Laboratorio (Automatica)

Silvio Simani

Dipartimento di Ingegneria
Università di Ferrara
Tel. 0532 293844
Fax. 0532 768602

E-mail: ssimani@ing.unife.it

URL: http://www.ing.unife.it/~simani

URL: http://www.ing.unife.it/~simani/lessons.html

Automatica (Laboratorio)

Schema delle lezioni

1. Informazioni generali sul corso
2. Introduzione a Matlab

⇛ Simulazione di Sistemi Dinamici

3. Introduzione a Simulink
4. Osservatori e retroazione uscita-stato-ingresso
5. Modelli approssimati di sistemi dinamici
6. Progetto di Reti Correttrici
7. Identificazione di Sistemi Dinamici
8. Sintonizzazione di Controllori PID
9. Analisi di Sistemi a Dati Campionati

Università di Ferrara, Dip. di Ingegneria
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

Bibliografia

⇒ Dispense del Corso di Laboratorio di Automatica. Sergio Beghelli, Cesare Fantuzzi, Silvio Simani. (Fotocopisteria, tutorato, www)

⇒ Matlab, The Language of Technical Computing. Getting Started with MATLAB. *Version 5.1* (In formato pdf su CD Matlab)

⇒ MATLAB Primer. Second Edition. Kermit Sigmon. Department of Methematics. University of Florida.

⇒ The MathWorks Inc., Matlab User's Guide, 1993.

⇒ L. F. Shampine and M. W. Reichel, "The Matlab Ode Suite", Tech. Rep., The MathWorks, Inc, 1997. (Disponibile anche come file in formato pdf).

1. The MathWorks Inc., Simulink User's Guide, 1995.
2. B. C. Kuo, Automatic Control Systems. Englewood Cliffs, New Jersey: Prentice Hall, 7th ed., 1995.
3. P. Bolzern, R. Scattolini, and N. Schiavoni, Fondamenti di controlli automatici. Milano: McGraw- Hill, I ed., Marzo 1998.
4. G. Marro, TFI: insegnare e apprendere i controlli automatici di base con matlab. Bologna: Zanichelli, I ed., Ottobre 1998.
5. C. Fantuzzi, Controllori Standard PID. Versione 1.2, Appunti del Corso, 1a ed., Maggio 1997.

La funzione Q = obsv(A,C)

```
function Q = my_obsv(A,C)
%OBSV   Costruisce la matrice di
%       osservabilita'
%   Q = obsv(A,C)  ritorna la
%   matrice di osservabilita'
%   Q = [C; CA; CA^2; CA^(n-1)]
n = size(A,1);
Q = C;
for i=1:n-1
    Q = [C; Q*A];
end
return
```

```
function Q = my_obsv1(A,C)
%OBSV   Costruisce la matrice di
%       osservabilita'
%   Q = obsv(A,C)  ritorna la
%   matrice di osservabilita'
%   Q = [C; CA; CA^2; CA^(n-1)]
n = size(A,1);
Q = C;
for i=1:n-1,
    Q = [Q; C*A^i];
end
return
```

Università di Ferrara, Dip. di Ingegneria
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

La funzione `Q = obsv(A,C)`

`my_obsv.m` **funzione implementata da Matlab e computazionalmente più efficiente**

$\Rightarrow$ Per $A_{3\times3}$ e $C_{1\times3} \rightarrow$ flops $= 55$

`my_obsv1.m` **funzione non efficiente**

$\Rightarrow$ Per $A_{3\times3}$ e $C_{1\times3} \rightarrow$ flops $= 99$

Le funzioni `Q = obsv(A,C)` e `P = ctrb(A,B)`

- **Matrici di sistema ($A_{n\times n}$, $B_{n\times r}$, $C_{m\times n}$)**
- **Proprietà di dualità:**

$$\Rightarrow \quad \mathrm{P} = [\mathrm{B},\ \mathrm{A}*\mathrm{B},\ \ldots,\ \mathrm{A}^{(n-1)}*\mathrm{B}]$$

$$\Rightarrow \quad \mathrm{Q}^T = [\mathrm{C}^T,\ \mathrm{A}^T*\mathrm{C}^T,\ \ldots,\ \mathrm{A}^{T^{n-1}}*\mathrm{C}^T] = [\mathrm{C};\ \mathrm{C}*\mathrm{A};\ \ldots;\ \mathrm{C}*\mathrm{A}^{n-1}]$$

- **1)** `ctrb(A,B) = obsv(A',B')'`
- **2)** `obsv(A,C) = ctrb(A',C')'`

La funzione H = my_hankel(X,NrH,NcH,shift)

```
function H = my_hankel(X,NrH,NcH,shift)

%
% H = my_hankel(X,NrH,NcH,shift) costruisce
% la matrice di Hankel H di dimensione (NrH
% x NcH) a partire dall'elemento di indice
% 1+shift del vettore colonna X. X deve avere
% almeno NrH+NcH+shift-1 elementi con NrH, NcH
% e shift numeri naturali non nulli.
%

H = zeros(NrH,NcH);

for i=1:NcH,
   H(:,i) = X(shift+i:shift+NrH+i-1);
end;

return
```

Simulazione di Sistemi Dinamici

- **Analisi di Sistemi Lineari**

  $\Rightarrow$ Possibile la risoluzione analitica delle equazioni

- **Analisi di sistemi non lineari**

  $\Rightarrow$ Possibile la risoluzione numerica delle equazioni

  $\Rightarrow$ Analogie e Differenze

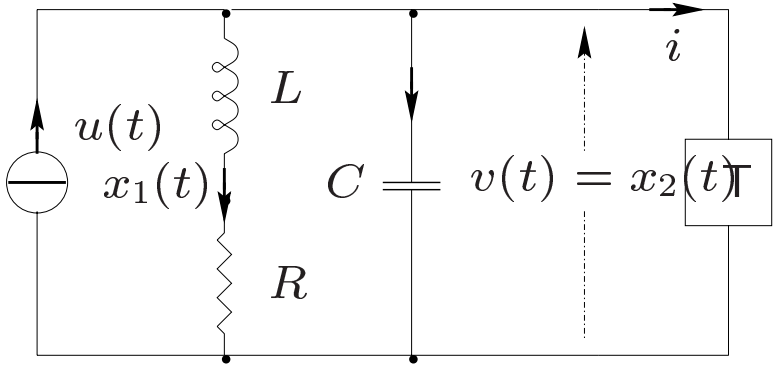
- **Analisi di un circuito non lineare**

- **Metodi numerici per l'integrazione**

- **Istruzioni di grafica in Matlab**

Analisi di un circuito non lineare

$$\begin{aligned}\dot{x}_1(t) &= -\tfrac{R}{L}\, x_1(t) + \tfrac{1}{L}\, x_2(t)\\ \dot{x}_2(t) &= -\tfrac{1}{C}\, x_1(t) + \\ &+\tfrac{1}{C}\big(G_1 x_2(t) - G_2 x_2^3(t)\big) + \tfrac{1}{C} u(t)\end{aligned}$$



ove

$$i = -G_1 * v + G_2 * v^3$$

con i seguenti valori

$G_1 = 0.8$, $G_2 = 0.05$, $R = 2$, $L = 1$ e $C = 1$.

Simulazione del circuito non lineare

La funzione `tunnel1.m`

```
function xd = tunnel1(t,x,flag,param)
% Funzione che implementa il circuito
%  non lineare.
%
% d x1(t) / dt = - R/L x1(t) + 1/L x2(t)
% d x2(t) / dt = - 1/C x1(t) + 1/C ( G1 x2(t)
%                - G2 x2(t)^3 ) + 1/C u(t)
%

R = param(1);
L = param(2);
C = param(3) ;
G1 = param(4);
G2 = param(5);

x1d = - (R/L) * x(1) + (1.0/L) * x(2);
x2d = - (1.0/C) * x(1) + (1.0/C)*( G1 * x(2)
                                 - G2 * x(2)^3 );

xd = [x1d; x2d];

return
```

Integrazione di Equazioni Differenziali

La funzione `ode45.m`...

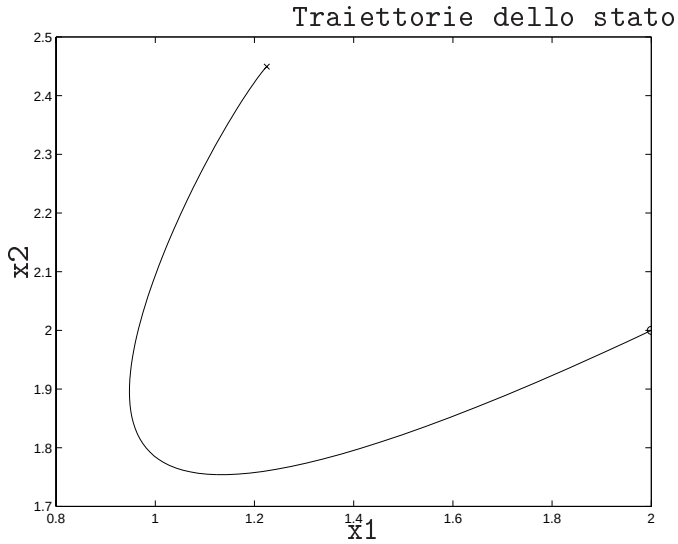
```
% Script-file che integra il sistema differenziale
% non lineare del diodo tunnel e grafica i risultati.

options = odeset('RelTol',1e-6); % Opzioni per
                                 % la funzione
                                 % di integrazione

% Parametri fisici del modello
R = 2; L = 1; C = 1; G1 = 0.8; G2 = 0.05;

param = [R,L,C,G1,G2]; % Parametri fisici del modello

ci = [2 2]; % Condizioni iniziali per l'integrazione

time = [0 40]; % Tempo di integrazione

[t,x] = ode45('tunnel1',time,ci,options,param);

            % Funzione che effettua l'integrazione
```

Integrazione di Equazioni Differenziali

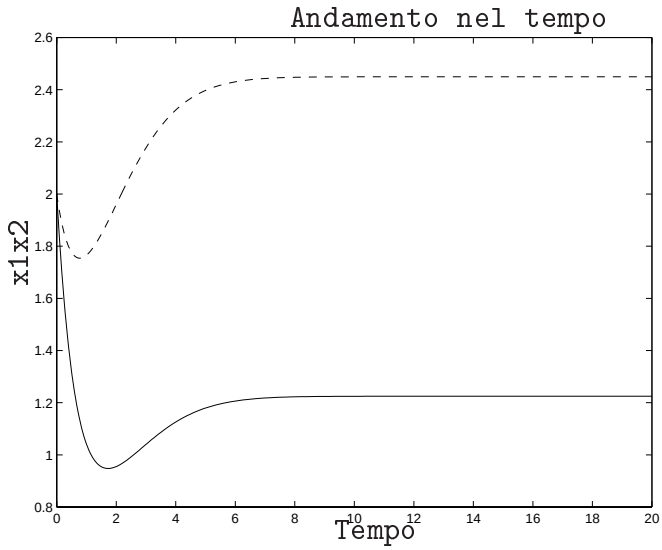
La funzione `ode45.m` (... continua)

```
%% Grafico delle traiettorie dello stato

figure

plot(x(:,1),x(:,2),'-') % Disegna i vettori passati
                        % come argomenti

title('Traiettorie dello stato') % Titolo del grafico
xlabel('x1') % Etichetta dell'asse delle ascisse
ylabel('x2') % Etichetta dell'asse delle ordinate

%% Grafico del moto dello stato

figure

plot(t,x(:,1),'-',t,x(:,2),'--')

title('Andamento nel tempo')
xlabel('Tempo')
ylabel('x1(t) e x2(t)')
```

Grafici dei Risultati: `tunnel1.m`



Traiettorie dello stato.



Andamento nel tempo delle variabili di stato.

Nuovi elementi di Matlab

File Ode e Ode Suite Solvers

⇒ Un *Ode file* è un file di tipo `.m` per definire un problema di equazioni differenziali che sono risolte dalle *Ode Suite Solvers*

`Y = odefile(T,Y0,FLAG,P1,P2,...)`

⇒ `T` e `Y0` sono variabili di integrazione

⇒ `FLAG` è una stringa che indica il tipo di informazione restituita dall' Ode file. `P1,P2,...` sono parametri addizionali richiesti.

Nuovi elementi di Matlab

- **File Ode e Ode Suite Solvers**
- **Ode Suite Solvers: risolutori di equazioni differenziali (sono funzionali)**
- `[T,Y] = ode45('odefile',TSPAN,YCI,options,P1,P2,...)`

  ⇒ ODE23, ODE113, ODE15S, ODE23S, ODE23T, ODE23TB

  ⇒ `options = odeset('RelTol',1e-4,'AbsTol',1e-4,'Maxstep',1e-5);`

  ⇒ `[TSPAN,YCI,options] = odefile([],[],'init')`

Nuovi elementi di Matlab

Integrazione *numerica* di equazioni differenziali ordinarie

$\Rightarrow$ `options = odeset('RelTol',1e-4,'AbsTol',1e-4,'MaxStep',10);`

$\Rightarrow$ `RelTol, AbsTol` : tolleranze relativa ed assoluta sull'errore ε per il controllo della convergenza

$$|\varepsilon| \leq \texttt{RelTol}|y| + \texttt{AbsTol}$$

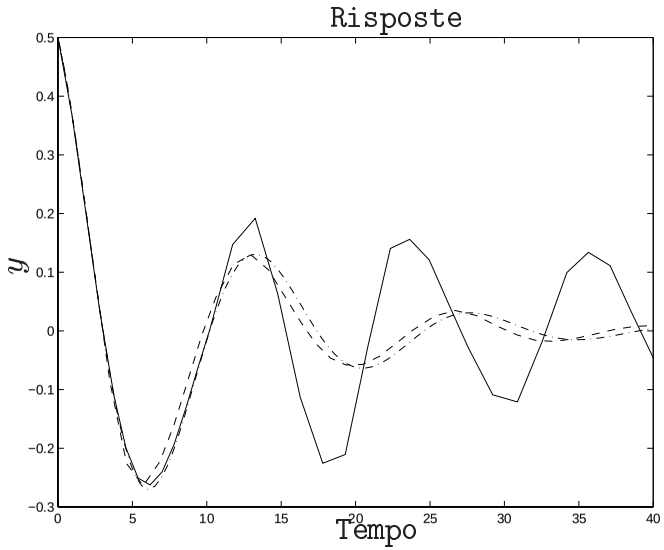
Errore ε, soluzione y

$\Rightarrow$ `maxStep`: limite superiore al passo di integrazione

Integrazione di Equazioni Differenziali: Problemi

⇒ Esempio del circuito `tunnel1.m` e `ode45`: tolleranze e passi di integrazione diversi.

- `'-':  RelTol = Abstol = 10e-1`
- `'--':  RelTol = Abstol = 10e-2`
- `'-.':  RelTol = Abstol = 10e-6`



Risposta del sistema con diversi passi di integrazione.

Simulazione del circuito non lineare

La funzione `tunnel2.m`

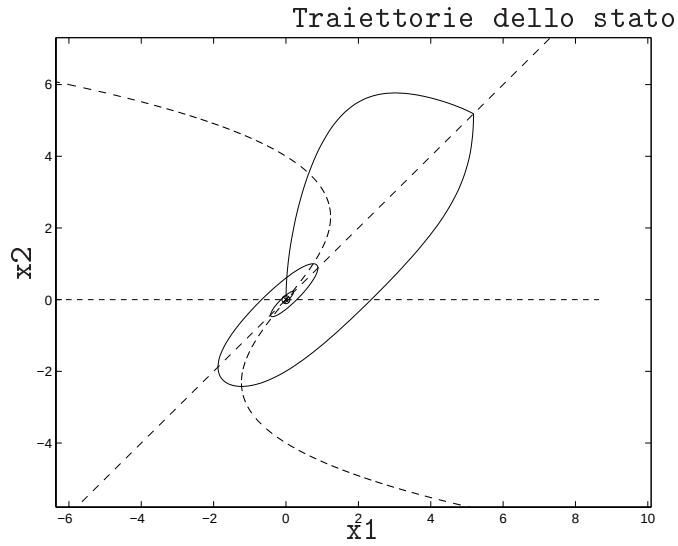
```
function xd = tunnel2(t,x,flag,param)
% Funzione che implementa il circuito di Eq.
% (2.1). E' presente anche una funzione del=
% =l'ingresso udt funzione del tempo.
%
% d x1(t) / dt = - R/L x1(t) + 1/L x2(t)
% d x2(t) / dt = - 1/C x1(t) + 1/C ( G1 x2(t)
%                - G2 x2(t)^3 ) + 1/C u(t)
%

R = param(1); % Parametri del circuito non
L = param(2); % lineare
C = param(3);
G1 = param(4);
G2 = param(5);
Tstart = param(6); % Istante di inizio del gradino
Tstop = param(7); % Istante finale del gradino
Value = param(8); % Ampiezza del gradino
```

Simulazione del circuito non lineare

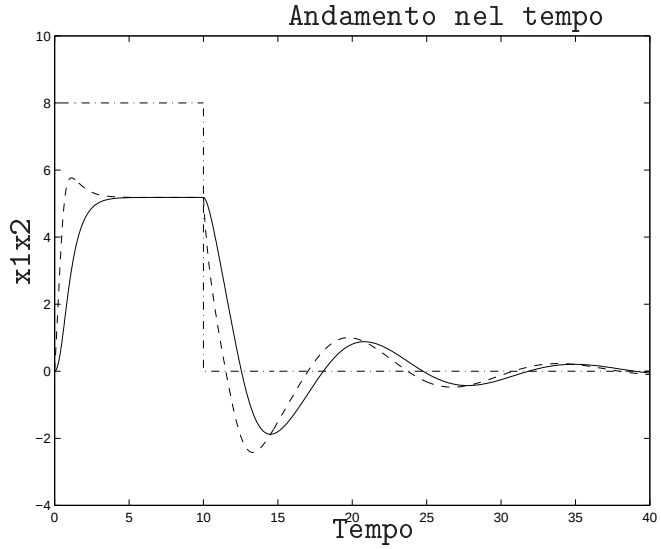
La funzione `tunnel2.m` (continua)

```
if((t>=Tstart)&(t<Tstop)),udt=Value; % Definizione
 else udt=0.0;                       % del gradino
end;

x1d = - (R/L) * x(1) + (1.0/L) * x(2);
x2d = - (1.0/C) * x(1) + (1.0/C)*( G1 * x(2) -
                                   G2 * x(2)^3 );

xd = [x1d; x2d + (1.0/C)*udt];

return
```

Grafici dei Risultati: `tunnel2.m`



Traiettorie dello stato e punto di equilibrio.



Andamento nel tempo delle variabili di stato.

Nuovi elementi di Matlab

Funzioni di grafica `plot(X,Y,S)`

⇒ Grafica il vettore `Y` in funzione di `X`

⇒ `X` e `Y` sono vettori con le stesso numero di elementi

⇒ S è una stringa formata da 3 caratteri:

y	yellow	.	point	-	solid
m	magenta	o	circle	:	dotted
c	cyan	x	x-mark	-.	dashdot
r	red	+	plus	--	dashed
g	green	*	star		
b	blue	s	square		
w	white	d	diamond		
k	black	v	triangle (down)		

Nuovi elementi di Matlab

```
figure

plot(x(:,1),x(:,2),'-')
title('Traiettorie dello stato')
xlabel('x1'), ylabel('x2')
```

- **Apertura finestra grafica:** `figure` $\rightarrow$ `figure(n)`
- **Visualizzazione grafici:** `plot()`
- **Titolo grafico:** `title()`
- **Etichette assi:** `xlabel(), ylabel()`

Linearizzazione di un sistema dinamico non lineare

$\Rightarrow$ Modello non lineare

$$\begin{cases} \dot{x}_1(t) &= f_1(x(t), u(t)) \\ \dot{x}_2(t) &= f_2(x(t), u(t)) \\ y(t) &= g(t) \end{cases}$$

$\Rightarrow$ Modello linearizzato

$$\begin{cases} \delta\dot{x}(t) &= A\delta x(t) + B\delta u(t) \\ \\ \delta y(t) &= C\delta x(t) \end{cases}$$

$\Rightarrow$ $\delta x(t)$, $\delta u(t)$ e $\delta y(t)$ gli scostamenti dai valori di equilibrio

Università di Ferrara, Dip. di Ingegneria
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

Linearizzazione di un sistema dinamico non lineare

$\Rightarrow$ Modello linearizzato

$$\begin{cases} \delta\dot{x}(t) &= A\delta x(t) + B\delta u(t) \\ \delta y(t) &= C\delta x(t) \end{cases}$$

$\Rightarrow$ Matrici calcolate nel punto di equilibrio

$$\mathrm{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}, \mathrm{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} \text{ e } \mathrm{C} = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} \end{bmatrix},$$

$\Rightarrow$ Linearizzazione del modello del circuito non lineare (`tunnel1.m`)

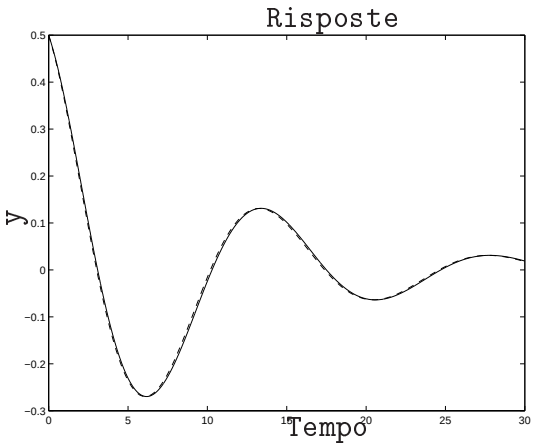
Linearizzazione del modello del circuito

```
%% Modello non lineare
options = odeset('RelTol',1e-6);

R = 1; % Parametri del circuito non lineare.
L = 1;
C = 1;
G1 = 0.8;
G2 = 0.05;

param = [R,L,C,G1,G2]; % Parametri del
                       % circuito non lineare.
ci = [0.5 0.5]; % Condizioni iniziali
ti = 0;
tf = 40;
time = [ti tf]; % Istante iniziale e
                % finale di integrazione

[t,x] = ode45('tunnel1',time,ci,options,param);
                         % Integrazione del sistema
y = x(:,2); % Uscita del sistema
```

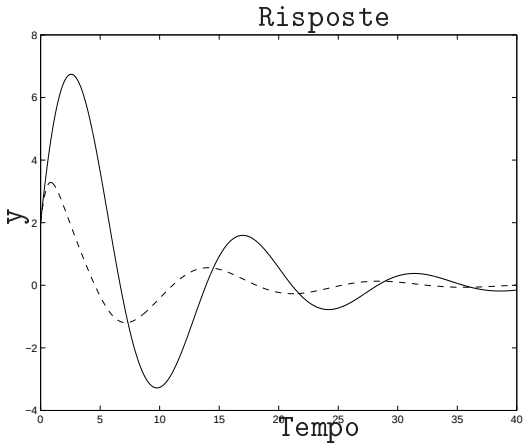
Linearizzazione del modello del circuito

```
%% Modello lineare nello spazio degli stati

A = [-R/L   1.0/L;
     -1.0/C G1/C ];
B = [1.0/C 0]';
C = [0 1];
D = 0;

Sys = ss(A,B,C,D); % Crea il modello nello spazio
                   % degli stati
tl = ti:0.01:tf;
Ul = zeros(size(tl));
[yl,tl,xl] = lsim(Sys,Ul,tl,ci);

figure, plot(tl,yl,'-',t,y,'--')
title('Risposte'), xlabel('Tempo'), ylabel('y')

figure, plot(x(:,1),x(:,2),'-'), hold on
plot(xl(:,1),xl(:,2),':'), hold on
title('Traiettorie dello stato'), xlabel('x1')
ylabel('x2')
```

Linearizzazione del circuito: proprietà



Risposta del sistema lineare e del sistema non lineare.



Risposta dei sistemi per una condizione iniziale distante dall'origine.

Simulazione di sistemi dinamici

La funzione `[Y,T,X] = lsim(SYS,U,T,X0)`

⇒ Simula la risposta nel tempo di un sistema LTI per ingressi arbitrari `U`

⇒ `SYS = ss(A,B,C,D)`

⇒ `U` vettore degli ingressi, `T` istanti di simulazione

⇒ `X0` condizioni iniziali della simulazione

⇒ `Y`, uscite del sistema: `length(T)` righe e `m` colonne

⇒ `X`: `length(T)` righe, `n` colonne

Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

$$\begin{cases} \dot{x}_1(t) = a_1(1 - x_1(t)/k)x_1(t) - a_2 x_1(t)x_2(t) + u(t) & \textit{Prede} \\ \dot{x}_2(t) = -a_3 x_2(t) + a_4 x_1(t)x_2(t) & \textit{Predatori} \end{cases}$$

$\Rightarrow$ $x_1(t)$ e $x_2(t)$ numero di prede di predatori

$\Rightarrow$ $u(t)$ cibo per le prede

$\Rightarrow$ k, numero massimo di prede in assenza di predatori e di cibo ($u(t) = 0$)

Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

$$\begin{cases} \dot{x}_1(t) = a_1(1 - x_1(t)/k)x_1(t) - a_2 x_1(t)x_2(t) + u(t) & \textit{Prede} \\ \dot{x}_2(t) = -a_3 x_2(t) + a_4 x_1(t)x_2(t) & \textit{Predatori} \end{cases}$$

$\Rightarrow$ a_3 (> 0), tasso di crescita del predatore

$\Rightarrow$ a_1 (> 0), tasso di crescita delle prede

$\Rightarrow$ $-a_2 x_1(t)x_2(t)$, decremento delle prede per la presenza dei predatori

$\Rightarrow$ $a_4 x_1(t)x_2(t)$, incremento dei predatori per la presenza delle prede

Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

$\Rightarrow$ Se $a_1 = 20$, $a_2 = 1$, $a_3 = 7$, $a_4 = 0.5$ e $k = 30$, si determinino:

1) l'andamento nel tempo del numero di prede e predatori, supponendo nullo l'ingresso u(t) e nelle ipotesi di partire da un ecositema contenente 10 prede e 10 predatori. Si calcoli anche la traiettoria percorsa dal sistema nello spazio degli stati.
2) gli stati di equilibrio del sistema in assenza di ingresso.
3) i valori di regime raggiunti dal numero di prede e predatori nelle ipotesi che $u(t)$ sia un gradino di ampiezza $u(t) = 20$ e a partire dalle stesse condizioni proposte al punto 1). Si determini per tentativi l'ampiezza del gradino che consente di mantenere a regime un numero di predatori pari a 15.

Esercizi Proposti (2, facoltativo)

Sistema ibrido

$$\dot{\mathbf{x}}(t) = \begin{cases} \mathrm{A}_1\mathbf{x}(t) & \text{se } x_1(t) * x_2(t) < 0 \\ \\ \mathrm{A}_2\mathbf{x}(t) & \text{se } x_1(t) * x_2(t) \geq 0 \end{cases}$$

$\Rightarrow \quad \mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$, durata della simulazione 10s

$\Rightarrow$ Condizioni iniziali $(1, 0)$, $(0, 1)$, $(10^{-6}, 10^{-6})$

Esercizi Proposti (2, facoltativo)

- **Date le matrici:**

$$\Rightarrow \quad \mathrm{A}_1 = \begin{bmatrix} -0.1 & 1.0 \\ -10.0 & -0.1 \end{bmatrix}, \mathrm{A}_2 = \begin{bmatrix} -0.1 & 10 \\ -1.0 & -0.1 \end{bmatrix}$$

- **Disegnare le traiettorie dello stato per i singoli sistemi e per quello ibrido, per le diverse condizioni iniziali**

- **Disegnare l'andamento dello stato nel tempo per i singoli sistemi e per quello ibrido, per le diverse condizioni iniziali**