Progetto Controlli Automatici

### Autori: Raffaele Giacomo Giovanni Di Maio & Emilio Meroni

# Esperimenti Equilibrio

clc

clear

close all

addpath(genpath("..\"));

set(groot, 'DefaultTextInterpreter', 'latex');

set(groot, 'DefaultAxesTickLabelInterpreter', 'latex');

set(groot, 'DefaultLegendInterpreter', 'latex');

load("modello\data.mat");

warning("off" , "all");

load\_system("modello\model.mdl");

N = length(time);

Ts = time(2) - time(1);

colori = lines(2);

Simulazione con i segnali dati:

sim("model");

i = 1

figure;

subplot(2 , 1 ,1);

plot(time , y , Color=colori(1 , :));

ylabel("Angolo Ginocchio")

grid on;

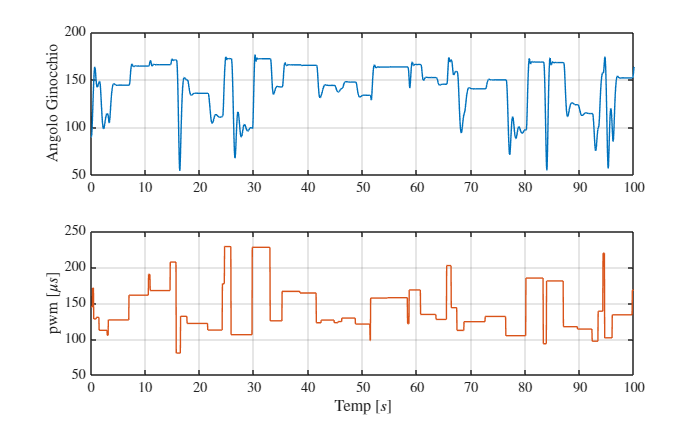
subplot(2 , 1 , 2);

plot(time , u , Color=colori(2 , :));

ylabel("pwm $[\mu s]$");

xlabel("Temp $[s]$");

grid on;



### Test di diverse intensità

pw = [zeros(201,1); 25 \* ones(200 , 1) ; 50 \* ones(200 , 1);

75\* ones(200 , 1);100 \* ones(200 , 1); linspace(100 , 120 , 200)';

120\*ones(800 , 1)];

sim("model");

i = 1

figure

subplot(2 , 1 , 1);

plot(time , y , DisplayName="Angolo Ginocchio" , Color=colori(1 , :));

ylim([80 180]);

ylabel("Angolo Ginocchio");

xlabel("Temp $[s]$");

grid on

subplot(2 , 1 , 2);

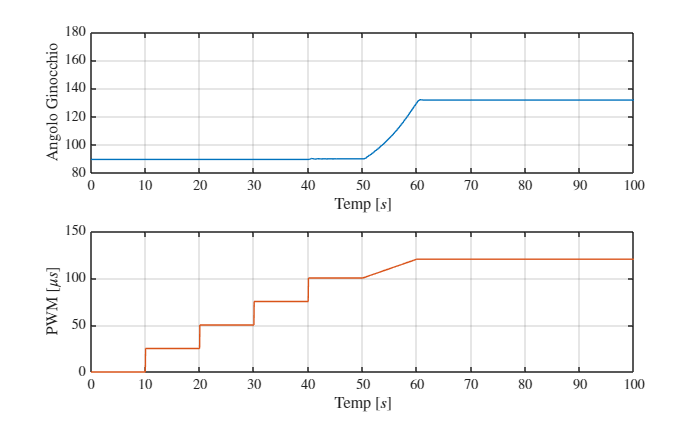
plot(time , u , DisplayName="PWM" , Color=colori(2 , :));

ylim([0 150]);

ylabel("PWM $[\mu s]$");

xlabel("Temp $[s]$");

grid on



# Si sceglie un punto di equilibrio a

Risposta allo scalino:

pw\_eq = 121.35;

pw = [zeros(10/Ts , 1) ; % 10 secondi a 0

pw\_eq \* ones(20/Ts , 1); % 20 secondi a 121.35

zeros(70/Ts +1 , 1)];

sim("model");

i = 1

figure

subplot(2 , 1 , 1);

plot(time , y , DisplayName="Angolo Ginocchio" , Color=colori(1 , :));

ylabel("Angolo Ginocchio");

xlabel("Temp $[s]$");

xlim([0 30])

grid on

subplot(2 , 1 , 2);

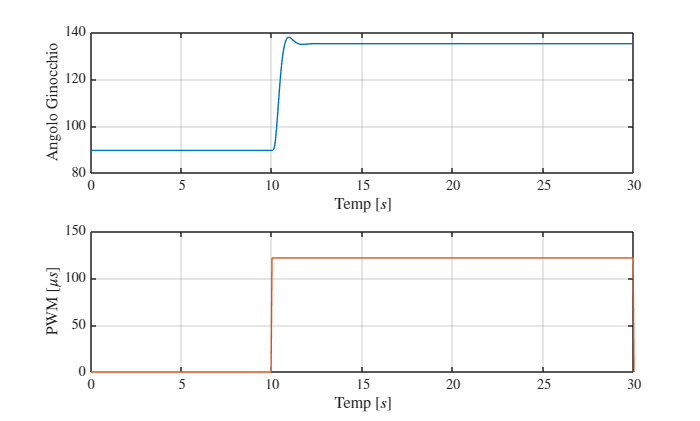
plot(time , u , DisplayName="PWM" , Color=colori(2 , :));

ylabel("PWM $[\mu s]$");

xlabel("Temp $[s]$");

xlim([0 30])

grid on



### Osservazione del ritardo puro d'ingresso

Si trovano circa  di ritardo che corrispondono a passi di simulazione:

figure;

indice\_inizio = floor(9.95/Ts);

indice\_fine = floor(10.5/Ts);

intervallo = indice\_inizio:indice\_fine;

y\_zoom = y(intervallo);

passo\_1 = 202;

passo\_2 = 203;

time\_zoom = time(intervallo);

hold on

plot(time\_zoom , y\_zoom);

plot(time(passo\_1) , y(passo\_1) , Marker="x" , LineWidth=3);

plot(time(passo\_2) , y(passo\_2) , Marker="x" , LineWidth=3);

xline(10)

ylabel("Angolo Ginocchio");

xlabel("Tempo $[s]$")

hold off;

xlim([9.875 10.133])

ylim([87.78 92.21])

hDataTip = findobj(gca,"DataIndex",1);

set(hDataTip,"Location","northwest");

ax2 = gca;

chart = ax2.Children(2);

datatip(chart,10.1,89.95,"Location","southeast");

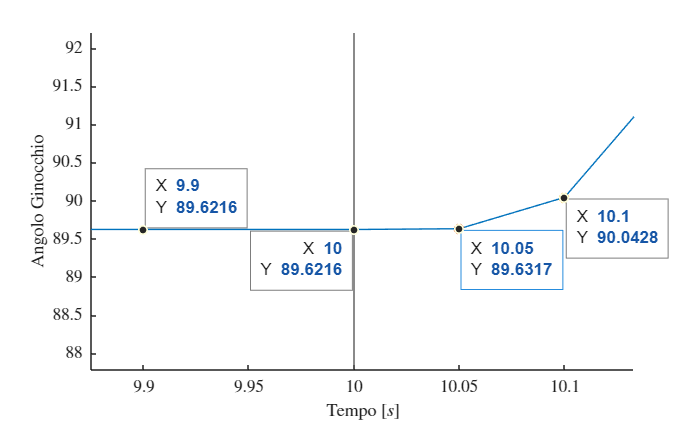
chart = ax2.Children(4);

datatip(chart,10,89.62,"Location","southwest");

datatip(chart,9.9,89.62,"Location","northeast");

chart = ax2.Children(3);

datatip(chart,10.05,89.63,"Location","southeast");



**Identificazione**

Si vuole controllare l'angolo nell'introno  rispetto all'equilibrio (, ) si è scelto di creare 3 segnali con intensità in un range di circa  e :

1. Randomico a blocchi costanti di  con valori compresi tra  e ;
2. PRBS con ampiezza  e ;
3. Valori randomici tra  e  a blocchi con tempi variabili tra  e .

pw\_max = 135;

pw\_min = 113.5;

% i primi tre secondi tolti poiché il segnale non è stazionario in quel

% periodo

primi\_secondi = 3/0.05;

time\_ridotto = time(primi\_secondi:end);

## Randomico a blocchi costanti di con valori compresi tra e

load("pw\_1.mat");

pw = pw\_1;

sim("model");

i = 1

y\_1 = y(primi\_secondi:end);

y\_1\_media = mean(y\_1);

y\_1\_no\_media = y\_1 - y\_1\_media;

pw\_1 = pw\_1(primi\_secondi:end);

pw\_1\_media = mean(pw\_1);

pw\_1\_no\_media = pw\_1 - pw\_1\_media;

figure;

subplot(2 , 1 , 1);

plot(time\_ridotto , y\_1, Color = colori(1 ,:));

ylabel("Angolo Ginocchio");

xlabel("Tempo $[s]$")

xlim([5 100])

grid on;

subplot(2 , 1 , 2);

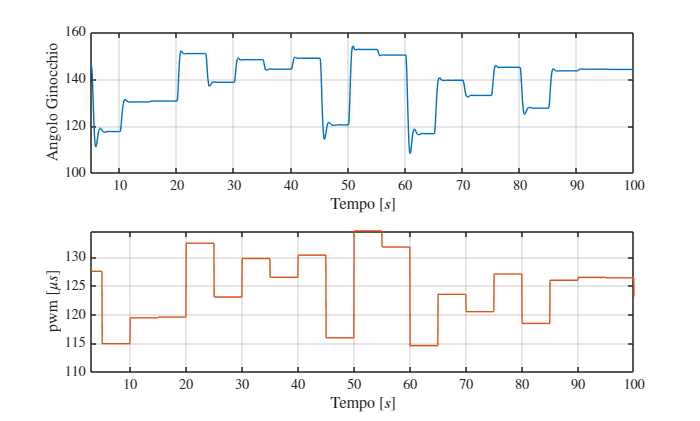
plot(time\_ridotto , pw\_1, Color = colori(2 ,:));

ylabel("pwm $[\mu s]$")

xlabel("Tempo $[s]$")

xlim([3 100])

grid on



## PRBS con ampiezza e ;

range\_PRBS = [0 1];

pw\_2 = idinput(N , 'prbs' , range\_PRBS , [pw\_min , pw\_max]);

pw = pw\_2;

sim("model");

i = 1

y\_2 = y(primi\_secondi:end);

y\_2\_media = mean(y\_2);

y\_2\_no\_media = y\_2 - y\_2\_media;

pw\_2 = u(primi\_secondi:end);

pw\_2\_media = mean(pw\_2);

pw\_2\_no\_media = pw\_2 - pw\_2\_media;

figure;

subplot(1 , 1 , 1);

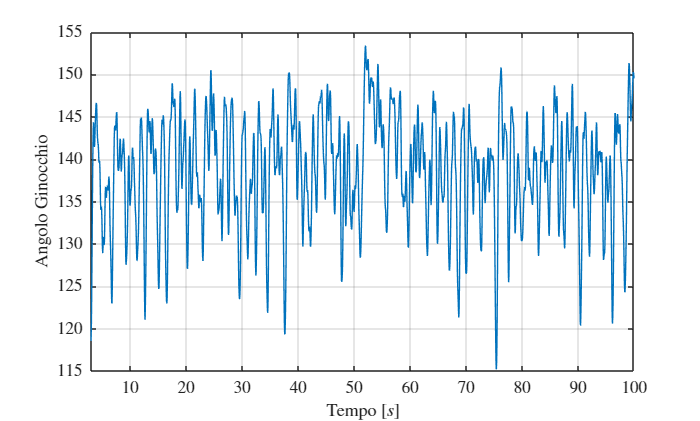
plot(time\_ridotto , y\_2, Color = colori(1 ,:));

ylabel("Angolo Ginocchio");

xlabel("Tempo $[s]$")

xlim([3 100])

grid on;



## Valori randomici tra e a blocchi con tempi variabili tra e

load("pw\_3.mat");

pw = pw\_3;

sim("model");

i = 1

y\_3 = y(primi\_secondi:end);

y\_3\_media = mean(y\_3);

y\_3\_no\_media = y\_3 - y\_3\_media;

pw\_3 = u(primi\_secondi:end);

pw\_3\_media = mean(pw\_3);

pw\_3\_no\_media = pw\_3 - pw\_3\_media;

figure;

subplot(2 , 1 , 1);

plot(time\_ridotto , y\_3 , Color=colori(1 , :));

ylabel("Angolo Ginocchio");

xlabel("Tempo $[s]$")

xlim([3 100])

grid on;

subplot(2 , 1 , 2);

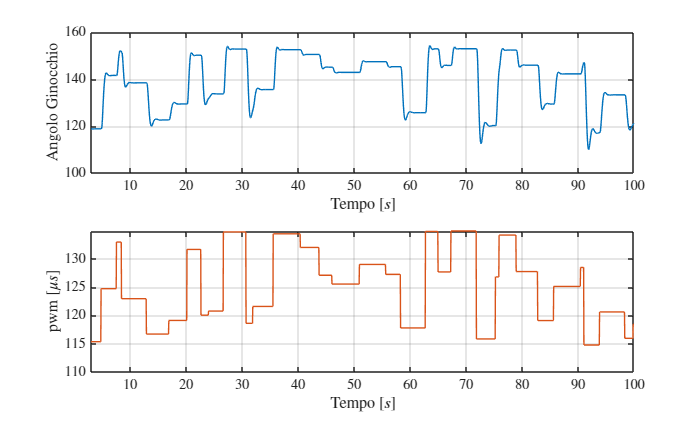
plot(time\_ridotto , pw\_3 , Color=colori(2 , :));

ylabel("pwm $[\mu s]$")

xlabel("Tempo $[s]$")

xlim([3 100])

grid on



# Identificazione del Modello

## Si utilizza il segnale randomico a blocchi costanti di

% Divisione del data set 70% e 30%

N = length(y\_1);

n\_70 = floor(N\*0.7);

y\_modello = y\_1\_no\_media(1:n\_70);

u\_modello = pw\_1\_no\_media(1:n\_70);

data\_ident = iddata(y\_modello , u\_modello , Ts);

% Validazione con il 30% del primo set

n\_30 = N - n\_70;

y\_val = y\_1\_no\_media(1:n\_30);

u\_val = pw\_1\_no\_media(1:n\_30);

data\_val = iddata(y\_val , u\_val , Ts);

%definizione ordini AR MA X e K

n\_ar = 4;

n\_x = 3;

n\_ma = 1;

k = 2;

%Sima del modello

modello\_armax\_1 = armax(data\_ident , [n\_ar n\_x n\_ma k])

modello\_armax\_1 =

Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)

A(z) = 1 - 3.31 z^-1 + 4.159 z^-2 - 2.359 z^-3 + 0.5113 z^-4

B(z) = 0.01985 z^-2 + 0.009946 z^-3 - 0.02851 z^-4

C(z) = 1 + 0.8559 z^-1

Sample time: 0.05 seconds

Parameterization:

Polynomial orders: na=4 nb=3 nc=1 nk=2

Number of free coefficients: 8

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "data\_ident".

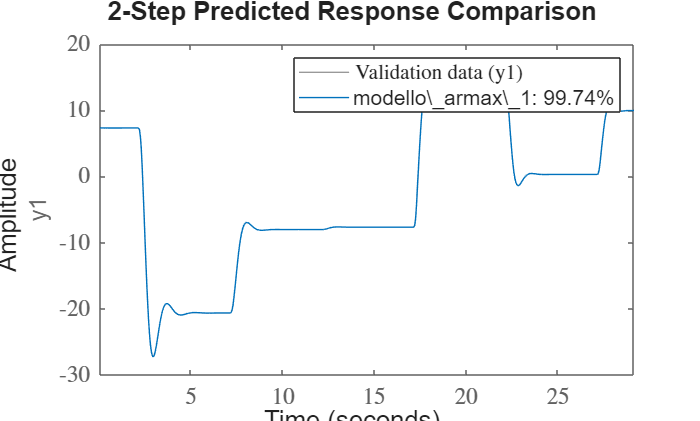
Fit to estimation data: 99.94% (prediction focus)

FPE: 5.431e-05, MSE: 5.336e-05

Model Properties

figure

compare(data\_val, modello\_armax\_1 , k);



% Validazione con gli altri set

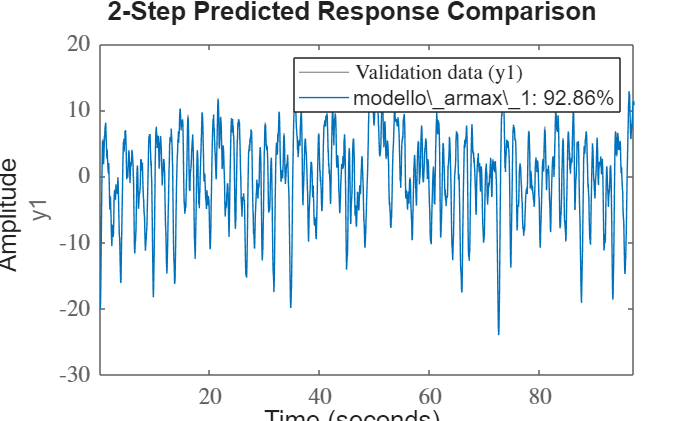
y\_val = y\_2\_no\_media;

u\_val = pw\_2\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

figure

compare(data\_val, modello\_armax\_1 , k);

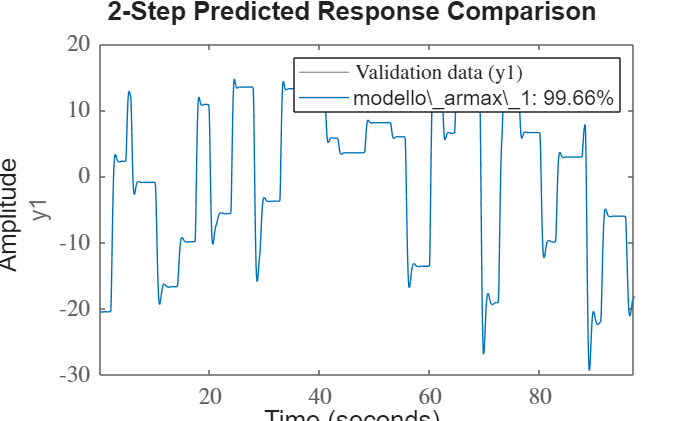


y\_val = y\_3\_no\_media;

u\_val = pw\_3\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

compare(data\_val, modello\_armax\_1 , k);



# Si utilizza il segnale PRBS

% 70% dei dati

y\_modello = y\_2(1:n\_70);

u\_modello = pw\_2(1:n\_70);

data\_ident = iddata(y\_modello , u\_modello , Ts);

% Validazione con il 30% del secondo set

n\_30 = N - n\_70;

y\_val = y\_2\_no\_media(1:n\_30);

u\_val = pw\_2\_no\_media(1:n\_30);

data\_val = iddata(y\_val , u\_val , Ts);

%definizione ordini AR MA X e K

% n\_ar = 5;

% n\_x = 3;

% n\_ma = 1;

% k = 2;

%Stima del modello

modello\_armax\_2 = armax(data\_ident , [n\_ar n\_x n\_ma k])

modello\_armax\_2 =

Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)

A(z) = 1 - 2.699 z^-1 + 2.877 z^-2 - 1.487 z^-3 + 0.3371 z^-4

B(z) = 0.01928 z^-2 + 0.02255 z^-3 - 0.01093 z^-4

C(z) = 1 + 0.5572 z^-1

Sample time: 0.05 seconds

Parameterization:

Polynomial orders: na=4 nb=3 nc=1 nk=2

Number of free coefficients: 8

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "data\_ident".

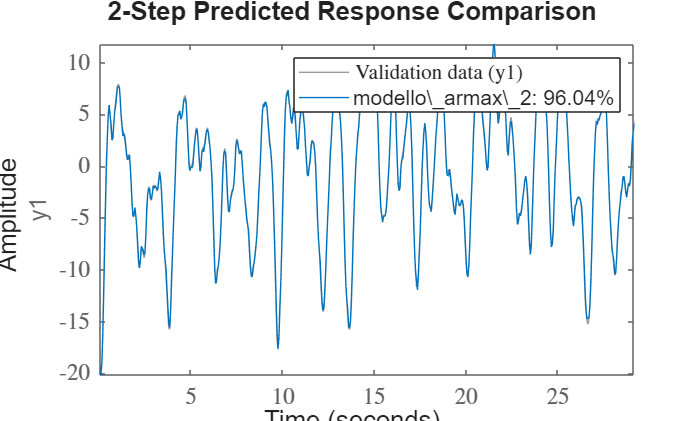
Fit to estimation data: 98.86% (prediction focus)

FPE: 0.005429, MSE: 0.005334

Model Properties

figure

compare(data\_val, modello\_armax\_2 , k);



% Validazione con gli altri set

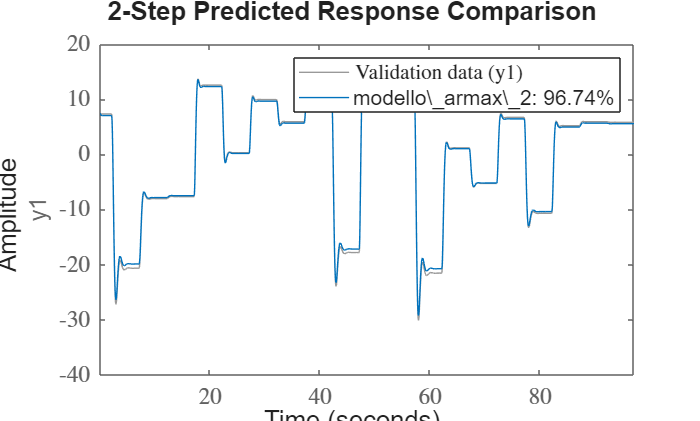
y\_val = y\_1\_no\_media;

u\_val = pw\_1\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

figure

compare(data\_val, modello\_armax\_2 , k)



% Validazione con il terzo set

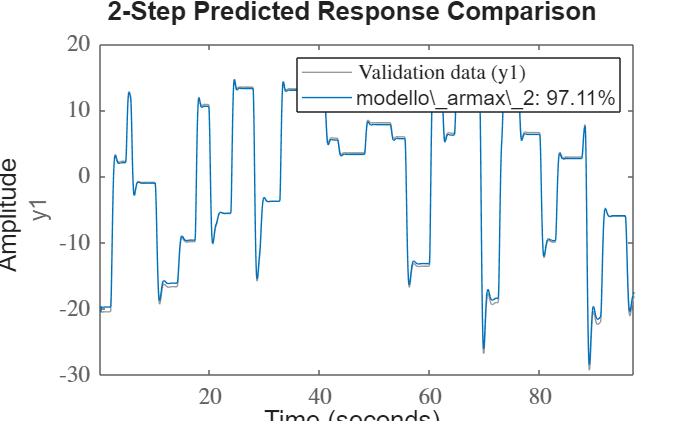
y\_val = y\_3\_no\_media;

u\_val = pw\_3\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

figure

compare(data\_val, modello\_armax\_2 , k)



# Si utilizza il segnale randomico a blocchi variabili

% 70% dei dati

n\_70 = floor(N\*0.7);

y\_modello = y\_3\_no\_media(1:n\_70);

u\_modello = pw\_3\_no\_media(1:n\_70);

data\_ident = iddata(y\_modello , u\_modello , Ts);

% Validazione con il 30%

n\_30 = N - n\_70;

y\_val = y\_3\_no\_media(1:n\_30);

u\_val = pw\_3\_no\_media(1:n\_30);

data\_val = iddata(y\_val , u\_val , Ts);

%definizione ordini AR MA X e K

% n\_ar = 5;

% n\_ma = 1;

% n\_x = 3;

% k = 2;

%Sima del modello

modello\_armax\_3 = armax(data\_ident , [n\_ar n\_x n\_ma k])

modello\_armax\_3 =

Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)

A(z) = 1 - 3.248 z^-1 + 4.015 z^-2 - 2.253 z^-3 + 0.4866 z^-4

B(z) = 0.01999 z^-2 + 0.01016 z^-3 - 0.02898 z^-4

C(z) = 1 + 0.7998 z^-1

Sample time: 0.05 seconds

Parameterization:

Polynomial orders: na=4 nb=3 nc=1 nk=2

Number of free coefficients: 8

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "data\_ident".

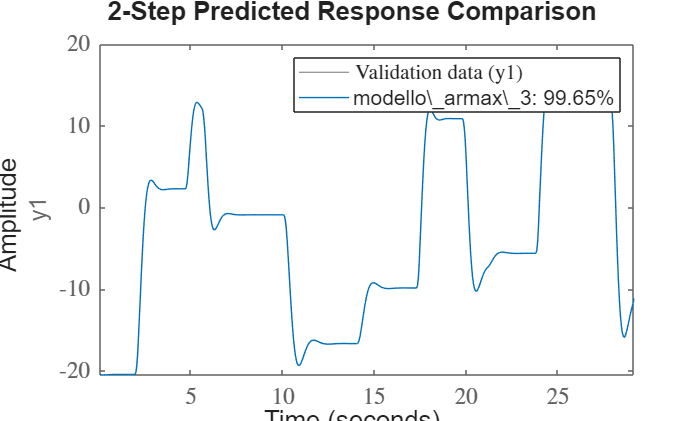
Fit to estimation data: 99.92% (prediction focus)

FPE: 6.339e-05, MSE: 6.228e-05

Model Properties

figure

compare(data\_val, modello\_armax\_3 , k)



% Validazione con il primo set

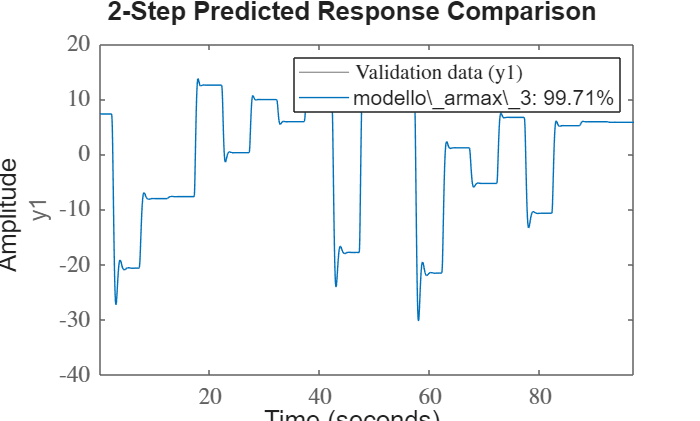
y\_val = y\_1\_no\_media;

u\_val = pw\_1\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

figure

compare(data\_val, modello\_armax\_3, k)



% Validazione con il secondo set

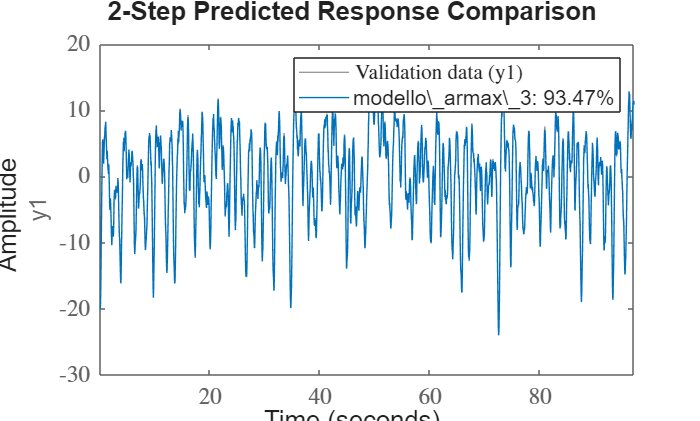
y\_val = y\_2\_no\_media;

u\_val = pw\_2\_no\_media;

data\_val = iddata(y\_val , u\_val , Ts);

figure

compare(data\_val, modello\_armax\_3 , k)



Di seguito si riportano i determinanti di  con  il grado di persistenza eccitazione per i tre diversi segnali:

1.Randomico a blocchi costanti:

gamma\_uu = autocorr(pw\_1\_no\_media, "NumLags", n\_x);

for i = 1 : n\_x+1

R\_z = toeplitz(gamma\_uu(1:i));

det\_R\_pw\_1(i) = det(R\_z);

end

det\_R\_pw\_1

det\_R\_pw\_1 = *1×4*

1.0000 0.0244 0.0006 0.0000

2. PRBS

gamma\_uu = autocorr(pw\_2\_no\_media, "NumLags", n\_x);

for i = 1 : n\_x+1

R\_z = toeplitz(gamma\_uu(1:i));

det\_R\_pw\_2(i) = det(R\_z);

end

display(det\_R\_pw\_2);

det\_R\_pw\_2 = *1×4*

1.0000 0.9998 0.9997 0.9995

3. Randomico a blocchi variabili:

gamma\_uu = autocorr(pw\_3\_no\_media, "NumLags", n\_x);

for i = 1 : n\_x+1

R\_z = toeplitz(gamma\_uu(1:i));

det\_R\_pw\_3(i) = det(R\_z);

end

display(det\_R\_pw\_3);

det\_R\_pw\_3 = *1×4*

1.0000 0.0345 0.0012 0.0000

Controllo a Minima Varianza

# Verifica Preliminare sulle radici di

modello = modello\_armax\_2;

modello

modello =

Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)

A(z) = 1 - 2.699 z^-1 + 2.877 z^-2 - 1.487 z^-3 + 0.3371 z^-4

B(z) = 0.01928 z^-2 + 0.02255 z^-3 - 0.01093 z^-4

C(z) = 1 + 0.5572 z^-1

Sample time: 0.05 seconds

Parameterization:

Polynomial orders: na=4 nb=3 nc=1 nk=2

Number of free coefficients: 8

Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:

Estimated using ARMAX on time domain data "data\_ident".

Fit to estimation data: 98.86% (prediction focus)

FPE: 0.005429, MSE: 0.005334

Model Properties

disp("Valore assoluto delle radici di B(z)")

Valore assoluto delle radici di B(z)

abs(roots(modello.B))

ans = *2×1*

1.5382

0.3686

disp("Valore assoluto delle radici di A(z)")

Valore assoluto delle radici di A(z)

abs(roots(modello.A))

ans = *4×1*

0.9141

0.9141

0.6351

0.6351

# Lunga Divisione

### 

C\_z = NumeratoreTF(modello.C , Ts)

C\_z =

1 + 0.5572 z^-1

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

A\_z = NumeratoreTF(modello.A , Ts)

A\_z =

1 - 2.699 z^-1 + 2.877 z^-2 - 1.487 z^-3 + 0.3371 z^-4

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

B\_z = NumeratoreTF(modello.B , Ts)

B\_z =

0.01928 + 0.02255 z^-1 - 0.01093 z^-2

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

## Eseguiamo la lunga divisione

[E\_array , R\_array] = LungaDivisione(cell2mat(C\_z.Numerator), cell2mat(A\_z.Numerator) , k);

E\_z = NumeratoreTF(E\_array , Ts)

E\_z =

1 + 3.257 z^-1

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

R\_z = NumeratoreTF(R\_array ,Ts)

R\_z =

5.914 - 7.882 z^-1 + 4.506 z^-2 - 1.098 z^-3

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

z2 = RitardoPuro(2 , Ts);

z1 = RitardoPuro(1 , Ts);

b=100;

### 

G\_z = E\_z \* B\_z + b\*(1 - z1) \* C\_z

G\_z =

100 - 44.2 z^-1 - 55.66 z^-2 - 0.03559 z^-3

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

H\_z = C\_z

H\_z =

1 + 0.5572 z^-1

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

L\_z = (1/G\_z) \* (B\_z/A\_z)\*z2 \* R\_z

L\_z =

0.114 z^-2 - 0.0186 z^-3 - 0.1555 z^-4 + 0.1666 z^-5 - 0.07399 z^-6 + 0.012 z^-7

----------------------------------------------------------------------------------------------

100 - 314.2 z^-1 + 351.4 z^-2 - 125.7 z^-3 - 60.59 z^-4 + 67.76 z^-5 - 18.71 z^-6 - 0.012 z^-7

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

phi\_z = cell2mat(L\_z.Numerator) + cell2mat(L\_z.Denominator);

### Radici di

abs(roots(phi\_z))'

ans = *1×7*

0.5572 0.9887 0.9195 0.9195 0.6350 0.6350 0.0000

Fs\_z = ((1/G\_z) \* (B\_z/A\_z)\*z2)/(1+L\_z)

Fs\_z =

1.928 z^-2 - 3.801 z^-3 - 1.403 z^-4 + 8.934 z^-5 - 7.842 z^-6 + 1.314 z^-7 + 1.829 z^-8 - 1.163 z^-9 + 0.2042 z^-10 + 0.0001311 z^-11

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1e04 - 6.285e04 z^-1 + 1.69e05 z^-2 - 2.46e05 z^-3 + 1.904e05 z^-4 - 3.667e04 z^-5 - 7.323e04 z^-6 + 7.472e04 z^-7 - 2.655e04 z^-8 - 3523 z^-9 + 6879 z^-10 - 2543 z^-11 + 350.6 z^-12 + 0.2253 z^-13 + 2.081e-20 z^-14

Sample time: 0.05 seconds

Discrete-time transfer function.

Model Properties

### Guadagno della

dcgain(Fs\_z)

ans = 0.6422

# Simulazione

### Risposta allo scalino / Sinusoide

load\_system("controllo\_simulink")

%Segnale di ingresso

valore\_medio = 135;%[deg]

ampiezza\_sinusoide = 0;

tempo\_simulazione =100 ;%[secondi]

delay = 15; %[secondi]

T =10; % Periodo di oscillazione [s]

n\_tempo\_simulazione = floor(tempo\_simulazione/Ts);

n\_delay = floor(delay/Ts);

y\_0 = [125\*ones(1 , n\_delay) , valore\_medio \* ones(1 , n\_tempo\_simulazione-n\_delay)];

time = 0:Ts:tempo\_simulazione-Ts;

w = 2\*pi/T;

y\_0 = [120\*ones(1 , n\_delay) , ampiezza\_sinusoide \* sin(w \* time(1:n\_tempo\_simulazione-n\_delay)) + valore\_medio];

out = sim("controllo\_simulink.slx");

i = 1

figure

plot(time , y\_0 , displayName = "Riferimento");

hold on

plot(time , out.y1(1:n\_tempo\_simulazione) , displayName = "Posizione");

xlim([0 , tempo\_simulazione]);

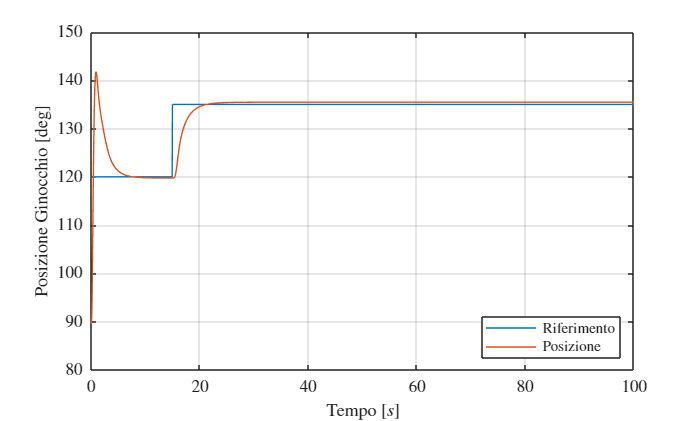
legend(location= "southeast");

ylabel("Posizione Ginocchio [deg]" , interpreter = "latex");

xlabel("Tempo $[s]$" , interpreter = "latex");

grid on;

hold off



### Rumore Bianco

val=[125 145] ;

min\_val = val(1);

max\_val = val(2);

y\_0 = [120\*ones(1 , n\_delay) , min\_val + (max\_val - min\_val) \* rand(1 , length(1:n\_tempo\_simulazione-n\_delay))];

out = sim("controllo\_simulink.slx");

i = 1

figure

plot(time , y\_0 , displayName = "Riferimento");

hold on

plot(time , out.y1(1:n\_tempo\_simulazione) , displayName = "Posizione");

xlim([0 , tempo\_simulazione]);

legend(location= "southeast");

ylabel("Posizione Ginocchio [deg]" , interpreter = "latex");

xlabel("Tempo $[s]$" , interpreter = "latex");

grid on;

hold off

