

## Integrazione in Matlab

### ➤ Ode Suite Solvers

### ➤ Ode Suite Solvers: risolutori di equazioni differenziali (sono funzioni di funzione)

➤ `[t,x] = odesolver('odefile',tspan,ci,options,params)`

⇒ (Esempio:  $\int_{ci,tspan} F(t, x, \theta) dt$ )

⇒ `ode23, ode113, ode15s, ode23s, ode23t, ode23tb`

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

➤ `[tspan,ci,options] = odefile([],[],'init')`



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

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)		



### 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 dai caratteri:

## Elementi di grafica in Matlab

## 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](mailto:ssimani@ing.unife.it)

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

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



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

Silvio Simani

## Integrazione in Matlab



### File Ode

⇒ Un *Ode file* è un file di tipo .m per definire un sistema di equazioni differenziali

⇒ Un *Ode file* è risolto dagli *Ode Suite Solvers*



`y = odefile(t,x,flag,params)`. (Esempio:  $y = \dot{x} = F(t, x, \theta)$ )

⇒ t e x sono variabili di integrazione

⇒ flag è una stringa che indica il tipo di informazione restituita dall' *Ode file*. params sono parametri aggiuntivi eventualmente richiesti.



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

Silvio Simani

**Esercizi Proposti (1)**

⇒ **Modello matematico di Lotka-Volterra**

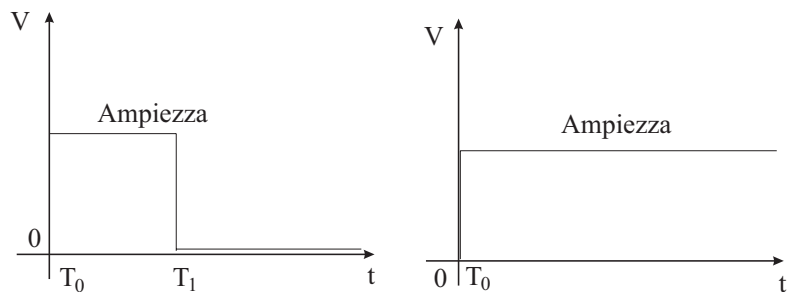
⇒ Se  $a_1 = 20$ ,  $a_2 = 1$ ,  $a_3 = 7$ ,  $a_4 = 0.5$  e  $k = 30$ , si determinino:

- 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.



**Esercizi Proposti**

⇒ **Modello matematico di Lotka-Volterra**



(a) Circuito diodo tunnel

(b) Modello Lotka-Volterra



**Elementi di Grafica in Matlab**

figure, plot(t,x,'-',t,y,'--')

⇒ **Apertura nuova finestra grafica e visualizzazione grafici sovrapposti**

figure, plot(t,x,'-'), hold on, plot(t,y,'--')

⇒ **Apertura nuova finestra grafica e visualizzazione grafici sovrapposti**

⇒ Sono equivalenti: hold on mantiene il grafico corrente



**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 ( $u(t) = 0$ )



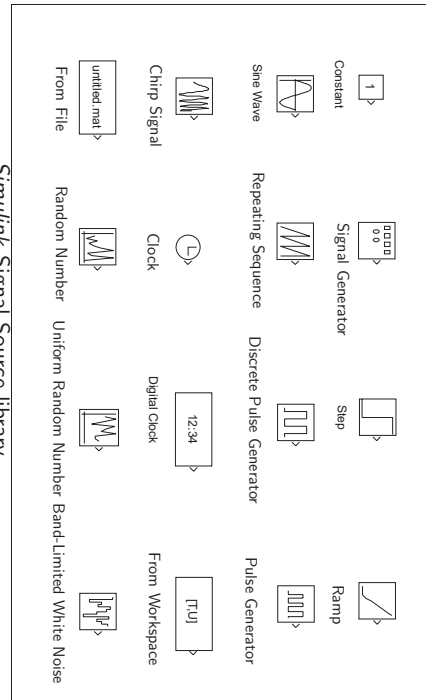
### Introduzione a Simulink

- ⇒ *Simulink*, prodotto dalla *Mathworks Inc.*
- ⇒ È un programma per la simulazione di sistemi dinamici
- ⇒ Progetto di un sistema dinamico
  - ⇒ Definizione del modello da simulare
  - ⇒ Analisi del sistema
- ⇒ Ambiente a finestre ed interfaccia grafica: *Block diagram windows e mouse*
- ⇒ *Simulink* riutilizza l'ambiente e i comandi di *Matlab*
  - ⇒ I risultati sono disponibili nel *Workspace* di *Matlab*

### Istruzioni di base di Simulink

#### Sources Library

⇒ Sources: (Library: simulink/Sources),  
generatori di segnale



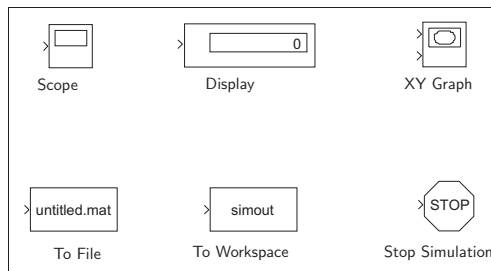
Simulink Signal Source library.



### Istruzioni di base di Simulink

#### Sink Library

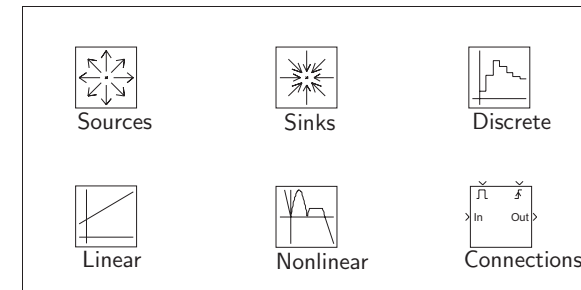
⇒ Library: *simulink/Sinks* contiene alcuni rivelatori di segnale



Simulink Signal Sinks library.

### Istruzioni di base di Simulink

- ⇒ >> `simulink`
- ⇒ Visualizzazione della finestra *Simulink block library*



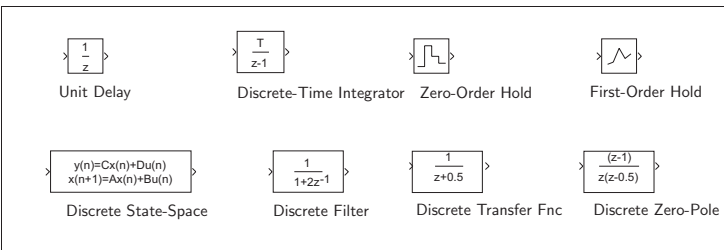
Simulink block library.



## Istruzioni di Simulink

### Discrete Library

⇒ Library: *simulink/Discrete* analisi dei sistemi lineari tempo-discreti



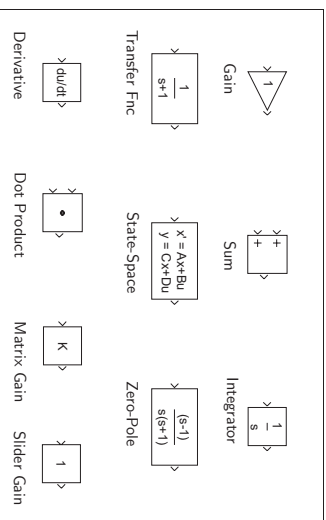
Simulink Discrete-Time library.



## Istruzioni di Simulink

### Linear Library

⇒ Library: *simulink/Linear* analisi dei sistemi lineari tempo-continui



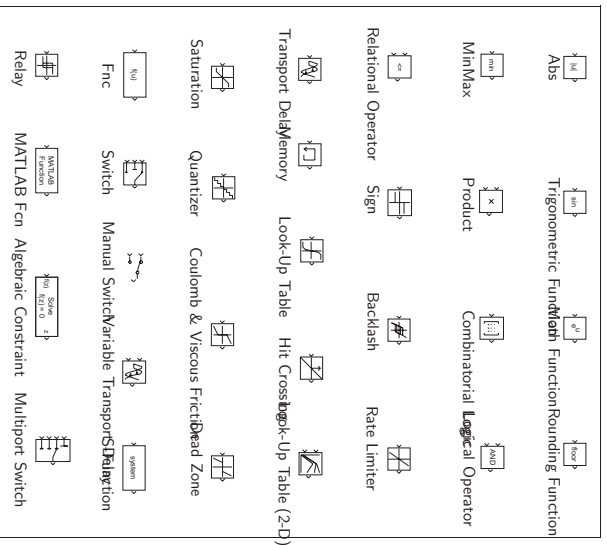
Simulink Linear library.



## Istruzioni di Simulink

### Nonlinear Library

⇒ Library: *simulink/Nonlinear* funzioni non lineari



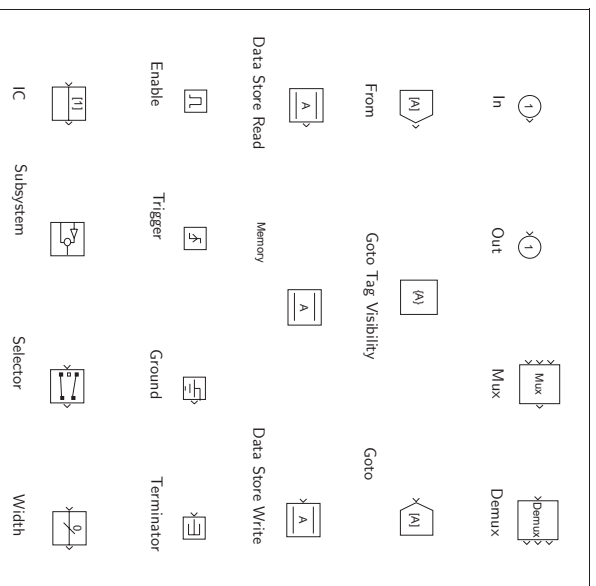
Simulink Nonlinear library.



## Istruzioni di Simulink

### Connections Library

⇒ Library: *simulink/Connections* blocchi per effettuare connessioni



Simulink Connection library.



**Simulazione di sistemi dinamici con SIMULINK**

- ⇒ Simulazione → integrazione delle equazioni differenziali
- ⇒ Simulatore di Simulink
- ⇒ Simulink Control Panel ↔ Simulation ↔ Parameters
- ⇒ Opzione Solver nella finestra Simulation Parameters (cfr. odesolver Matlab)
- ⇒ Simulation time: Start time: istante iniziale della simulazione
- ⇒ Simulation time: Stop time: istante finale della simulazione
- ⇒ Solver Options: Type: passo di integrazione fisso (Fixed-step) o variabile (Variable-step)
- ⇒ Solver Options: scelta della funzione di integrazione ottimale: ode45, ode23, ode113, ode15s, ode23s, e discrete
- ⇒ Solver Options: Max step size e Initial step size, Relative e Absolute tolerance

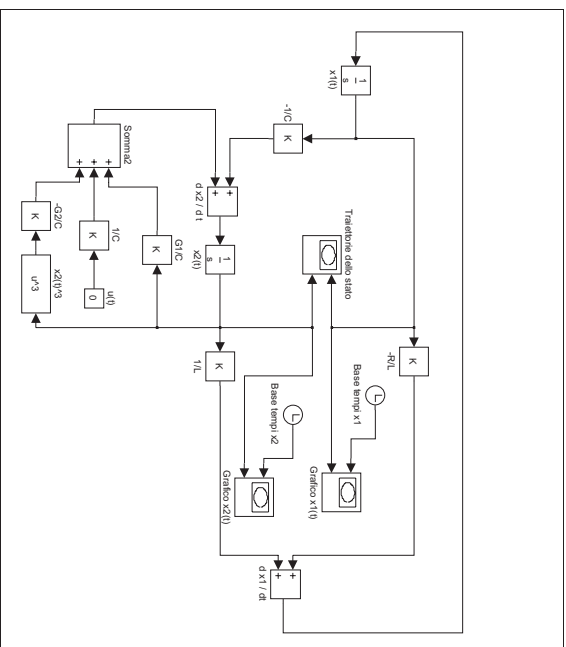


Università di Ferrara, Dip. di Ingegneria  
v. Saragat, 1, I-44100, Ferrara, Italia

Silvio Simani

**Analisi di un circuito non lineare**

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



Circuito non lineare in Simulink.



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

Silvio Simani

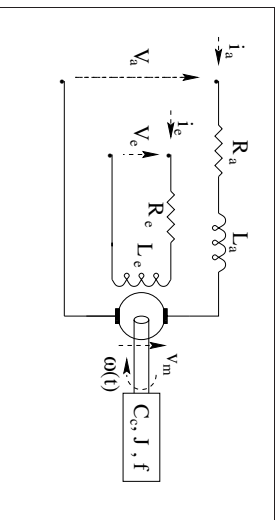
**Modello di un motore in corrente continua**

⇒ Controllo d'armatura e avvolgimento d'eccitazione ad alimentazione costante

$$\begin{bmatrix} \dot{i}_a(t) \\ \dot{\omega}(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{J} \\ \frac{k_m}{J} & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix}$$

$$\omega(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix}$$

⇒ Motore in corrente continua



Schema di un motore in corrente continua



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

Silvio Simani

**Modello di un motore in corrente continua**

⇒ Matrici di sistema

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{1}{J} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

⇒ Ingressi ed uscite:  $y(t) = \omega(t)$

$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} \text{ e } u(t) = \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix}.$$



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

Silvio Simani

## Modello di un motore in corrente continua



### Condizioni di funzionamento

1. Alimentazione del motore con un gradino di tensione di armatura costante a 0V da 0s a 50s e al valore di 5V per altri 50s.
2. Impulso di tensione di ampiezza  $V_a(t) = 10V$  e durata  $\tau = 40s$ .
3. Posizione del rotore  $\alpha(t)$

$$\dot{\alpha}(t) = \omega(t), \quad x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \\ \alpha(t) \end{bmatrix} \quad \text{e} \quad y(t) = \begin{bmatrix} \omega(t) \\ \alpha(t) \end{bmatrix}$$

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} & 0 \\ \frac{k_m}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{bmatrix} \quad \text{e} \quad C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$



## Esercizi proposti

1. Si realizzi in ambiente *Simulink* il sistema di equazioni differenziali relative al modello del motore in corrente continua. Se ne verifichi successivamente la correttezza confrontandolo con le realizzazioni equivalenti nello spazio degli stati.
2. Utilizzando gli stessi valori dei parametri del motore in corrente continua, determinare l'ampiezza del gradino  $V_a(t)$  necessaria a raggiungere una velocità angolare *di regime* pari a  $\omega(t) = 10\text{rad/s}$ , nelle ipotesi di assenza del carico  $C_c = 0$  e con il modello del motore del secondo ordine. Si verifichi analiticamente il risultato ottenuto.
3. Fissata l'ampiezza della tensione di armatura a  $V_a = 10V$ ., progettare la durata dell'impulso  $\tau$  in modo da raggiungere una posizione assegnata  $\alpha = 20\text{rad}$ ., sempre nelle ipotesi di assenza di carico.
4. Fissato  $\tau$ , graficare l'andamento temporale della posizione del rotore per una tensione pari alla metà e al doppio della tensione fissata al punto precedente.

