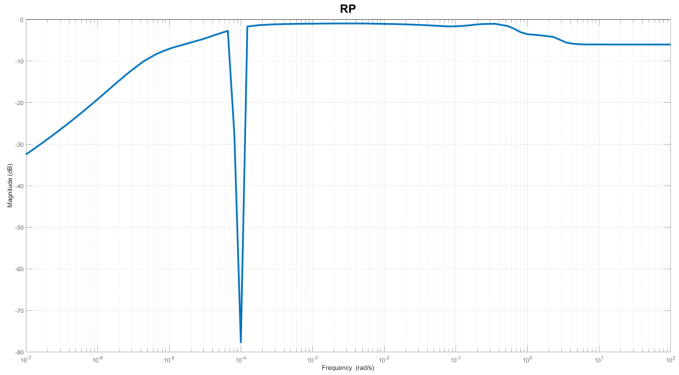

Figura 19: grafico RP del controllore DK

In questo caso essendo inferiore a 1, abbiamo aumentato la banda aspettandoci migliori prestazioni.

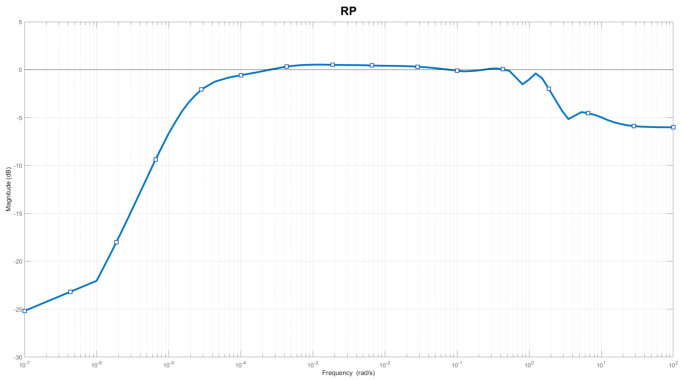

Figura 20: grafico RP del controllore DK con banda aumentata

Aumentando la banda sia di z che di θ da 0.15 a 0.25, arriviamo ad avere $\mu_{max} = 0.1.06$, quindi appena sopra 1. Garantiamo così la robusta prestazione e aumentando le performance. Precisamente il tempi di assestamento del sistema passano da $ta_\theta = 9$ e $ta_z = 13.3$ a $ta_\theta = 8.6$ e $ta_z = 11.5$.

Poi abbiamo testato l'efficacia del controllore sul sistema non lineare, ma come per il Mix-Sensitivity, z è instabile e non converge mai a 0, al contrario di θ che converge sempre.

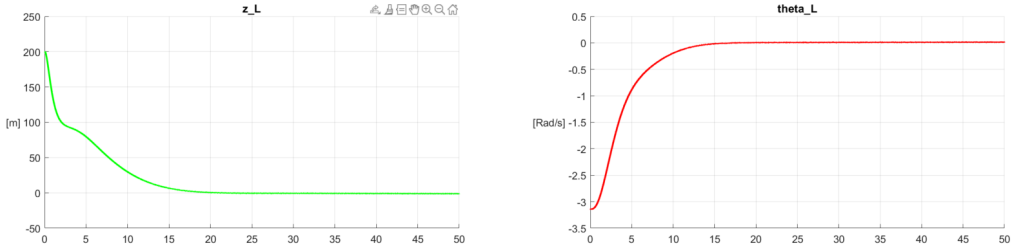

Figura 21: tempo di assestamento al limite della RP per DK

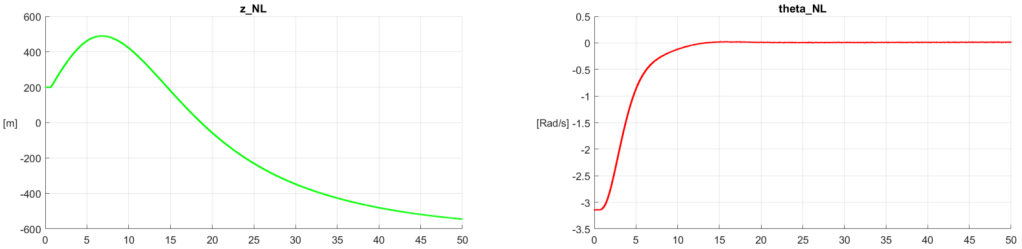

Figura 22: Applicazione su sistema non lineare del DK

6 Conclusioni

Concludendo, possiamo cosi riassumere l'insieme dei risultati ottenuti:

- L'obbiettivo di controllo 'e stato generalmente soddisfatto con tutti i controllori proposti sul sistema linearizzato.
- Nel confronto tra Mix-sensitivity e Mu-Synthesi notiamo che con solita larghezza di banda di partenza le prestazioni robuste del secondo sono nettamente superiori. I valori strutturati sono rispettivamente $\mu_{ms} = 1.04$ e $\mu_{dk} = 0.71$ che fanno sì che i tempi di assestamento del DK siano più piccoli. Da considerare però un notevole aumento di poli nel sistema e quindi di difficoltà di implementazione.
- Lo studio delle incertezze ha rivelato che si ottiene robusta stabilità con buoni margini per entrambi i controllori DK e Mixed Sensitivity, tuttavia quando si includono le prestazioni la situazione peggiora, nel senso che il margine sulla RP è piccolo ed aumentando leggermente le incertezze rispetto a quelle proposte non si soddisfa il requisito di performance. La simulazione con raggio incerto, fatta sul controllore LQG nel test 3, rivela che, seppur questo non garantisca a priori alcun margine di robustezza, mantiene la stabilità.
- Per quanto riguarda la complessità in termini di spazio di stato dei controllori il DK si rivela, come prevedibile, di gran lunga il pi'u complesso; seguito dal MS, infine l'LQG. Nella maggioranza della applicazioni pratiche si preferisce utilizzare controllori con spazio di stato ridotto (minore complessità), quindi, in questi termini, il DK non è preferibile. Sarebbe possibile impostare la sintesi del DK in modo che si ottimizzi un controllore a struttura fissa e sintonizzabile mediante un numero limitato di parametri (Ad esempio un PID), risultando cosi di ordine ridotto.