Laboratorio 3 per fondamenti di automatica - ing. fisica

Esercizio 1: Legge di controllo e ricostruzione dello stato; progetto del regolatore

Innanzitutto occorre definire il sistema in spazio di stato nel workspace di matlab (vedi esercitazione 1); pertanto i comandi da digitare nel prompt sono i seguenti:

```
A=[2 1 -1; -3 -2 -1; -1 1 -4];
b=[1 0 1]';
c=[0 0 1];
d=0;                          % definizione matrici del sistema
```

a) Valutando gli autovalori della matrice A

```
autoval=eig(A)
autoval =
    1.5616
   -2.5616
   -3.0000
```

si può notare che hanno modulo superiore a 1 così che il sistema risulta instabile.

b) Affinché il sistema sia stabilizzabile mediante un regolatore lineare composto da legge di controllo *k* e ricostruttore dello stato *l*, occorre che questo sia completamente raggiungibile e completamente osservabile. I comandi matlab `ctrb()` e `obsv()` restituiscono, rispettivamente, le matrici di raggiungibilità e di osservabilità.

```
R=ctrb(A,b);                          % matrice di raggiungibilità
det_R=det(R)
det_R =
    12                                % completa raggiungibilità
O=obsv(A,c);                          % matrice di osservabilità
det_O=det(O)
det_O =
    8                                 % completa osservabilità
```

Poiché le matrici di raggiungibilità e di osservabilità hanno determinante non nullo, il sistema è, rispettivamente, completamente raggiungibile e osservabile. Pertanto, il sistema è stabilizzabile mediante un regolatore lineare e la dinamica del sistema regolato è fissabile ad arbitrio. In particolare, sarà possibile determinare una legge di controllo *k* e un ricostruttore dello stato *l* tale che tutti i transitori di annullino in tempo finito.

c) Progetto della legge di controllo *k*.
Per progettare una retroazione dello stato che assegni gli autovalori come chiesti (tutti nulli) si può utilizzare la formula di Ackermann e il comando matlab `acker()`. (digitare `help acker` per una discussione dettagliata sull'uso del comando; in particolare, il comando fa riferimento a una legge di controllo $u = -kx$ così che, nell'uso del comando, il vettore *b* andrà sostituito con il vettore $-b$).

```
autovalori=[0 0 0];
```

```
k = acker(A,-b,autovalori)
k =
    2.5000    3.0000    1.5000
eig(A + b*k)                    % verifica autovalori
```

Progetto del ricostruttore dello stato *l*.

In modo analogo si può ottenere il ricostruttore dello stato; si noti però che la formula di Ackermann permette di posizionare i poli del sistema in anello chiuso ($A+bk$), mentre la dinamica dell'osservatore è descritta dalla matrice ($A+lc$) (si noti la posizione di *l*, duale rispetto a quella di *k* – il prodotto matriciale non gode della proprietà commutativa).

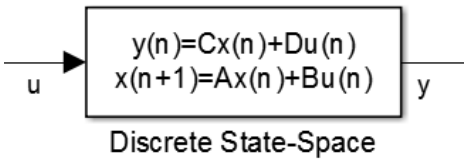
Pertanto, per risolvere il problema utilizzando la formula di Ackermann, basta osservare che $(A+lc)^T$ ha gli stessi autovalori di ($A+lc$). Ma $(A+lc)^T = A^T + c^T l^T$

Pertanto basta applicare la formula di Ackermann alla coppia (A^T, $-c^T$) per ottenere l^T.

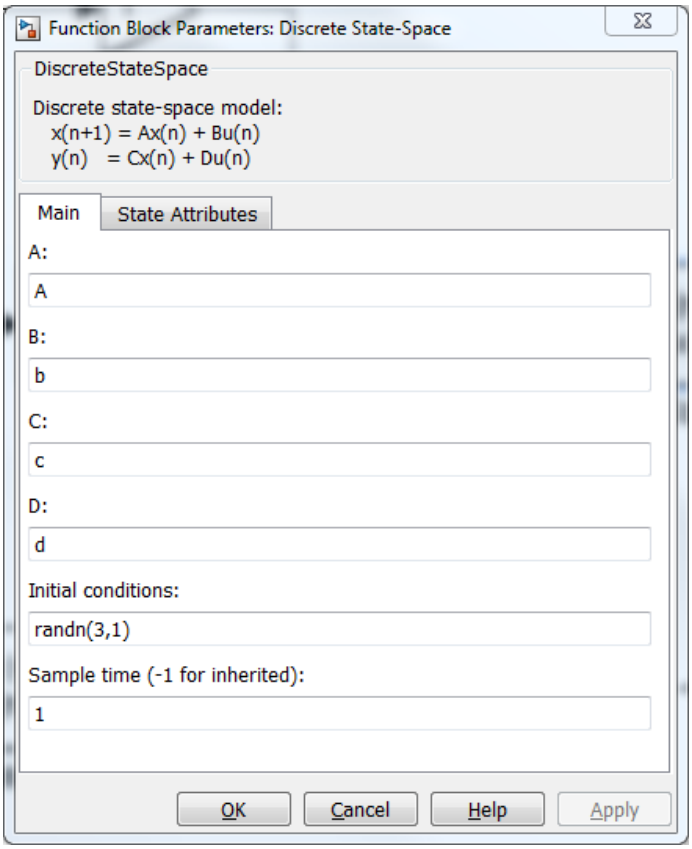
```
lT = acker(A',-c',autovalori);
l = lT'
l = 1.3750
    0.3750
    4.0000

eig(A + l*c)                    % verifica autovalori
```

d) Implementazione in simulink

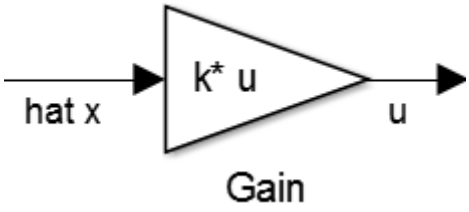
Il sistema può essere definiti tramite Simulink utilizzando il blocco Discrete State-Space che riceve in ingresso *u*, ingresso del sistema, e da cui esce *y*, uscita del sistema.

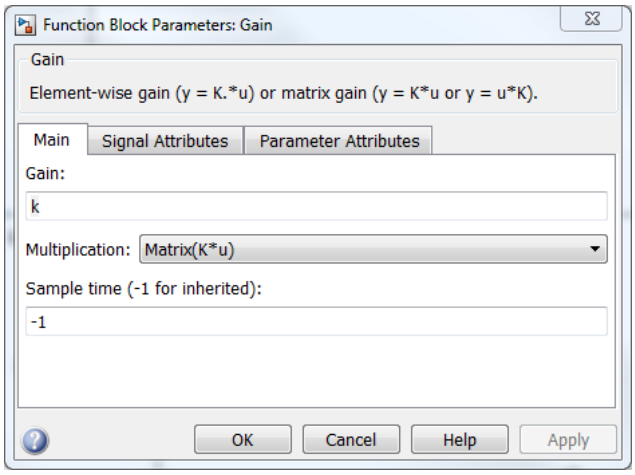


in cui la condizione iniziale è impostata come casuale (siccome noi non la conosciamo).

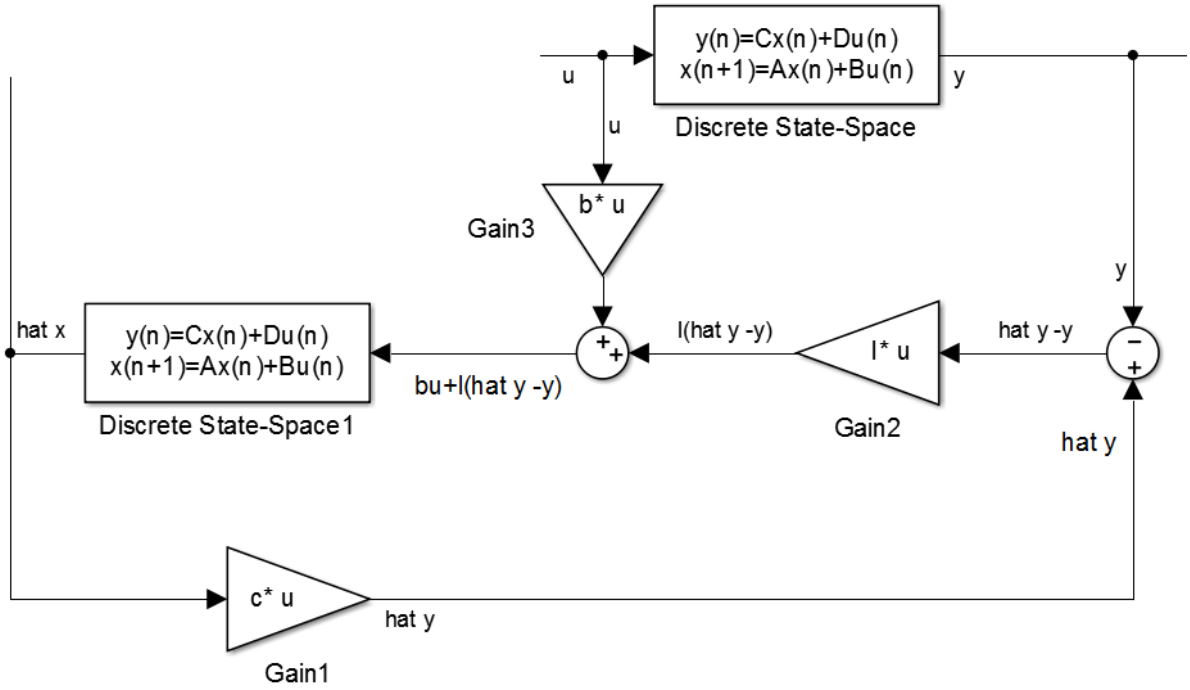


Il controllore viene implementato tramite un blocco di tipo guadagno, che riceve in ingresso lo stato ricostruito $\hat{x}$ e ha come uscita l'ingresso u che vogliamo dare al sistema (si noti che il prodotto del blocco è impostato come prodotto matriciale).





Per l'implementazione del ricostruttore dobbiamo fare una copia del sistema e cambiare gli ingressi e le uscite. Esso avrà come uscita lo stato ricostruito (quindi la C del sistema sarà la matrice identità, in modo che $y = C\hat{x} = \hat{x}$, e come ingresso $bu + l(\hat{y} - y)$. Per far questo quindi possiamo costruire il segnale di ingresso all'esterno del blocco, e mettere come matrice di ingresso del nostro sistema ancora l'identità.



Unendo il blocco del controllore a quello del ricostruttore abbiamo progettato un sistema di controllo (senza ingressi) che porta il sistema a 0 in tempo finito. Volendolo portare ad un determinato valore di riferimento y^0 ci ricordiamo che nel progetto del regolatore imponiamo l'ingresso $u = k\hat{x} + v$: possiamo utilizzare la v quindi per modulare l'uscita.

L'intero sistema regolato può essere scritto come

$$\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & b\,k \\ -l\,c & A + bk + lc \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} b \\ b \end{bmatrix} v$$

$$y = \begin{bmatrix} c & 0 \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + d\,v$$

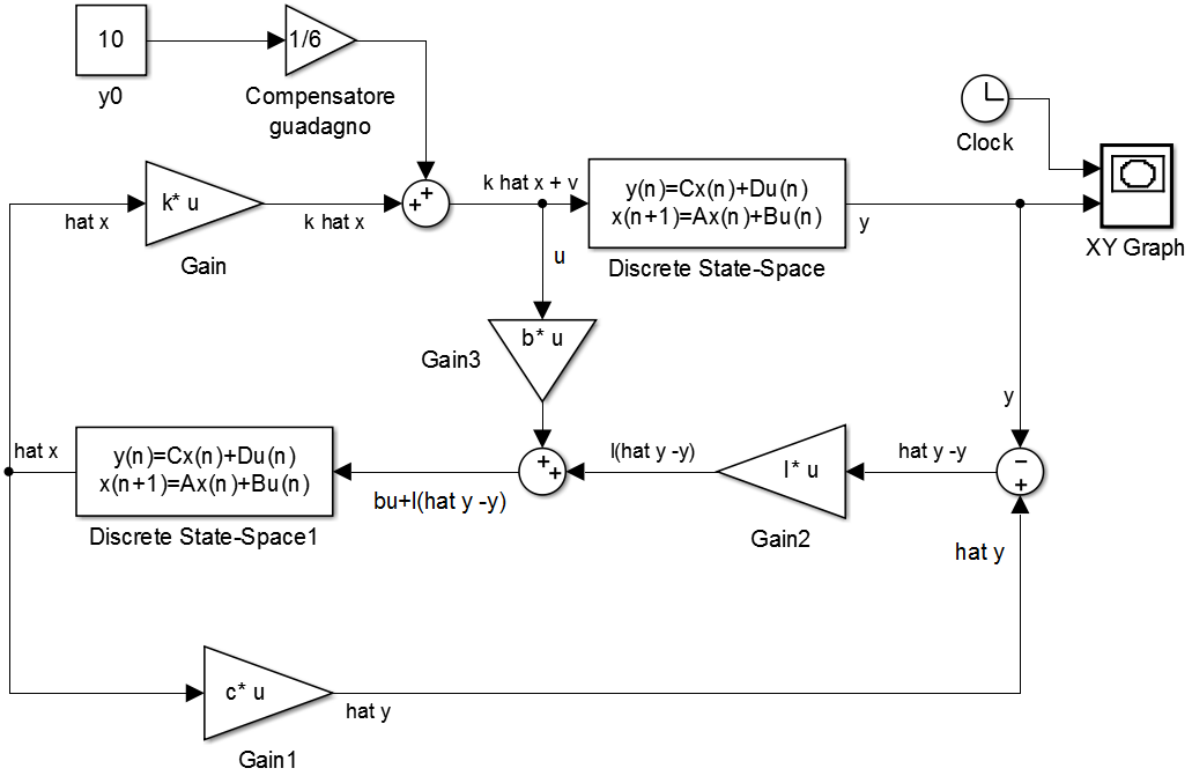
Volendo che il sistema converga a $y = y^0$ è necessario che il guadagno del sistema tra i due segnali sia unitario.
Il guagadno del sistema regolato è

$$G = [c \quad 0]\left(\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}A & b\,k\\ -l\,c & A + bk + lc\end{bmatrix}\right)^{-1}\begin{bmatrix}b\\ b\end{bmatrix} + d$$

e possiamo calcolarlo tramite i comandi Matlab

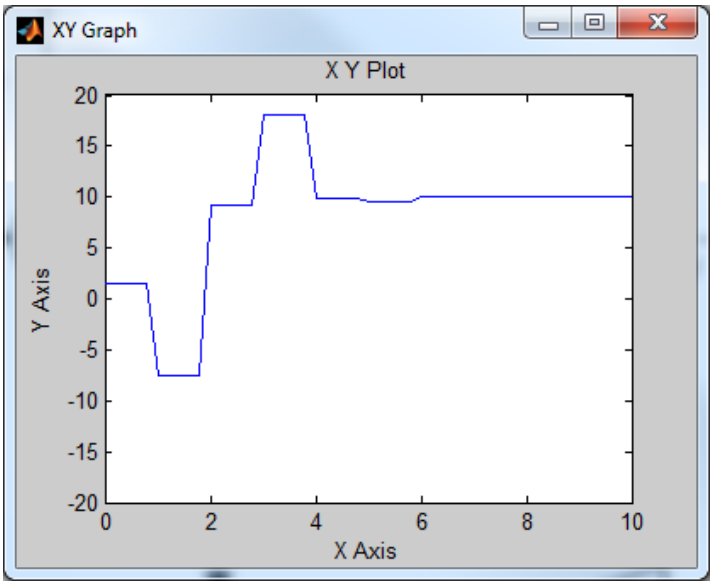
```
matriciona= [A, b*k; -l*c, A+l*c+b*k];
guadagnochiuso=[c,zeros(1,3)]*inv(eye(6)-matriciona)*[b;b]

guadagnochiuso =
   6
```

Compensando quindi il guadagno in anello chiuso otteniamo quanto richiesto



Il risutlato della simulazione sarà quindi (ad esempio, poichè la condizione iniziale è casuale) quello riportato in figura.

Si noti che il sistema converge al valore desiderato in 6 (ovvero 2 N) costanti di tempo.



Esercizio 2: Legge di controllo e ricostruzione dello stato; progetto del regolatore

1. Il sistema in forma di stato, omettendo la dipendenza delle variabili dal tempo, può essere riscritto nel seguente modo:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{m}{M} g x_3 + \frac{1}{M} u_1 - \frac{1}{lM} u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = \frac{g}{l}\frac{M+m}{M} x_3 - \frac{1}{lM} u_1 + \frac{1}{l^2}\frac{M+m}{Mm} u_2 \end{cases}$$

   Pertanto, le matrici del sistema in forma di stato sono le seguenti:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{M}{m} g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g}{l}\frac{M+m}{M} & 0 \end{bmatrix}; B = \begin{bmatrix} 0 & 0 \\ \frac{1}{M} & -\frac{1}{lM} \\ 0 & 0 \\ -\frac{1}{lM} & \frac{1}{l^2}\frac{M+m}{Mm} \end{bmatrix}; C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$$

2. Per verificare che il sistema sia raggiungibile e osservabile con gli ingressi e le uscite scelte occorre mostrare che le matrici di raggiungibilità R e di osservabilità O abbiano rango pieno (ovvero determinante non nullo); per fare ciò occorre innanzitutto selezionare le matrici del sistema in modo corretto; si noti infatti che l'ingresso "forza" corrisponde alla prima colonna della matrice B, mentre l'ingresso "coppia" corrisponde alla seconda colonna della stessa. Analogamente, l'uscita "posizione" corrisponde alla prima riga della matrice C, mentre l'uscita "posizione angolare" corrisponde alla seconda riga della stessa; pertanto la richiesta del problema può essere risolta nel modo seguente:

```
% - Parametri
M = 10;
m = 1;
l = 1;
g = 9.81;

% - Scrittura delle equazioni di moto in forma matriciale
A = [0 1 0 0;0 0 -m/M*g 0;0 0 0 1;0 0 g/l*(m+M)/M 0];
B = [0 0 ;1/M -1/(l*M);0 0;-1/(l*M) 1/(l^2)*(M+m)/(m*M)];
C = [1 0 0 0;0 0 1 0];
D = [0 0;0 0];

% - Osservabilità e raggiungibilità

% Prima coppia: forza - posizione
disp('Controllabilità forza')
co = ctrb(A,B(:,1))              % selezione ingresso forza
det(co)
rank(co)
disp('Osservabilità posizione')
ob = obsv(A,C(1,:))              % selezione misura posizione
det(ob)
```

```
rank(ob)

% Seconda coppia: coppia - posizione angolare
disp('Controllabilità coppia')
co = ctrb(A,B(:,2))              % selezione ingresso coppia
det(co)
rank(co)
disp('Osservabilità posizione angolare')
ob = obsv(A,C(2,:))              % selezione misura angolo
det(ob)
rank(ob)


Controllabilità forza

co =

         0    0.1000         0    0.0981
    0.1000         0    0.0981         0
         0   -0.1000         0   -1.0791
   -0.1000         0   -1.0791         0


ans =

    0.0096


ans =

     4

Osservabilità posizione

ob =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0   -0.9810         0
         0         0         0   -0.9810


ans =

    0.9624


ans =

     4

Controllabilità coppia

co =

         0   -0.1000         0   -1.0791
   -0.1000         0   -1.0791         0
```

```
         0    1.1000         0   11.8701
    1.1000         0   11.8701         0


ans =

     0


ans =

     2

Osservabilità posizione angolare

ob =

         0         0    1.0000         0
         0         0         0    1.0000
         0         0   10.7910         0
         0         0         0   10.7910


ans =

     0


ans =

     2
```

Il comando `rank()`, restituisce il valore del rango della matrice.
Pertanto si è dimostrato quanto chiesto dal testo al punto 2.

3. Per progettare una retroazione dello stato (legge di controllo *k*) che assegni gli autovalori come chiesti si utilizza la formula di Ackermann e il comando matlab `acker()` (vedi esercizio 1). Pertanto si procede come segue:

```
% - Assegnamento degli autovalori
% Ricavo le radici del polinomio
disp('Autovalori desiderati')
autovalori = [roots([1 1.5 1]);roots([1 2 1])]

Autovalori desiderati

autovalori =

  -0.7500 + 0.6614i
  -0.7500 - 0.6614i
  -1.0000
  -1.0000

% - Progetto la legge di controllo
disp('Legge di controllo k')
k = acker(A,-B(:,1),autovalori)
```

```
Legge di controllo k

k =

   1.0194   3.5678   158.9294   38.5678

disp('Verifica autovalori')
eig(A + B(:,1)*k)
autovalori

Verifica autovalori

ans =

  -0.7500 + 0.6614i
  -0.7500 - 0.6614i
  -1.0000 + 0.0000i
  -1.0000 - 0.0000i


autovalori =

  -0.7500 + 0.6614i
  -0.7500 - 0.6614i
  -1.0000
  -1.0000
```

4. In modo analogo si può ottenere il ricostruttore dello stato (vedi esercizio 1 per l'uso del comando `acker()`);

```
% - Progetto ricostruttore di stato
% Ricavo le radici del polinomio
disp('Autovalori desiderati')
autov_ricostruttore = [roots([1 15 100]);roots([1 20 100])]

Autovalori desiderati

autov_ricostruttore =

  -7.5000 + 6.6144i
  -7.5000 - 6.6144i
 -10.0000
 -10.0000

% - Ricostruttore dello stato
disp('Ricostruttore dello stato l')
l = acker(A',-C(1,:).',autov_ricostruttore)'

Ricostruttore dello stato l

l =

  1.0e+004 *

    -0.0035
```

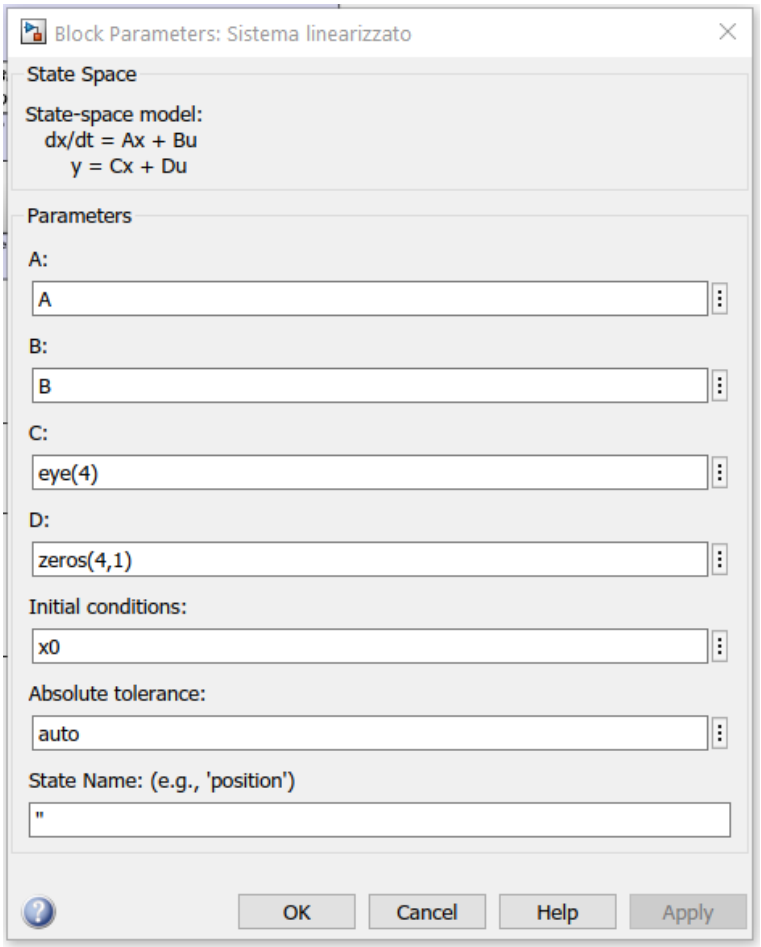
```
    -0.0511
     0.3953
     1.5812

disp('Verifica autovalori')
eig(A + l*C(1,:))
autov_ricostruttore

Verifica autovalori

ans =

  -7.5000 + 6.6144i
  -7.5000 - 6.6144i
 -10.0000 + 0.0000i
 -10.0000 - 0.0000i

autov_ricostruttore =

  -7.5000 + 6.6144i
  -7.5000 - 6.6144i
 -10.0000
 -10.0000
```

5. Iniziamo a considerare il caso più semplice, in cui è possibile accedere allo stato del sistema. In questo caso ovviamente non occorre ricostruire lo stato, ma è necessario implementare solamente il controllo.

Innanzitutto, per poter retroazionare il sistema bisogna definire in simulink il sistema stesso; pertanto possiamo utilizzare il blocco State-Space ricordando che il sistema è a tempo continuo; come uscita di tale blocco ci occorre proprio lo stato (a cui supponiamo di poter accedere). Utilizzando il file esercizio2sim_vuoto.slx, è possibile copiare il blocco relativo al sistema linearizzato (e anche quello al sistema non lineare) già implementati dentro il sotto sistema “Esportazione dati”.

Attenzione: tutti i blocchi presenti in questo file simulink non devono essere modificati!!!

Le proprietà del blocco relativo al sistema linearizzato saranno le seguenti:

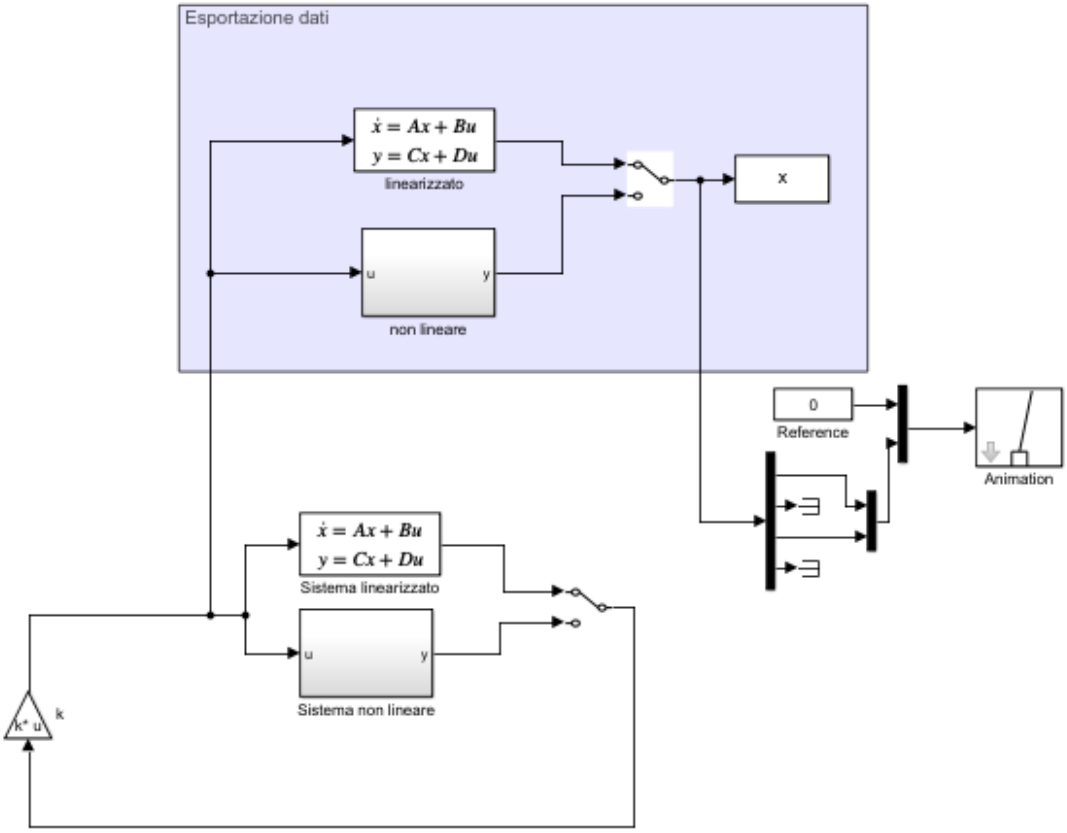


Questo perché le matrici *A* e *B* descrivono la dinamica e gli ingressi del blocco, mentre le matrici *C* e *D* descrivono l'uscita dello stesso; se si vogliono ottenere le informazioni su tutto lo stato occorre che la matrice *C* sia una matrice identità e che il vettore *D* sia nullo.

Infatti si ha:

$$Y = Cx + Du = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_1 = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Per quanto riguarda il blocco relativo al sistema non lineare, controllare al suo interno che il blocco gain abbia un guadagno pari a `eye(4)`. Il motivo di questa scelta è analogo a quello già spiegato per il sistema linearizzato.
La legge di controllo stabilizzante non è altro (si veda al punto 4) che una retroazione dello stato, agente sul primo dei due ingressi del sistema (la forza impressa sul carrello).
Dunque, per realizzare lo schema, basta moltiplicare lo stato con un blocco gain di simulink che abbia come parametri il vettore di guadagni trovato al punto 4.
Lo schema del sistema con controllo in retroazione risulta essere il seguente:



Il sotto sistema dedicato all'esportazione dei dati permette di salvare l'andamento delle quattro variabili di stato del sistema linearizzato (se il blocco switch è posizionato come in figura) o di quello non lineare (fare doppio click sul blocco switch passare dal sistema linearizzato a quello non lineare, e viceversa). Il blocco Animation è utile in quanto permette di avere una semplice animazione della dinamica simulata.
Si noti che sono presenti dei blocchi terminator: tali blocchi servono solo per evitare che simulink dia un messaggio di warning dovuto a un arco non connesso. Sono blocchi non necessari, che però rendono più ordinato lo schema e velocizzano la simulazione (nel caso di sistemi più complessi limitare i messaggi di warning risulta critico).

Procediamo ora a simulare entrambi i sistemi (linearizzato e non lineare), per un orizzonte di 25 secondi, partendo da una condizione iniziale relativamente vicina all'equilibrio intorno a cui è stato linearizzato il sistema.

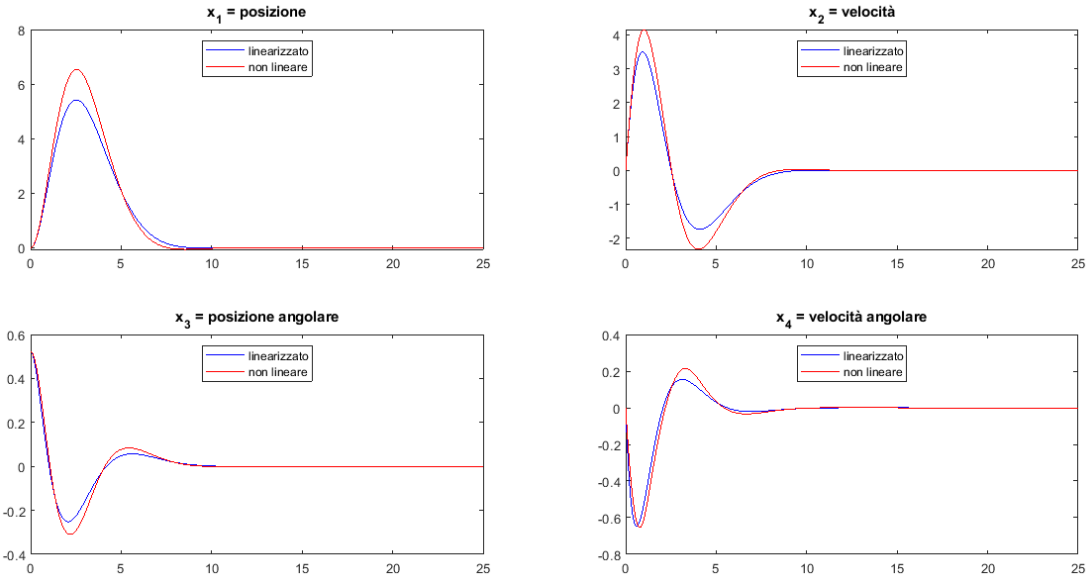
```
%% STATO INIZIALE VICINO ALL'EQUILIBRIO

x0 = [0 0 pi/6 .0]';


%% LANCIARE IL MODELLO SIMULINK LINEARIZZATO (settare orizzonte
simulazione = 25)

tempo = x.Time;
```

```
x1_lin = x.Data(:, 1);

x2_lin = x.Data(:, 2);

x3_lin = x.Data(:, 3);

x4_lin = x.Data(:, 4);
```

A questo punto, cambiare la posizione degli interruttori nei blocchi switch e simulare il sistema non lineare.

```
%% LANCIARE IL MODELLO SIMULINK NON LINEARE (settare orizzonte
simulazione = 25)

x1_nonLin = x.Data(:, 1);

x2_nonLin = x.Data(:, 2);

x3_nonLin = x.Data(:, 3);

x4_nonLin = x.Data(:, 4);


%% plot lin vs non lin

close all

figure

subplot(2, 2, 1)

plot(tempo, x1_lin, 'b')

hold on

plot(tempo, x1_nonLin, 'r')

title('x_1 = posizione')

legend('linearizzato', 'non lineare', 'location', 'north')

subplot(2, 2, 2)

plot(tempo, x2_lin, 'b')

hold on

plot(tempo, x2_nonLin, 'r')

title('x_2 = velocità')

legend('linearizzato', 'non lineare', 'location', 'north')
```

```
subplot(2, 2, 3)
plot(tempo, x3_lin, 'b')
hold on
plot(tempo, x3_nonLin, 'r')
title('x_3 = posizione angolare')
legend('linearizzato', 'non lineare', 'location', 'north')
subplot(2, 2, 4)
plot(tempo, x4_lin, 'b')
hold on
plot(tempo, x4_nonLin, 'r')
title('x_4 = velocità angolare')
legend('linearizzato', 'non lineare', 'location', 'north')
```

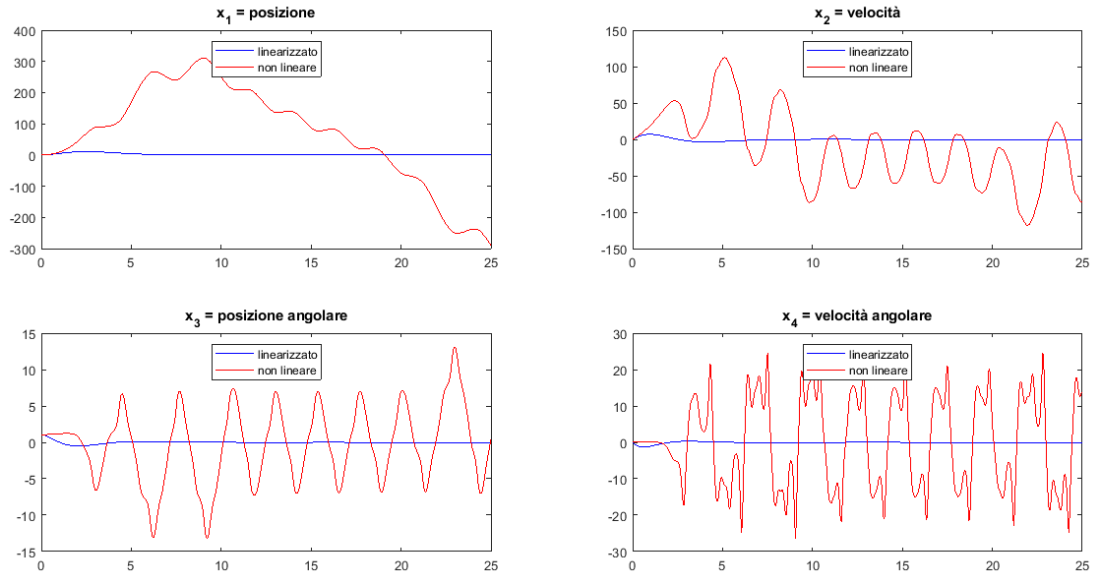


Il grafico mostra che gli andamenti delle variabili relative al sistema non lineare (quello che effettivamente dev'essere controllato) convergono all'equilibrio nullo (asta ferma in verticale) come desiderato.

Vediamo ora come cambia la situazione utilizzando uno stato iniziale più lontano dall'equilibrio.

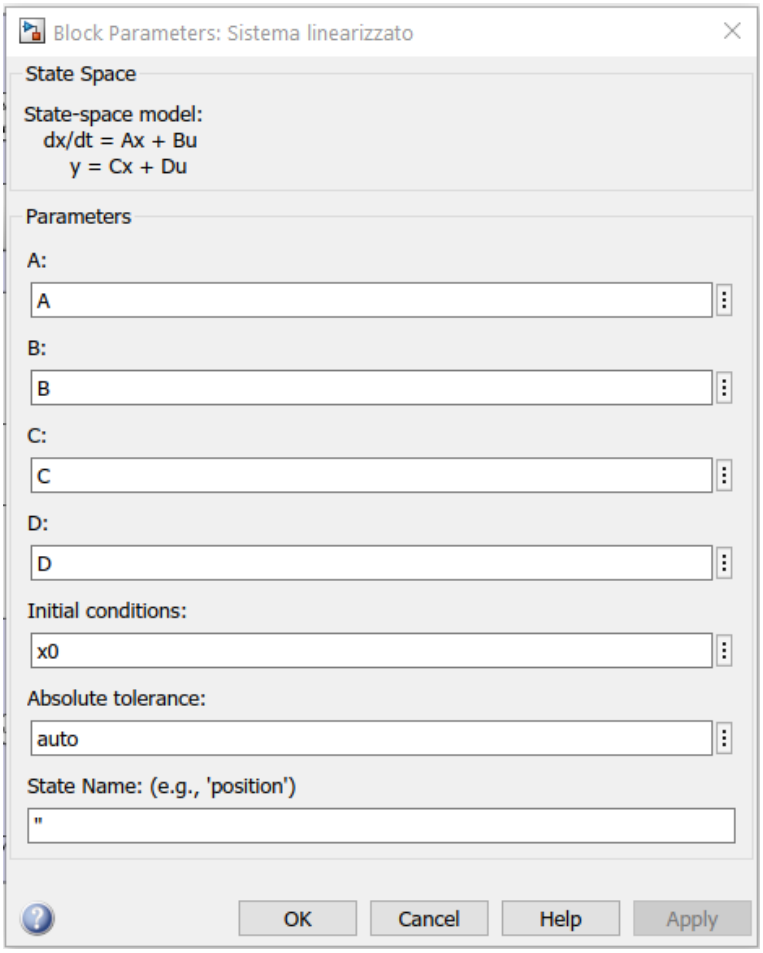
```
%% STATO INIZIALE LONTANO DALL'EQUILIBRIO
x0 = [0 0 pi/3 .0]';
```

Utilizzando nuovamente il codice matlab e lo schema simulink già visti in precedenza, si ottiengono i seguenti grafici:



In questo caso, la condizione iniziale è troppo distante dall'equilibrio e quindi non siamo in grado di stabilizzare il sistema non lineare.
Gli andamenti delle variabili del sistema linearizzato ovviamente convergono all'equilibrio desiderato. Se ciò non accadesse, avremmo commesso qualche errore nell'implementazione del sistema. Il sistema linearizzato però non esiste nella realtà, è solo un'approssimazione matematematica del sistema non lineare valida nell'intorno dell'equilibrio considerato. La bontà del controllo va dunque valutata in relazione alle performance che otteniamo sul sistema non lineare.

6. Si consideri ora lo schema del punto 5; il fatto che non si possa conoscere lo stato implica il dover ricostruire dalle misure (e quindi dall'uscita) lo stato stesso; pertanto non occorre più che la matrice C sia una matrice identità; si può dunque scrivere, nei parametri dello state space, la matrice C e la matrice D del sistema vero.



Analogamente, all'interno del sistema non lineare, nella trasformazione d'uscita (blocco gain) la matrice identità `eye(4)` va sostituita con la matrice *C*.

A questo punto, è necessario implementare in simulink lo schema a blocchi del ricostrutore dello stato. Si ricorda che le equazioni di un generico ricostruttore sono le seguenti:

$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + l(\hat{y}(t) - y(t)) \\ \hat{y}(t) = C\hat{x}(t) \end{cases}$$

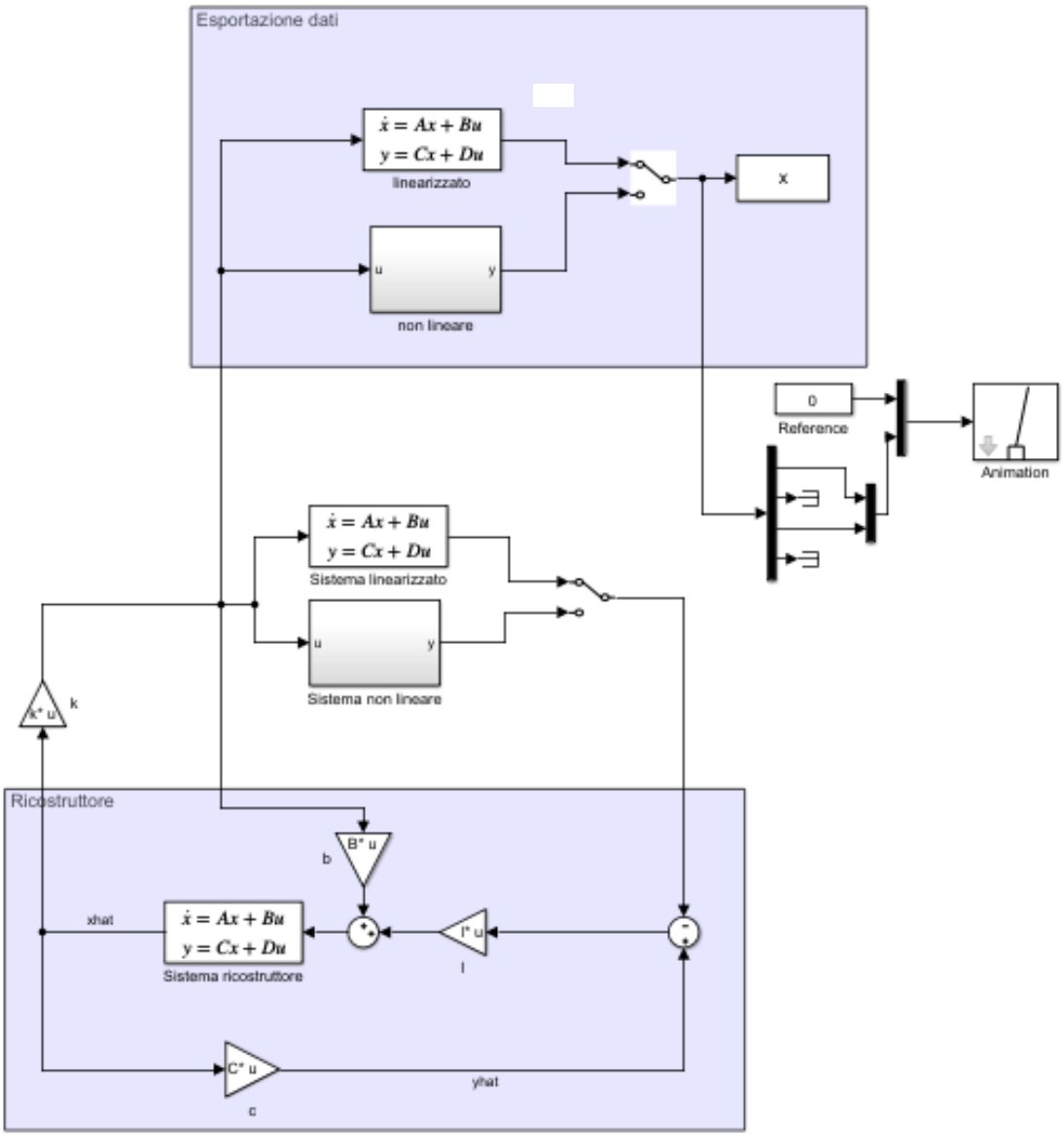
Dove *A*, *B* e *C* sono le matrici del sistema e *l* è il ricostruttore dello stato calcolato al punto 4.

Pertanto si può di nuovo utilizzare il blocco state space di simulink; si noti che ora l'ingresso a tale blocco è dato da $Bu(t) + l(\hat{y}(t) - y(t))$; per costruire questo ingresso si divida in due parti la somma:

- $Bu(t)$: per questa parte basta inserire un blocco gain con i coefficienti appropriati (quelli presenti nella matrice *B*)
- $l(\hat{y}(t) - y(t))$: per questo addendo occorre inserire un nodo sommatore che sottragga $y(t)$ alla misura della sua stima; come si nota dall'equazione del ricostruttore, la stima di $y(t)$ è l'uscita del blocco stesso. Infine l'uscita del nodo sommatore va moltiplicata per *l* (blocco gain).

Il blocco state space “Sistema ricostruttore”, avrà quindi matrice degli input (*B*) pari a `eye(4)`, e matrici degli output rispettivamente `eye(4)` (*C*) e `zeros(4)` (*D*). La condizione iniziale sarà uguale a quella vera a meno dell’errore di osservazione riportato nel testo (variabile `xhat0`).

A questo punto, si può retroazionare lo stato stimato attraverso il guadagno *k* calcolato al punto 3 per stabilizzare il sistema; lo schema complessivo sarà dunque:



Procediamo ora a simulare entrambi i sistemi (linearizzato e non lineare), per un orizzonte di 25 secondi, partendo da una condizione iniziale relativamente vicina all’equilibrio intorno a cui è stato linearizzato il sistema.

```
%% STATO INIZIALE VICINO ALL'EQUILIBRIO

x0 = [0 0 pi/6 .0]';
```

```
xhat0 = x0 + [0.01 -0.02 -0.01 0.02]';
```

Non è ovviamente necessario riscrivere tutto il codice, essendo esso analogo a quello ultilizzato nel caso del sistema controllato (senza osservatore) visto al punto 5, basta copiare e incollare o rilanciare il codice scritto in precedenza.

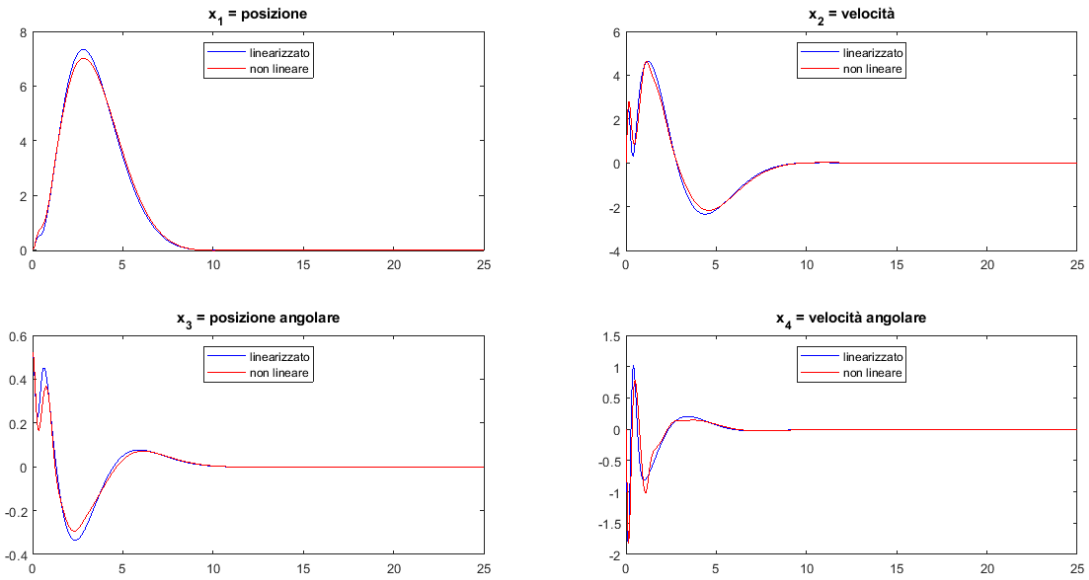
```
%% LANCIARE IL MODELLO SIMULINK LINEARIZZATO (settare orizzonte
simulazione = 25)

tempo = x.Time;

x1_lin = x.Data(:, 1);

x2_lin = x.Data(:, 2);

x3_lin = x.Data(:, 3);

x4_lin = x.Data(:, 4);
```

A questo punto, cambiare la posizione degli interruttori nei blocchi switch e simulare il sistema non lineare.

```
%% LANCIARE IL MODELLO SIMULINK NON LINEARE (settare orizzonte
simulazione = 25)

x1_nonLin = x.Data(:, 1);

x2_nonLin = x.Data(:, 2);

x3_nonLin = x.Data(:, 3);

x4_nonLin = x.Data(:, 4);


%% plot lin vs non lin

close all

figure

subplot(2, 2, 1)

plot(tempo, x1_lin, 'b')

hold on

plot(tempo, x1_nonLin, 'r')
```

```
title('x_1 = posizione')
legend('linearizzato', 'non lineare', 'location', 'north')
subplot(2, 2, 2)
plot(tempo, x2_lin, 'b')
hold on
plot(tempo, x2_nonLin, 'r')
title('x_2 = velocità')
legend('linearizzato', 'non lineare', 'location', 'north')
subplot(2, 2, 3)
plot(tempo, x3_lin, 'b')
hold on
plot(tempo, x3_nonLin, 'r')
title('x_3 = posizione angolare')
legend('linearizzato', 'non lineare', 'location', 'north')
subplot(2, 2, 4)
plot(tempo, x4_lin, 'b')
hold on
plot(tempo, x4_nonLin, 'r')
title('x_4 = velocità angolare')
legend('linearizzato', 'non lineare', 'location', 'north')
```

I risultati che si ottengono sono i seguenti:



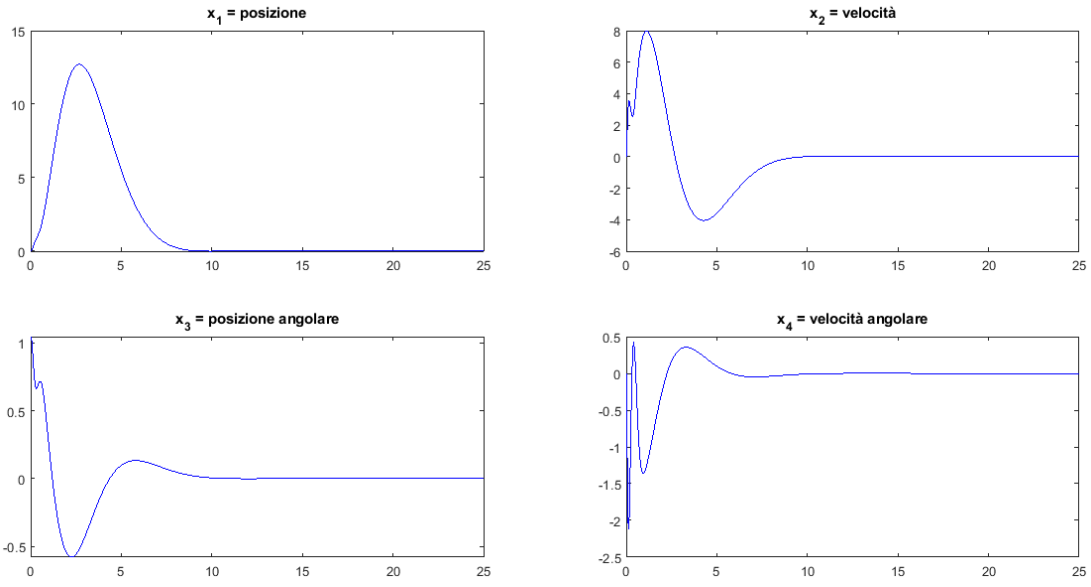
È quindi possibile stabilizzare il sistema anche quando non si ha accesso all'intero stato ma si osserva solo l'uscita del sistema (purchè la condizione di osservabilità sia rispettata).

Attenzione, un'incertezza sulla misura dello stato iniziale troppo elevata potrebbe avere lo stesso effetto di quello mostrato al punto 5 quando lo stato iniziale è lontano dall'equilibrio.

Vediamo il caso in cui la codizione iniziale è distante dall'equilibrio. In questo caso, simuliamo il sistema non lineare su un orizzonte temporale di 3 secondi, per evitare problemi numerici (mentre per il sistema linearizzato utilizziamo il solito orizzonte di 25 secondi).

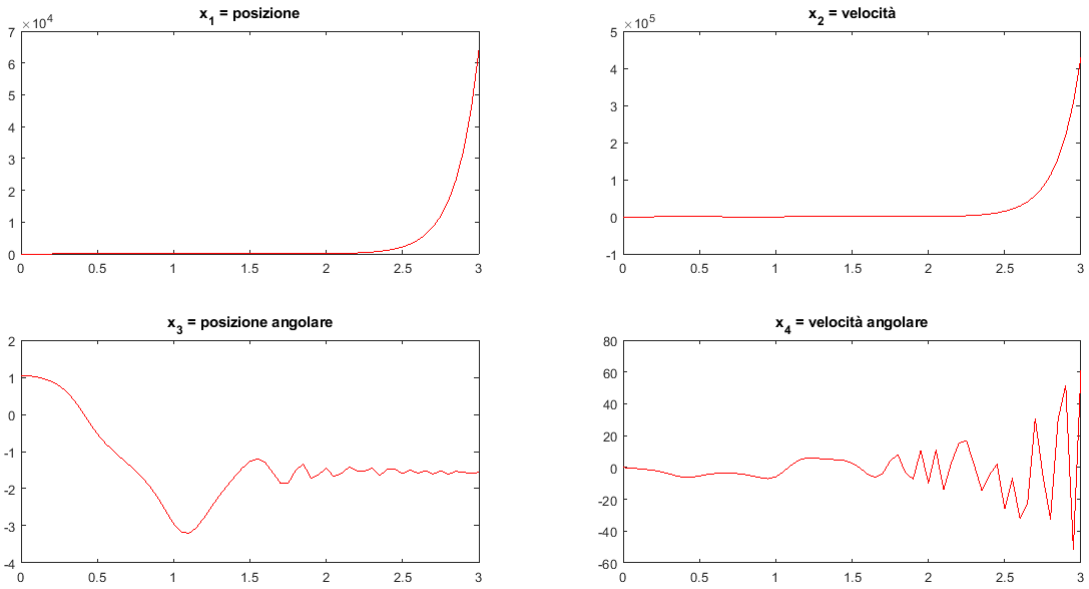
```
%% STATO INIZIALE LONTANO DALL'EQUILIBRIO

x0 = [0 0 pi/3 .0]';

xhat0 = x0 + [0.01 -0.02 -0.01 0.02]';


%% LANCIARE IL MODELLO SIMULINK LINEARIZZATO (settare orizzonte
simulazione = 25)

tempo_lin = x.Time;

x1_lin = x.Data(:, 1);

x2_lin = x.Data(:, 2);

x3_lin = x.Data(:, 3);

x4_lin = x.Data(:, 4);
```

```
figure
subplot(2, 2, 1)
plot(tempo_lin, x1_lin, 'b')
title('x_1 = posizione')
subplot(2, 2, 2)
plot(tempo_lin, x2_lin, 'b')
title('x_2 = velocità')
subplot(2, 2, 3)
plot(tempo_lin, x3_lin, 'b')
title('x_3 = posizione angolare')
subplot(2, 2, 4)
plot(tempo_lin, x4_lin, 'b')
title('x_4 = velocità angolare')
```




```
%% LANCIARE IL MODELLO SIMULINK NON LINEARE (settare orizzonte
simulazione = 3)
tempo_nonLin = x.Time;
x1_nonLin = x.Data(:, 1);
```

```
x2_nonLin = x.Data(:, 2);

x3_nonLin = x.Data(:, 3);

x4_nonLin = x.Data(:, 4);


subplot(2, 2, 1)

plot(tempo_nonLin, x1_nonLin, 'r')

title('x_1 = posizione')

subplot(2, 2, 2)

plot(tempo_nonLin, x2_nonLin, 'r')

title('x_2 = velocità')

subplot(2, 2, 3)

plot(tempo_nonLin, x3_nonLin, 'r')

title('x_3 = posizione angolare')

subplot(2, 2, 4)

plot(tempo_nonLin, x4_nonLin, 'r')

title('x_4 = velocità angolare')
```

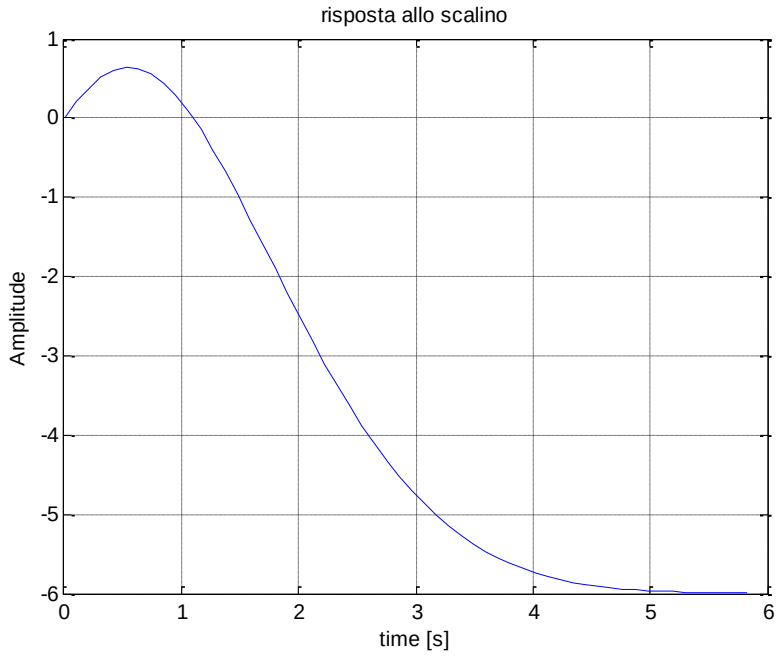


Anche dovendo ricostruire lo stato, la situazione è analoga a quella vista al punto 5. La stabilizzazione del sistema non lineare è possibile solo quando la condizione inziale è vicina all'equilibrio.

Esercizio 3: risposte canoniche

Per tracciare le risposte canoniche a scalino e impulso del sistema dato occorre definire il sistema in spazio di stato nel workspace di matlab (vedi esercitazione 1); pertanto i comandi da digitare nel prompt sono i seguenti:

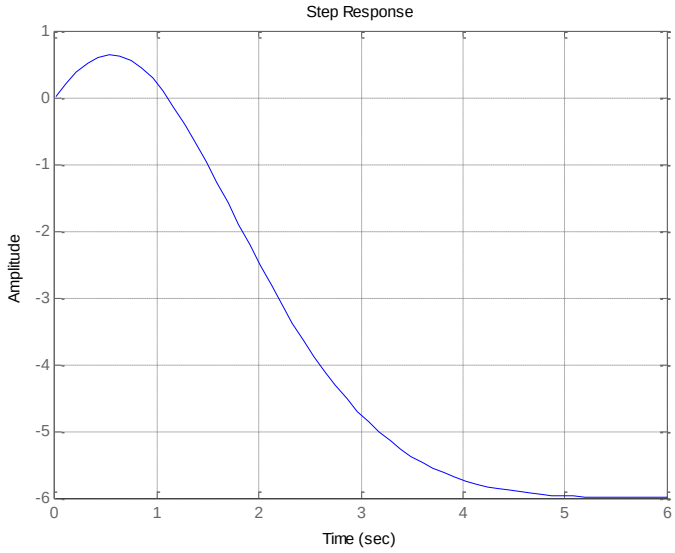
```
>> A=[1 1 0; -5 -3 0; 2 3 -1];
>> b=[1; 1; 0];
>> c=[1 1 1];
>> d=0;                          % definizione matrici del sistema
>> S=ss(A,b,c,d);                % definizione del sistema
>> [out_sca,time_sca]=step(S);   % generazione dati risposta scalino
>> plot(time_sca,out_sca);       % rappresentazione risposta scalino
```



Alternativa: matlab mostra direttamente il grafico del risultato della risposta a scalino se il comando è dato nel seguente modo:

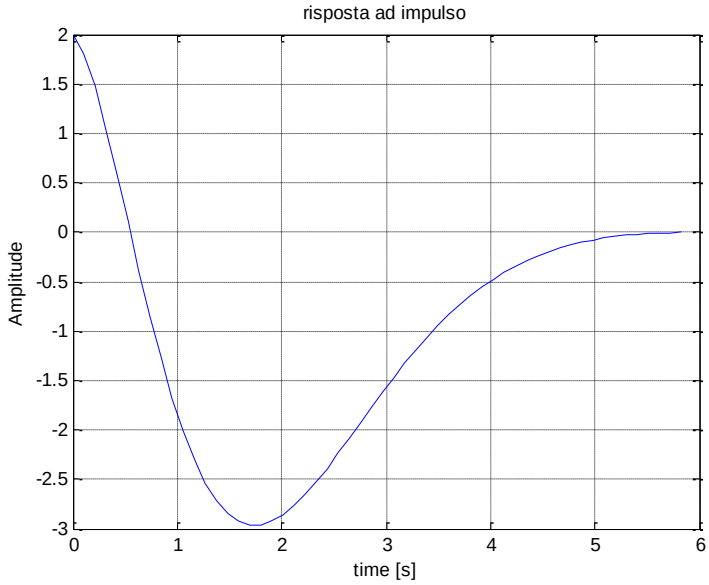
```
>> step(S);
```

Lo svantaggio in tal caso è che non si hanno a disposizione i vettori in cui sono memorizzati i valori della risposta allo scalino.



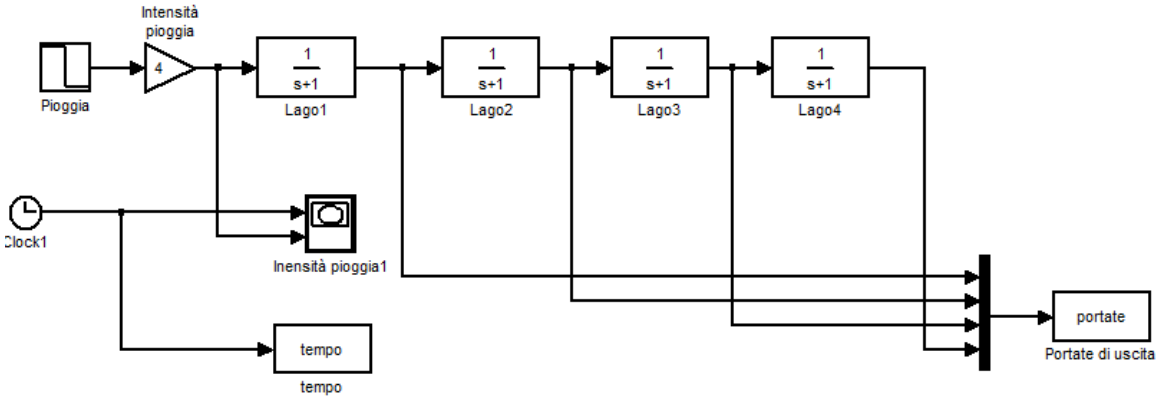
Analogamente, per la risposta all'impulso si può usare il comando `impulse()`:

```
>> [out_imp,time_imp]=impulse(S); % gener. dati risposta impulso
>> plot(time_imp,out_imp);        % rappr. risposta scalino
```



Esercizio 4: Risposte canoniche

Lo schema simulink della rete idrica composta, per esempio, da quattro laghi in cascata (identici e con costante di deflusso unitaria) è mostrato nella figura seguente (file laghi.mdl)



Il grafico seguente (`plot(tempo,portate)`) dimostra quanto richiesto