



Progetto di Laboratorio di Automatica

Giorgio Ubbriaco
209899



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES

Indice

INTRODUZIONE	4
ESERCIZIO 1	5
RAPPRESENTAZIONE INGRESSO-STATO-USCITA DEL SISTEMA	5
ANALISI DEL SISTEMA A CICLO APERTO E A CICLO CHIUSO	5
TARATURA DEL TEMPO DI CAMPIONAMENTO T_c	6
METODO DI ZIEGLER-NICHOLS AD ANELLO CHIUSO	7
CALCOLO DEL K CRITICO PER ISPEZIONE SUPPONENDO DI NON CONOSCERE IL SISTEMA	8
CONFERMA DEI CALCOLI SUPPONENDO DI CONOSCERE IL SISTEMA	9
TARATURA PARAMETRI DEL REGOLATORE	11
<i>Azione proporzionale</i>	12
<i>Azione integrale</i>	12
<i>Azione derivativa</i>	13
REGOLATORE STANDARD P	13
<i>P analogico</i>	13
<i>P digitale</i>	14
<i>Analisi dei plot</i>	14
REGOLATORE STANDARD PI	16
<i>PI analogico</i>	16
<i>PI digitale</i>	16
<i>Analisi dei plot</i>	17
REGOLATORE STANDARD PID	19
<i>PID analogico</i>	19
<i>PID digitale</i>	19
<i>Analisi dei plot</i>	20
REGOLATORI STANDARD ANALOGICI	22
REGOLATORI STANDARD DIGITALI	23
PID TUNER	24
<i>Analogici</i>	25
<i>Digitali</i>	27
ESERCIZIO 2	30
MODELLISTICA E IDENTIFICAZIONE	30
LA SOLUZIONE AI MINIMI QUADRATI	31

ANALISI PRELIMINARE DI UN MODELLO ARMA GENERICO	33
VERIFICA LTI	34
GENERAZIONE MISURE DEL PROCESSO BLACK-BOX	36
COSTRUZIONE E ANALISI DEL MODELLO	37
IMPLEMENTAZIONE MODELLO ARMA	39
IMPLEMENTAZIONE FORMA CANONICA I	39
IMPLEMENTAZIONE FORMA CANONICA II	40
VALIDAZIONE MODELLO	41
ANALISI DELLE RISPOSTE	42
<i>Risposta all'ingresso random</i>	42
<i>Risposta al gradino</i>	43
<i>Risposta al treno di impulsi</i>	43
<i>Risposta alla sinusoide</i>	44
ESERCIZIO 3	46
Cos'è il campionamento?	46
Cos'è una sequenza periodica? E come si ricava la DFT?	49
Cos'è la rappresentazione spettrale di un segnale?	50
La DFT in MATLAB	51
Segnale non noto	54
<i>Dispersione Spettrale</i>	54
<i>Funzione di Hann</i>	56
<i>Funzione di Bartlett</i>	57
<i>Finestratura</i>	58

Introduzione

Svolgere il progetto descritto dalla seguente traccia:

1. Dato il sistema descritto dalla seguente funzione di trasferimento

$$G(s) = \frac{-s + 2}{(s + 0.3)(s + 4)}$$

progettare un regolatore standard digitale in grado di garantire stabilità a ciclo chiuso ed errore nullo rispetto ad un riferimento a gradino.

Passi da svolgere:

- a. Implementare in Simulink il sistema;
 - b. Tarare i parametri per regolatori standard del tipo P, PI e PID, ipotizzando di NON conoscere il modello dell'impianto;
 - c. Discretizzare i regolatori ottenuti dopo opportuna scelta del passo di campionamento.
-
2. Si consideri il processo contenuto nello schema Simulink localizzato al seguente link:

https://www.dropbox.com/sh/3gmw9bnz2nd02xj/AAB4rID3qjNxiMQC_Qcy-8Na?dl=0

Approssimare la dinamica del processo con un modello ARMA di ordine 4 identificando opportunamente i parametri. Determinare successivamente il grafico della risposta al gradino attraverso le tre forme canoniche (ARMA, Numero minimo di registri I e II).

3. Sia dato il seguente segnale periodico

$$x(t) = \sin\left(2\pi t + \frac{\pi}{6}\right) + \cos\left(2\pi 8t + \frac{\pi}{3}\right)$$

scrivere uno script matlab che, dopo opportuna scelta del passo di campionamento e dell'intervallo di osservazione, mostri le ampiezze e le fasi del segnale mediante l'algoritmo DFT.

Esercizio 1

Considerando il sistema descritto dalla seguente funzione di trasferimento

$$G(s) = \frac{-s + 2}{(s + 0.3)(s + 4)}$$

implemento la rappresentazione ingresso-stato-uscita così da essere utilizzata successivamente nel modello Simulink.

Rappresentazione ingresso-stato-uscita del sistema

La rappresentazione ingresso-stato-uscita (*ISU*) è un altro modo per descrivere il sistema oltre alla rappresentazione tramite funzione di trasferimento. Essa si presenta nella seguente forma:

$$\text{processo} = \begin{cases} \dot{x}(t) = f(x(t), u(t), t) \\ y(t) = g(x(t), u(t), t) \end{cases} = \begin{cases} \dot{x}(t) = A(t) \cdot x(t) + B(t) \cdot u(t) \\ y(t) = C(t) \cdot x(t) + D(t) \cdot u(t) \end{cases} \rightarrow \begin{array}{l} \text{equazione di stato} \\ \text{trasformazione di uscita} \end{array}$$

Pertanto, implemento il modello *ISU* in MATLAB tramite il comando `ss(transfer_function)` il quale mi permetterà di ottenere la rappresentazione ingresso-stato-uscita passando come argomento la funzione di trasferimento in questione:

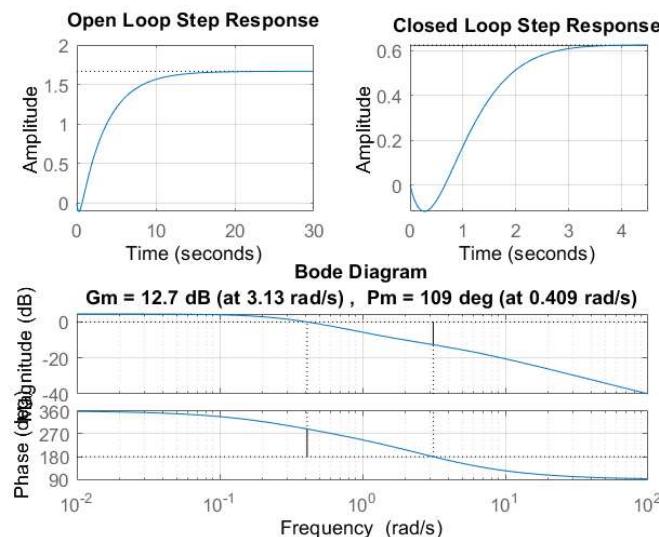
```
clear; clc; close all;
num = [-1 2]; den = [1 4.3 1.2]; G = tf(num,den);
processo = ss(G);
A = processo.A; B = processo.B; C = processo.C; D = processo.D;
```

Analisi del sistema a ciclo aperto e a ciclo chiuso

Successivamente analizzo il comportamento del sistema considerando la risposta a ciclo aperto, la risposta a ciclo chiuso e, infine, il diagramma dei moduli e delle fasi:

```
figure(1);
subplot(2,2,1); step(G); title( "Open Loop Step Response" );
subplot(2,2,2); step(feedback(G,1)); title( "Closed Loop Step Response" );
subplot(2,2,[3,4]); margin(G); grid;
```

ottenendo i seguenti plot:



Possiamo notare che la risposta a ciclo aperto, che rappresenta la situazione nella quale il sistema la cui azione di controllo è libera dall'output, si assesta ad un valore di regime compreso tra 2 e 1.5. Ovviamente, considerare semplicemente tale risposta non va bene poiché essa è un'analisi superficiale per uno studio di un sistema (non affidabile) e proprio per questo motivo si considera anche la risposta a ciclo chiuso. Quest'ultima risulta essere meno stabile in generale a causa del feedback però risulta essere molto più affidabile rispetto al sistema a ciclo aperto. In questo caso la risposta del sistema a ciclo chiuso risulta assestarsi ad un valore di regime pari poco più di 0.6. Infine, il diagramma di Bode, composto dal diagramma dei moduli e delle fasi, ci permette di condurre un'analisi della risposta in frequenza del sistema. Possiamo notare che il margine di guadagno Gm^1 è pari a 12.7 dB in corrispondenza della frequenza $Wcg = 3.13 \text{ rad/s}$ (*relativa al margine di guadagno*), ed il margine di fase Pm^2 risulta essere pari a 109 deg in corrispondenza della pulsazione critica $\omega_c = 0.409 \text{ rad/s}$. Essendo il margine di fase positivo possiamo dedurne che il sistema in catena chiusa è stabile. Dopo aver definito e analizzato il sistema, posso ora procedere con la scelta del tempo di campionamento che sarà fondamentale nella taratura del regolatore standard digitale.

Taratura del tempo di campionamento Tc

Il tempo di campionamento Tc è un parametro di fondamentale importanza perché influisce in maniera significativa sulle prestazioni di un sistema di controllo a dati campionati. Ovviamente, la taratura di questo parametro è parecchio delicata poiché bisogna considerare altri parametri, legati al sistema in questione, affinché esso soddisfi qualsiasi situazione possa avvenire. In particolare, bisogna focalizzare l'attenzione sulla pulsazione di banda passante³ ω_{bw} e sul tempo di salita⁴ t_r . Tanto è vero che Tc è legato a tali parametri tramite le seguenti diseguaglianze:

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

$$\frac{\pi}{50\omega_{bw}} \leq T_c \leq \frac{\pi}{10\omega_{bw}}$$

¹ Il margine di guadagno è un indicatore del grado di robustezza della stabilità del sistema in catena chiusa a fronte di possibili incertezze o variazioni del guadagno della funzione d'anello.

² Il margine di fase è un indicatore del grado di robustezza della stabilità del sistema in catena chiusa a fronte di possibili incertezze o variazioni della fase della funzione d'anello alla pulsazione critica ω_c o a fronte della presenza di eventuali ritardi di tempo.

³ La pulsazione di banda passante indica la frequenza approssimativa della larghezza di banda di un sistema.

⁴ Il tempo di salita di un sistema è un indice della velocità di risposta del controllo, cioè il tempo necessario al sistema per variare dal 10% al 90% del valore di regime dello stesso.

Pertanto, calcolo t_r , ω_{bw} e anche il tempo di assestamento⁵ t_s che mi servirà in seguito:

```
roots = roots(den);
if roots(1)>=roots(2)
    polo_dominante = roots(1);
else
    polo_dominante = roots(2);
end
cost_temp_polo_dominante = -1/polo_dominante;

ta = 3*cost_temp_polo_dominante;
wbt = 1/cost_temp_polo_dominante;
tr = (0.8*pi)/wbt;

estremo_sinistro_trTc = tr/20;
estremo_destro_trTc = tr/10;

estremo_sinistro_wbtC = pi/(50*wbt);
estremo_destro_wbtC = pi/(10*wbt);
```

Dai seguenti calcoli ottengo le seguenti diseguaglianze:

$$0.418879020478639 \leq T_c \leq 0.837758040957278$$

$$0.209439510239320 \leq T_c \leq 1.047197551196598$$

e scelgo un tempo di campionamento $T_c = 0.45$:

$T_c = 0.45$;

Essendo che il nostro obiettivo è quello di tarare i PID supponendo di non conoscere il sistema in questione, dobbiamo decidere quale metodo utilizzare per stimare i parametri caratteristici del controllore. A tal proposito scelgo di utilizzare il metodo di Ziegler-Nichols ad anello chiuso (*metodo del K critico*) il quale consiste nello stimare il guadagno critico tale che la curva polare $K_{cr}G(j\omega)$ passi per il punto critico $-1 + j0$.

Metodo di Ziegler-Nichols ad anello chiuso

L'obiettivo di tale metodo è quello di portare sulla soglia dell'instabilità il sistema in questione tale che la risposta ottenuta dal sistema presenta oscillazioni autosostenute. Infatti, si stima un guadagno critico K_{cr} (*praticamente un gain da applicare al sistema in questione*) tale per cui la curva polare risultante tra il margine di ampiezza in questione e il sistema passi per il punto critico del diagramma di Nyquist⁶ corrispondente. Quello che

⁵ Il tempo di assestamento è il tempo necessario alla risposta per portarsi definitivamente a valori vicini al valore di regime. Più precisamente dato in ingresso, ad un sistema dinamico asintoticamente stabile, un riferimento a gradino, esso è il tempo necessario affinché la risposta entri in una certa fascia di valori vicina al valore di regime con scostamenti pari al 5% senza più uscirne.

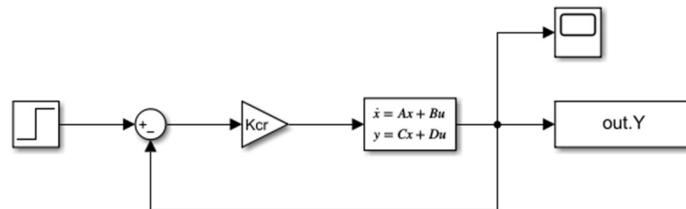
⁶ Il diagramma di Nyquist è una particolare rappresentazione grafica della funzione di trasferimento di un sistema dinamico utile all'analisi e alla verifica della stabilità dello stesso. Se il numero di giri N che la curva polare compie intorno al punto

otterremo sarà una configurazione di passaggio dalla stabilità all'instabilità a ciclo chiuso tale per cui la risposta del sistema presenterà oscillazioni autosostenute, cioè uguali per $t \rightarrow +\infty$. Tali oscillazioni, infatti, non devono né diminuire con il tempo perché significherebbe che la risposta del sistema per una t molto grande si assesta ad un determinato valore di regime, e, pertanto con un tempo di assestamento molto grande, né devono aumentare perché significherebbe che siamo di fronte ad una risposta "esplosiva" che comporterebbe la rottura dell'impianto.

Calcolo del K critico per ispezione supponendo di non conoscere il sistema

Ovviamente, essendo che dobbiamo presupporre di non conoscere il sistema, il calcolo del K critico deve essere effettuato tramite tentativi, cioè tramite *ispezione*, tale da verificare ogni volta che le oscillazioni ottenute siano autosostenute. Trovo un K critico pari a 4.29.

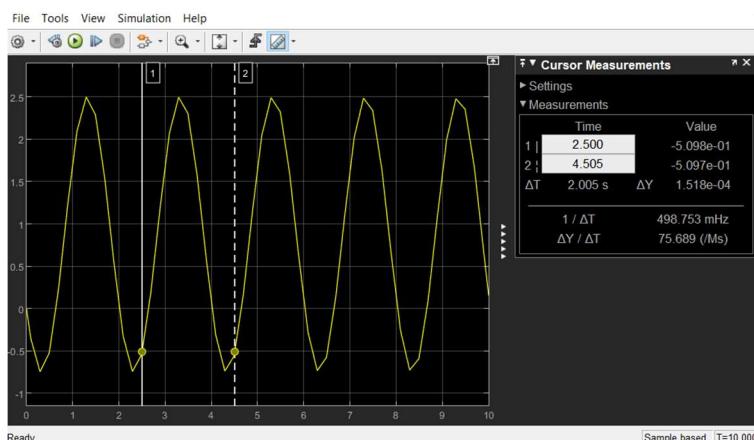
Pertanto, ora considerando il seguente sistema progettato in Simulink:



posso effettuare la simulazione di tale sistema per rendermi conto se effettivamente ho portato il sistema alla soglia dell'instabilità:

```
simulazione = sim('IMPIANTO_INSTABILE_Kcr');
```

Posso notare un tempo critico $T_{cr} = 2$.



```
Tcr = 2;
```

critico $-1 + j0$ è maggiore di zero allora significa che il sistema è instabile mentre se pari a zero allora significa che non sono presenti poli instabili nella funzione di trasferimento a ciclo chiuso ed il sistema in retroazione è stabile.

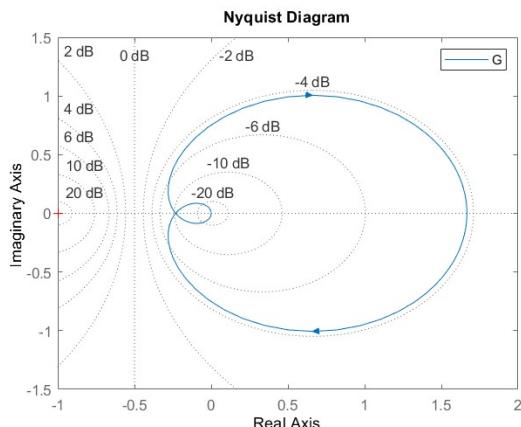
Invece, nel caso in cui siamo a conoscenza del sistema e, quindi, della funzione di trasferimento possiamo analizzare la curva polare e di conseguenza ottenere un calcolo molto più preciso del precedente.

Conferma dei calcoli supponendo di conoscere il sistema

Pertanto, avendo già calcolato K_{cr} e T_{cr} per ispezione presupponendo di non conoscere il sistema, analizzo la curva polare del sistema in questione così da dimostrare i valori ottenuti precedentemente:

```
figure(2); nyquist(G); grid; legend;
```

ottenendo il seguente plot:



Possiamo notare come i *giri* intorno al punto critico $-1 + j0$ sono praticamente pari a zero tale che si può concludere che il sistema è stabile.

Pertanto, ora, considerando il metodo del K critico possiamo procedere con i dovuti calcoli. Tenendo conto che il K critico, che porta il sistema alla soglia dell'instabilità, non è altro che il margine di ampiezza del sistema in questione, possiamo calcolare effettivamente il margine di ampiezza dello stesso considerando la seguente formula:

$$m_A = \frac{1}{|OA|}$$

dove OA è la lunghezza del segmento compreso tra l'origine ed il punto di intersezione della curva polare in questione. Pertanto, effettuo i dovuti calcoli in MATLAB nella seguente maniera tale da calcolare il margine di ampiezza, e, quindi, il guadagno critico in questione:

```
realNy = -0.232;
O = 0;
OA = O-realNy;
mA = 1/OA;
Kcr = mA;
(realNy è l'ascissa reale di intersezione sopra citata)
```

Tali calcoli possono essere confermati comunque tramite il comando MATLAB $\text{margin}(G)$ il quale se assegnato ad una variabile restituisce il margine di ampiezza del sistema descritto dalla funzione di trasferimento passata come argomento.

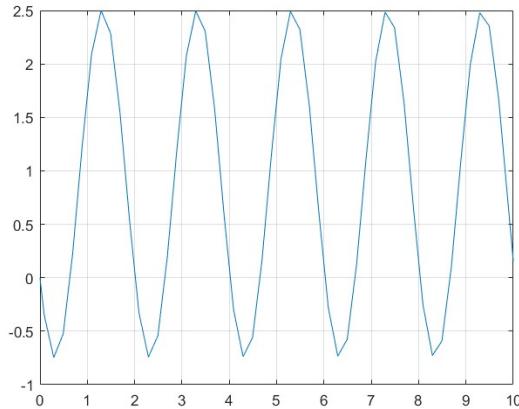
```
Kcr = margin(G);
```

Otteniamo un margine di ampiezza, e, quindi, un guadagno critico:

$$K_{cr} = 4.300122186679132$$

Plottando il grafico della risposta con il valore preciso otteniamo:

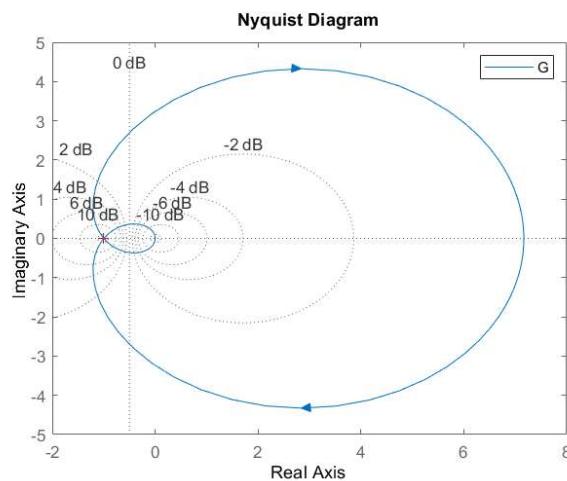
```
figure(4); plot(simulazione.Y.Time,simulazione.Y.Data); grid;
```



Possiamo notare come le oscillazioni sono autosostenute e perfettamente uguali per $t \rightarrow +\infty$. Inoltre, considerando il plot ottenuto tramite lo scope in simulink, posso notare che il periodo critico (*tempo critico*) T_{cr} di un'unica oscillazione è pressoché pari a 2. Come conferma che il sistema è stato portato sulla soglia dell'instabilità possiamo considerare il diagramma polare del sistema *totale* costituito dal gain K_{cr} e dal sistema in questione G :

```
figure(5); nyquist(Kcr*G); grid; legend('G');
```

ottenendo il seguente plot:



Possiamo notare come il sistema sia stato portato sulla soglia dell'instabilità. Tanto è vero che esso passa precisamente sul punto critico $-1 + j0$ senza oltrepassarlo anche perché altrimenti la curva polare compierebbe un giro intorno ad esso ed il sistema risulterebbe

essere instabile a catena chiusa (*criterio di Nyquist*). Pertanto, posso concludere che il valore del K critico calcolato durante la fase di ispezione risultava essere parecchio accurato.

Taratura parametri del regolatore

I parametri K_p , T_i e T_d del regolatore si determinano in via sperimentale tramite i metodi di taratura di Ziegler-Nichols:

	P	PI	PD	PID
K_p	$0.5 K_c \simeq \frac{1}{\mu \tau}$	$0.45 K_c \simeq \frac{0.9 \tau}{\mu \tau}$	$0.5 K_c \simeq \frac{1.2 \tau}{\mu \tau}$	$0.6 K_c \simeq \frac{1.2 \tau}{\mu \tau}$
T_i		$0.85 T_{cr} \simeq 3.3 \tau$		$0.5 T_{cr} \simeq 2 \tau$
T_d			$0.2 T_{cr} \simeq 0.3 \tau$	$0.12 T_{cr} \simeq 0.5 \tau$

I metodi di taratura più classici di Ziegler-Nichols danno in genere risultati poco soddisfacenti dal punto di vista delle prestazioni, anche se la stabilità è solitamente ottenuta. I metodi di taratura più avanzati, sia in anello aperto sia in anello chiuso come ad esempio quelli ad anello aperto di Cohen-Coon⁷ o di IMC⁸, sono preferibili ogni volta in cui sia necessario garantire migliori indici di robustezza e/o un migliore comportamento del sistema durante il transitorio.

Pertanto, definisco le costanti proporzionali, integratrici e derivative per ogni regolatore P, PI e PID utilizzando le costanti K_{cr} e T_{cr} calcolate in precedenza.

$$KP_P = 0.5 * K_{cr};$$

$$KP_PI = 0.45 * K_{cr};$$

$$TI_PI = 0.85 * T_{cr};$$

$$KP_PID = 0.6 * K_{cr};$$

$$TI_PID = 0.5 * T_{cr};$$

$$TD_PID = 0.12 * T_{cr};$$

$$N = 100;$$

$$T = ta/40;$$

(Le variabili N e T sono rispettivamente il Filter Coefficient ed il Sample Time che risulteranno utili nell'implementazione degli schemi in Simulink delle varie tipologie di regolatori standard)

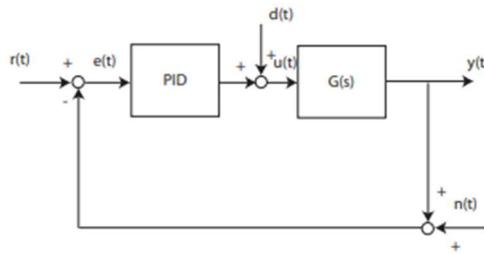
⁷ Il metodo in anello aperto di Cohen-Coon impone una condizione sulle oscillazioni durante il transitorio della risposta al gradino: impone che ogni picco sia smorzato del 25% rispetto al picco precedente.

⁸ Il metodo in anello aperto di controllo a modello interno (IMC) cerca di ricavare il regolatore confrontando il sistema da controllare $F(s)$ con il suo modello approssimato $G(s)$ tale da diminuire la banda passante ω_B e da aumentare i margini di guadagno m_G e di fase m_φ .

Ottenendo i seguenti valori:

$$\begin{aligned}KP_P &= 2.1501 \\KP_PI &= 1.9351 \\TI_PI &= 1.7000 \\KP_PID &= 2.5801 \\TI_PID &= 1.0000 \\TD_PID &= 0.2400\end{aligned}$$

Prima di procedere è utile definire e analizzare i comportamenti di ogni *azione* contenuta in ogni regolatore. Considerando il seguente schema di PID generico



possiamo descrivere le seguenti *azioni*:

Azione proporzionale

L'azione proporzionale in un regolatore standard è identificata dal guadagno proporzionale K_p il quale permette una riduzione dell'errore di inseguimento al gradino ed una riduzione dell'errore in uscita dovuto alla presenza di disturbi costanti sulla catena di controllo come ad esempio disturbi di carico o disturbi sull'uscita dell'impianto. Tanto è vero che è possibile azzerare l'errore di inseguimento a regime o l'errore sull'uscita con il solo utilizzo dell'azione proporzionale. L'azione proporzionale può essere identificata nel seguente ingresso al sistema:

$$u(t) = K_p e(t) = K_p(r(t) - y(t))$$

Azione integrale

Il vantaggio dell'azione integrale è legato alla possibilità di ottenere una operazione di reset automatico dell'errore di inseguimento a gradino senza dover conoscere necessariamente il valore dell'ampiezza del riferimento. Invece, lo svantaggio è che essa introduce uno sfasamento di 90° sulla risposta in frequenza della funzione di anello dando luogo a problemi seri sulla stabilità dello schema di controllo a causa dell'integratore elementare. L'azione integrale possiamo identificarla nel seguente ingresso al sistema:

$$u(t) = K_i \int_0^t e(\sigma) d\sigma$$

Azione derivativa

L'azione derivativa risulta essere fondamentale nel caso in cui viene combinata con l'azione proporzionale poiché riusciamo a dotare il regolatore di capacità anticipativa (*predittiva*) rispetto alla configurazione dell'errore di inseguimento. Dal punto di vista numerico, l'azione derivativa può essere identificata nel seguente ingresso al sistema:

$$u(t) = K_d \frac{d e(t)}{d t}$$

Dopo aver definito quali sono i contributi da considerare per tarare un regolatore standard, posso descrivere quali tipologie di regolatori standard ho implementato tale che per ognuno verrà descritto sia la tipologia continua sia la tipologia discreta:

(allego qui di seguito alcune righe di codice che saranno utili per il codice descritto nelle varie implementazioni dei regolatori standard sia digitali che analogici)

```
continuos = sim('continuos'); yc = continuos;
discrete = sim('discrete'); yd = discrete;
```

(tale codice mi permetterà di eseguire la simulazione in automatico sia del file di Simulink relativo ai PID analogici sia del file relativo ai PID digitali ed, inoltre, mi permetterà di plottare in maniera molto semplice i grafici relativi ai regolatori standard tramite le variabili "yc" ed "yd")

Regolatore Standard P

P analogico

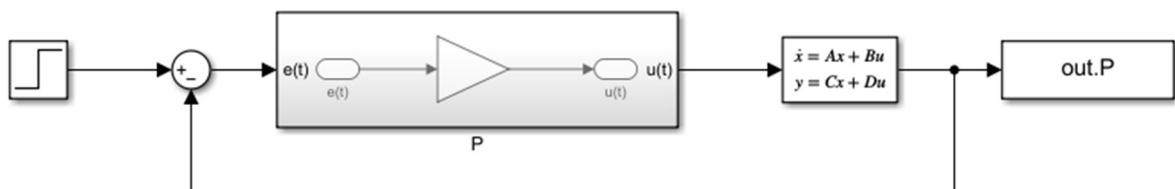
Esso è costituito soltanto dall'azione proporzionale. Tanto è vero che in Simulink esso può essere implementato tramite il *Gain Block* il quale verrà inizializzato al valore del K_P corrispondente al regolatore standard di tipologia P:



tale che la sua funzione di trasferimento sarà:

$$C_P(s) = K_P$$

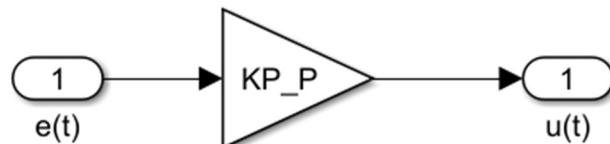
Pertanto, il corrispondente schema Simulink tale per cui il controllore P analogico verrà applicato al sistema generico supponendo di non conoscerlo sarà:



L'output corrispondente verrà salvato in un apposito *To Workspace Block* il quale ci permetterà di plottare i dati ottenuti nello script "esercizio1".

P digitale

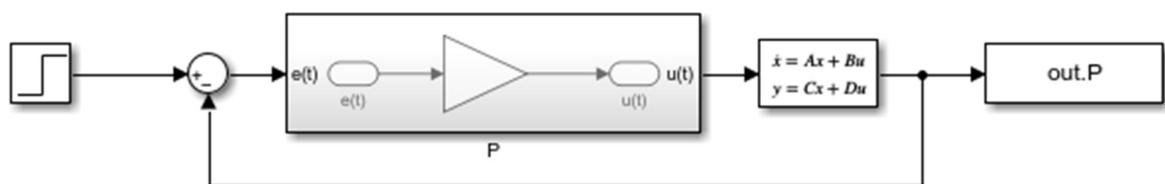
Essendo che l'unico parametro che caratterizza il regolatore standard in questione è il guadagno proporzionale, l'implementazione Simulink sarà la medesima del controllore standard P digitale:



tale che la funzione di trasferimento sarà la medesima della precedente:

$$C_P(z) = K_P$$

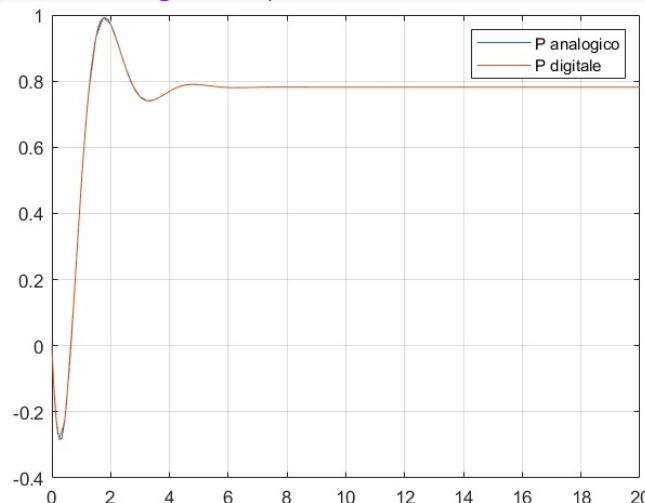
Ovviamente anche lo schema completo del sistema sarà il medesimo del regolatore standard P analogico:



Analisi dei plot

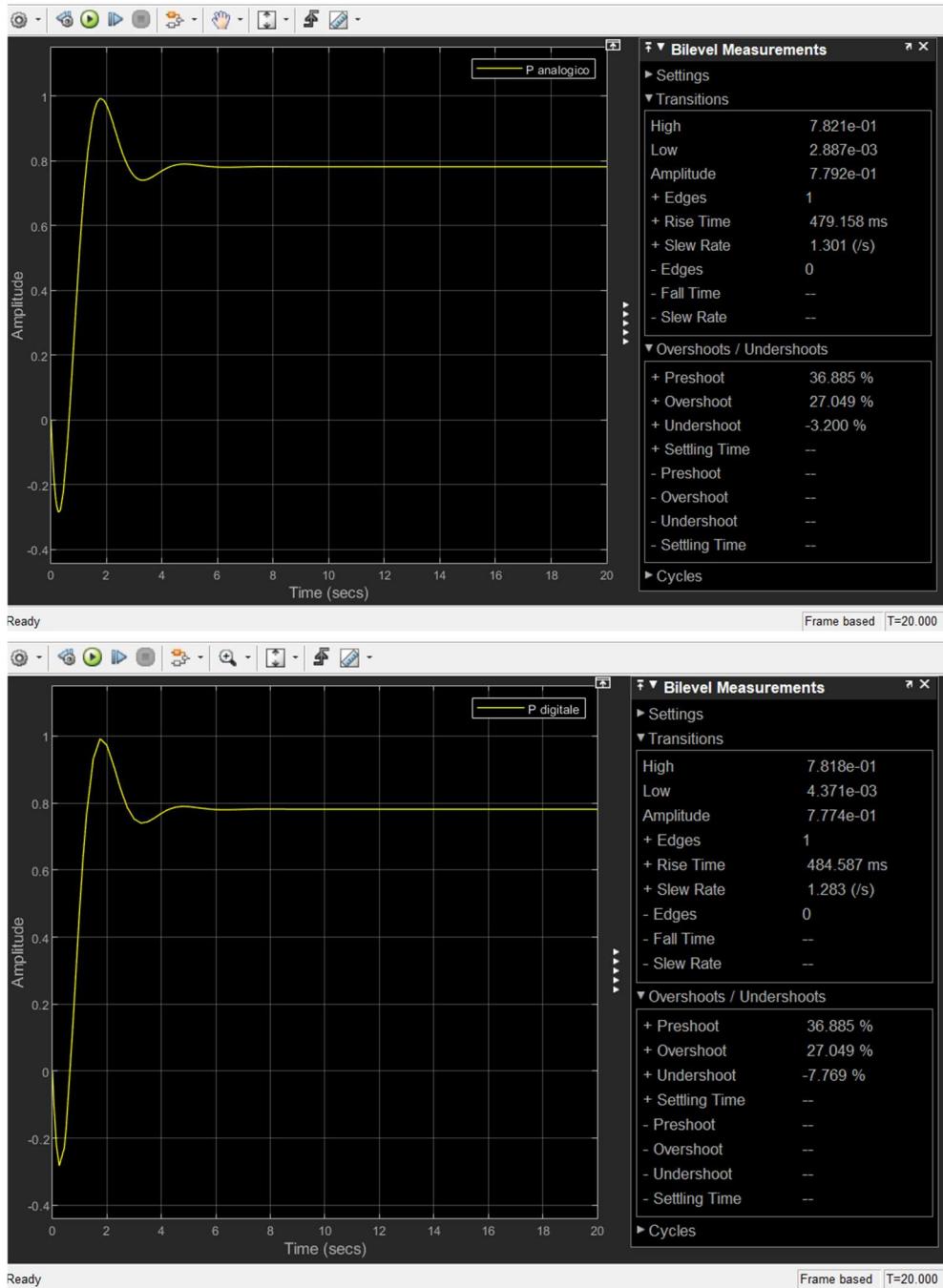
Allego qui di seguito il codice e i plot di esecuzione relativi al regolatore standard P analogico e al regolatore standard P digitale:

```
figure(6);
plot(yc.P.Time,yc.P.Data);
hold on
plot(yd.P.Time,yd.P.Data);
grid; legend('P analogico','P digitale');
```



Possiamo notare come i due plot rispettivamente relativi alla risposta del sistema dopo aver applicato P analogico e P digitale siano praticamente identiche. D'altronde come lo sono le due implementazioni in Simulink dei regolatori standard in questione. Inoltre,

analizzo più nello specifico i parametri che caratterizzano entrambi i plot tramite l'ausilio del *Time Scope Block* in Simulink:



Possiamo notare come i due plot sono pressocché identici. Tale risposta presenta un *overshoot* pari al 27%, un tempo di salita pari a 0.48 s e, inoltre, la *slew rate*, cioè la pendenza della linea che collega i livelli di riferimento del 10% e del 90% misurata per unità di tempo e quindi la velocità della risposta in questione, approssimativamente pari a 1.3.

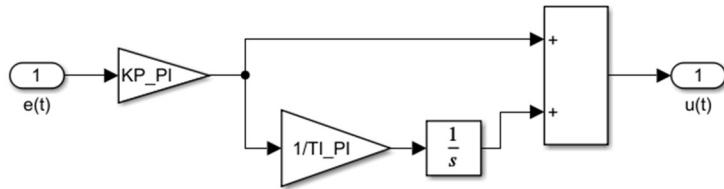
Regolatore Standard PI

PI analogico

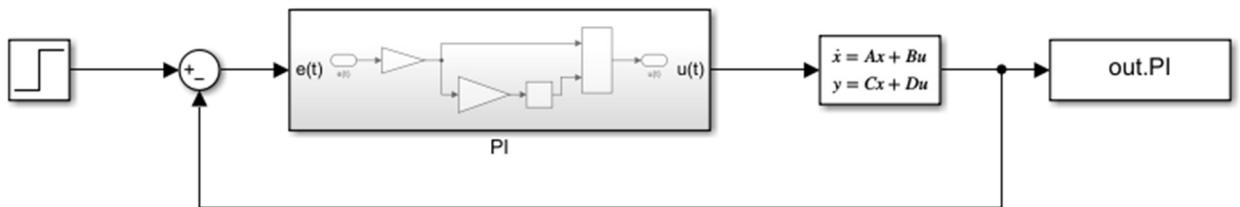
In questo caso il regolatore standard sarà costituito sia dall'azione proporzionale sia dall'azione integrale. Pertanto, la sua funzione di trasferimento sarà:

$$C_{PI}(s) = K_P \cdot \left(1 + \frac{1}{s \cdot T_I} \right)$$

tal che l'implementazione Simulink del *Block* corrispondente al PI analogico sarà



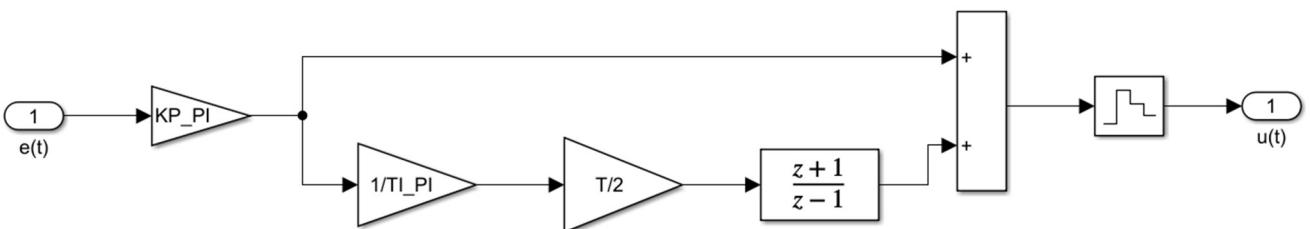
Quindi, il corrispondente schema Simulink tale per cui il controllore PI analogico verrà applicato al sistema generico supponendo di non conoscerlo sarà:



PI digitale

In questo caso, essendo che oltre all'azione proporzionale abbiamo anche l'azione integrale, dovremo aggiungere anche il ricostruttore ZOH⁹ che permette di ricostruire il segnale per campioni ed effettivamente ottenere il segnale corrispondente. Pertanto, la funzione di trasferimento corrispondente al controllore PI digitale sarà:

$$C_{PI}(z) = K_P \left(1 + \frac{1}{T_I} \cdot \frac{T}{2} \cdot \frac{z+1}{z-1} \right)$$



⁹ ZOH è l'acronimo di Zero Order Hold cioè un modello matematico utile per la ricostruzione del segnale. Tale ricostruzione viene eseguita da un convertitore digitale-analogico (DAC) convenzionale. Supponendo di avere un segnale campionato $f(kT_c)$, il segnale ricostruito $f_0(t)$ sarà approssimativamente ricostruito al segnale $f(kT_c) \cdot [sca(t - kT_c) - sca(t - (k + 1) \cdot T_c)]$ dove $sca(t)$ indica la successione canonica del gradino unitario.

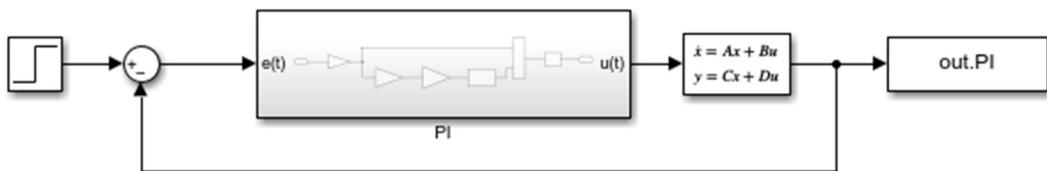
Possiamo notare nella funzione di trasferimento e ovviamente anche nello schema simulink due termini: il termine T che indica il tempo di campionamento considerato ed inizializzato al quoziente tra il tempo di assentamento t_a e 40 ed il *Discrete Transfer Function Block* che corrisponde alla trasformata Z del termine integrale $\frac{1}{s}$ applicando la trasformazione di Tustin. Ovviamente la trasformata di Tustin corrisponde alla serie di questi due termini appena citati, cioè la trasformazione di Tustin ci dice che il cambio di variabile da L-Trasformata a Z-Trasformata avviene secondo la seguente relazione:

$$s = \frac{2}{T} \cdot \left(\frac{z - 1}{z + 1} \right)$$

tale che

$$\frac{1}{s} = \frac{T}{2} \cdot \left(\frac{z + 1}{z - 1} \right) = \frac{t_a}{(40 \cdot 2)} \cdot \left(\frac{z + 1}{z - 1} \right)$$

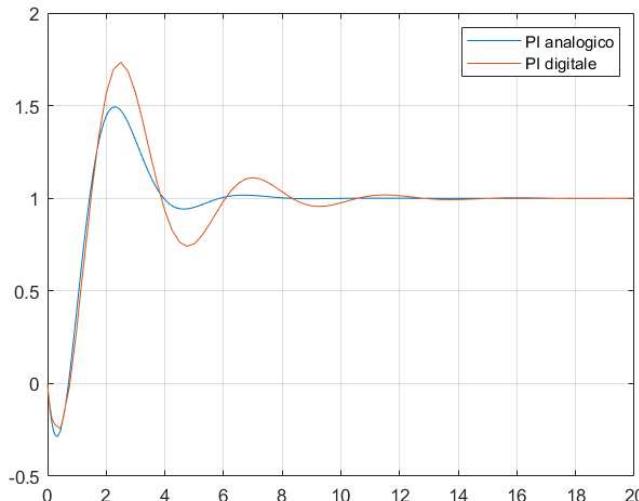
Pertanto, il corrispondente schema Simulink tale per cui il controllore PI digitale verrà applicato al sistema generico supponendo di non conoscerlo sarà:



Analisi dei plot

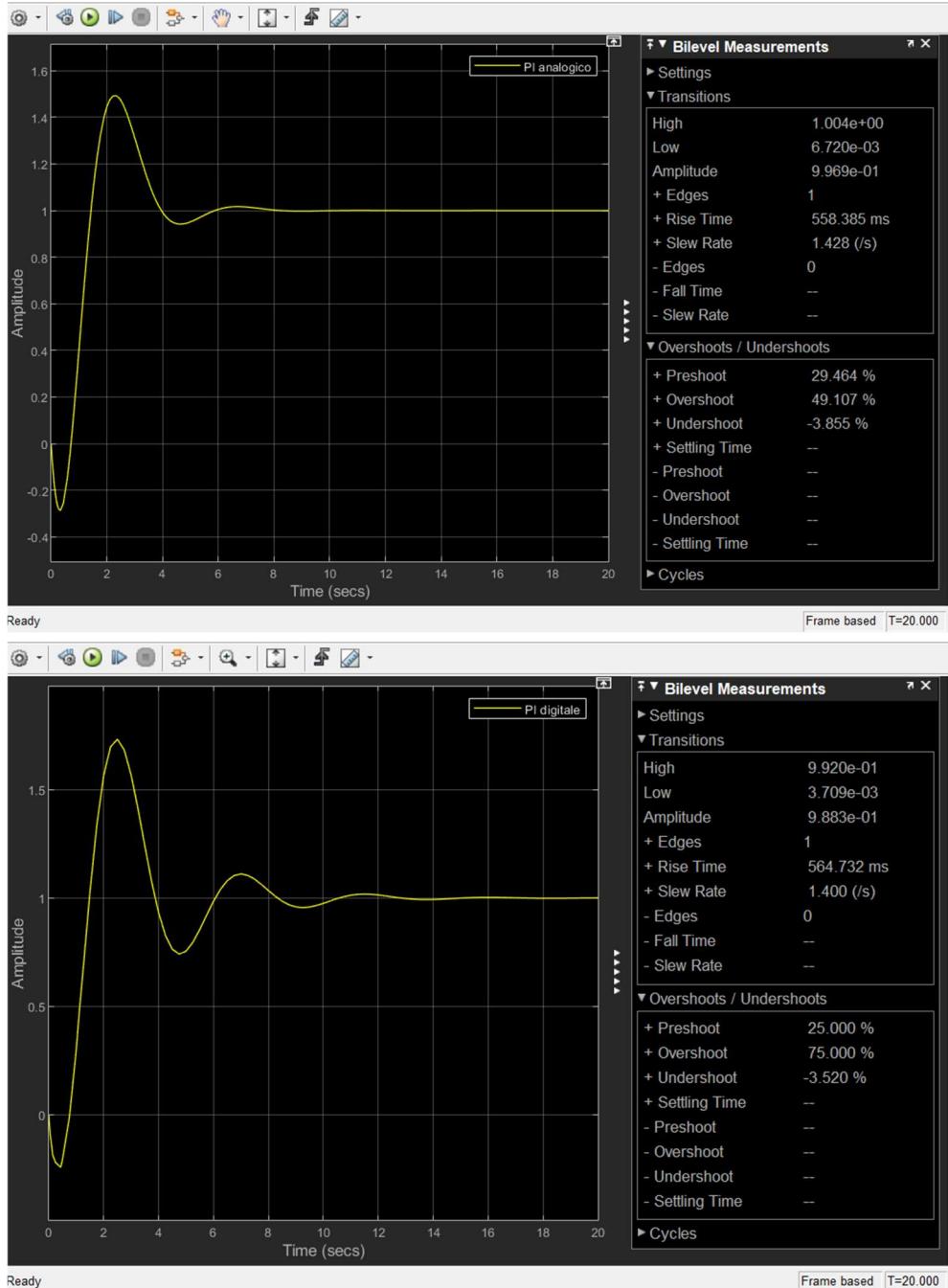
Allego qui di seguito il codice e i plot di esecuzione relativi al regolatore standard PI analogico e al regolatore standard PI digitale:

```
figure(7);
plot(yc.PI.Time, yc.PI.Data);
hold on
plot(yd.PI.Time, yd.PI.Data);
grid; legend('PI analogico', 'PI digitale');
```



In questo caso le due risposte, rispettivamente del regolatore PI analogico e del regolatore PI digitale, sono pressoché diverse. Questo dovuto al fatto che, rispetto ai

regolatori P analogico e digitale dove era presente solo il termine proporzionale, in questo caso oltre all'azione proporzionale è presente anche il termine integrale che impone una risposta *digitale* ben diversa rispetto a quella analogica. Analizzando più da vicino le risposte tramite i plot ottenuti dai *Time Scope Block*



possiamo notare come ci sia stato un incremento considerevole dell'overshoot nella risposta ottenuta applicando il PI digitale. Infatti, la risposta corrispondente al PI analogico presenta un *overshoot* pari al 49% mentre la risposta corrispondente al regolatore PI digitale presenta un *overshoot* pari al 75%. Invece, in entrambi i casi risultano essere pressoché uguali sia il *rise time*, il quale è all'incirca pari a 0.56 s, sia la *slew rate*, la quale è all'incirca pari a 1.4 per unità di tempo.

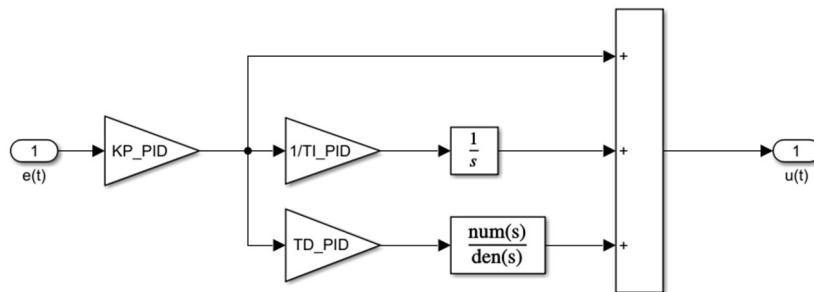
Regolatore Standard PID

PID analogico

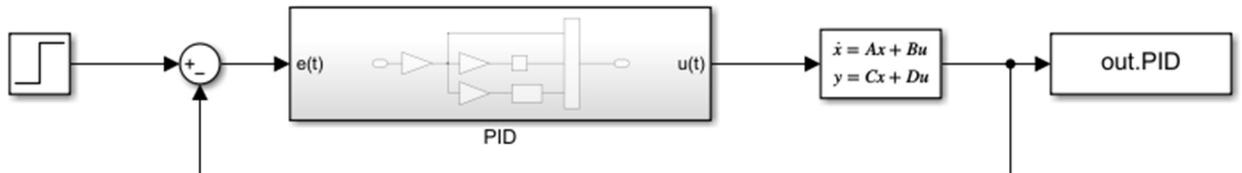
In questo caso tenendo conto dell'azione proporzionale, dell'azione integrale e dell'azione derivativa, otteniamo il regolatore standard di tipologia PID. Pertanto, la sua funzione di trasferimento corrispondente sarà:

$$C_{PID}(s) = K_P \cdot \left(1 + \frac{1}{s \cdot T_I} + T_D \frac{N \cdot s}{s + N} \right)$$

talche l'implementazione Simulink del *Block* corrispondente al PID analogico sarà



Quindi, il corrispondente schema Simulink tale per cui il controllore PID analogico verrà applicato al sistema generico supponendo di non conoscerlo sarà:



PID digitale

In questo caso, essendo che oltre all'azione proporzionale abbiamo anche l'azione integrale e l'azione derivativa, dovremo aggiungere anche il ricostruttore ZOH che mi permette di ricostruire il segnale per campioni ed effettivamente ottenere il segnale corrispondente. La funzione di trasferimento sarà ottenuta considerando la trasformazione di Tustin per il cambiamento di variabile per la trasformata Z:

$$C_{PID}(z) = K_P \cdot \left(1 + \frac{1}{T_I} \cdot \frac{T}{2} \cdot \frac{z+1}{z-1} + T_D \cdot N \cdot \frac{z-1}{z \cdot \left(N \cdot \frac{T}{2} + 1 \right) + \left(N \cdot \frac{T}{2} - 1 \right)} \right)$$

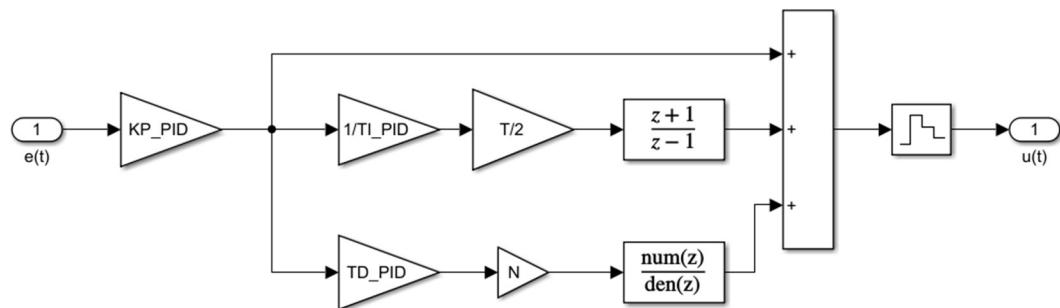
dove $T = \frac{t_a}{40}$ è il tempo di campionamento già citato precedentemente, $N = 100$ è il *filter coefficient*, anch'esso già citato precedentemente, ed infine i termini moltiplicativi $\frac{z+1}{z-1}$ e $\frac{z-1}{z \cdot \left(N \cdot \frac{T}{2} + 1 \right) + \left(N \cdot \frac{T}{2} - 1 \right)}$ sono frutto delle trasformazioni di Tustin sull'integratore e sul derivatore:

$$s = \frac{2}{T} \cdot \left(\frac{z-1}{z+1} \right)$$

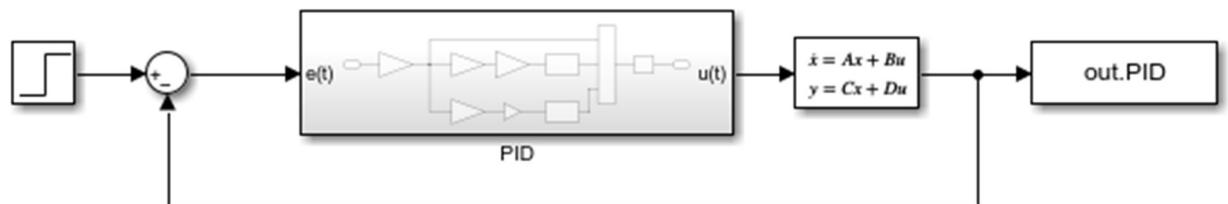
tale che:

$$\begin{aligned}
 T_D \frac{N \cdot s}{s + N} &= T_D \frac{N \cdot \frac{2}{T} \cdot \left(\frac{z-1}{z+1} \right)}{\frac{T}{2} \cdot \left(\frac{z-1}{z+1} \right) + N} = T_D \frac{N \cdot \left(\frac{z-1}{z+1} \right)}{\left(\frac{z-1}{z+1} \right) + N \cdot \frac{T}{2}} = \\
 &= T_D \frac{N \cdot (z-1)}{(z-1) + N \cdot \frac{T}{2} \cdot (z+1)} = \\
 &= T_D \cdot N \cdot \frac{z-1}{z-1 + z \cdot \left(N \cdot \frac{T}{2} \right) + \left(N \cdot \frac{T}{2} \right)} = \\
 &= T_D \cdot N \cdot \frac{z-1}{z \cdot \left(N \cdot \frac{T}{2} + 1 \right) + \left(N \cdot \frac{T}{2} - 1 \right)}
 \end{aligned}$$

Quindi, il *Discrete PID Block* avrà il seguente schema Simulink:



Pertanto, il corrispondente schema Simulink tale per cui il controllore PID digitale verrà applicato al sistema generico supponendo di non conoscerlo sarà:



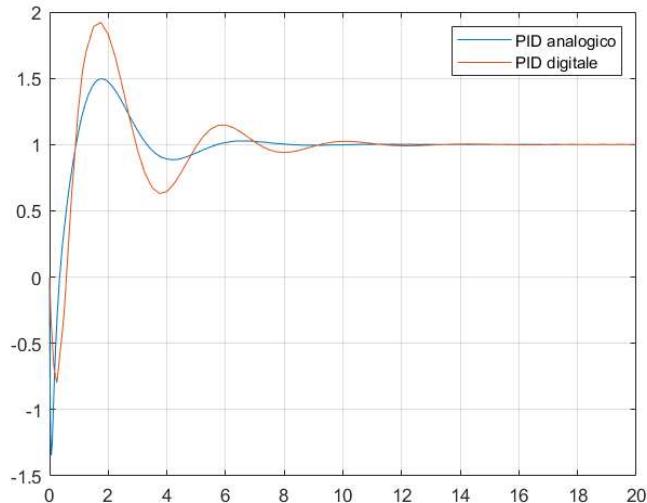
Analisi dei plot

Allego qui di seguito il codice e i plot di esecuzione relativi al regolatore standard PID analogico e al regolatore standard PID digitale:

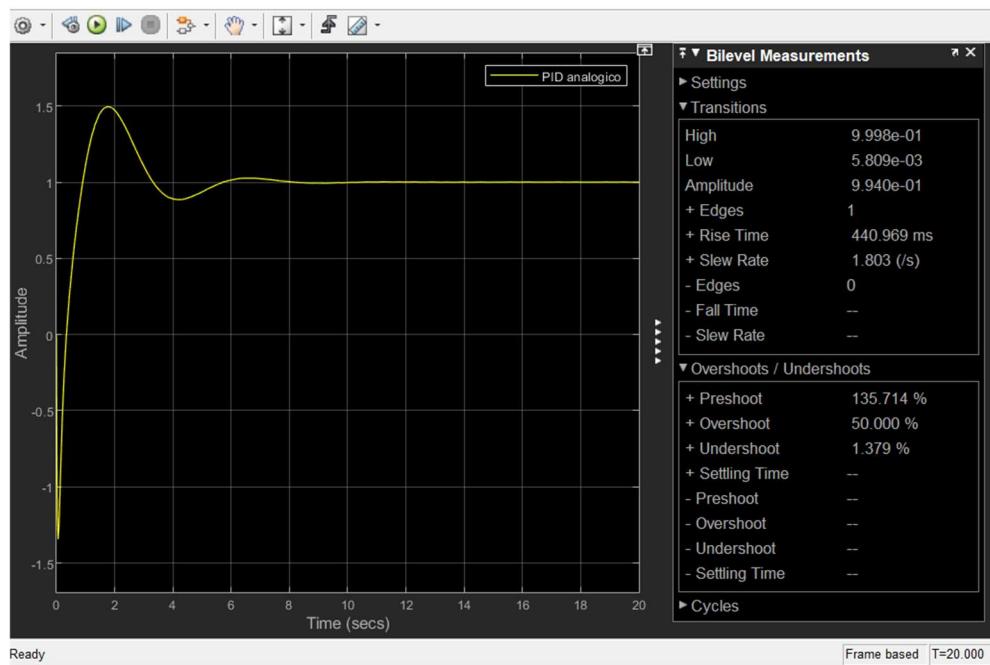
```

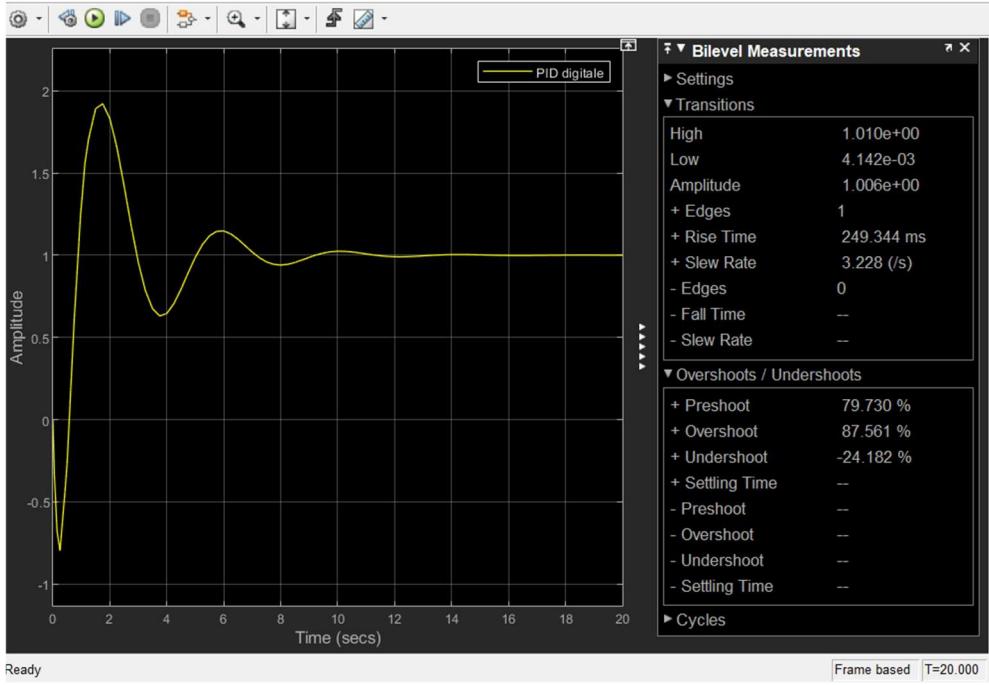
figure(8);
plot(yc.PID.Time, yc.PID.Data);
hold on
plot(yd.PID.Time, yd.PID.Data);
grid; legend('PID analogico', 'PID digitale');

```



Anche in questo caso le due risposte, rispettivamente del regolatore PID analogico e del regolatore PID digitale, sono pressoché diverse. Questo dovuto al fatto che, come già citato precedentemente, in questo caso oltre all'azione proporzionale sono presenti anche il termine integrale ed il termine derivativo che impone una risposta *digitale* ben diversa rispetto a quella analogica. Analizzando più da vicino le risposte tramite i plot ottenuti dai *Time Scope Block*



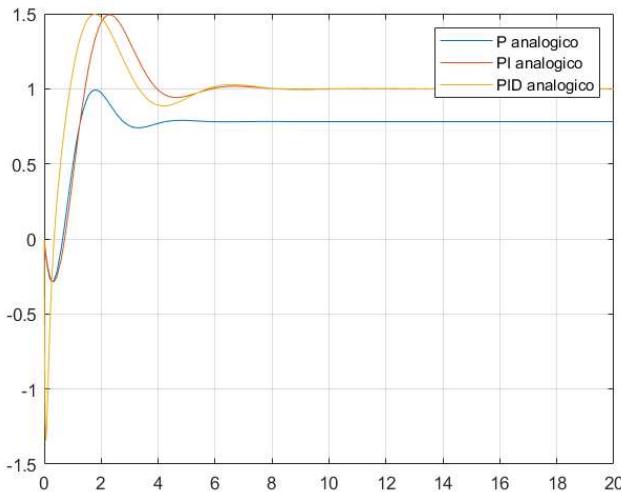


possiamo notare come ci sia stato un incremento considerevole dell'*overshoot* nella risposta ottenuta applicando il PI digitale. Infatti, la risposta corrispondente al PID analogico presenta un *overshoot* pari al 50% mentre la risposta corrispondente al regolatore PID digitale presenta un *overshoot* pari al 87.5%. Anche i tempi di salita corrispondenti risultano essere diversi: quello corrispondente al PID analogico risulta essere quasi il doppio di quello discreto. Inoltre, la *slew rate* del PID digitale risulta essere quasi il doppio rispetto a quella del PID analogico. Pertanto, il PID digitale rispetto a quello analogico presenta un *rise time* migliore ed una velocità di risposta peggiore.

Regolatori Standard Analogici

Allego qui di seguito il codice e le risposte del sistema ottenute rispetto ai regolatori standard analogici:

```
figure(9); title('continuos');
plot(yc.P.Time,yc.P.Data);
hold on
plot(yc.PI.Time,yc.PI.Data);
hold on
plot(yc.PID.Time,yc.PID.Data);
legend('P analogico','PI analogico','PID analogico'); grid;
```

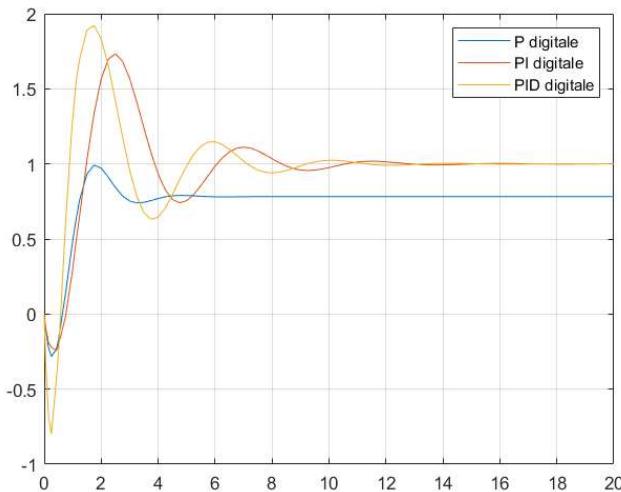


Possiamo notare come la risposta corrispondente al P analogico presenta un valore di regime minore rispetto a quelli corrispondenti al PI e al PID analogico. Tanto è vero che essa è come se presentasse un errore a regime “*negativo*” proprio perché le altre due risposte presentano errore a regime nullo. Infatti, per definizione l’azione proporzionale garantisce una riduzione dell’errore di inseguimento. Invece, gli altri regolatori standard presentano anche l’azione integrale e questo fa sì che l’azione integrale controbilanci l’azione proporzionale. Pertanto, è come se l’azione proporzionale nel regolatore standard P analogico faccia diminuire ancora l’errore nullo!!

Regolatori Standard Digitali

Allego qui di seguito il codice e le risposte del sistema ottenute rispetto ai regolatori standard digitali:

```
figure(10); title('discrete');
plot(yd.P.Time,yd.P.Data);
hold on
plot(yd.PI.Time,yd.PI.Data);
hold on
plot(yd.PID.Time,yd.PID.Data);
legend('P digitale','PI digitale','PID digitale'); grid;
```



Possiamo notare anche in questo caso che l'errore a regime del regolatore standard P digitale sia minore rispetto all'errore nullo a regime degli altri due regolatori standard digitali per le motivazioni sopra citate.

PID Tuner

L'applicazione PID Tuner presente in MATLAB permette di mettere a punto i *gain* di un controllore PID per un impianto SISO per ottenere un equilibrio tra prestazioni e robustezza. I grafici di analisi consentono di esaminare le prestazioni del controller nei domini del tempo e della frequenza. Inoltre, è possibile perfezionare in modo interattivo, tramite due slider, le prestazioni del controller per regolare la larghezza di banda del circuito e il margine di fase. Allego qui di seguito i parametri relativi alle risposte *tuned*:

```
KP_P_analogico_tuned = 1.47219531686421;
KP_P_digitale_tuned = 1.47219531686421;

KP_PI_analogico_tuned = 1.47593963796892;
I_PI_analogico_tuned = 0.325410877804267;
KP_PI_digitale_tuned = 1.4003674925568;
I_PI_digitale_tuned = 0.28478840442912;

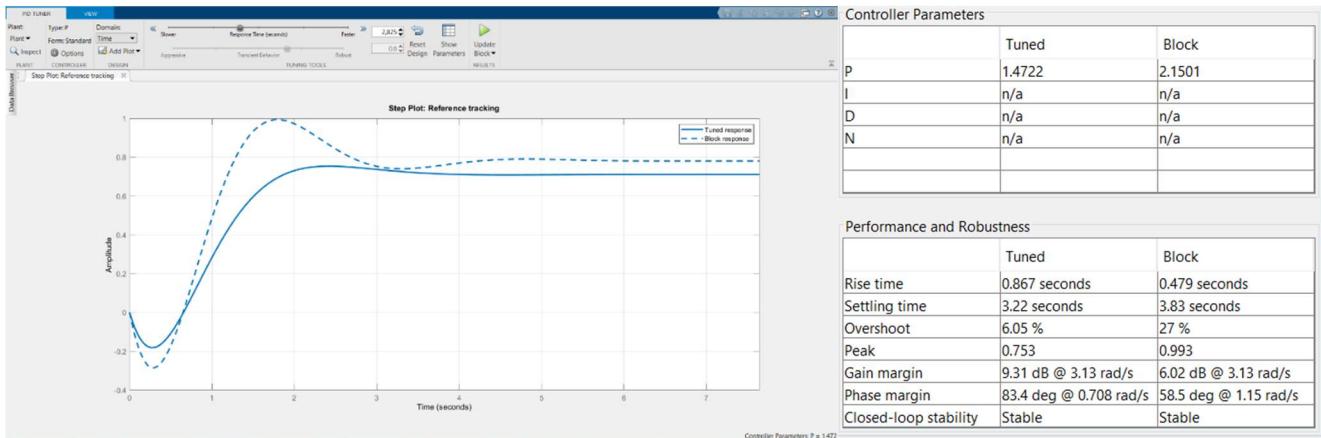
KP_PID_analogico_tuned = 1.9364815341473;
I_PID_analogico_tuned = 0.279951410984123;
TD_PID_analogico_tuned = 0.0554825984150835;
KP_PID_digitale_tuned = 1.83497063890365;
I_PID_digitale_tuned = 0.172208500364823;
TD_PID_digitale_tuned = 0.0992484035615046;

tuned = sim('tuned'); yt = tuned;
```

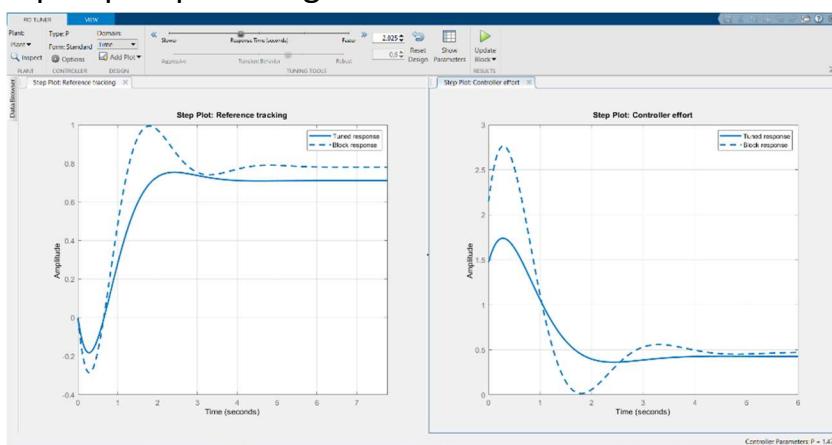
Analogici

Regolatore Standard P analogico

Effettuo il *tune* sulla risposta corrispondente al regolatore standard P analogico ed ottengo il seguente plot:



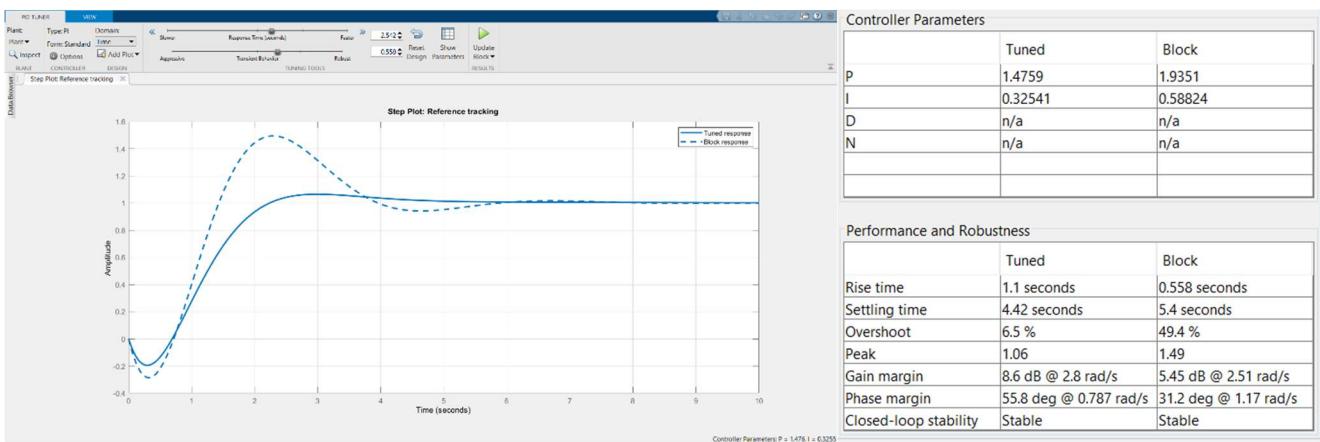
Ho scelto di modificare il tune proposto da MATLAB così da avere un compromesso sia per *overshoot* sia per i parametri di *rise time* e *settling time*. Infatti, la *tuned response* presenta un *overshoot* pari al 6.05% rispetto al 27% della *block response* e, inoltre, otteniamo un *settling time* ed un *rise time* pari rispettivamente a 0.867 s e 3.22 s. Abbiamo, quindi, un miglioramento dal punto di vista del tempo di assestamento ma anche un peggioramento di quasi il 50% per quanto riguarda il tempo di salita. Allego qui di seguito anche il plot per quanto riguarda il controller effort:



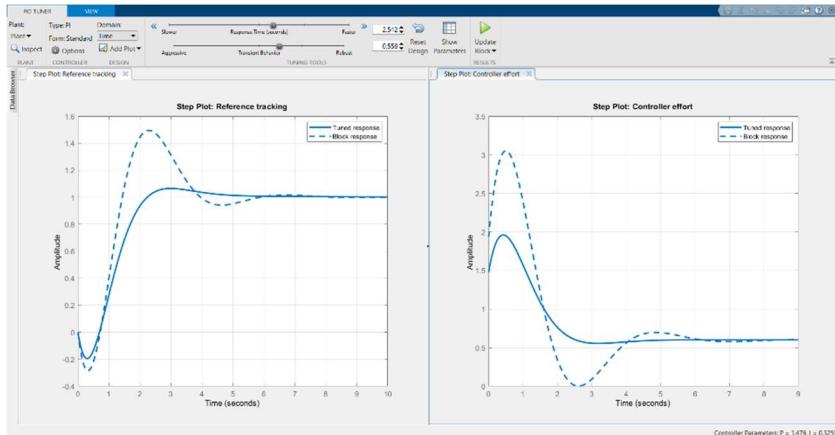
Possiamo notare come il controller effort rispetto alla *tuned response* risulta essere minore rispetto alla *block response*. Quindi, abbiamo avuto anche in questo ambito un miglioramento tale da garantire uno sforzo minore al controllore per ottenere questa risposta *tuned*.

Regolatore Standard PI analogico

Effettuo il *tune* sulla risposta corrispondente al regolatore standard PI analogico ed ottengo il seguente plot:



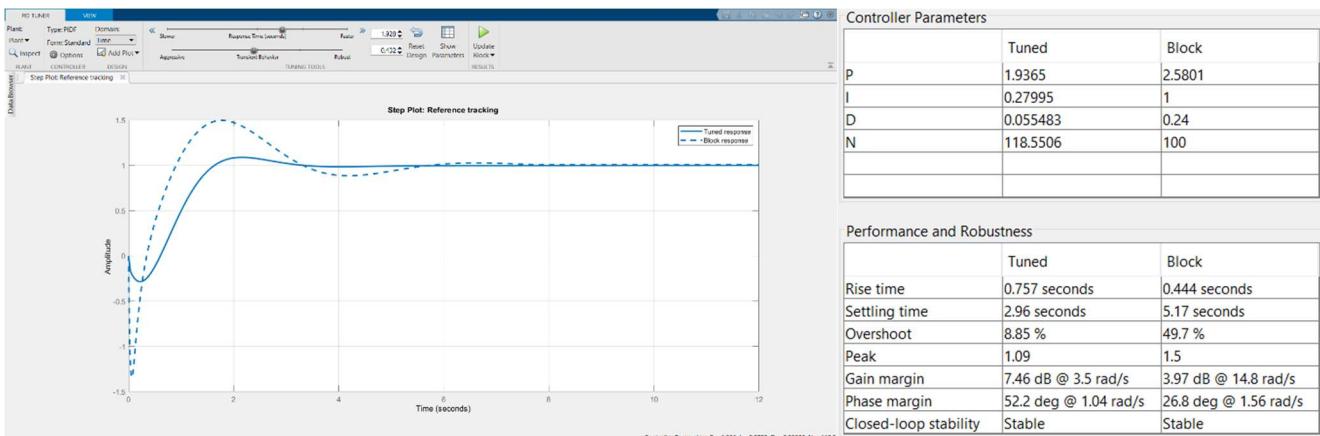
Possiamo notare, come l'*overshoot* della *tuned response* risulta essere pari al 6.5% rispetto al 49.4% della *block response*. Inoltre, è migliorato anche il tempo di assestamento tale da essere pari a 4.42 s. Invece, il *rise time* risulta essere peggiorato di circa il 50%. Allego qui di seguito anche il plot per quanto riguarda il controller effort:



Anche in questo caso il controller effort della *tuned response* risulta essere minore rispetto a quello del *block response*.

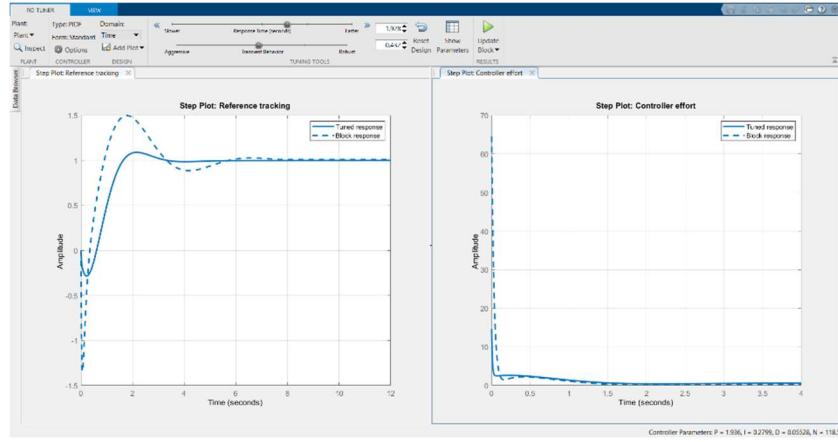
Regolatore Standard PID analogico

Effettuo il *tune* sulla risposta corrispondente al regolatore standard PID analogico ed ottengo il seguente plot:



In questo caso possiamo notare una diminuzione dell'*overshoot* fino all'8.85% e, inoltre, anche una diminuzione del *settling time* di oltre il 57%. Invece, il tempo di salita risulta

essere quasi il doppio della *block response*. Allego qui di seguito anche il plot per quanto riguarda il *controller effort*:

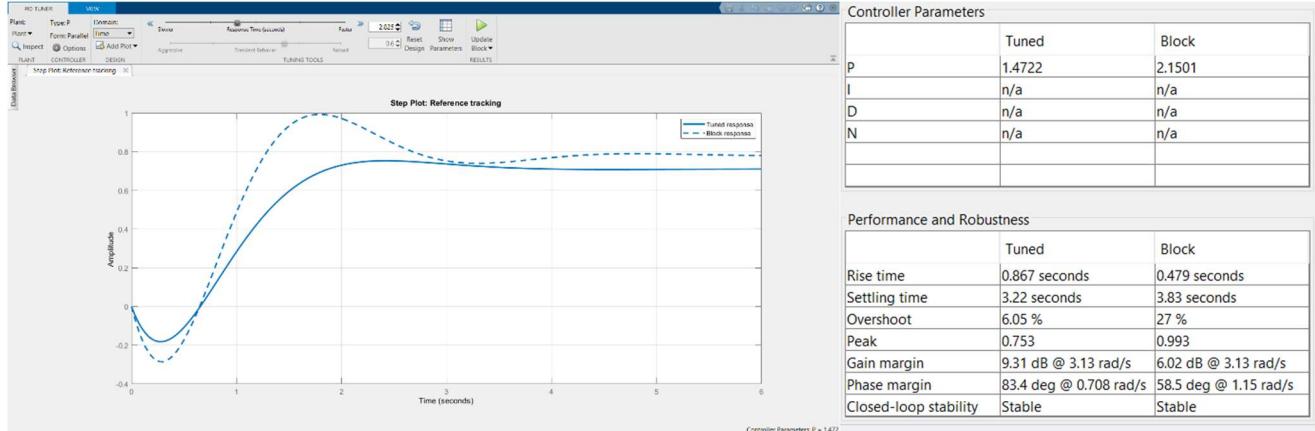


Possiamo notare come il picco iniziale del *controller effort* della *tuned response* risulta essere quasi 6 volte minore del picco iniziale del *controller effort* della *block response*.

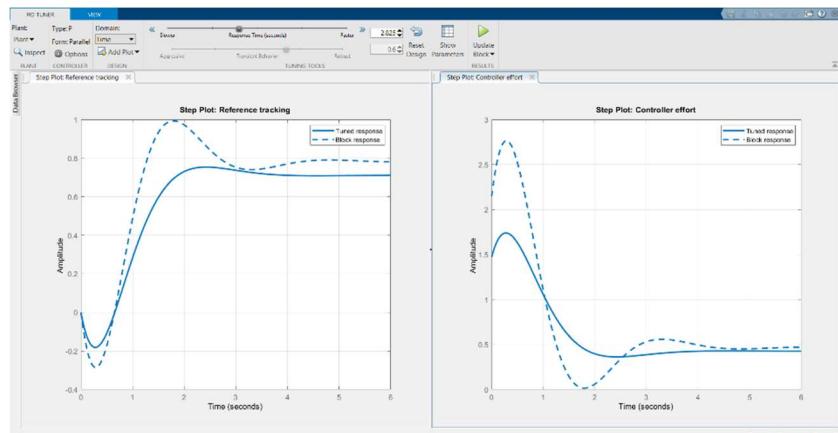
Digitali

Regolatore Standard P digitale

Effettuo il *tune* sulla risposta corrispondente al regolatore standard P digitale ed ottengo il seguente plot:



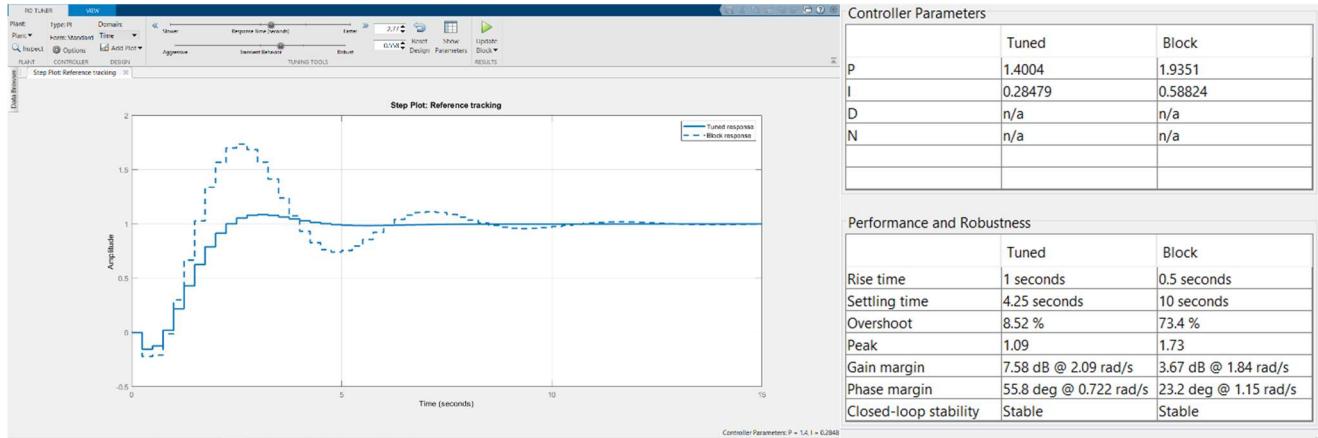
Possiamo notare come anche in questo caso l'*overshoot* risulta essere pari al 6%. Si ha un piccolo miglioramento del tempo di assestamento, mentre il tempo di salita peggiora di quasi il 50%. Allego qui di seguito anche il plot per quanto riguarda il *controller effort*:



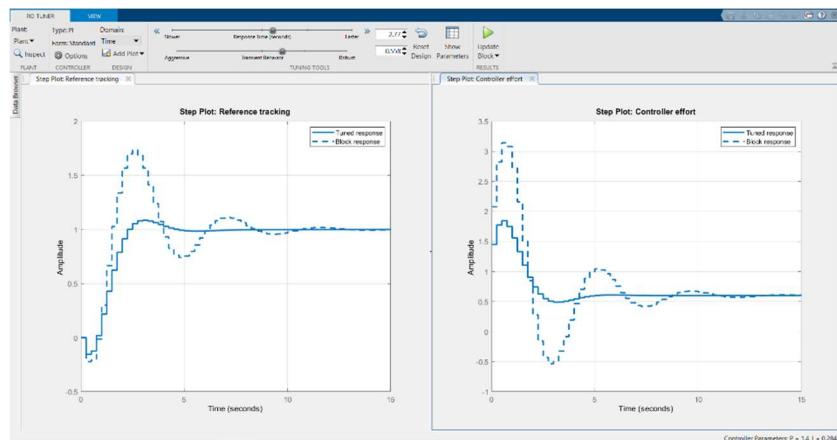
Anche in questo caso si ha un miglioramento in termini di *controller effort*.

Regolatore Standard PI digitale

Effettuo il *tune* sulla risposta corrispondente al regolatore standard PI digitale ed ottengo il seguente plot:



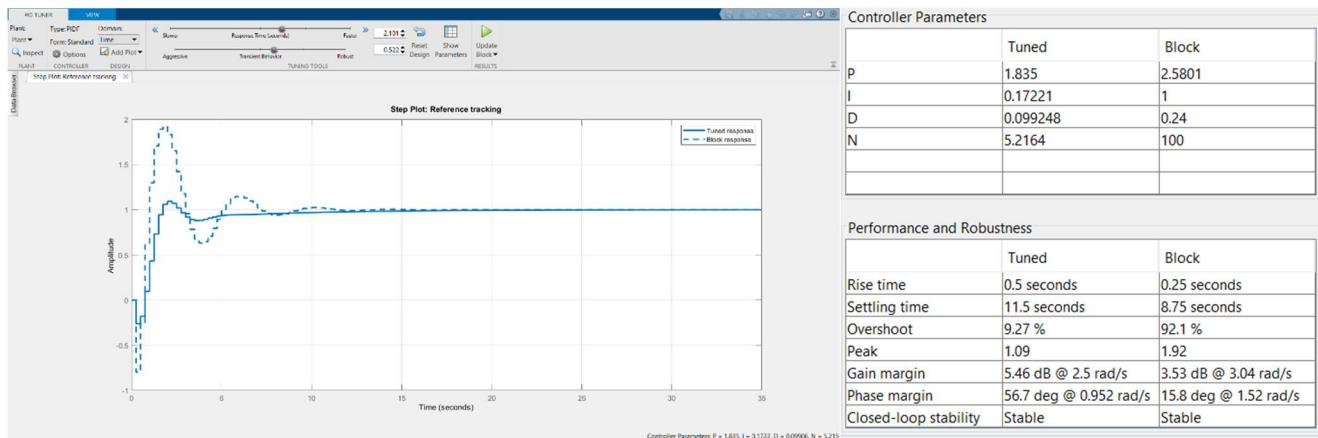
Possiamo notare come in questo caso il segnale risulta avere delle discontinuità di prima specie a causa dell'azione del ricostruttore ZOH. Infatti, quest'ultimo introduce di default delle discontinuità di prima specie dovuto al fatto che campionando cerca di *unire* tali punti con un determinato tempo di campionamento i campioni passati in input. In questo caso possiamo notare come l'*overshoot* sia stato praticamente *abbattuto* fino all'8% rispetto al 73.4% della *block response*. Inoltre, il tempo di assestamento risulta essere migliore di oltre il 50%, mentre il tempo di salita peggiora di 0.5 s. Allego qui di seguito anche il plot per quanto riguarda il *controller effort*:



In questo caso possiamo notare come il *controller effort* presenti un piccolo picco iniziale anche se tale segnale si assesta dopo pochissimi secondi a differenza del *controller effort* della *block response* che presenta oscillazioni prima di assestarsi.

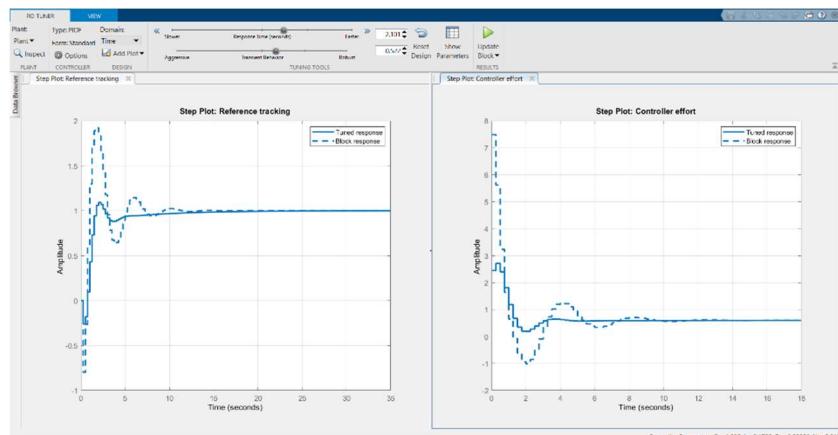
Regolatore Standard PID digitale

Effettuo il *tune* sulla risposta corrispondente al regolatore standard PID digitale ed ottengo il seguente plot:



Possiamo notare come in questo caso l'*overshoot* sia diminuito drasticamente. Infatti, la *block response* presentava un *overshoot* del 92.1%, mentre la *block response* ne presenta uno del 9.27%. Si hanno peggioramenti, invece, nel tempo di assestamento e nel tempo di salita che rispettivamente diventano pari a 11.5 s e 0.5 s.

Allego qui di seguito anche il plot per quanto riguarda il *controller effort*:



Anche in questo caso lo sforzo che il controllore deve compiere per ottenere la risposta sopra descritta è comunque minore allo sforzo che compieva per ottenere la *block response*.

Esercizio 2

Modellistica e Identificazione

Un modello matematico è una rappresentazione quantitativa di un fenomeno naturale o meglio il suo scopo è quello di rappresentare in maniera molto fedele un determinato oggetto, un fenomeno reale o un insieme di fenomeni. Esso è spesso costruito con lo scopo di fornire previsioni sullo *stato futuro* di un fenomeno o di un sistema. Generalmente, il modello descrive la probabile evoluzione di un fenomeno o di un sistema sulla base di dati iniziali (*condizioni iniziali*) forniti dall'utente (*input*) restituendo dei dati finali (*output*). L'efficacia del modello può essere, quindi, misurata comparando i dati finali ottenuti con il risultato effettivo osservato dell'evoluzione del fenomeno o del sistema. Pertanto, un modello è un rappresentazione astratta, parziale ed approssimata di un fenomeno naturale, di un processo tecnologico, di un dispositivo o, più in generale, di un sistema concreto costituito da un insieme di entità tra loro interagenti. In genere, nell'ambito della modellistica e dell'identificazione, vengono associati due tipologie di problemi: il problema legato alla *modellistica*, cioè l'individuazione di una struttura di tipo matematica o di tipo informatica (può trattarsi anche di un algoritmo) per lo studio di un certo fenomeno, ed il problema legato all'*identificazione*, cioè identificare i parametri della struttura matematica o informatica determinata nel problema precedentemente descritto o che viene data a priori senza svolgere il problema della modellistica. Il primo problema ha come obiettivi la comprensione del fenomeno, la replica del fenomeno in un dominio diverso, la previsione sul futuro della realtà di interesse e caso mai la determinazione di leggi di controllo se si tratta di problema legato all'automazione. Il secondo problema, invece, ha come obiettivo la determinazione di valori numerici congrui affinchè la struttura replichi fedelmente la realtà di interesse. Per quanto riguarda il problema della modellistica, esistono due tipi di approcci: l'approccio *physical based*, cioè basato sulla fisica del fenomeno tale che i modelli ottenuti risultano essere molto accurati anche se è richiesta una conoscenza a priori delle leggi fisiche che caratterizzano il fenomeno stesso, e l'approccio *data driven*, cioè basato sui dati tale da non richiedere nessuna conoscenza specialistica anche se non si ha cognizione dei meccanismi interni del modello ed inoltre è necessaria una mole elevata di dati per determinare i parametri del *physical based*. In questo caso bisogna tenere conto dell'approccio *data driven* poiché disponiamo già di un processo, anche se sconosciuto, tale da effettuare delle misure e successivamente procedere con i successivi passi dell'approccio. Nello specifico, l'identificazione viene scomposta in:

1. misurare le grandezze di interesse (*raccolta dei dati*)

2. definire un modello che si ritiene adeguato a rappresentare il fenomeno in esame
3. evidenziare i parametri incogniti all'interno del modello ipotizzato
4. definire di un indice numerico che sia funzione di $e(t) = y(t) - z(t)$ tale che $e(t)$ sia il più piccolo possibile. L'indice numerico può essere ottenuto tramite la somma dei quadrati degli errori nella seguente maniera:

$$I = \frac{1}{N} \sum_{k=0}^{N-1} e^2(t)$$

5. determinare l'insieme dei parametri che rendono minimo l'indice rispettando eventuali vincoli di ammissibilità
6. validazione del risultato tale che se il risultato non va bene bisogna ritornare sui passi precedenti.

La soluzione ai minimi quadrati

Considerando, quindi, N esperimenti su un sistema (*processo nel nostro caso*). In ciascun esperimento si fissano i valori di p grandezze $\varphi_1, \varphi_2, \dots, \varphi_p$ e si misura il corrispondente valore di una ulteriore grandezza z , ritenendo che quest'ultima grandezza sia legata alle variabili φ sopra citate, tale che:

$$\begin{cases} z_1 \text{ legata a } (\varphi_{11}, \varphi_{12}, \dots, \varphi_{1p}) \\ z_2 \text{ legata a } (\varphi_{21}, \varphi_{22}, \dots, \varphi_{2p}) \\ \dots \\ z_N \text{ legata a } (\varphi_{N1}, \varphi_{N2}, \dots, \varphi_{Np}) \end{cases}$$

Nel nostro caso il processo è identificato da una *black box* di cui non abbiamo informazioni:



Pertanto, consideriamo un sistema di N equazioni dove i parametri $\theta_1, \theta_2, \dots, \theta_p$ sono incogniti:

$$\begin{cases} z_1 = \varphi_{11}\theta_1 + \varphi_{12}\theta_2 + \dots + \varphi_{1p}\theta_p \\ z_2 = \varphi_{21}\theta_1 + \varphi_{22}\theta_2 + \dots + \varphi_{2p}\theta_p \\ \dots \\ z_N = \varphi_{N1}\theta_1 + \varphi_{N2}\theta_2 + \dots + \varphi_{Np}\theta_p \end{cases}$$

dove $N \gg p$ perché per $N < p$ avremmo infinite soluzioni e, quindi, infiniti modelli mentre per $N = p$ avremmo un sistema lineare, cioè una sola soluzione ed un solo

modello. In generale, la soluzione esatta a tale sistema non esiste. Raccogliendo i termini nei rispettivi vettori e nella rispettiva matrice otteniamo la seguente soluzione analitica:

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix} \quad \Phi = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1p} \\ \varphi_{21} & \ddots & \cdots & \varphi_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{N1} & \cdots & \cdots & \varphi_{Np} \end{bmatrix}$$

(dove Φ viene denominata matrice dei regressori)

tale che:

$$Z = \Phi\theta$$

ed essendo che Φ è una matrice rettangolare, non esiste la sua inversa e, quindi, non esiste la soluzione esatta. Pertanto, viene calcolata una soluzione approssimata. Bisogna notare che comunque lo scarto di errore tra l'uscita approssimata e l'uscita reale non è nullo altrimenti si ricondurrebbe al caso ideale che non si verifica mai ($z_i - y_i = 0$):

$$z_i - y_i = z_i - (\varphi_{i1}\theta_1 + \cdots + \varphi_{ip}\theta_p) = e_i(\theta_1, \dots, \theta_p) \neq 0$$

con $i = 1, 2, \dots, N$.

Pertanto, il nostro obiettivo è quello di minimizzare l'errore che dipende effettivamente dai parametri $\theta_1, \dots, \theta_p$ tale da ottenere, quindi, di quest'ultimi una *stima ai minimi quadrati*, cioè la p-pla di parametri (denotata con il vettore $\hat{\theta}$) che rende minima la somma dei quadrati degli scarti $e_i(\theta_1, \dots, \theta_p)$. Quindi, calcolo l'indice di costo dipendente dai parametri $\theta_1, \dots, \theta_p$: (N.B. $x^T y = y^T x$)

$$\begin{aligned} I(\theta) &= \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (z_i - y_i)^2 = \|z - y\|^2 = \|z - \Phi\theta\|^2 \\ &= (z - \Phi\theta)^T \cdot (z - \Phi\theta) \\ &= \theta^T \Phi^T \Phi \theta - \theta^T \Phi^T z - z^T \Phi \theta + z^T z \\ &= \theta^T \Phi^T \Phi \theta - 2\theta^T \Phi^T z + z^T z \end{aligned}$$

Questo ottenuto è l'indice che deve essere minimizzato. Pertanto, calcolo il gradiente di $I(\theta)$ rispetto a θ :

$$\nabla(I(\theta)) = 2\Phi^T \Phi \theta - 2\Phi^T z$$

essendo che il punto di minimo è il punto in cui il gradiente si annulla, calcolo il punto in cui si annulla:

$$\begin{aligned} 2\Phi^T \Phi \theta &= 2\Phi^T z \\ \Phi^T \Phi \theta &= \Phi^T z \end{aligned}$$

Se $N > p$, cioè il numero dei dati è maggiore del numero di parametri incogniti, e se $\text{rango}(\Phi) = p$, cioè colonne di Φ linearmente indipendenti, allora $\det(\Phi^T \Phi) \neq 0$ e, quindi, il sistema di equazioni lineari sopra citato ha una sola soluzione. Tale soluzione è il minimizzatore di $I(\theta)$. Formalmente si può scrivere:

$$\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T z$$

dove $(\Phi^T \Phi)^{-1} \Phi^T$ è conosciuta come *matrice pseudo-inversa di Φ* . Il vettore $\hat{\theta}$ è denominato *soluzione ai minimi quadrati del sistema sovradeterminato di equazioni lineari $Z = \Phi\theta$* . Viene denominato “*ai minimi quadrati*” poiché c’è un’ipotesi di fondo, cioè che sia presente a priori un errore di misura tale che questo errore di misura seguia una distribuzione di tipo Gaussiana a media nulla. Infatti, essendo la varianza della Gaussiana legata al quadrato allora si considera questo errore quadratico. Ovviamente, tale vettore non è una vera soluzione ma risulta che $Z \neq \Phi\theta$. In generale, $\Phi^T \Phi$ non è detto che sia invertibile! Si fa in modo, quindi, che Φ sia *persistentemente citata*, cioè che alcune righe non siano dipendenti dalle altre. Teoricamente viene naturale perché quando si prendono delle misure si commettono degli errori.

Analisi preliminare di un modello ARMA generico

Pertanto, considerando un modello ARMA del tipo

$$y(t) = -a_1 y(t-1) - a_2 y(t-2) - \cdots - a_n y(t-n) \\ + b_0 u(t) + b_1 u(t-1) + \cdots + b_m u(t-m)$$

e considerando T misurazioni

$$\begin{cases} Z(n) = -a_1 z(n-1) - a_2 z(n-2) - \cdots - a_n z(0) + b_0 u(n) + b_1 u(n-1) + \cdots + b_m u(n-m) \\ Z(n+1) = -a_1 z(n) - a_2 z(n-1) - \cdots - a_n z(1) + b_0 u(n+1) + b_1 u(n) + \cdots + b_m u(n-m+1) \\ \vdots \\ Z(k) = -a_1 z(k-1) - a_2 z(k-2) - \cdots - a_n z(0) + b_0 u(k) + b_1 u(k-1) + \cdots + b_m u(k-m) \\ \vdots \\ Z(T) = -a_1 z(T-1) - a_2 z(T-2) - \cdots - a_n z(0) + b_0 u(T) + b_1 u(T-1) + \cdots + b_m u(T-m) \end{cases}$$

avrò rispettivamente i termini noti Z e i regressori al k -esimo istante $\varphi(k)$:

$$Z = \begin{bmatrix} Z(n) \\ Z(n+1) \\ \vdots \\ Z(T) \end{bmatrix} \quad \varphi(k) = \begin{bmatrix} Z(k-1) \\ Z(k-2) \\ \vdots \\ Z(k-n) \\ u(k) \\ u(k-1) \\ \vdots \\ u(k-m) \end{bmatrix}$$

pertanto, la matrice dei regressori sarà:

$$\Phi = \begin{bmatrix} \varphi^T(0) \\ \varphi^T(1) \\ \vdots \\ \varphi^T(T-n) \end{bmatrix}$$

Quindi, il nostro vettore incognito sarà:

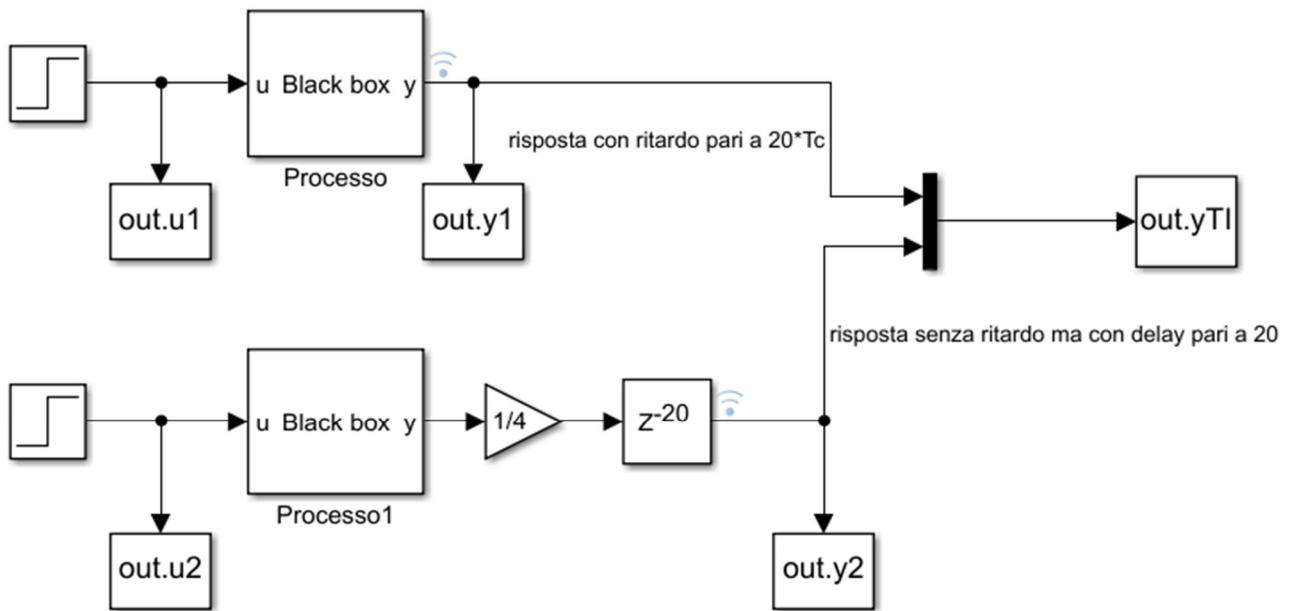
$$\theta = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \\ b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix}$$

tal che la soluzione sarà data considerando la pseudo-inversa $\theta = (\Phi^T \Phi)^{-1} \Phi^T Z$. Ovviamente, se vogliamo che θ minimizzi l'indice di costo $I_{M\theta}(\theta) = \|z - \Phi\theta\|^2$ allora dobbiamo considerare $\hat{\theta} = (\Phi^T \Phi)^{-1} \Phi^T Z$.

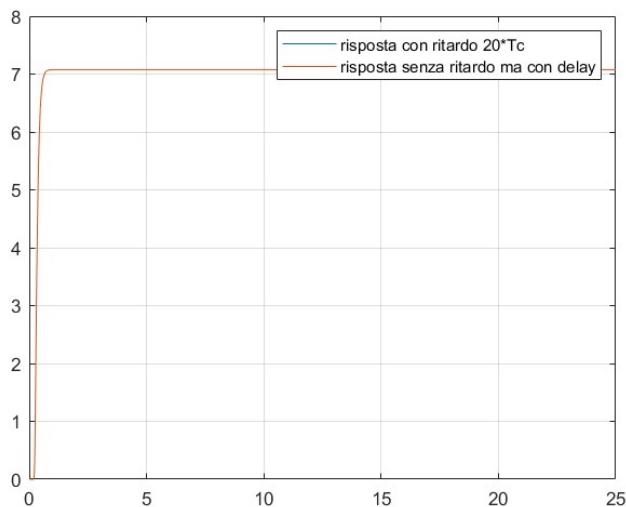
Verifica LTI

Verifico inizialmente che il sistema sia lineare e tempo-invariante, cioè tale che il sistema sia soggetto al principio di sovrapposizione degli effetti e tale che il suo comportamento sia costante nel tempo. Più nello specifico, un sistema è tempo-invariante quando l'uscita generata da un segnale ritardato è uguale all'uscita generata dal segnale originale, ritardata della stessa quantità:

$$u(t - \tau) \rightarrow y(t - \tau)$$



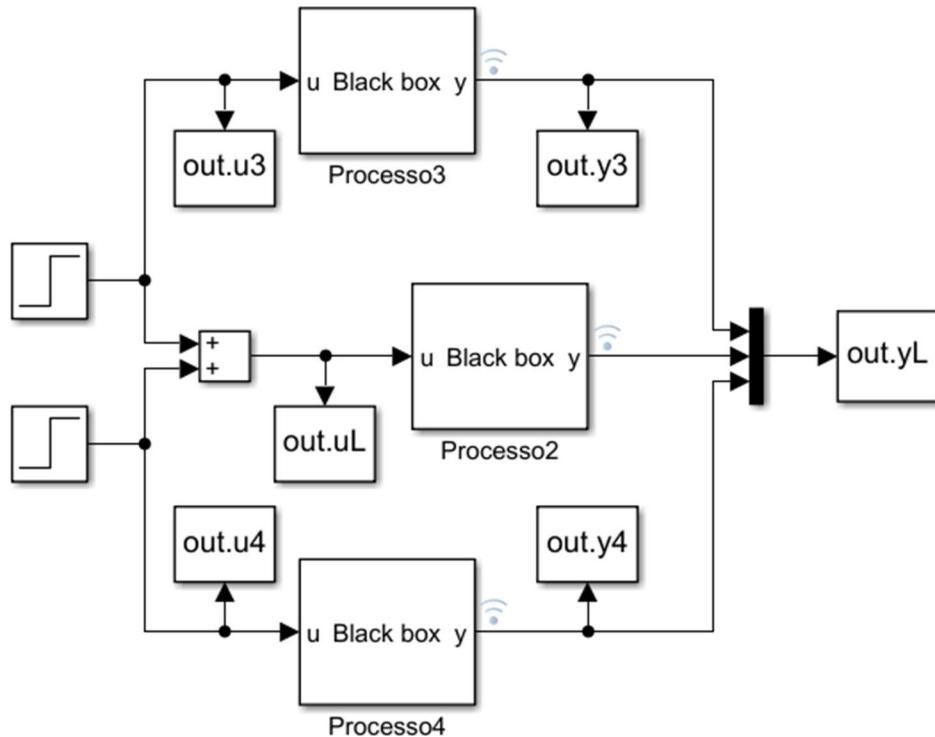
Ottenendo il seguente plot:



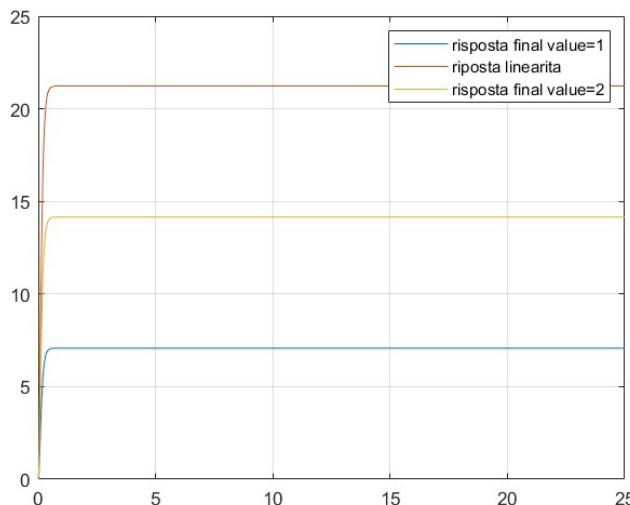
Possiamo notare come le due risposte coincidono. Pertanto, il sistema in questione è in tempo-invariante.

Invece, un sistema è lineare quando l'uscita generata dalla combinazione lineare di due o più ingressi è uguale alla combinazione lineare delle uscite generate dai singoli ingressi:

$$u_3 + u_4 = uL \rightarrow yL = y_3 + y_4$$



Ottenendo il seguente plot:



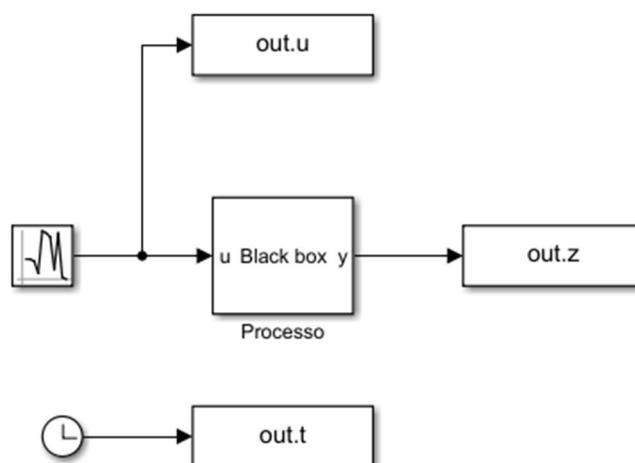
Possiamo notare come la riposta del sistema, dopo aver applicato la somma dei due gradini in ingresso, è pari alla somma dei due output ottenuti applicando i singoli *step*, uno con final value pari a 1 e l’altro con final value pari a 2, al sistema in questione. Pertanto, essendo che il sistema descritto da questa black-box risulta essere lineare e tempo-invariante (*LTI*), tale sistema può essere rappresentato mediante una funzione di trasferimento in forma di funzione razionale fratta.

Generazione misure del processo black-box

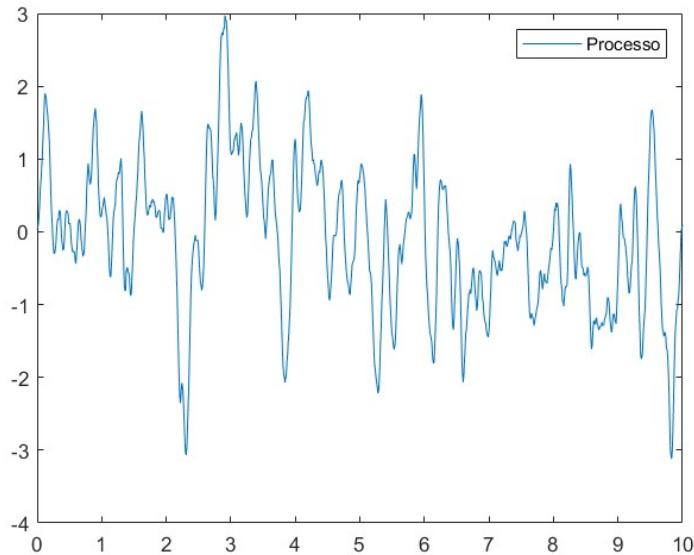
Allego qui di seguito il codice e lo schema corrispondente alla generazione delle misure del processo black-box considerando in input un *Random Number Block* così da ottenere le misure Z :

```
clear; clc; close all;

Tc = 0.01;
simulazione = sim('genera_misure');
t = simulazione.t; u = simulazione.u; z = simulazione.z;
figure(1);
plot(t,z); legend('Processo');
```



ottenendo il seguente plot:



Ho utilizzato l'ingresso random così da considerare un segnale perturbato, cioè persistentemente eccitato e tale che la matrice dei regressori corrispondente avrà tutte le colonne linearmente indipendenti così da ricavare la pseudo-inversa e così da identificare, di conseguenza, in maniera corretta il modello corrispondente al processo della black-box. Nella realtà solitamente sono presenti degli errori di misura nelle misure. Tale aspetto si potrebbe implementare aggiungendo alle misure calcolate un errore random nella seguente maniera:

```
z = z + rand(length(z),1)*0.01;
```

Costruzione e analisi del modello

Applicando, ora, il principio di parsimonia ad n ed m , i quali rappresentano rispettivamente gli indici n -esimi ed m -esimi dei parametri a e b del modello ARMA generico, posso calcolare il vettore dei termini noti Z e i regressori i -esimi f_i tale da approssimare la soluzione θ :

```
n = 4; m = 4;
Z = z(max(n,m)+1:end); fi = zeros(1,n+m+1); FI = [];
T = length(t);

for k=max(n,m):T-1
    for j=1:n
        fi(j) = -z(k-j+1);
    end
    for j=1:m+1
        fi(j+n) = u(k-j+2);
    end
    FI=[FI;fi];
end
```

Lo *statement for* serve per la costruzione della matrice dei regressori *FI*. Tale statement inizia da *n o m* in base a quale dei due è maggiore dell'altro e itera fino a $T - 1$. All'interno del *for* vengono prima inizializzati i parametri *z* in un *for* interno e poi in un altro *for* interno vengono inizializzati i parametri *u*. Pertanto, ora sono in grado di generare il vettore dei parametri incogniti in accordo al metodo dei minimi quadrati:

```
teta = FI\Z;
```

Quindi, ora posso creare una funzione di trasferimento in forma filtro con variabile z^{-1} , a partire da questi parametri appena calcolati, dove al numeratore e al denominatore andrò rispettivamente a porre le *b* e le *a*:

```
Gz = tf(teta(n+1:end)',[1;teta(1:n)]',Tc,'Variable','z^-1');
```

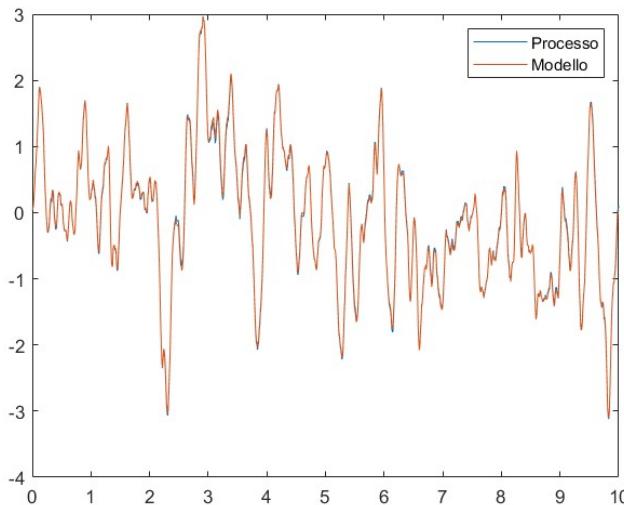
tale che essa sarà:

$$G(z) = \frac{0.03098 - 0.03105z^{-1} + 0.06166z^{-2} + 0.0287z^{-3} + 0.02848z^{-4}}{1 - 1.981z^{-1} + 1.244z^{-2} - 0.2657z^{-3} + 0.0188z^{-4}}$$

Pertanto, ora posso plottare e analizzare il processo della black-box in confronto al modello ottenuto e descritto sopra:

```
figure(4);
plot(t,z);
hold on
y = lsim(Gz,u,t); plot(t,y); legend('Processo','Modello');
```

utilizzo il comando *lsim* di MATLAB per tracciare la risposta temporale del sistema dinamico, descritto dalla funzione di trasferimento in forma filtro sopra descritta, avente come input arbitrari *u*. Pertanto, otterrò il seguente plot:



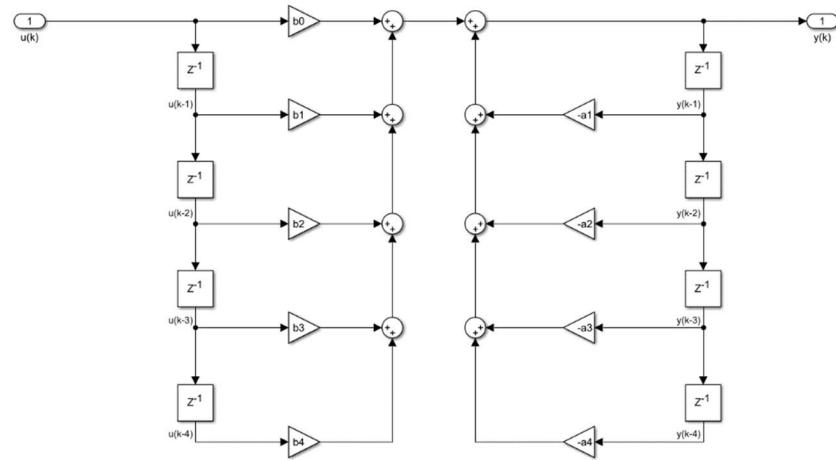
Possiamo notare come la risposta del modello sopra descritto approssima fedelmente la risposta del processo descritto dalla black-box.

Definisco le variabili $b_0, b_1, b_2, b_3, b_4, a_0, a_1, a_2, a_3, a_4$ che mi saranno utili per la costruzione dei modelli ARMA, Forma Canonica I e Forma Canonica II:

```
[b a] = tfdata(Gz,'v');
b0 = b(1); b1 = b(2); b2 = b(3); b3 = b(4); b4 = b(5);
a0 = a(1); a1 = a(2); a2 = a(3); a3 = a(4); a4 = a(5);
```

Implementazione Modello ARMA

Allego qui di seguito l'implementazione del modello mediante rappresentazione in modello ARMA in Simulink:



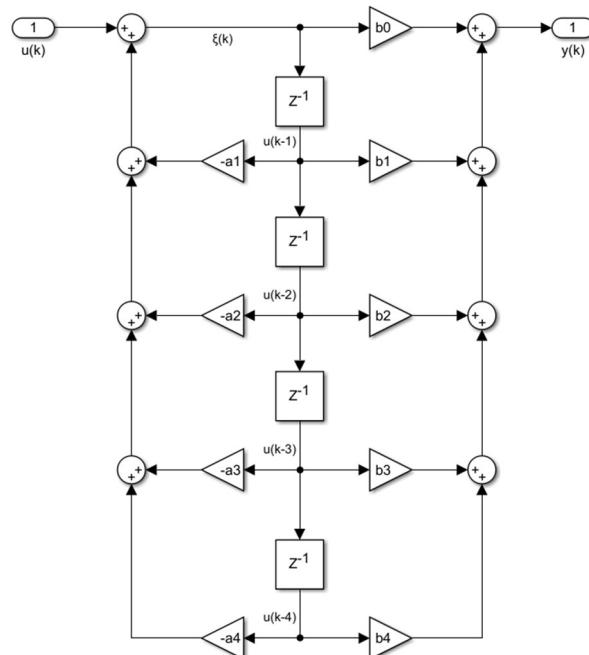
descritto dal seguente filtro digitale:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \cdots - a_n y(k-n) + b_0 u(k-1) + \cdots + b_n u(k-n)$$

tale che il termine in cui sono presenti i coefficienti $a_i \in R$ è denominato componente *Auto-Regressive*, mentre il termine in cui sono presenti i coefficienti $b_i \in R$ è denominato componente *Moving-Average*.

Implementazione Forma Canonica I

Allego qui di seguito l'implementazione del modello mediante rappresentazione in Forma Canonica di tipologia I:



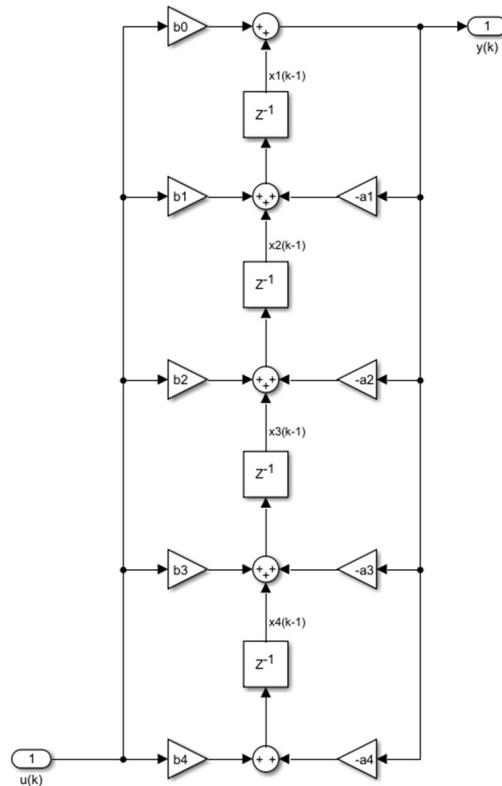
descritto dalla seguente rappresentazione:

$$\left\{ \begin{array}{l} \xi_s(k-1) \leftarrow u(k) - a_1\xi_p(k-1) - \cdots - a_n\xi_p(k-n) \\ \xi_s(k-2) \leftarrow \xi_p(k-1) \\ \xi_s(k-3) \leftarrow \xi_p(k-2) \\ \vdots \\ \xi_s(k-n) \leftarrow \xi_p(k-n+1) \end{array} \right.$$

dove con la variabile temporanea $\xi(k)$ ed i suoi ritardi rappresento lo stato del filtro mentre con i pedici s e p indico rispettivamente i valori successivi al calcolo dell'uscita del registro e i valori precedenti al calcolo dell'uscita.

Implementazione Forma Canonica II

Allego qui di seguito l'implementazione del modello mediante rappresentazione in Forma Canonica di tipologia II:

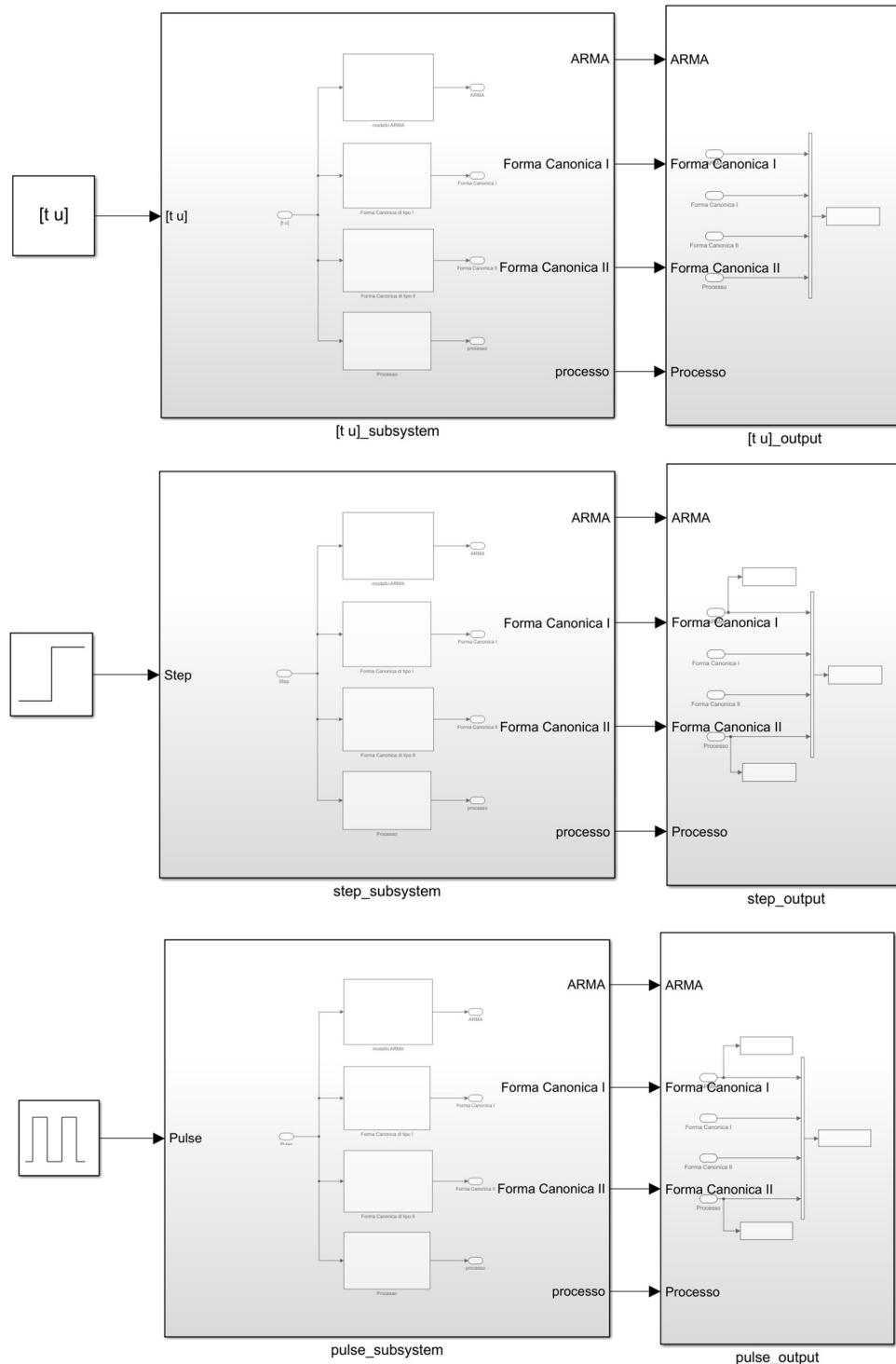


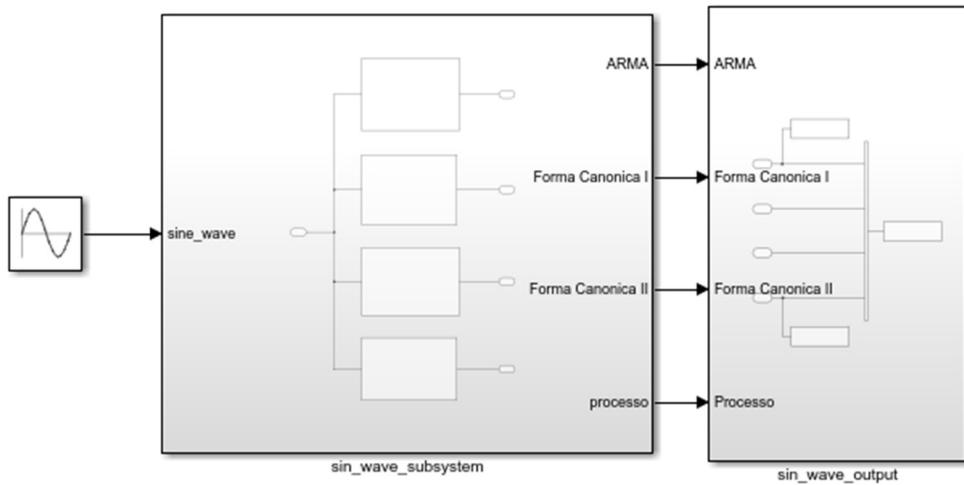
descritto dalla seguente rappresentazione:

$$\left\{ \begin{array}{l} y(k) = b_0u(k) + x_1(k-1) \\ x_1(k) = b_1u(k) - a_1y(k) + x_2(k-1) \\ x_2(k) = b_2u(k) - a_2y(k) + x_3(k-1) \\ \vdots \\ x_{n-1}(k) = b_{n-1}u(k) - a_{n-1}y(k) + x_n(k-1) \\ x_n(k) = b_nu(k) - a_ny(k) \end{array} \right.$$

Validazione modello

Dopo aver implementato le tre tipologie di rappresentazioni sopra citate sono in grado di plottare e analizzare il modello mediante tali rappresentazioni rispetto al processo. Ovviamente prima, però, devo costruire il modello simulink del sistema complessivo per ogni risposta che dovrà essere plottata. Allego, pertanto, i sistemi complessivi per le risposte relativi agli input generici u , all'input *gradino*, all'input *treno di impulsi* e all'input *onda sinusoidale*:





Posso, ora, analizzare le varie risposte:

Analisi delle risposte

Risposta all'ingresso random

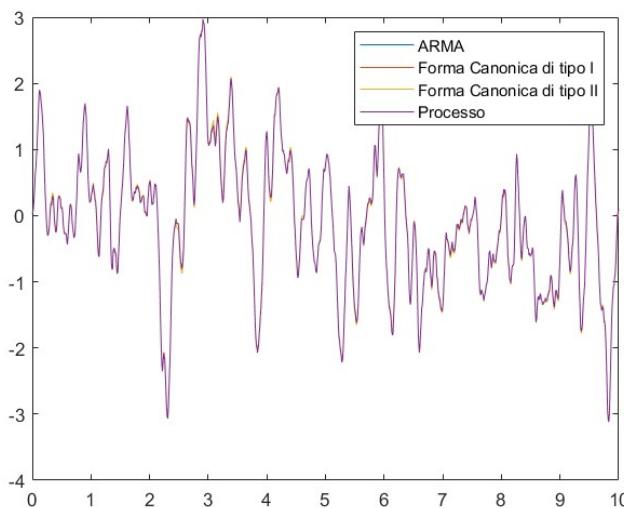
Pertanto, allego il codice e il plot corrispondente qui di seguito:

```

validazione_modello = sim('validazione_modello');
time = validazione_modello.time;

figure(5); title('[t u] output');
tu_output = validazione_modello.tu_output; plot(time, tu_output);
legend('ARMA','Forma Canonica di tipo I','Forma Canonica di tipo II','Processo');

```

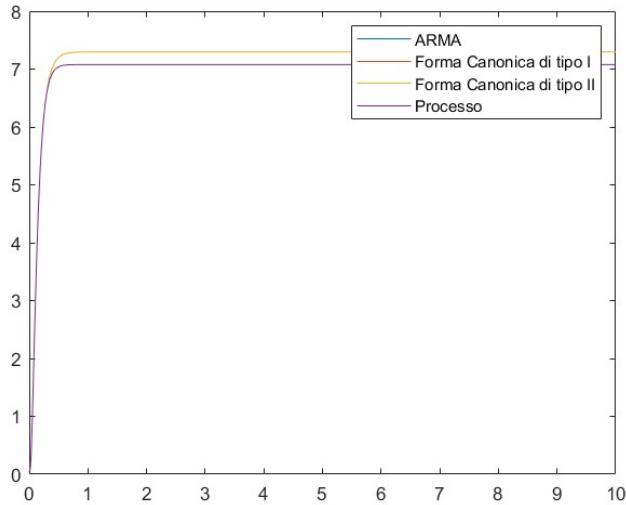


Possiamo notare come la risposta del processo descritto dalla black-box è approssimato fedelmente sia dal modello ARMA, sia dalla Forma Canonica di tipo I e sia dalla Forma Canonica II, avendo queste ultime tre risposte del sistema dinamico coincidenti essendo equivalenti tra loro.

Risposta al gradino

Calcolo e rappresento la risposta al gradino del processo e del modello ARMA, della Forma Canonica I e della Forma Canonica II così da vedere se la risposta delle tre rappresentazioni risultano approssimare fedelmente la risposta del processo in black-box. Allego il codice e il plot corrispondente:

```
figure(6); title('step output');
step_output = validazione_modello.step_output; plot(time, step_output);
legend('ARMA', 'Forma Canonica di tipo I', 'Forma Canonica di tipo II', 'Processo');
```



Posso notare che le risposte delle rappresentazioni risultano essere quasi perfettamente uguali rispetto alla risposta al gradino del processo tranne che per un piccolo scarto dopo che la risposta al gradino si assesta al suo valore di regime. Questo *distacco* è dovuto al fatto che il gradino non è un ingresso perturbato, cioè non è persistentemente eccitato, e tale che la matrice dei regressori corrispondente presenta delle colonne linearmente dipendenti dalle altre anziché avere tutte le colonne linearmente indipendenti. Di conseguenza, non riusciamo in questo caso a identificare bene i parametri b tale da far comparire un distacco nella risposta. Se proprio si vuole scegliere il gradino come ingresso è buona norma considerare un segnale gradino sommato con del rumore così da avere un segnale perturbato. Comunque, posso notare come l'errore assoluto sia relativamente basso:

```
stepP = validazione_modello.step_processo; stepA = validazione_modello.step_ARMA;
err_ass_step_array = abs(stepA-stepP);
err_ass_step = mean( err_ass_step_array );
```

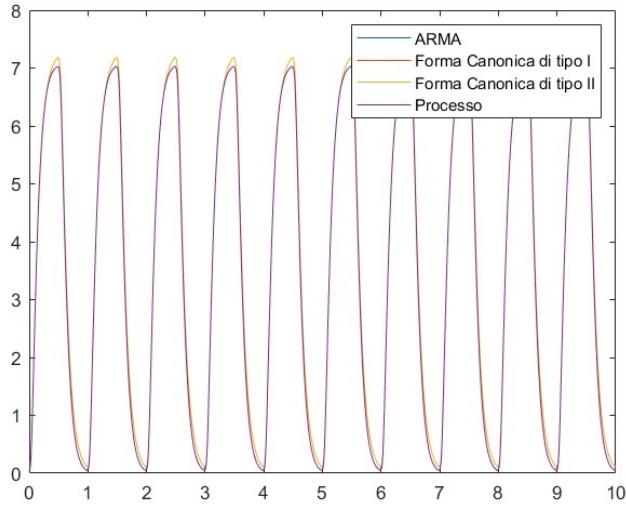
tale che l'errore assoluto della risposta al gradino rispetto al modello approssimato ARMA e il modello vero della black-box è pari a 0.209282360701343.

Risposta al treno di impulsi

Calcolo e rappresento la risposta al treno di impulsi del processo e del modello ARMA, della Forma Canonica I e della Forma Canonica II così da vedere se la risposta delle tre

rappresentazioni risultano approssimare fedelmente la risposta del processo in black-box. Allego il codice e il plot corrispondente:

```
figure(7); title('pulse output');
pulse_output = validazione_modello.pulse_output; plot(time, pulse_output);
legend('ARMA','Forma Canonica di tipo I','Forma Canonica di tipo II','Processo');
```



Posso notare come le risposta al treno di impulsi delle tre rappresentazioni risultano essere quasi perfettamente coincidenti con la risposta al treno di impulsi del processo in black-box tranne per un piccolo scarto in corrispondenza del picco massimo delle risposte. Infatti, posso notare come l'errore assoluto sia praticamente quasi nullo:

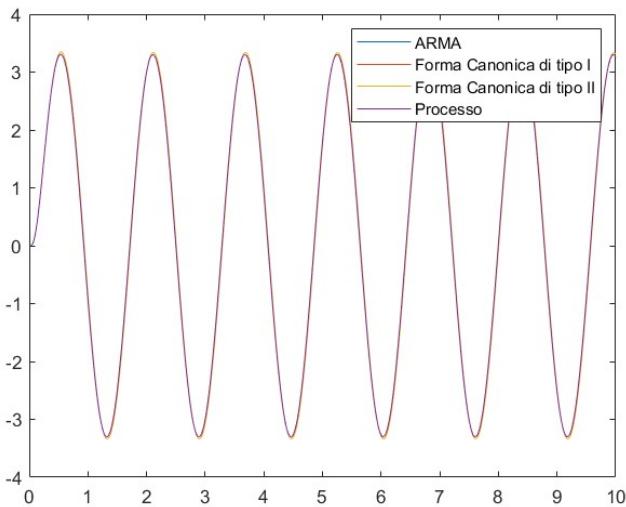
```
pulseP = validazione_modello.pulse_processo; pulseA = validazione_modello.pulse_ARMA;
err_ass_pulse_array = abs(pulseA-pulseP);
err_ass_pulse = mean( err_ass_pulse_array );
```

tale che l'errore assoluto della risposta al treno di impulsi rispetto al modello approssimato ARMA e il modello vero della black-box è pari a 0.011635107250898.

Risposta alla sinusoide

Calcolo e rappresento la risposta alla sinusoide del processo e del modello ARMA, della Forma Canonica I e della Forma Canonica II così da vedere se la risposta delle tre rappresentazioni risultano approssimare fedelmente la risposta del processo in black-box. Rappresento anche questa risposta, cioè rispetto ad un ulteriore segnale di ingresso, perché è di buona norma verificare se il modello ottenuto rispecchi fedelmente il processo della black-box identificato. Allego il codice e il plot corrispondente:

```
figure(8); title('sine wave output');
sine_wave = validazione_modello.sine_wave_output; plot(time, sine_wave);
legend('ARMA','Forma Canonica di tipo I','Forma Canonica di tipo II','Processo');
```



Posso notare come le risposte alla sinusoide delle tre rappresentazioni risultano essere quasi perfettamente coincidenti con la risposta alla sinusoide del processo in black-box tranne per un piccolo scarto in corrispondenza del picco massimo delle risposte. Infatti, posso notare come l'errore assoluto sia praticamente quasi nullo:

```
sine_waveP = validazione_modello.sine_wave_processo;
sine_waveA = validazione_modello.sine_wave_ARMA;
err_ass_sine_wave_array = abs( sine_waveA-sine_waveP );
err_ass_sine_wave = mean( err_ass_sine_wave_array );
```

tale che l'errore assoluto della risposta alla sinusoide rispetto al modello approssimato ARMA e il modello vero della black-box è pari a 0.061676429557399.

Esercizio 3

Considerando il segnale periodico $x = \sin\left(2\pi t + \frac{\pi}{6}\right) + \cos\left(2\pi 8t + \frac{\pi}{3}\right)$ posso notare che le corrispondenti ampiezze sono $A_1 = 1$ e $A_2 = 1$ e le corrispondenti frequenze sono $f_1 = 1$ e $f_2 = 8$. Inoltre, posso notare che le fasi delle rispettive armoniche sono $\varphi_1 = \frac{\pi}{6}$ e $\varphi_2 = \frac{\pi}{3}$. Dopo aver descritto il segnale periodico in questione posso definire il tempo di campionamento che mi permetterà di condurre un'apposita analisi di questo segnale per quanto riguarda le ampiezze e le fasi tramite l'algoritmo DFT che verrà introdotto in seguito.

Cos'è il campionamento?

Campionare un segnale $x(t)$ significa “estrarre” dal segnale stesso i valori che esso assume a istanti temporali equispaziati, cioè multipli di un intervallo T_c detto *periodo di campionamento*. Con questa operazione viene a crearsi una sequenza tale che il valore n -esimo $x[n]$ è il valore assunto dal segnale a tempo continuo all'istante nT_c :

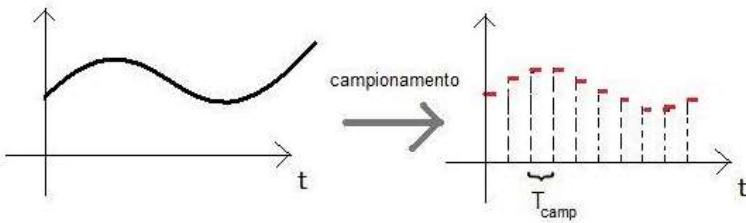
$$x[n] = x(nT_c)$$

L'operazione di campionamento viene simbolicamente effettuata da un dispositivo, il *campionatore*, indicato con una sorta di “*interruttore*” che si chiude per un intervallo di durata infinitesima. La cadenza con cui l'interruttore si chiude, cioè con la quale il segnale viene campionato, è pari a

$$f_c = \frac{1}{T_c}$$

e prende il nome di *frequenza di campionamento (sampling frequency)*, misurata in Hz o in *campioni/s*. L'operazione di campionamento viene effettuata dai convertitori analogico/digitale A/D. Questi dispositivi sono comandati da un segnale di clock (temporizzazione) alla frequenza f_c , che fornisce gli impulsi di comando al circuito per effettuare le varie operazioni di campionamento. Ovviamente, esiste anche il corrispondente D/A tale che il segnale a tempo discreto¹⁰ $y[n]$ risultante dall'elaborazione numerica deve essere poi convertito in forma analogica, cioè in un segnale a tempo continuo. Il dispositivo che produce in uscita un segnale a tempo continuo a partire da un segnale d'ingresso a tempo discreto è chiamato *interpolatore*.

¹⁰ Un segnale a tempo discreto è una successione x_n o sequenza $x[n]$ di numeri, ed è quindi rappresentabile con una funzione di variabile intera relativa avente valori reali o complessi.



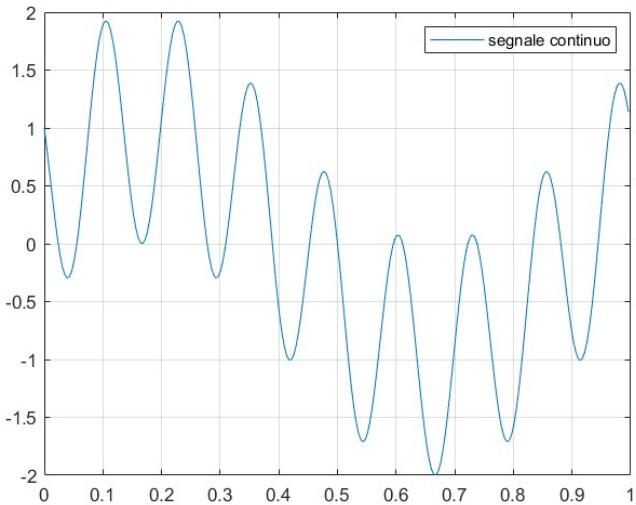
Ritornando al campionamento, ovviamente, più breve sarà il tempo di campionamento più il segnale campionato assomiglierà al segnale originale. Ovviamente, aumentando la frequenza di campionamento aumenta anche il numero di campioni (*sample*) che bisogna acquisire e occorrerà utilizzare dispositivi di campionamento più veloci. Pertanto, la scelta del periodo di campionamento dovrà dunque essere in generale un compromesso fra la necessità di riprodurre il più fedelmente possibile il segnale e i problemi tecnologici derivanti dal numero eccessivo di campioni da trattare e dalla rapidità del campionamento stesso. Ovviamente, per campionare correttamente, senza perdita di informazioni, un segnale a banda limitata bisogna rispettare un importante teorema: il *teorema di campionamento o teorema di Nyquist-Shannon*. Esso afferma che per campionare correttamente un segnale a banda limitata è sufficiente campionarlo con una frequenza di campionamento pari almeno al doppio della massima frequenza del segnale. Tale frequenza è conosciuta anche come *frequenza di Nyquist*. Il mancato rispetto di questo teorema comporta il fenomeno conosciuto come *aliasing*, cioè si ha un sottocampionamento del segnale analogico nel dominio del tempo. Nel nostro caso, essendo che il segnale è composto da due armoniche, considero la frequenza massima fra le due. Pertanto, avendo $f_1 = 1$ e $f_2 = 8$, per il teorema di Nyquist-Shannon, la frequenza di campionamento dovrà essere pari almeno a $f_c = 2 \cdot f_2 = 16$. Quindi, il tempo di campionamento t_c dovrà essere pari minore di $\frac{t_c}{2} = \frac{f_c}{2} = \frac{1}{2} \cdot \frac{1}{16} = 0.03125$. Pertanto, scelgo una frequenza di campionamento $f_c = 256$ tale da avere un tempo di campionamento $T_c = 0.00390625$. Inoltre, scelgo un numero di campioni $N = 256$ così da essere in grado di implementare il dominio temporale nella seguente maniera:

$$tn = [0:Tc:(N - 1) * Tc]$$

Poteva essere anche definito come $tn = [0:Tc:\frac{1}{f} - Tc]$ tale che il numero di campioni N sarebbe stato pari a *length(tn)*.

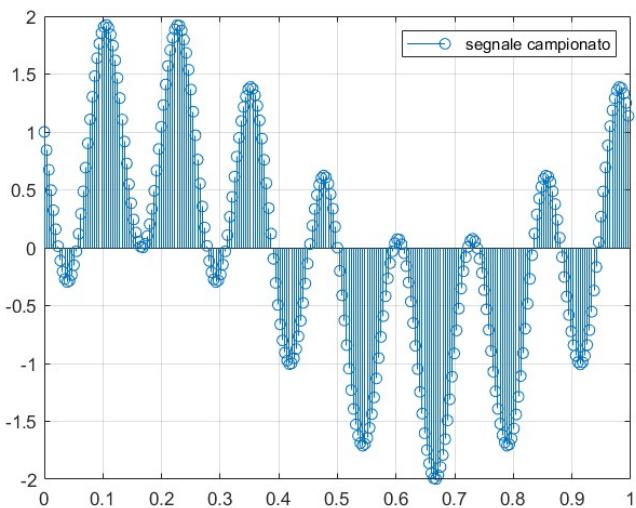
Pertanto, ora sono in grado di rappresentare il segnale continuo in questione plottandolo tramite i seguenti comandi:

```
x = A1*sin(2*pi*f1*tn + phi1) + A2*cos(2*pi*f2*tn + phi2);
figure(1);
plot( tn, x );
legend('segnale continuo'); grid;
```



Ora, considerando il fatto che il segnale x in questione viene sottoposto a campionamento con periodo di campionamento T_c e quindi verranno estrapolati da x bene f_c valori al secondo, possiamo plottere il corrispondente segnale campionato tramite i seguenti comandi MATLAB:

```
figure(2);
stem( tn, x );
legend('segnalet campionato'); grid;
```



Il comando `stem(X,Y)` traccia la sequenza di dati Y ai valori specificati da X . Gli input X e Y , ovviamente, devono essere vettori o matrici della stessa dimensione.

Dopo aver plotto il segnale campionato, posso ora introdurre quella che viene conosciuta come DFT che sarà utile per l'analisi delle ampiezze e delle fasi del segnale. Prima di descrivere cos'è la DFT bisogna spiegare cos'è una sequenza periodica e il relativo campionamento.

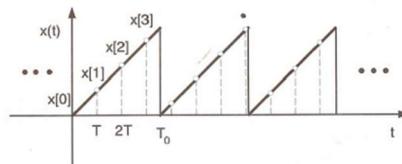
Cos'è una sequenza periodica? E come si ricava la DFT?

Una sequenza $x[n]$ è periodica se esiste un intero positivo N_0 (*il periodo della sequenza*) per il quale è verificata la seguente relazione:

$$x[n] = x[n + N_0]$$

per ogni valore della variabile n . Una sequenza $x[n]$ periodica di periodo N_0 è individuata, quindi, da N_0 numeri reali (o complessi) che rappresentano i valori assunti da $x[n]$ in un periodo, ad esempio nell'intervallo $n = 0, 1, \dots, N_0 - 1$.

Si può osservare che il campionamento di un segnale periodico a tempo continuo non genera necessariamente una sequenza periodica. È necessario che un numero intero N_0 di intervalli di campionamento sia pari ad un qualche numero intero m di periodi di ripetizione del segnale originario: $N_0 T = m T_0$. In pratica, gli impulsi di campionamento del convertitore A/D devono essere sincronizzati con il segnale periodico analogico: essi non possono avere una cadenza arbitraria senza un legame preciso con la cadenza di ripetizione fondamentale del segnale dato. Se il rapporto $\frac{T}{T_0}$ non è un numero razionale, l'operazione di campionamento non dà origine ad una sequenza $x[n]$ periodica.



Sequenza non periodica generata dal campionamento di un segnale periodico a tempo continuo

Supponendo che $x[n]$ sia una sequenza periodica di periodo N_0 , essa può essere rappresentata mediante uno sviluppo del tutto analogo alla serie di Fourier per i segnali periodici a tempo continuo, chiamato *serie discreta o antitrasformata discreta di Fourier*:

$$x[n] = \sum_{k=0}^{N_0-1} \overline{X_k} e^{j \frac{2\pi k n}{N_0}}$$

La sequenza $\overline{X_k}$ dei coefficienti discreti di Fourier è chiamata *trasformata discreta di Fourier* della sequenza periodica data ed è pari a

$$\overline{X_k} = \frac{1}{N_0} \sum_{n=0}^{N_0-1} x[n] e^{-j \frac{2\pi k n}{N_0}}$$

Possiamo notare come la trasformata periodica di una sequenza $x[n]$ periodica di periodo N_0 è essa stessa periodica con il medesimo periodo:

$$\begin{aligned}\overline{X_{k+N_0}} &= \frac{1}{N_0} \sum_{k=0}^{N_0-1} x[n] e^{-j\frac{2\pi(k+N_0)n}{N_0}} = \frac{1}{N_0} \sum_{k=0}^{N_0-1} x[n] e^{-j\frac{2\pi kn}{N_0}} e^{-j2\pi n} \\ &= \frac{1}{N_0} \sum_{k=0}^{N_0-1} x[n] e^{-j\frac{2\pi kn}{N_0}} = \overline{X_k}\end{aligned}$$

Pertanto, la sequenza periodica, come nel caso del segnale periodico a tempo continuo, è espressa da una somma di oscillazioni sinusoidali a frequenze armoniche, cioè multiple di una *frequenza fondamentale*:

$$x[n] = \sum_{k=0}^{N_0-1} \overline{X_k} e^{j\frac{2\pi k}{N_0 T} n T}$$

tale che i vari esponenziali complessi oscillano alle frequenze $f_k = \frac{k}{N_0 T}$ con $k = 0, \dots, N_0 - 1$.

Quello che a noi interessa è analizzare la rappresentazione spettrale dei segnali che è basata fondamentalmente sulla trasformata discreta di Fourier o meglio sull'algoritmo veloce della FFT.

Cos'è la rappresentazione spettrale di un segnale?

La rappresentazione spettrale dei segnali è una descrizione dei segnali nel dominio della frequenza. In tale descrizione ogni frequenza di cui è composto un segnale è detta armonica e da un punto di vista matematico ad ogni armonica corrisponde un vettore di una base di uno spazio vettoriale infinito-dimensionale con prodotto interno scalare sul campo complesso, ovvero la base di uno spazio di Hilbert¹¹. Pertanto, il segnale viene scritto come combinazione lineare in tale spazio. Infatti, considerando due vettori $x, y \in \mathbb{C}^N$ e definendo il loro prodotto scalare nel seguente modo:

$$\langle x, y \rangle = \sum_{n=0}^{N-1} x(n) \bar{y}(n)$$

(la proiezione di x lungo y)

la Discrete Fourier Transform *DFT* è

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{-kn}$$

tale che la DFT è anch'esso un prodotto scalare. Pertanto, essa sarà pari a

¹¹ Uno spazio di Hilbert $\mathbf{H}=(\mathbf{H}, \langle \cdot, \cdot \rangle)$ è uno spazio vettoriale \mathbf{H} reale o complesso sul quale è definito un prodotto interno scalare $\langle \cdot, \cdot \rangle$ tale che, detta d la distanza indotta da $\langle \cdot, \cdot \rangle$ su \mathbf{H} , lo spazio metrico (\mathbf{H}, d) sia completo, cioè dato un insieme di punti nel quale è definita una distanza (*metrica*) tutte le successioni di Cauchy definite in esso sono convergenti ad un elemento dello spazio e, quindi, un punto definito in esso.

$$\begin{bmatrix} 1 & W_N^{-k} & W_N^{-2k} & \dots & W_N^{-(N-1)k} \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

Quindi, potremo scrivere il tutto come:

$$\begin{bmatrix} X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \dots & W_N^{-(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & \dots & \dots & W_N^{-(N-1)(N-1)} \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

Tale matrice, in generale, si implementa tramite un doppio for tale che si potrebbe ottimizzare il tutto poiché in tale matrice sono presenti delle simmetrie. Analogamente per ottenere la *IDFT* basta considerare la matrice relativa alla *DFT* e calcolarne l'inversa. Possiamo notare che la coppia *DFT-IDFT* è un'operazione di cambiamento di base, cioè avendo il vettore x di partenza lo mappiamo in un dominio complesso. I vettori della nuova base sono le colonne della matrice di trasformazione e dipendono dai termini W_N^{kn} . Tale base è una base ortonormale, cioè una base composta da vettori v_1, \dots, v_n a due a due ortonormali tale che ogni vettore ha norma¹² uno.

La *DFT* in MATLAB

In MATLAB, la trasformata di Fourier discreta viene implementata utilizzando il comando `fft(x)` che calcola la *DFT* del segnale x utilizzando l'algoritmo veloce *FFT*. Un algoritmo *FFT* ottiene lo stesso risultato della *DFT* con un numero di operazioni $\mathcal{O}(N \log N)$ anziché una quantità di operazioni aritmetiche $\mathcal{O}(N^2)$ nell'ipotesi che N sia una potenza di 2. Nel caso in cui il numero di campioni non soddisfi l'ipotesi si aggiungono dei campioni nulli (*operazione di Zero Padding*), cioè viene calcolata la *FFT* su N punti aggiungendo zeri, se x ha meno di N punti, e troncando se ne ha di più.

Pertanto, implemento la rappresentazione spettrale in MATLAB nel seguente modo:

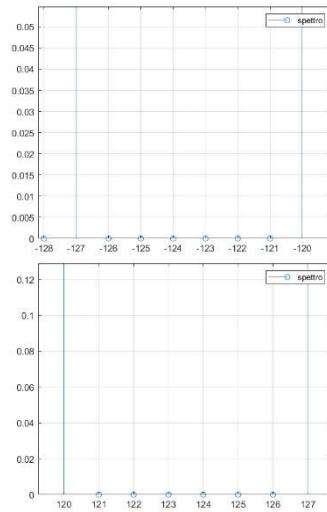
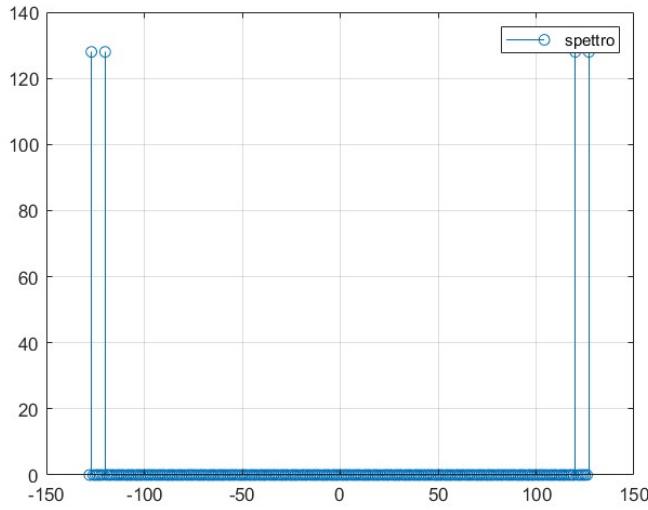
```
x = fft(x);

wc = 2*pi/Tc;
fc = 1/Tc;
wk = [-fc/2: fc/N :fc/2-fc/N];

figure(3);
stem( wk, abs(X) );
grid; legend('spettro');
```

¹² Funzione che assegna ad un vettore una lunghezza positiva.

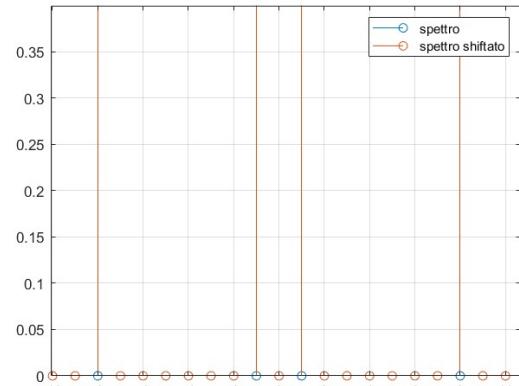
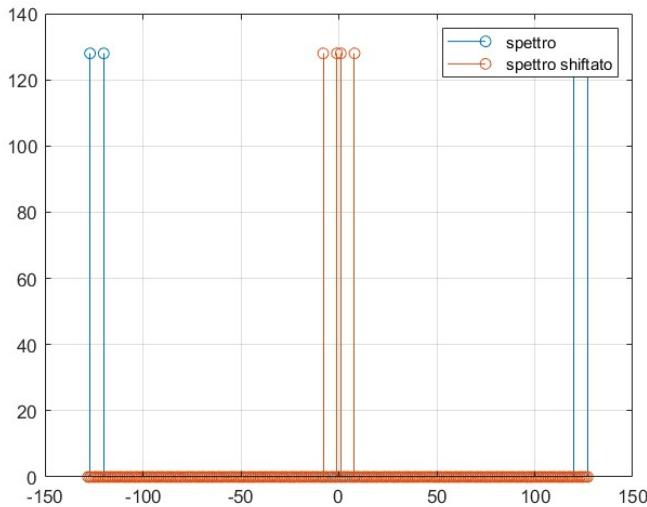
ottenendo, di conseguenza, la seguente rappresentazione:



Posso notare che i 4 impulsi corrispondenti alle due armoniche che compongono il segnale x risultano essere collocati alle frequenze $f_{11} = -120\text{Hz}$, $f_{12} = 120\text{Hz}$, $f_{21} = -127\text{Hz}$ e $f_{22} = 127\text{Hz}$. Tale risultato non è corretto perché questo starebbe a significare che le frequenze delle due armoniche dovrebbero essere rispettivamente **120Hz** e **127Hz**. Pertanto, “correggo” tale risultato effettuando uno shifting nel dominio delle frequenze. Tanto è vero che la funzione *fftshift* presente in MATLAB permette di *riordinare* le frequenze e ottenere la giusta rappresentazione frequenziale:

```
figure(4);
stem( wk, abs(X) );
hold on;
Xs = fftshift(X);
stem( wk, abs(Xs) );
grid; legend('spettro', 'spettro shiftato');
```

ottenendo, di conseguenza, la seguente rappresentazione:



Posso notare che i 4 impulsi ora ricadono precisamente nelle rispettive frequenze: -1, 1 e -8, 8.

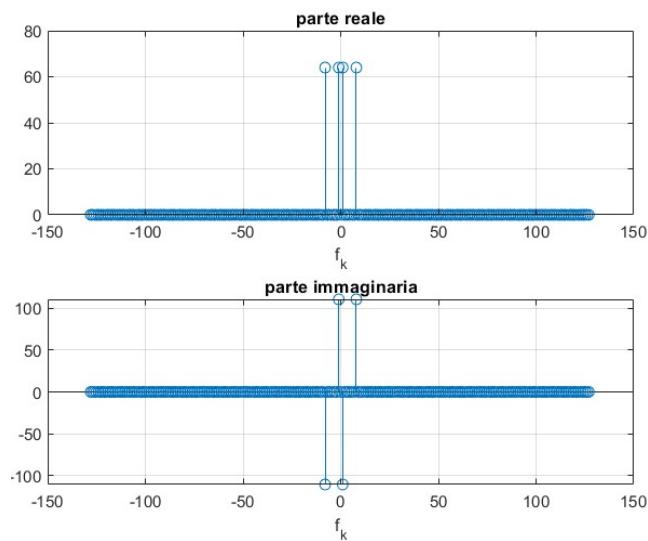
Anche se la parte informativa della *DFT* sta tutta nel modulo dello spettro, la *Discrete Fourier Transform* è comunque una sequenza di numeri complessi, pertanto, essi hanno parte reale, parte immaginaria e, quindi, anche una fase. *Ripulisco* il segnale da eventuali termini che possono essere soggetti ad errori numerici così da avere una *DFT* “pulita”:

```
Xreal = real(Xs);
j = find( abs(Xreal) <= 10e-10 );
Xreal(j) = 0;

Ximaginary = imag(Xs);
j = find( abs(Ximaginary) <= 10e-10 );
Ximaginary(j) = 0;

figure(5);
subplot(2,1,1), stem( wk, Xreal );
title('parte reale'); xlabel('f_k'); grid;
subplot(2,1,2), stem( wk, Ximaginary );
title('parte immaginaria'); xlabel('f_k'); grid;
```

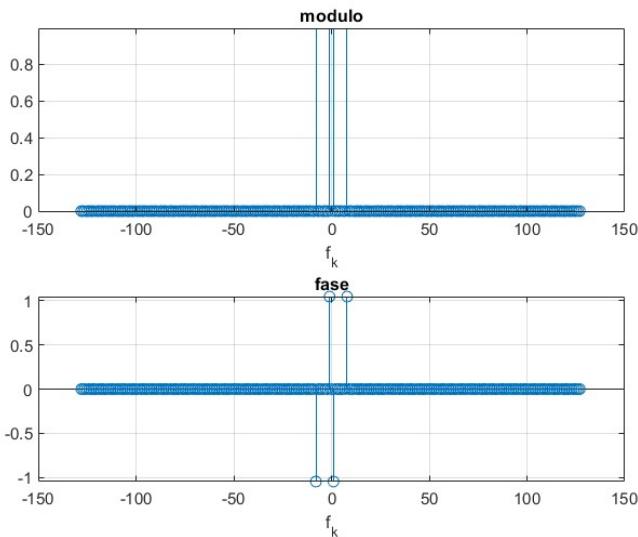
ottenendo le seguenti rappresentazioni per la parte reale e immaginaria:



Posso notare che la parte reale e il modulo della trasformata sono simmetrici rispetto all'origine (funzione pari), mentre la parte immaginaria e la fase sono anti-simmetriche (funzione dispari). Pertanto, ora plotto la rappresentazione del modulo e della fase corrispondenti. Prima, però, considero il modulo normalizzato (essendo che moltiplico per $\frac{2}{N}$), cioè faccio in modo che dato un segnale generico $f(t)$, il modulo sia tale che $|f(t)| \leq 1$.

```
figure(6);
subplot(2,1,1), stem( wk, (2/N)*abs(complex(Xreal, Ximaginary)) );
title('modulo'); xlabel('f_k'); grid;
subplot(2,1,2), stem(wk, angle(complex(Xreal, Ximaginary)) );
title('fase'); xlabel('f_k'); grid;
```

ottenendo le seguenti rappresentazioni per il modulo (*spettro di ampiezza*) e la fase (*spettro di fase*):



Infatti, come già anticipato prima, il modulo della trasformata è simmetrico rispetto all'origine mentre la fase è anti-simmetrica.

Segnale non noto

Nella realtà, però, solitamente il segnale non è noto ma bisogna estrapolarne le informazioni da esso in maniera indiretta. Pertanto, considerando sempre il segnale in questione, posso tentare di troncarlo anche se non è detto che riusciremo nell'intento. Anzi, solitamente i segnali nella realtà sono parecchio lontani dalle solite sinusoidi e, pertanto, risulta essere molto difficoltoso riuscire a scegliere un tempo di campionamento esatto tale da effettuare una troncatura adeguata. Ipotizzando di non conoscere il periodo fondamentale, aggiungo un contributo non multiplo del tempo di campionamento T_c rispetto all'asse temporale precedentemente considerato:

```
t = [-T/2 : Tc : T/2 - Tc + 0.56];
x = A1*sin(2*pi*f1*t+phi1)+A2*cos(2*pi*f2*t+phi2);
N = length(x);
```

In questa maniera le frequenze non rientrano nel set discreto delle frequenze.

Dispersione Spettrale

Pertanto, ora, eseguo la *DFT* e plotto il risultato:

```
X = fft(x);
Xs = fftshift(X);

Xreal = real(Xs);
j = find( abs(Xreal) <= 10e-10 );
Xreal(j) = 0;

Ximaginary = imag(Xs);
j = find( abs(Ximaginary) <= 10e-10 );
```

```

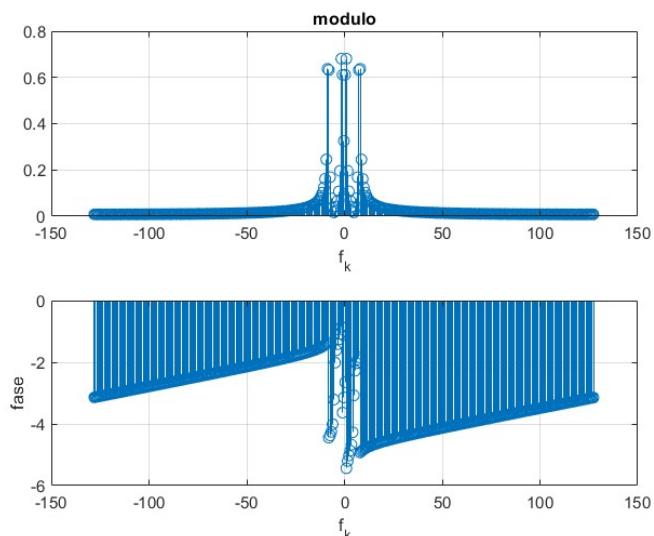
Ximaginary(j) = 0;

bin = fc/N;
fk = [-fc/2 : bin : (fc/2) - bin];

figure(1);
subplot(2,1,1),stem(fk,2*abs(complex(Xreal,Ximaginary))/N);
xlabel('f_k'); title('modulo'); grid;
subplot(2,1,2),stem(fk,(unwrap(angle(complex(Xreal,Ximaginary))))) ;
xlabel('f_k'); ylabel('fase'); grid;

```

Dove *bin* è la lunghezza del bin su cui andiamo a definire poi il dominio delle frequenze. Abbiamo, inoltre, *fftshift* che effettua lo shifting delle frequenze ottenute con la *fft* e anche la normalizzazione del modulo nel *subplot*. Invece, la funzione *unwrap* permette di scartare gli angoli quando il salto tra angoli consecutivi è maggiore o uguale a π radianti tale che essa aggiungerà multipli di $\pm 2\pi$ finché il salto non è inferiore a π .



Possiamo notare evidente dispersione spettrale, cioè il fenomeno per cui non vi è una corrispondenza tra la k -esima frequenza e ampiezza/fase del segnale in questione. Infatti, possiamo notare come le armoniche sono presenti anche in corrispondenza di frequenze non *ammissibili*. Quindi, l'idea è quella di pre-processare il segnale che si va ad analizzare in modo da renderlo molto vicino a zero sia all'inizio che alla fine, cioè utilizzando la cosiddetta *operazione di affusolatura del segnale* o anche conosciuto come *tapering* (ovviamente senza perdere le caratteristiche spettrali fondamentali). Esso si ottiene moltiplicando il segnale campionato $X(n)$ per una *funzione finestra* $W(n)$, cioè un insieme di campioni della stessa lunghezza tale che il *segnale finestrato* $Y_w(n)$ sarà del tipo:

$$Y_w(n) = X(n) \cdot W(n)$$

(prodotto componente per componente)

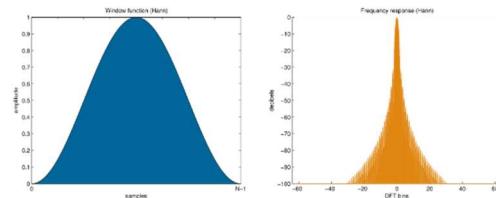
Le funzioni finestra sono diverse. Tra le più conosciute ci sono quelle di *Hann* e di *Bartlett*:

Funzione di Hann

Anche conosciuta come *funzione finestra a campana*, essa fa parte insieme alla funzione finestra di Hamming della famiglia nota come *finestre a coseno rialzato* e presenta la seguente funzione:

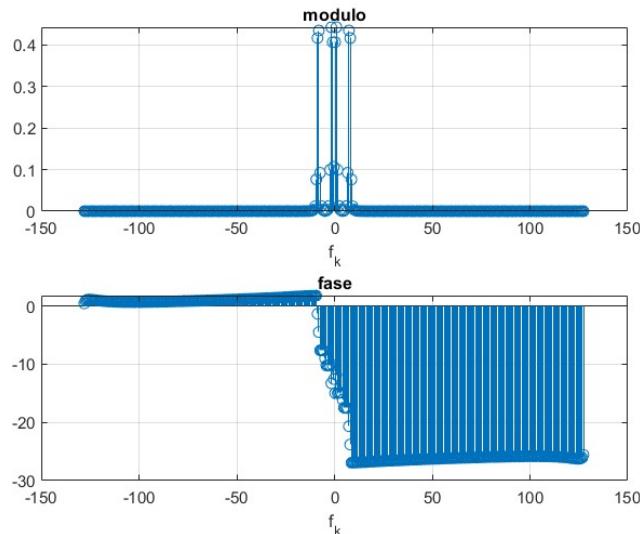
$$W(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right)$$

con $\alpha = 0.5$ e $\beta = \pm 0.5$.



Allego il codice e il corrispondente plot ottenuto applicando l'operazione di finestratura tramite la funzione finestra di Hann:

```
w1 = hanning(length(x))';
y1 = x.*w1;
X1 = fftshift(fft(y1));
Xr1 = real(X1); j = find(abs(Xr1)<=10e-10); Xr1(j) = 0;
Xi1 = imag(X1);j = find(abs(Xi1)<=10e-10); Xi1(j) = 0;
figure(2);
subplot(2,1,1),stem(fk,2*abs(complex(Xr1,Xi1))/N);
xlabel("f_k") ; title('modulo'); grid;
subplot(2,1,2),stem(fk,(unwrap(angle(complex(Xr1,Xi1))))));
xlabel("f_k"); title('fase'); grid;
```

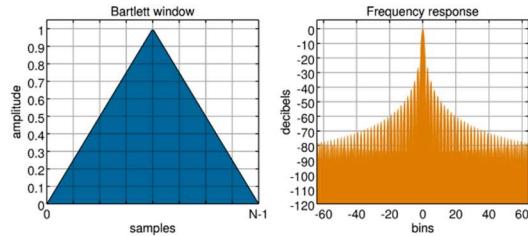


Si può notare come le componenti spettrali dominanti rimaste siano concentrate per lo più nelle frequenze corrispondenti al segnale in questione, cioè -1, 1 e -8, 8 mentre nel resto del grafico, cioè nelle frequenze non di interesse, le componenti spettrali sono state praticamente annullate.

Funzione di Bartlett

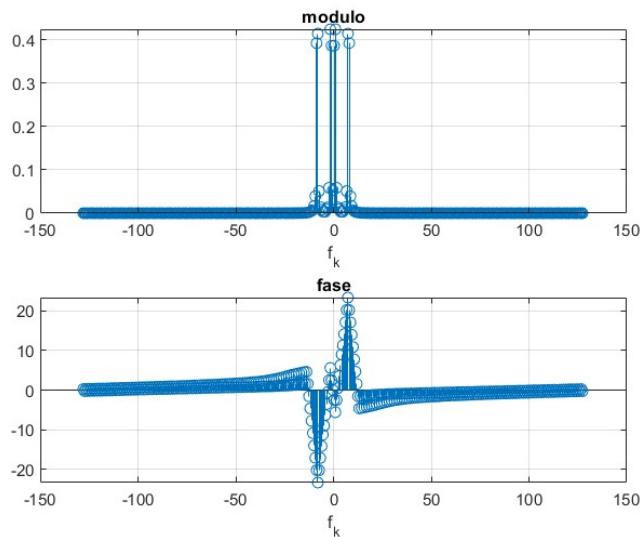
Anche conosciuta come *funzione finestra rettangolare*, essa presenta la seguente funzione:

$$W(n) = 1 - \left| \frac{n - \frac{N-1}{2}}{\frac{N}{2}} \right|$$



Allego il codice e il corrispondente plot ottenuto applicando l'operazione di finestratura tramite la funzione finestra di Bartlett:

```
w2 = bartlett(length(x))';
y2 = x.*w2;
X2 = fftshift(fft(y2));
Xr2 = real(X2); j = find(abs(Xr2)<=10e-10); Xr2(j) = 0;
Xi2 = imag(X2);j = find(abs(Xi2)<=10e-10); Xi2(j) = 0;
figure(3);
subplot(2,1,1),stem(fk,2*abs(complex(Xr2,Xi2))/N);
xlabel("f_k"); title('modulo'); grid;
subplot(2,1,2),stem(fk,(unwrap(angle(complex(Xr2,Xi2))))));
xlabel("f_k"); title('fase'); grid;
```

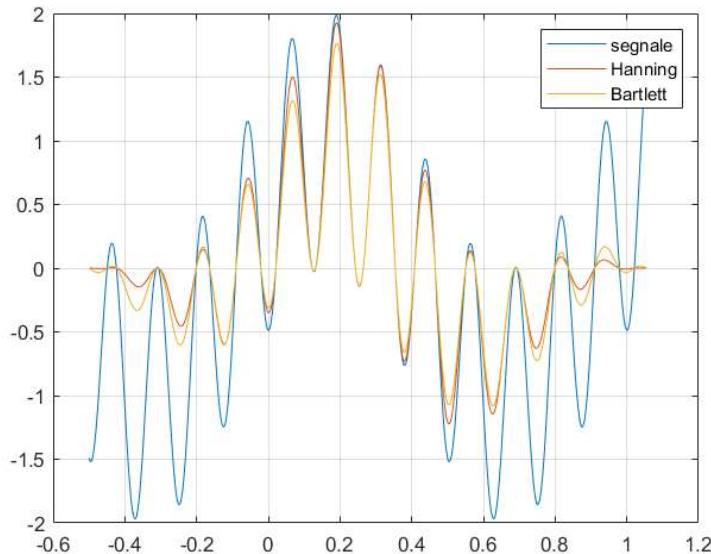


Anche in questo caso le componenti spettrali sono concentrate nelle frequenze di interesse, mentre il resto è stato praticamente annullato.

Finestratura

Infine, effettuo il plot della finestratura applicato al segnale in questione:

```
figure(4);
plot(t,x); hold on; plot(t,y1); hold on; plot(t,y2);
grid; legend('segnale','Hanning','Bartlett');
```



Possiamo notare, come già anticipato prima, che le funzioni di finestratura analizzano il segnale tale da renderlo molto vicino a zero sia all'inizio che alla fine, cioè effettuano proprio la cosiddetta *affusolatura del segnale*.