Laboratorio 2 per Fondamenti di Automatica (Matematici)

Esercizio 1 - Soluzione

Per risolvere questo esercizio occorre tradurre in uno schema Simulink il sistema non lineare dato

$$\dot{x}_1 = r_1 x_1 \left(1 - \frac{x_1}{K_1}\right) - a x_1 x_2$$

$$\dot{x}_2 = r_2 x_2 \left(1 - \frac{x_2}{K_2}\right) - a x_1 x_2$$

dove $r_1 = 5$, $r_2 = 5$, $K_1 = 1$, $K_2 = 2$, $a = 10$.[1]

A tale scopo si può utilizzare il blocco Simulink *Fcn* contenuto nella libreria *User Defined Function*; infatti il sistema si può vedere come:

$$\dot{x}_1 = f_1(x_1, x_2)$$

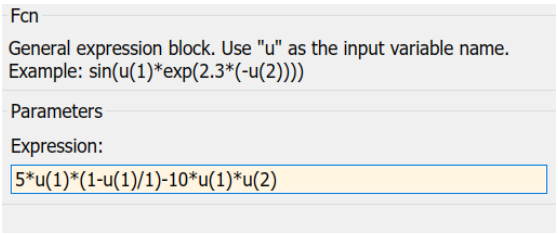
$$\dot{x}_2 = f_2(x_1, x_2)$$

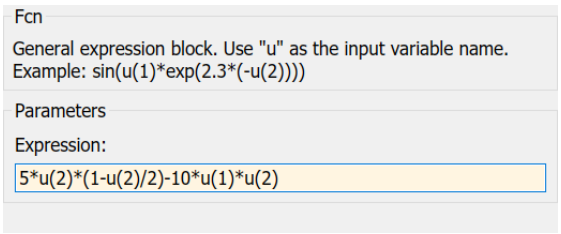
Pertanto basta definire due funzioni, che abbiano ognuna i due ingressi x_1 e x_2 e siano fatte come f_1 e f_2. Per costruire lo schema occorrono dei blocchi multiplexer (*mux* nella libreria *Signal Routing*) per generare gli ingressi delle funzioni; infatti il blocco *Fcn* accetta in ingresso un vettore colonna del tipo:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

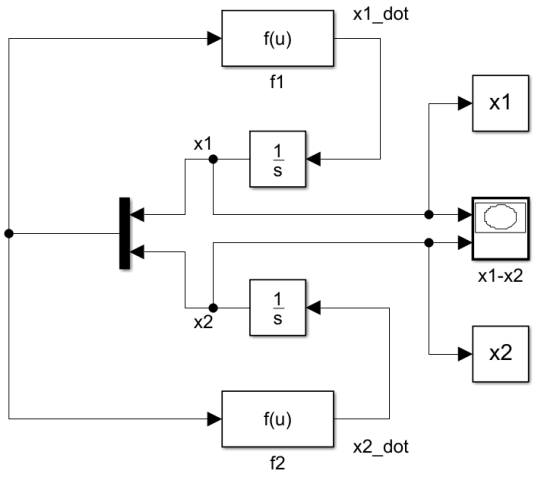
Nel nostro caso u deve contenere x_1 e x_2. Le variabili x_1 e x_2 sono inoltre ottenute integrando (tramite un integratore con funzione di trasferimento $1/s$ - libreria *Continuous*) l'uscita dai blocchi *Fcn* (le derivate di x_1 e x_2).

I blocchi Fcn, caratterizzanti le funzioni f_1 e f_2, sono rispettivamente così definiti:



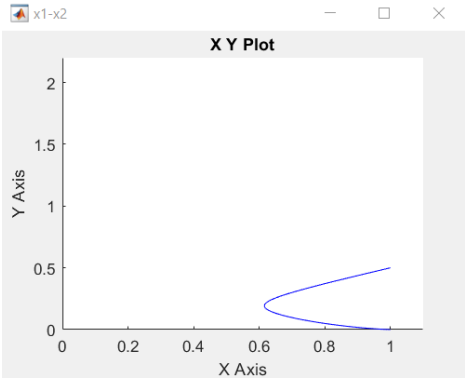


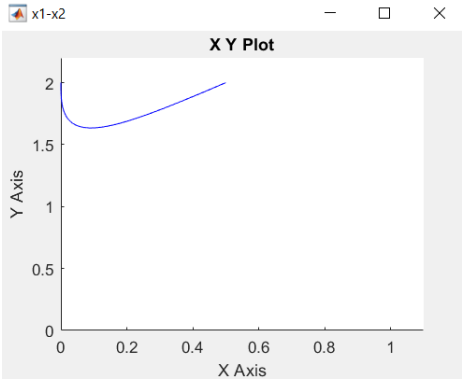
Per rappresentare le traiettorie nello spazio di stato, utilizziamo il blocco *X-Y graph*, all'interno della libreria *Sinks*. Quindi lo schema, completo anche dei grafici per osservare i risultati della simulazione risulta essere:



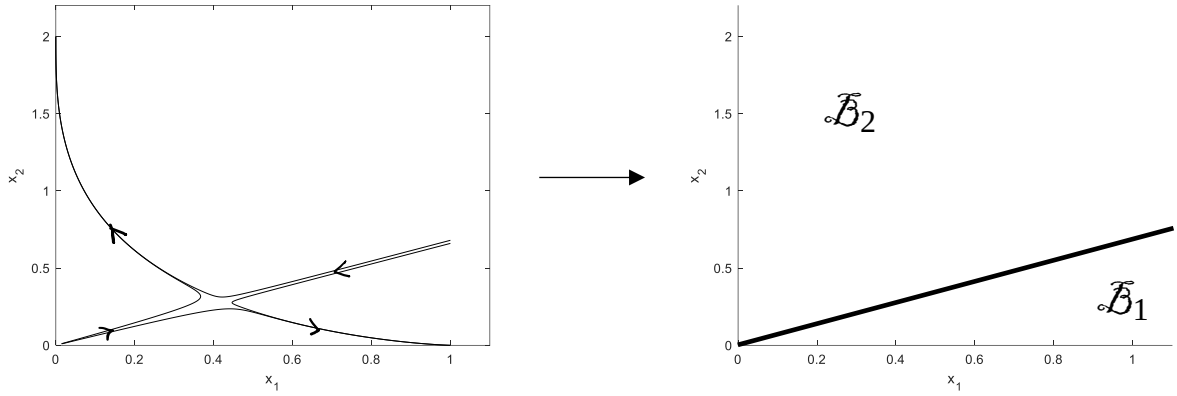
[1] Gli equilibri del modello sono: $E_0 = (0, 0)$, $E_1 = (K_1, 0) = (1, 0)$, $E_2 = (0, K_2) = (0, 2)$, $E_3 = (3/7, 2/7)$. Con il metodo della linearizzazione, si dimostra che E_0 ed E_3 sono instabili mentre E_1 ed E_2 sono asintoticamente stabili.

Simulando il comportamento del sistema (per 10 unità di tempo) si ottengono i seguenti risultati (la condizione iniziale fissata nei blocchi integratori è pari a (1, 0.5) e (0.5, 2)):

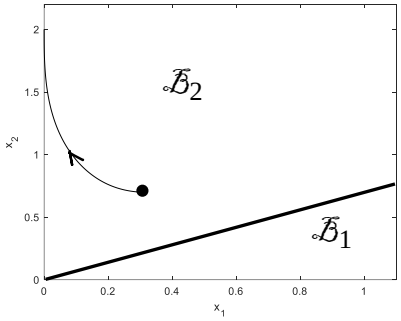




Il sistema presenta quindi due comportamenti asintoticamente stabili alternativi: $(K_1, 0)$ e $(0, K_2)$. Per valutarne i bacini di attrazione, si eseguono varie simulazioni per varie condizioni iniziali ((1, 0.68), (1, 0.66), (0.016, 0.01) e (0.014, 0.01)) e il risultato è mostrato in figura[2]:

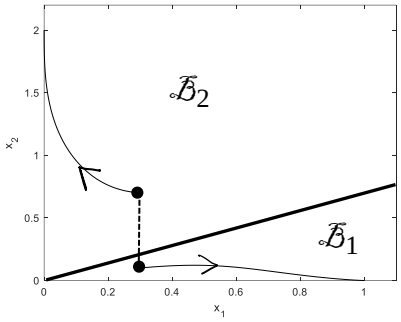


Supponiamo ora che si stia instaurando una situazione che porti a uno stato di malattia (dominanza di batteri nocivi). Possiamo ipotizzare quindi che lo stato iniziale del sistema sia, ad esempio, il punto (0.3, 0.7); essendo questo nel bacino di attrazione $\mathcal{B}_2$ dell'equilibrio $(0, K_2)$, tenderanno a dominare i batteri nocivi.



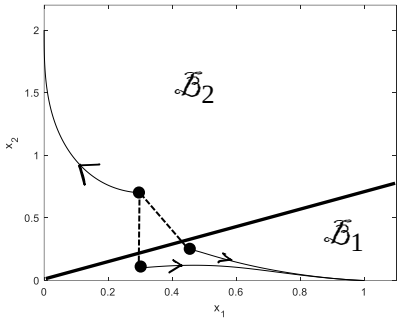
Per portare a guarigione l'individuo si possono utilizzare antibiotici che riducono i batteri nocivi. Supponendo quindi che la condizione iniziale passi, a seguito dell'assunzione, al valore (0.3, 0.1) (appartenente al bacino $\mathcal{B}_1$), il sistema si porterà alla dominanza dei batteri utili $(K_1, 0)$.

[2] La frontiera dei bacini di attrazione è una traiettoria rettilinea per il sistema (quella segnata in neretto). Tale traiettoria è percorsa, allontanandosi da E_0, convergendo verso E_3. L'equazione di tale retta ($x_2 = \alpha\, x_1$) si trova determinando il parametro α per il quale, presa una condizione iniziale $x_2(0) = \alpha\, x_1(0)$, si ha: $\dot{x}_2(0) = \alpha \dot{x}_1(0)$. Con alcuni passaggi si ottiene $\alpha = 2/3$.



Tuttavia, per ridurre in tale modo la densità dei batteri nocivi, è necessario un forte uso (abuso) di antibiotici che può essere nocivo per l'individuo. Pertanto, si può pensare a una cura combinata di antibiotici (in moderata quantità) e fermenti lattici (che aumentano la densità di batteri utili).

Supponendo pertanto una nuova condizione iniziale pari a (0.45, 0.25) in cui la riduzione di densità batterica nociva è inferiore rispetto al caso del solo uso di antibiotici, il sistema si può comunque trovare in $\mathcal{B}_1$ e portare così l'individuo a piena guarigione.



Esercizio 2 - Soluzione

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 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:

```
%% dati del problema
M = 10;
m = 1;
L = 1;
g = 9.81;

%% equazioni del sistema 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];

%% stabilità del sistema
autovalori = eig(A)

autovalori =

         0
         0
    3.2850
   -3.2850
```

Il sistema è instabile.

```
%% raggiungibilità e osservabilità, prima coppia: forza -
posizione
ra = ctrb(A,B(:,1));    % selezione ingresso forza

disp(det(ra))

    0.0096
```

Essendo il determinante della matrice di raggiungibilità non nullo, si ha la completa raggiungibilità del sistema considerando come ingresso la forza $u_1 = \delta F$ agente sul carrello.

```
os = obsv(A,C(1,:));    % selezione misura posizione

disp(det(os))

    0.9624
```

Essendo il determinante della matrice di osservabilità non nullo, si ha la completa osservabilità del sistema considerando come uscita la posizione $x_1 = \delta P$ del carrello.

```
%% raggiungibilità e osservabilità, seconda coppia: coppia -
posizione angolare
ra = ctrb(A,B(:,2));    % selezione ingresso coppia

disp(det(ra))

    0
```

Essendo il determinante della matrice di raggiungibilità nullo, non si ha la completa raggiungibilità del sistema considerando come ingresso la forza $u_2 = \delta \tau$ agente sull'asta.

```
ob = obsv(A,C(2,:));    % selezione misura posizione angolare

disp(det(ob))

    0
```

Essendo il determinante della matrice di osservabilità nullo, non si ha la completa osservabilità del sistema considerando come uscita la posizione $x_3 = \delta \vartheta$ del carrello.

Pertanto si è dimostrato quanto chiesto dal testo al punto 2.

```
%% utilizzo la prima coppia. ridefinisco le matrici del sistema
B = B(:, 1);
C = C(1, :);
D = D(1, 1);
```

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()` (digitare `help acker` per una discussione dettagliata sull'uso del comando). In particolare, il comando fa

riferimento a una legge di controllo $u = -kx$ e conseguentemente a una matrice dinamica del sistema controllato pari ad $(A - bk)$. Pertanto, nell'uso del comando, il vettore b andrà sostituito con il vettore $-b$

```
%% Progetto della legge di controllo
% ricavo le radici del polinomio
autov_controllore = [roots([1 1.5 1]); roots([1 2 1])]

autov_controllore =

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

k = acker(A, -B, autov_controllore)

k =

   1.0194    3.5678  158.9294   38.5678
```

Verifichiamo che `A + B*k` abbia gli autovalori desiderati

```
disp(eig(A + B*k))

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

4. In modo analogo, cioè mediante il comando `acker`, si può ottenere il ricostruttore dello stato. Il problema della stabilizzazione dell'errore di stima dell'osservatore è infatti il duale del problema della stabilizzazione del sistema affrontato al punto precedente.
Osserviamo che $(A - lc)^T$ ha gli stessi autovalori di $(A - lc)$. Ma $(A - lc)^T = A^T - c^T l^T$ dove A^T, c^T e l^T sono in posizione corrispondente ad A, b e k per l'assegnamento degli autovalori nella legge di controllo.
Pertanto basta applicare la formula di Ackermann alla coppia $(A^T, -c^T)$ per ottenere l^T.

```
%% Progetto del ricostruttore dello stato
% ricavo le radici del polinomio
autov_ricostruttore = [roots([1 15 100]); roots([1 20 100])]

autov_ricostruttore =

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

lT = acker(A', -C', autov_ricostruttore);
```

```
l = lT'

l =

  1.0e+004 *

    -0.0035
    -0.0511
     0.3953
     1.5812
```

Verifichiamo che `A + l*C` abbia gli autovalori desiderati

```
disp(eig(A + l*C))

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

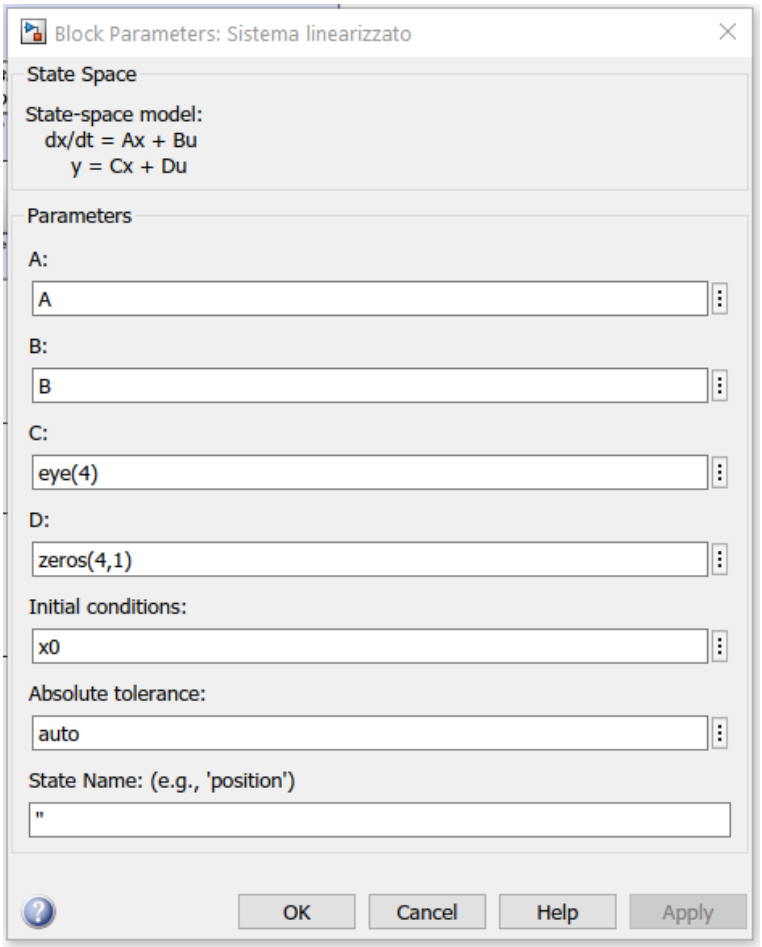
Verifichiamo che il sistema complessivo abbia gli autovalori desiderati

```
disp(eig([A, B*k; -l*C, A+l*C+B*k]))

  -7.5000 + 6.6144i
  -7.5000 - 6.6144i
 -10.0000 + 0.0000i
 -10.0000 + 0.0000i
  -0.7500 + 0.6614i
  -0.7500 - 0.6614i
  -1.0000 + 0.0000i
  -1.0000 + 0.0000i
```

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:

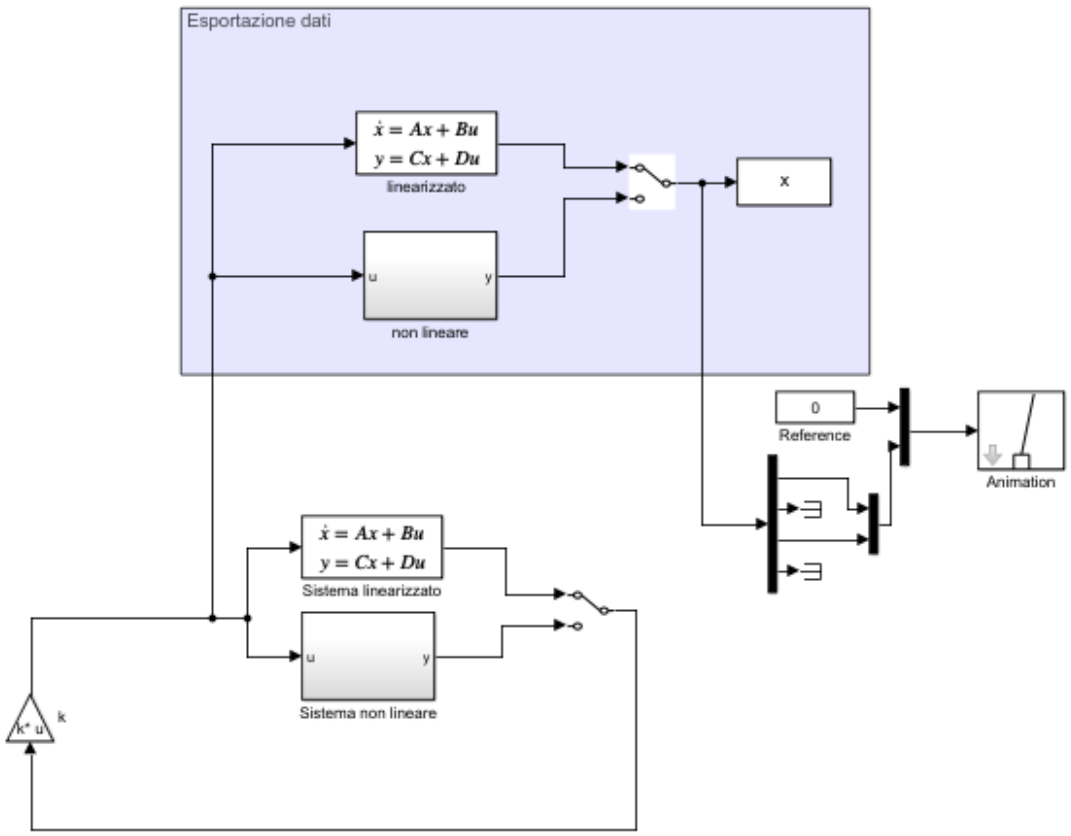


Questo perché le matrici *A* e *B* descrivono la dinamica e gli ingressi del blocco, mentre le matrici *C* e *D* descrivono l'uscita dello stesso; se si vogliono ottenere le informazioni su tutto lo stato occorre che la matrice *C* sia una matrice identità e che il vettore *D* sia nullo.

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:



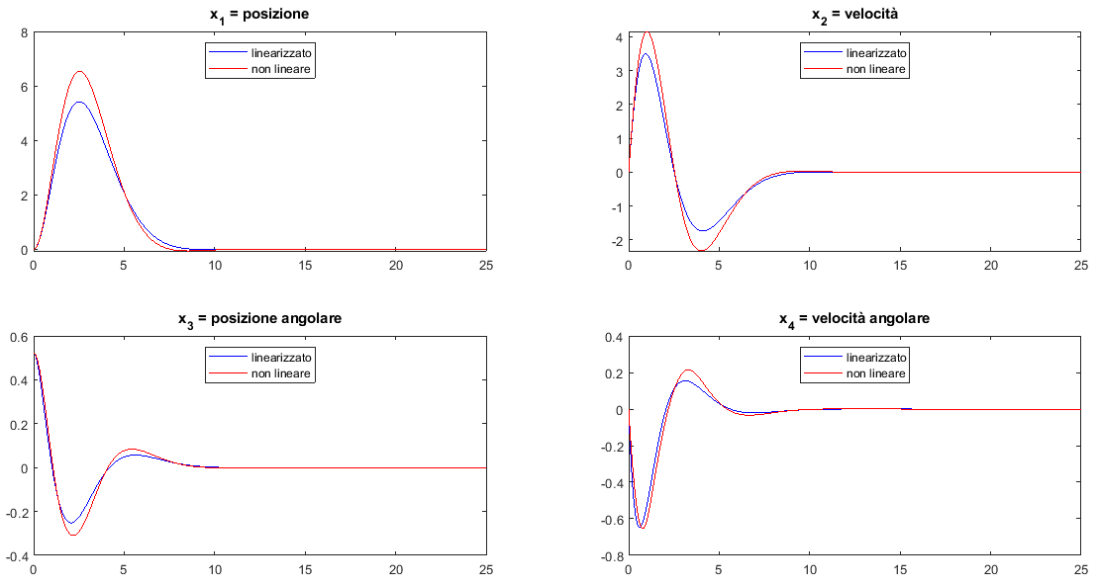
Il sotto sistema dedicato all'esportazione dei dati permette di salvare l'andamento delle quattro variabili di stato del sistema linearizzato (se il blocco switch è posizionato come in figura) o di quello non lineare (fare doppio click sul blocco switch passare dal sistema linearizzato a quello non lineare, 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.

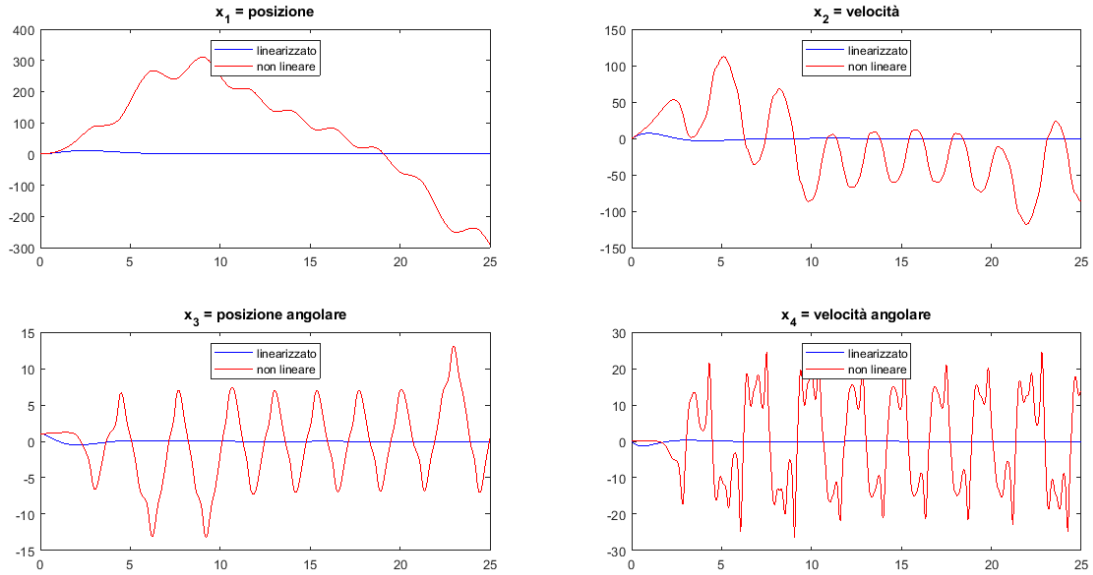
```
%% LANCIARE IL MODELLO SIMULINK NON LINEARE (settare orizzonte
simulazione = 25)
x1_nonLin = x.Data(:, 1);
x2_nonLin = x.Data(:, 2);
x3_nonLin = x.Data(:, 3);
x4_nonLin = x.Data(:, 4);

%% 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]';
```

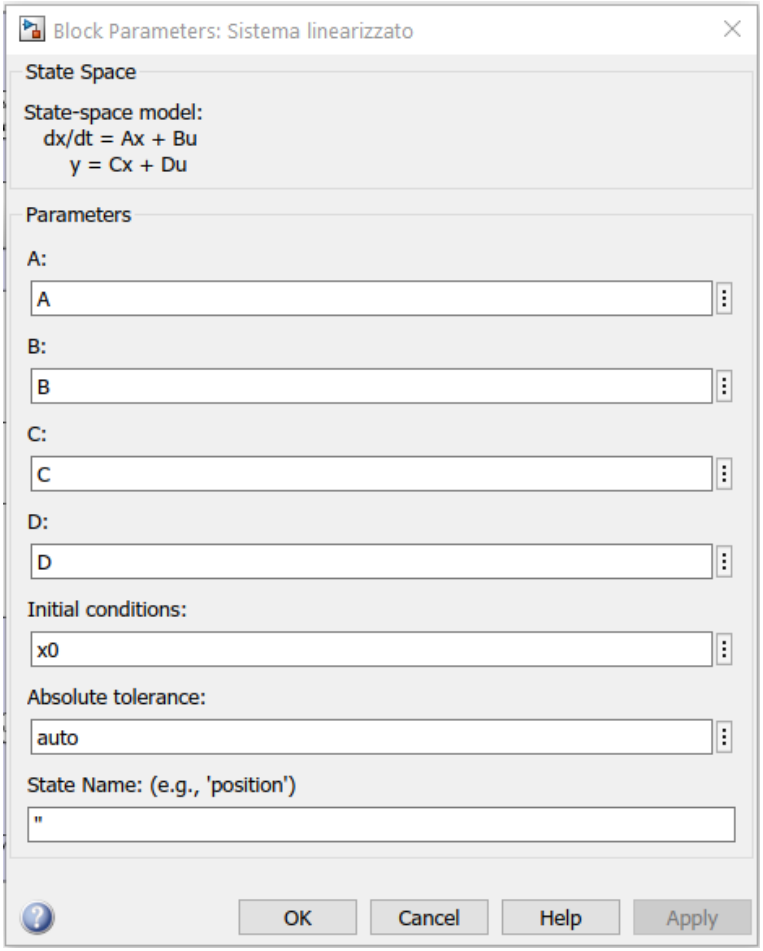
Utilizzando nuovamente il codice matlab e lo schema simulink già visti in precedenza, si ottiengono i seguenti grafici:



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:

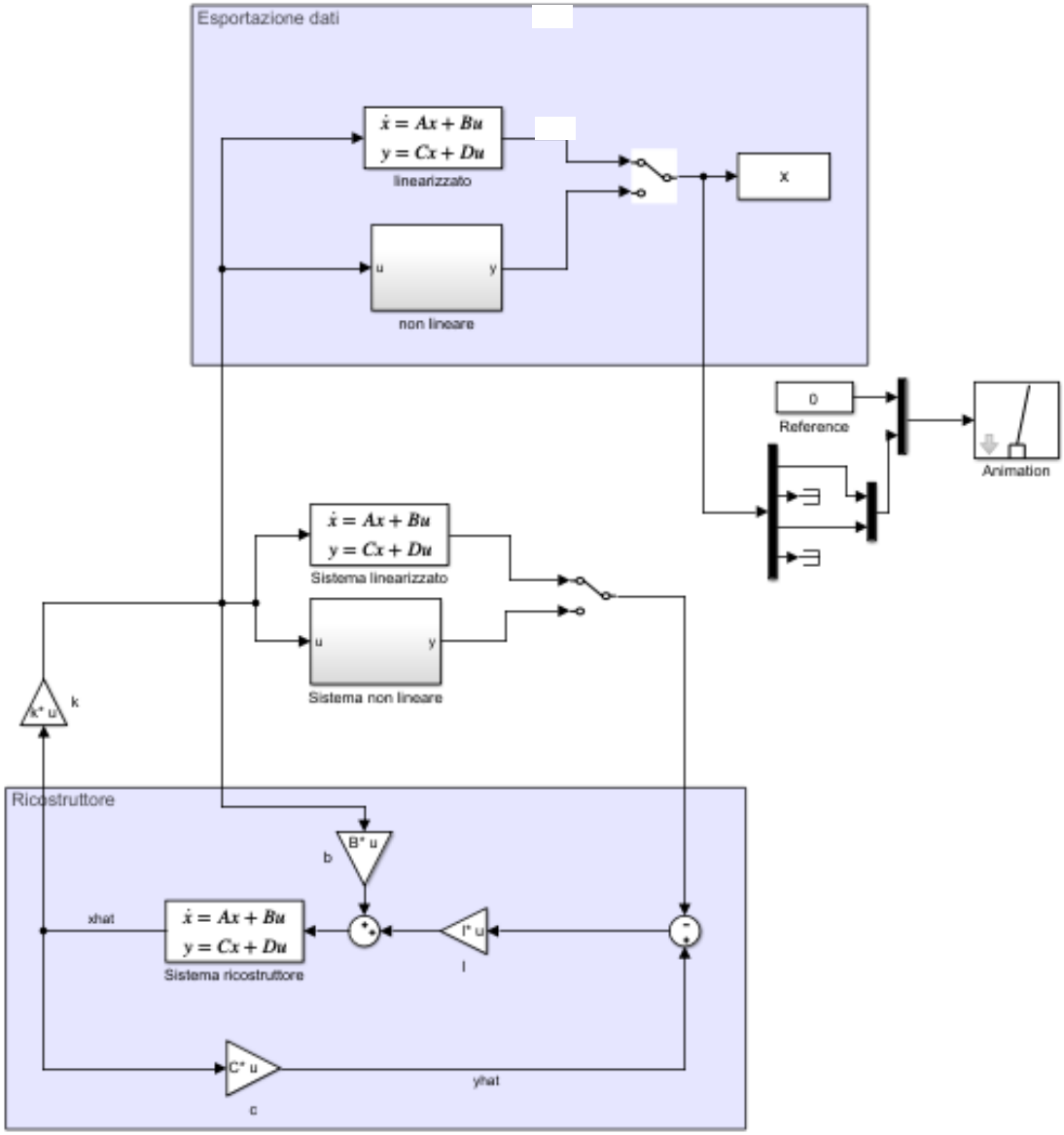
$$\begin{cases} \dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + l(\hat{y}(t) - y(t)) \\ \hat{y}(t) = C\hat{x}(t) \end{cases}$$

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.

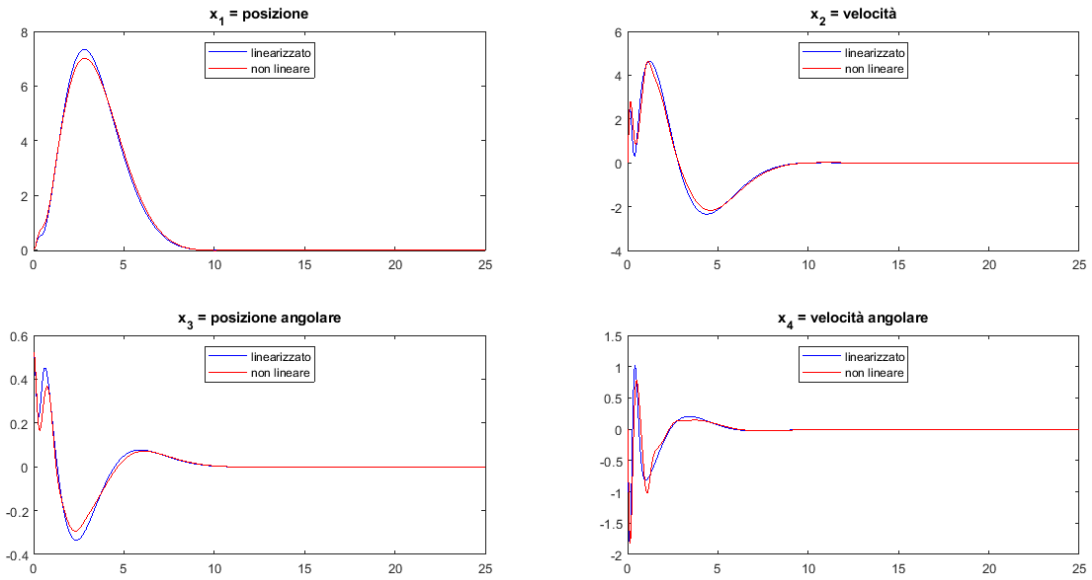
```
%% LANCIARE IL MODELLO SIMULINK LINEARIZZATO (settare orizzonte
```

```
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')
```

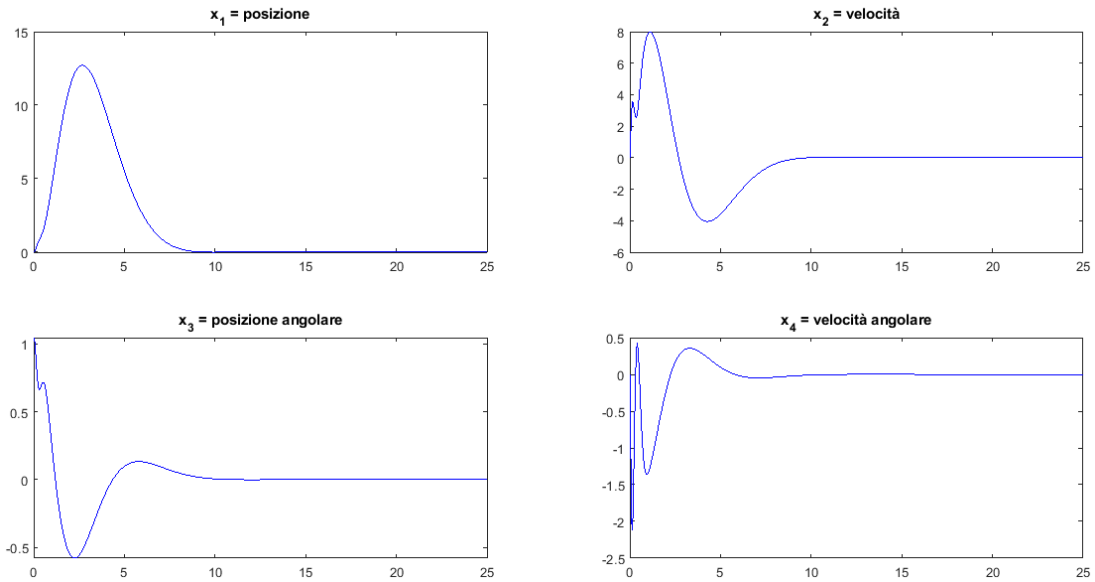
I risultati che si ottengono sono i seguenti:



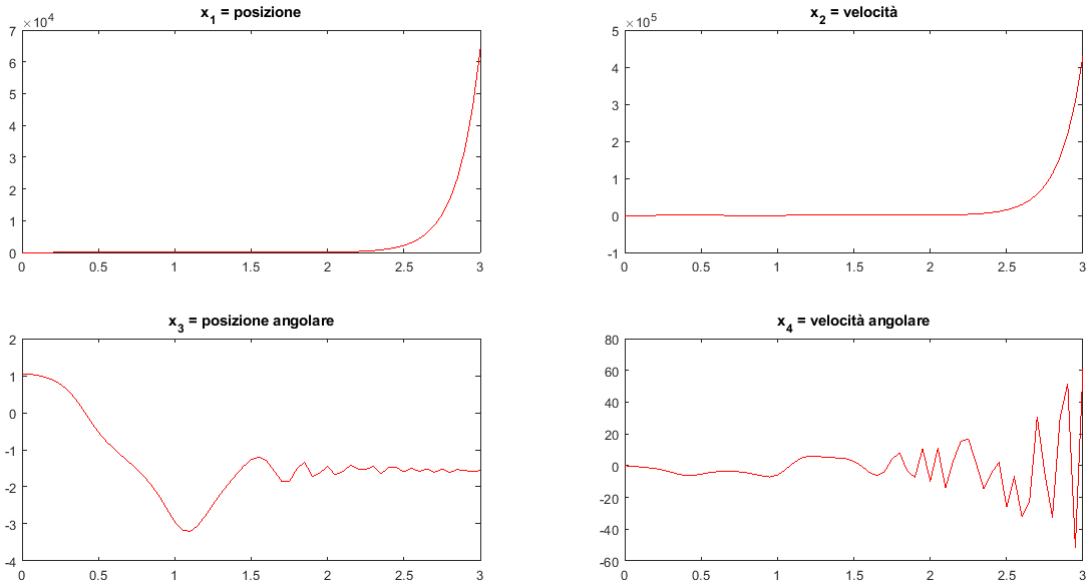
È quindi possibile stabilizzare il sistema anche quando non si ha accesso all'intero stato ma si osserva solo l'uscita del sistema (purchè la condizione di osservabilità sia rispettata).
Attenzione, un'incertezza sulla misura dello stato iniziale troppo elevata potrebbe avere lo stesso effetto di quello 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.