UNIVERSITÀ
DELLA CALABRIA

Progetto Laboratorio di automatica

Vaccaro Francesca 239641

Traccia 1

Progetto Laboratorio di automatica

Vaccaro Francesca 239641

L.T. Ingegneria Informatica 24/25

Contents

1 Introduzione

Il seguente progetto si propone di risolvere due esercizi finalizzati nella realizzazione di controllori digitali, il primo utilizzerà regolatori digitali che non faranno uso del modello del sistema, il secondo invece sarà *model based*.

1.1 Strumenti e software utilizzati

A tale scopo verrà utilizzato come principale strumento **Matlab**, il suo acronimo sta per *Matrix Laboratory*, in quanto nasce proprio come software per l'agevolazione dei problemi e dei calcoli matriciali, con il tempo questo software è diventato fondamentale per le applicazioni nell'ambito dell'automatica.

In particolare per rendere tutto più fluido e comprensibile verrà utilizzato il **live script** che permetterà di avere una migliore panoramica di codice, testo e grafici.

Per ultimo come ambiente simulativo si farà riferimento a **simulink**, una sezione di matlab dedicata alla simulazione dei sistemi dinamici, anche questo inizialmente utilizzato per la risoluzione di equazioni differenziali, è poi diventato indispensabile nel settore industriale e via dicendo; senza entrare troppo nei particolari, questo si compone di vari blocchi estratti da differenti librerie, e ha un collegamento diretto con matlab, lo si può considerare difatti un'interfaccia molto avanzata.

2 Primo esercizio

Il primo esercizio richiede la progettazione di un regolatore standard digitale, tale che possa soddisfare le seguenti caratteristiche :

- errore nullo rispetto ad un riferimento a gradino.
- sovraelongazione percentuale minore del 10% e tempo di assestamento minore di 3 secondi.

Il sistema che sarà al centro dell'esercizio è un sistema del tipo :

$$G(s) = \frac{-s+2}{\frac{1}{2}s^3 + 2s^2 + 4s}$$

Durante la trattazione però viene richiesto di non conoscere il modello, l'unica conoscenza a disposizione sarà quella della risposta a gradino.

2.1 Regolatori Standard

È importante capire cosa è un regolatore standard, prima di andarne a progettare uno; questi sono dei dispositivi che vengono utilizzati all'interno di problemi di regolazione, l'azione del regolatore si ottiene andando a tarare quelli che sono i parametri da cui è

formato, cercando di soddisfare i requisiti richiesti a ciclo chiuso. Più in particolare si noti che questi prendono il nome di **PID** in quanto vanno a combinare tre azioni :

- P : azione proporzionale,
- I : azione integrale,
- D : azione derivativa.

In particolare rappresentiamo u(t) del pid analogico nel seguente modo :

$$u(t) = k_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt})$$

In questa formula :

- K_p rappresenta il guadagno proporzionale
- $\frac{K_p}{T_i}$ rappresenta il guadagno integrale
- $K_p \cdot T_d$ rappresenta il guadagno derivativo

L'obiettivo della taratura del PID è proprio quella di andare a determinare i valori dei parametri sopra elencati, in modo da riuscire a soddisfare le caratteristiche richieste.

La cosa interessante dei regolatori standard è la possibilità di progettazione, senza la conoscenza del modello, questo ovviamente li rende molto versatili.

In particolare nel caso dell'approccio a questo esercizio viene fornita solo l'informazione riguardante la risposta a gradino della funzione di trasferimento.

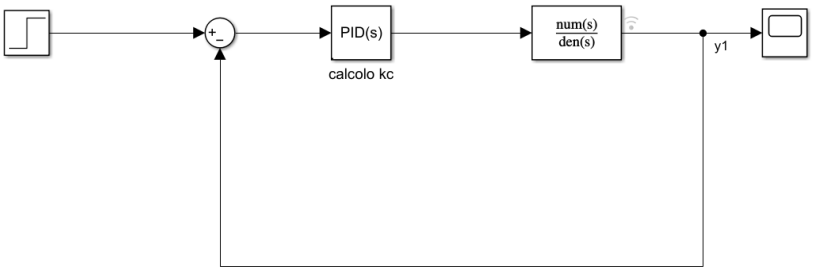
2.2 Metodi di taratura

È bene sapere che per la taratura dei pid esistono differenti tecniche, alcune ad anello chiuso altre ad anello aperto, alcune permetteranno di avere delle informazioni maggiormente dettagliate, mentre altre forniranno risultati più laschi. Le tecniche più conosciute ed utilizzate sono :

- Ziegler-Nichols ad anello chiuso,
- Ziegler-Nichols ad anello aperto,
- Metodo delle aree,
- Metodo della tangente.

2.3 Applicazione Ziegler-Nichols anello chiuso

Il metodo che personalmente ho preferito utilizzare è quello di **Ziegler-Nichols ad anello chiuso**, si parte dunque con l'implementazione su simulink del sistema, andando però ad aggiungere due particolari blocchi :

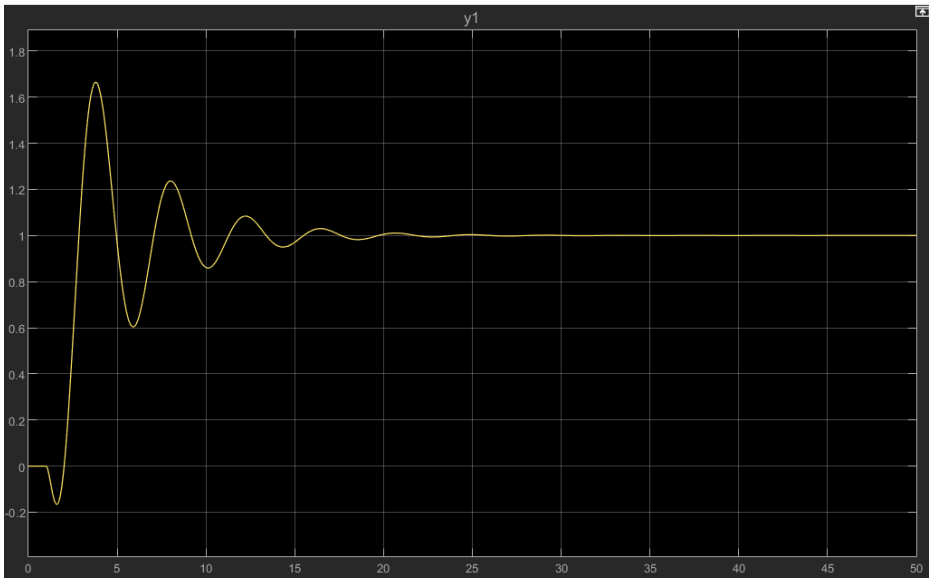


- PID-Controller
- scope

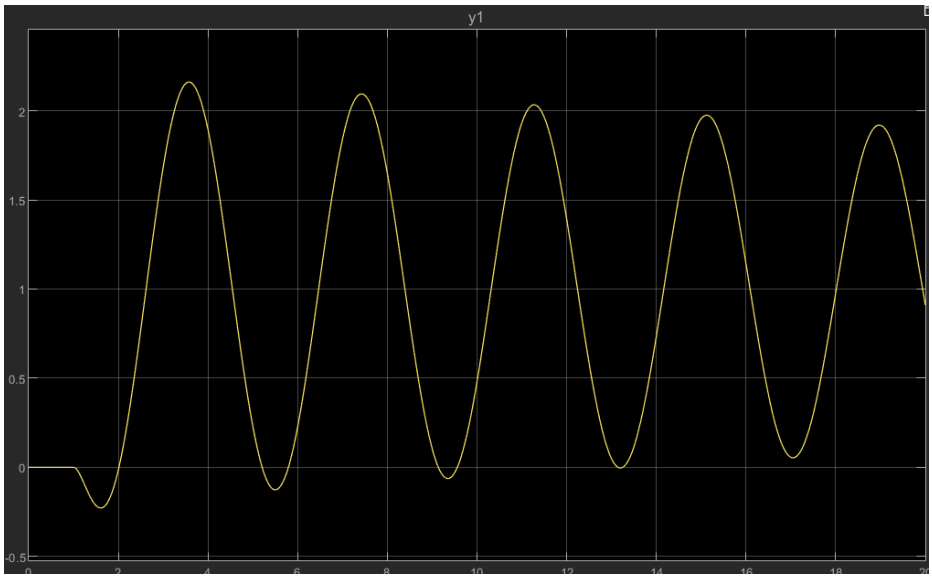
Il primo permette di aggiungere i parametri del pid, il secondo di visualizzare a schermo l'andamento della risposta a gradino.

La prima operazione che va fatta per attuare il metodo Ziegler Nichols è spingere il sistema fino all'instabilità; si deve quindi trovare il valore del guadagno critico k_c, e lo si nota quando nel grafico ricavato dallo scope si evidenziano oscillazioni costanti; come si può notare questo metodo è un metodo empirico, basato su diverse prove, dunque potrebbe anche dare dei risultati leggermente laschi rispetto a quello che si cerca.

Quello che si ottiene se si mette il coefficiente del parametro proporzionale pari a 2 è :

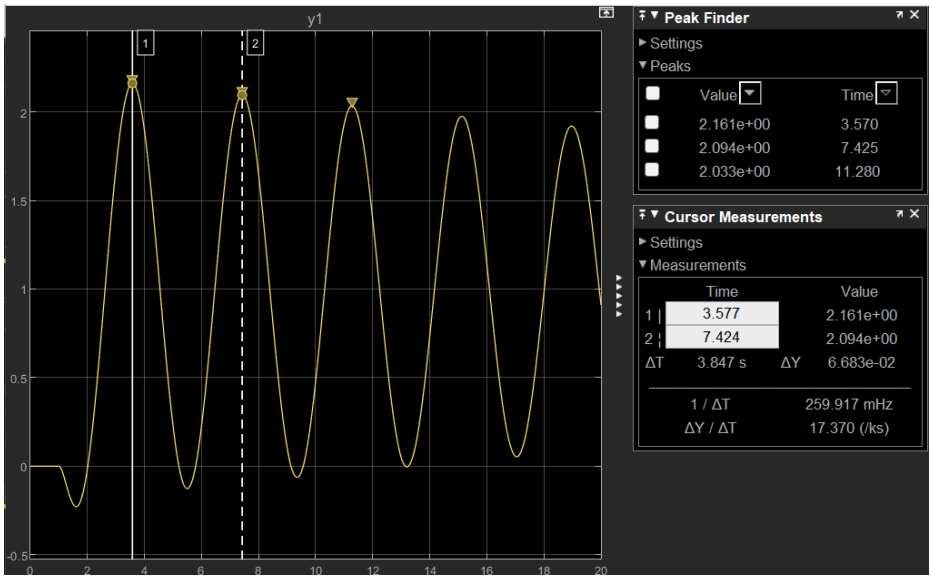


Facendo altre prove si arriva alla determinazione di un k_p pari a 2.666 :



Questo valore va tenuto bene a mente, perchè insieme al valore del periodo che intercorre tra le oscillazioni, si potrà risalire ai valori dei parametri del pid, attraverso una tabella.

Per individuare il periodo si possono utilizzare strumenti dello scope di misurazione.



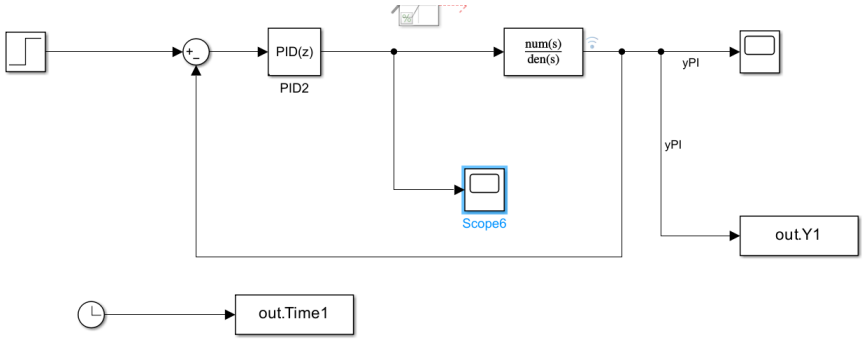
Il periodo individuato è : 3.847 (ovviamente questa misurazione potrebbe presentare valori non precisi al millimetro).

A questo punto si può utilizzare la tabella per trovare i parametri prima citati, in base al regolatore che si vuole progettare (PID, PI, PD).

	K_p	T_i	T_d
P	$0.5\bar{K}_p$		
PI	$0.45\bar{K}_p$	$0.8T$	
PID	$0.6\bar{K}_p$	$0.5T$	$0.125T$

Ora si hanno tutti gli strumenti per poter passare alla taratura dei parametri, la maggior parte di questo lavoro verrà svolta all'interno di simulink, per poter modificare i vari valori, e portarli più vicino possibile alle richieste del problema.

Si cominci dunque con la costruzione del PID vero e proprio, oltre ai blocchi prima elencati sono stati aggiunti anche dei blocchi che permetteranno di "trasportare" il sistema direttamente su un live script, così da poter applicarvi la funzione *stepinfo()*, che darà tutte le informazioni di cui si ha bisogno; inoltre bisogna ricordare che per matlab il *tempo di assestamento* viene calcolato con il 2% come intervallo, nella richiesta invece lo si vuole al 5%, dunque al momento dell'utilizzo della funzione di stepifo, dovrà essere specificata la "bontà" di valutazione.



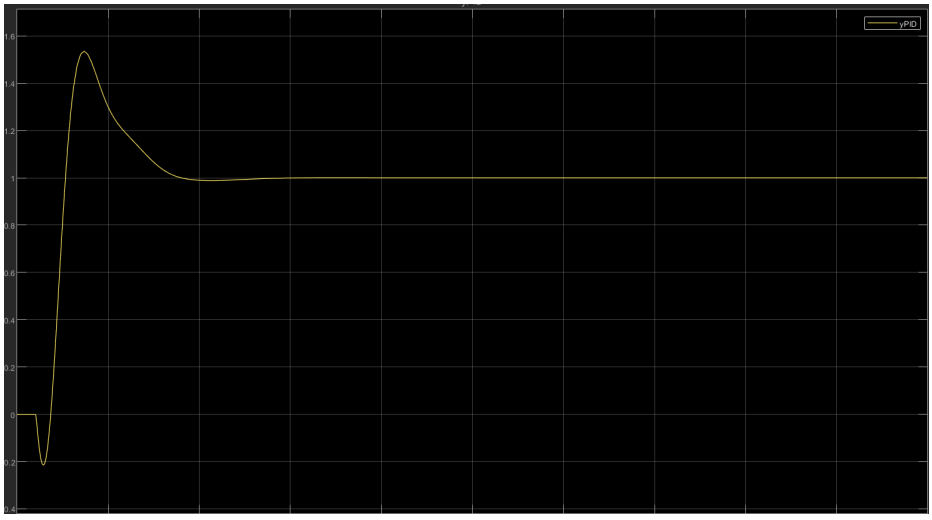
L'altro scope serve invece per valutare il segnale di comando dal pid. Calcolando i vari parametri in accordo con la tabella sopra riportata e i valori di k_c e T, vengono fuori i seguenti risultati:

- $k_p = 1.5996$
- $T_i = 1.9195$
- $T_d = 0.4799$

Ovviamente si tenga conto che il fattore del guadagno integrale sarà corrispondente a $\frac{1}{T_i}$.

Mettendo questi primi risultati all'interno di matlab, quello che viene fuori non rispetta per nulla i requisiti imposti, che si ricordano essere : una sovraelongazione percentuale minore del 10% e un tempo di assestamento minore di 3 secondi.

Il grafico del segnale che viene fuori è questo :



E queste sono le sue caratteristiche

```
stepinfo(out.Y, "SettlingTimeThreshold", 0.05)
```

```
ans = struct with fields:
          RiseTime: 4.9999
      TransientTime: 74.1099
       SettlingTime: 74.9022
        SettlingMin: 0.9883
        SettlingMax: 1.5357
          Overshoot: 53.5678
         Undershoot: 21.5199
               Peak: 1.5357
           PeakTime: 55
```

2.3.1 Miglioramento del sistema

C'è quindi molto da lavorare sui parametri per abbassare le caratteristiche, ovviamente si dovrà fare il tutto in piccoli passi, perchè altrimenti si rischierebbe di portare il sistema alla rottura.

Vengano fatte dunque alcune considerazioni sui vari guadagni e su come una loro eventuale modifica andrebbe ad incidere sulla **sovraelongazione** e sul **tempo di assestamento** :

- Ridurre il tempo di assestamento :

  - Aumentare K_p
  - Aumentare T_d
  - Aumentare $\frac{1}{T_i}$
- Ridurre la sovraelongazione :
  - Ridurre K_p
  - Aumentare T_d
  - Ridurre $\frac{1}{T_i}$

Inizialmente quello che si va a fare è proprio aumentare il contributo del guadagno derivativo, ovviamente bisogna prestare attenzione, andare ad applicare un aumento troppo elevato, potrebbe causare un fenomeno che prende il nome di *"calcio derivativo"*.

Attuando varie prove e facendo piccole variazioni, si arriva alla determinazione di questi parametri :

- $K_p = 1.2105$
- $\frac{1}{T_i} = 0.045606$
- $T_d = 0.25687$

Visualizziamo il grafico e le caratteristiche

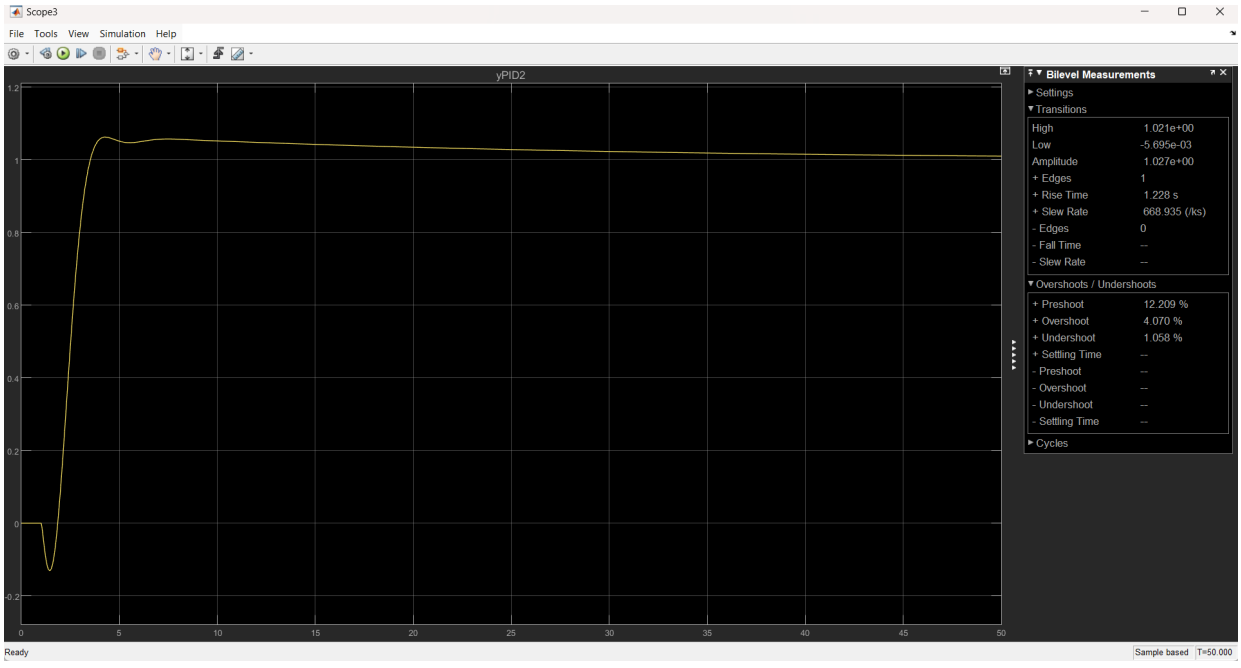


Figure 1: Sovraelongazione

Per il calcolo del tempo di assestamento è stato necessario servirsi degli strumenti di misurazione messi a disposizione dallo scope, andando a creare un intervallo del 5% e servendosi della definizione stessa.

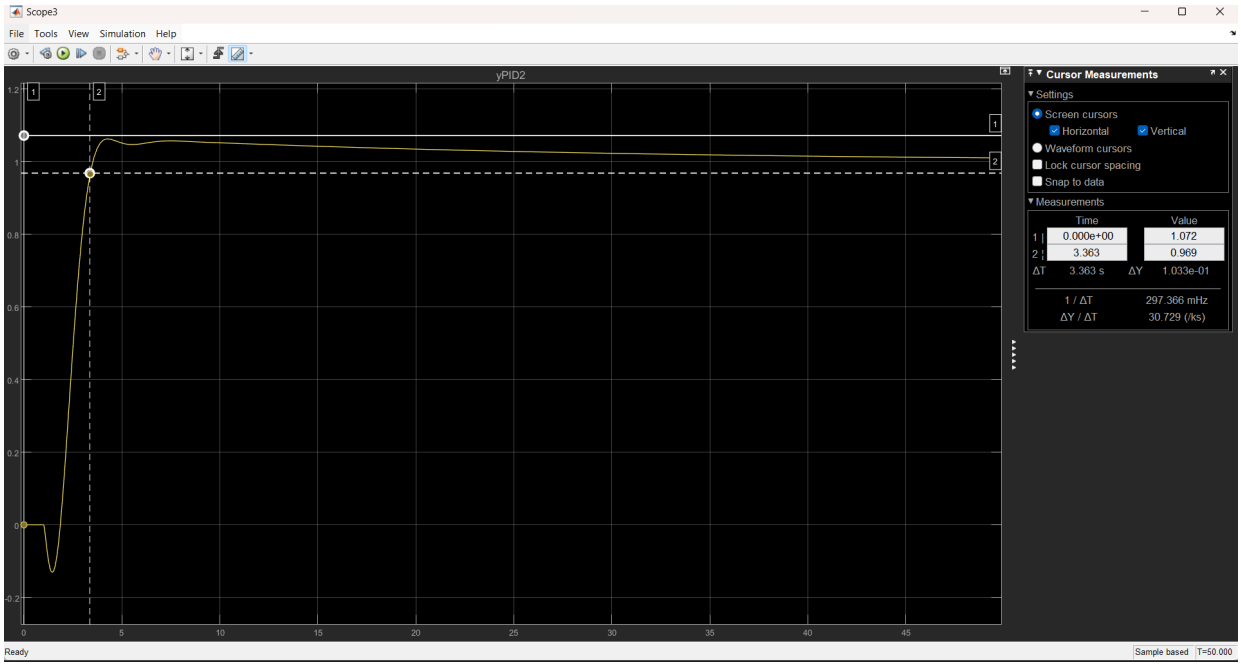
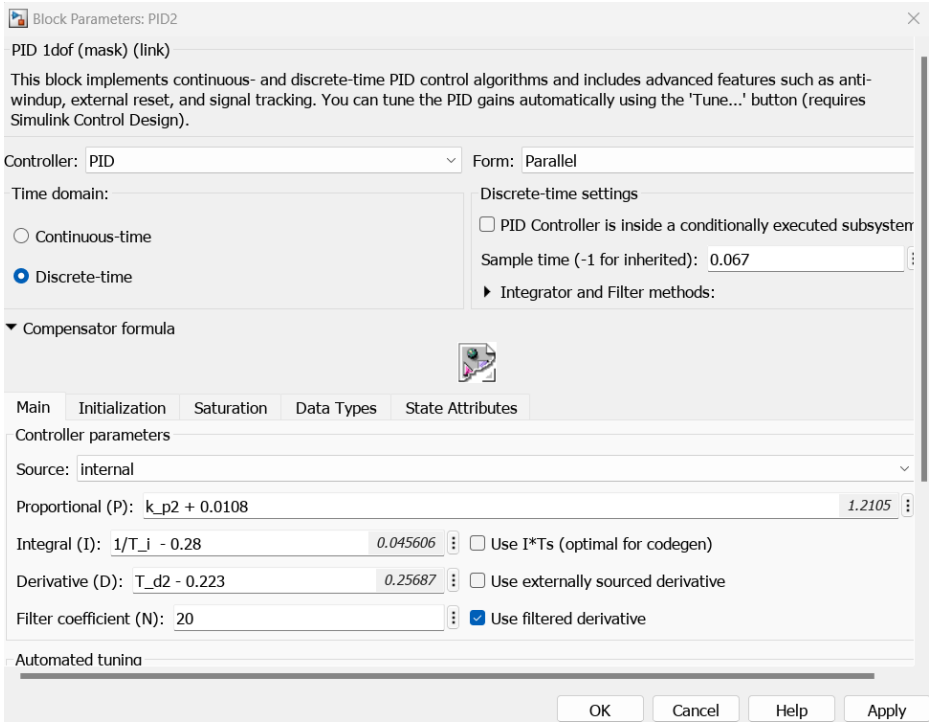
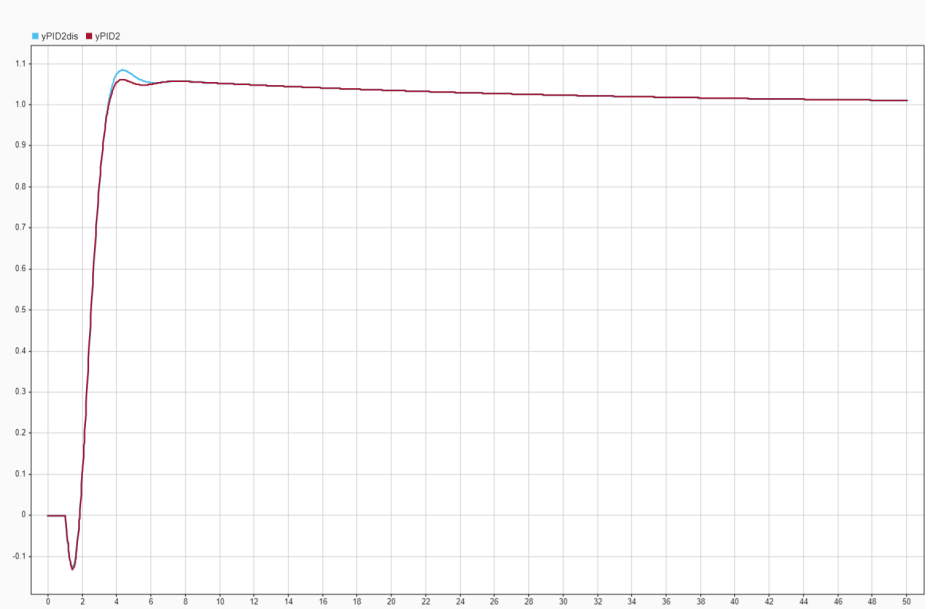


Figure 2: Tempo di assestamento

Come si può notare la richiesta relativa alla sovraelongazione è stata ampiamente rispettata, di poco invece non è stata rispettata quella relativa al tempo di assestamento che comunque ha raggiunto ottimi livelli, è arrivato il momento di metterlo nel discreto, per farlo basterà andare a selezionare il blocco del pid, selezionare il tempo discreto e infine mettere il tempo di campionamento che si vuole utilizzare, attenendosi sempre a quelle che sono le caratteristiche richieste, dunque $T_c \leq \frac{T_s}{40}$



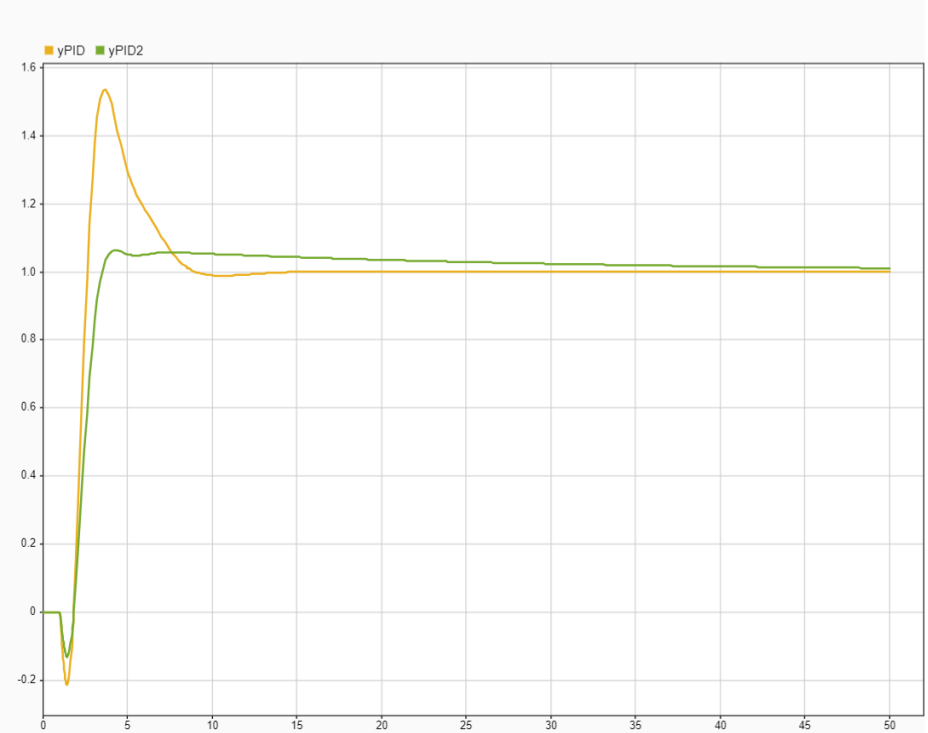
Questo è il confronto tra il PID analogico e quello digitale :



Come si può notare quello che succede discretizzando è un aumento della sovraelongazione.

2.3.2 Confronti

Vediamo a confronto il primo e l'ultimo risltato



2.4 Altri Regolatori

Per completezza vediamo anche le altre due combinazioni di regolatori, quindi **PI** e **PD**, e anche il singolo **P**.

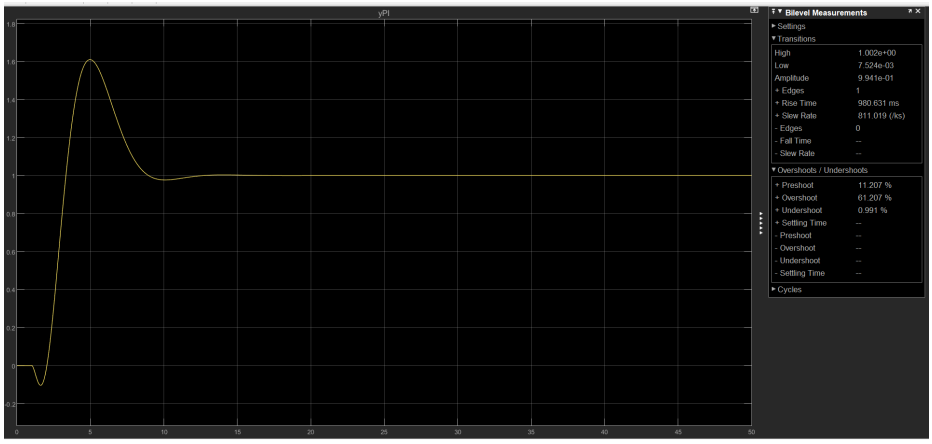
2.4.1 Regolatore PI

Il regolatore PI vede l'unione tra l'elemento *proporzionale*, e l'elemento *integrale*, dunque si vanno a combinare gli effetti di controllo proporzionale degli errori e di memorizzazione della storia passata.

$$u(t)_{PI} = k_p(e(t) + \frac{1}{T_i}\int_0^1 (e(\tau)d\tau))$$

Trasformandolo nel dominio di s si ottiene l'equivalente di una rete attenuatrice.

Vediamo cosa si ottiene andando a utilizzare la tabella per il tuninig.



2.4.2 Regolatore PD

Il PD vede l'unione del guadagno proporzionale e dell'effetto derivativo, che permettono di avere il controllo proporzionale sull'errore e la predizione di quest'ultimo.

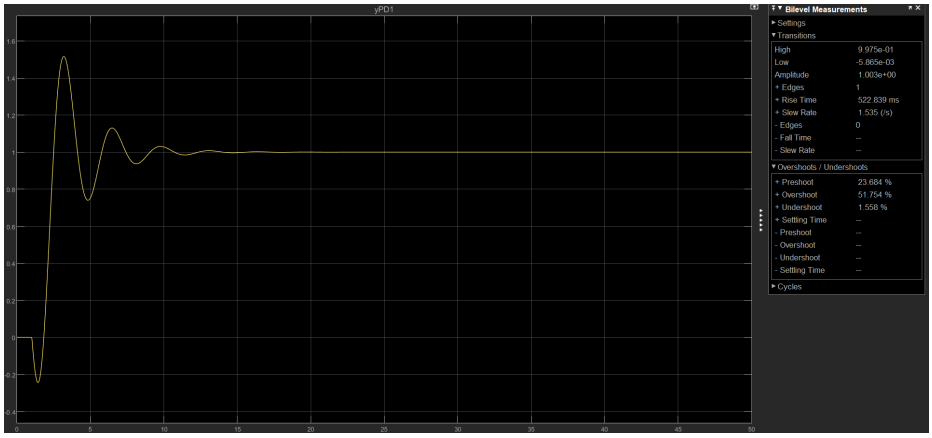
$$u(t)_{PD} = k_p(e(t) + T_d \cdot \frac{de(t)}{dt})$$

Quando lo portiamo nel dominio della variabile s ci rendiamo conto che "puro" non è realizzabile, data la presenza di uno zero in $-\frac{1}{T_d}$, quindi si modifica in :

$$U(s) = k_p(1 + \frac{T_d \cdot s}{1 + \frac{T_d}{N} \cdot s})$$

A differenza di PI questa rappresenta una rete anticipatrice.

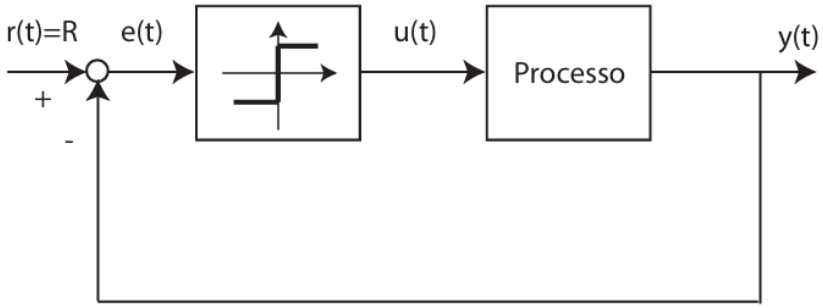
Vediamolo in simulink



Come si può notare ci sono delle oscillazioni.

2.5 Considerazione riguardo il metodo di taratura

Prima di passare al prossimo esercizio, è bene riflettere su una cosa, la risoluzione di questo punto è stata possibile in quanto è stato applicato un metodo di *tuning* adeguato, in realtà però quando nel pratico si va ad utilizzare il metodo di *Ziegler-Nichols* ad anello chiuso, bisogna prestare particolare attenzione alla determinazione di K_c, che essendo appunto un guadagno critico porta o potrebbe portare alla rottura del sistema, questo fattore in un'applicazione reale genera diversi problemi. Motivo per cui si è sviluppato un metodo che sfrutta un ***relè a due stati ON-OFF***, in modo tale che quando *l'errore* è positivo, l'ingresso del sistema(uscita del relè) scatta verso l'alto(ON), quando è negativo va OFF, inizialmente l'uscita del relè, ovvero l'ingresso del sistema è al massimo, così che poi si vadano a formare le varie oscillazioni che permettono la determinazione di K_c e di conseguenza di T_c.



3 Secondo esercizio

Questo esercizio propone la realizzazione di un ***controllore digitale***, a partire da un modello di sistema dato, e rispettando vincoli relativi alla *precisione statica* e alla *precisione dinamica*.

3.1 Modello del sistema

$$G(s) = \frac{1}{s \cdot (s+4)^3}$$

Si possono fare già alcune considerazioni riguardanti il sistema che è stato assegnato; una molto importante è relativa al sistema che ne determina il ***tipo***, esistono infatti diversi tipi di sistemi che si basano sul numero di *poli*[1] che hanno nell'origine, distinguiamo infatti:

- sistema di tipo 0 : 0 poli nell'origine;
- sistema di tipo 1 : 1 polo nell'origine;
- sistema di tipo 2 : 2 poli nell'origine.

Solitamente non si va troppo oltre con i poli nell'origine, in quanto questi potrebbero causare un'instabilità del sistema. Il sistema sotto trattazione è di **tipo 1**, in quanto ha un solo polo nell'origine, lo si può già vedere senza il bisogno di un calcolo formale, qualora però si volesse una conferma si può sfruttare matlab.

Si istanzi per comodità un live script, dove verrà inserito il seguente codice:

```
s = tf('s'); %tf introduce la transfer function
Gs = 1/(s*(s + 4)^3);
pole(Gs)
```

e si ottiene:

```
ans = 4×1 complex
```

	1
1	0.0000 + 0.0000i
2	-4.0000 + 0.0000i
3	-4.0000 + 0.0000i
4	-4.0000 - 0.0000i

Come si può notare, un polo nell'origine e un polo corrispondente a -4 con molteplicità algebrica 3; bisognerebbe fare attenzione alla molteplicità dei poli che è abbastanza alta e potrebbe andare a incidere sulla perdita di fase, che a sua volta nel progetto di controllore potrebbe compromettere il **margine di fase**, e si potrebbero rallentare le dinamiche.

Le richieste della progettazione sono:

1. errore massimo del 5% rispetto ad un riferimento a rampa con pendenza unitaria.
2. Sovra-elongazione percentuale massima pari al 20% e tempo di salita minore di 3 sec.

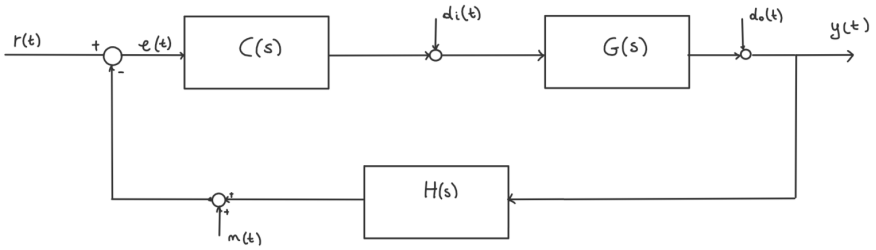
La prima richiesta si riferisce alle caratteristiche di **precisione statica**, mentre la seconda a quelle di **precisione dinamica**.

[1] i poli sono considerati in maniera semplicistica come gli zeri del denominatore

Si può osservare distintamente nelle successive sottosezioni il processo di creazione del controllore a partire dalle richieste.

3.2 Cos'è un controllore?

Prima di progettare, la cosa più immediata è capire al meglio cos'è un controllore, e per farlo si parte da un modello in retroazione negativa:



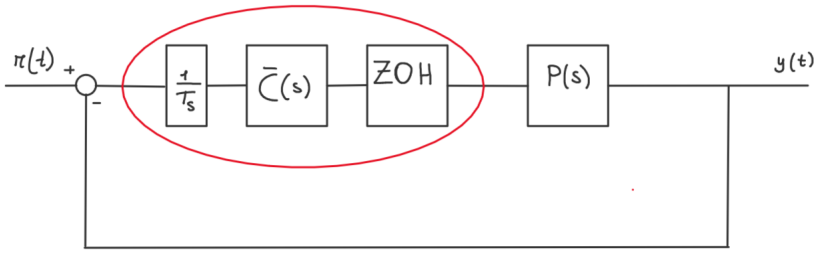
Quello che andrà progettato è il C(s) che è un algoritmo, una funzione matematica che deve fare in modo che il segnale y(t) (ovvero l'uscita del nostro schema) sia quanto più approssimato al segnale di riferimento r(t) a regime, dunque deve essere modellato in modo che eventuali disturbi che vediamo contrassegnati come $d_i(t), d_0(t)$ e l'errore $e(t)$ sia inferiore a una certa soglia, e poi devono essere garantite anche delle specifiche nel transitorio (prima che il sistema vada a regime).

La prima cosa importante da tenere d'occhio in un sistema è la **stabilità**, in particolare si fa riferimento alla **BIBO stabilità**[2] e per questo va regolato il G(s) in modo da rispettare i vincoli imposti sui segnali.

3.3 Controllore digitale e determinazione Tempo di Campionamento

In particolare questo problema richiede la progettazione di un controllore **digitale**, dunque bisogna andare a trovare un modo per *discretizzare* il sistema, per farlo bisognerà attuare delle operazioni di ***campionamento*** e ***ricostruzione***, difatti lo schema del sistema retroazionato risulterà essere in questo modo:

[2]Bounded Input Bounded Output



Quindi, prima di iniziare a verificare le varie condizioni, bisogna andare a stabilire un tempo di campionamento T_c che dovrà rispettare dei requisiti, basati sulle richieste relative alle caratteristiche del sistema, in particolare:

$$-T_c \leq \frac{t_s}{40}$$

$$-\frac{t_s}{20} \leq T_c \leq \frac{t_r}{10}$$

Dove t_s corrisponde al *tempo di assestamento*[3] e t_r al *tempo di salita*[4]. Il tempo di campionamento che è stato scelto è $\frac{2}{50}$ che rientra nelle richieste del problema. A questo punto una volta determinato T_c, si può passare al calcolo di H(s) :

$$H(s) = \frac{T_s}{(1 + \frac{T_s}{2} \cdot s)}$$

E infine si calcola P(s) :

$$P(s) = \frac{1}{T_s} \cdot H(s) \cdot G(s)$$

Dal codice scritto sul live script, viene fuori il seguente valore :

$$\frac{1}{0.02s^5 + 1.24s^4 + 12.96s^3 + 49.28s^2 + 64s}$$

Adesso si ha a disposizione la base per poter lavorare sulle varie caratteristiche richieste.

3.4 Prima richiesta : precisione statica

La prima richiesta riguarda la precisione statica, difatti richiede di costruire un controllore che permetta la presenza di un errore massimo pari al 5%, rispetto ad un riferimento di

[3] È il tempo necessario affinché la risposta del sistema si porti e rimanga all'interno di un intervallo di confidenza(per matlab è del 2%)

[4] Tempo necessario affinché il sistema passi da un valore misurato al 10% ad uno misurato al 90%

rampa unitaria.

La rampa unitaria è un segnale di ingresso formato nel seguente modo :

Dominio del tempo

$$\begin{cases} t & t \geq 0 \\ 0 & t < 0 \end{cases}$$

Dominio variabile complessa "s"

$$\begin{cases} \frac{1}{s^2} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

3.4.1 Gestione dell'errore

Ora, l'errore in un modello retroazionato non è altro che la differenza che si misura tra il riferimento e l'uscita calcolata, quello che si vuole è che questo a regime sia minore del 5%, si scriva il limite:

$$\lim_{t \to \infty} e(t)$$

A questo punto sfruttando i risultati ottenuti dal teorema del valore finale, si scriva il limite in funzione non di "t", bensì della variabile complessa "s", dunque sarà del tipo:

$$\lim_{s \to 0} s \cdot E(s) \Rightarrow \lim_{s \to 0} s \cdot \frac{1}{1 + C(s)G(s)} \cdot \frac{1}{s^2}$$

Si riporti il sistema richiesto numericamente, ricordando che in questo caso G(s) corrisponde in realtà a P(s):

$$\lim_{s \to 0} s \cdot \frac{1}{1 + k \cdot \frac{1}{0.02s^5 + 1.24s^4 + 12.96s^3 + 49.28s^2 + 64s}} \cdot \frac{1}{s^2}$$

Siccome deve essere minore del 5% :

$$\lim_{s \to 0} s \cdot \frac{1}{1 + k \cdot \frac{1}{0.02s^5 + 1.24s^4 + 12.96s^3 + 49.28s^2 + 64s}} \cdot \frac{1}{s^2} \leq 0.05$$

Prima di procedere nel calcolo effettivo, è bene notare un punto essenziale, ovvero quello della scelta di C(s) = k, questa infatti non è casuale, dipende proprio dall'osservazione fatta in precedenza sul tipo di sistema che si sta trattando, e le cose continuano a valere anche dopo il ricostruttore, in quanto non ci sono state modifiche nel tipo di sistema.

Quando un sistema è di tipo 1 e si ricerca l'errore alla rampa, questo sarà finito, il perché lo si vede dal limite sopra riportato; se invece il sistema fosse stato di tipo 0, quello che sarebbe venuto fuori, sarebbe stato un limite infinito, e quindi il controllore sarebbe dovuto essere $\frac{k}{s}$ perché la "s" va ad aggiungere un polo e dunque a "sistemare" il sistema.

In particolare queste informazioni sono racchiuse all'interno di un *teorema*, che è quello del ***modello interno***

"Se il controllore contiene la stessa struttura del riferimento, allora il problema dell'inseguimento si risolve"

Dopo aver fatto le dovute precisazioni, si passi nuovamente al calcolo del k, senza riportare tutti i calcoli fatti, basti riconoscere l'annullamento delle varie "s", quello che rimane è il seguente risultato:

$$k \leq 1280$$

A questo punto, si scelga un k che rispetti la condizione e si proceda con alcune osservazioni sullo stato del sistema : in particolare è stato scelto il minimo valore possibile, quindi proprio 1280, sicuramente un guadagno così alto potrebbe andare a gravare sulla stabilità del sistema, nonostante possa comunque migliorarne la velocità.

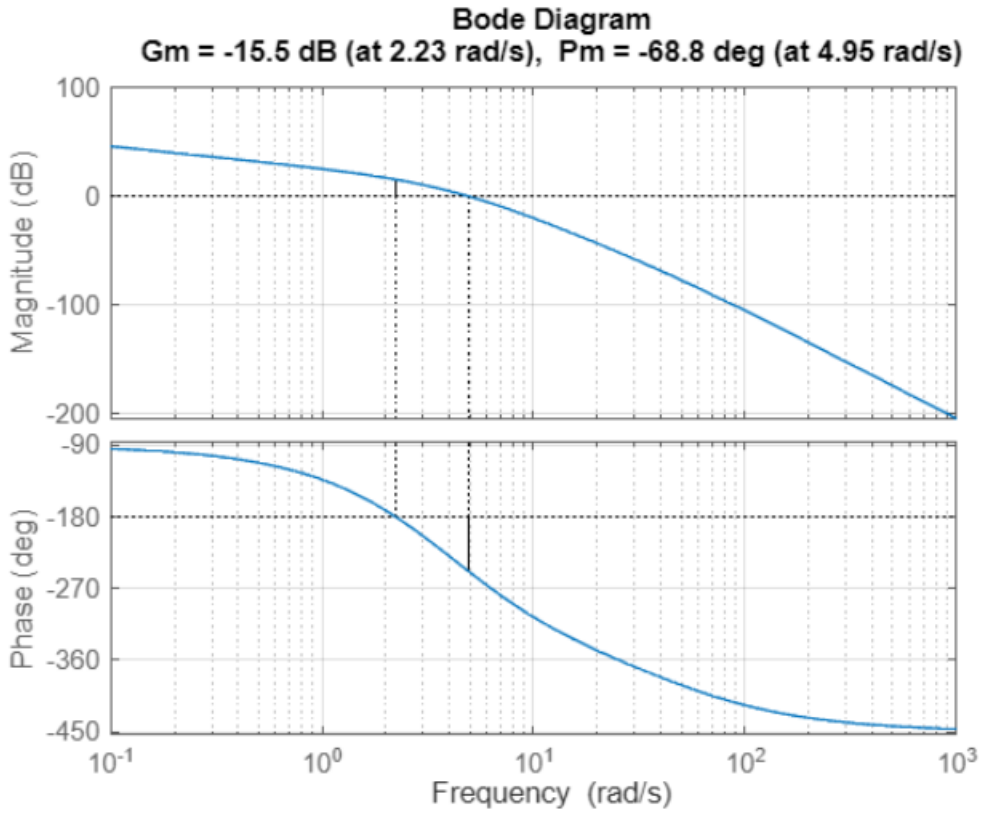
Si metta il risultato all'interno della funzione ad anello L(s) :

$$L(s) = C(s)G(s)$$

3.4.2 Considerazioni sulla stabilità

Una considerazione relativa proprio alla *stabilità* la si può fare a partire dal **diagramma di Bode**, oppure dal **diagramma di Nyquist**; se si considera il primo, conviene utilizzare su matlab un particolare comando che da' informazioni relative ai margini di stabilità :

```
margin(Ls) %plotta i margini di fase e di ampiezza nel
    contesto del diagramma di Bode
grid
```



Sia il **margine di fase**, che il **margine di ampiezza**, risultano negativi, secondo questo risultato si può dedurre che il sistema non rispetta il criterio di **Bode**, dunque non è bibo stabile.

Si potrebbe aprire una piccola parentesi sul significato di margine di ampiezza e margine di fase, questi sono degli indicatori che per sistemi stabili indicano la loro "tendenza" all'instabilità(in questo caso il sistema è già instabile).

Il **margine di ampiezza** indica di quanto si può aumentare il guadagno prima che sistema diventi instabile, e si trova nel seguente modo :

$$G_m = \frac{1}{|L(j\omega\pi)|}$$

Nell'ottica della misurazione in decibel :

$$G_m = -20log_{10}(|L(j\omega\pi)|)$$

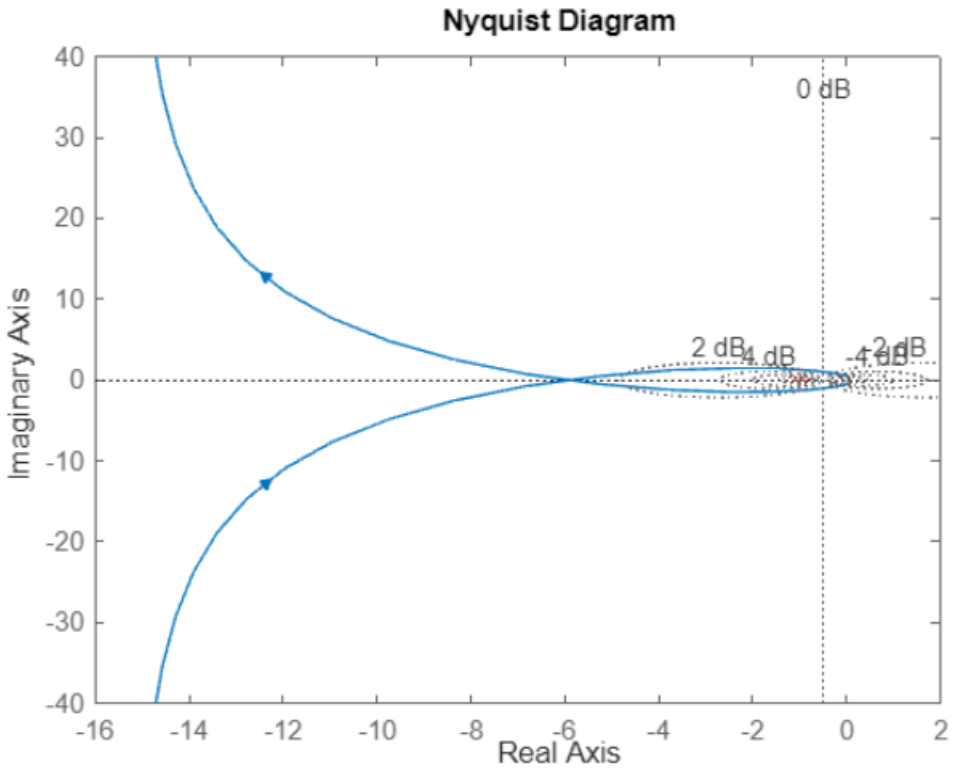
Mentre per quanto riguarda il **margine di fase**, questo è la distanza angolare tra la fase del sistema e - 180 gradi, quando il modulo $|L(j\omega)| = 1$, quindi quando il guadagno risulta essere *unitario*.

$$\Phi_m = 180 + \angle L(j\omega_{dB})$$

Si noti che ω_{dB} è la frequenza in cui il modulo del guadagno è pari a 1.

Questo lo si può notare ancora meglio con il diagramma di Nyquist:

```
nyquist(Ls)
grid
```



Quello che salta all'occhio è che il punto in rosso (-1) viene circondato, questo indica l'instabilità del sistema. Tuttavia era ipotizzabile dal momento che il guadagno aggiunto è molto alto. Non c'è da preoccuparsi, in quanto ci si può lavorare e renderlo stabile man mano che si vanno a soddisfare le specifiche di precisione dinamica.

3.5 Seconda richiesta

3.5.1 Precisione dinamica

Le specifiche di precisione **dinamica**, valutano il sistema non quando è a regime, bensì quando si trova in transitorio, e in particolare, queste specifiche si hanno sotto forma di : se ci si trova nel dominio del tempo :

- Sovraelongazione massima
- Tempo di assestamento t_s

- Tempo di salita t_r

se invece si è nel dominio delle frequenze :

- Margine di fase
- Banda passante

Per poter lavorare al meglio con queste specifiche è importante conoscere i legami che legano i due domini, che prendono il nome di **Legami globali**

L'ipotesi alla base della precisione dinamica e dei legami globali, è che F(s) possa approssimarsi ad un sistema del tipo :

$$\frac{\omega_n}{s^2 + 2\delta\omega_n * s + \omega_n}$$

Fatto un piccolo richiamo teorico, si ritorni alle richieste date dal suddetto problema :

- Sovraelongazione percentuale (S%) massima pari al 20%;
- Tempo di salita (t_r) ¡ 3 secondi.

3.5.2 Determinazione parametri con legami globali

Si può sfruttare fin da subito il legame globale che lega δ di F(s) che indica il coefficiente di *smorzamento*, e la Sovraelongazione S :

$$S = e^{(-\pi \cdot \frac{\delta}{\sqrt{(1-\delta^2)}})}$$

Da qui, assumendo che S = 0.2 (20 %) :

$$\delta = \frac{1}{\sqrt{1 + (\frac{\pi}{log(S)})^2}}$$

Lo si ricavi numericamente anche in matlab :

```
S = 0.2;
delta = 1/sqrt(1 + (pi/log(S))^2) %fattore di smorzamento
    desiderato
```

delta = 0.4559

Una volta che si è in possesso del valore relativo al fattore di smorzamento, si può procedere trovando il *lower bound* per la **banda passante**, sfruttando anche le caratteristiche del tempo di salita :

$$\omega_b = 0.8 \cdot \frac{\pi}{t_r}$$

```
time_rising = 3;
omega_b = 0.8 * pi / time_rising
```

$$\omega_b = 0.8378$$

Allo stesso modo, sfruttando un ulteriore risultato dei legami globali si individua anche il valore di Φ_m, dunque del margine di fase, che è legato proporzionalmente al δ :

```
%calcoliamo a questo punto il margine di fase
phi_m = 100*delta
```

$$\Phi_m = 45.5950$$

A questo punto va determinato un ω_c che risulti maggiore del *lower bound* che si era ricavato in precedenza, dunque $\omega_c > \omega_b$.

$$\omega_c = 0.88$$

;

Ovviamente i valori che possono essere presi in maniera arbitraria, seppur delimitati da vincoli, non sono poi così tanto "casuali", difatti sono state fatte diverse prove nel sistema, per renderlo al più possibile efficiente, come per esempio nella scelta del guadagno nel primo punto, si era tenuto conto dell'effetto che poteva avere sulla stabilità, difatti nonostante un k alto avrebbe diminuito il tempo di salita, avrebbe anche reso più complesso il processo di stabilizzazione del sistema in un secondo tempo.

Ad ogni modo anche per la determinazione di ω_c è stato fatto lo stesso ragionamento, difatti attribuirgli un valore troppo elevato avrebbe contribuito a diminuire il margine di fase, e quindi avrebbe influito anch'essa sulla stabilità(già critica) del sistema.

Fatta questa osservazione, si può passare al calcolo del margine di fase, e al modulo del sistema a ciclo aperto L(s) e confrontarle con le specifiche ricavate, è proprio in base ai risultati che si otterranno, che si sceglierà che tipo di ***rete correttrice*** adottare.

```
[M, phi] = bode(Ls, omega_c)
phi_corrente = 180 - abs(phi)
```

Quello che si ha è questo :

$$\begin{cases} |L(j\hat{\omega}_c)| \\ \pi - |\angle L(j\hat{\omega}_c)| \end{cases}$$

I valori che vengono fuori sono:

$$M = 21.1685$$

$$phi_{corrente} = 51.7694$$

Risulta che

$$\begin{cases} M > 1 \\ phi_{corrente} > \Phi_m \end{cases}$$

Quella che deve essere progettata è una rete attenuatrice.

3.5.3 Rete correttrice : attenuatrice

Una rete correttrice, è un controllore che ha la seguente struttura :

$$C(s) = \frac{k}{s^r} \cdot C_1(s)...C_l(s)$$

Dove :

- $\frac{k}{s^r}$ è relativo alla precisione statica
- $C_i(s)$ corrisponde a una coppia polo-zero che vanno a modificare il comportamento del sistema $L(j\omega)$ alle medie frequenze.

Quello che risulta dal sistema che è stato ricavato è che vanno attenuati entrambi i parametri, per farlo si avrà un controllore del tipo :

$$C_{lag}(s) = \frac{1 + s\alpha T}{1 + sT}$$

Vanno determinati α e T, per farlo si deve partire dalla pulsazione di attraversamento di progetto, la funzione di anello non-compensata è :

$$L(j\hat{\omega_c}) = Me^{-j\gamma}$$

Quello che si vuole è che in corrispondenza di questa pulsazione, la rete correttrice fornisca un m (attenuazione) ¡ M , ed un guadagno in fase negativo, e trascurabile, ma indispensabile.

Svolgendo il calcolo :

$$m = \frac{1}{M} \Rightarrow m = 0.0472$$

Mentre come detto prima il valore del guadagno in fase θ, lo si può scegliere :

$$\theta = -1.2$$

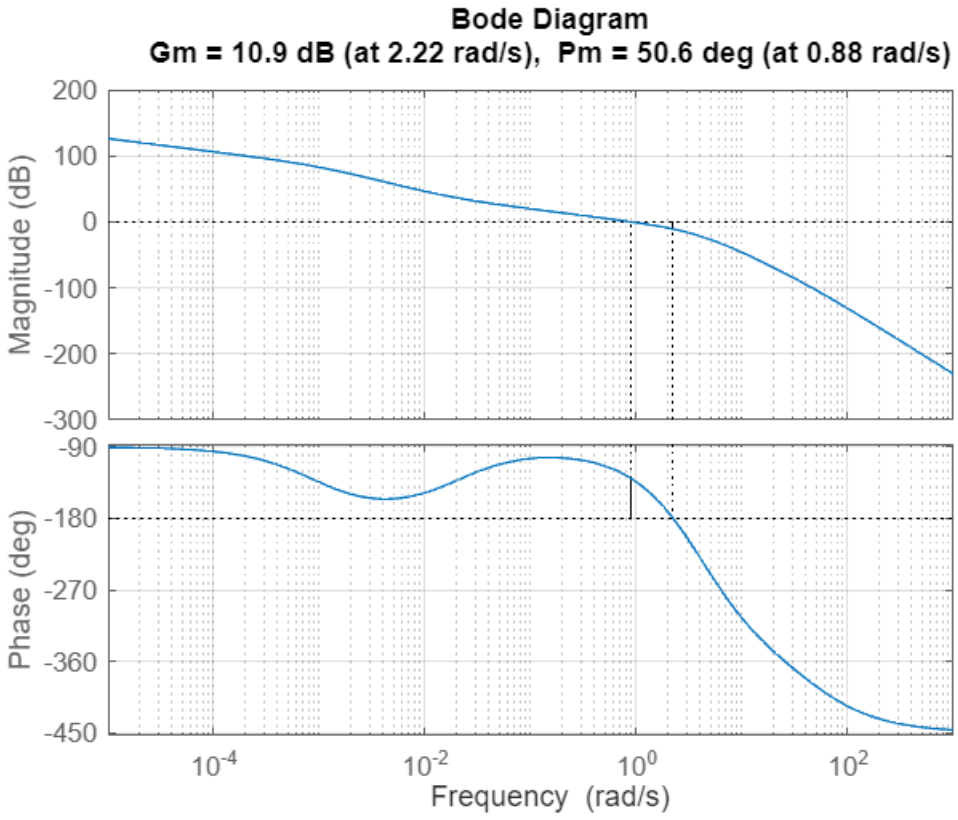
Ora, attraverso varie derivazioni geometriche si arriva al calcolo finale di e T :

```
alpha = m*(cos(deg2rad(theta))-m)/(1-m*cos(deg2rad(theta)))
T = (1/omega_c)*sqrt((1-m^2)/(m^2-alpha^2))
Clag = (1+s*alpha*T)/(1+s*T)
```

3.5.4 Valutazione sulla nuova L(s) e su F(s)

A questo punto si hanno tutti gli espedienti necessari per la verifica finale del sistema, L(s) corrisponderà al prodotto tra la il controllore che si è appena ricavato come rete correttrice, e la L(s) precedente, quella che teneva conto solamente delle caratteristiche statiche, una volta fatto questo passaggio si passi alla verifica grafica del sistema, anche attraverso i vari diagrammi di Bode e di Nyquist per capire se il sistema si è stabilizzato.

```
Ls = Clag * Ls
margin(Ls)
grid
```



Ora sia il margine di fase che quello di ampiezza risultano non negativi, quindi si è riusciti a portare il sistema alla stabilità, anche se come si può notare il margine di ampiezza è molto piccolo, margine di ampiezza e di fase danno informazioni riguardo la stabilità relativa

del sistema, quindi quanto è robusto un sistema, dunque questo ottenuto potrebbe essere poco robusto, ma comunque risulta stabile. Resta da capire se le specifiche, che erano il principale obbiettivo, sono soddisfatte, si osservino dunque le caratteristiche di F(s)

```
Fs = minreal(feedback(Ls,1))
step(Fs)
grid
stepinfo(Fs)
[y,t] = step((1/s)*Fs); %faccio il gradino di Fs * 1/s
r = t;
plot(t, r - y)
grid
```

```
ans = struct with fields:
         RiseTime: 1.2297
    TransientTime: 9.1515
     SettlingTime: 9.1515
      SettlingMin: 0.9019
      SettlingMax: 1.1899
        Overshoot: 18.9862
       Undershoot: 0
             Peak: 1.1899
         PeakTime: 3.0239
```

Nelle specifiche abbiamo il ***RiseTime*** che indica il tempo di salita e risulta abbondantemente sotto il tempo di 3 secondi, e poi l'***Overshoot***, che invece è la sovraelongazione percentuale, e anche questa è sotto il 20%.

Tutte le specifiche richieste sono state rispettate, non rimane altro che concludere con la discretizzazione del segnale, parte del processo molto importante che ci permette di avere un controllore digitale.

3.5.5 Discretizzazione

Per discretizzare il segnale esistono, come già anticipato differenti metodi :

- Eulero in avanti.
- Eulero all'indietro.
- Tustin.

Questi metodi nascono dall'esigenza di approssimare il mapping che si fa per passare dal continuo al discreto; tecnicamente il mapping sarebbe questo:

$$z = e^{sT_c}$$

Ora, utilizzando questo mapping, ipotizzando di avere un cotrollore del tipo :

$$C(s) = \frac{1+\alpha s}{1+\beta s}$$

Verrebbe un elemento logaritmico al denominatore, dunque non si starebbe più parlando di una funzione razionale fratta.

Per capire meglio come funzionano i tre metodi e l'accuratezza della loro discretizzazione, alterneremo anche delle operazioni su un file simulink.

3.5.6 Metodo Eulero in avanti

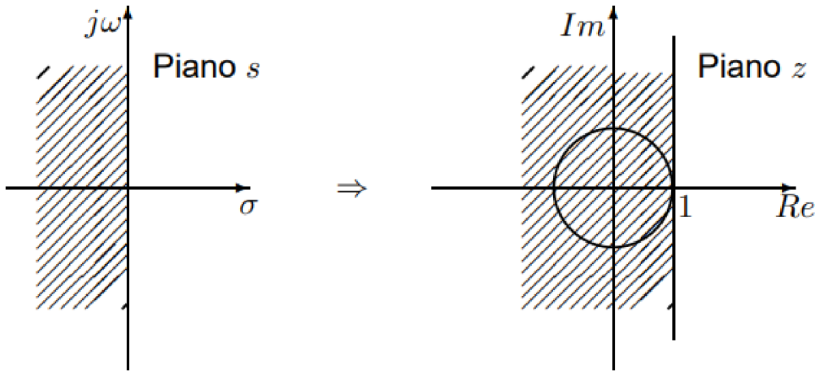
Si parta dal primo metodo, questo va ad approssimare il mapping nel seguente modo :

$$z = e^{sTc} \simeq 1 + sT_c$$

Dunque la singola s diventerà :

$$s = \frac{z-1}{T_c}$$

Il problema sta nel fatto, che quando si utilizza un mapping esatto, la circonferenza di raggio unitario corrisponde all'interno del semipiano sinistro, in questo mapping invece il semipiano sinistro del piano s si mappa all'interno del cerchio $|z-1| \leq 1$ nel piano z.

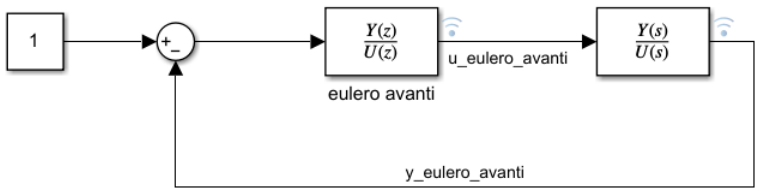


Osserviamo gli effetti che ha questo metodo, dapprima numericamente, dunque all'interno del live script di Matlab e poi graficamente, implementando il modello simulink.

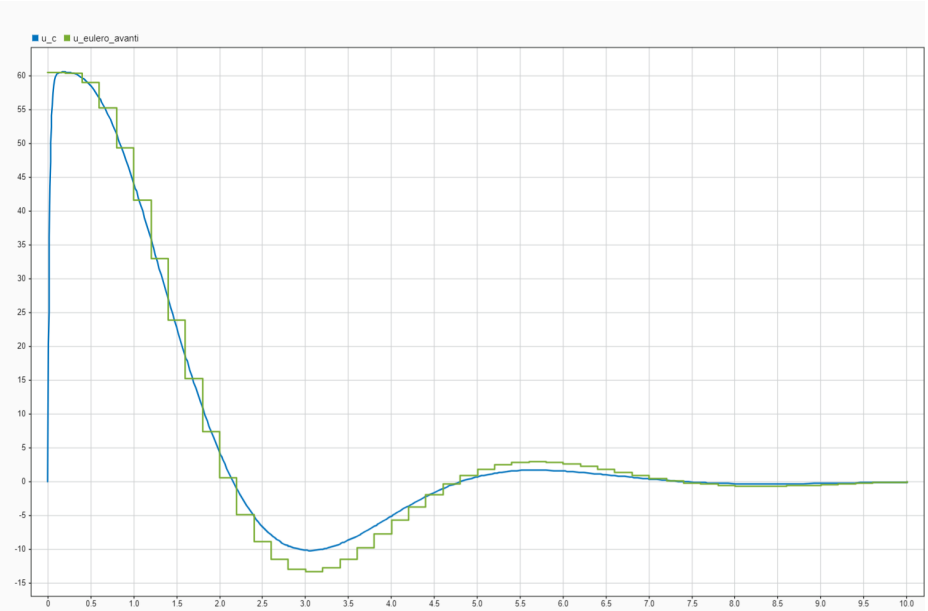
```
%eulero in avanti
s = (z - 1)/(Ts)
Cs_eulero_avanti = minreal((6.616e04*s + 1280)/(1094*s + 1))
[numeratore_eulero_avanti, denominatore_eulero_avanti] =
   tfdata(Cs_eulero_avanti, 'v');
```

$$Cs\_eulero\_avanti = \frac{60.48z - 60.43}{z - 1}$$

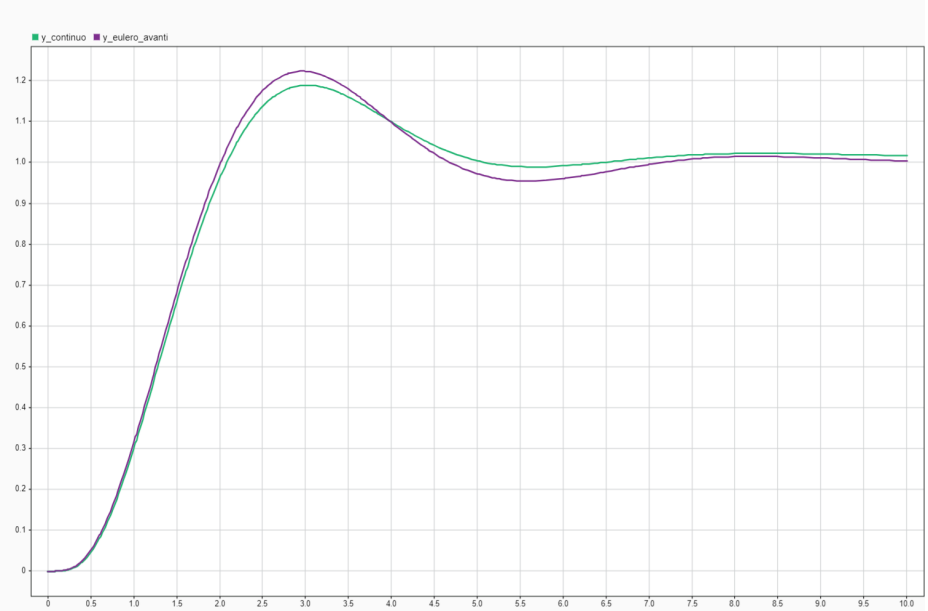
Lo schema di simulink è il seguente :



Qui si notino gli ingressi (quello in continuo e quello con eulero in avanti)



Mentre queste sono le risposte



3.5.7 Metodo Eulero indietro

Il mapping qui è approssimato come:

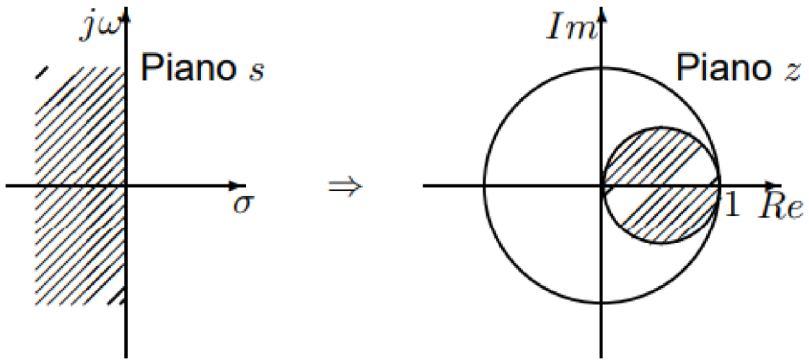
$$s = \frac{z-1}{z \cdot T_c}$$

Si implementi ora tutto in matlab

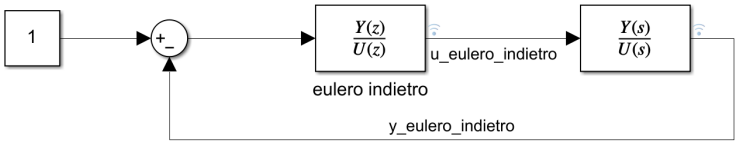
```
%eulero indietro
z = tf('z');
%s = (z - 1)/(z*Ts)
Cs_eulero_indietro = minreal((6.616e04*((z - 1)/(z*Ts)) +
    1280)/(1094*((z - 1)/(z*Ts)) + 1))
[numeratore_eulero_indietro, denominatore_eulero_indietro] =
    tfdata(Cs_eulero_indietro, 'v');
```

$$Cs\_eulero\_avanti = \frac{60.52z - 60.47}{z-1}$$

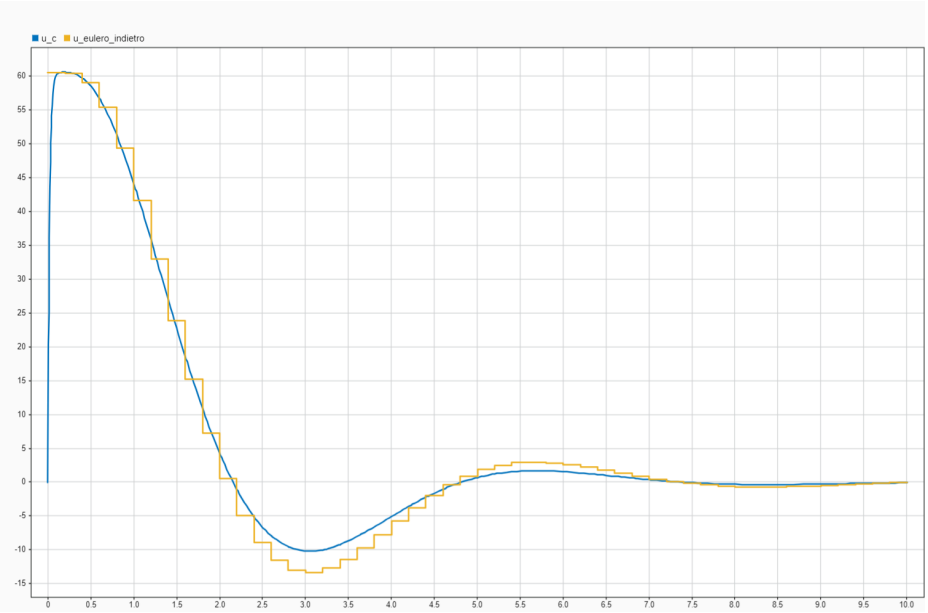
Questa volta la regione di convergenza si trova all'interno di un cerchio unitario con $|z| \leq 1$

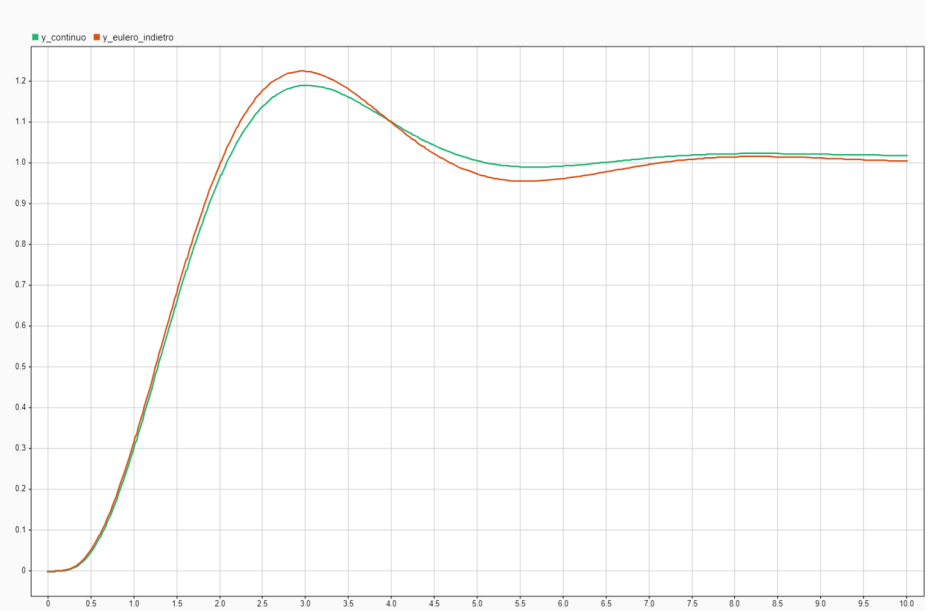


Schema:



Grafici:





Come si può facilmente notare, dato che risultano molto simili Cs_avanti e Cs_indietro la differenza nei grafici non è troppa.

3.5.8 Metodo Tustin

Questo ultimo metodo è anche conosciuto come *bilineare* e la sua mappatura è la seguente :

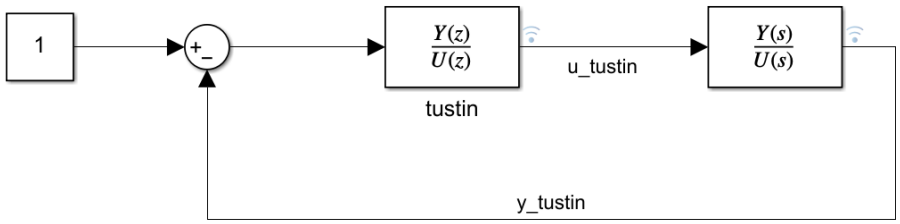
$$s = \frac{2}{T_c} \cdot \frac{z-1}{z+1}$$

Questo a differenza degli altri due ha già una funzione, che permette di implementarlo in matlab.

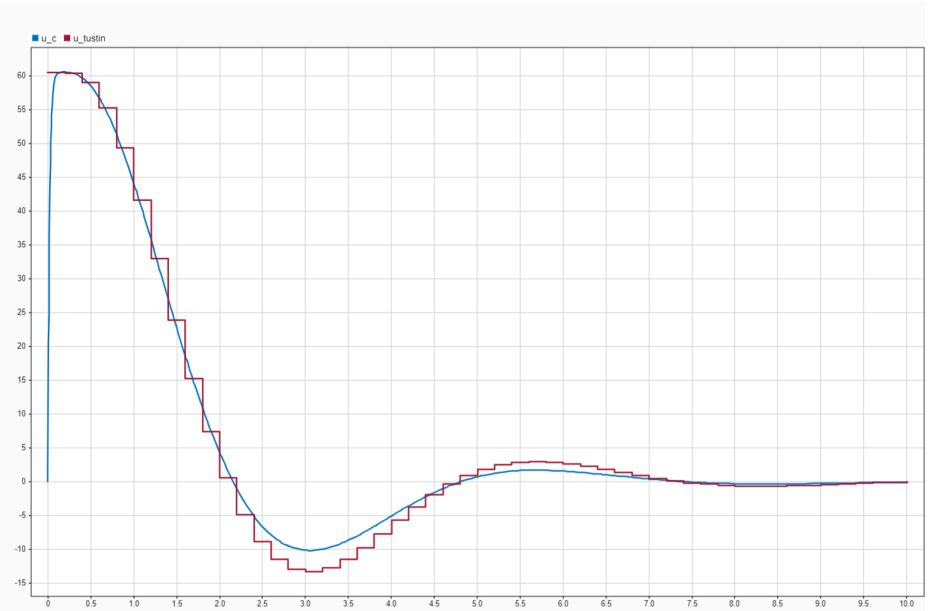
```
%tustin
Cs_tustin = c2d(Cs,Ts,'tustin')
[numeratore_tustin, denominatore_tustin] = tfdata(Cs_tustin,
    'v');
```

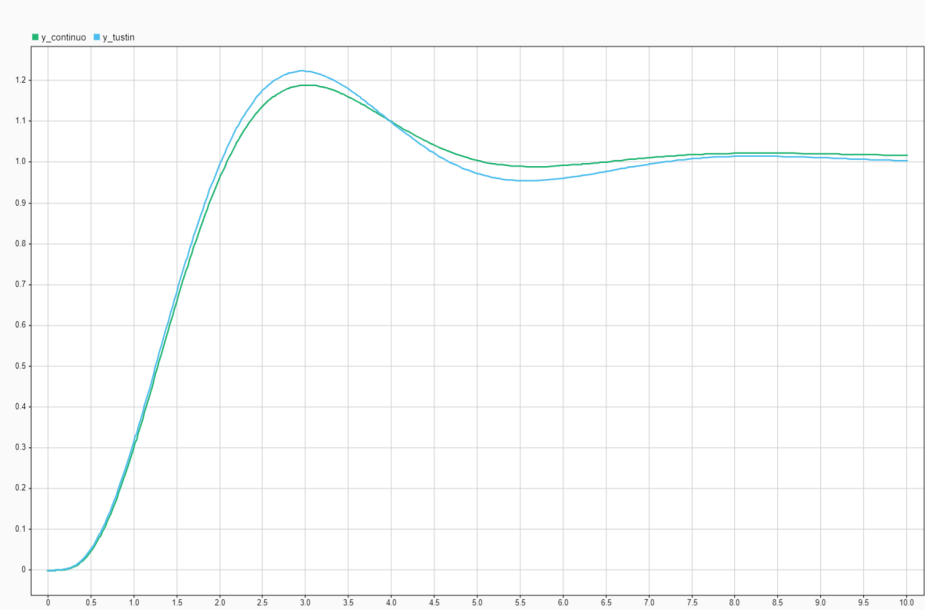
$Cs\_tustin = \frac{60.47z-60.43}{z-1}$

Implementazione


1
+
_
$\frac{Y(z)}{U(z)}$
tustin
u_tustin
$\frac{Y(s)}{U(s)}$
y_tustin

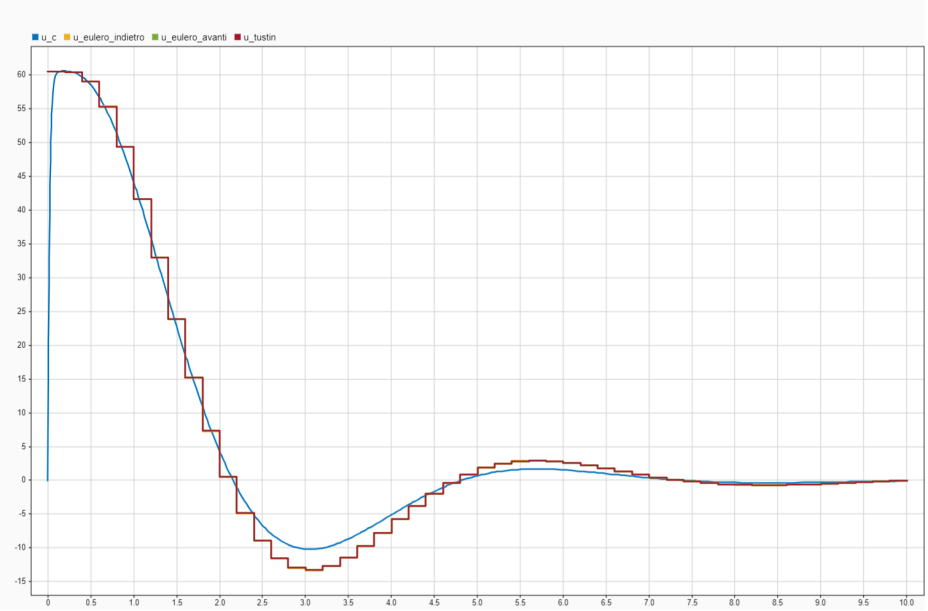
Grafici :


u_c
u_tustin

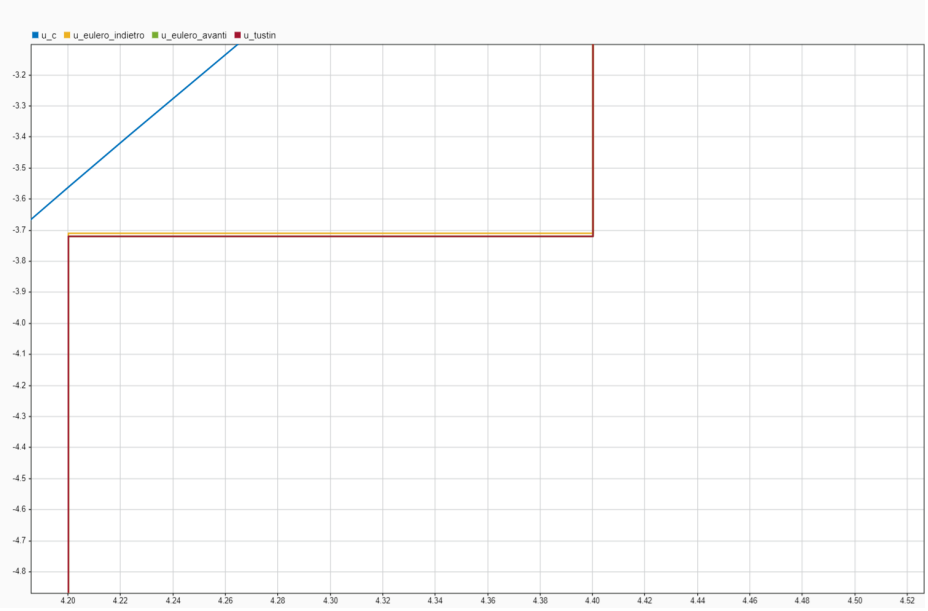

y_continuo
y_tustin

3.5.9 Confronto finale

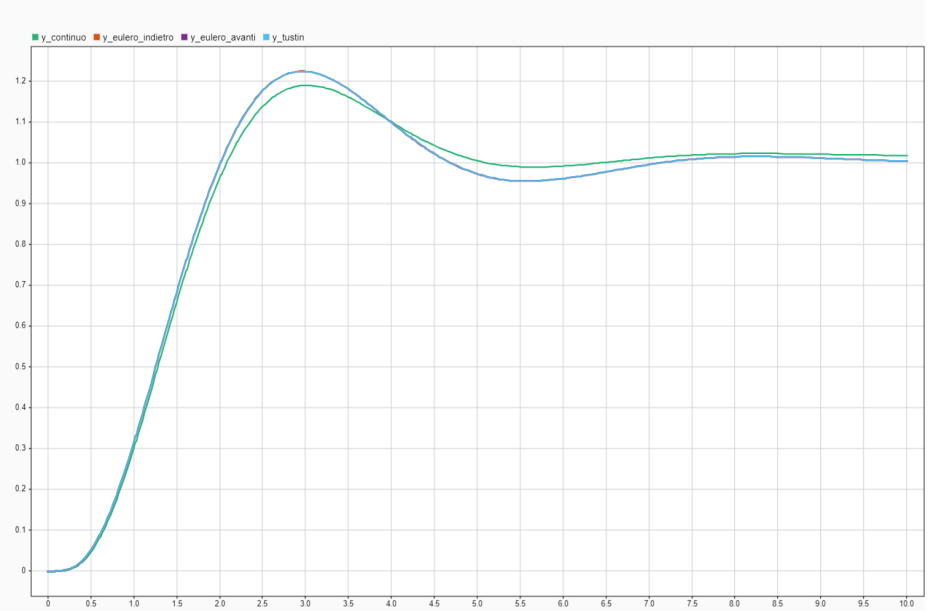
Per ultima cosa, vengano messi a confronto tutti e tre i metodi di discretizzazione, all'interno dello stesso grafico.



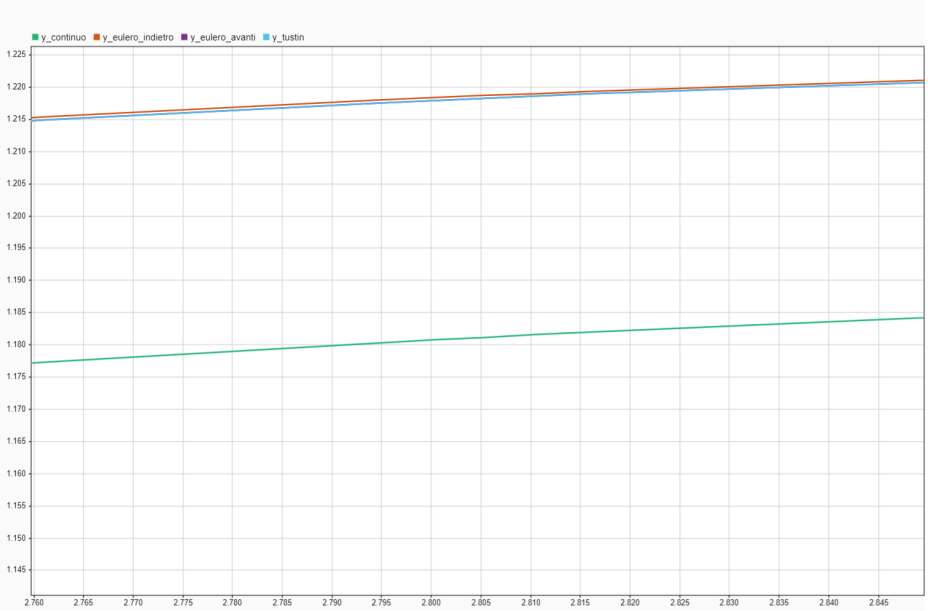
Come già detto prima non sembra quasi esserci differenza, nonostante tutti e tre i metodi sono "attivati" nel grafico, dunque facciamo dello zoom per vedere meglio.



Osserviamo ora le risposte:



Anche qui mettiamo dello zoom



3.6 Conclusione

Si conclude dunque la progettazione del controllore digitale, con tutte le caratteristiche che rispettano le richieste iniziali, c'è da dire che con un tempo di campionamento piccolo, ma comunque scelto consapevolmente tutti e tre i metodi di discretizzazione risultano essere ampiamente adatti allo scopo; non è sempre così difatti solitamente bisogna capire cosa si vuole guadagnare e perdere per il proprio controllore. Un metodo di mapping come quello di eulero in avanti è adatto per sistemi lenti e non garantisce la stabilità a quei sistemi che hanno i poli vicino alla frontiera del semipiano sinistro, il mapping dell'eulero all'indietro è invece più robusto e più adatto ai sistemi veloci, mentre il metodo di Tustin

offre molta stabilità.