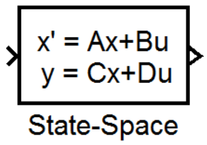
# *Simulink*

### Tracce di possibili soluzioni

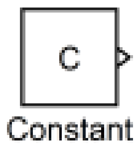
## Esercizio 1

Per studiare la stabilità del sistema in esame utilizzando Simulink occorre innanzitutto realizzare il sistema in modo che sia simulabile, per poi alimentarlo con ingresso nullo, a partire da uno stato iniziale non nullo, ed osservare il movimento libero del sistema.

Avendo a disposizione le matrici del sistema espresso in forma di stato è possibile utlizzare il blocco *State-Space* di simulink:

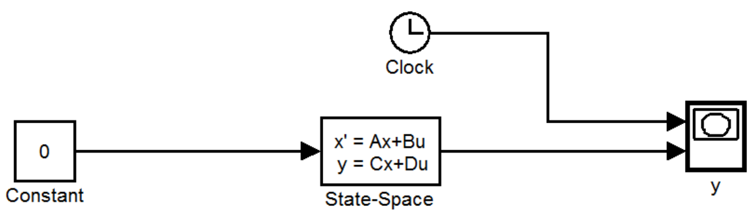



Per poter simulare il sistema non alimentato (u=0), occorre definire come ingresso una costtante e porla pari a 0. Si utilizzi il blocco *Constant*, all'interno della libreria *Sources*:



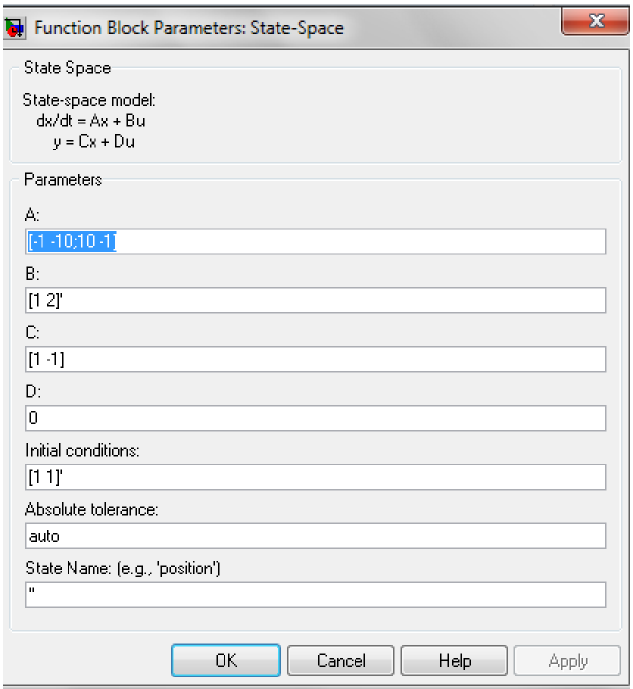

Per rappresentare l'uscita in un grafico, utilizziamo il blocco *X-Y graph*, all'interno della libreria *sinks*; questo blocco necessita di due ingressi: il primo è il vettore che verrà rappresentato lungo l'asse $x$, il secondo quello che verrà rappresentato sull'asse $y$. La variabile indipendente, in questo caso è il tempo. Prendiamo dunque il blocco clock per generare il tempo e colleghiamolo alla porta "$x$" del blocco *X-Y graph*.


Connettendo i blocchi come descritto si trova il seguente schema:



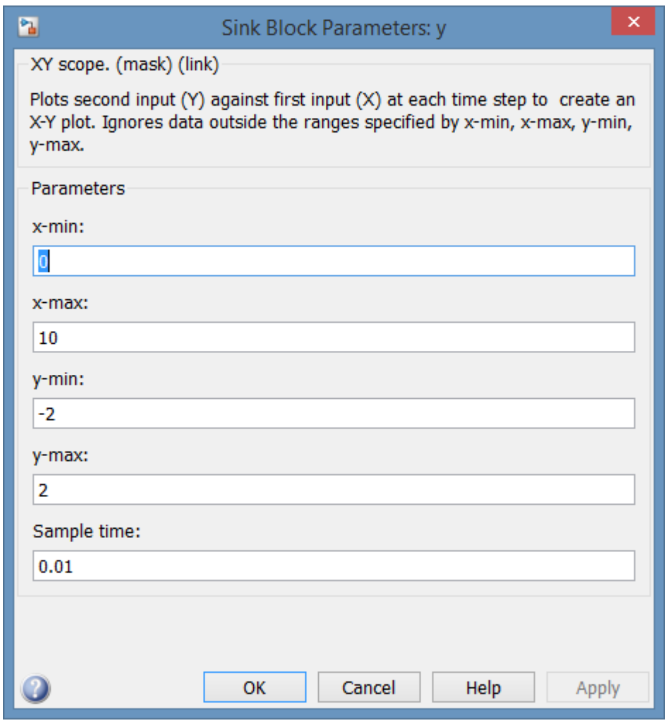
Si noti che il valore dell'ingresso è stato già messo a 0. Ora occorre impostare le matrici del blocco *State-Space*.

Cliccando due volte sul blocco appaiono le proprietà dello stesso; qui si possono definire le matrici come indicate nel testo dell'esercizio; dunque si avrà:

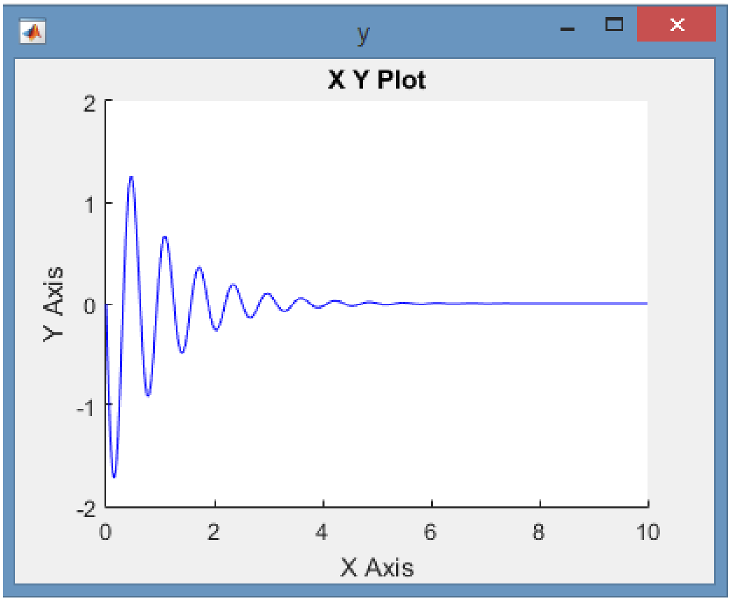



Si noti che è stato impostato uno stato iniziale non nullo per poter osservare il moto libero del sistema, in particolare x(0) = $[1\ 1]^T$.

Ora è possibile simulare il comportamento; infine, una volta che si ha un riscontro grafico è possibile cambiare la scala del blocco *X-Y graph* in modo da ottenere una visualizzazione più chiara, agendo sulle proprietà del blocco:



Simulando ora per 10 secondi si ottiene il seguente risultato:




Si noti che l'uscita del sistema tende a zero, quindi il sistema è asintoticamente stabile.

Ciò poteva essere osservato calcolando gli autovalori della matrice $A$, grazie al comando `eig()` di `matlab`; si sarebbe ottenuto:

```
>> A=[-1 -10;10 -1];
>> eig(A)

ans =

  -1.0000 +10.0000i
  -1.0000 -10.0000i
```

Pertanto, avendo il sistema due autovalori complessi coniugati a parte reale negativa, si spiega il comportamento oscillante e smorzato del sistema.

Per tracciare l'andamento degli stati in simulink si noti che l'equazione di uscita, essendo il sistema strettamente proprio ($d = 0$), si può riscrivere come:

$$y(t) = cx(t) = [c_1 \quad c_2] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = c_1 x_1(t) + c_2 x_2(t)$$

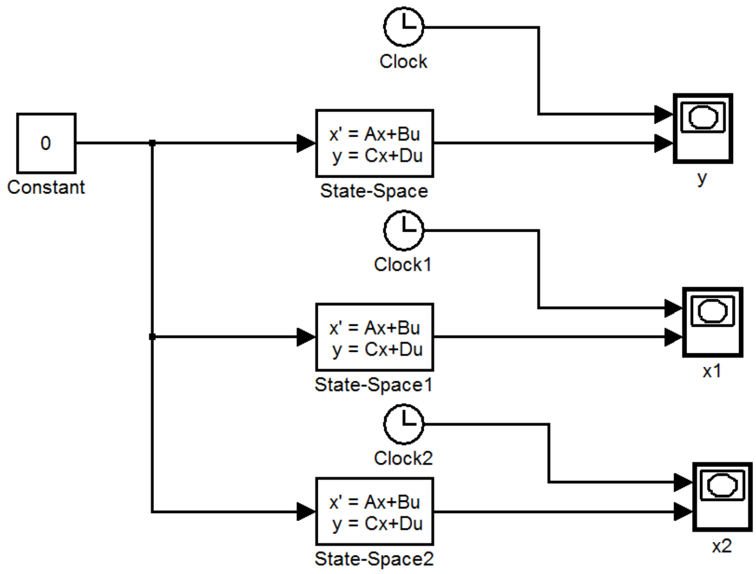
Pertanto se vogliamo rappresentare $x_1$ occorre prendere un sistema che abbia come uscita l'equazione:

$$y_1(t) = [1 \quad 0] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = x_1(t)$$

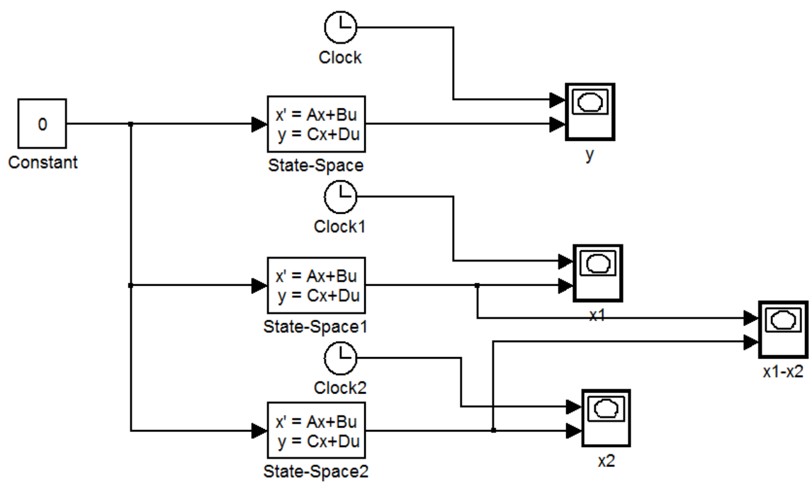
e analogamente per rappresentare $x_2$ deve essere:

$$y_2(t) = [0 \quad 1] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = x_2(t)$$

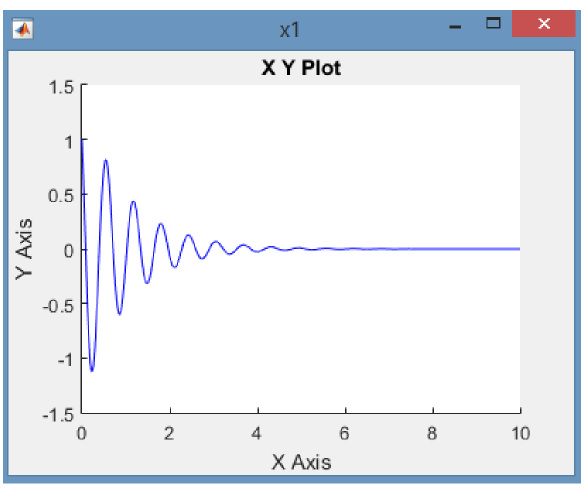
Quindi, basta copiare il sistema precedentemente costruito cambiando i parametri della matrice $C$, ottenendo lo schema:

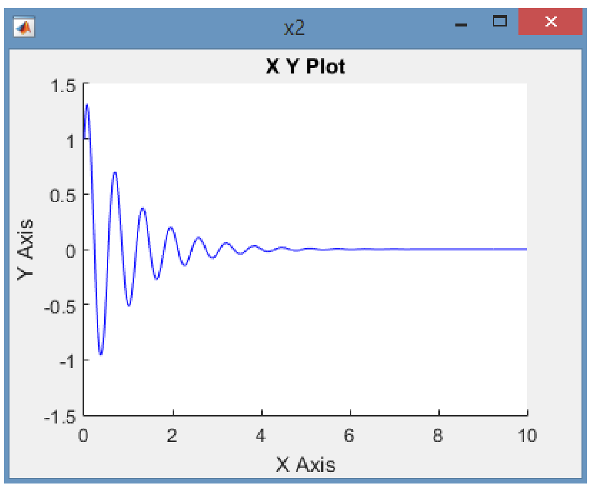



Per sistemi a due stati un modo sintetico per osservarne il comportamento è rappresentare le traiettorie nel piano $(x_1, x_2)$; quindi basta aggiungere un altro *X-Y graph* alimentato da $x_1$ e $x_2$:

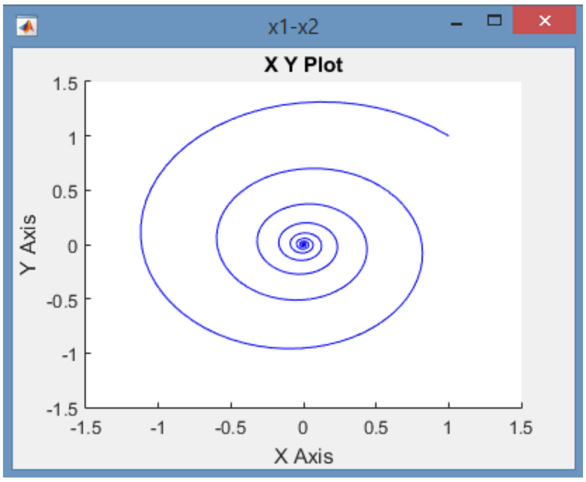



Simulando il sistema si ottengono i seguenti andamenti:

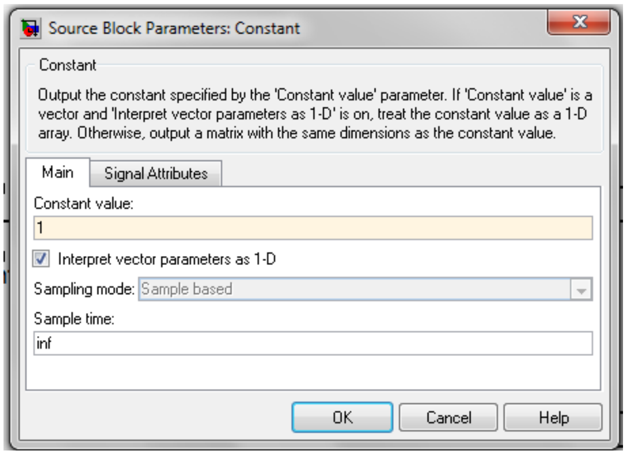


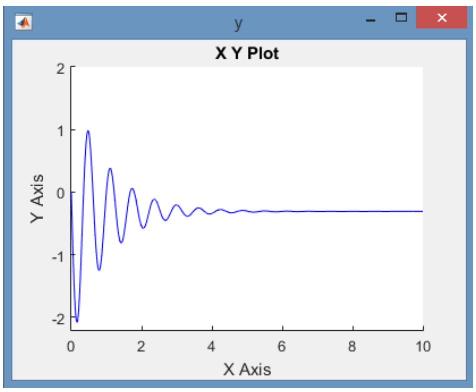
Il piano delle fasi risulta essere:

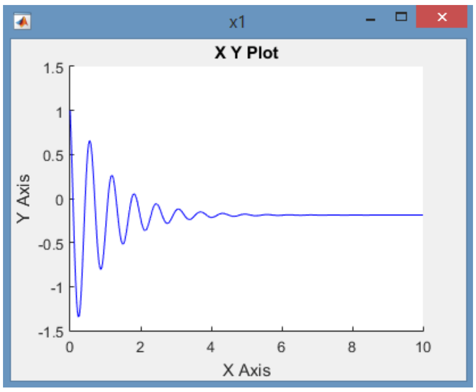


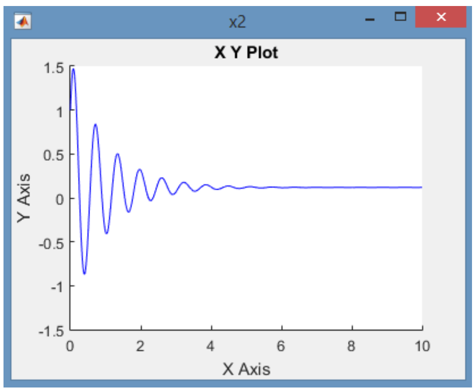

Infine, per simulare il sistema sottoposto a un ingresso forzato $u = 1$, è sufficiente cambiare il valore della costante in ingresso, andando a cambiare i parametri del blocco *Constant*:



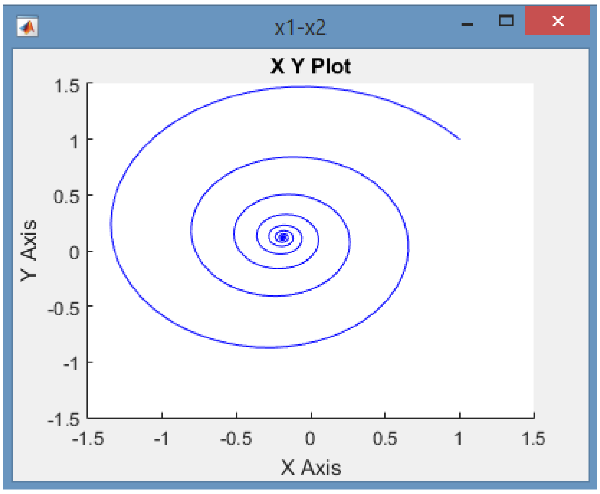

I risultati che si ottengono in questo caso sono i seguenti:





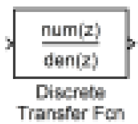


Infine il piano delle fasi in questo caso è dato da:

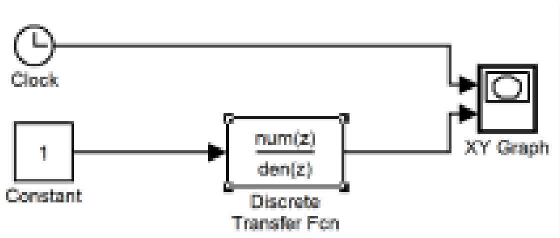



## Esercizio 2

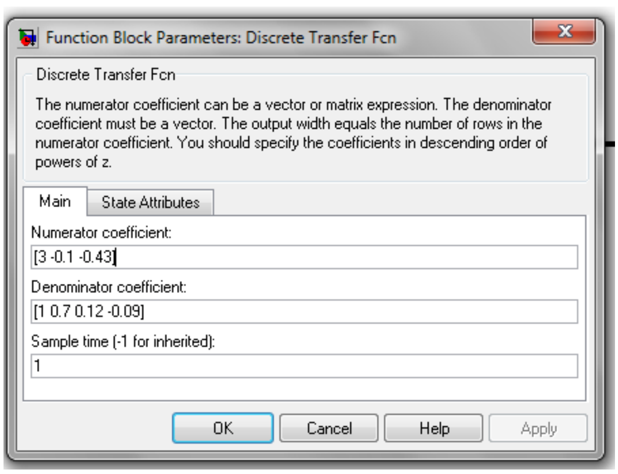
In questo caso ci è già data la funzione di trasferimento del sistema (che è a tempo discreto). Pertanto, in simulink, si può  utilizzare direttamente il blocco *transfer fcn*, che nel caso discreto si chiama *Discrete Transfer Fcn*:



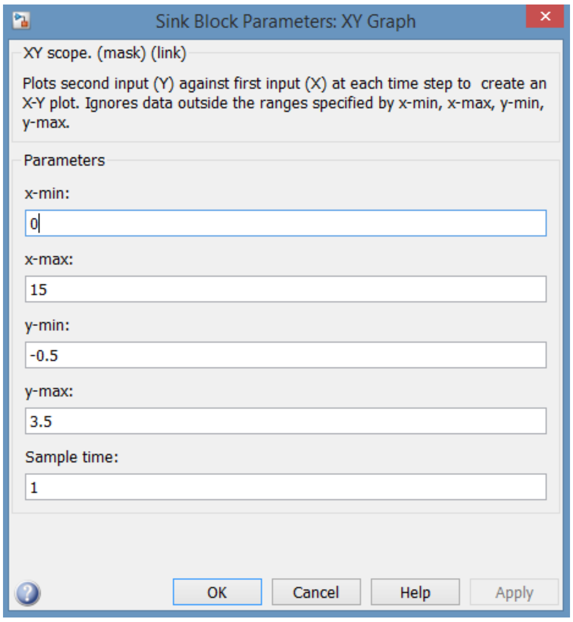

Come ingresso diamo una costante $u = 1$ e come uscita rappresentiamo il tutto, come fatto nell'esercizio precedente, in un grafico X-Y; lo schema è dunque:



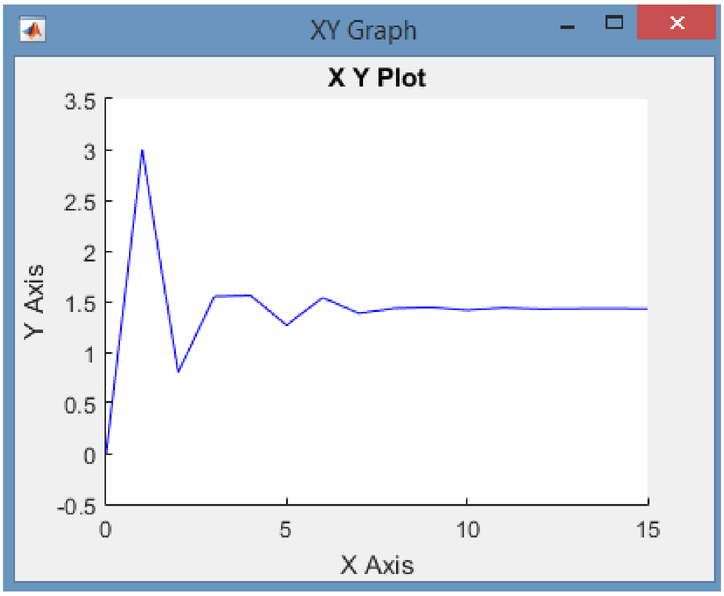

Per inserire i parametri nel blocco della funzione di trasferimento basta aprire le proprietà e scrivere numeratore e denominatore. Ciò va fatto come vuole matlab e cioè sotto forma di vettori riga:



Una volta inseriti i parametri, si può simulare il sistema e successivamente cambiare, se necessario, la scala del grafico e il tempo di simulazione (nel box di fianco al tasto play). In questo caso si è selezionato un tempo di simulazione di 15 istanti di tempo; i limiti dati al grafico $x, y$ sono i seguenti:




Il grafico che ne risulta è il seguente:




Si noti che il comportamento lascia presupporre che il sistema sia stabile asintoticamente; calcolando i poli della funzione di trasferimento si ottiene:

```
>> poli=roots([1 0.7 0.12 -0.09])

poli =

  -0.4755 + 0.3641i
  -0.4755 - 0.3641i
```

```
   0.2509

>> abs(poli)

ans =

    0.5989
    0.5989
    0.2509
```

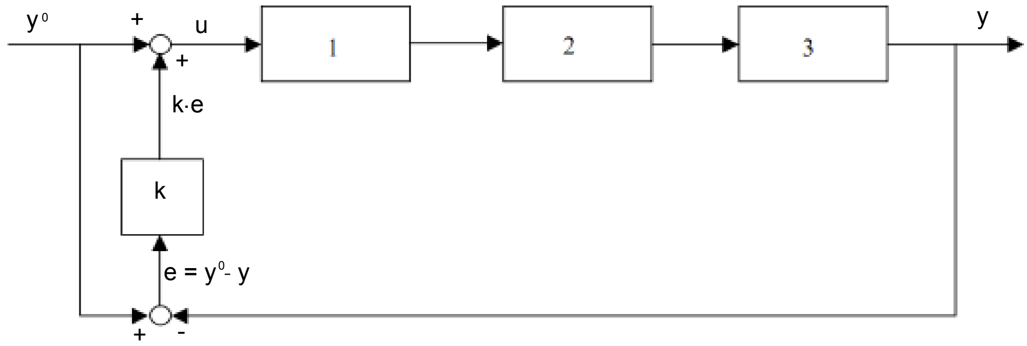
Il che conferma la stabilità asintotica del sistema.

Per rappresentare le variabili di stato è possibile riscrivere il sistema in forma di stato e utilizzare un metodo analogo a quanto già visto nell'esercizio 1; per trovare il sistema in forma di stato basta utilizzare il comando di matlab ; successivamente si utilizzi il blocco state-space discreto inserendo le matrici $(A, B, C, D)$ trovate e proseguire come per l'esercizio 1, duplicando il sistema per selezionare, una per volta, le variabili di stato.

```
>> [A,B,C,D]=tf2ss([0 3 -0.1 -0.43],[1 0.7 0.12 -0.09])

A =

   -0.7000   -0.1200    0.0900
    1.0000         0         0
         0    1.0000         0


B =

     1
     0
     0


C =

    3.0000   -0.1000   -0.4300


D =

     0
```

# Esercizio 3

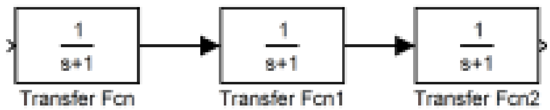
Per risolvere questo esercizio occorre tradurre lo schema a blocchi dato nel testo in uno schema simulink:




Sapendo che i blocchi 1, 2 e 3 hanno funzione di trasferimento data da:

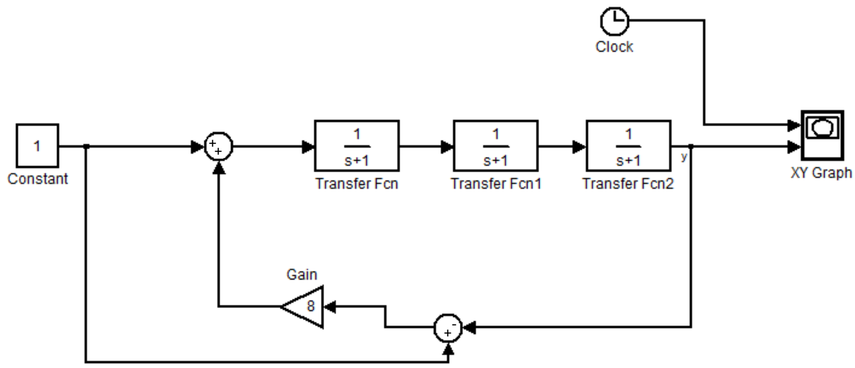
$$G(s) = \frac{1}{s+1}$$

per rappresentare lo schema precedente in simulink si può utilizzare il blocco *Transfer Fcn* per tre volte in serie:




per sommare o sottrarre due segnali esiste il nodo sommatore (denominato appunto *sum*); per cambiare i segni del nodo sommatore occorre entrare nelle proprietà dello stesso e specificarli nel campo "listo of signs".

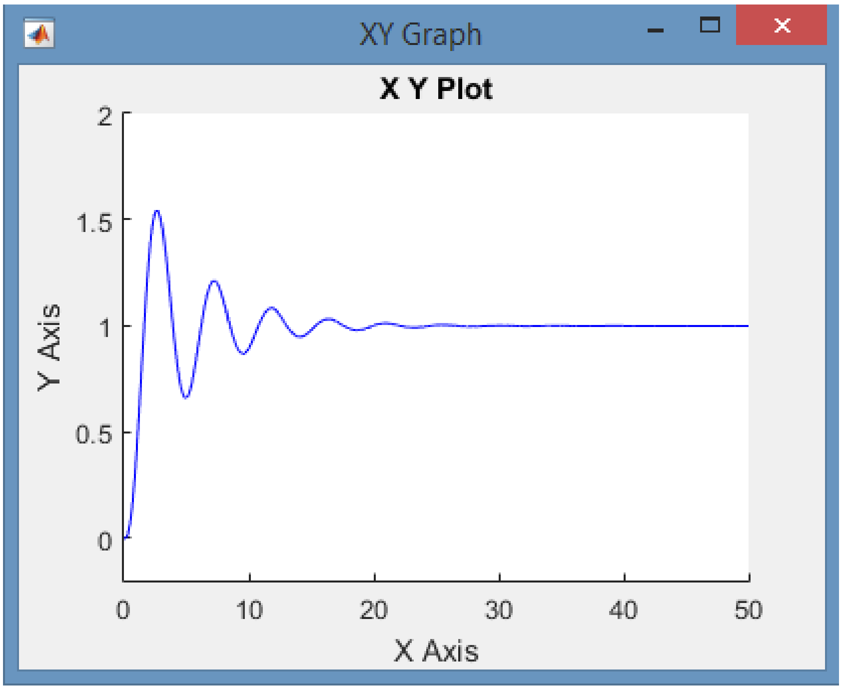
Infine, per moltiplicare un segnale (guadagno "k") si utilizza un blocco denominato *gain*. L'ingresso, come richiesto dal testo, sarà una costante con valore 1. Si può pertanto costruire il seguente schema:



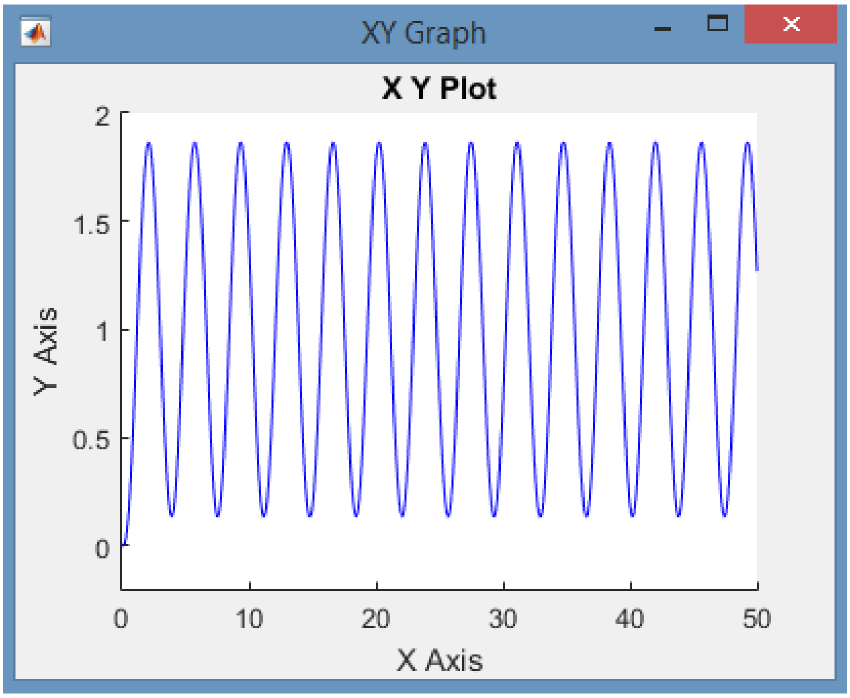

Si noti che è stato inserito il solito blocco *clock* per generare il tempo da inserire nel blocco *X-Y Graph*.

La richiesta dell'esercizio è di confermare i calcoli svolti a lezione fossero corretti, ovvero se il limite di stabilità del sistema fosse raggiunto per $k = 8$. Per fare ciò si svolgano 3 simulazioni con 3 diversi valori di $k$ ($k < 8, k = 8, k > 8$); si ottengono:

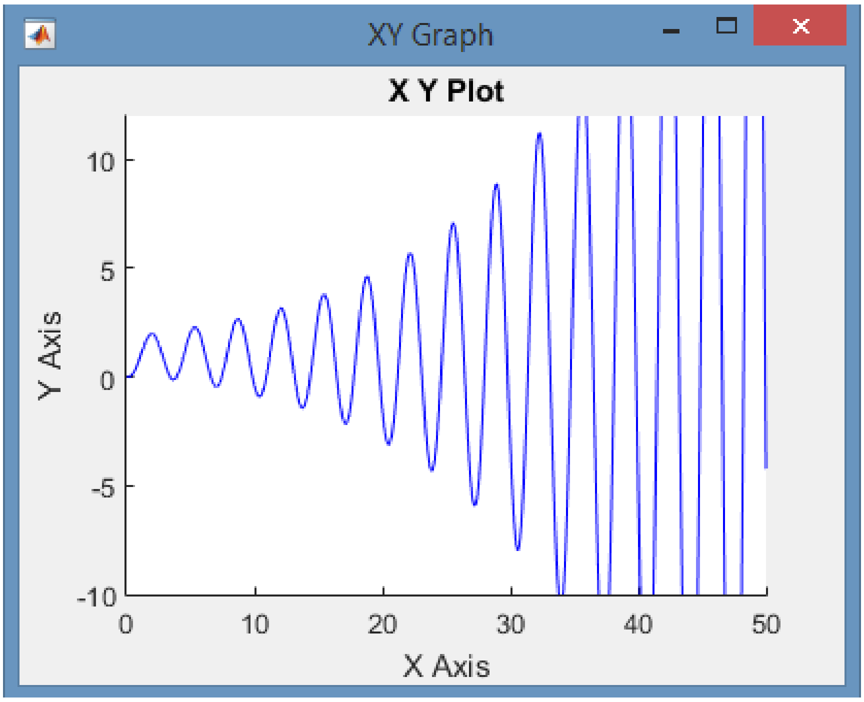
- $k = 4$:




- $k = 8$:



- $k = 10$:




Come si nota, il sistema risulta semplicemente stabile per $k = 8$ e perde di stabilità per $k > 8$.

## Esercizio 4

Per risolvere questo esercizio bisogna scrivere in linguaggio simulink il sistema dinamico non lineare dato

$$\begin{cases} \dot{x}(t) = rx(t)\left(1 - \dfrac{x(t)}{k}\right) - \dfrac{ax(t)}{b + x(t)}y(t) \\ \dot{y}(t) = e\dfrac{ax(t)}{b + x(t)}y(t) - my(t) \end{cases}$$

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

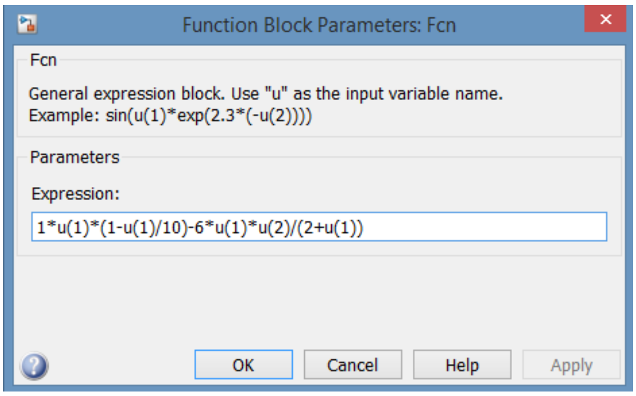
$$\begin{cases} \dot{x}(t) = f(x(t), y(t)) \\ \dot{y}(t) = g(x(t), y(t)) \end{cases}$$

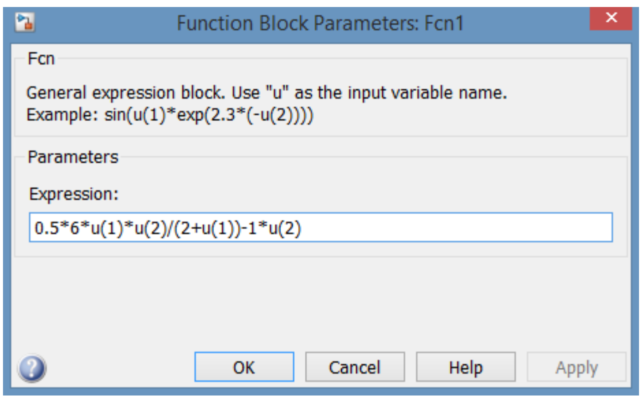
Pertanto basta definire due funzioni, che abbiano ognuna i due ingressi $x$ e $y$ e siano fatte come $f$ e $g$. Per costruire lo schema occorrono dei blocchi multiplexer (*mux* nella libreria *Signal Routing*) per generare gli ingressi delle funzioni; infatti il blocco *Fcn* accetta in ingresso un vettore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

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{bmatrix}$$

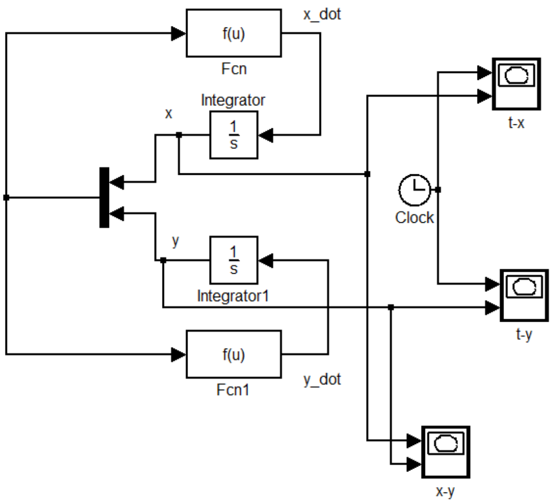
Nel nostro caso **u** deve contenere $x$ e $y$. Le variabili $x$ e $y$ sono inoltre ottenute integrando (tramite un integratore con funzione di trasferimento $1/s$) l'uscita dai blocchi *Fcn* (le derivate appunto di $x$ e $y$).

I blocchi *Fcn*, caratterizzanti le funzioni $f$ e $g$, sono rispettivamente così definiti:

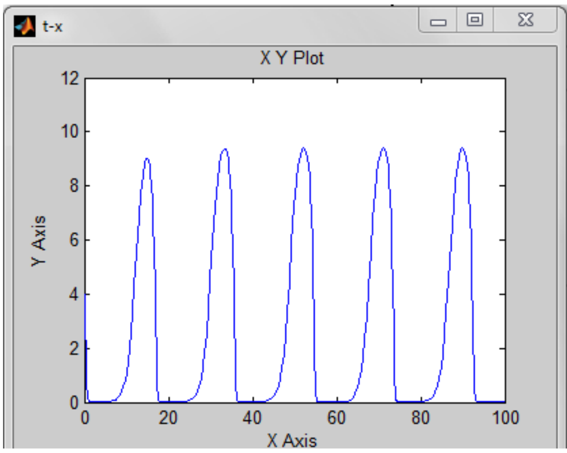


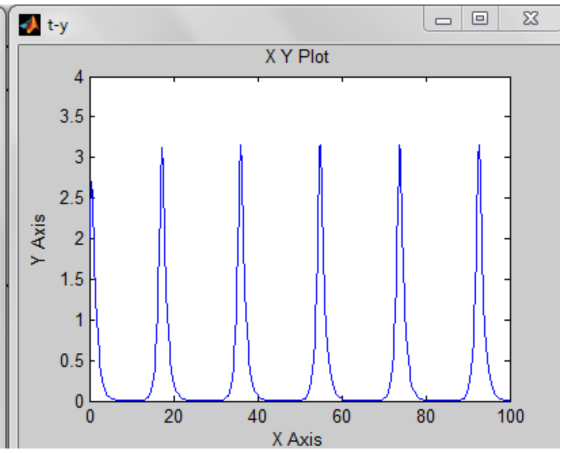



Quindi lo schema, completo anche dei grafici per osservare i risultati della simulazione risulta essere:

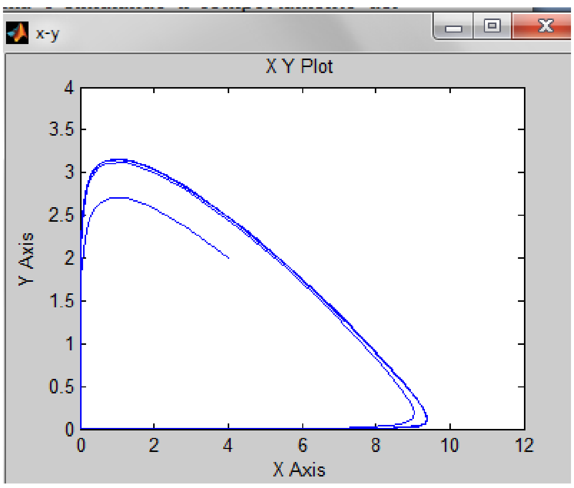



Simulando il comportamento del sistema si ottengono i seguenti risultati (la condizione iniziale fissata nei blocchi integratori è pari a $x = 4$ e $y = 2$):



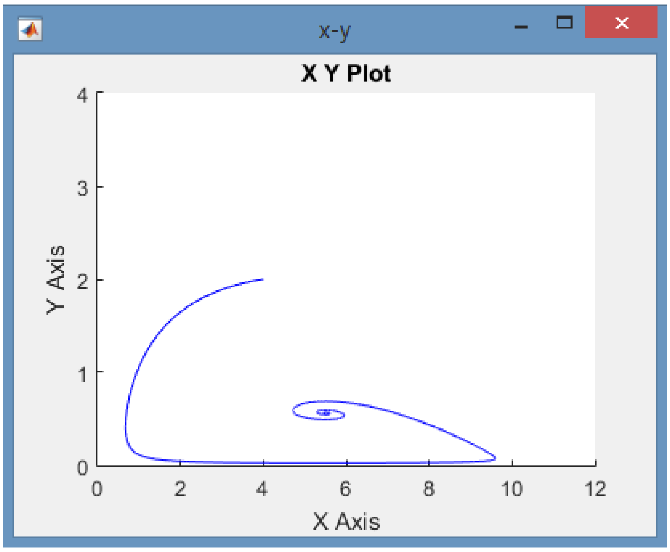




Nel piano di stato la traiettoria risulta essere:




Si noti come nasca un comportamento ciclico delle variabili $x$ e $y$.

Considerando ora $m$ = 2.2 (cambiare il termine di mortalità naturale nel blocco relativo alla funzione $g$) si ottiene la seguente traiettoria




Prede e predatori tendono ora verso un equilibrio di coesistenza.

Supponendo infine $m$ = 2.5 si ottiene la traiettoria seguente in cui il sistema tende verso un equilibrio i cui solo le prede sono presenti (alla capacità portsante k) mentre i predatori sono assenti.