

Es. 1

Scrivere una funzione che, assegnato un numero intero n in input, costruisca un vettore x contenente i punti x_1, \dots, x_n , equispaziati nell'intervallo $[0, 1]$ ($x_1=0, x_n=1$), e la matrice quadrata V di dimensione n la cui i -esima riga sia

$$\left[x_1^{i-1}, x_2^{i-1}, \dots, x_n^{i-1} \right], \quad i = 1, 2, \dots, n$$

La funzione deve restituire la matrice V ed il vettore x contenente i punti.

Scrivere quindi uno script che per ogni dimensione $n = 10, 11, 12, \dots, 80$ costruisca la matrice V utilizzando la funzione creata, memorizzi in tre vettori il numero di autovalori di V reali positivi, reali negativi o complessi (si usi la funzione `eig` per calcolarli). Lo script deve disegnare infine un unico grafico che riporti il contenuto dei tre vettori al variare di n , inserendo un titolo appropriato e una legenda per distinguere le linee.

Corpo della funzione

```
function [V,x] = creaVx(n)
```

```
x=linspace(0,1,n);
```

```
V=zeros(n);
```

Preallocare una matrice nulla velocizza l'esecuzione del codice

```
for i=1:n
```

```
V(i,:)=x.^(i-1);
```

Costruzione per righe della matrice mediante un ciclo FOR

```
end
```

```
end
```

Esempio di utilizzo della funzione

```
[V1,x1]=creaVx(5)
```

Decidiamo che i 3 vettori che saranno generati dallo script avranno come nomi: RP, RN ed IM

```
clc, close all, clear all
```

```
for n=10:80  
    [M,a]=creaVx(n)  
    AUT=eig(M)  
  
end
```

Ciclo FOR principale, che in corrispondenza dei valori di n richiesti valuta la matrice M utilizzando la function e ne calcola gli autovalori, che vengono salvati nel vettore AUT,

Ora vanno analizzati gli autovalori, ed in particolare debbono essere contati il numero di autovalori complessi, reali positivi e reali negativi. Tali numeri vanno inseriti, ad ogni passo del ciclo FOR, nelle successive componenti dei vettori RP, RN ed IM

NB in sede di scrittura e debug del codice, conviene limitare il ciclo FOR principale ad una sola iterazione: `for n=10`

```
clc, close all, clear all
```

```
for n=10:80  
    [M,a]=creaVx(n)  
    AUT=eig(M)
```

```
numrp=0;  
numrn=0;  
numim=0;
```

Definiamo 3 contatori. Questi contatori saranno reinizializzati a zero ad ogni istanza dei ciclo FOR

Ora dobbiamo analizzare (con un altro ciclo FOR) tutti gli elementi del vettore AUT, ed in funzione del fatto che il generico autovalore sia complesso, reale positivo, o reale negativo dovremo incrementare di 1 il relativo contatore.

```
end
```

```
clc, close all, clear all
```

```
for n=10:80  
    [M,a]=creaVx(n)  
    AUT=eig(M)
```

```
    numrp=0;  
    numrn=0;  
    numim=0;
```

Definiamo 3 contatori. Questi contatori saranno reinizializzati a zero ad ogni istanza dei ciclo FOR

```
    for i=1:n  
  
    end
```

Ora dobbiamo analizzare (con un altro ciclo FOR) tutti gli elementi del vettore AUT, ed in funzione del fatto che il generico autovalore sia complesso, reale positivo, o reale negativo dovremo incrementare di 1 il relativo contatore.

```
end
```

```
clc, close all, clear all
```

```
for n=10:80  
    [M,a]=creaVx(n)  
    AUT=eig(M)
```

```
    numrp=0;  
    numrn=0;  
    numim=0;
```

```
    for i=1:n
```

```
        if imag(AUT(i)) ~= 0  
            numim=numim+1;  
        elseif real(AUT(i))>0  
            numrp=numrp+1;  
        else if real(AUT(i))<0  
            numrn=numrn+1;  
        end  
    end
```

```
end
```

```
end
```

~ -> ALT+126

Mediante questa sintassi IF-ELSEIF-ELSE implementiamo l'analisi desiderata.

Se l'autovalore è complesso (cioè se la sua parte immaginaria è diversa da zero) si incrementa il contatore `numim` e si esce dal ciclo.

Se l'autovalore non risulta essere complesso si entra dentro la parte ELSEIF, e se la parte reale è maggiore di zero si incrementa il contatore `numrp`. In ultimo, si valuta se la parte reale è negativa e si incrementa il contatore `numrn`.

```
clc, close all, clear all
```

```
k=1;
```

```
for n=10:80
    [M,a]=creaVx(n)
    AUT=eig(M)

    numrp=0;
    numrn=0;
    numimag=0;

    for i=1:n
        if imag(AUT(i)) ~= 0
            numimag=numimag+1;
        elseif real(AUT(i))>0
            numrp=numrp+1;
        else if real(AUT(i))<0
            numrn=numrn+1;
        end
    end
end
```

```
RP(k)= numrp;
RN(k)=numrn;
IM(k)=numim;
k=k+1;
```

```
end
```

Ora i valori di numim, numrp e numrn vanno inseriti nei vettori IM, RP ed RN.

Serve un contatore che valga 1 alla prima iterazione del ciclo for, e venga incrementato di 1 ad ogni iterazione. Lo creiamo sotto forma della variabile k.

In questo modo, alla prima iterazione del ciclo FOR vengono scritte le prime componenti dei vettori RP, RN ed IM, alla seconda iterazione vengono scritte le seconde componenti, e così via.

```
Clc, clear all, close all
```

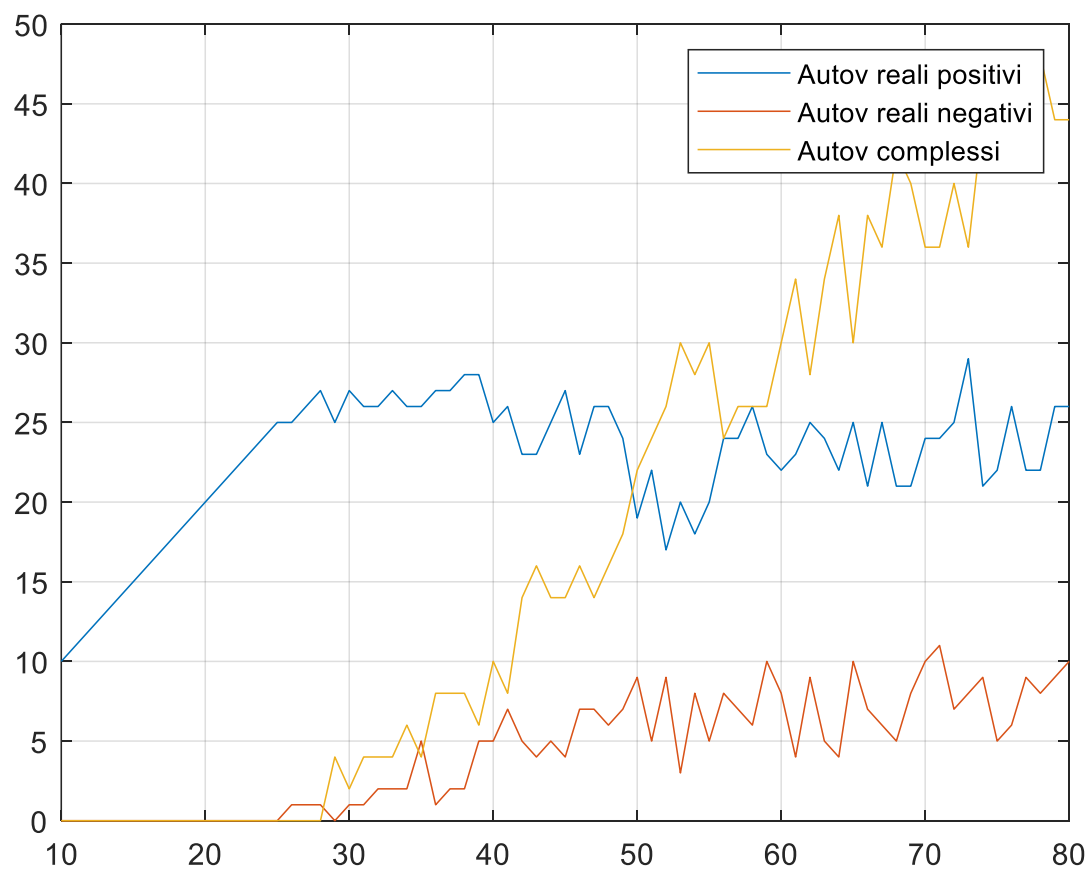
```
k=1;
for n=
    [M,a]=creaVx(n)
    AUT=eig(M)
    numrp=0;numrn=0;numimag=0;

    for i=1:n
        if imag(AUT(i)) ~= 0
            numimag=numimag+1;
        elseif real(AUT(i))>0
            numrp=numrp+1;
        else if real(AUT(i))<0
            numrn=numrn+1;
        end
    end
end
RP(k)= numrp;
RN(k)=numrn;
IM(k)=numimag;
k=k+1;
end
```

La parte «computazionale» dello script è completa, i vettori **riga** RP, RN ed IM contengono il numero di autovalori complessi, reali positivi e reali negativi al variare di n da 10 ad 80.

Ora creiamo il grafico.

```
%% creazione del grafico
plot(10:80,[RP' RN' IM']),grid
legend('Autov reali positivi','Autov reali negativi', 'Autov complessi')
```



Files:

Es1_script.m
creaVx.m

Es. 2

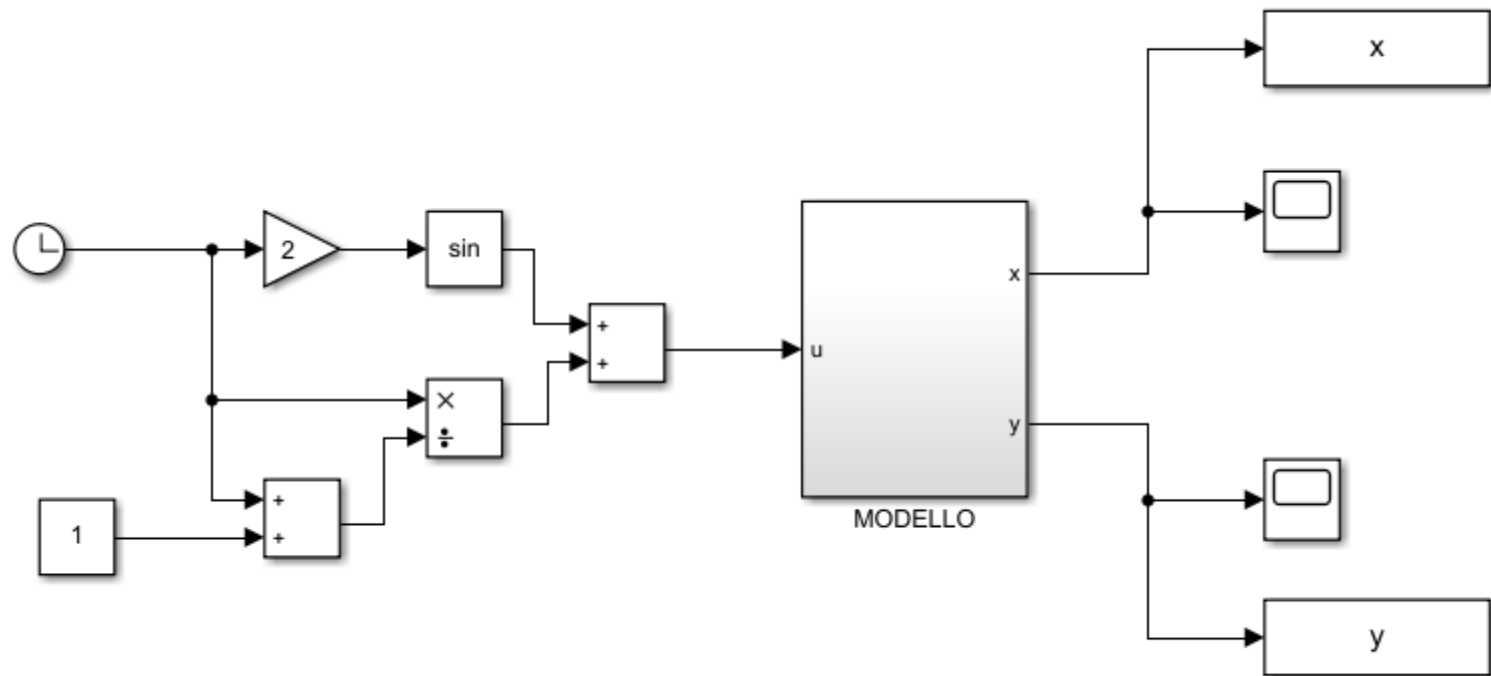
Si consideri il seguente sistema di equazioni differenziali

$$4\ddot{x} = -2x^3 - 3\dot{x} - \frac{\sin(y)}{|y|+1}$$

$$\dot{y} = -\frac{y^3}{1+y^2+x^2} - 3x - 4\sin(y) + u(t)$$

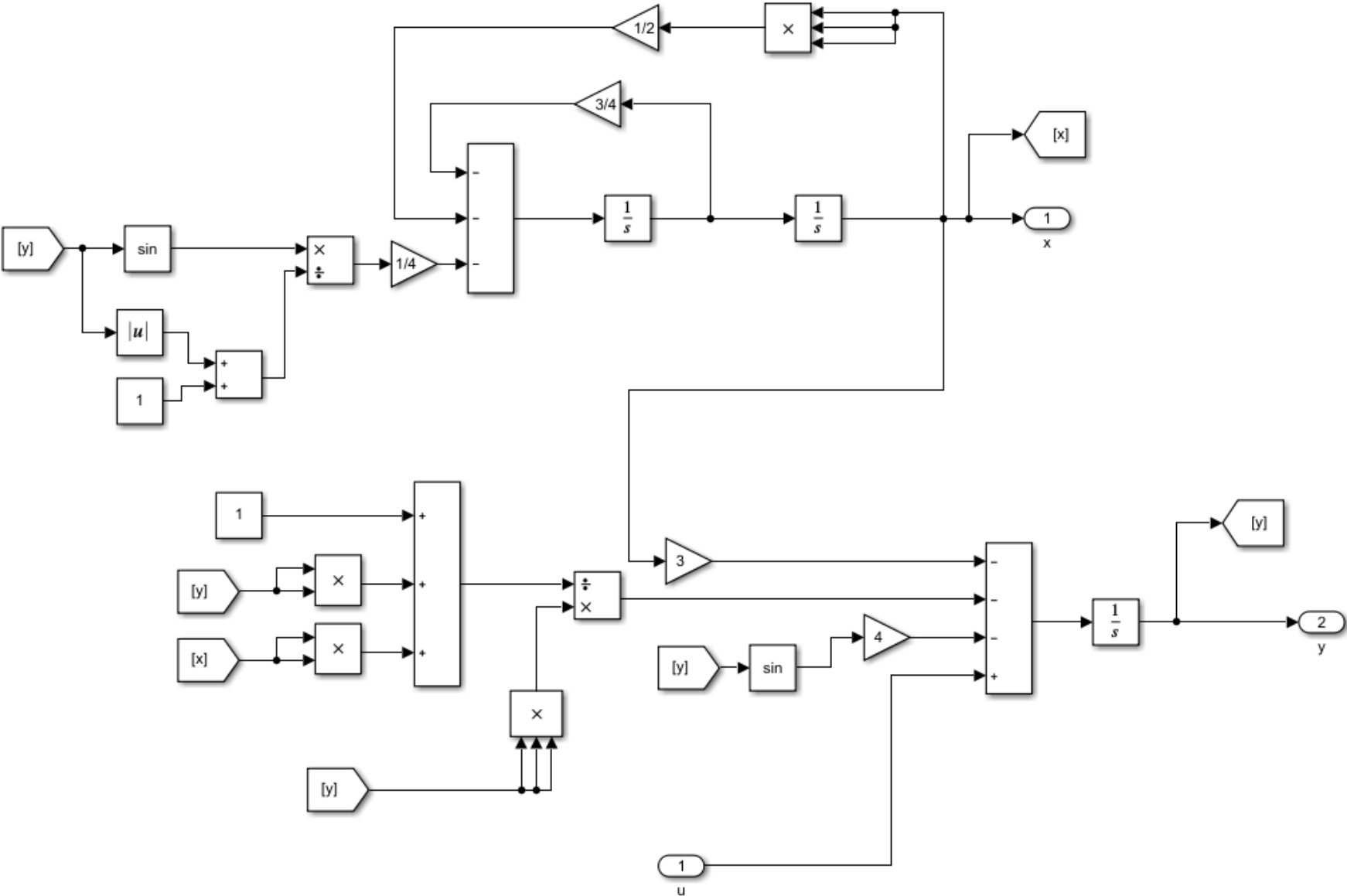


Si realizzi il modello Simulink mediante un *Subsystem* come in figura, e si valuti la soluzione per $t \in [0, 20]$ a partire dalle condizioni iniziali $x(0)=1$, $\dot{x}(0)=2$, $y(0)=-1$ in corrispondenza dell'ingresso applicato $u(t) = \sin(2t) + \frac{t}{1+t}$. Il modello Simulink deve esportare in Matlab i risultati della simulazione. Scrivere uno script che apra ed avvii il modello simulink (comando `sim`) e crei successivamente un grafico, dotato di opportune etichette e una legenda esplicativa di commento, che mostri sovrapposte le evoluzioni temporali dei segnali $x(t)$ e $y(t)$.



File: Es2_modello.slx

Contenuto del subsystem «MODELLO»

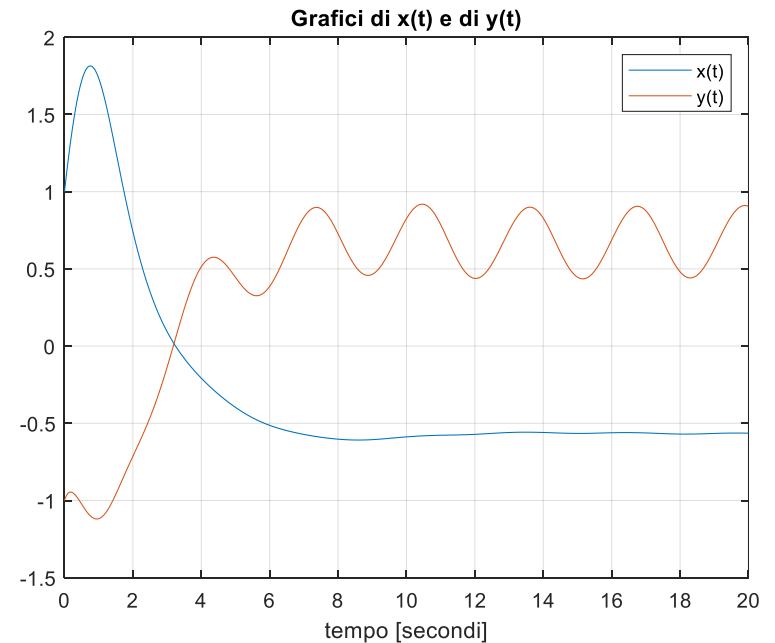


File: Es2_script.slx

```
clear all
close all
clc

open_system('Es2_modello')
sim('Es2_modello')

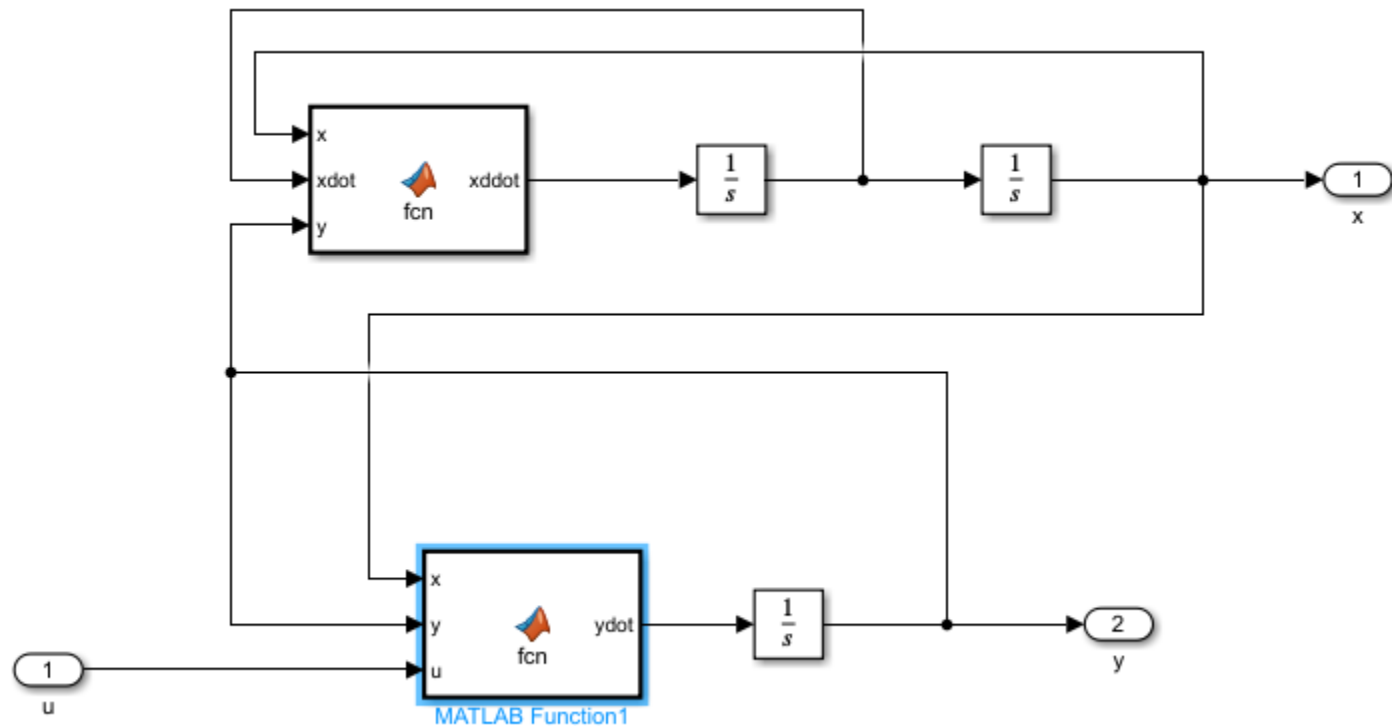
%% creazione grafico
figure
plot(x.time,x.data,y.time,y.data),grid
xlabel('tempo [secondi]')
title('Grafici di x(t) e di y(t)')
legend('x(t)', 'y(t)')
```



Seconda versione del modello

File: Es2_modello_v2.slx

Contenuto del subsystem «MODELLO»



```
function xddot = fcn(x,xdot,y)
```

```
xddot=-1/2*x^3-3/4*xdot-1/4*sin(y)/(abs(y)+1);
```

```
function ydot = fcn(x,y,u)
```

```
ydot=-y^3/(1+y^2+x^2)-3*x-4*sin(y)+u;
```