

La funzione $P = \text{ctrb}(A,B)$: come da definizione

```
function P = ctrb(A,B)
%CTRB Form the controllability matrix.
%
% P = CTRB(A,B) returns the controllability matrix
% P = [B A*B A^2*B ...A^(n-1)*B].
% See also CTRBF.

n = size(A,1);
P = B;
for i=1:n-1,
    P = [P (A^i)*B];
end

return
% end ctrb
```



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

Silvio Simani

Automatica I (Laboratorio)

Silvio Simani

Dipartimento di Ingegneria
Università di Ferrara
Tel. 0532 97 4844
Fax. 0532 97 4870

E-mail: ssimani@ing.unife.it

URL: <http://www.ing.unife.it/simani>



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

Silvio Simani

La funzione $P = \text{ctrb}(A,B)$: ottimizzata

```
function P = ctrb(A,B)
%CTRB Form the controllability matrix.
%
% P = CTRB(A,B) returns the controllability matrix
% P = [B A*B A^2 * B ... A^(n-1) * B].
% See also CTRBF.

n = size(A,1);
P = B;
for i=1:n-1,
    P = [B A*P];
end
return
% end ctrb
```



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

Silvio Simani

Automatica (Laboratorio)



Schema delle lezioni

1. Informazioni generali sul corso
2. Introduzione a *Matlab*[®]
- ⇒ Simulazione di Sistemi Dinamici
3. Introduzione a *Simulink*[®]
4. Elementi di Controllo Digitale



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

Silvio Simani

Simulazione di Sistemi Dinamici



Analisi di Sistemi Lineari

⇒ Possibile la risoluzione analitica delle equazioni



Analisi di sistemi non lineari

⇒ Possibile la risoluzione numerica delle equazioni

⇒ Analogie e Differenze



Analisi di un circuito non lineare



Metodi numerici per l'integrazione



Istruzioni di grafica in Matlab



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

Silvio Simani

Analisi di un circuito non lineare

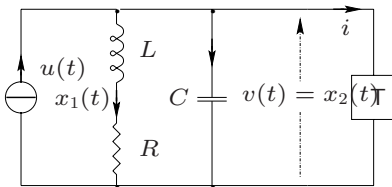
$$\begin{aligned}\dot{x}_1(t) &= -\frac{R}{L}x_1(t) + \frac{1}{L}x_2(t) \\ \dot{x}_2(t) &= -\frac{1}{C}x_1(t) + \frac{1}{C}u(t) \\ &\quad + \frac{1}{C}(G_1x_2(t) - G_2x_2^3(t))\end{aligned}$$

ove

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

con i seguenti valori

$$G_1 = 0.8, G_2 = 0.05, R = 2, L = 1 \text{ e } C = 1.$$



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

Silvio Simani

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
```



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

Silvio Simani

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
```



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

Silvio Simani

Integrazione di Equazioni Differenziali

↪ La funzione ode45.m (... continua)

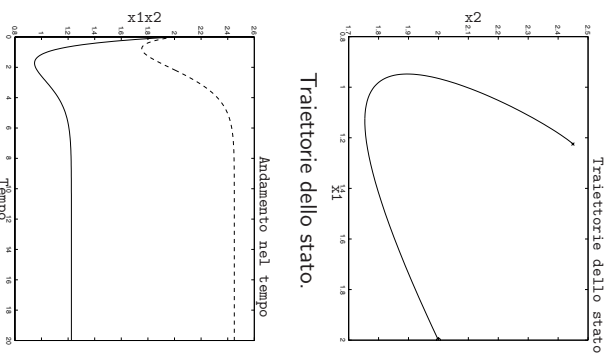
```
% Grafico delle traiettorie dello stato
figure
plot(x(:,1),x(:,2),'-') % Disegna i vettori passati
% come argomenti
title('Traiettorie dello stato') % Titolo del grafico
xlabel('x1') % Etichetta dell'asse delle ascisse
ylabel('x2') % Etichetta dell'asse delle ordinate
%% Grafico del moto dello stato
figure
plot(t,x(:,1),'-',t,x(:,2),'-')
title('Andamento nel tempo')
xlabel('Tempo')
ylabel('x1(t) e x2(t)')
```



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

Silvio Simani

Grafici dei Risultati: tunnel1.m



Andamento nel tempo delle variabili di stato.



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

Silvio Simani

Elementi di Matlab: Ode Suite

↪ 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*

↪ $Y = \text{odefile}(T, Y0, \text{FLAG}, P1, P2, \dots)$

⇒ 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 aggiuntivi richiesti.



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

Silvio Simani

Elementi di Matlab: Ode Suite

↪ File Ode e Ode Suite Solvers

↪ **Ode Suite Solvers: risolutori di equazioni differenziali (sono funzionali)**

↪ $[T, Y] = \text{ode45}(\text{'odefile'}, \text{TSPAN}, \text{YCI}, \text{options}, P1, P2, \dots)$

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

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

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



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

Silvio Simani

Elementi di Matlab: *Ode Suite*



Integrazione numerica di equazioni differenziali ordinarie

⇒ options = odeset('RelTol',1e-4,'AbsTol',1e-4,'MaxStep',10);

⇒ RelTol, AbsTol : tolleranze relativa ed assoluta sull'errore ε per il controllo della convergenza

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



Errore ε , soluzione y

⇒ maxStep: limite superiore al passo di integrazione



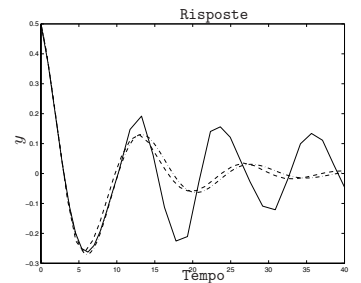
Università di Ferrara, Dip. di Ingegneria
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

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.



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

Silvio Simani

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=
% =1'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
```



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

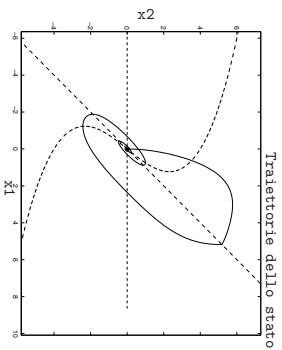
Silvio Simani



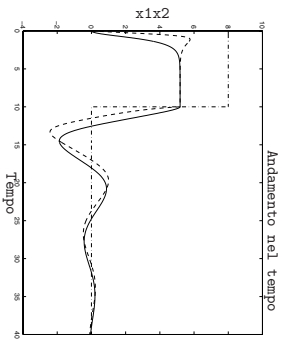
Università di Ferrara, Dip. di Ingegneria
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

Grafici dei Risultati: tunnel12.m



Traiettorie dello stato e punto di equilibrio.



Andamento nel tempo delle variabili di stato.



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

Silvio Simani

Elementi Grafici di Matlab



Funzioni di grafica plot(X, Y, S)

⇒ Grafica il vettore Y in funzione di X

⇒ X e Y sono vettori con lo 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)		



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

Silvio Simani

Elementi Grafici di Matlab

figure

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



Apertura finestra grafica: figure → figure(n)



Visualizzazione grafici: plot()



Titolo grafico: title()



Etichette assi: xlabel(), ylabel()



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

Silvio Simani

Linearizzazione di un sistema dinamico non lineare

⇒ Modello non lineare

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

⇒ Modello linearizzato

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

⇒ $\delta x(t)$, $\delta u(t)$ 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

⇒ Modello linearizzato

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

⇒ Matrici calcolate nel punto di equilibrio

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

⇒ 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;
```

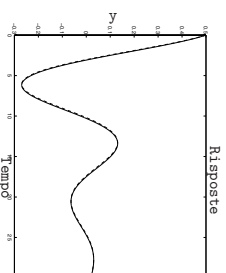
```
Sys = ss(A,B,C,D); % Crea il modello nello spazio
% degli stati
```

```
t1 = ti:0.01:tf;
U1 = zeros(size(t1));
[y1,t1,x1] = lsim(Sys,U1,t1,ci);
```

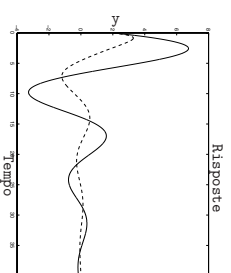
```
figure, plot(t1,y1,'-',t,y,'--')
title('Risposte'), xlabel('Tempo'), ylabel('y')
```

```
figure, plot(x(:,1),x(:,2),'-'), hold on
plot(x1(:,1),x1(:,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.



Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

$$\begin{cases} \dot{x}_1(t) = a_1(1 - x_1(t)/k)x_1(t) - a_2x_1(t)x_2(t) + u(t) & \text{Prede} \\ \dot{x}_2(t) = -a_3x_2(t) + a_4x_1(t)x_2(t) & \text{Predatori} \end{cases}$$

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

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

⇒ $-a_2x_1(t)x_2(t)$, decremento delle prede per la presenza dei predatori

⇒ $a_4x_1(t)x_2(t)$, incremento dei predatori per la presenza delle prede



Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

⇒ 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 ecosistema 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.



Simulazione di sistemi dinamici

La funzione $[Y, T, X] = \text{lsim}(\text{SYS}, U, T, X0)$

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

⇒ $\text{SYS} = \text{ss}(A, B, C, D)$

⇒ U vettore degli ingressi, T istanti di simulazione

⇒ $X0$ condizioni iniziali della simulazione

⇒ Y , uscite del sistema: $\text{length}(T)$ righe e m colonne

⇒ X : $\text{length}(T)$ righe, n colonne



Esercizi Proposti (1)

Modello matematico di Lotka-Volterra

$$\begin{cases} \dot{x}_1(t) = a_1(1 - x_1(t)/k)x_1(t) - a_2x_1(t)x_2(t) + u(t) & \text{Prede} \\ \dot{x}_2(t) = -a_3x_2(t) + a_4x_1(t)x_2(t) & \text{Predatori} \end{cases}$$

⇒ $x_1(t)$ e $x_2(t)$ numero di prede di predatori

⇒ $u(t)$ cibo per le prede

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



Esercizi Proposti (2a)

⇒ **Sistema ibrido**

$$\dot{\mathbf{x}}(t) = \begin{cases} A_1 \mathbf{x}(t) & \text{se } x_1(t) * x_2(t) < 0 \\ A_2 \mathbf{x}(t) & \text{se } x_1(t) * x_2(t) \geq 0 \end{cases}$$

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

⇒ Condizioni iniziali (1, 0), (0, 1), (10⁻⁶, 10⁻⁶)

**Esercizi Proposti (2b)**

⇒ **Date le matrici:**

$$\Rightarrow A_1 = \begin{bmatrix} -0.1 & 1.0 \\ -10.0 & -0.1 \end{bmatrix}, A_2 = \begin{bmatrix} -0.1 & 10 \\ -1.0 & -0.1 \end{bmatrix}$$

⇒ **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**

