Università di Pisa

Scuola di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Robotica e dell'Automazione (LM-25)

Relazione Progetto "Controllo di Sistemi Incerti".

Two-wheeled inverted pendulum mobile platform (Segway)

Students

Antonio N. Brigida 640484
Mirco Calzaretta 638275
Angelo Massara 641075

ANNO ACCADEMICO 2022/2023

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Definizione del problema

Il problema di controllo preso in esame per questo progetto, prevede di stabilizzare un Segway, veicolo modellato come un pendolo inverso poggiato su una base mobile che presenta due ruote, entrambe attuate, che possono muoversi contemporaneamente sia nella stessa direzione che in direzione opposta. a base mobile a due ruote si

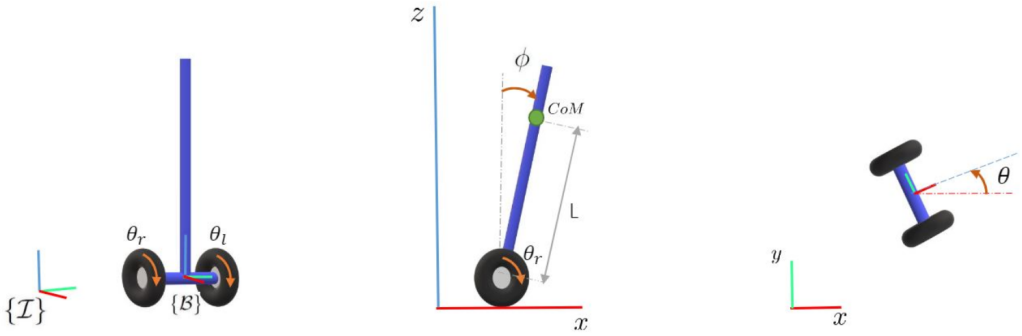

Figura 1.1: Modello del Segway preso in esame

muove su di una superficie piana e la sua configurazione rispetto ad un sistema di rifermento globale $\{I\}$ è $q = [x\ y\ \theta]^T$. Il pendolo inverso invece è incastrato solidalmente in corrispondenza della mezzeria dell'interasse delle ruote e quindi può ruotare solo attorno all'asse y_B del sistema di riferimento $\{B\}$, anch'esso posizionato nella mezzeria dell'interasse delle ruote. La posizione angolare del pendolo rispetto alla direzione verticale ϕ. Il nostro obiettivo è quello di creare un controllo sull'angolo ϕ e sulla sua derivata $\dot{\phi}$, e vogliamo che il Segway si mantenga nella posizione per cui $\phi = 0$ e, conseguentemente, $\dot{\phi} = 0$.

1.2 Analisi della Dinamica del Sistema

Sia m e r la massa delle ruote ed il loro raggio rispettivamente,M la massa del Segway, a la lunghezza del semiasse delle ruote, L la distanza del Centro di Massa (CoM) del robot dall'asse delle ruote, J_m il momento di inerzia dei motori ed infine g l'accelerazione di gravità. Indicando con

$$\begin{pmatrix} \tilde{M}_{\dot{v}} & 0 & \tilde{M}_{\dot{v}\ddot{\phi}} \\ 0 & \tilde{M}_{\ddot{\theta}} & 0 \\ \tilde{M}_{\ddot{\phi}\dot{v}} & 0 & \tilde{M}_{\ddot{\phi}} \end{pmatrix} \begin{pmatrix} \dot{v} \\ \ddot{\theta} \\ \ddot{\phi} \end{pmatrix} + \begin{pmatrix} \tilde{c}_v \\ \tilde{c}_{\dot{\theta}} \\ \tilde{c}_{\dot{\phi}} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \tilde{g}_{\dot{\phi}} \end{pmatrix} = \begin{pmatrix} \frac{1}{r} & \frac{1}{r} \\ \frac{a}{r} & -\frac{a}{r} \\ -1 & -1 \end{pmatrix} \begin{pmatrix} \tau_{\theta_r} \\ \tau_{\theta_l} \end{pmatrix} \tag{1.1}$$

e ponendo $c_\alpha = cos(\alpha), s_\alpha = sin(\alpha)$, ricaviamo le seguenti equazioni:

$$\tilde{M}_{\dot{v}} = 2 \cdot (\frac{3}{2} \cdot m + \frac{1}{2} \cdot M) \tag{1.2}$$

$$\tilde{M}_{\ddot{\theta}} = 2 \cdot (\frac{3}{2} \cdot ma^2 + \frac{1}{6} \cdot Ma^2 + \frac{1}{2} \cdot ML^2 s_\phi^2) \tag{1.3}$$

$$\tilde{M}_{\ddot{\phi}} = 2 \cdot (\frac{1}{2} \cdot mr^2 + \frac{2}{3} \cdot ML^2) \tag{1.4}$$

$$\tilde{M}_{\ddot{\phi}\dot{v}} = \tilde{M}_{\dot{v}\ddot{\phi}} = -mr + MLc_\phi \tag{1.5}$$

$$\tilde{c}_v = -MLs\phi \cdot (\dot{\theta}^2 + \dot{\phi}^2) \tag{1.6}$$

$$\tilde{c}_{\dot{\theta}} = 2ML^2 s_\phi c_\phi \dot{\theta}\dot{\phi} + MLs_\phi v\dot{\theta} \tag{1.7}$$

$$\tilde{c}_{\dot{\phi}} = -ML^2 s_\phi c_\phi \dot{\theta}^2 \tag{1.8}$$

$$\tilde{g}_{\dot{\phi}} = -MgLs_\phi \tag{1.9}$$

Si assuma inoltre che le funzioni di trasferimento nominali degli attuatori siano ben approssimabili con un modello del primo ordine del tipo:

$$\tilde{G}_m = \frac{\tilde{K}_m}{\tilde{T}_m s + 1} \tag{1.10}$$

Vengono inoltre considerati i seguenti dati:

- $m = 1.22$;
- $r = 0.13$;
- $a = \frac{0.496}{2}$;
- $L = 0.2924$ con incertezza pari a $\pm 10\%$;
- $\tilde{K}_m = 1.08$ con incertezza pari a $\pm 10\%$;
- $\tilde{T}_m = 0.005$ con incertezza pari a $\pm 10\%$;
- $g = 9.81$.

Una volta che è stata definita la dinamica si è passati all'implementazione su Matlab.

1.3 Inizializzazione su Matlab

Dopo aver definito la dinamica del sistema (esprimendo in un primo momento le variabili in simbolico), è stato definito il vettore di stato derivato

$$\dot{\xi} = \begin{pmatrix} v \cdot c_\theta \\ v \cdot s_\theta \\ \dot{\theta} \\ \dot{\phi} \\ (I)^{-1} \cdot (Ing \cdot \tau^T) \\ -(I)^{-1}C \\ -(I)^{-1}G \end{pmatrix} \tag{1.11}$$

con

$$I = \begin{pmatrix} \tilde{M}_{\dot{v}} & 0 & \tilde{M}_{\dot{v}\ddot{\phi}} \\ 0 & \tilde{M}_{\ddot{\theta}} & 0 \\ \tilde{M}_{\ddot{\phi}\dot{v}} & 0 & \tilde{M}_{\ddot{\phi}} \end{pmatrix}$$

$$Ing = \begin{pmatrix} \frac{1}{r} & \frac{1}{r} \\ \frac{a}{r} & -\frac{a}{r} \\ -1 & -1 \end{pmatrix}$$

$$\tau = \begin{pmatrix} \tau_{\theta_r} \\ \tau_{\theta_l} \end{pmatrix}$$

mentre il vettore di stato risulta essere

$$\xi = \begin{pmatrix} x \\ y \\ \theta \\ \phi \\ v \\ \dot{\theta} \\ \dot{\phi} \end{pmatrix} \tag{1.12}$$

Definite poi le matrici jacobiane A e B, dal momento che le uscite di nostro interesse erano esattamente il quarto e il settimo elemento del vettore di stato ξ, si è scelta come matrice C

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

mentre la matrice D è scelta nulla. A questo punto, utilizzando il comando *ureal* di Matlab abbiamo introdotto i parametri incerti del nostro sistema e ricavato la matrice di trasferimento perturbata G_p (di dimensione 2x2) in forma di stato e la funzione di trasferimento degli attuatori: avendo la stessa funzione di trasferimento per entrambi gli attuatori, la funzione $\tilde{G}_m$ è inserita all'interno di una matrice di trasferimento diagonale G_τ. Note quindi G_p e G_τ, abbiamo ricavato la G complessiva del sistema perturbato semplicemente andando a moltiplicare queste due matrici di trasferimento. Lo schema che abbiamo considerato è con incertezza di tipo additivo, quindi siamo in una situazione come quella descritta dalla figura 1.2.

Sono quindi stati tracciati i diagrammi dei valori singolari degli impianti perturbati e, sulla base della funzione di trasferimento rappresentate dalla curva più

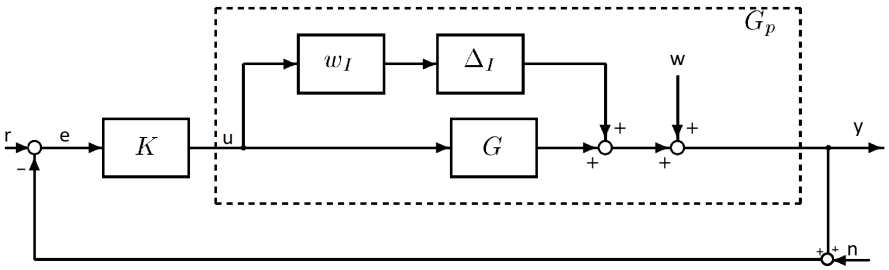

Figura 1.2: Schema Sistema con Incertezza Additiva

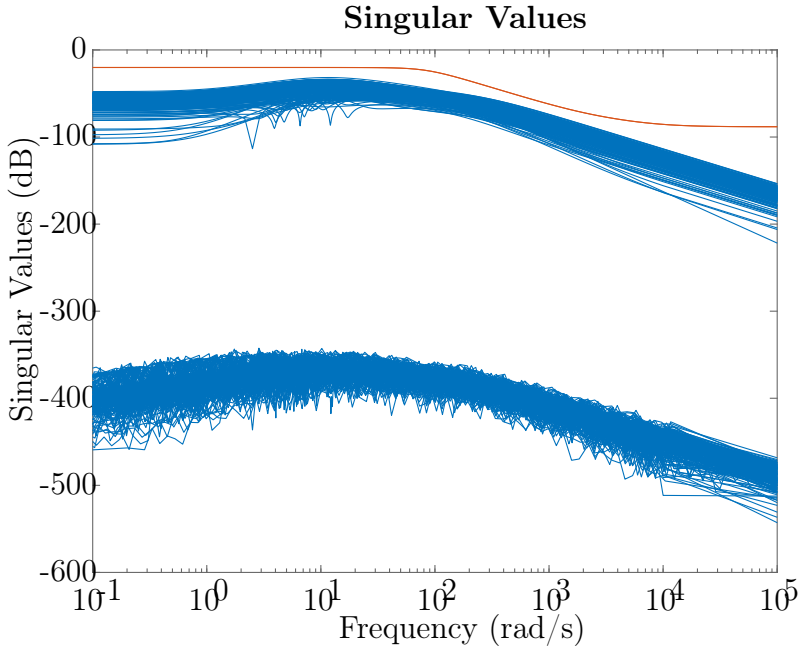

Figura 1.3: Funzione peso Wi

"alta", si è scelta la funzione peso delle incertezze mostrata in rosso in figura 1.3, la quale è stata poi inserita all'interno di una matrice diagonale 2x2.

Adesso abbiamo definito completamente l'impianto perturbato e l'ultima cosa che ci resta da fare è definire il modello dell'incertezza Δ, scelta anche in questo caso diagonale, di dimensione 2x2 e con i singoli elementi della diagonale uguali tra loro e ottenuti mediante il comando *ultidyn*. Una volta ricavato il modello nominale dell'impianto e degli attuatori, si può passare alla sintesi dei controllori.

1.3.1 Andamenti delle funzioni di trasferimento

Prima di passare alla vera e propria sintesi dei controllori, vengono presentati gli andamenti, in termini di daigrammi di Bode, delle funzioni di trasferimento di nostro interesse.

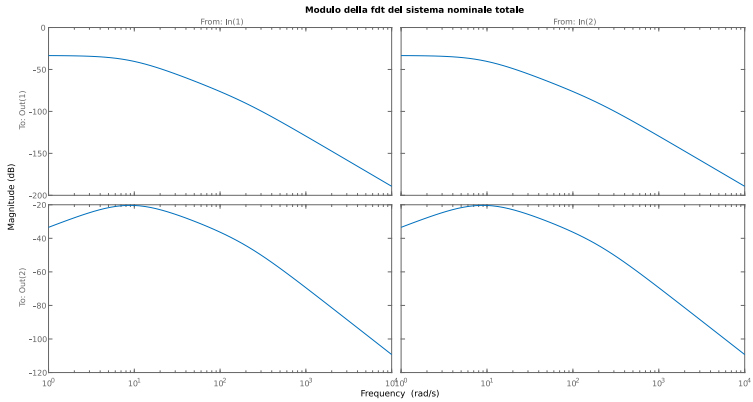

Figura 1.4: Funzione di trasferimento del Sistema Nominale

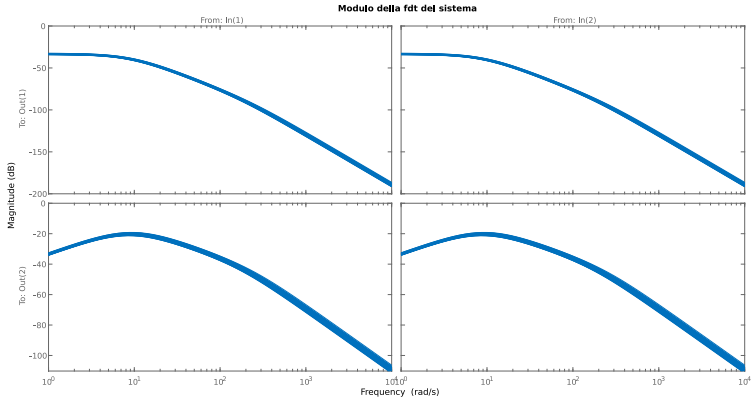

Figura 1.5: Funzione di Trasferimento del Sistema Perturbato

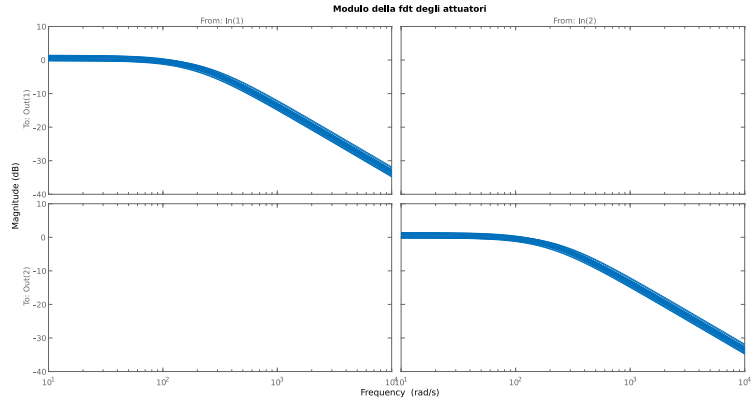

Figura 1.6: Funzione di trasferimento degli attuatori

Capitolo 2

LQG Control

L'obiettivo de l'LQG è il seguente: a partire dal modello dinamico descrivente un impianto, si vuole definire il controllo ottimo U(t) che minimizza un funzionale di costo J.

Il modello del sistema, in forma di stato, è il seguente:

$$\begin{cases} \dot{x} = Ax + Bu + w_d \\ y = Cx + Du + w_n \end{cases} \tag{2.1}$$

dove w_d e w_n sono rispettivamente rumori di ingresso e rumore di misura, i quali sono distribuiti secondo una distribuzione gaussiana di media nulla e di rispettiva varianza pari a W e V. Inoltre si assume che i due rumori siano tra di loro incorrelati.

Il funzionale di costo che si intende minimizzare è il seguente:

$$J = E\{\lim_{T\to\infty} \frac{1}{T} \int_0^T [x^T Q x + u^T R u] dt\} \tag{2.2}$$

con

$$Q \geq 0$$

$$R > 0$$

Al fine di risolvere il problema di controllo LQG, quello che si fa è risolvere dapprima un problema di controllo ottilo Lineare Quadratico (LQR) ottenendo come soluzione $u(t) = -K_r x(t)$ con K_r costante e indipendente da w_d e w_n.

Il passo successivo è trovare una stima ottima $\hat{x}$ dello stato x , in modo che sia minimizzata la varianza dell'errore di stima $\tilde{x} = x - \hat{x}$. La stima dello stato ottimo è data da un filtro di Kalman ed è indipendente dalle matrici Q e R . La soluzione richiesta al problema di controllo LQG viene quindi trovata sostituendo x con $\hat{x}$. Così facendo la soluzione è $u(t) = -K_r \hat{x}(t)$. In sintesi, il problema di controllo LQG può essere separato in due parti come illustrato nella figura 2.1

2.1 Controllore LQR

Quindi, come anticipato nel paragrafo precendente, si risolve dapprima un controllo LQR trovando il controllore K_r che minimizza il funzionale di costo J, e poi si trova la matrice K_f che minimizza la varianza dell'errore di stima $\tilde{x}$. La struttura del controllore LQG è la seguente:

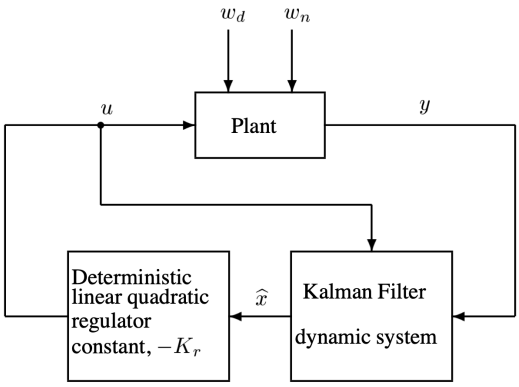

Figura 2.1: Costruzione della soluzione del controllo LQG

$$K_{LQG}(s) = \begin{pmatrix} A - BK_r & K_f \\ -K_r & 0 \end{pmatrix} \tag{2.3}$$

con $K_r = R^{-}1B^TX$ e $K_f = YC^TV^{-1}$ con $X \geq 0$ $Y \geq 0$ soluzioni dell'equazione algebrica di Riccati.

Può essere incluso anche un riferimento. Cosi facendo l'errore viene posto in ingresso ad un integratore. Lo schema a blocchi che include la presenza del riferimento e dell'integratore è di seguito riportato.

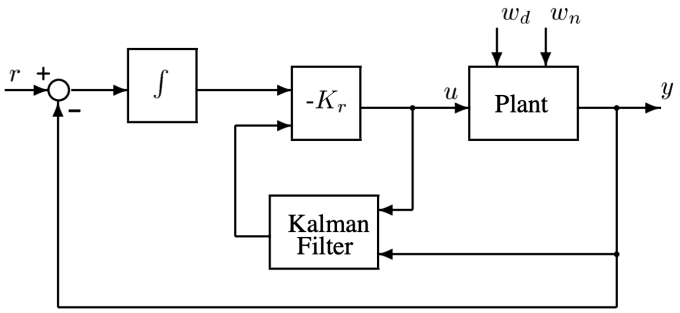

Figura 2.2: Controllo LQG con azione integrale

2.2 Proprietà di robustezza del Controllo LQG

Con un controllo LQG non sono garantiti i margini di stabilità, mentre se si sceglie un controllo LQR con R diagonale, la funzione si sensitività S sarà uguale a:

$$S = [I + K_r(sI - A)^{-1}B]^{-1} \tag{2.4}$$

Quest'ultima soddisfa la disuguaglianza di Kalman $\sigma(S(jw)) \leq 1\ \forall w$ e il sistema presenta un margine di guadagno ∞ e margine di fase di 60° in ogni canale di controllo di ingresso dell'impianto.

2.3 Controllo LQG applicato ad un Segway

Il progetto richiedeva l'applicazione di un controllo LQG con e senza azione ingrale al modello lineare e non lineare di un segway, il cui modello era fornito.

2.3.1 Inizializzazione controllo LQG

Vengono presentate di seguito le scelte delle matrici Q ed R utilizzate sia per il controllo LQR che per il filtro di Kalman

$$Q_{LQR} = \begin{pmatrix} \frac{1}{(\frac{20\pi}{180})^2} & 0 & 0 & 0 \\ 0 & \frac{1}{(\frac{20\pi}{180})^2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

$$R_{LQR} = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} \tag{2.6}$$

$$Q_k = \begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix} \tag{2.7}$$

$$R_k = \begin{pmatrix} 0.035^2 & 0 \\ 0 & 0.035^2 \end{pmatrix} \tag{2.8}$$

dove Q_k ed R_k rappresentano le covarianze associate a disturbo in ingresso e rumore di misura.

La funzione di trasferimento nominali degli attuatori é:

$$\frac{K_m}{T_m s + 1} \tag{2.9}$$

2.4 Controllo LQG per sistema non lineare

Lo schema a blocchi relativo al controllo LQG non lineare, realizzato in ambiente Simulink/Matlab è riportato in figura 2.3

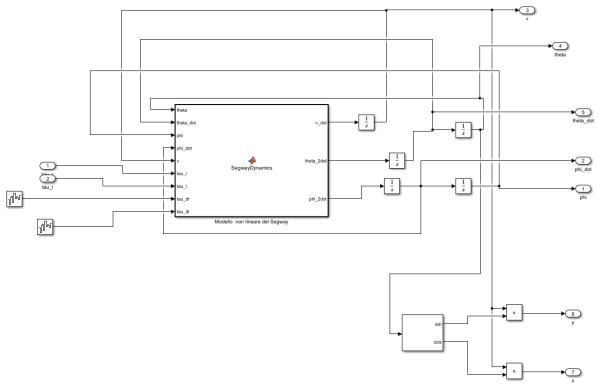

Figura 2.3: Controllo LQG senza integratore non lineare

dove il guadagno K è il guadagno prodotto dal controllo LQR, ricavato utilizzando il comando matlab 'lqr'.

$$k = -lqr(A_{tot}, B_{tot}, Q_r, Rr) \tag{2.10}$$

Matrici A_{tot} B_{tot}

Le matrici A_{tot}, B_{tot} sono le matrici descriventi il sistema nello spazio degli stati, includendo anche gli attuatori. Di seguito è riportato il procedimento utilizzato per ricavare le matrici.

Sia G_m funzione di trasferimento nominali degli attuatori é:

$$\frac{K_m}{T_m s + 1} \tag{2.11}$$

utilizzando il comando Matlab .NominalValue, ottiniamo la funzione di trasferimento nominale degli attuatori, ovvero la fdt con incertezze nulle. $Gm_{nom} = Gm.NominalValue$

Utilizzando il comando 'ss' si definisce il modello degli attuatori nello spazio degli stati

$$G_m ss = ss(Gm_{nom}, 'min') \tag{2.12}$$

da cui abbiamo ricavato le matrici A_{att} , B_{att}, C_{att}, D_{att} le quali sono le matrici del modello degli attuatori. Definendo

$$\begin{cases} \dot{x}_a = A_{att} x_a + B_a u \\ \dot{x} = Ax + Bu + w_d \\ y = Cx + Du_s \end{cases} \tag{2.13}$$

$$u_s = \begin{pmatrix} C_m & 0 \\ 0 & C_m \end{pmatrix} \tag{2.14}$$

Definiamo lo stato aumentato $\dot{x}$ nel seguente modo

$$\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{x}_a \end{pmatrix} \tag{2.15}$$

Esplicitano i calcoli ed effettuando le varie sostituzioni, definiamo le matrici A_{tot}, B_{tot}, C_{tot}, D_{tot} che sono di seguito riportate.

$$A_{tot} = \begin{pmatrix} A & BC_{att} \\ 0 & A_{att} \end{pmatrix} \tag{2.16}$$

$$B_{tot} = \begin{pmatrix} 0 \\ B_{att} \end{pmatrix} \tag{2.17}$$

$$B_{tot} = \begin{pmatrix} C & 0 \end{pmatrix} \tag{2.18}$$

$$D_{tot} = \begin{pmatrix} 0 \end{pmatrix} \tag{2.19}$$

2.5 Controllo LQG per sistema lineare

Lo schema a blocchi relativo al controllo LQG lineare, realizzato in ambiente Simulink/Matlab è riportato in figura 2.4

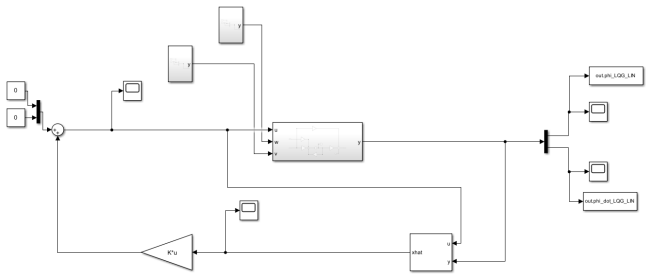

Figura 2.4: Controllo LQG senza integratore lineare

Il sistema lineare è stato ricavato tramite le matrici Jacobiane, valutate nel punto di equilibrio (origine).

2.6 Controllo LQG con azione integrale

Per quanto riguarda il controllo LQG con azione integrale, condizione necessaria e sufficiente affinchè sia possibile risolvere il controllo LQR è che il sistema sia almeno stabilizzabile.

Definendo $p = numero\ di\ integratori = 2$ e con $n = rang(R(A_{tot}, B_{tot}))$, sappiamo che, per avere un sistema stabilizzabile abbiamo bisogno che la matrice

$$A_{tot} = \begin{pmatrix} A_{tot} & B_{tot} \\ -C_{tot} & O_{2\times 2} \end{pmatrix} \tag{2.20}$$

abbia rango pari a p+n.

Poiche non è rispettata questa condizione, il controllo LQG con azione integrale non è stato trattato.

Capitolo 3

H_∞ control

Come primo passo andiamo a definire cosa intendiamo per norma H_∞ e successivamente verranno forniti tutti gli strumenti matematici necessari per la comprensione di questo tipo di problema di controllo.

3.1 Significato del termine H_∞

La norma H_∞ di una funzione di trasferimento scalare $f(s)$ è semplicemente il valore di picco di $|f(j\omega)|$ come funzione della frequenza:

$$||f(s)||_\infty = max_\omega |f(j\omega)| \tag{3.1}$$

Il simbolo ∞ deriva dal fatto che il massimo valore del modulo di una funzione al variare della frequenza può essere scritto come:

$$max_\omega |f(j\omega)| = \lim_{p\to+\infty} (\int_{-\infty}^{+\infty} |f(j\omega)|^p d\omega)^{1/p} \tag{3.2}$$

mentre il termine H sta per "Spazio di Hardy": indichiamo quindi con H_∞ il set di funzioni di trasferimento con norma ∞ limitata, che è semplicemente il set delle funzioni di trasferimento stabili e proprie. Andando invece a considerare un sistema dinamico, ne definiamo la norma H_∞ come

$$||G(s)||_\infty = max_\omega \bar{\sigma}(G(j\omega)) \tag{3.3}$$

In controllo robusto, è molto diffuso l'utilizzo della norma H_∞ in quanto è molto conveniente per rappresentare l'incertezza di modello non strutturata e soprattuto perchè è una nroma che rispetta la proprietà moltiplicativa:

$$||A(s)B(s)||_\infty \leq ||A(s)||_\infty \cdot ||B(s)||_\infty \tag{3.4}$$

Questa è una proprietà importante in quanto i sistemi di controllo con cui lavoriamo sono interconnessi e questa proprietà ci permette di andare a studiare singolarmente le proprietà dei sistemi che compongono il sistema interconnesso.

3.2 Approccio Mixed Sensitivity (Stacked S/T/KS)

Questo tipo di approccio consiste nell'andare a considerare delle funzioni peso che moltiplicano la funzioni di trasferimento a ciclo chiuso. Considerando lo schema in figura 1.2, definiamo:

- S(funzione di sensitività), che è la funzione di trasferimento da r a -e;
- T (funzione di sensitività complementare) che è la funzione di trasferimento da -n a y;
- KS, che è la funzione di trasferimento da r a u.

Cominciamo con l'analisi della funzione di sensitività: S è un buon indicatore di performance a ciclo chiuso sia per i sistemi SISO che per i sistemi MIMO e le specifiche che generalmente si richiedono su S sono:

1. Minima frequenza di banda ω_b^* (che è la frequenza alla quale $|S(j\omega|$ taglia 0.707 dal basso);
2. Massimo errore di inseguimento del riferimento alle frequenze selezionate;
3. Massimo errore di inseguimento del riferimento a regime (A);
4. Massimo valore di picco di S ($||S(j\omega)||_\infty \leq M$, con M generalmente scelto pari a 2).

tutte queste specifiche possono essere ottenute congiuntamente imponendo la condizione:

$$||w_p S||_\infty < 1 \tag{3.5}$$

dove w_p è una funzione peso scelta dal progettista, tipicamente della forma:

$$w_p(s) = \frac{(\frac{s}{M^{1/n}} + \omega_b^*)^{1/n}}{(s + \omega_b^* A^{1/n})^{1/n}} \tag{3.6}$$

con n che rappresenta l'ordine del sistema: se vogliamo migliorare le performance del sistema (e quindi avere una curva più ripida) aumentiamo l'ordine del sistema. A questo punto, il "problema" è che la condizione espressa dall'equazione 3.5 impone un lower bound sulla banda ma non un upper bound e soprattutto non ci permette di specificare il roll-off di $L(s) = G(s)K(s)$ oltre la banda. Quello che facciamo è quindi andare a considerare un peso W_u su KS, tipicamente della forma dell'Identità oppure:

$$W_u = \frac{s}{s + \omega_1} \tag{3.7}$$

dove ω_1 è la banda a ciclo chiuso, e W_t su T che presenta una forma diagonale in cui il singolo elemento è, in modulo, minore di 1 alle basse frequenze grande ad alta frequenza. Mettendo insieme queste condizioni imponiamo la specifica:

$$||N||_\infty = max_\omega \bar{\sigma}(N(j\omega)) < 1 \tag{3.8}$$

dove

$$N = \begin{pmatrix} w_p S \\ w_t T \\ w_u KS \end{pmatrix} \tag{3.9}$$

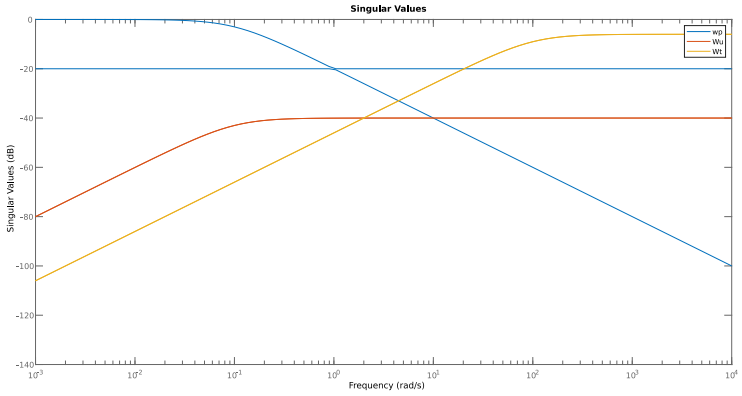

Figura 3.1: Valori Singolari delle funzioni di trasferimento a ciclo chiuso S,T,KS

3.3 Sintesi controllore in Matlab/Simulink

Il primo passo per la sintesi di questo controllore H_∞ è stato quindi una scelta di pesi. Dopo una serie di prove sperimentali, sono stati selezionati i pesi, per la funzione di sensitività S, pari a:

$$W_p = \begin{pmatrix} \frac{0.1}{s+0.1} & 0 \\ 0 & 0.1 \end{pmatrix} \tag{3.10}$$

Per KS è stata invece selezionata una matrice diagonale con gli elementi uguali:

$$W_u = \begin{pmatrix} \frac{0.01s}{s+0.1} & 0 \\ 0 & \frac{0.01s}{s+0.1} \end{pmatrix} \tag{3.11}$$

Infine, per la funzione di sensitività complementare T abbiamo:

$$W_t = \begin{pmatrix} \frac{0.5s}{s+100} & 0 \\ 0 & \frac{0.5s}{s+100} \end{pmatrix} \tag{3.12}$$

Le funzioni risultanti sono riportate nella figura 3.1 A questo punto ci siamo portati nella forma di Doyle, mostrata in figura 3.2.

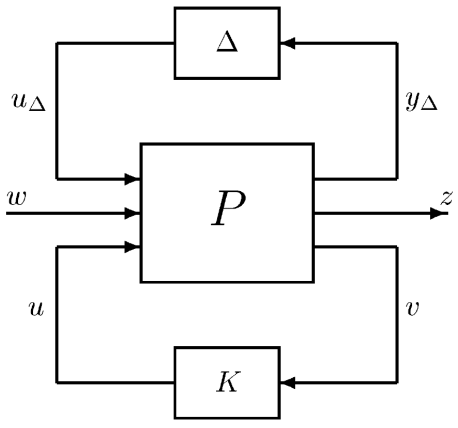

Figura 3.2: Sistema espresso in forma di Doyle

Successivamente, tramite il comando sysic abbiamo ricavato il sistema P.

Essendo Δ nota, abbiamo ricavato il controllore mediante il comando matlab *hinfsyn*. A tal proposito è importante sottolineare che non si è implementato il controllore con il comando *mixsyn* a causa di una serie di errori mostrati dal compilatore, quindi si è preferito includere le funzioni peso dentro P e sintetizzare un semplice controllore che minimizzasse la norma H_∞. Il controllore ottenuto è risultato essere molto efficiente in termini di minimizzazione di questa norma, restituendo infatti un valore $\gamma = 0.2563$ e un controllore K il cui modello in spazio di stato presentava 2 ingressi, 2 uscite e 9 stati. Possiamo infatti osservare che i valori singolari delle funzioni di trasferimento a ciclo chiuso stanno sotto i valori singolari dell'inversa delle funzioni peso e quindi che:

$$||N||_\infty < 1 \tag{3.13}$$

e questo risulta evidente dai successivi grafici:

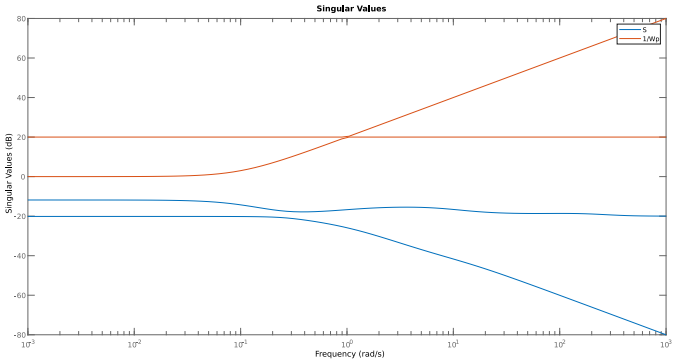

Figura 3.3: Funzione S e peso $1/W_p$

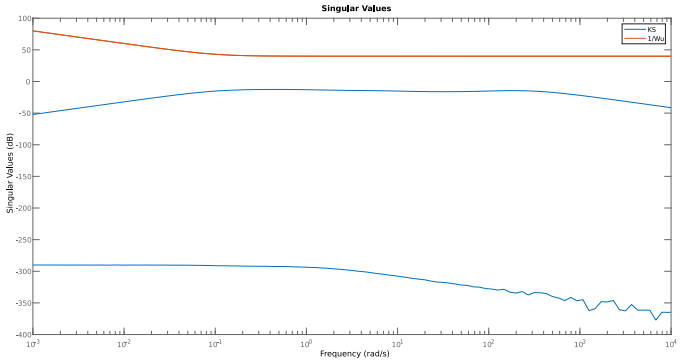

Figura 3.4: Funzione KS e peso $1/W_u$

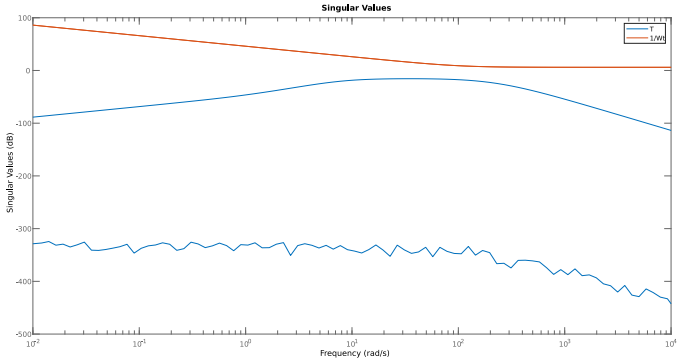

Figura 3.5: Funzione T e peso $1/W_t$

3.4 μ-analysisis

La μ-analysis viene utilizzata per effetuare un'analisi del sistema in termini di robustezza. Come primo passo, bisogna passare dalla forma di Doyle alla forma $N - \Delta$, in cui inglobiamo il controllore K all'interno del sistema P e questo viene fatto mediante una *Lower Linear Fraction Transformation*, ottenendo la struttura mostrata in figura 3.6

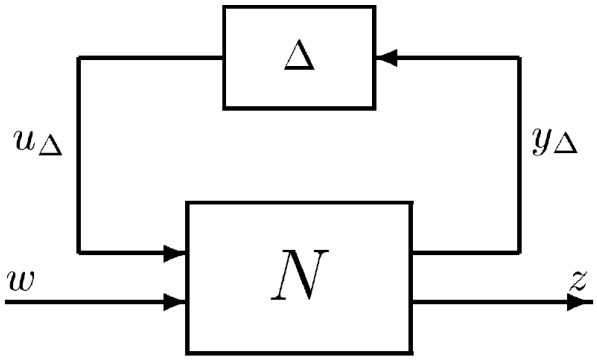

Figura 3.6: Struttura N-Δ

A partire da questa struttura, per valutare la funzione di trasferimento da w a z, usiamo Δ per richiudere l'anello superiore intorno a N, usando un *Upper Linear Fraction Transformation* A questo punto, si può cominciare con l'analisi del sistema, valutando prima stabilità e prestazioni nominali e poi stabilità e prestazioni robuste.

3.4.1 Stabilità Nominale

Per l'analisi della stabilità nominale, si è utilizzata la funzione Matlab *robuststab*: sebbene infatti sia pensata per sistemi che presentano incertezze, utilizzandola sul sistema nominale restituisce le informazioni circa la sua stabilità nominale. Nel nostro caso abbiamo ottenuto dei margini infiniti e quindi il sistema risultante è nominalmente stabile. Infatti la variabile *info_ Nmix* restituisce il seguente messaggio:

$'Uncertain\ system\ is\ not\ actually\ uncertain,\ hence\ these\ margins\ are\ infinite$

since the nominal USS system is stable.'

3.4.2 Prestazioni Nominali

Ragionamento analogo al precedente può essere fatto andando a considerare il comando Matlab *robustperf*: anche in questo caso i requisiti sono stati rispettati e quindi abbiamo ottenuto le prestazioni nominali. In uscita abbiamo una variabile *perfmarg_Nmix* che restituisce: Si ha un margine di prestazione $k_m = 3.9023$ che è quin-

```
perfmarg_N_mu =

  struct with fields:

          LowerBound: 1.2621
          UpperBound: 1.2621
    CriticalFrequency: 1.6170
```

Figura 3.7: Margini di Prestazione H_∞ Mixed Sensitivity

di pari ad un picco del valore singolare strutturato di $\mu_{\Delta_P}(N22) = 1/k_m = 0.2563$. Le prestazioni sono state quindi rispettate ampiamente.

3.4.3 Stabilità Robusta

Prima di eseguire il comando *robuststab* abbiamo dovuto calcolare nuovamente il sistema P in forma di Doyle ma questa volta considerando il modello del sistema perturbato (quindi quello che include le incertezze) e non quello nominale. I risultati in questo caso non sono stati eccellenti, in quanto il controllore non è risultato essere stabile in maniera robusta, ma è in grado di tollerare solamente il 55.1% delle incertezze modellate: successivamente è infatti presente una perturbazione destabilizzante che causa instabilità alla frequenza $1.66 \cdot 10^{-5}$ *rad/s* e inoltre abbiamo una sensitività del 97% su Δ, quindi incrementando Δ del 25% decresce il margine del 24.2%. Ci viene infatti restituito il seguente messaggio:

```
info_Nmix_p =
  6×88 char array
    'System is not robustly stable for the modeled uncertainty.                          '
    ' -- It can tolerate up to 55.1% of the modeled uncertainty.                        '
    ' -- There is a destabilizing perturbation amounting to 55.3% of the modeled uncertainty.'
    ' -- This perturbation causes an instability at the frequency 1.66e-05 rad/seconds.  '
    ' -- Sensitivity with respect to each uncertain element is:                         '
    '      97% for delta. Increasing delta by 25% decreases the margin by 24.2%.        '
```

Figura 3.8: Margini di Stabilità Robusta H_∞ Mixed Sensitivity

3.4.4 Prestazioni Robuste

Come possiamo aspettarci, il sistema non rispetta i requisiti di prestazioni robuste. Un'incertezza di modello del 48.6% porta il guadagno ingresso/uscita a 2.06 a 0.196 *rad/s* e abbiamo una sensitività dell'85% su Δ, quindi incrementando Δ del 25%, il margine decresce del 21.2%.

Capitolo 4

μ Synthesis e D-K iterations

Per presentare questa tecnica di sintesi e il nostro lavoro è necessario mostrare qualche strumento matematico utile alla comprensione

4.1 Valore singolare strutturato μ e scaling del valore singolare massimo

Nell'analisi e nella sintesi H_∞ presentata nel capitolo 3 abbiamo lavorato considerando il valore singolare massimo $\overline{\sigma}$ che produce condizioni *necessarie e sufficienti* per la stabilità e le prestazioni con un modello di incertezza Δ non strutturato.

Nel caso strutturato (matrice Δ diagonale complessa) il valore singolare $\overline{\sigma}$ non restituisce delle condizioni *necessarie* ma solo *sufficienti*. Questo fatto ci dà un indizio sulla possibile riformulazione del problema utilizzando una norma che sia *meno conservativa* rispetto il semplice valore singolare massimo, in modo tale da avere condizioni necessarie e sufficienti per ogni modello di incertezza dato (strutturato e non).

Un metodo per avere una condizione biunivoca è quello di "scalare" la matrice di trasferimento su cui calcoliamo il $\overline{\sigma}$ tramite una matrice che chiamiamo D. Infatti, la stabilità di un sistema è indipendente dal ridimensionamento fatto agli ingressi e alle uscite.

Per mostrare il procedimento facciamo riferimento al caso delle condizioni per la stabilità robusta, la condizione è:

$$\overline{\sigma}(M) < 1 \quad \forall\omega \tag{4.1}$$

In cui M è la matrice tale per cui

$$u_\Delta = M\, y_\Delta \tag{4.2}$$

Poniamo D pari a

$$D = diag(d_i I_i) \tag{4.3}$$

dove d_i sono scalari e I_i sono matrici identità delle stesse dimensioni dell'i-esimo blocco di incertezza Δ_i. Andiamo a ridimensionare gli input e gli output del sistema M ottenendo un nuovo sistema scalato DMD^{-1} e una nuova matrice di incertezza $D\Delta D^{-1}$. Le condizioni di stabilità robusta non cambiano:

$$RS \quad se \quad \overline{\sigma}(DMD^{-1}) < 1 \quad \forall\omega \tag{4.4}$$

Questo vale per ogni D con la struttura definita in 4.3. Possiamo quindi immaginare di trovare una condizione meno conservativa andando a trovare un D che minimizzi ad ogni valore di frequenza il valore singolare massimo:

$$RS \quad se \quad \min_{D(\omega)\in D} \overline{\sigma}(D(\omega)M(j\omega)D^{-1}(\omega)) < 1 \quad \forall\omega \tag{4.5}$$

Un altro metodo è quello di utilizzare il *valore singolare strutturato* μ definito come:

$$\mu(M) = \frac{1}{\min\{k_m \mid det(I - k_m M\Delta) = 0 \; per \; un \; \Delta \; strutturato, \overline{\sigma}(\Delta) < 1\}} \tag{4.6}$$

Questa "norma" è una versione generalizzata del valore singolare massimo ed è l'inverso del minimo guadagno sulle incertezze che rende il sistema instabile. Un $\mu = 1$ significa che, se consideriamo il 100% dell'incertezza modellata, esiste una particolare perturbazione che rende il sistema instabile. Se $\mu = 0.5$ allora il sistema è stabile anche considerando il 200% dell'incertezza modellata. Si nota anche che il valore singolare strutturato dipende fortemente dalla struttura definita per le incertezze e che nel caso non strutturato equivale al semplice valore singolare massimo. Sostituendo quindi nelle condizioni di stabilità e robustezza il $\overline{\sigma}$ con il valore μ otterremmo biunivocità in tutti i casi (strutturato e non)

Si ha inoltre che:

$$\mu_\Delta(M) \leq \min_{D(\omega)\in D} \overline{\sigma}(D(\omega)M(j\omega)D^{-1}(\omega)) < 1 \quad \forall\omega \tag{4.7}$$

La relazione 4.7 suggerisce un metodo per approssimare numericamente il valore singolare strutturato ed è alla base del metodo di sintesi chiamato *D-K iterations*.

4.2 Algoritmo di sintesi D-K iteration

L'idea dell'algoritmo è di trovare un controllore K_μ che minimizzi il valore di picco del valore singolare strutturato definito dalla relazione 4.7, formalmente:

$$K_\mu = \{K| \; \min_K(\min_D \parallel DN(K)D^{-1} \parallel_\infty) < 1\} \tag{4.8}$$

Con $N(K) = F_l(P, K)$ sistema a ciclo chiuso generalizzato che descrive le prestazioni richieste dal controllore rispetto le uscite da reiettare.

Questa minimizzazione viene fatta iterando in sequenza una minimizzazione della norma H_∞ e una ricerca della matrice di scaling *D(s)*. Per iniziare con le iterazioni bisogna porre una matrice D iniziale che sia stabile e che abbia una struttura appropriata. I passi dell'algoritmo sono i seguenti:

1. *K-step*: Si sintetizza un controllore H_∞ per il problema ridimensionato con $D(s)$ fissato.

2. *D-step*: Si cerca un $D(j\omega)$ che minimizzi as ogni frequenza il limite superiore del valore singolare strutturato con N fissato.

3. *Fit-step*: Si cerca un modello $D(s)$ stabile e a fase minima che si adatti ad ogni elemento della matrice $D(j\omega)$ trovata al punto precedente e si ritorna al passo 1.

Le iterazioni vengono fatte finché non sono soddisfatte le prestazioni volute o se la norma H_∞ non decresce.

4.3 μ - Synthesis su Matlab sul sistema "Segway"

Nella nostra sintesi abbiamo parametrizzato il sistema con l'approccio "generalizzato" tramite matrici P K e Δ (Fig. 3.2) definendo i vari segnali del sistema:

$$w = [r\ d\ n]^T \qquad z = [e_P\ u_P\ y_P]^T \tag{4.9}$$

$$v = r - y - n = r - Gu - u_\Delta - d - n \tag{4.10}$$

Con:
$e_P = W_P\ e = W_P(r - y) = W_P(r - Gu - u_\Delta - d)$,
$u_P = W_u\ u$
$y_P = W_T(Gu + u_\Delta)$

In questi segnali compaiono:

- u: Legge di controllo del controllore
- v: Segnale di ingresso del controllore
- z: Segnale "errore" da reiettare con l'azione di controllo
- r: Segnale di riferimento
- d: Disturbo in uscita
- n: Rumore di misura
- u_Δ: Uscita dal blocco di incertezza Δ
- y_Δ: Ingresso al blocco di incertezza Δ

Otteniamo:

$$\begin{pmatrix} y_\Delta \\ e_P \\ u_P \\ y_P \\ v \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & W_I \\ -W_P & W_P & -W_P & 0 & -W_PG \\ 0 & 0 & 0 & 0 & W_u \\ W_T & 0 & 0 & 0 & W_TG \\ -I & I & -I & -I & -G \end{pmatrix} \begin{pmatrix} u_\Delta \\ r \\ d \\ n \\ u \end{pmatrix} \tag{4.11}$$

Con W_I peso di parametrizzazione delle incertezze.
Otteniamo una matrice P così divisa:

$$P_{11} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -W_P & W_P & -W_P & 0 \\ 0 & 0 & 0 & 0 \\ W_T & 0 & 0 & 0 \end{pmatrix} \tag{4.12}$$

$$P_{12} = \begin{pmatrix} W_I \\ -W_P G \\ W_u \\ W_T G \end{pmatrix} \tag{4.13}$$

$$P_{21} = \begin{pmatrix} -I & I & -I & -I \end{pmatrix} \tag{4.14}$$

$$P_{22} = -G \tag{4.15}$$

Nel nostro progetto di sintesi abbiamo utilizzato le stesse matrici di peso utilizzato nella *Mixed Sensitivity* (Cap. 3.3) con qualche variazione sul peso W_P. Di fatto il peso sull'errore angolare è stato diminuito del 55%. Questa scelta è stata fatta perché il peso utilizzato precedentemente non dava risultati accettabili in termine di picco del valore singolare strutturato μ (Cosa non tenuta in considerazione nel controllo *Mixed Sensitivity*). Abbiamo quindi utilizzato i seguenti pesi:

$$W_P = \begin{pmatrix} \frac{0.055}{s+0.1} & 0 \\ 0 & 0.1 \end{pmatrix} \tag{4.16}$$

$$W_u = \begin{pmatrix} \frac{0.01s}{s+0.1} & 0 \\ 0 & \frac{0.01s}{s+0.1} \end{pmatrix} \tag{4.17}$$

$$W_T = \begin{pmatrix} \frac{0.5s}{s+100} & 0 \\ 0 & \frac{0.5s}{s+100} \end{pmatrix} \tag{4.18}$$

L'algoritmo di μ-Synthesis (DK-Iteration) non è stato personalmente implementato dal gruppo ma è stato utilizzato il comando *musyn* di Matlab. Questo comando permette di visualizzare a schermo i vari valori di picco del valore singolare strutturato $\mu(N)$ calcolati ad ogni iterazione e di impostare il numero di iterazioni massime da far eseguire all'algoritmo.

```
D-K ITERATION SUMMARY:
------------------------------------------------------------------
                        Robust performance              Fit order
------------------------------------------------------------------
  Iter          K Step        Peak MU        D Fit            D
    1           2.739          2.543         2.575           20
    2           1.368          1.317         1.332           28
    3           1.307          1.277         1.284           20
    4           1.268          1.234         1.245           24
    5           1.245          1.198         1.212           28
    6           1.208          1.157         1.175           28
    7           1.165          1.117         1.133           30
    8           1.127          1.084           1.1           30
    9           1.094          1.055         1.074           30
   10           1.065          1.032         1.049           26
   11           1.039           1.01         1.025           28
   12           1.014         0.9897         1.002           28
   13          0.9892         0.9663        0.9964           28
   14           0.967         0.9442         0.966           30
   15          0.9477          0.927        0.9678           30
   16          0.9261         0.9121        0.9391           26
   17          0.9191         0.9033        0.9222           26
   18          0.9056         0.8943        0.9013           28
   19          0.8964         0.8848        0.8913           26
   20          0.8896         0.8772        0.8867           28
   21          0.8827         0.8721        0.8785           26
   22          0.8773         0.8659        0.8726           28
   23          0.8721          0.861        0.8677           26
   24           0.867         0.8569        0.8634           26
   25          0.8626         0.8531        0.8596           26

Best achieved robust performance: 0.853

Computing bounds... points completed (of 47) ... 40
Computing peak...  Percent completed: 100/100
```

Figura 4.1: Report dell'algoritmo DK-Iteration

Il numero di iterazioni eseguite sono state 25 con un picco pari a $\max \mu(N) = 0.853$ (Fig 4.1)

4.4 μ-Analysis

Come nel caso del controllore *Mixed Sensitivity* dobbiamo utilizzare la *Linear Fraction Transformation* per chiudere l'anello di retroazione con il controllore K_μ e ricavare quindi la struttura $N - \Delta$ (Fig 3.6).

La μ-Analysis in questo caso è stata fatta considerando un unico modello creato con il comando *sysic*.

4.4.1 Stabilità Nominale

Anche in questo caso abbiamo utilizzato il comando *robuststab* che oltre alle informazioni nel caso robusto ci informa se il sistema nominale è stabile o meno. La variabile $info\_N\_mu$ restituisce il seguente messaggio:

'Uncertain system is not actually uncertain, hence these margins are infinite since the nominal USS system is stable.'

Il sistema nominale è quindi stabile.

4.4.2 Prestazioni Nominali

Per le prestazioni abbiamo pescato la sotto-matrice N_{22} dalla matrice N e ne abbiamo calcolato i margini di stabilità col comando *robustperf*. In uscita abbiamo la variabile $perfmarg\_N\_mu$ che ha la seguente struttura e valori:

```
perfmarg_N_mu =

  struct with fields:

          LowerBound: 1.2621
          UpperBound: 1.2621
    CriticalFrequency: 1.6170
```

Figura 4.2: Margini di prestazione nel controllo μ-Synthesis

Si ha un margine di prestazione $k_m = 1.2621$ che è quindi pari ad un picco del valore singolare strutturato di $\mu_{\Delta_P}(N_{22}) = \frac{1}{k_m} = 0.7923$. Le prestazioni sono state quindi rispettate ampiamente. Riportiamo di seguito l'andamento dei valori singolari in frequenza (Fig 4.3):

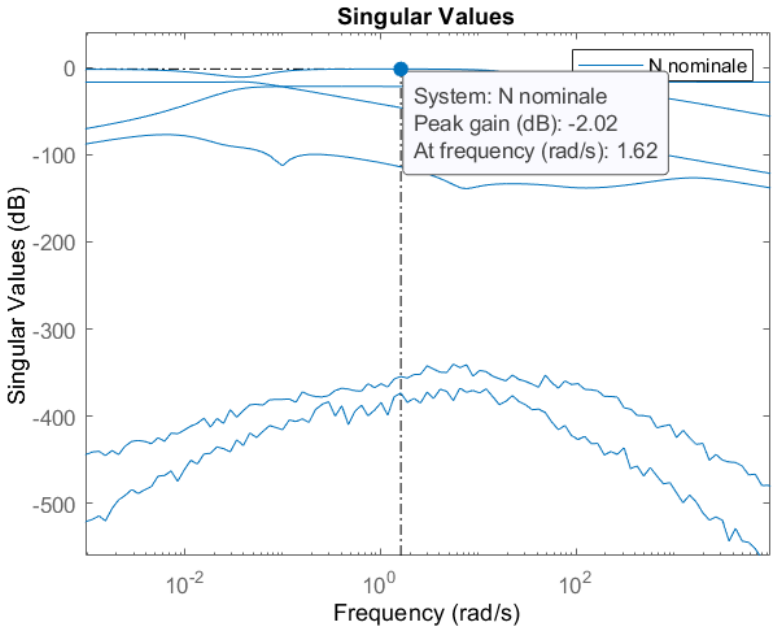

Figura 4.3: Valori singolari nella μ-Synthesis (caso nominale)

4.4.3 Stabilità Robusta

Nella stabilità robusta si è lavorato con il sistema $N_{tot} = F_u(N, \Delta)$ che rappresenta quindi la relazione tra la coppia $w-z$ chiudendo l'anello delle incertezze. Utilizzando *robuststab* abbiamo ottenuto:

```
info_N_mu_p =

  6×87 char array

    'System is robustly stable for the modeled uncertainty.                                '
    ' -- It can tolerate up to 803% of the modeled uncertainty.                           '
    ' -- There is a destabilizing perturbation amounting to 805% of the modeled uncertainty.'
    ' -- This perturbation causes an instability at the frequency 6.04e-06 rad/seconds.    '
    ' -- Sensitivity with respect to each uncertain element is:                           '
    '      66% for delta. Increasing delta by 25% decreases the margin by 16.5%.          '
```

Figura 4.4: Margini di stabilità robusta nel controllo μ-Synthesis

I margini di stabilità sono $k_m = 8.03$ ($\mu(M) = \frac{1}{8.0316} = 0.1245$). Ciò significa che il sistema rimane stabile fino ad un aumento di incertezza di circa 803%. Infatti esiste una incertezza che rende il sistema instabile dell'805% rispetto all'incertezza considerata, alla frequenza di $6.04 \times 10^{-6} rad/s$.

4.4.4 Prestazioni Robuste

Per quanto riguarda le prestazioni robuste il comando $robustperf$ su N_{tot} restituisce:

```
perfmarg_N_mu_p =

  struct with fields:

          LowerBound: 1.1723
          UpperBound: 1.1800
    CriticalFrequency: 6.0401e-06
```

Figura 4.5: Margini di prestazione robusta nel controllo μ-Synthesis

In questo caso abbiamo un $k_m = 1.1723$ cioè un valore singolare strutturato di $\mu_{\hat{\Delta}}(N_{tot}) = \frac{1}{1.1723} = 0.853$ (Come del resto visto nell'analisi del valore singolare strutturato fatta nella DK-Iteration in Fig 4.1). Il sistema rispetta le condizioni di prestazione robusta fino ad un massimo di incertezza del 118% rispetto a quella considerata nel problema iniziale, ad una frequenza di circa $6.04 \times 10^{-6} rad/s$. Mostriamo infine l'andamento del valore singolare strutturato con relativo picco.

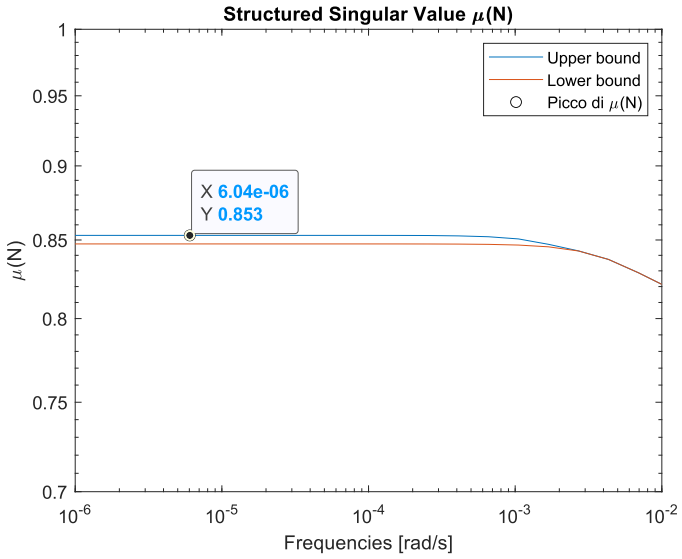

Figura 4.6: Valore singolare strutturato nella μ-Synthesis

Capitolo 5

Risultati Sperimentali

5.1 Confronto schemi Simulink

In questa sezione vengono presentati gli schemi simulink adottati per fare le simulazioni e il confronto dei risultati, presentato successivamente. Sono stati realizzati uno schema relativo al modello lineare e uno relativo al modello non lineare per ciascun controllore studiato. Una nota importante deve essere fatta sul controllore LQG: era stato richiesto di valutare sia il caso con integratore sia il caso senza integratore. Poichè non è stato possibile realizzare il caso con integratore, non è stato costruito alcuno schema simulink

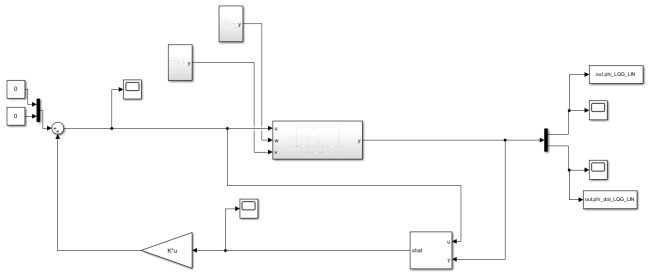

Figura 5.1: Schema Simulink LQG lineare senza integratore

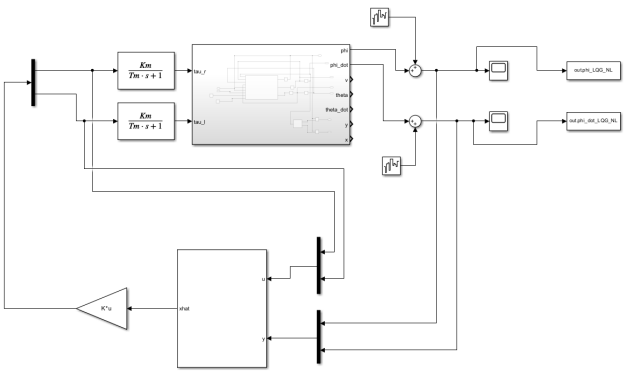

Figura 5.2: Schema Simulink LQG non lineare senza integratore

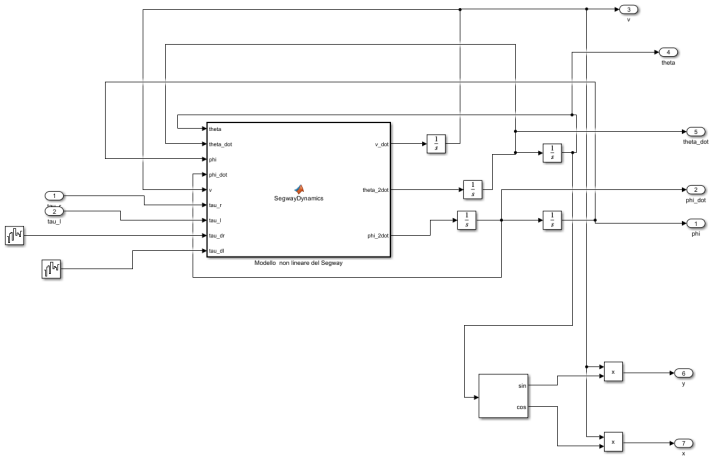

Figura 5.3: Subsystem LQG non lineare senza integratore

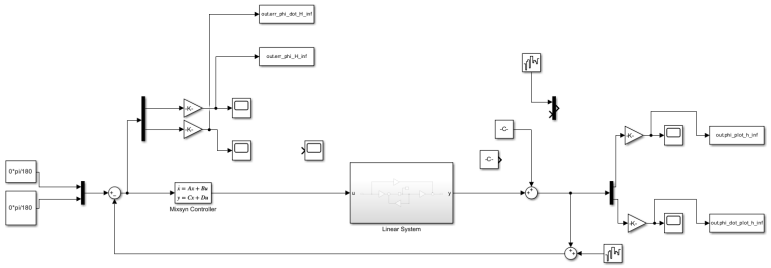

Figura 5.4: Schema Simulink H_∞ Mixed Sensitivity lineare

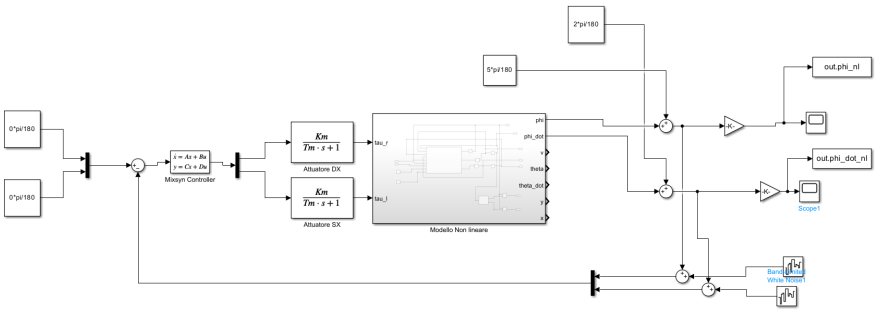

Figura 5.5: Schema Simulink H_∞ Mixed Sensitivity non lineare

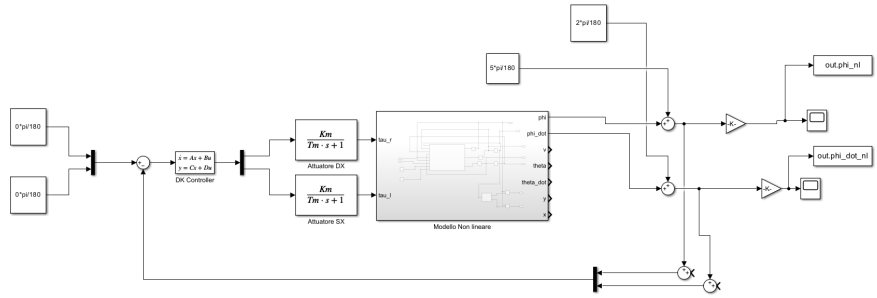

Figura 5.7: Schema Simulink DK non lineare

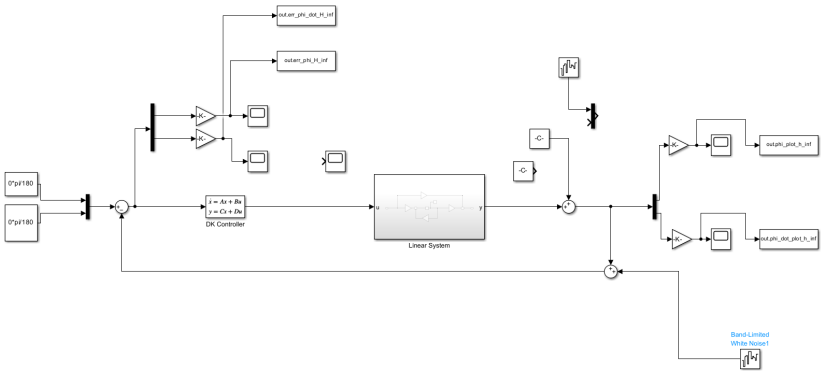

Figura 5.6: Schema Simulink DK lineare

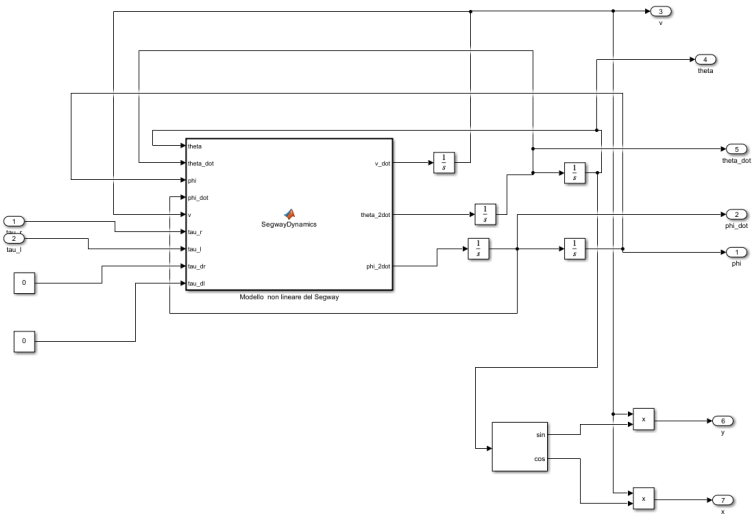

Figura 5.8: Subsystem DK e H_∞ lineare e non lineare

5.2 Confronto Performance

In questa sezione vengono invece presentati i risultati che sono stati ottenuti per ciascun controllore e messi a confronto, al fine di capire quale è risultato più adatto in termini di performance e quale in termini di robustezza. Per ciascun controllore ($Mixsyn$ e $\mu - syn$) è stato valutato sia il caso lineare, sia il caso non lineare e in ognuno di questi due casi abbiamo sottoposto il sistema a un rumore di misura bianco con deviazione standard pari a 0.57° (0.01 rad) e a un disturbo in uscita costante di $d = [d_\phi, d_{\dot{\phi}}] = [5°, 2°/\text{s}]$.

Cominciamo con l'analisi del controllore LQG. Nel sistema lineare il controllore LQG si comporta molto bene ma questo c'era da aspettarselo dato che il nostro era un problema di *regolazione*. L'angolo infatti rimane nell'intorno di $\phi = 0$ con dei picchi di 0.35° e la velocità angolare $\dot{\phi}$ ha dei picchi di circa 0.8°/s. Nel caso non lineare ovviamente abbiamo delle prestazioni peggiorate ma comunque molto buone con dei picchi di angolo di circa $\phi = 1.25°$ e di velocità angolare approssimativamente di $\dot{\phi} = 1.5°/\text{s}$.

Continuiamo con il controllore H_∞: i risultati ottenuti mostrano come le prestazioni ottenute siano molto soddisfacenti:nel caso lineare con rumore bianco l'oscillazione del pendolo inverso risulta infatti molto contenuta. Diverso è il discorso per quanto riguarda il caso non lineare dove, sebbene i risultati possano essere considerati accettabili, non sono eccellenti. Inoltre, per quanto detto nel capitolo 3, nonostante abbiamo ottenuto una norma infinito molto bassa, il controllore risultante non è robusto.

Resta infine il controllore ottenuto con la DK-iteration. Questo controllore, nonostante vada in crisi nel caso non lineare, è quello che ha prodotto i risultati migliori: siamo infatti riusciti ad avere prestazioni eccellenti nel caso lineare, con una norma infinito minore di 1, e siamo inoltre riusciti a garantire tutte le proprietà di robustezza (sia per la stabilità che per le performance).

Di seguito sono riportati tutti i grafici che mostrano gli andamenti di ϕ e $\dot{\phi}$ in tutti i casi precedentemente elencati.

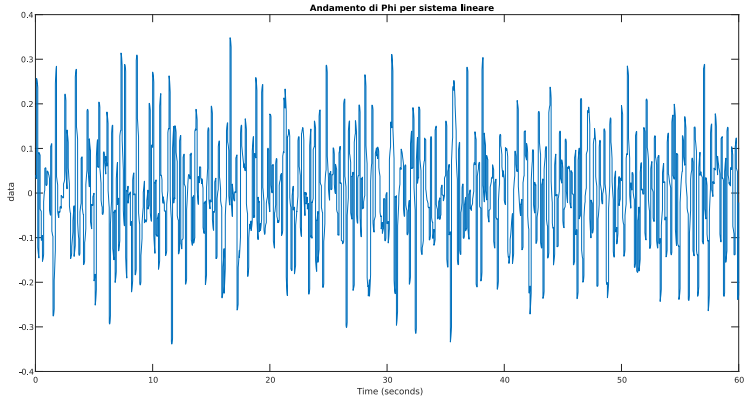

Figura 5.9: Andamento di ϕ caso LQG lineare senza integratore

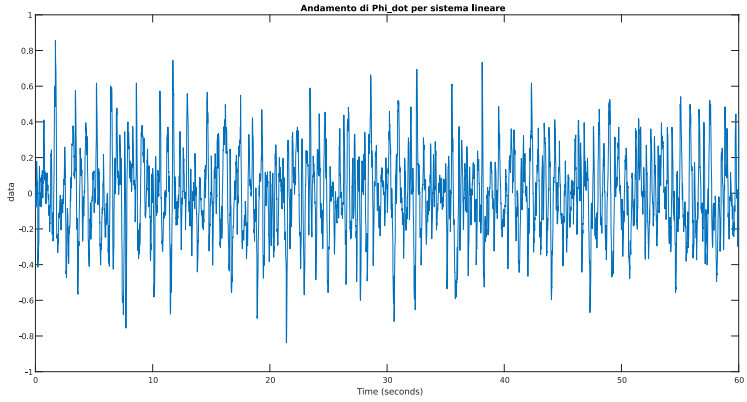

Figura 5.10: Andamento di $\dot{\phi}$ caso LQG lineare senza integratore

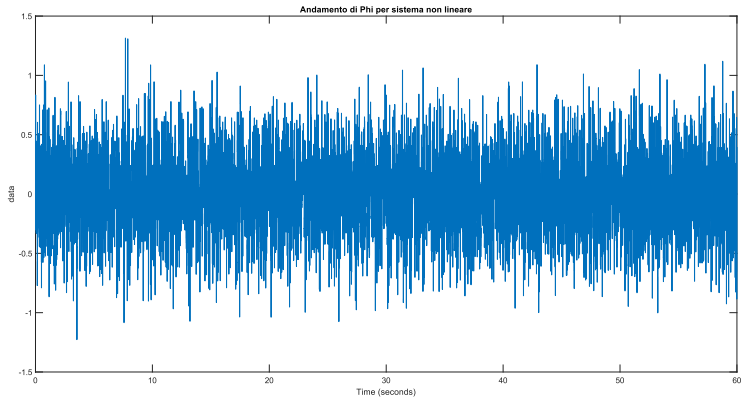

Figura 5.11: Andamento di ϕ caso LQG non lineare senza integratore

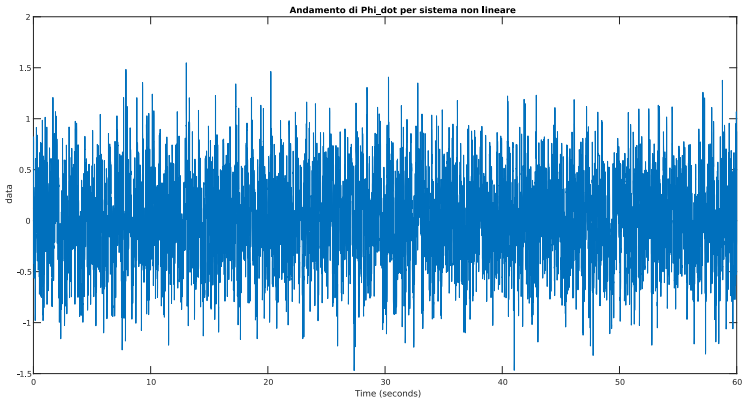

Figura 5.12: Andamento di $\dot{\phi}$ caso LQG non lineare senza integratore

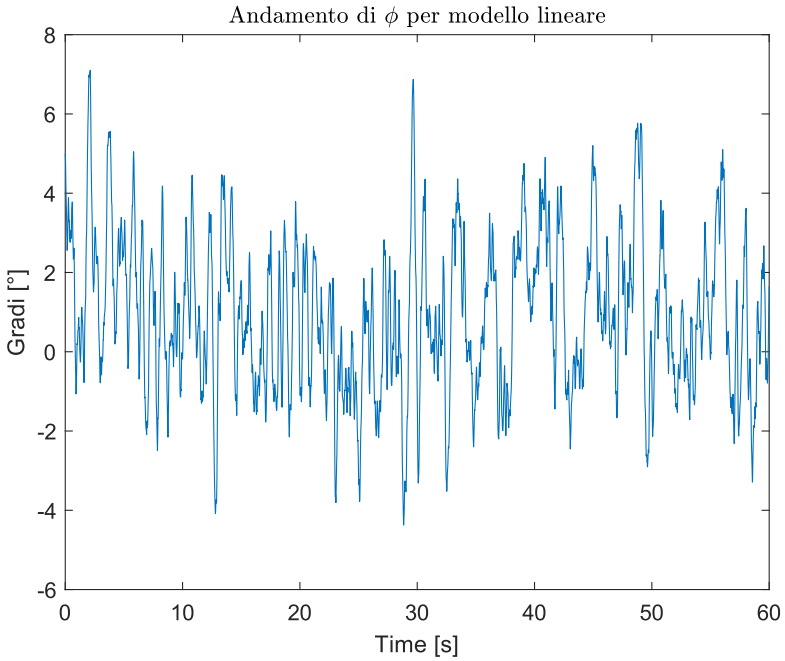

Figura 5.13: Andamento di ϕ caso Mixed Sensitivity lineare con disturbo di misura rumore bianco

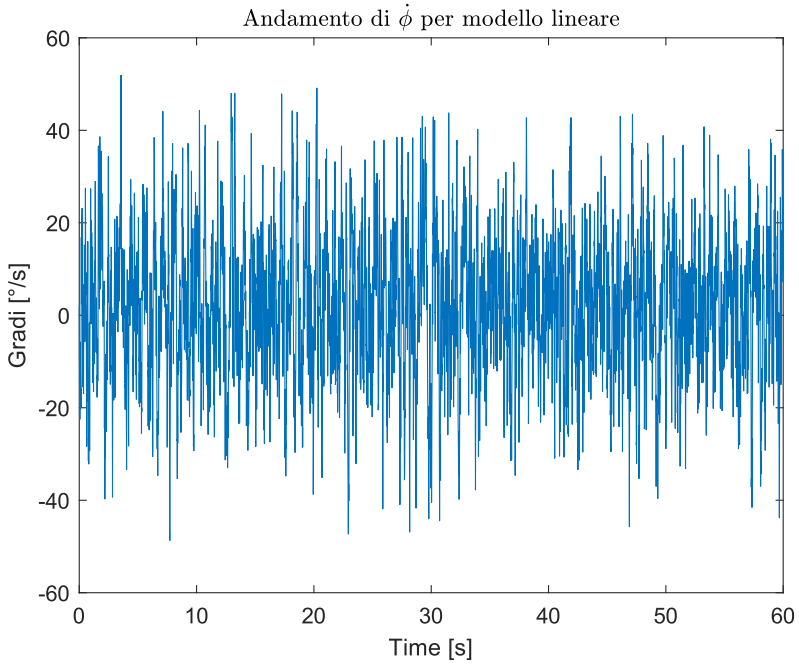

Figura 5.14: Andamento di $\dot{\phi}$ caso Mixed Sensitivity lineare con disturbo di misura rumore bianco

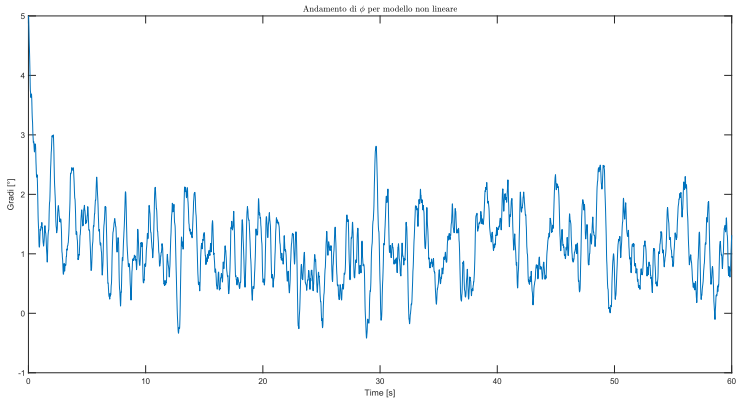

Figura 5.15: Andamento di ϕ caso Mixed Sensitivity non lineare con disturbo di misura rumore bianco

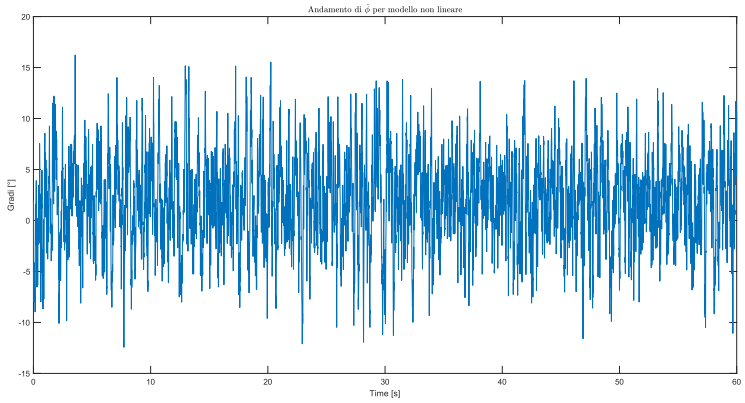

Figura 5.16: Andamento di $\dot{\phi}$ caso Mixed Sensitivity non lineare con disturbo di misura rumore bianco

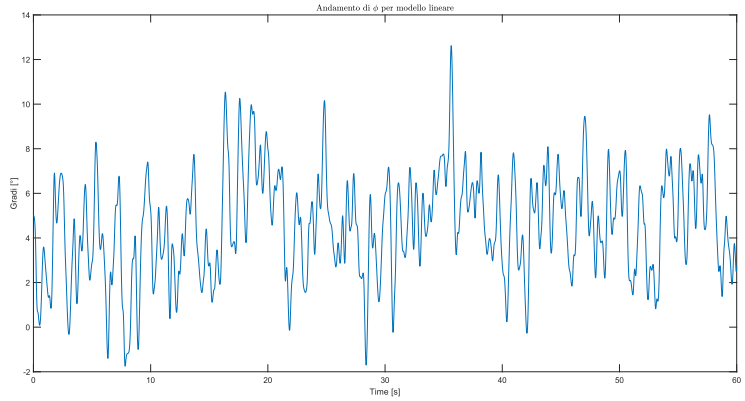

Figura 5.17: Andamento di ϕ caso DK-iteration lineare con disturbo di misura rumore bianco

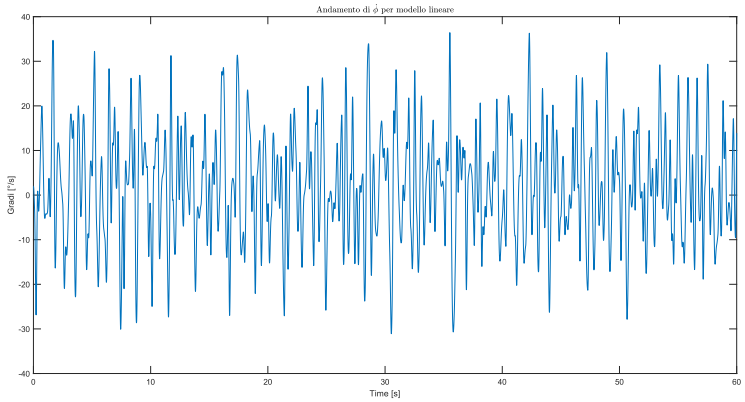

Figura 5.18: Andamento di $\dot{\phi}$ caso DK-iteration lineare con disturbo di misura rumore bianco

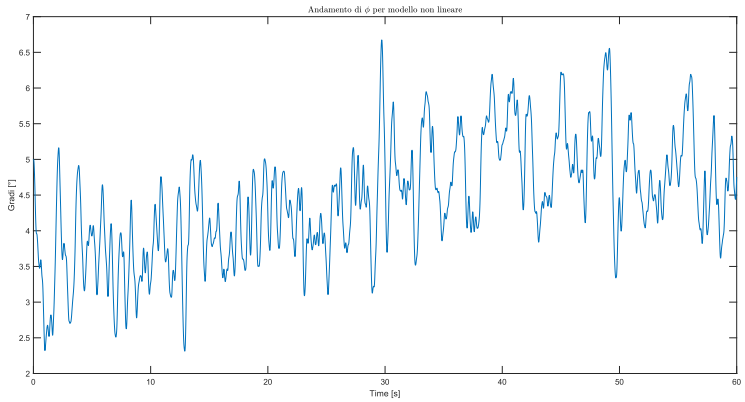

Figura 5.19: Andamento di ϕ caso DK-iteration non lineare con disturbo di misura rumore bianco

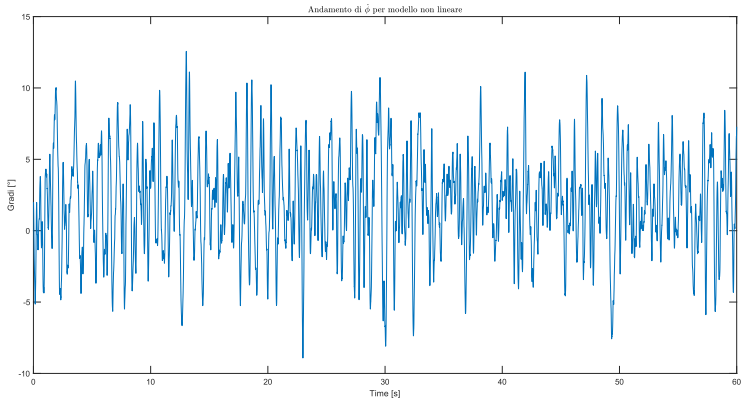

Figura 5.20: Andamento di $\dot{\phi}$ caso DK-iteration non lineare con disturbo di misura rumore bianco

Appendice A

Initialization Script

```
%% Script di inizializzazione progetto CSI Brigida-Calzaretta-Massara

% definiamo i parametri del nostro sistema
g = 9.81;
syms m r a L Km Tm M
syms x y theta phi v theta_dot phi_dot x_dot y_dot theta_dot v_dot...
    theta_dotdot phi_dotdot tau_r tau_l

M_vdot = 2*(3/2*m +0.5*M);
M_theta2dot = 2*(3/2*m*a^2 + 1/6*M*a^2 + 1/2*M*L^2*sin(phi)^2);
M_phi2dot = 2*(1/2*m*r^2 + 2/3*M*L^2);
M_vdot_phi2dot = -m*r+M*L*cos(phi);
c_v = -M*L*sin(phi)*(theta_dot^2 + phi_dot^2);
c_thetadot = 2*M*L^2*sin(phi)*cos(phi)*theta_dot*phi_dot+M*L*sin(phi)...
                *v*theta_dot;
c_phidot = -M*L^2*sin(phi)*cos(phi)*theta_dot^2;
g_phidot = -M*L*g*sin(phi);

I = [M_vdot 0 M_vdot_phi2dot;
     0 M_theta2dot 0;
     M_vdot_phi2dot 0 M_phi2dot];

C = [c_v c_thetadot c_phidot]';
G = [0 0 g_phidot]';

Ing = [1/r 1/r;
       a/r -a/r;
       -1 -1];

% Equazioni delle dinamica
csi1_dot = v*cos(theta);
csi2_dot = v*sin(theta);
csi3_dot = theta_dot;
csi4_dot = phi_dot;
csi567_dot = inv(I)*(Ing*[tau_r,tau_l]' - C -G);
```

```
csi_dot = [csi1_dot; csi2_dot;csi3_dot;csi4_dot;csi567_dot];
csi = [x,y,theta,phi,v,theta_dot,phi_dot];

% Linearizzazione attorno a csi_eq
A_sym = jacobian(csi_dot,csi);
B_sym = jacobian(csi_dot,[tau_r,tau_l]);

% Modello lineare dei sensori (phi e phi_dot)
C = [0 0 0 1 0 0 0
     0 0 0 0 0 0 1];

D = zeros(2,2);

% Punto di lavoro
csi_eq = [0 0 0 0 0 0 0];

A_eq = subs(A_sym,csi,csi_eq);
B_eq = subs(B_sym,csi,csi_eq);

m = 1.22;
r = 0.13;
a = 0.496/2;
L = 0.2924;
Km = 1.08;
Tm = 0.005;
g = 9.81;
M = 54;

% Parametri incerti
u_L = ureal('u_L',L,'Percentage',[-10 +10]);
u_Km = ureal('u_Km',Km,'Percentage',[-10 +10]);
u_Tm = ureal('u_Tm',Tm,'Percentage',[-20 +20]);


A = [0 0 0 0 0 1 0;
     0 0 0 0 0 0 0;
     0 0 0 0 0 1 0;
     0 0 0 0 0 0 1;
     0 0 0 (2943*u_L*M*...
(m*r - u_L*M))/(100*(u_L^2*M^2+12*u_L^2*M*m+6*u_L*M*m*r...
+ 3*M*m*r^2 + 6*m^2*r^2)) 0 0 0;
     0 0 0 0 0 0 0;
     0 0 0 (2943*u_L*M*(M + 3*m))/(100*(u_L^2*M^2+12*u_L^2*M*m...
+6*u_L*M*m*r + 3*M*m*r^2 + 6*m^2*r^2)) 0 0 0];

B = [0,0;
     0,0;
     0,0;
```

```
     0,0;
(4*M*u_L^2 + 3*m*r^2)/(r*(u_L^2*M^2 + 12*u_L^2*M*m + 6*u_L*M*m*r +...
3*M*m*r^2 + 6*m^2*r^2)) - (3*(m*r - u_L*M))/(u_L^2*M^2 + ...
12*u_L^2*M*m + 6*u_L*M*m*r + 3*M*m*r^2 + 6*m^2*r^2),(4*M*u_L^2 +...
3*m*r^2)/(r*(u_L^2*M^2 + 12*u_L^2*M*m + 6*u_L*M*m*r + 3*M*m*r^2 +...
6*m^2*r^2)) - (3*(m*r - u_L*M))/(u_L^2*M^2 +...
12*u_L^2*M*m + 6*u_L*M*m*r + 3*M*m*r^2 + 6*m^2*r^2);
(3*a)/(r*(M*a^2 + 9*a^2*m)),-(3*a)/(r*(M*a^2 + 9*a^2*m));
(3*(m*r - u_L*M))/(r*(u_L^2*M^2 + 12*u_L^2*M*m + 6*u_L*M*m*r +...
3*M*m*r^2 + 6*m^2*r^2)) - (3*(M + 3*m))/(u_L^2*M^2 + 12*u_L^2*M*m +...
6*u_L*M*m*r + 3*M*m*r^2 + 6*m^2*r^2),(3*(m*r - u_L*M))/(r*...
(u_L^2*M^2 + 12*u_L^2*M*m + 6*u_L*M*m*r + 3*M*m*r^2 +...
6*m^2*r^2)) - (3*(M + 3*m))/(u_L^2*M^2 + 12*u_L^2*M*m +...
6*u_L*M*m*r + 3*M*m*r^2+6*m^2*r^2)];


% Modello "state space" del sistema perturbato (senza attuatori)
G_p = ss(A,B,C,D);

s = tf('s');

% Funzione di trasferimento dell'attuatore
Gm = u_Km/(u_Tm*s + 1);

G_tau = [Gm,0;
         0,Gm];

% Matrice di trasferimento del sistema totale
G_attuata_p = G_p*G_tau;

% Modello di incertezza additiva
E = (G_attuata_p-G_attuata_p.Nominalvalue);

E_samples = usample(E,200);

% n_points = 20;
% ord = 2;
%
% sigma(E_samples);
% [freq,resp_db] = ginput(n_points); % pick n points
% for i = 1:n_points                    % Converts the logarithmic
%     resp(i) = 10^(resp_db(i)/20); % response to magnitude
% end % response
% sys = frd(resp,freq); % creates frd object
% W = fitmagfrd(sys,2); % fits the frequency
% % response

% Abbiamo ottenuto un peso per approssimare le incertezze
```

```
Wi = load("Pesi incertezza.mat").W;
Wi = [Wi,0;
      0, Wi];

% Modello "state space" del sistema non perturbato (senza attuatori)
SYS = ss(G_p,'min');
A_nom = SYS.A;
B_nom = SYS.B;
C_nom = SYS.C;

% Matrice di trasferimento e modello "state space" degli attuatori
% (nominale)
Gm_nom = Gm.NominalValue;
G_mss = ss(Gm_nom, 'min');


Am = G_mss.A;
Bm = G_mss.B;
Cm = G_mss.C;
Dm = G_mss.D;

A_att = Am*eye(2);
B_att = Bm*eye(2);
C_att = Cm*eye(2);


% Modello "state space" del collegamento sistema + attuatori
A_tot = [A_nom B_nom*C_att;
         zeros(2,2) A_att];
B_tot = [zeros(2,2); B_att];

C_tot = [C_nom zeros(2,2)];
D_tot = zeros(2,2);

G_attuata_nom = ss(A_tot,B_tot,C_tot,D_tot);

% Modello delle incertezze
delta = ultidyn('delta',[1 1]);
Delta = [delta,0;0,delta];

save('Modello schema a blocchi.mat',"Delta","G_attuata_nom");
save("Matrici spazio di stato.mat","A_tot","B_tot","C_tot","D_tot");
figure
sigma(E_samples,Wi);

%
% figure
% bodemag(G_p);
```

```
% title('Modulo del sistema perturbato senza attuatori');
%
% figure
% bodemag(G_tau);
% title('Modulo della fdt degli attuatori');
%
% figure
% bodemag(G_attuata_p)
% title('Modulo della fdt del sistema');
%
% figure
% bodemag(G_attuata_nom);
% title('Modulo della fdt del sistema nominale totale');
%
```

Listing 1: Initialization Script

Appendice B

LQG Controller

```
%% Controllo LQG

% Importiamo le matrici della dinamica del sistema
A_tot = load("Matrici spazio di stato.mat","A_tot").A_tot;
B_tot = load("Matrici spazio di stato.mat","B_tot").B_tot;
C_tot = load("Matrici spazio di stato.mat","C_tot").C_tot;
D_tot = load("Matrici spazio di stato.mat","D_tot").D_tot;

%% Controllo senza integratore
% Definiamo i rumori di misura e i disturbi in ingresso

Qk = 0.01*diag([1 1]);          % Covarianza dei disturbi in ingresso
Rk = (0.035)^2*diag([1 1]);     % Covarianza dei rumori di misura

% Definiamo i pesi del funzionale di costo (LQR)
Qr = diag([(20*pi/180)^-2 (20*pi/180)^-2 1 1]);
Rr = (0.1)^-2*eye(2);

% Troviamo un controllore ottimo LQR
K = -lqr(A_tot,B_tot,Qr,Rr);

% Sistema in spazio di stato
G_tot = ss(A_tot,B_tot,C_tot,D_tot);

% Troviamo il guadagno del filtro di Kalman
[kalmf,L,P] = kalman(G_tot,Qk,Rk);

%% Controllo con Integratore LQI
p = 2;
n = rank(ctrb(A_tot,B_tot));
Qri = [Qr zeros(4,2);
       zeros(2,4) diag([1 1])];
Rri = Rr;
Aug = [A_tot,B_tot;
       -C_tot,zeros(2,2)];
```

```
is_possible_LQI = (rank(Aug) == n + p); % Se è true allora è possibile
                                        % fare la sintesi LQI

% non è possibile realizzare il controllo LQG con integratore in quanto
% passando al sistema in variabili aumentate (phi,phi_dot e integrali
% del segnale di errore), otteniamo che esso non è
% stabilizzabile e quindi non è possibile sintetizzare il controllore
% (questo vale sia per il caso lineare che per quello non lineare)

% figure
% plot(out.phi_LQG_NL);
% title('Andamento di Phi per sistema non lineare');
%
% figure
% plot(out.phi_dot_LQG_NL)
% title('Andamento di Phi_dot per sistema non lineare');
%
% figure
% plot(out.phi_LQG_LIN);
% title('Andamento di Phi per sistema lineare');
%
% figure
% plot(out.phi_dot_LQG_LIN)
% title('Andamento di Phi_dot per sistema lineare');
```

Listing 2: LQG Controller Script

Appendice C

H_∞ Mixed Sensitivity Controller

```
%% Controllo Mixsyn

% Importiamo il sistema dal file "Modello schema a blocchi.mat"

Delta = load("Modello schema a blocchi.mat","Delta").Delta;
G_attuata_nom = load("Modello schema a blocchi.mat","G_attuata_nom").G_attuata_nom;
wi = load("Pesi incertezza.mat","W").W;
Wi = [wi,0;
      0,wi];

s = tf('s');
% Pesi sulla funzione di sensitivity
wb_p1 = 0.1;
wp1 = wb_p1/(s+wb_p1);

wp2 = 0.1;

Wp = [wp1, 0;
      0, wp2];
Wpss = ss(Wp);


% Pesi sullo sforzo di controllo u
wu = 0.01*s/(s+0.1);
Wu = [wu, 0;
      0, wu];
Wuss = ss(Wu);


% Pesi sulla funzione a ciclo chiuso T
Wt = 0.5*s/(s+100)*eye(2);
Wtss = ss(Wt);



% creo la variabili per sysic (con il controllore mixsyn)
```

```
systemnames = 'G_attuata_nom Wpss Wuss Wtss';
inputvar = '[r(2);u(2)]'; % sarebbe [w u]'
input_to_G_attuata_nom = '[u]';
input_to_Wtss = '[G_attuata_nom]';
input_to_Wpss = '[r-G_attuata_nom]';
input_to_Wuss = '[u]';
outputvar = '[Wpss;Wuss;Wtss;r-G_attuata_nom]'; % sarebbe [z v]'

sysoutname = 'P_mixsyn'; % P generalizzata nel caso mixsyn (nominale)
cleanupsysic = 'yes';
sysic;

clear systemnames inputvar input_to_G_attuata_nom input_to_Wtss
clear input_to_Wpss input_to_Wuss outputvar sysoutname cleanupsysic

[K_ms,CL,gamma]=hinfsyn(P_mixsyn,2,2);  % Algoritmo ottimo di sintesi H inf
gamma

N = lft(P_mixsyn,K_ms);   % Matrice N nel caso nominale

%% Mu-analysis

% Caso Nominale
    % Stabilità
[stabmarg_Nmix,wcu_Nmix,info_Nmix] = robuststab(N);

    % Performance
[perfmarg_Nmix,wcu_perf_Nmix,info_perf_Nmix]=robustperf(N);



% Caso perturbato (analisi di robustezza)

% Dobbiamo ricavare il sistema perturbato a ciclo chiuso considerando
% il controllore mixsyn
systemnames = 'G_attuata_nom Wi Wpss Wuss Wtss';
inputvar = '[u_delta(2);r(2);u(2)]'; % ingresso [u_delta w u]'

input_to_G_attuata_nom = '[u]'; % lista dei sistemi che abbiamo dichiarato
input_to_Wtss = '[G_attuata_nom + u_delta]';
input_to_Wi = '[u]';
input_to_Wpss = '[r-G_attuata_nom-u_delta]';
input_to_Wuss = '[u]';

% uscite [y_delta z v]'
outputvar = '[Wi;Wpss;Wuss;Wtss;r-G_attuata_nom-u_delta]';

sysoutname = 'P_mixsyn_p';  % P generalizzata nel caso mixsyn (perturbato)
```

```
cleanupsysic = 'yes';
sysic;

clear systemnames inputvar input_to_G_attuata_nom input_to_Wtss
clear input_to_Wpss input_to_Wuss outputvar sysoutname cleanupsysic
clear input_to_Wi

P_p = lft(Delta,P_mixsyn_p);      % Aggiungiamo il blocco Delta al modello
N_p = lft(P_p,K_ms);       % Ricaviamo la matrice N del modello perturbato


    % Stabilità
[stabmarg_Nmix_p,wcu_Nmix_p,info_Nmix_p] = robuststab(N_p);

    % Performance
[perfmarg_Nmix_p,wcu_perf_Nmix_p,info_perf_Nmix_p] = robustperf(N_p);


%%Plot
%
% figure
% sigma(N(1:2,:),1/Wp);
% legend('S','1/Wp');
%
% figure
% sigma(N(3:4,:),1/Wu);
% legend('KS','1/Wu');
%
% figure
% sigma(N(5:6,:),1/Wt);
% legend('T','1/Wt');
%
% figure
% sigma(Wp,Wu,Wt);
% legend("wp",'Wu','Wt');

figure
plot(out.phi_nl);
str = 'Andamento di ${\phi}$ per modello lineare';
title(str,'Interpreter','latex')
xlabel('Time [s]')
ylabel('Gradi [°]')

figure
plot(out.phi_dot_nl);
str = 'Andamento di ${\dot\phi}$ per modello lineare';
title(str,'Interpreter','latex')
xlabel('Time [s]')
```

```
ylabel('Gradi [°]')

% figure
% plot(out.phi_plot_h_inf);
% str = ['Andamento di ${\phi}$ per modello lineare'];
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
%
% figure
% plot(out.phi_dot_plot_h_inf);
% str = ['Andamento di ${\dot\phi}$ per modello lineare'];
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
```

Listing 3: Mixed Sensitivty Controller

Appendice D

DK-iteration Controller

```
%% DK-iteration

% Importiamo il sistema dal file "Modello schema a blocchi.mat"

Delta = load("Modello schema a blocchi.mat","Delta").Delta;
G_attuata_nom = load("Modello schema a blocchi.mat", ...
                                                "G_attuata_nom").G_attuata_nom;
wi = load("Pesi incertezza.mat","W").W;
Wi = [wi,0;
      0,wi];

s = tf('s');

% Pesi sulla funzione di sensitivity
wb_p1 = 0.01;
wp1 = 0.55*wb_p1/(s+wb_p1);

wp2 = 0.1;

Wp = [wp1, 0;
      0, wp2];
Wpss = ss(Wp);
%% Plot pesi sulla funzione di sensitività

% Pesi sullo sforzo di controllo u
wu = 0.01*s/(s+0.1);
Wu = [wu, 0;
      0, wu];
Wuss = ss(Wu);

%% Plot pesi sullo sforzo di controllo u

% Pesi sulla funzione a ciclo chiuso T
Wt = 0.5*s/(s+100)*eye(2);
Wtss = ss(Wt);
```

```
%% Plot pesi sulla funzione a ciclo chiuso T


% creo la variabili per sysic
systemnames = 'G_attuata_nom Wpss Wuss Wtss Wi';
inputvar = '[u_delta(2);r(2);d(2);n(2);u(2)]'; % sarebbe [w u]'
input_to_G_attuata_nom = '[u]';
input_to_Wtss = '[G_attuata_nom + u_delta]';
input_to_Wi = '[u]';
input_to_Wpss = '[r-G_attuata_nom-d-u_delta]';
input_to_Wuss = '[u]';
outputvar = '[Wi;Wpss;Wuss;Wtss;r-G_attuata_nom-u_delta-d-n]'; % sarebbe [z u]' --

sysoutname = 'P';
cleanupsysic = 'yes';
sysic;
P_Delta = lft(Delta,P);

opts_musyn = musynOptions('MaxIter',30,'TargetPerf',0.7);
[K_musyn,CLperf_musyn,info_musyn] = musyn(P_Delta,2,2,opts_musyn);

N_mu = lft(P,K_musyn);
BlockStructure = [2 0;6 6];
[bounds,mu_info] = mussv(ss(N_mu),BlockStructure,'o');
mu = squeeze(bounds.ResponseData);
w_mu = bounds.Frequency;

% loglog(w_mu,mu);
N_tot = lft(P_Delta, K_musyn);

%% Mu Analysis

% Stabilità
    % Nominale
[stabmarg_N_mu,wcu_N_mu,info_N_mu] = robuststab(N_mu);
    % Robusta
[stabmarg_N_mu_p,wcu_N_mu_p,info_N_mu_p] = robuststab(N_tot);

%Prestazioni
    % Nominali
[perfmarg_N_mu,wcu_perf_N_mu,info_perf_N_mu]=robustperf(N_mu(3:end,3:end));
    % Robuste
[perfmarg_N_mu_p,wcu_perf_N_mu_p,info_perf_N_mu_p] = robustperf(N_tot);



%% Plot
```

```
% figure('name','Valori singolari N nominale DK-iteration')
% sigma(N_mu(3:end,3:end));
% legend('N nominale');
%
% figure('name','Valori singolari N perturbato DK-iteration')
% sigma(N_tot);
% legend('N perturbata');
%
% figure
% plot(out.phi_plot_h_inf);
% str = 'Andamento di ${\phi}$ per modello lineare';
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
%
% figure
% plot(out.phi_dot_plot_h_inf);
% str = 'Andamento di ${\dot\phi}$ per modello lineare';
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
%
% figure
% plot(out.phi_nl);
% str = 'Andamento di ${\phi}$ per modello non lineare';
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
%
% figure
% plot(out.phi_dot_nl);
% str = 'Andamento di ${\dot\phi}$ per modello non lineare';
% title(str,'Interpreter','latex')
% xlabel('Time [s]')
% ylabel('Gradi [°]')
%
```

Listing 4: DK-iteration Script