UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Magistrale in Ingegneria Robotica e Automazione

Progetto C: Autonomous Guided Vehicle

Docente:

Prof. Paolo Salaris

Studentesse:

Alexia Sprecacè
Francesca Nocentini
Giorgia Pazzi

ANNO ACCADEMICO 2022/2023

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Descrizione del modello AGV

Il veicolo industriale a guida autonoma preso in esame presenta due ruote folli anteriori e una motoruota posteriore. Quest'ultima è attuata per mezzo della coppia di rotolamento τ_ϕ, che ne permette i movimenti di traslazione, ed attraverso la coppia di sterzo τ_ψ, che ne permette la manovrabilità agendo sull'angolo di sterzo ψ della motoruota posteriore.

Lo schema dell'AGV e le dimensioni in gioco sono mostrati nella figura 1.1.

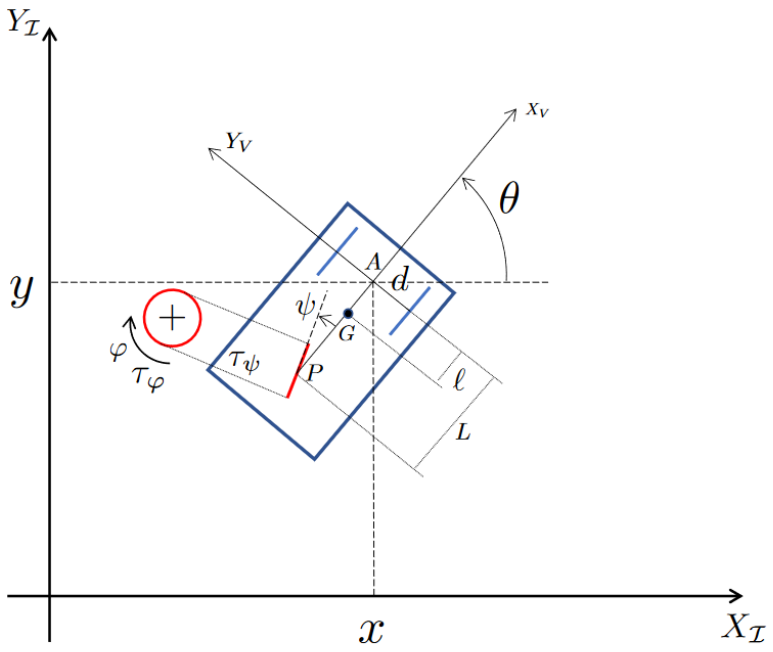


Figura 1.1: Rappresentazione del veicolo industriale a guida autonoma.

La posizione del veicolo è rappresentata dalle coordinate x e y espresse rispetto a

un sistema di riferimento fisso, mentre con θ si indica l'orientazione dell'AGV. Altre grandezze di interesse sono rappresentate dalle variabili ϕ e ψ, le quali indicano rispettivamente la rotazione della motoruota posteriore e il suo angolo di sterzo.

Il centro di massa è identificato nel punto G in figura, e il momento d'inerzia dell'intero veicolo (I_{GZ}) è calcolato rispetto a tale punto. Gli altri momenti di inerzia di interesse sono quelli delle ruote anteriori e della ruota posteriore, tutti espressi rispetto agli assi y e z, rispettivamente I_{AY}, I_{AZ}, I_{PY} e I_{PZ}.

Le equazioni che regolano la cinematica e la dinamica dell'AGV sono rispettivamente in 1.1 e 1.2:

$$\begin{cases} \dot{x} = rp\dot{\phi}cos\theta cos\psi \\ \dot{y} = rp\dot{\phi}sin\theta cos\psi \\ \dot{\theta} = \frac{-rp\dot{\phi}sin\psi}{L} \end{cases} \tag{1.1}$$

$$\begin{cases} \tau_\phi = (arp^2cos\psi^2 + b(\frac{rp}{L})^2sin\psi^2 + IPy)\ddot{\phi} - IPz\frac{rp}{L}\ddot{\psi}sin\psi + (b(\frac{rp}{L})^2 - arp^2)\dot{\phi}\dot{\psi}sin\psi cos\psi \\ \tau_\psi = -IPz\frac{rp}{L}\ddot{\phi}sin\psi + IPz\ddot{\psi} - IPz\frac{rp}{L}\dot{\phi}\dot{\psi}cos\psi \end{cases} \tag{1.2}$$

dove

$$a = (M_V + m_P + 2m_A + 2\frac{I_{AY}}{r_A^2})$$

$$b = \frac{1}{2}(M_V l^2 + m_P L^2 + 2m_A d^2 + I_{GZ} + I_{PZ} + 2I_{AZ} + 2I_{AY}\frac{d^2}{r_A^2})$$

I valori che sono stati utilizzati per le diverse variabili sono indicati nella tabella 1.1. Alcuni di questi presentano un'incertezza percentuale.

1.1.1 Obiettivi di controllo

Gli obiettivi da raggiungere nel seguente lavoro sono stati dunque:

1. Implementazione su Matlab/Simulink (sia sul modello linearizzato che su quello nonlineare) di un **controllo LQG** (con e senza integratore).

2. Progetto ed implementazione su Matlab/Simulink (sia sul modello linearizzato che su quello nonlineare) del controllore **mix-sensitivity** e $\mathbf{H}_\infty$ e studio della robustezza attraverso la **mu-analysis**;

3. Progetto ed implementazione su Matlab/Simulink (sia sul modello linearizzato che su quello nonlineare) di un controllore robusto attraverso la **DK-iteration** con conseguente analisi di robustezza (mu-analysis).

Parametro	Valore Nominale	Incertezza
l	0.4 m	
L	1 m	
rp	0.3 m	±5%
ra	0.3 m	±5%
Mv	15 kg	
m_p	0.5 kg	
m_a	0.5 kg	
I_{GZ}	0.655 Nm	
I_{PZ}	0.23 Nm	
I_{PY}	0.15 Nm	
I_{AZ}	0.8 Nm	
I_{AY}	0.12 Nm	
$\bar{K}_{m1}$	1.08	±10%
$\bar{T}_{m1}$	0.005 s	±20%
$\bar{K}_{m2}$	0.335	±10%
$\bar{T}_{m2}$	0.002 s	±20%
$\bar{T}_1$	$\frac{\bar{T}_{m1}}{2}$	±5%

1.1.2 Definizione dei requisiti di prestazione desiderati

In tutti i controllori implementati la linearizzazione del sistema non lineare è stata realizzata intorno alla retta $\boldsymbol{y = 0}$. Dunque tra gli obiettivi di controllo vi è quello di mantenersi nell'intorno di questa retta.
Sono state fatte delle semplificazioni allo schema cinematico per soddisfare gli obiettivi di performance.
In particolare nel vettore di stato del sistema non è stato considerato l'andamento di x. Infatti, considerando che l'obiettivo di controllo era quello di imporre una traiettoria rettilinea lungo l'asse x, non era di interesse regolare tale variabile visto che poteva assumere qualunque valore. Di conseguenza non è stata considerata la prima equazione della cinematica (1.1).
Quindi il vettore di stato considerato è

$$x = [y, \theta, \dot{\phi}, \psi, \dot{\psi}]^T = [X(1), X(2), X(3), X(4), X(5)]^T$$

I sensori che si hanno a disposizione sono quelli che ci forniscono le uscite di prestazione: $[y, \theta, \dot{\phi}, \psi]^T$.

Siccome occorreva linearizzare attorno ad y=0, il punto di equilibrio scelto è stato $x_{eq} = [0, 0, -1, 0, 0]^T$. Il valore di equilibrio per *phi_dot* è stato scelto negativo in quanto a seguito di scostamenti anche piccoli dell'angolo di sterzo la ruota posteriore tende a far ruotare il carrello che si muove quindi a marcia indietro a regime.

Capitolo 2

Controllo LQG

Nel controllo LQG (*Controllo Lineare Quadratico Gaussiano*) tradizionale, si assume che la dinamica dell'impianto sia lineare e nota, e che il rumore di misura e i segnali di disturbo (rumore di processo) siano processi stocastici con proprietà statistiche note.

Dato il sistema lineare

$$\dot{x} = Ax + Bu + w_d$$

$$y = Cx + Du + w_n$$

il problema di controllo ottimo LQG si pone l'obiettivo di determinare l'azione di controllo ottima $u(t)$ che minimizza

$$J = E\left\{\lim_{T\to\infty}\frac{1}{T}\int_0^T [x^T Q x + u^T R u]dt\right\}$$

dove Q ed R sono matrici di peso tali che $Q = Q^T \geq 0$ ed $R = R^T > 0$.

Per la progettazione del regolatore LQG si effettuano i seguenti passaggi:

- Costruzione del guadagno LQR ottimale.
- Costruzione di un filtro di Kalman (stimatore di stato).
- Formazione della progettazione LQG attraverso la connessione del guadagno LQR ottimale e del filtro di Kalman.

Come visto nel capitolo precedente, il modello preso in esame presenta equazioni non lineari, per cui il primo passo per la progettazione del controllore LQG è quello di linearizzare il sistema. Questo passaggio è svolto nello script *get_linearization*. All'interno di tale script sono definite la funzione di stato f e il modello di osservazione h attraverso il formato *symbolic* di MATLAB. Sfruttando la funzione *jacobian()* di MATLAB si ricavano le matrici A, B, C e D del sistema linearizzato. La linearizzazione viene calcolata rispetto alla traiettoria descritta dall'equazione $y = 0$. Per tale motivo vengono sostituiti, tramite la funzione *subs()*, i valori numerici dello stato di equilibrio e degli ingressi (all'equilibrio) descritto al capitolo precedente.

2.1 LQG senza integratore

Il requisito principale per la realizzazione del controllore è stato quello di garantire la raggiungibilità o almeno la stabilizzabilità del sistema (vedi codice in appendice 7). Gli autovalori della matrice del linearizzato A sono:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -0.3 \end{bmatrix}$$

Calcolando la matrice di raggiungibilità del sistema con il comando **ctrb** è stato calcolato il rango ed è stato visto che questo è pieno, il che soddisfa la condizione di raggiungibilità e dunque quella di stabilizzabilità, anche se il sistema risulta già essere stabile di per se attorno a tale punto di equilibrio.
Il primo tipo di controllore LQG realizzato ha necessitato della definizione delle seguenti matrici:

1. Q che è la matrice di peso sullo stato x, che è stata presa come matrice identità di dimensione pari al vettore di stato;

2. R è la matrice di peso sull'ingresso, presa come identità di dimensione pari al vettore di ingresso;

3. B_{noise} è la matrice di disturbo sulle componenti dello stato. Dato che supponiamo di avere rumore bianco solo sugli ingressi le componenti influenzate sono solo quelle ottenute tramite l'inversione della dinamica e dunque $\dot{\phi}$ e $\dot{\psi}$.

$$B\_noise = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

4. la matrice W del rumore di processo è stata scelta come segue

$$W = \begin{bmatrix} std\_dev\_tau\_phi^2 & 0 \\ 0 & std\_dev\_tau\_psi^2 \end{bmatrix} = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.01^2 \end{bmatrix}$$

5. la matrice V del rumore di misura è stata scelta come segue

$$V = \begin{bmatrix} std\_dev\_dy^2 & 0 & 0 & 0 \\ 0 & std\_dev\_theta^2 & 0 & 0 \\ 0 & 0 & std\_dev\_phi\_dot^2 & 0 \\ 0 & 0 & 0 & std\_dev\_psi^2 \end{bmatrix}$$

scegliendo

$$\begin{bmatrix} std\_dev\_dy \\ std\_dev\_theta \\ std\_dev\_phi\_dot \\ std\_dev\_psi \end{bmatrix} = \begin{bmatrix} 0.05m \\ 0.05rad \\ 0.017rad/s \\ 0.08rad \end{bmatrix}$$

Dopo aver calcolato il guadagno **Kr** del controllore LQR è stato implementato un **filtro di Kalman lineare** che produce il guadagno **Ke**.

2.2 LQG con integratore

Dato che il primo controllore che è stato realizzato non prevede azione integrale ne è stata tentata la realizzazione di un successivo come nello schema nella figura 2.1 per annullare l'errore a regime.
Ciò ha comportato il dover aumentare l'impianto, ovvero ingrandire le matrici della dinamica come

$$A\_aug = \begin{bmatrix} A & 0_{nxp} \\ -C & 0_{pxp} \end{bmatrix}$$

$$B\_aug = \begin{bmatrix} B \\ -D \end{bmatrix}$$

$$C\_aug = \begin{bmatrix} C & 0_{pxp} \end{bmatrix}$$

$$Q\_aug = \begin{bmatrix} 0_{nxn} & 0_{nxp} \\ 0_{pxn} & I_{pxp} \end{bmatrix}$$

dove p ed n sono rispettivamente il numero di output al sistema e il numero degli stati del sistema. Si considera questa struttura perchè si presuppone che il riferimento sia nullo.
Con queste nuove matrici il rango della nuova matrice di raggiungibilità e di osservabilità non sono pieni e pari a 9, ma 7 e 5. Tramite un controllo fatto con **ctrbf** e **obsvf** che portano il sistema in forma standard di raggiungibilità ed osservabilità si è notato che gli autovalori non raggiungibili/osservabili erano proprio autovalori instabili. Dunque non è stato possibile in questo caso realizzare un controllore con integratore linearizzando attorno alla traiettoria di equilibrio y=0, sia per l'impossibilità di trovare un controllore LQR, sia per la successiva realizzazione del Kalman Filter.

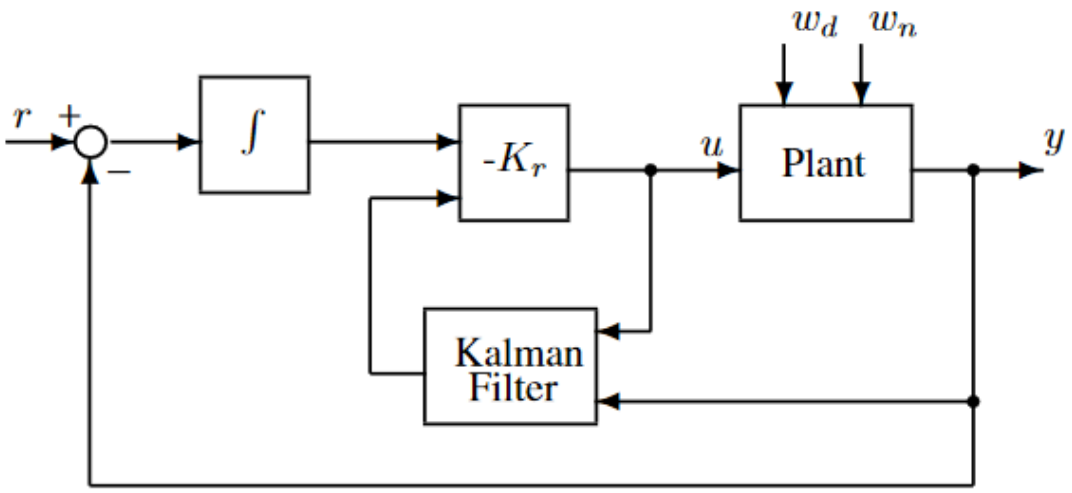


Figura 2.1: Schema del modello linearizzato controllato con il controllore LQG con integratore

2.3 Validazione in Simulink del modello lineare senza integratore

In Simulink è stato implementato lo schema dell'LQG del sistema sul modello lineare come in figura 2.3. Sono stati aggiunti i rumori di misura e quelli di processo ed è stato analizzato come il sistema reagisse ai disturbi in uscita. In particolare è stato applicato un disturbo alla prima componente del segnale di uscita. Il sistema è in grado di reiettare i disturbi a gradino in quanto nella matrice di trasferimento è già presente un intregratore (polo nell'origine). E' stato osservato che se si applica un disturbo di ordine superiore il sistema non è in grado di reiettarlo.

Si nota come avendo aggiunto il controllore il sistema in evoluzione libera converga a 0 in uscita a meno dei rumori di processo per condizioni iniziali anche lontane dal punto di equilibrio.

2.4 LQG senza integratore e con attuatori

Il controllore LQG senza integratore è stato realizzato anche sul modello che predispone di attuatori con le seguenti funzioni di trasferimento:

$$G_{\tau_\psi} = \frac{\bar{K}_{m1} e^{\bar{T}1 s}}{\bar{T}_{m1} s + 1} \cong \bar{K}_{m1} \frac{(1 - \frac{\bar{T}1}{2} s)}{(\bar{T}_{m1} s + 1)(1 + \frac{\bar{T}1}{2} s)}$$

$$G_{\tau_\phi} = \frac{\bar{K}_{m2}}{(\bar{T}_{m2} s + 1)(\bar{T}_{m2}/100 s + 1)}$$

La prima funzione di trasferimento presenta un ritardo che è stato approssimato con Padé del primo ordine in quanto il comando `ureal` non genera ritardi incerti di

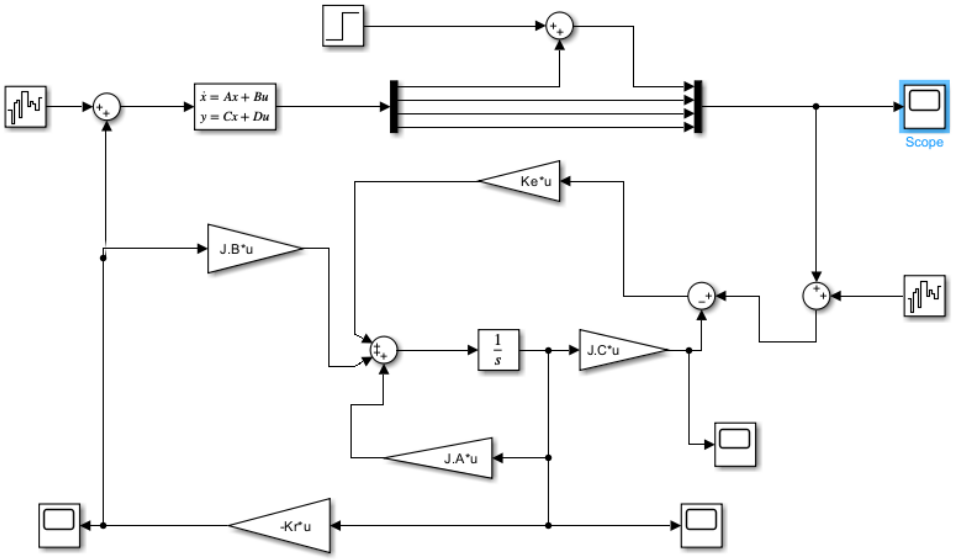


Figura 2.2: Schema del modello linearizzato controllato con il controllore LQG

tipo esponenziale. Dunque la nuova funzione di anello che è stata realizzata per fare l'LQG è quella che mette in cascata G e G_{act} dove questa ultima è la funzione di trasferimento degli attuatori ottenuta come matrice diagonale a blocchi dove sulla diagonale ci sono le due funzioni di trasferimento degli attuatori.

2.5 Validazione in Simulink del modello lineare con attuatori

Anche in questo caso è stata fatta una validazione del modello su Simulink. Lo schema realizzato è stato fatto come in figura 2.4. I risultati ottenuti mostrano come le funzioni di trasferimento degli attuatori vadano a modificare leggermente l'effetto stabilizzante dell'LQG (vedi figura 2.5 nella quale si vede l'andamento dell'uscita sensore per sensore). Si nota come tutte le componenti convergano a 0 a meno del rumore mentre la phi dot impiega tanto tempo per convergere a 0.

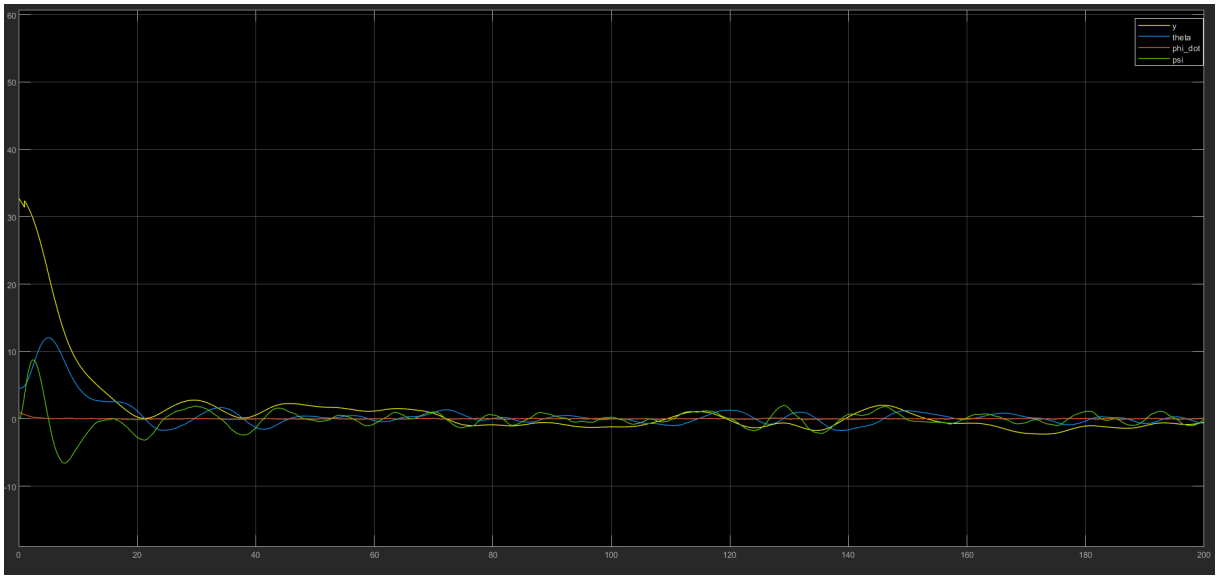


Figura 2.3: Risultati delle uscite al modello linearizzato controllato con il controllore LQG con condizioni iniziali [36.05;3.28;5.44;0.84;0.05]

2.6 Controllo LQG sul modello non lineare

Il controllore LQG è stato provato anche sul non lineare (schema della figura 2.6). Affinchè il controllore desse risultati corretti anche sul non lineare è stato implementato uno stimatore EKF che utilizza la funzione di transizione dello stato 7 e la funzione che descrive il modello di misura.

2.7 Analisi dei risultati del modello non lineare senza attuatori

I risultati mostrano come il controllore faccia convergere a 0 (vedi figura 2.7) tutte le componenti delle uscite se si è vicini al punto di equilibrio del sistema eccetto la $\dot{\phi}$ che converge al valore di regime, ovvero velocità costante a -1rad/s. Importante è stato traslare le coordinate del sistema non lineare attorno al punto di equilibrio prima del controllore Kr.

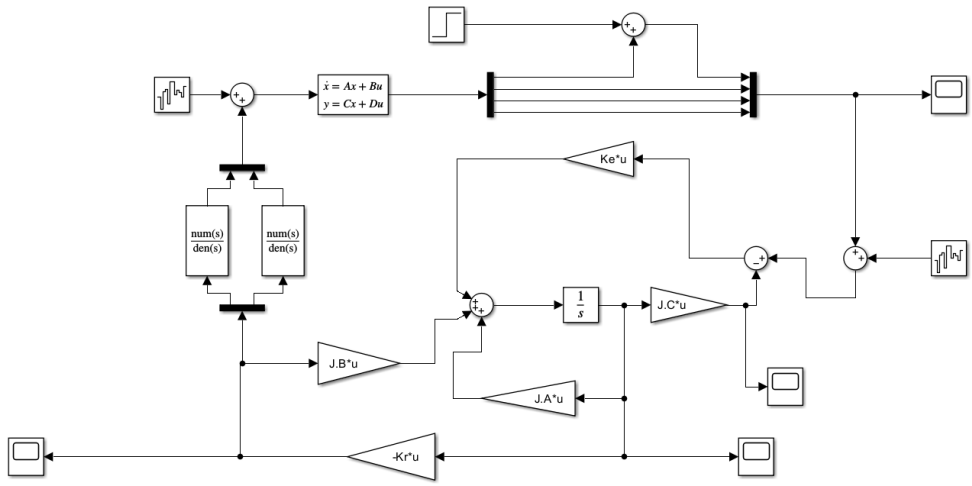


Figura 2.4: Schema del modello linearizzato controllato con il controllore LQG con aggiunta la dinamica degli attuatori

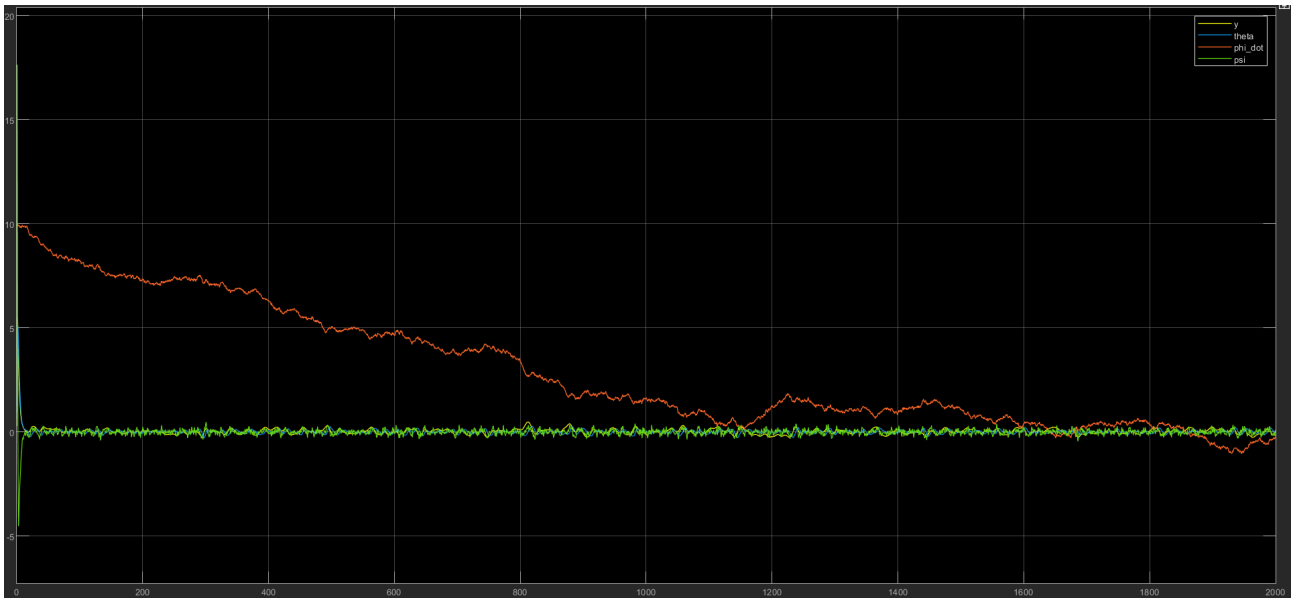


Figura 2.5: Risultati della simulazione con il sistema linearizzato con gli attuatori con condizioni iniziali [5.48;0.39;9.93;0.28;0.01]

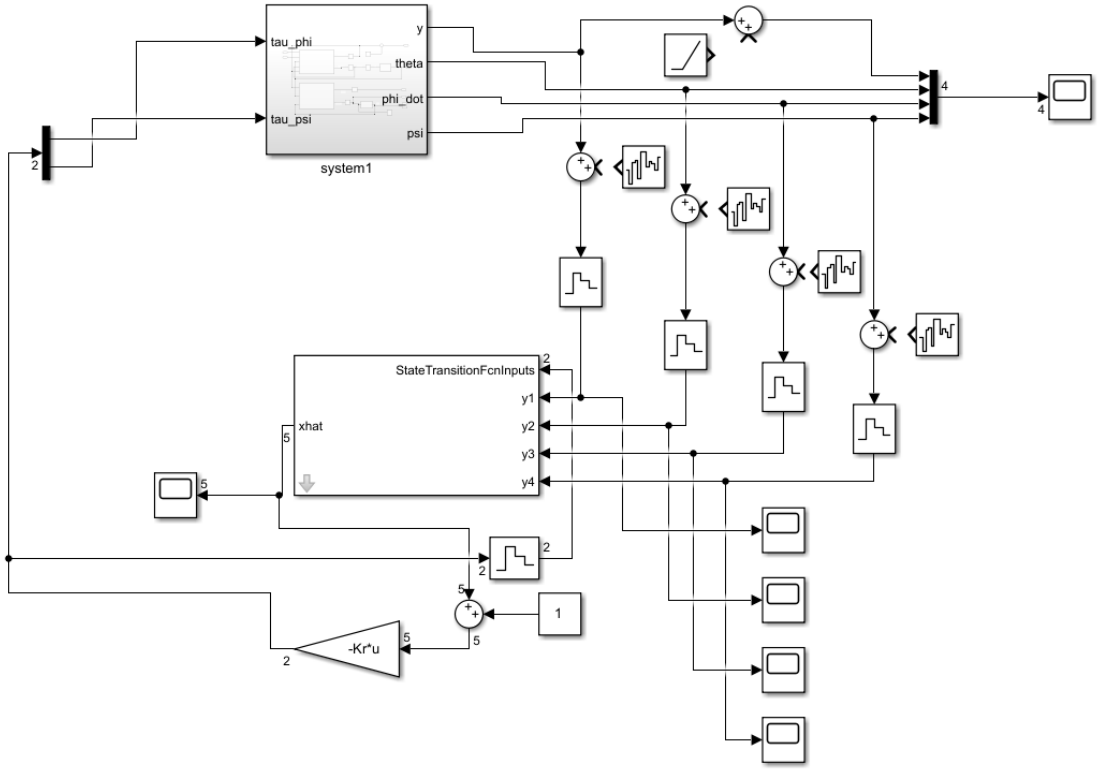


Figura 2.6: Schema del sistema non lineare con LQG

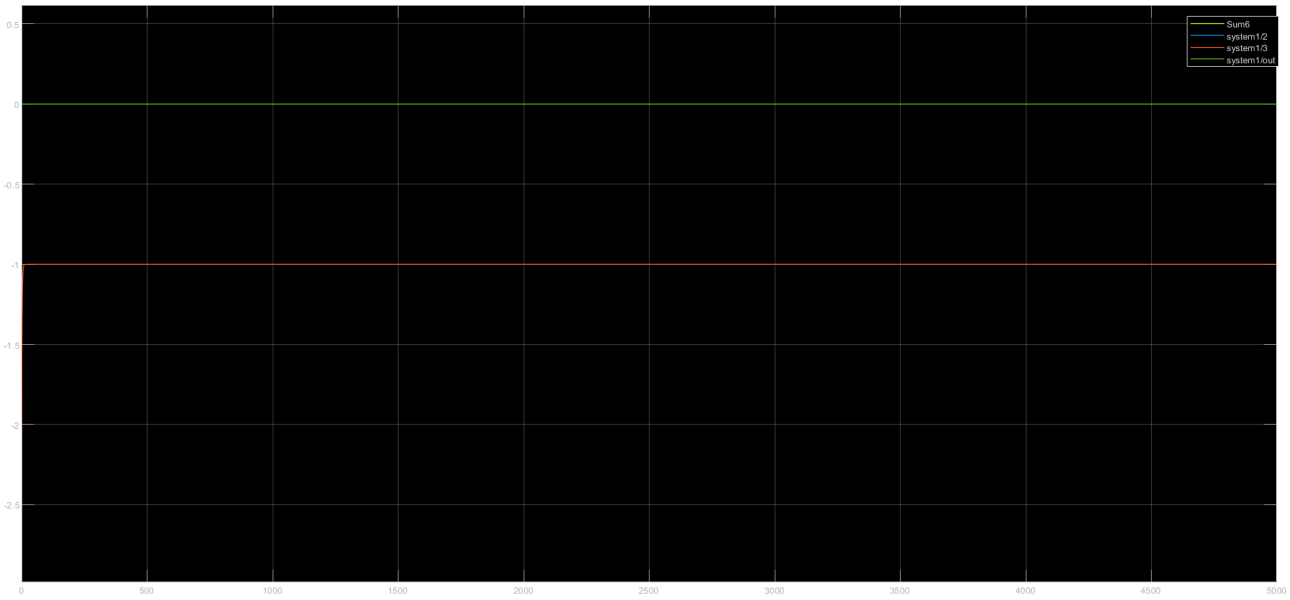
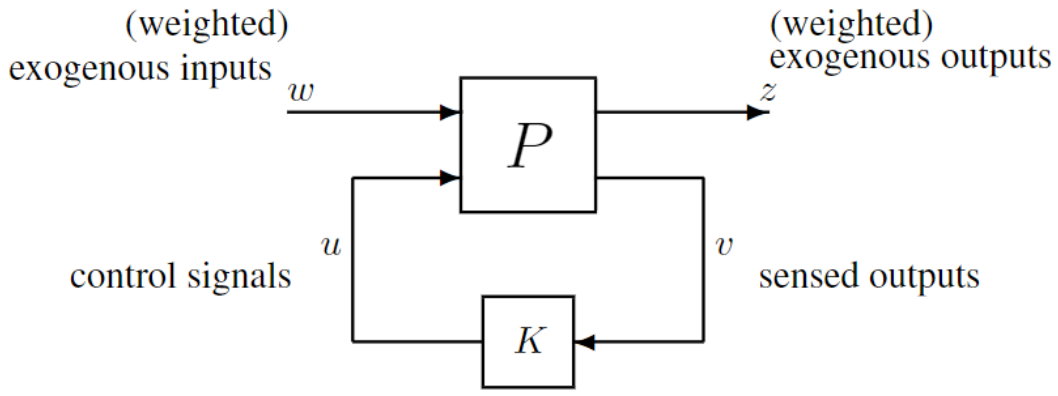


Figura 2.7: Risultati della simulazione per il sistema non lineare con condizioni iniziali diverse dal punto di equilibrio ovvero [0.001; 0.001; -1.001; 0.001]

Capitolo 3

Controllo $\mathcal{H}_\infty$

Un controllore $\mathcal{H}_\infty$ permette di ottenere stabilizzazione rispettando determinate specifiche. Si imposta un problema di ottimizzazione matematica e si progetta un controllore che risolva tale ottimizzazione. Questa tecnica presenta il vantaggio di poter essere applicata a sistemi MIMO mentre uno svantaggio potrebbe essere la richiesta di una buona conoscenza del modello.
La routine in MATLAB per la sintesi di un controllore $\mathcal{H}_\infty$ prevede che il sistema venga portato nella seguente forma, dove w sono gli input esterni, z rappresentano quelli che sono gli obiettivi del controllo, v sono le uscite misurate e u il segnale di controllo.



3.1 Obiettivi del controllo

Prima di definire quelli che saranno i nostri obiettivi di controllo, è utile introdurre il concetto di valore singolare massimo.
Nei sistemi SISO, dato un certo ingresso d(ω) e un'uscita y(ω), il guadagno ad una certa frequenza è indipendente dal modulo di d(ω), mentre nei MIMO il guadagno dipende dalla direzione dell'input d.

Il valore singolare massimo rappresenta il guadagno massimo al variare della direzione dell'input.
Si considera dunque un problema $\mathcal{H}_\infty$ dove vogliamo limitare:

- Il valore singolare massimo di S per il rispetto delle performance
- Il valore singolare massimo di T per garantire robustezza e per diminuire la sensibilità al rumore di misura
- Il valore singolare massimo di KS per limitare lo sforzo di controllo

Definito quindi N come: $\begin{bmatrix} WpS \\ WtT \\ WuKS \end{bmatrix}$ l'obiettivo è quello di trovare K che minimizzi la norma infinito di N(K).
Inoltre si vuole che il controllore realizzato sia robustamente stabile (ovvero che il sistema rimanga stabile per tutti gli impianti nel set di incertezza) e robustamente performante (ovvero che le richieste di performance vengano rispettate per tutti gli impianti nel set di incertezza).
Tale problema è stato risolto in MATLAB in due differenti modi. Nel primo il peso sull'incertezza (Wi) viene costruito a mano mentre nel secondo si utilizza la funzione MATLAB **lftdata**.
Nel presente capitolo si analizza il primo di questi metodi mentre il secondo verrà discusso in seguito.

3.2 MATLAB

Il sistema preso in considerazione presenta due ingressi e 4 uscite. Esse sono: $\begin{bmatrix} y & \theta & \dot{\phi} & \psi \end{bmatrix}^T$.
All'interno del codice sono presenti due Script riguardanti la sintesi di un controllore $\mathcal{H}_\infty$: **Hinf_controller_mixsyn** e **Hinf_controller_hinfsyn** i quali utilizzano rispettivamente per la sintesi i comandi mixsyn e hinfsyn.
Lo Script **Hinf_controller_mixsyn** è strutturato nel seguente modo:

- Definizione delle matrici della dinamica certe e incerte e funzioni di trasferimento che ne conseguono
- Scelta delle funzioni peso: Wp, Wt e Wu
- Progetto del controllore $\mathcal{H}_\infty$ e verifica delle funzioni peso precedentemente scelte
- Definizione della funzione peso sull'incertezza (Wi)
- Interconnessione e costruzione del sistema in forma di Doyle
- Mu-analisi

3.3 Definizione delle matrici della dinamica

Come prima cosa all'interno dello Script si richiama la funzione **get_linearization()** dove viene effettuata la linearizzazione del modello non lineare. Tale funzione restituisce una struct con le 4 matrici della dinamica certe (A,B,C,D) e le due matrici incerte (A_i e B_i).
Si costruiscono poi la funzione di trasferimento incerta (Gp), e la funzione di trasferimento nominale (sys).

3.4 Scelta delle funzioni peso

Per garantire una buona reiezione ai disturbi si sceglie una S (funzione di sensibilità) piccola a basse frequenze che comporta una funzione di sensibilità complementare (T) di modulo circa 1 la quale garantisce invece robustezza e un buon inseguimento del riferimento. Ad alte frequenze vale l'opposto: S circa 1 e T circa 0 che permette di attenuare il rumore di misura.
Quindi, ricapitolando, S è una funzione che cresce mentre T è una funzione decrescente.
Volendo quindi minimizzare $\| \begin{bmatrix} WpS \\ WtT \\ WuKS \end{bmatrix} \|_\infty$, la Wp viene scelta in modo tale che i valori singolari di S siano sotto a 1/Wp, la Wt in modo che i valori singolari di T siano sotto a 1/Wt e la Wu in modo che i valori singolari di KS siano sotto a 1/Wu. La condizione sulla Wp permette di rispettare quelle che sono le performance nominali quali l'inseguimento di un riferimento o la reiezione dei disturbi, la condizione sulla Wt è necessaria per la robustezza e l'attenuazione del disturbo di misura, infine la condizione sulla Wu consente di limitare lo sforzo di controllo.

3.4.1 Peso sulla performance

Per costruire la WP si sono utilizzati i seguenti parametri:

- $M = 2$: picco massimo di S che da prassi garantisce buoni margini di guadagno sul sistema
- $AP = 3$: errore massimo a regime
- $wBp = 1$: frequenza minima di banda per la performance

La Wp scelta è dunque della forma:

$$Wp = (\frac{\frac{s}{M} + wBp)}{(s + wBpAP)} = \frac{s+2}{2s+6} \tag{3.1}$$

3.4.2 Peso sulla T

Per la costruzione della Wt si è scelta come banda passante wBt 1 rad/s. Essa è della forma:

$$Wt = \frac{s}{s + wBt} = \frac{s}{s + 1} \tag{3.2}$$

3.4.3 Peso sullo sforzo di controllo

Il peso WU scelto per limitare lo sforzo di controllo è una semplice funzione costante, in tal caso la matrice identità.

$$W_U = tf(eye(2)) \tag{3.3}$$

Ovviamente anche i pesi sopra riportati (Wp e Wt) sono stati in seguito impilati in matrici diagonali di opportune dimensioni con il comando MATLAB **blkdiag**.

3.5 Progetto del controllore e verifica delle funzioni peso

Il comando **mixsyn** prende in ingresso la funzione di trasferimento nominale sys e le 3 funzioni peso, WP, WU e WT, e restituisce il controllore Kmix oltre ad uno scalare gopt che rappresenta il miglior livello di performance raggiunto.

$$\begin{bmatrix} Kmix & ghinf & gopt \end{bmatrix} = mixsyn(sys, WP, WU, WT); \tag{3.4}$$

Una volta costruito il controllore si è verificata la correttezza delle funzioni peso precedentemente scelte.
Sono state realizzate 3 figure:

- Nella prima si sono messi a confronto i valori singolari di S con il valore singolare di $\frac{1}{Wp}$
- Nella seconda si sono messi a confronto i valori singolari di T con il valore singolare di $\frac{1}{Wt}$
- Nella terza si sono messi a confronto i valori singolari di KS con il valore singolare di $\frac{1}{Wu}$

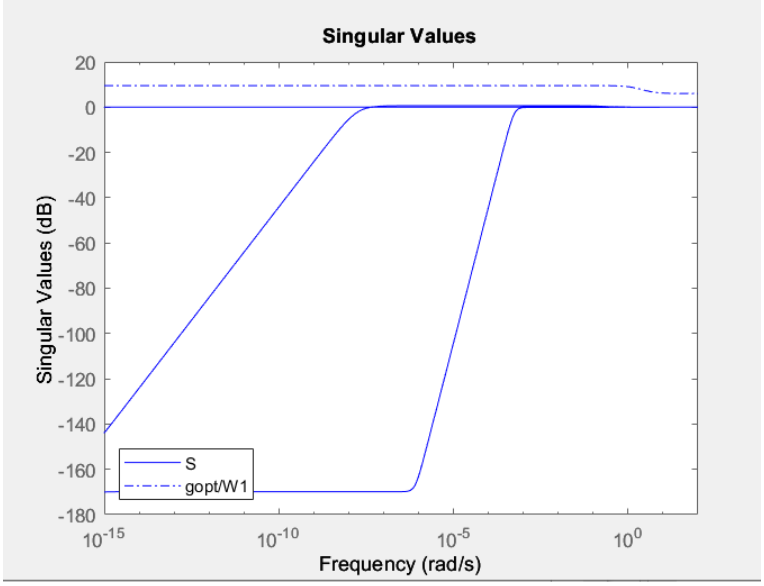


Figura 3.1: S confrontata con 1/WP

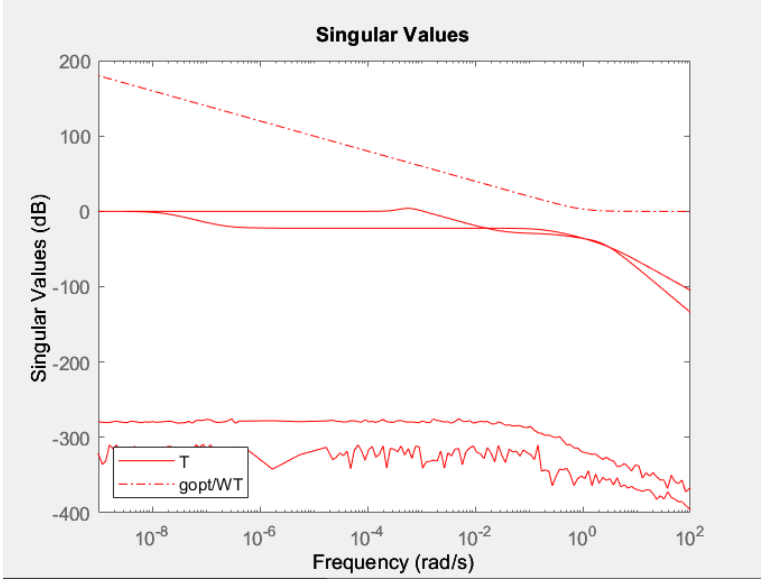


Figura 3.2: T confrontata con 1/WT

3.6 Costruzione della Wi

Prima di poter definire il peso sulla Wi, è bene introdurre il tipo di incertezza preso in considerazione per il nostro sistema.
Siamo in presenza di incertezza parametrica, in particolare si ha incertezza sui raggi delle ruote anteriori e posteriore.
Tale incertezza viene modellata come incertezza moltiplicativa in ingresso, ovvero la nostra Gp (funzione di trasferimento incerta) è data da: $Gp = G(1 + Wi\Delta_i)$ dove G rappresenta la funzione di trasferimento nominale.

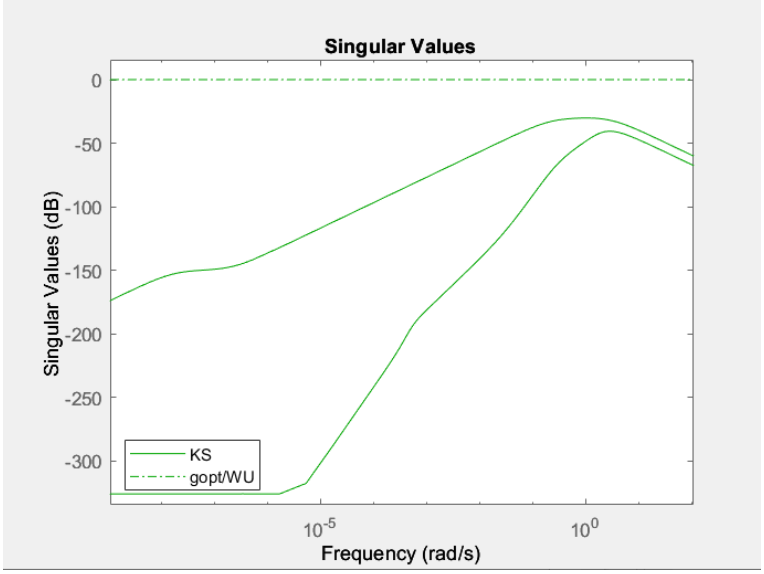


Figura 3.3: KS confrontata con 1/WU

A causa della presenza di parametri incerti e della loro variazione, vengono generate delle regioni di numeri complessi le quali hanno in generale una difficile formulazione matematica.
Tali regioni vengono approssimate come dischi di incertezza.
Nel caso specifico di incertezza moltiplicativa, bisogna trovare, ad ogni frequenza, il più piccolo raggio $l_i(\omega)$ che include tutti i possibili impianti.
$|W_i(j\omega)| > l_i(\omega) = \max_{Gp}(|\frac{Gp-G}{G}|)$
Essendo la nostra G non quadrata, e non essendo possibile applicare su MATLAB la pseudoinversa a delle funzioni di trasferimento, si è utilizzata l'inversa sinistra (G_inv nello Script).
Si sono poi plottati i valori singolari di $G\_inv(Gp-G)$ e si è cercato di costruire una Wi al di sopra di quest'ultimi. Come è possibile notare dalla figura 3.4, tali valori singolari presentano una grande pendenza a basse frequenze dovuta alla presenza di integratori, di conseguenza non è stato possibile trovare una funzione peso Wi che ne costituisse un upper bound a tutte le frequenze.
La funzione peso scelta è stata realizzata con il comando MATLAB **ucover** in modo tale che rispetti la condizione sopra riportata soltanto nel range di frequenze in cui lavora il nostro sistema.

$$Wi = \frac{0.3903s^2 + 0.5745s + 0.3903}{s^2 + 1.474s + 1} \tag{3.5}$$

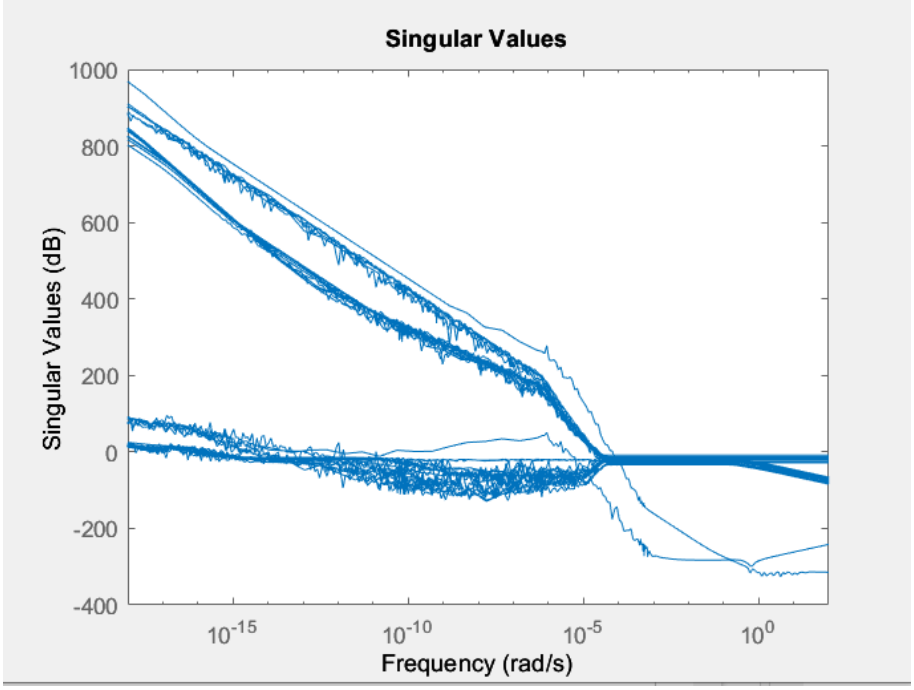


Figura 3.4: Autovalori singolari di $\frac{Gp-G}{G}$

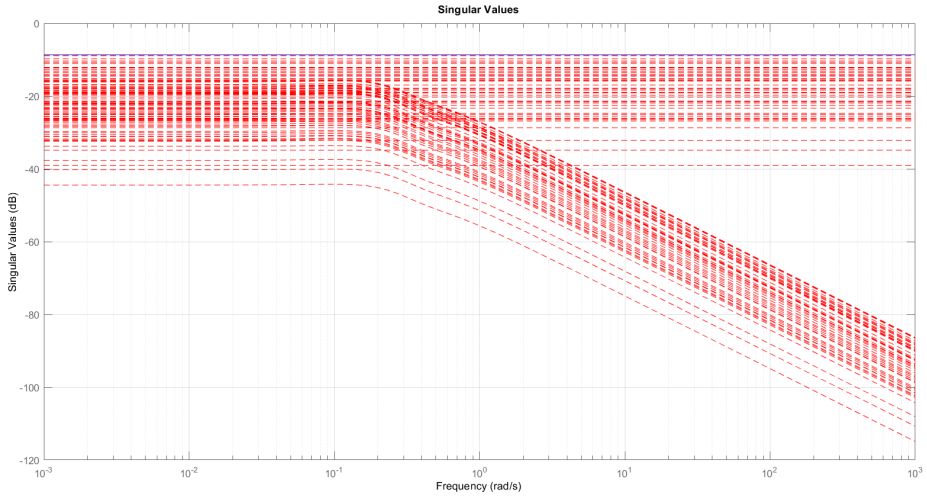


Figura 3.5: Autovalori singolari di $\frac{Gp-G}{G}$ e Wi

3.7 Interconnessione

Definito lo schema a blocchi nel modo indicato in figura 3.6 , si utilizza il comando MATLAB **sysic** che permette di interconnettere vari sottosistemi. Si devono dunque definire le seguenti variabili:

- systemnames: Nomi dei sottosistemi i quali devono essere già stati definiti nel

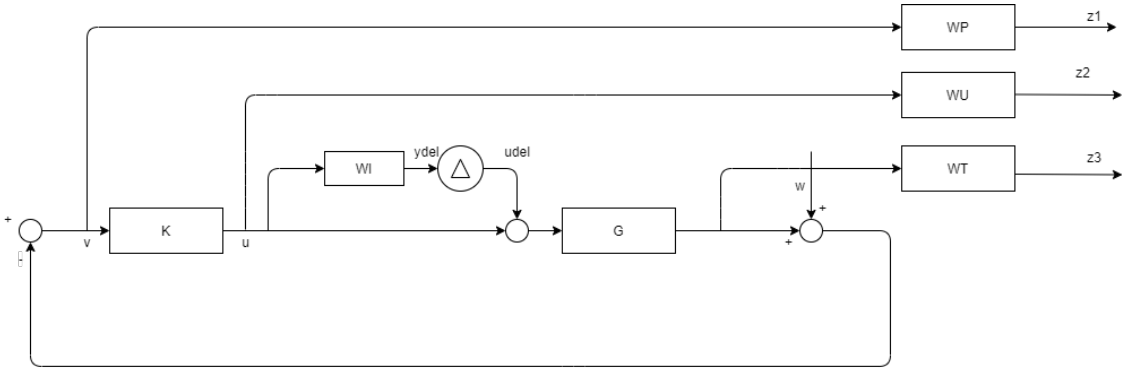


Figura 3.6: Schema a blocchi di riferimento

Workspace

- inputvar: Nomi delle variabili di ingresso all'interconnessione (ovvero alla nostra P)
- outputvar: Nomi delle variabili di uscita dal sistema P
- input_to_systemnames: Nomi degli ingressi ai particolari sottosistemi
- sysoutname: Nome del sistema interconnesso

Nel caso particolare preso in considerazione, si ha:

```
systemnames = 'sys Wi WP WU WT';
inputvar = '[udel{2}; w{4}; u{2}]';
outputvar = '[Wi ; WP ; WU; WT; -w-sys]';
input_to_sys = '[u+udel]';
input_to_Wi = '[udel]';
input_to_WP = '[-w-sys]';
input_to_WU = '[u]';
input_to_WT = '[sys]';
sysoutname = 'P';
cleanupsysic = 'yes';
sysic;
P = minreal(ss(P));
```

Figura 3.7: sysic

3.8 Forma Di Doyle

Dopo aver portato il sistema nella forma della figura 3, si partiziona la P in modo tale che:

$$\begin{aligned} z &= P11w + P12u \\ v &= P21w + P22u \end{aligned} \tag{3.6}$$

Per l'analisi della performance a ciclo chiuso il controllore è dato e possiamo dunque assorbire K nella struttura interconnessa e ottenere il sistema N tale per cui: $z = Nw$ dove N è una funzione di K. Per ottenere N basta sostituire nelle due equazioni precedenti l'equazione del controllore: $u = Kv$.
Ciò che risulta è:

$$N = P11 + P12K(I - P22K)^{-1}P21 \tag{3.7}$$

che viene definita come $F_l(P, K)$ (Lower Linear Fractional Transformation (LFT) di P con K come parametro).
Il comando MATLAB per effettuare la Lower Linear Fractional Transformation è **lft**. Quindi una volta definito N come lft(P,Kmix) si calcola la risposta in frequenza con il comando **frd**.
Si definisce poi la matrice Delta come una matrice incerta piena attraverso il comando **ultidyn**.
Si hanno ora tutti gli strumenti per effettuare la mu-analisi e verificare dunque se il controllore costruito in questo modo è robustamente stabile e performante.

3.9 Studio della robustezza: μ-*analysis*

Prima di presentare il codice MATLAB, è bene introdurre il concetto di μ o di valore singolare strutturato.
Il primo passo è quello di trovare il valore minimo di Δ (in termini di valore singolare massimo) che rende la matrice I-MΔ singolare. La μ non è altro che l'inverso di tale valore singolare. In termini di prestazioni e stabilità le condizioni da verificare sono:

- NS: N internamente stabile
- RS: $\mu_{\Delta_i}(N11) < 1$ e NS
- NP: $\mu_{\Delta_p}(N22) < 1$ e NS
- RP: $\mu_{\Delta}(N) < 1$ e NS

dove

$$\Delta = \begin{bmatrix} \Delta_i & 0 \\ 0 & \Delta_p \end{bmatrix} \tag{3.8}$$

Dunque per la nominale stabilità si sono verificati gli autovalori di N attraverso il comando **eig**. Per la robusta stabilità si è utilizzato il comando MATLAB **mussv**

```
>> eig(N)

ans =

  -1.6901 + 1.4457i
  -1.6901 - 1.4457i
  -2.9518 + 0.0000i
  -3.0223 + 0.0000i
  -0.7368 + 0.6761i
  -0.7368 - 0.6761i
  -3.0000 + 0.0000i
  -1.1395 + 0.0000i
  -0.4329 + 0.0000i
  -0.2466 + 0.1491i
  -0.2466 - 0.1491i
  -0.3000 + 0.0000i
  -0.2200 + 0.0000i
  -0.0005 + 0.0000i
  -0.0003 + 0.0005i
  -0.0003 - 0.0005i
  -0.0000 + 0.0000i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
  -3.0000 + 0.0000i
  -3.0000 + 0.0000i
  -3.0000 + 0.0000i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
```

Figura 3.8: Autovalori di N

il quale prende in ingresso la matrice N_{11} che mette in relazione udel con ydel e la struttura dell'incertezza (nel nostro caso 2x2 piena), e restituisce in uscita l'upper e il lower bound del valore singolare strutturato. Attraverso poi il comando **norm** è possibile estrarre la norma infinito e ottenere il valore di μ desiderato.
Stesse considerazioni valgono per la prestazione nominale e per la prestazione robusta facendo attenzione alle matrici di ingresso al mussv e all'incertezza che vi è associata. Si sono poi testate la robusta stabilità e la robusta performance attraverso i comandi MATLAB **robstab** e **robustperf**. Per il primo dei due, come primo passo si è creata la Upper lft tra Delta (matrice di incertezza) ed N ($M = lft(Delta, N)$) , poi si è riportata tale M in frequenza attraverso il comando ufrd ($Mf = ufrd(M, omega)$).
Robstab prende in ingresso tale Mf, ed eseguendo il comando, possiamo notare che nella Command Window ci viene direttamente stampato il livello di incertezza che può ancora supportare il sistema.
Per il secondo, si sono effettuati gli stessi passaggi di prima, dove la delta utilizzata stavolta è una matrice diagonale che comprende anche l'incertezza fittizia Δ_p.

```
Stabilità robusta muNP: 0.620027
Prestazione nominale muNPinf: 0.034102
Robusta prestazione muRPinf: 0.620027
****************************************
```

Figura 3.9: Indici di RS, NP e RP con mixsyn

```
System is robustly stable for the modeled uncertainty.
 -- It can tolerate up to 138% of the modeled uncertainty.
 -- There is a destabilizing perturbation amounting to 138% of the modeled uncertainty.
 -- This perturbation causes an instability at the frequency 1e+06 rad/seconds.
 -- Sensitivity with respect to each uncertain element is:
      100% for Delta. Increasing Delta by 25% decreases the margin by 25%.
```

Figura 3.10: Robstab con mixsyn

Il comando **robustperf** stampa sulla linea di comando se il sistema è robustamente performante o meno.

3.10 Risultati con Hinfsyn

Come già accennato in precedenza, è stato realizzato un ulteriore Script dove il controllore è stato implementato con il comando MATLAB **Hinfsyn**.
La scelta delle funzioni peso WP, WU, WT e Wi è la stessa dello Script Hinf_controller_mixsyn così come l'interconnessione.
Il comando Hinfsyn prende in ingresso la P creata tramite il sysic e il numero di uscite e ingressi dell'impianto restituendo il controllore Khinf e uno scalare gamma, indice del livello di performance raggiunto.
Per l'analisi dei risultati si sono svolti i medesimi passaggi sopra riportati. In questo caso la robusta stabilità viene soddisfatta mentre lo stesso non vale per performance nominale e performance robusta. Tale risultato verrà in seguito migliorato attraverso la DK-iteration.

```
'Assuming nominal UFRD system is stable...↵Uncertain system achieves performance robustness to modeled uncertainty.'
' -- The tradeoff of model uncertainty and system gain is balanced at a level of 138% of the modeled uncertainty.  '
' -- A model uncertainty of 138% can lead to input/output gain of 0.723 at 1e+06 rad/seconds.                      '
' -- Sensitivity with respect to each uncertain element is:                                                       '
'      100% for Delta. Increasing Delta by 25% decreases the margin by 25%.                                       '
'      0% for Deltap. Increasing Deltap by 25% decreases the margin by 0%.                                        '
```

Figura 3.11: Robustperf con mixsyn

```
****************************************:
Stabilità robusta muRS: 0.767560
Prestazione nominale muNPinf: 2.622556
Robusta prestazione muRPinf: 2.622556
****************************************:
```

Figura 3.12: Indici di RS,NP E RP con hinfsyn

3.11 Simulink

I controllori creati sono stati poi validati all'interno di uno schema Simulink, sia nel caso di modello lineare che non.

3.11.1 Modello lineare

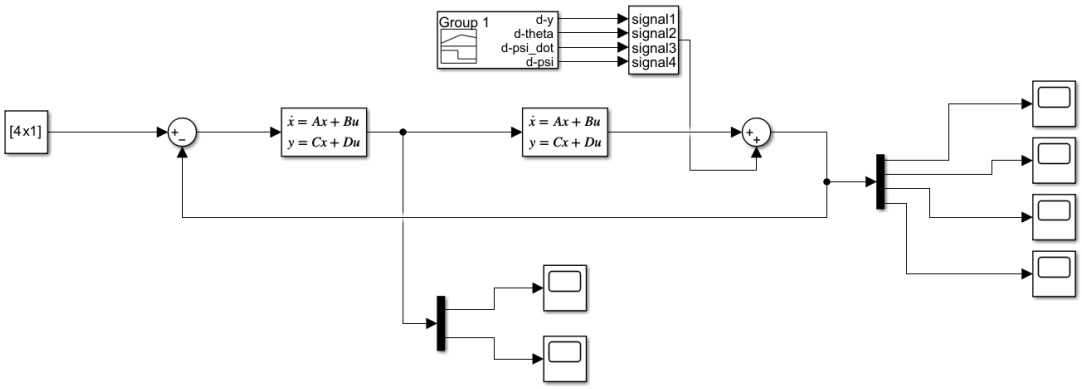


Figura 3.13: Simulink modello lineare

Tale schema a blocchi è costituito da un primo blocco in state space che rappresenta il controllore progettato e da un secondo blocco che rappresenta il sistema.
Il riferimento è nullo e inoltre in uscita all'impianto sono stati aggiunti dei rumori per poter vedere il comportamento del sistema anche in loro presenza.
Le condizioni iniziali scelte per questo tipo di simulazione sono state le seguenti:

$$X\_start = \begin{bmatrix} 1 \\ 0.1 \\ -0.5 \\ 0 \\ 0 \end{bmatrix} \tag{3.9}$$

Come si può notare dalla figura 3.14, nonostante la presenza di disturbi che rendono l'andamento non regolare, tutte le variabili di stato convergono a zero. Nell'esempio

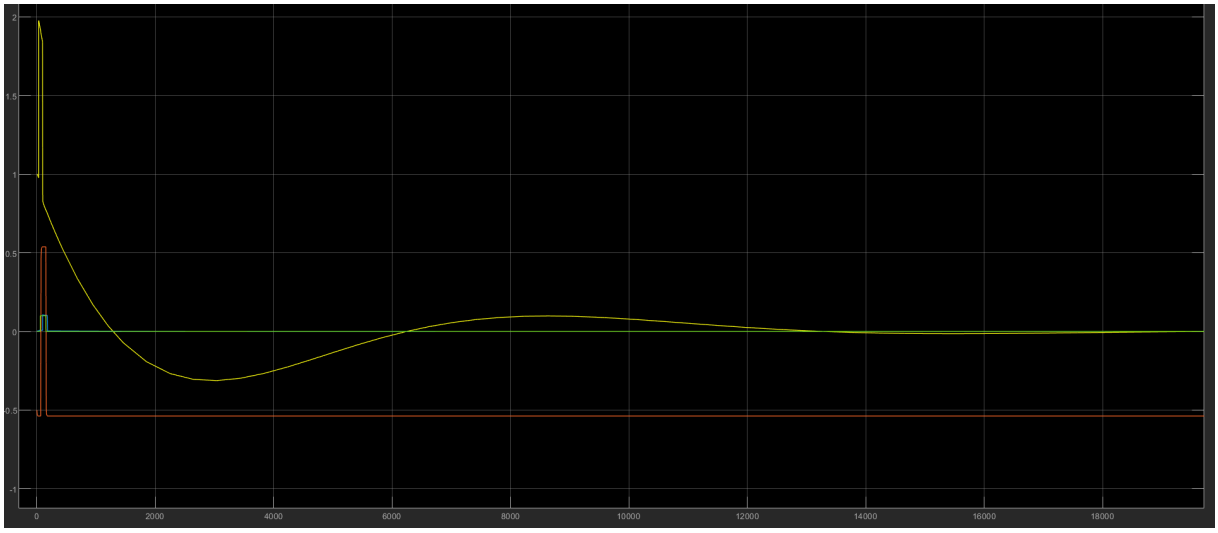


Figura 3.14: Risultati Simulink modello lineare

in figura non si riesce a vedere la convergenza della $\dot{\phi}$ poichè è più lenta a convergere. Tali simulazioni sono state effettuate con il controllore calcolato tramite il comando

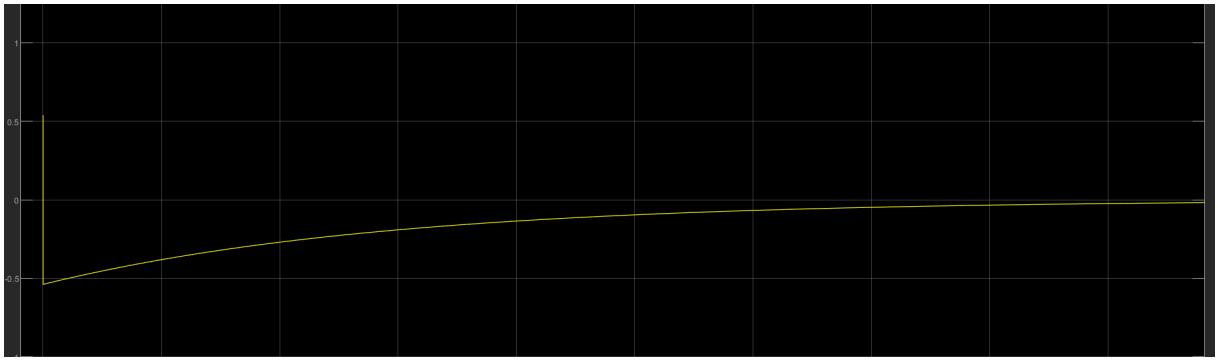


Figura 3.15: Andamento della $\dot{\phi}$

mixsyn; considerazioni analoghe possono essere fatte usando quello generato da **hinfsyn**.

3.11.2 Modello non lineare

I controllori generati sono stati poi testati sul modello non lineare, al fine di verificare se la traiettoria desiderata venisse veramente eseguita dall'AGV.

Lo schema a blocchi è simile a quello descritto per il modello lineare ad eccezione del sistema lineare che è stato ora sostituito con un subsystem che contiene al suo interno la cinematica e la dinamica del modello non lineare.

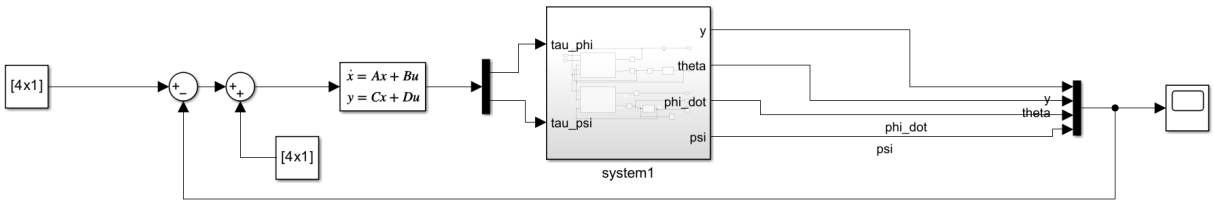


Figura 3.16: Schema Simulink Modello Non Lineare

Il vettore di condizioni iniziali scelto è stato:

$$X\_start = \begin{bmatrix} 0.3 \\ 0.4 \\ -1 \\ 0 \\ 0 \end{bmatrix} \tag{3.10}$$

Tale scelta è stata fatta in modo da mantenere le condizioni iniziali sufficientemente vicine alla traiettoria di equilibrio. Osservando la figura 3.17 si può notare come la

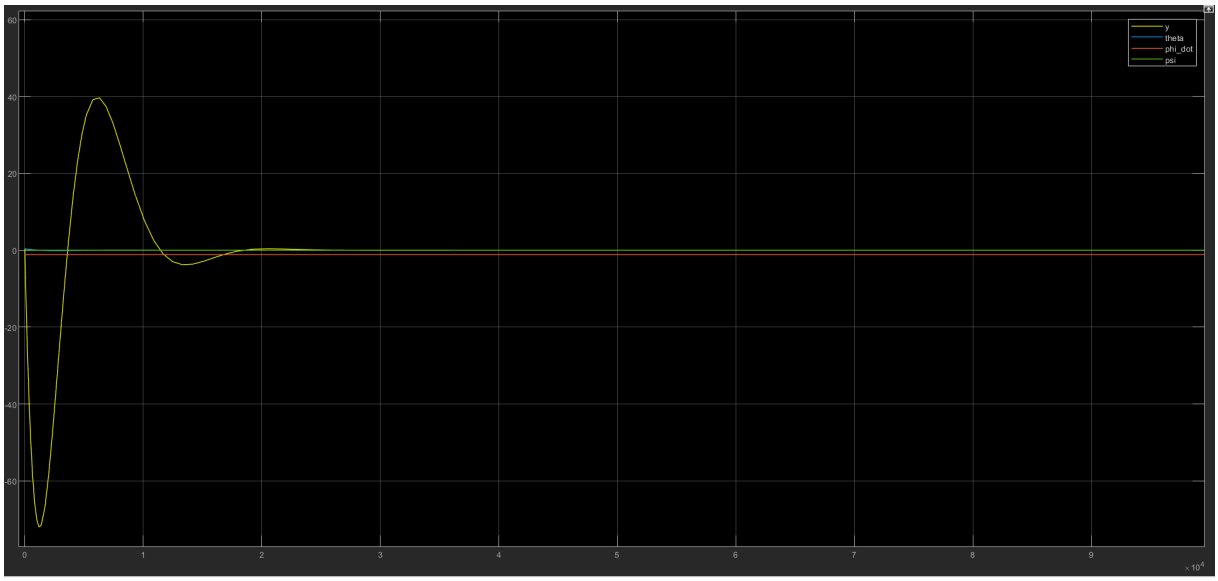


Figura 3.17: Scope del Modello Non Lineare

y, la θ e la ψ convergano a zero, dovendo l'AGV seguire la traiettoria y=0, e come invece la $\dot{\phi}$ rimanga invece ad un valore costante.

Capitolo 4

Controllo con DK-iteration: *μ-synthesis*

Il valore singolare strutturato μ è uno strumento molto potente per studiare la robusta performance partendo da un controllore K noto. Tuttavia può essere interessante chiedersi se sia possibile determinare tale controllore partendo già dall'obiettivo di minimizzare una data condizione su μ. Tale problema rientra nel campo della *μ-analysis*. Ad oggi non esiste un algoritmo in letteratura in grado di risolvere direttamente tale problema, ma, in presenza di perturbazioni complesse, è possibile sfruttare il metodo DK-*iteration* che presenta buoni risultati da un punto di vista euristico. Tale metodo iterativo combina in due step la sintesi del controllo ottimo $\mathcal{H}_\infty$ e la *μ-analysis* per la robustezza. L'idea che sta alla base di tale metodo è quella di trovare il controllore che minimizzi il valore di picco sulla frequenza del limite superiore di μ espresso in termini del valore singolare scalato:

$$\min_K(\min_{D\in\mathcal{D}}(\|DN(K)D^{-1}\|_\infty))$$

Per l'inizializzazione si sceglie solitamente una matrice di trasferimento D(s) pari all'identità, strutturata in modo coerente con le incertezze definite nel modello. A questo punto il DK-iteration procede nel seguente modo:

1. **K-step**: sintetizzare un controllore $\mathcal{H}_\infty$ per il problema scalato tenendo fissa la matrice di trasferimento D(s)

$$\min_K(\|DN(K)D^{-1}\|_\infty)$$

2. **D-step**: fissato il controllore K trovato al passo precedente, trovare $D(j\omega)$ che minimizza

$$\bar{\sigma}(DND^{-1}(j\omega))$$

3. Adattare il modulo di ogni termine della matrice D(s) a funzioni di trasferimento stabili e a fase minima e tornare al passo 1.

L'implementazione del metodo iterativo D-K è stata effettuata all'interno dello script *DK_ iteration*. Il primo passo da effettuare riguarda la scrittura del sistema nella forma di **Doyle**, mostrato in figura 3.6. Si ricava l'impianto generalizzato P a partire dai blocchi contenenti *sys*, l'impianto nominale ottenuto sostituendo i valori medi alle incertezze, W_P, il peso per la prestazione, W_U, il peso per limitare l'azione di controllo, W_T, il peso per l'attenuazione del rumore di misura, e W_i, il peso sull'incertezza Δ_i rappresentata da una matrice complessa piena di dimensione 2x2.

Le funzioni peso sono state definite come segue:

- Matrice peso per l'incertezza W_i: la funzione peso scalare w_i è stata determinata attraverso il comando *ucover* di MATLAB applicato all'impianto nominale del sistema. Siccome i valori singolari presentano una pendenza tipica dell'integratore alle basse frequenze, non è stato possibile trovare una funzione peso in grado di rispettare per ogni frequenza la condizione $w_i \geq \frac{G_i - G}{G}$. Tale condizione viene tuttavia rispettata nelle frequenze di interesse (da 10^{-3} rad/s a 10^3 rad/s), come mostrato alla figura 4.1:

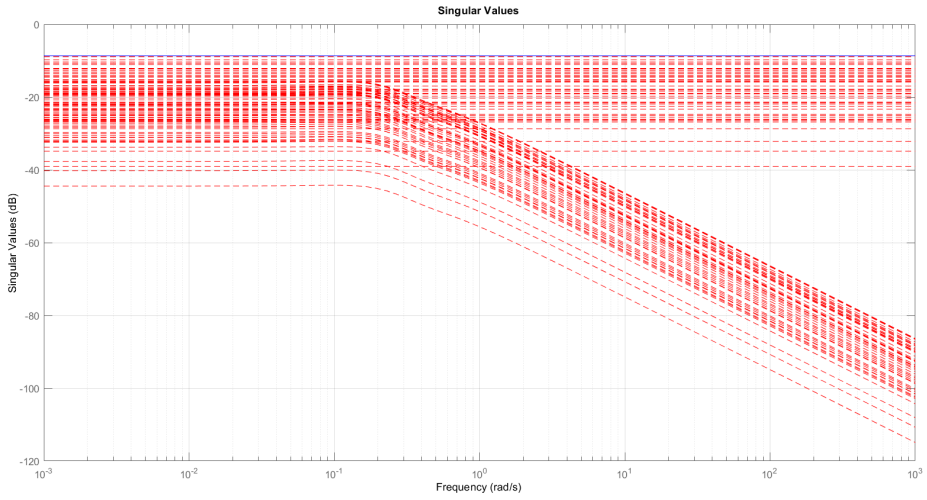


Figura 4.1: In blu: funzione peso wi. In rosso: valori singolari di $\frac{G_i - G}{G}$

La matrice complessiva W_i è quindi data da:

$$W_i = \begin{bmatrix} w_i & w_i \\ w_i & w_i \end{bmatrix} \tag{4.1}$$

- Matrice peso per la performance W_P: è stata definita la funzione scalare

$$w_P = \frac{\frac{s}{M + wBp}}{s + wBp * AP}$$

dove $M = 2$, $AP = 3$ e $wBp = 1$. Per cui la matrice peso W_P complessiva è data da:

$$W_P = \begin{bmatrix} w_P & 0 & 0 & 0 \\ 0 & w_P & 0 & 0 \\ 0 & 0 & w_P & 0 \\ 0 & 0 & 0 & w_P \end{bmatrix} \tag{4.2}$$

- Matrice peso per lo sforzo di controllo W_U: è stata definita la funzione scalare $w_U = 1$. Per cui la matrice peso W_U complessiva è data da:

$$W_U = \begin{bmatrix} w_U & 0 \\ 0 & w_U \end{bmatrix} \tag{4.3}$$

- Matrice peso per l'attenuazione del rumore di misura W_T: è stata definita la banda passante $w_{BT} = 1$ e quindi la funzione scalare $w_T = \frac{s}{s+w_{BT}}$. Per cui la matrice peso W_T complessiva è data da:

$$W_T = \begin{bmatrix} w_T & 0 & 0 & 0 \\ 0 & w_T & 0 & 0 \\ 0 & 0 & w_T & 0 \\ 0 & 0 & 0 & w_T \end{bmatrix} \tag{4.4}$$

A questo punto vengono eseguite le iterazioni della procedura DK sfruttando il comando automatico di MATLAB *musyn*. E' necessario specificare il numero di uscite ($nmeas = 4$) e il numero di ingressi ($nu = 2$) del sistema. *musyn* prende in ingresso la matrice Fu ottenuta applicando la *lower linear fractional transformation* tra il blocco delle incertezze Δ_i definito in precedenza e la P dell'impianto generalizzato.

```
...
nmeas = 4; nu = 2;
omega = logspace(-1,6,302);
Fu = lft(Deltai,P);

opts = musynOptions('Display','full','MaxIter',100,'
TolPerf',0.001,'FrequencyGrid',omega)
[K_DK,CLPperf,info_mu] = musyn(Fu,nmeas,nu,opts);
...
```

Il K-step e il D-step vengono ripetuti o finché non si superano le 100 iterazioni o finché non si ottengono risultati migliori con un margine di 0.001 sul picco di μ.

Una volta aver determinato il controllore K_DK, è possibile verificare graficamente che l'inverso delle funzioni peso selezionate per le prestazioni si trovano tutte al di sopra dei valori singolari massimi delle funzioni a cui sono associate.

In particolare, nella figura 4.2 osserviamo che l'inverso della funzione w_P si trova sempre sopra al valore singolare massimo della funzione sensitività S.

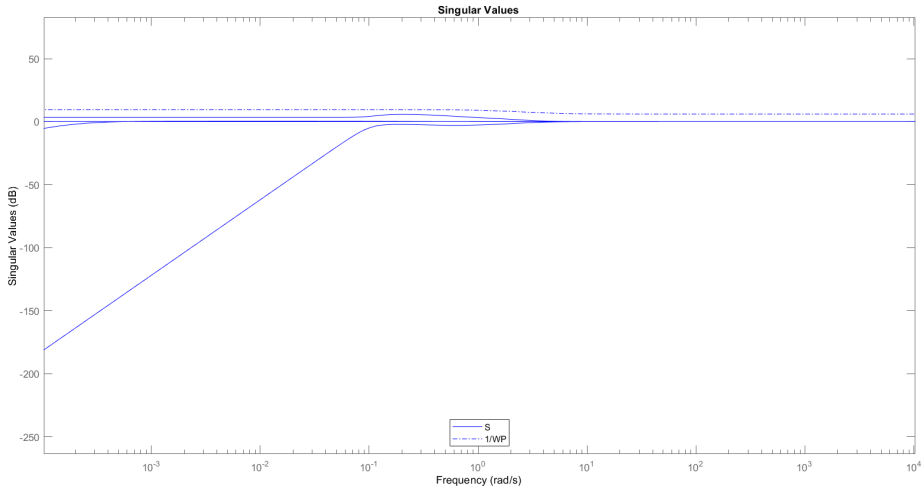


Figura 4.2: Funzione peso w_P confrontata con il valore singolare massimo della funzione sensitività S

Nella figura 4.3 osserviamo che l'inverso della funzione w_U si trova sempre sopra al valore singolare massimo della funzione KS, dove al posto di K è stato posto il controllore calcolato sopra.

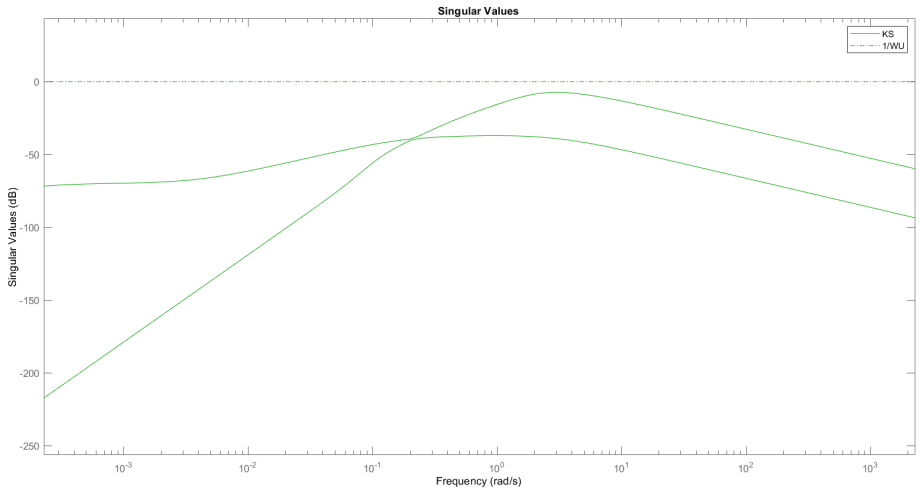


Figura 4.3: Funzione peso w_U confrontata con il valore singolare massimo della funzione KS

Infine, si verifica alla figura 4.4 che l'inverso della funzione w_T si trova sempre al di sopra del valore singolare massimo della funzione sensitività complementare T.

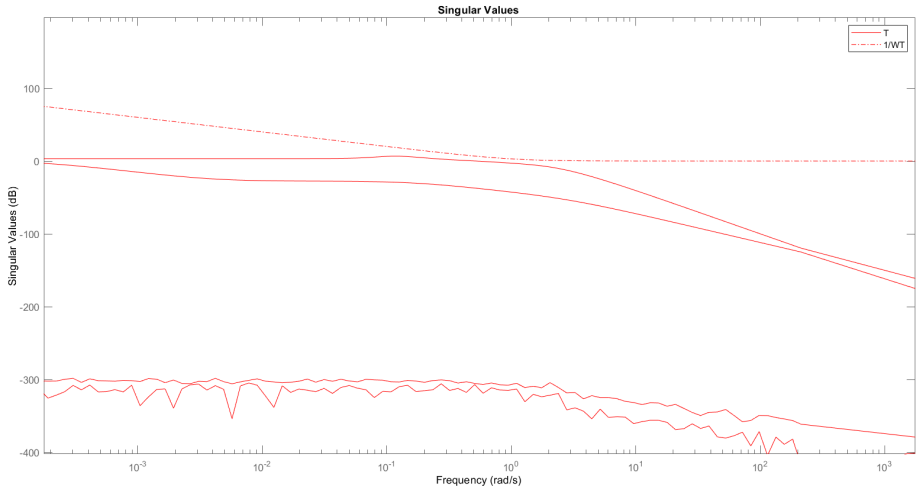


Figura 4.4: Funzione peso w_T confrontata con il valore singolare massimo della funzione sensitività complementare T

4.1 Studio delle prestazioni

Il primo controllo da effettuare per verificare il corretto funzionamento del controllore determinato, è quello relativo alla **stabilità nominale**. Infatti tale condizione deve sempre essere verificata affinché possano dirsi soddisfatte anche le specifiche nominali e robuste.

Per tale scopo si è definita la matrice di trasferimento a ciclo chiuso tramite il comando:

```
...
CL = feedback(sys*K_DK,eye(4))
...
```

Sono poi stati visualizzati gli autovalori di tale matrice e si è osservato che presentano tutti parte reale negativa. Per tale motivo possiamo concludere che il sistema a ciclo chiuso è nominalmente stabile.

Per validare la **robusta stabilità** del sistema a ciclo chiuso è stato utilizzato il comando *robstab* di MATLAB, il quale applica il controllo per la stabilità direttamente alla matrice $M = lft(\Delta_i, N)$. La struct *stabmarg* restituita presenta i seguenti upper e lower bound:

- LowerBound: 1.2924
- UpperBound: 1.2924

Questi valori indicano che il sistema è robustamente stabile, in quanto un margine di stabilità robusta maggiore di 1 indica che il sistema è stabile per tutti i

valori dell'incertezza modellata. Infatti, dal campo *info* dello stesso comando possiamo confermare che il sistema è robustamente stabile e può tollerare fino al 144% dell'incertezza modellata, come mostrato in figura 4.5.

```
Points completed: 302/302
System is robustly stable for the modeled uncertainty.
 -- It can tolerate up to 129% of the modeled uncertainty.
 -- There is a destabilizing perturbation amounting to 129% of the modeled uncertainty.
 -- This perturbation causes an instability at the frequency 1e+06 rad/seconds.
 -- Sensitivity with respect to each uncertain element is:
      100% for Deltai. Increasing Deltai by 25% decreases the margin by 25%.
```

Figura 4.5: Risultati dell'analisi di stabilità robusta.

A questo punto possiamo osservare i risultati ottenuti dal comando *musyn* per validare la **robusta prestazione** del sistema. Come mostrato alla figura 4.6, il picco minimo ottenuto per μ è 0.779.

```
D-K ITERATION SUMMARY:
------------------------------------------------------------------
                        Robust performance              Fit order
------------------------------------------------------------------
  Iter          K Step        Peak MU        D Fit             D
    1           2.514         2.514          2.514             0
    2           1.413         1.412          1.412             0
    3           1.245         1.245          1.245             0
    4           1.086         1.086          1.086             0
    5          0.8715        0.8714         0.8714             0
    6          0.7813        0.7789         0.7815             0
    7          0.7813        0.7793         0.7816             0
    8          0.7814        0.7797         0.7816             0

Best achieved robust performance: 0.779
```

Figura 4.6: Risultati DK-iteration

Questo valore indica che è possibile aumentare di un fattore pari a $\frac{1}{\mu}\% = 128\%$ l'incertezza garantendo ancora le prestazioni richieste.

A confermare tale risultato è possibile applicare un'ulteriore analisi tramite il comando *robustperf* di MATLAB, il quale applica l'analisi di robusta stabilità alla matrice $F_f = lft(\Delta, N)$, la quale tiene conto sia delle incertezze Δ_i che di quelle fittizie Δ_P. Anche in questo caso si ottiene una struct *stabmarg* che presenta i seguenti bound:

- LowerBound: 1.2838

- UpperBound: 1.2838

Tali valori indicano che il sistema è robustamente performante, in quanto un margine di prestazione robusta maggiore di 1 indica che il sistema garantisce le prestazioni richieste per tutti i valori dell'incertezza modellata. Inoltre si sottolinea il fatto che l'inverso del LowerBound coincide proprio con il valore ottimo di μ restituito dalla DK. Infine, alla figura 4.7, riportiamo il campo *info* del comando utilizzato.

```
info =

  6×114 char array

    'Assuming nominal UFRD system is stable...↵Uncertain system achieves performance robustness to modeled uncertainty.'
    ' -- The tradeoff of model uncertainty and system gain is balanced at a level of 128% of the modeled uncertainty.  '
    ' -- A model uncertainty of 128% can lead to input/output gain of 0.779 at 0.381 rad/seconds.                      '
    ' -- Sensitivity with respect to each uncertain element is:                                                        '
    '      100% for DeltaP. Increasing DeltaP by 25% decreases the margin by 25%.                                     '
    '      100% for Deltai. Increasing Deltai by 25% decreases the margin by 25%.                                     '
```

Figura 4.7: Risultati dell'analisi di stabilità robusta.

4.2 Validazione in Simulink

Il controllore ottenuto con il metodo della DK-iteration può essere sfruttato per simularne l'effetto sul sistema implementato. Per tale scopo è stato utilizzato SIMULINK, implementando sia una simulazione per il sistema lineare che per il sistema non lineare.

Per quanto riguarda il primo, lo schema a blocchi utilizzato è quello mostrato in figura 4.8.

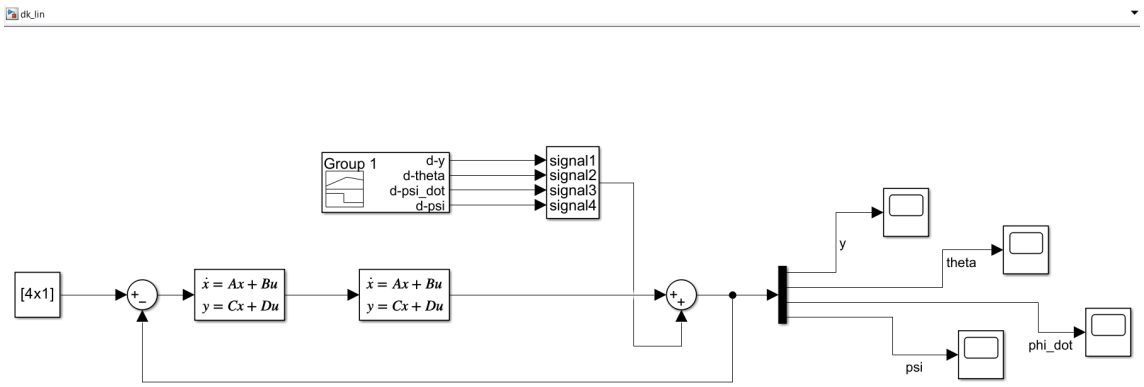


Figura 4.8: Schema SIMULINK del sistema lineare retroazionato sul controllore DK.

Il primo blocco state-space rappresenta il controllore DK, e le matrici della dinamica sono state ricavate nel seguente modo:

```
...
[A_DK B_DK C_DK D_DK] = ssdata(K_DK);
...
```

Il secondo blocco state-space rappresenta il sistema lineare le cui matrici della dinamica sono state ricavate in precedenza tramite linearizzazione intorno alla traiettoria di equilibrio.

Il riferimento fornito in ingresso al sistema è nullo.

In uscita sono sommati alcuni disturbi a gradino per mostrare come il sistema risulti comunque stabile.

Partendo da condizioni iniziali

$$X_{start} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$

ovvero coincidenti con la traiettoria di equilibrio, notiamo che tutti gli stati restano a zero, vedi figure 4.9, 4.10, 4.11, mentre la velocità angolare $\dot{\phi}$ parte da -1 per convergere lentamente verso 0, figura 4.12.

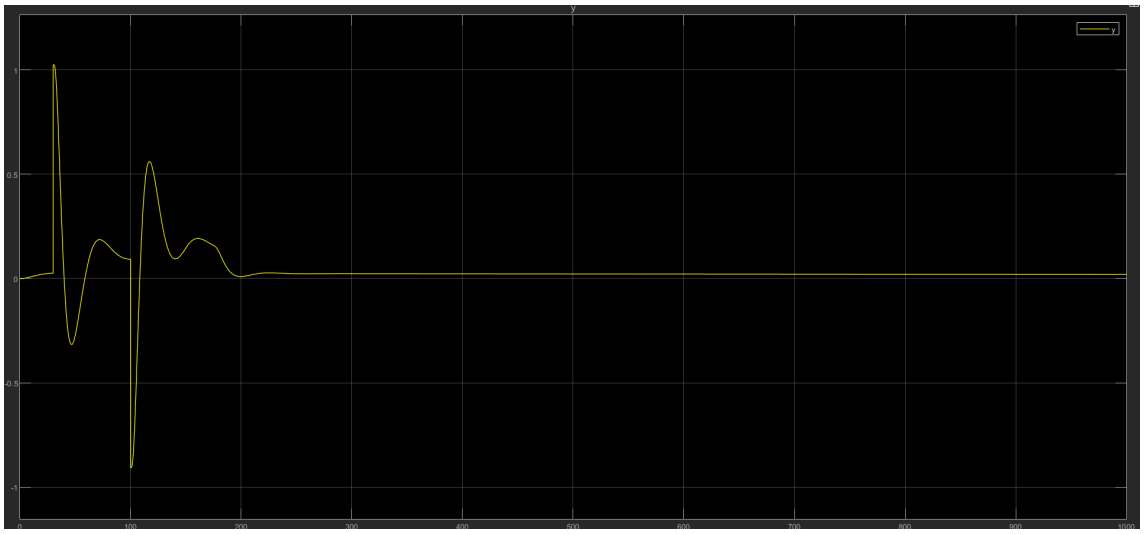


Figura 4.9: Andamento dell'uscita y.

E' interessante sottolineare che tali risultati sono ottenuti anche se le condizioni iniziali di partenza sono diverse da quelle dell'equilibrio.

Infine, è stato testato il controllore anche sul sistema non lineare. Lo schema SIMULINK implementato è quello mostrato in figura 4.13

Anche in questo caso il riferimento è nullo.

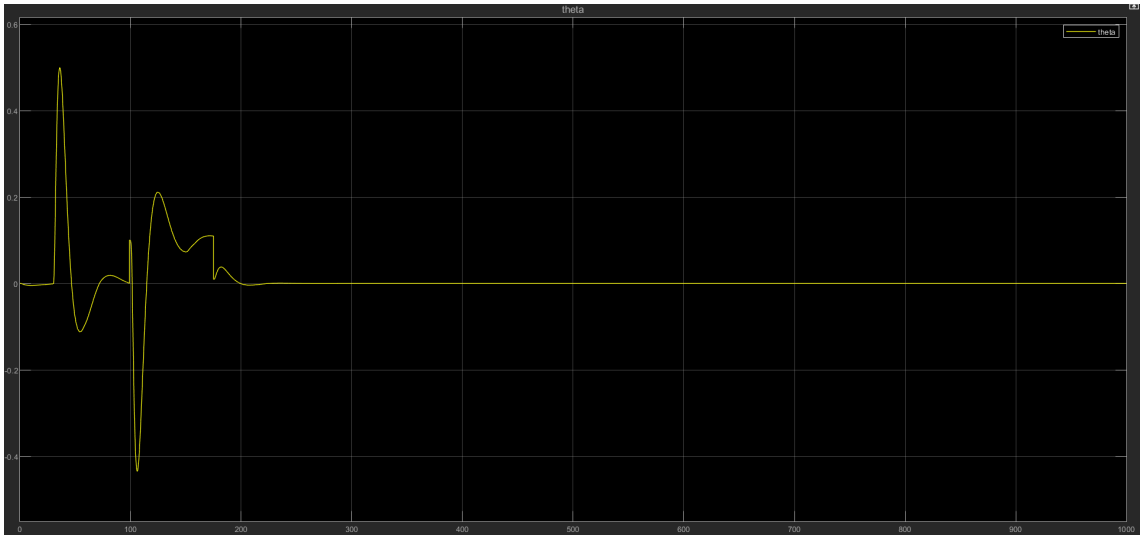


Figura 4.10: Andamento dell'uscita θ.

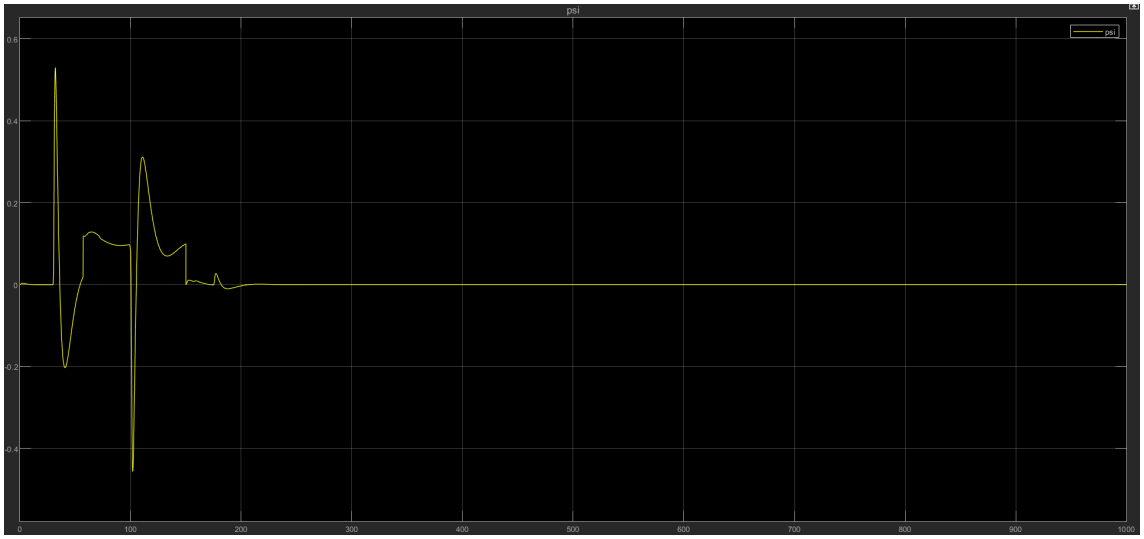


Figura 4.11: Andamento dell'uscita ψ.

In ingresso al controllore viene sommato il vettore $\begin{bmatrix}0;0;-1;0;0\end{bmatrix}$ in quanto il controllore lavora sulle variabili traslate sull'equilibrio. Le uscite ottenute partendo dal punto di equilibrio come condizione iniziale sono mostrate in figura 4.14.

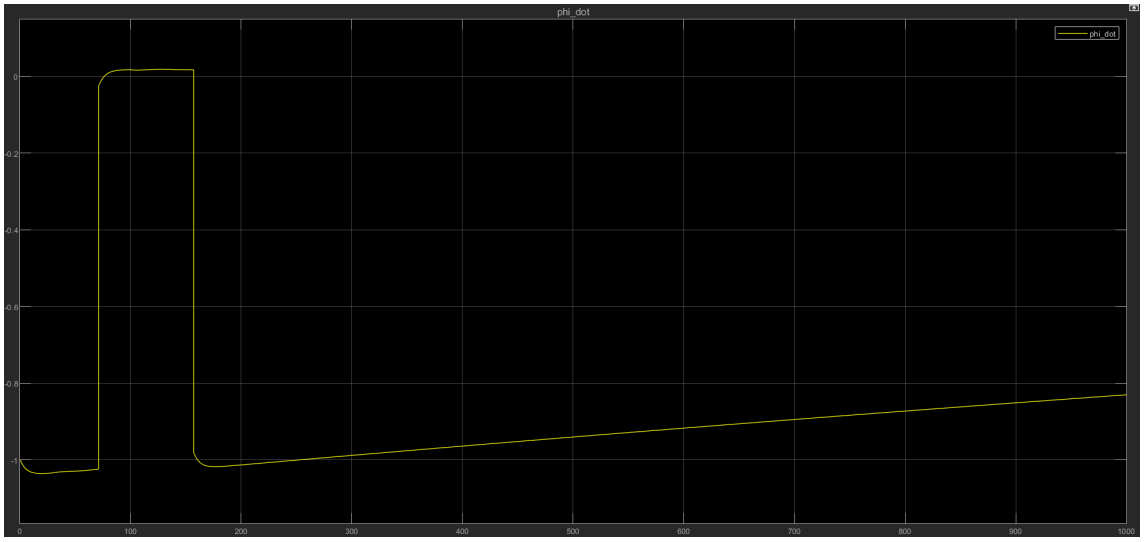


Figura 4.12: Andamento dell'uscita $\dot{\phi}$.

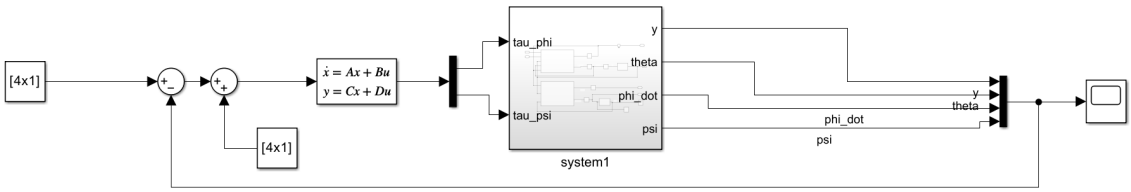


Figura 4.13: Schema SIMULINK del sistema non lineare retroazionato sul controllore DK.

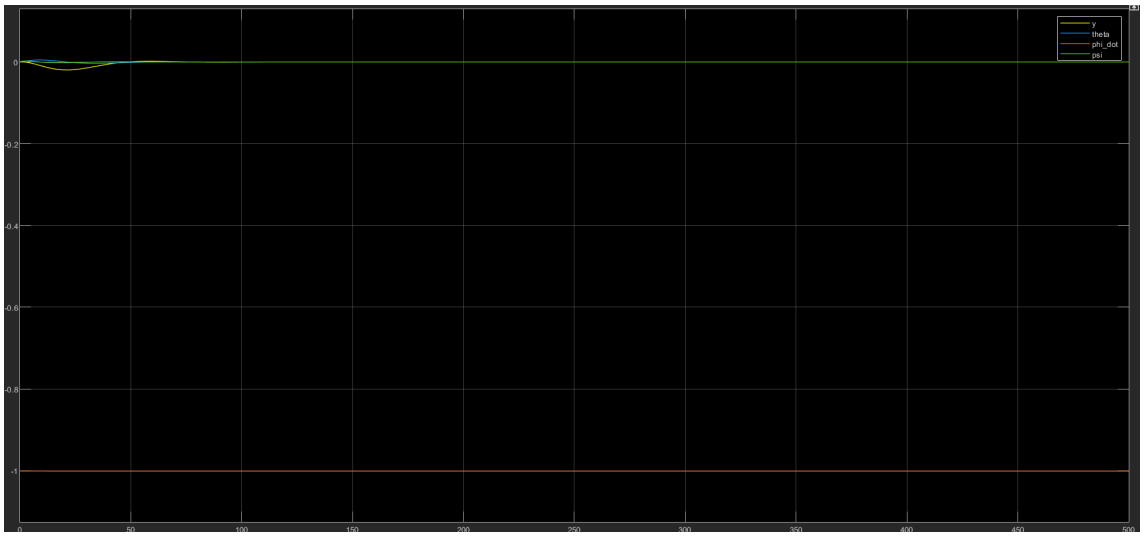


Figura 4.14: Andamento delle uscite dal sistema non lineare.

Capitolo 5

Una via alternativa: costruzione dell'incertezza tramite *lftdata*

In una seconda parte del progetto è stata scelta la via della realizzazione dei controllori H_∞ andando a realizzare la matrice di perturbazione Δ e la P del sistema certa in forma di Doyle tramite il comando **lftdata** che ha lo scopo di decomporre la matrice incerta realizzata con il comando di interconnessione *sysic*. Dunque l'incertezza è stata gestita completamente dal comando MATLAB.
In particolare i due script relativi ai controllori mixed-sensitivity e H_∞ sono stati strutturati così:

- Si ricava la matrice di trasferimento incerta come descritto nella sezione 3.3;
- scelta delle funzioni peso Wp, Wt e Wu;
- interconnessione del sistema con il comando sysic applicando ad esso il sistema con la dinamica incerta;
- Realizzazione del controllore dopo aver ricavato la struttura di delta e P con lftdata;
- Mu-analysis.

5.0.1 Scelta delle funzioni peso

Anche in questo caso i pesi sono stati scelti in modo da garantire le specifiche di prestazione su reiezione dei disturbi, robustezza e reiezione del rumore di misura e minimizzazione dello sforzo di controllo. In particolare sono state scelte le seguenti specifiche per wp, wt ed wu:

- $\mathbf{M = 1}$, picco massimo di S che da prassi garantisce buoni margini di guadagno sul sistema

- **AP** $= 10^{-1}$, errore massimo a regime, sta volta cercando di minimizzare quanto più possibile l'errore a regime. Quindi richiediamo migliori prestazioni da questo punto di vista rispetto al controllore sintetizzato nel capitolo 3;
- frequenza minima di banda wBp $= 1$.
- wBt $= 1$, frequenza minima di banda per attenuazione di rumore di misura;

Di conseguenza le funzioni di trasferimento dei pesi sono:

- $$Wp = (\frac{\frac{s}{M} + wBp)}{(s + wBpAP)} = \frac{0.05s + 0.05}{s + 0.1} \tag{5.1}$$
- $$Wt = \frac{s}{s + wBt} = \frac{500s + 100}{s + 1e04} \tag{5.2}$$
- $$Wu = 0.05. \tag{5.3}$$

Tali pesi sono stati utilizzati per realizzare le matrici di peso WP, WT e WU che hanno sulla diagonale tali funzioni di trasferimento.

5.0.2 Interconnessione

Le connessioni con sysic sono state realizzate con la matrice della dinamica incerta in questo modo:

```
systemnames = 'Gp WP WU WT';
inputvar = '[w{4}; u{2}]';
outputvar = '[WP ; WU; WT; -w-Gp]';
input_to_Gp = '[u]';
input_to_WP = '[-w-Gp]';
input_to_WU = '[u]';
input_to_WT = '[Gp]';
sysoutname = 'P_i';
cleanupsysic = 'yes';
sysic;
```

dunque si può far riferimento agli schemi a blocchi della sezione 3.3.

5.0.3 Realizzazione del controllore mixed-sensitivity

Dove aver fatto l'interconnessione abbiamo realizzato il controllore mixed-sensitivity con i comandi descritti nel capitolo 3 ma costruiti sulla dinamica incerta. Dopo aver sintetizzato il controllore sono state visualizzate a schermo le condizioni sui massimi valori singolari delle diverse funzioni di sensitività, affinchè essi rispettassero i limiti imposti dai pesi (vedi 5.1, 5.2 e 5.3).

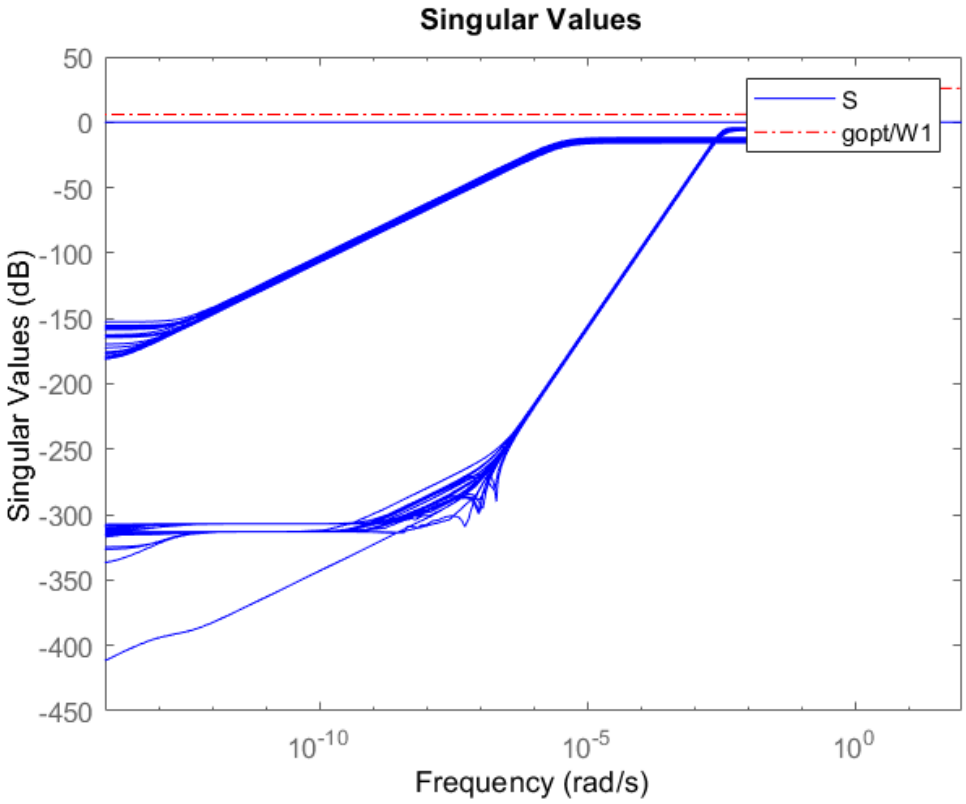


Figura 5.1: Funzione peso $1/WP$ confrontata con il valore singolare massimo della funzione S. Risulta che $\bar{\sigma}(S(j\omega) \leq \frac{1}{WP}$

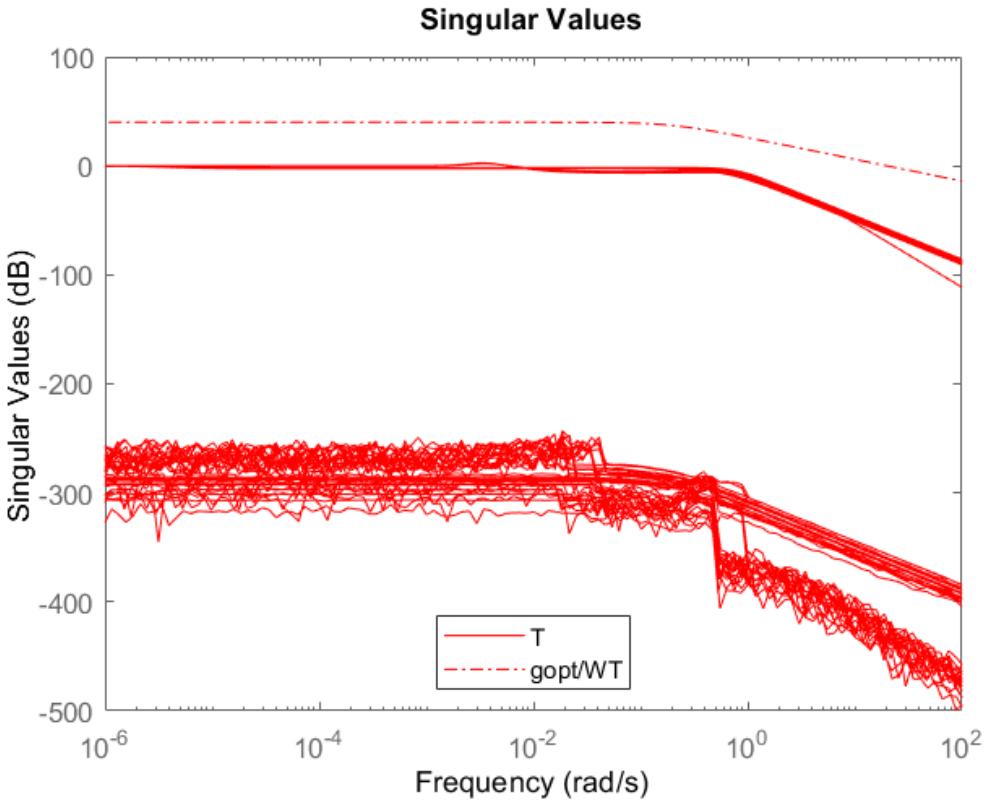


Figura 5.2: Funzione peso $1/WT$ confrontata con il valore singolare massimo della funzione T. Risulta che $\bar{\sigma}(T(j\omega) \leq \frac{1}{WT}$

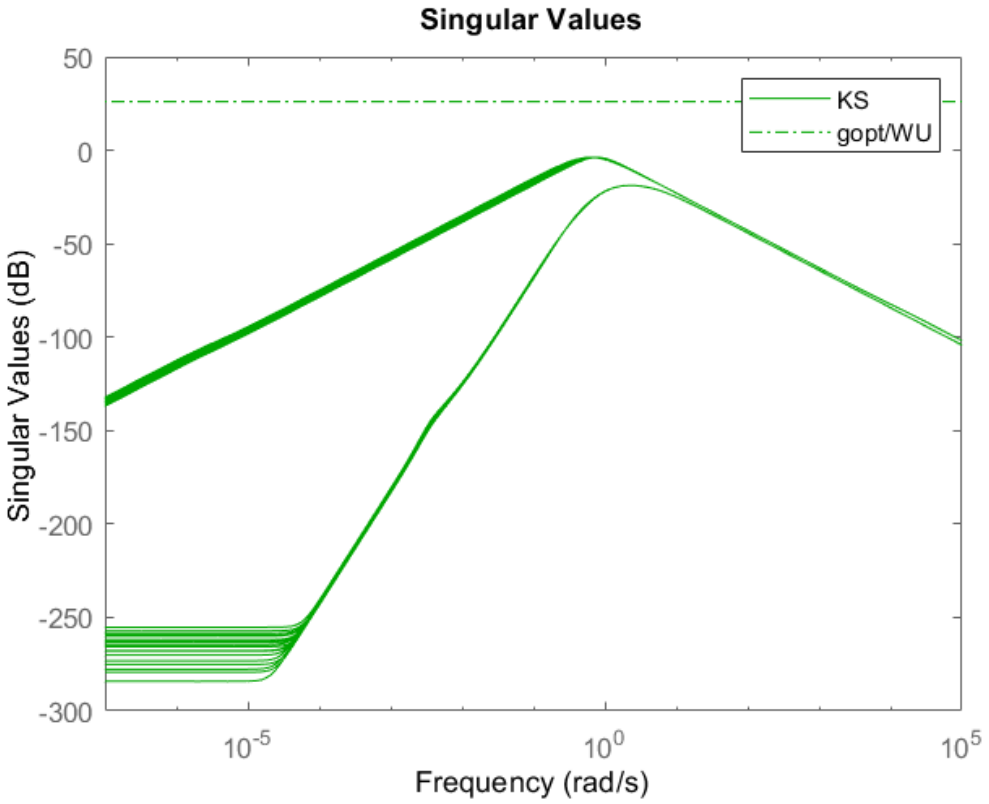


Figura 5.3: Funzione peso $1/WU$ confrontata con il valore singolare massimo della funzione KS. Risulta che $\bar{\sigma}(K(j\omega)S(j\omega) \leq \frac{1}{WU}$

Qui è stato utilizzato il comando **lftdata** in tale modo:

```
[P, Delta, blk] = lftdata(P_i); % estrae la matrice certa P
 e la matrice delle incertezze Delta

```

e sono state ottenute una Delta di dimensione 9x9 di cui **blk** indica la struttura (in questo caso diagonale) di Δ_I e la matrice P della forma di Doyle certa.

5.0.4 $\mu - analysis$

Per controllare le proprietà di robustezza del sistema controllato è necessario partizionare la matrice P certa nei sottoblocchi e ricavare la N tramite la linear fractional trasformation di P,K e ottenendo poi la risposta in frequenza nel modo seguente

```
 N = lft(P,K);   % funzione di trasferimento tra w e z
Nf = frd(N,omega);  % risposta in frequenza di N

```

Della matrice N stessa è stato necessario innanzitutto controlloare la stabilita nominale, controllando che N fosse internamente stabile. Gli autovalori di N sono proprio:

```
  1.0e+04 *
 -0.0000 + 0.0000i
 -0.0000 + 0.0000i
 -0.0000 + 0.0000i
 -0.0000 + 0.0000i
 -1.0000 + 0.0000i
 -1.0000 + 0.0000i
 -1.0000 + 0.0000i
 -1.0000 + 0.0000i
 -1.1334 + 0.0000i
 -1.0000 + 0.0000i
 -1.0000 + 0.0000i
 -1.0000 + 0.0000i
 -0.0004 + 0.0000i
 -0.0001 + 0.0001i
 -0.0001 - 0.0001i
 -0.0000 + 0.0000i
 -0.0000 - 0.0000i
 -0.0001 + 0.0000i
 -0.0001 - 0.0000i
 -0.0000 + 0.0000i
 -0.0000 - 0.0000i
 -0.0000 + 0.0000i
 -0.0000 + 0.0000i
```

```
  -0.0000 + 0.0000i
  -0.0000 + 0.0000i
  -0.0000 + 0.0000i
```

e dunque tutti negativi.
Dopo di che sono stati selezionati i diversi sottoblocchi di N per controllare le diverse condizioni su:

- RS, calcolando il $\mu_{\Delta}(N_{11}) < 1$, quindi abbiamo selezionato il blocco N_{11} di N;
- NP, calcolando il $\mu_{\Delta_P}(N_{22}) < 1$, quindi abbiamo selezionato il blocco N_{22} di N che ha la stessa dimensione di F^T dove F=lft(delta, N) con delta $= \hat{\Delta} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta_P \end{bmatrix}$ e nel nostro caso essendo N 19x13, F^T avrà dimensione 4x10.
- RP che si dimostra con $\mu_{\hat{\Delta}}(N) < 1$.

```
**********************************************************************
Stabilità robusta muRS: 0.000240
Prestazione nominale muNPinf: 0.365805
Robusta prestazione muRPinf: 0.402274
**********************************************************************
```

Figura 5.4: Risultati sulla stabilità e performance robuste ottenuti con il controllore mixsyn traminte mu analisi. Notiamo come sono state rispettate tutte le condizioni

I risultati mostrano come il sistema sia stabile e performante in modo robusto.

Tali risultati sono stati verificati anche con **robstab** e **robustperf**. Le figure 5.5 e 5.6 descrivono i risultati ottenuti con i rispettivi comandi.

```
System is robustly stable for the modeled uncertainty.
 -- It can tolerate up to 680% of the modeled uncertainty.
 -- No modeled uncertainty was found to cause instability.
 -- Sensitivity with respect to each uncertain element is:
      354% for rp_i. Increasing rp_i by 25% decreases the margin by 88.5%.
```

Figura 5.5: Risultati a schermo sulla robusta stabilità: vediamo come il sistema sia in grado di tollerare fino al 354% dell'incertezza sul parametro r_p prima di arrivare al margine.

```
'Assuming nominal UFRD system is stable...↵Uncertain system achieves performance robustness to modeled uncertainty.'
' -- The tradeoff of model uncertainty and system gain is balanced at a level of 261% of the modeled uncertainty.  '
' -- A model uncertainty of 261% can lead to input/output gain of 0.383 at 0.1 rad/seconds.                        '
' -- Sensitivity with respect to each uncertain element is:                                                       '
'      93% for Delta_P. Increasing Delta_P by 25% decreases the margin by 23.2%.                                  '
'      3% for rp_i. Increasing rp_i by 25% decreases the margin by 0.75%.                                         '
```

Figura 5.6: Risultati a schermo sulla robusta performance: vediamo come il sistema sia in grado di tollerare fino al 261% dell'incertezza sul parametro r_p. Vediamo come un'incertezza su tale parametro abbia molto più peso sulle performance rispetto alla stabilità. Infatti incrementatndo Δ_P del 25% riduce il margine di robustezza del 23.2% mentre per la stabilità decresce solo dello 0.75%.

5.0.5 Controllore Hinf con hinfsyn

Per il controllore H_∞ si è proceduto esattamente allo stesso modo di quello mixed-sensitivity, con gli stessi requisiti di performance.
I risultati ottenuti in questo caso sono molto simili a quelli ottenuti col controllore della sezione precedente. Qui si riporta la mu analisi che è stata fatta:

- per la NS sono stati osservati gli autovalori di N:

```
1      1.0e+04 *
2
3    -0.0000 + 0.0000i
4    -0.0000 + 0.0000i
5    -0.0000 + 0.0000i
6    -0.0000 + 0.0000i
7    -1.0000 + 0.0000i
8    -1.0000 + 0.0000i
9    -1.0000 + 0.0000i
```

```
   -1.0000 + 0.0000i
   -1.1334 + 0.0000i
   -0.0004 + 0.0000i
   -0.0001 + 0.0001i
   -0.0001 - 0.0001i
   -0.0000 + 0.0000i
   -0.0000 - 0.0000i
   -0.0000 + 0.0000i
   -0.0000 - 0.0000i
   -0.0000 + 0.0000i
   -1.0000 + 0.0000i
   -1.0000 + 0.0000i
   -1.0000 + 0.0000i
   -0.0001 + 0.0000i
   -0.0001 - 0.0000i
   -0.0000 + 0.0000i
   -0.0000 + 0.0000i
   -0.0000 + 0.0000i
   -0.0000 + 0.0000i
```

```
************************************************************
Stabilità robusta muRS: 0.000231
Prestazione nominale muNPinf: 0.365805
Robusta prestazione muRPinf: 0.402410
************************************************************
```

Figura 5.7: Risultati sulla stabilità e performance robuste ottenuti con il controllore hinf traminte mu analisi. Notiamo come sono state rispettate tutte le condizioni

I risultati mostrano come il sistema sia stabile e performante in modo robusto. Tali risultati sono stati verificati anche con **robstab** e **robustperf**. Le figure 5.8 e 5.9 descrivono i risultati ottenuti con i rispettivi comandi.

```
System is robustly stable for the modeled uncertainty.
 -- It can tolerate up to 680% of the modeled uncertainty.
 -- No modeled uncertainty was found to cause instability.
 -- Sensitivity with respect to each uncertain element is:
      357% for rp_i. Increasing rp_i by 25% decreases the margin by 89.2%.
```

Figura 5.8: Risultati a schermo sulla robusta stabilità: vediamo come il sistema sia in grado di tollerare fino al 354% dell'incertezza sul parametro r_p prima di arrivare al margine.

A differenza dei risultati ottenuti col solo controllore Hinf del capitolo 3, in questo caso l'incertezza è stata gestita in maniera ottimale, molto probabilmente in

```
'Assuming nominal UFRD system is stable...↵Uncertain system achieves performance robustness to modeled uncertainty.'
' -- The tradeoff of model uncertainty and system gain is balanced at a level of 261% of the modeled uncertainty.  '
' -- A model uncertainty of 261% can lead to input/output gain of 0.383 at 0.1 rad/seconds.                        '
' -- Sensitivity with respect to each uncertain element is:                                                       '
'      93% for Delta_P. Increasing Delta_P by 25% decreases the margin by 23.2%.                                  '
'      3% for rp_i. Increasing rp_i by 25% decreases the margin by 0.75%.                                         '
```

Figura 5.9: Risultati a schermo sulla robusta performance: vediamo come il sistema sia in grado di tollerare fino al 261% dell'incertezza sul parametro r_p. Vediamo come un'incertezza su tale parametro abbia molto più peso sulle performance rispetto alla stabilità. Infatti incrementatndo Δ_P del 25% riduce il margine di robustezza del 23.2% mentre per la stabilità decresce solo dello 0.75%.

questo caso il peso wi è stato scelto in maniera tale da soddisfare la condizione $|W_i(j\omega)| > l_i(\omega) = \max_{Gp}(|\frac{Gp-G}{G}|)$ su un range maggiore di frequenze. Per questo con lftdata abbiamo ottenuto risultati di robustezza migliori rispetto ad una gestione dell'incertezza come quella del capitolo 3.

5.0.6 Validazione simulink controllore Hinf

Mostriamo come con questo metodo anche il controllore hinf dia buoni risultati nella convergenza delle misure delle uscite. Lo schema Simulink utilizzato è quello della figura 3.13

Modello lineare

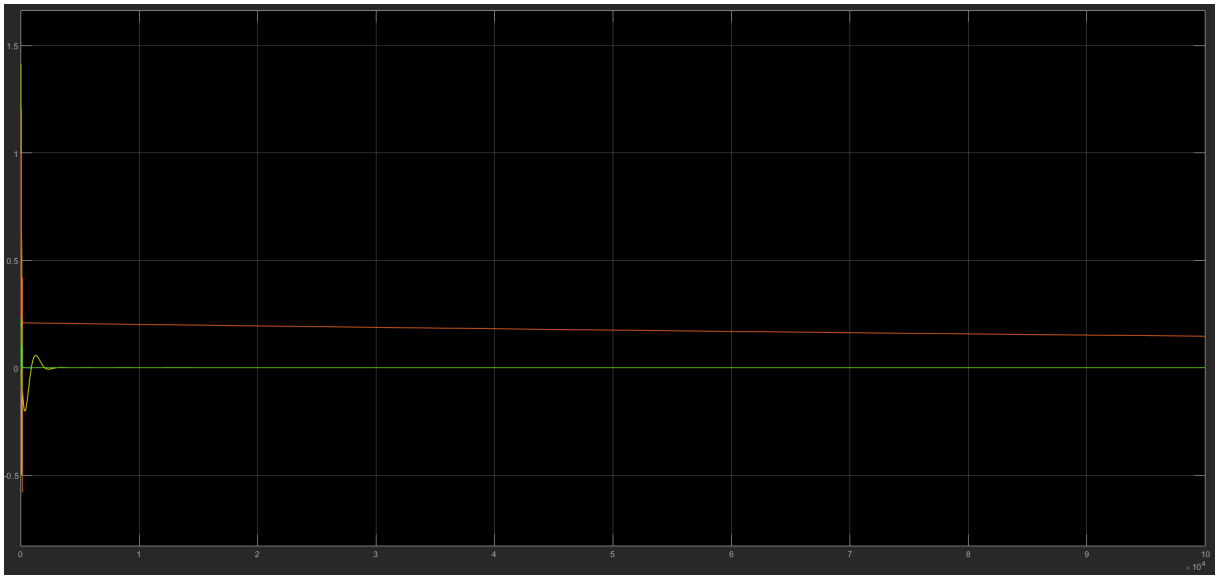


Figura 5.10: Si vede dalla simulazione come tutte le componenti convergono a zero anche se molto lentamente con condizioni iniziali anche lontane dal punto di equilibrio [1 0 1 0 0]

5.1 DK-iteration

Il comando *lftdata* è stato utilizzato anche per determinare un controllore ottimo tramite la procedura della DK-iteration.

Anche in questo caso, come nel caso descritto al capitolo 4, si eseguono una serie di iterazioni in cui prima si determina il controllore ottimo per il problema scalato

$$\min_K(\|DN(K)D^{-1}\|_\infty)$$

e successivamente si modifica la matrice peso D per minimizzare

$$\bar{\sigma}(DND^{-1}(j\omega)).$$

Anche in questo caso si costruisce il sistema incerto a partire dalle matrici incerte A_i e B_i. Al fine di confrontare i risultati della DK con quelli ottenuti da *hinfsyn* e da *mixsyn*, sono stati definite le stesse funzioni peso descritte in 5.0.1.

A questo punto è necessario riscrivere il sistema nella forma dell'impianto generalizzato. Per tale scopo si sfrutta il comando *sysic* specificando che in questo caso i blocchi presenti sono quelli relativi all'impianto incerto G_p e alle matrici peso W_P, W_U e W_T. Per cui, come già analizzato in 5.0.2, l'interconnessione viene creata senza specificare in modo esplicito la forma e la dimensione della matrice delle incertezze Δ_i.

Infatti, la rappresentazione delle incertezze del sistema viene stabilita in modo autonomo da MATLAB tramite l'uso del comando *lftdata* a cui viene fornito in ingresso direttamente l'impianto incerto P_i calcolato in precedenza.

In questo modo, il comando fornisce direttamente la matrice P e la matrice Δ_i che sarà in questo caso strutturata.

Il comando *musyn* viene applicato direttamente alla matrice incerta P_i per progettare un controllore che sia robusto anche alle possibili incertezze.

```
1    ...
2    nmeas = 4; nu = 2;
3    omega = logspace(-3,3,61);
4    opts = musynOptions('Display','full','MaxIter',20,'TolPerf
    ',0.001,'FrequencyGrid',omega)
5    [K_DK,CLPperf,info_mu] = musyn(P_i,nmeas,nu,opts);
6    ...
```

Il K-step e il D-step vengono ripetuti o finché non si superano le 100 iterazioni o finché non si ottengono risultati migliori con un margine di 0.001 sul picco di μ.

5.1.1 Studio delle prestazioni

La prima caratteristica da verificare è che il sistema sia nominalmente stabile quando viene chiuso in retroazione sul controllore determinato. Per tale scopo è possibilire controllare direttamente che gli autovalori del sistema a ciclo chiuso siano a parte reale negativa, come segue:

```
...
CL = feedback(sys*K_DK,eye(4));
eig(CL)
...
```

In alternativa è possibile effettuare lo stesso controllo sugli autovalori della matrice N ottenuta dalla *lower linear fractional transformation* tra P e il controllore K_DK:

```
...
N = lft(P,K_DK);
eig(N)
...
```

Tutti gli autovalori ottenuti risultano stabili e quindi è possibile concludere che il sistema è nominalmente stabile.

A questo punto è possibile effettuare un'analisi di robusta stabilità tramite il comando *robstab* applicato direttamente alla matrice M ottenuta dalla *upper linear fractional transformation* tra N e Δ_i e rappresentante la funzione di trasferimento tra i segnali y_Δ e u_Δ.

I risultati ottenuti sono mostrati alla figura 5.11.

```
Points completed: 61/61
System is robustly stable for the modeled uncertainty.
 -- It can tolerate up to 589% of the modeled uncertainty.
 -- No modeled uncertainty was found to cause instability.
 -- Sensitivity with respect to each uncertain element is:
      99% for rp_i. Increasing rp_i by 25% decreases the margin by 24.8%.
```

Figura 5.11: Risultati ottenuti dall'analisi di robusta stabilità con il comando *robstab*.

E' possibile concludere quindi che il sistema a ciclo chiuso risulta robustamente stabile per ogni incertezza consentita.

Per quanto riguarda, invece, la robusta prestazione, partiamo con l'analizzare i risultati ottenuti direttamente dal comando *musyn* e mostrati in figura 5.12.

Il picco minimo per μ ottenuto è pari a 0.669, per cui il sistema risulta essere robustamente performante e si può aumentare l'incertezza del 150% prima di perdere il rispetto delle performance definite.

Tale risultato viene confermato tramite l'uso del comando *robustperf* applicato alla matrice F ottenuta dalla *upper linear fractional transformation* tra N e Δ, dove in questo caso per Δ si intende la matrice diagonale a blocchi $\Delta = \begin{bmatrix} \Delta_i & 0 \\ 0 & \Delta_p \end{bmatrix}$.

Il risultato ottenuto è mostrato in figura 5.13.

D-K ITERATION SUMMARY:

	Robust performance		Fit order
Iter	K Step	Peak MU	D Fit	D
1	538.6	4.434	4.913	28
2	1.274	1.04	1.062	58
3	1.004	0.9246	0.9349	262
4	0.7504	0.7447	0.8566	256
5	0.6786	0.675	0.73	134
6	0.6932	0.6871	0.7295	188
7	0.6713	0.6693	0.7148	190
8	0.7079	0.7025	0.7334	152
9	0.6745	0.6698	0.7313	192

Best achieved robust performance: 0.669

Figura 5.12: Risultati forniti dal comando *musyn* per il calcolo del controllore tramite DK-iteration.

```
'Assuming nominal UFRD system is stable...↵Uncertain system achieves performance robustness to modeled uncertainty.'
' -- The tradeoff of model uncertainty and system gain is balanced at a level of 150% of the modeled uncertainty.  '
' -- A model uncertainty of 150% can lead to input/output gain of 0.667 at 0.001 rad/seconds.                       '
' -- Sensitivity with respect to each uncertain element is:                                                        '
'      102% for Delta_P. Increasing Delta_P by 25% decreases the margin by 25.5%.                                  '
'      7% for rp_i. Increasing rp_i by 25% decreases the margin by 1.75%.                                          '
```

Figura 5.13: Risultati ottenuti dall'analisi di robusta prestazione con il comando *robuststab*.

5.1.2 Validazione in Simulink

Per validare l'efficacia del controllore DK ottenuto, è stato implementato il sistema a ciclo chiuso in SIMULINK.

Per quanto riguarda il modello linearizzato, lo schema utilizzato è quello mostrato in figura 5.14.

Il primo blocco state-space contiene la matrici A_DK, B_DK, C_DK e D_DK ricavate dal controllore K_DK tramite il comando *ssdata*. Nel secondo blocco state-space sono invece presenti le matrici del sistema linearizzato sulla traiettoria di equilibrio $y = 0$. Il riferimento applicato è nullo, mentre la retroazione è negativa.

I risultati ottenuti dalla simulazione di 100 s a partire da condizioni iniziali $X\_start = \left[5; 0.5; -2; 0.6; 0.4\right]$ sono mostrati alla figura 5.15.

E' possibile osservare che l'uscita del sistema converge velocemente a zero.

E' stato successivamente implementato lo schema SIMULINK sul sistema non lineare come mostrato in figura 5.16.

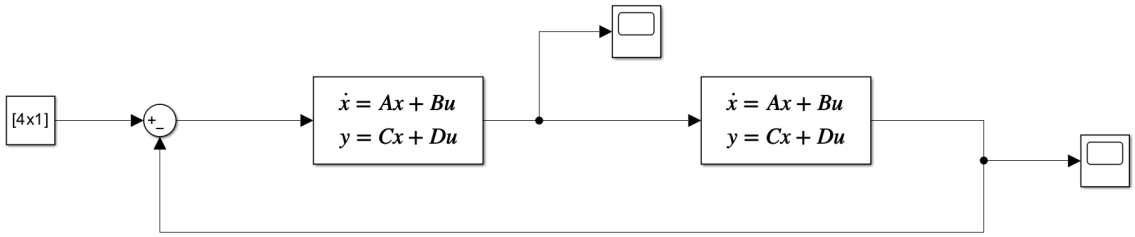


Figura 5.14: Schema Simulink per validazione controllore DK sul modello linearizzato.

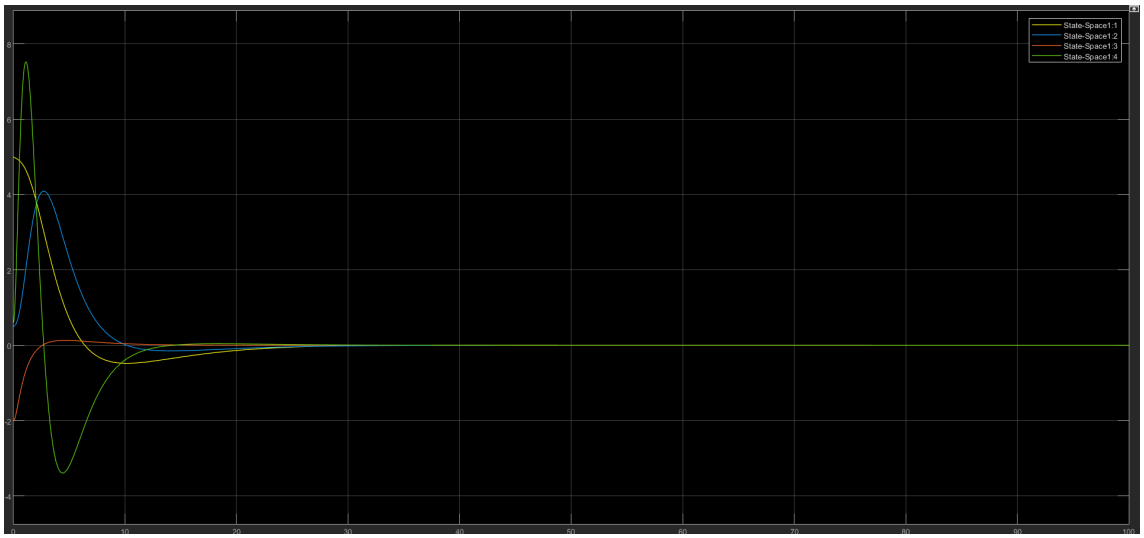


Figura 5.15: Andamento delle uscite del sistema linearizzato retroazionato sul controllore DK: y in giallo, θ in blu, $\dot{\phi}$ in rosso, ψ in verde.

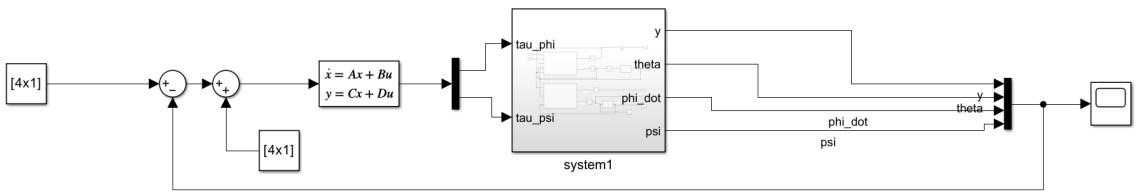


Figura 5.16: Schema Simulink per validazione controllore DK sul modello non lineare.

Il riferimento applicato è nullo e in ingresso al controllore si applica una traslazione delle variabili sul punto di equilibrio su cui è stata effettuata la linearizzazione.

Il risultato ottenuto dalla simulazione a partire dalle condizioni iniziali $X\_start = \begin{bmatrix}1; 1; -2; 1; 1\end{bmatrix}$ sono mostrati nella figura 5.17.

E' possibile notare che già dopo 20 s le uscite convergono all'equilibrio desiderato.

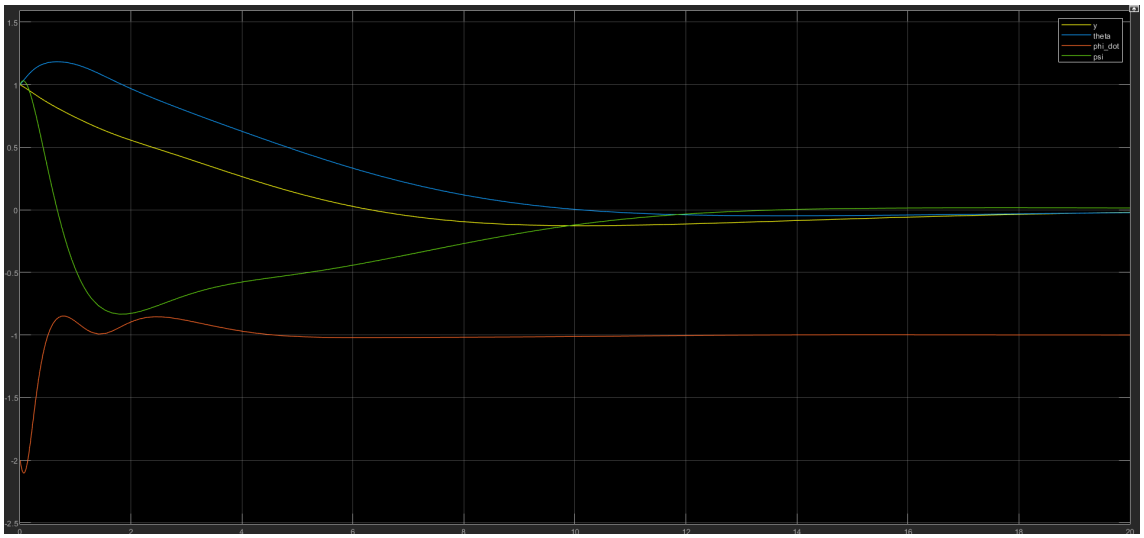


Figura 5.17: Andamento delle uscite del sistema non lineare retroazionato sul controllore DK: y in giallo, θ in blu, $\dot{\phi}$ in rosso, ψ in verde.

Capitolo 6

Conclusioni

Il progetto svolto riguardava l'implementazione di tre controllori in MATLAB/SIMULINK: il primo attraverso la tecnica LQG, il secondo attraverso la minimizzazione $\mathcal{H}_\infty$ e l'ultimo attraverso la DK-iteration.
Degli ultimi due sono state verificate le condizioni di robusta stabilità, performance nominale e performance robusta.
Inoltre sono stati poi sviluppati dei modelli SIMULINK per testare l'efficienza dei vari controllori, sia sul modello lineare che su quello non lineare.
Per quanto riguardo lo studio del sistema, si è potuto osservare come esso diventasse instabile una volta aggiunta la dinamica degli attuatori. Tramite la tecnica LQG si è dunque progettato un controllore in grado di stabilizzare tale dinamica.
Un'ulteriore prova è stata la realizzazione di un controllore LQG in presenza di un integratore. Aumentando la dinamica del sistema si è però osservato che essa perdeva di stabilizzabilità quindi non si è potuto procedere con la sintesi del controllore.
Il controllore LQG implementato è stato dunque testato su SIMULINK dove si può osservare che nel modello lineare tutte le variabili di stato convergono a zero mentre in quello non lineare (realizzato tramite ekf: Filtro di Kalman esteso) le variabili di stato convergono alla traiettoria di equilibrio $y = 0$
Il passo successivo è stata la creazione di un controllore $\mathcal{H}_\infty$ che minimizzasse appunto la norma infinito del vettore

$$N(k) = \begin{bmatrix} WpS \\ WtT \\ WuKS \end{bmatrix} \tag{6.1}$$

Tale problema è stato sviluppato in due diversi modi: Il primo prevede la costruzione a mano di un peso Wi sull'incertezza mentre il secondo utilizza una funzione di MATLAB: **lftdata** in grado di estrarre da una funzione incerta, il blocco di incertezza Delta e la matrice P certa.
In entrambi i casi sono stati sviluppati due controllori, uno con il comando MATLAB **mixsyn** e l'altro con il comando **Hinfsyn** e si è visto come in entrambi i casi vengano rispettate le condizioni di robusta stabilità, performance nominale e performance robusta.

Tutti i controllori generati sono stati testati in SIMULINK sia sul modello lineare che non lineare, ottenendo i risultati sperati.
Infine sono stati progettati dei controllori attraverso la DK_iteration in entrambi i modi descritti sopra. I risultati ottenuti sono stati buoni anche in questo caso, migliorando nelle maggior parte delle volte i risultati ottenuti attraverso le tecniche di controllo $\mathcal{H}_\infty$.

Capitolo 7

Appendice

Codice LQG

```
clc

load('dataset.mat');    % dataset loading
J = get_linearization();% matrices of the linearized system
s = tf('s');
A = J.A;
B = J.B;
C = J.C;
D = J.D;
G_act = blkdiag(G_tau_phi.NominalValue,G_tau_psi.NominalValue)
    ;
G = J.C*(s*eye(5)-J.A)^(-1)*J.B*G_act;    % transfer function

% Model dimensions:
p = size(J.C,1); % no. of outputs (y)
[n,m] = size(J.B); % no. of states and inputs (u)


% 1) Design state feedback regulator
Q = eye(n); % weight on integrated error
R = eye(m); % input weight
Kr = lqr(J.A,J.B,Q,R); % optimal state-feedback regulator

% 2) Design Kalman filter % donâĂŹt estimate integrator states
Bnoise = [0 0;0 0;1 0;0 0;0 1]; % process noise model (Gd)
W = log_vars.W; % process noise weight
V = log_vars.V; % measurement noise weight
Estss = ss(J.A,[J.B Bnoise],J.C,0);
[Kess, Ke] = kalman(Estss,W,V); % Kalman filter

K_lqg =[A-B*Kr-Ke*C, Ke;
        -Kr, zeros(2,4)];
```

```
K_LQG =ss(A-B*Kr-Ke*C,Ke,-Kr,zeros(2,4));
% Test simulation on simulink:
% For linear system: LQG_no_integrator.slx
% For non linear system: LQG_non_linear.slx
```

2

Codice LQG con attuatori

```
clc
%Inizializzazione
init

load('dataset.mat');    % dataset loading
J = get_linearization();% matrices of the linearized system
s = tf('s');
A1 = J.A;
B1 = J.B;
C1 = J.C;
D1 = J.D;
G_act = blkdiag(G_tau_phi.NominalValue,G_tau_psi.NominalValue)
    ;
G = C1*(s*eye(5)-A1)^(-1)*B1*G_act;    % transfer function
G = minreal(G);
[A, B, C, D] = ssdata(G)
% Model dimensions:
p = size(C,1); % no. of outputs (y)
[n,m] = size(B); % no. of states and inputs (u)
[A_act,B_act,C_act,D_act] = ssdata((ss(G_act)));
log_vars.A_act = A_act;
log_vars.B_act = B_act;
log_vars.C_act = C_act;
log_vars.D_act = D_act;
% 1) Design state feedback regulator
Q = eye(n); % weight on integrated error
R = eye(m); % input weight
Kr = lqr(A,B,Q,R); % optimal state-feedback regulator

% 2) Design Kalman filter % donâĂŹt estimate integrator states
Bnoise = [0 0;0 0;1 0;0 0;0 1;0 0; 0 0;0 0;0 0]; % process
    noise model (Gd)
W = log_vars.W; % process noise weight
V = log_vars.V; % measurement noise weight
Estss = ss(A,[B Bnoise],C,0);
[Kess, Ke] = kalman(Estss,W,V); % Kalman filter
Kr_sim = Kr(:,1:5);
save('dataset.mat','log_vars')
%K_lqg =[A-B*Kr-Ke*C, Ke;
%          -Kr, zeros(2,4)];
%K_LQG =ss(A-B*Kr-Ke*C,Ke,-Kr,zeros(2,4));
```

```
% Test simulation on simulink:
% For linear system: LQG_no_integrator.slx
% For non linear system: LQG_non_linear.slx
```

2

Codice della funzione di transizione dello stato

```
function x = function_f(x,tau)
    l = 0.4;
    L = 1;
    IPy = 0.15;
    IPz = 0.23;
    Mv = 15;
    mp = 0.5;
    ma = 0.5;
    IAy = 0.12;
    d = 0.6;
    IGz = 0.655;
    IAz = 0.8;
    rp = 0.3;
    ra = 0.3;
    a = Mv+mp+2*ma+2*IAy/(ra^2);
    b = 1/2*(Mv*l^2+mp*L^2+2*ma*d^2+IGz+IPz+2*IAz+2*IAy*(d/ra)
    ^2);

    dt = 0.01;  % tempo di campionamento per la funzione f
    discretizzata

    % definizione della f (funzione di transizione di stato)
    % Matrice della dinamica
    M=[a*rp^2*cos(x(4))^2+b*(rp/L)^2*sin(x(4))^2+IPy -IPz*(rp/
    L)*sin(x(4));-IPz*(rp/L)*sin(x(4)) IPz];
    N=[(rp^2*b/L^2-a*rp^2)*cos(x(4))*sin(x(4))*x(5)*x(3);-IPz*
    rp/L*cos(x(4))*x(5)*x(3)];
    q_ddot = inv(M)*([tau(1); tau(2)]-N);

    % funzione f discretizzata:
    x(1) = x(1) + [rp*x(3)*sin(x(2))*cos(x(4))]*dt;
    x(2) = x(2) + [-rp*x(3)*sin(x(4))/L]*dt;
    x(3) = x(3) + [q_ddot(1)]*dt;
    x(4) = x(4) + [x(5)]*dt;
    x(5) = x(5) + [q_ddot(2)]*dt;
end
```

2