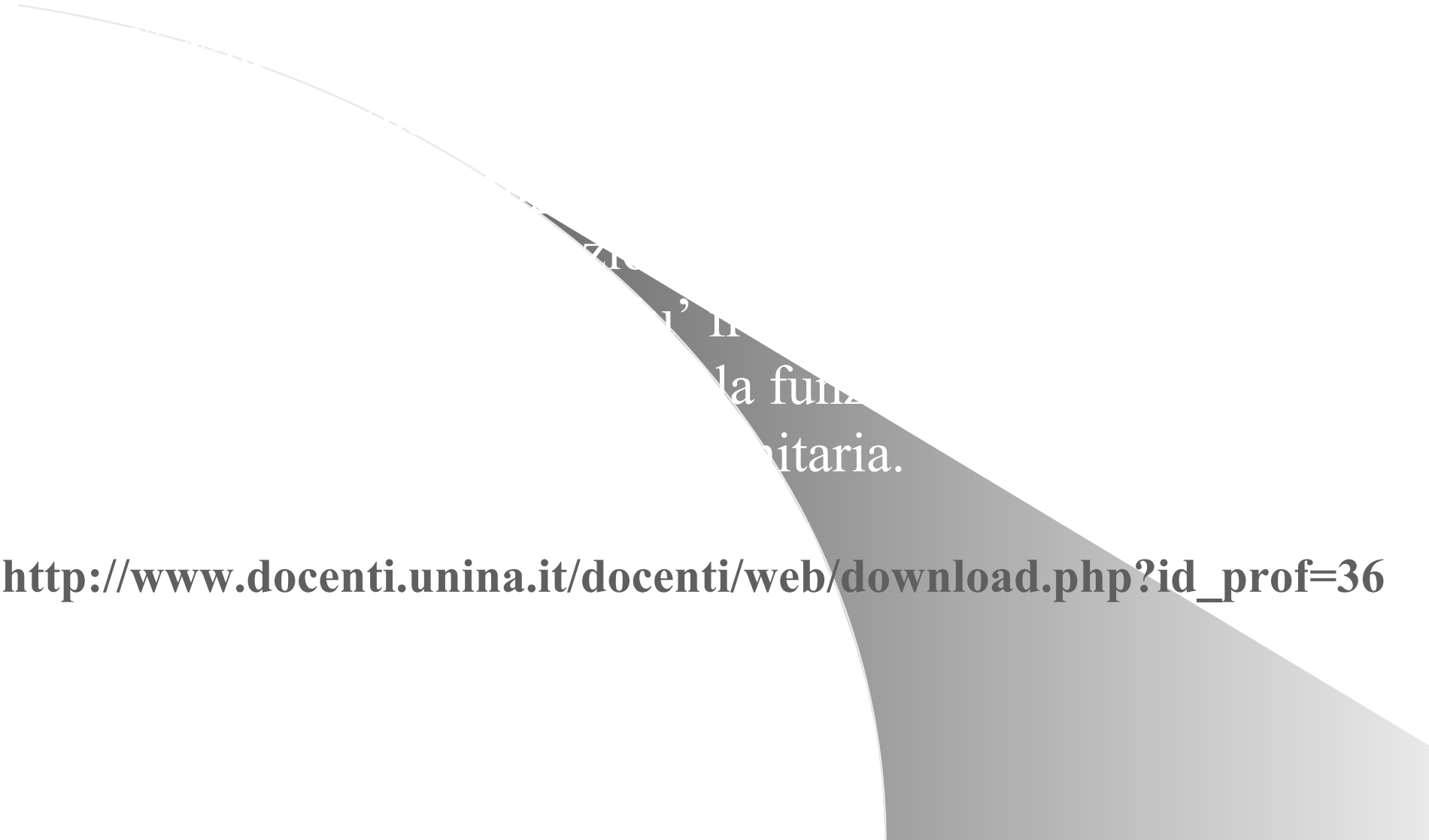
Funzioni di Ingresso canoniche
Impulso Scalino Rampa Unitaria



http://www.docenti.unina.it/docenti/web/download.php?id_prof=36

Una funzione scalino unitario e' una funzione del tempo indicata come $u(t-t_0)$ e definita da:

$$u(t-t_0)=\begin{cases}1 & t>t_0\\ 0 & t\le t_0\end{cases}$$

La funzione rampa unitaria e' una funzione del tempo, integrale della funzione scalino unitario, definita da:

$$\int_{-\infty}^{t} u(\tau-t_0)d\tau=\begin{cases}t-t_0 & t>t_0\\ 0 & t\le t_0\end{cases}$$

La funzione impulso unitario $\delta(t)$ si definisce come: $\lim_{\Delta t\to 0^+}\left[\frac{u(t)-u(t-\Delta t)}{\Delta t}\right]$ essendo u(t) la funzione scalino unitario.

L'operazione di limite da' luogo ad una funzione la cui ampiezza tende all'infinito e la cui durata tende a zero.

L'area sottesa alla curva e' uguale ad 1 per ogni valore di Δt.

$$\int_{-\infty}^{+\infty}\delta(t)dt=1$$

proprieta' di screening

$$\int_{-\infty}^{+\infty} f(t)\delta(t-t_0)dt = f(t_0)$$

Per risposta all'impulso unitario di un sistema lineare si intende l'uscita y(t) del sistema quando l'ingresso e' $x(t)=\delta(t)$ e tutte le condizioni iniziali sono nulle.

Esempio. Se la relazione ingresso-uscita di un sistema lineare e' data dall'integrale di convoluzione: $y(t) = \int_0^t w(t-\tau)x(\tau)d\tau$

ed $x(\tau)$ e' un impulso unitario applicato all'ingresso del sistema la corrispondente risposta $y_\delta(t)$ e' data da:

$$y_\delta(t) = \int_0^t w(t-\tau)\delta(\tau)d\tau = \int_{-\infty}^{+\infty} w(t-\tau)\delta(\tau)d\tau = w(t)$$

essendo $w(t-\tau)=0$ per $t > \tau$, $\delta(\tau)=0$ per $\tau < 0$ e la proprietà di screening dell'integrale e' stata applicata per il calcolo dell'integrale.

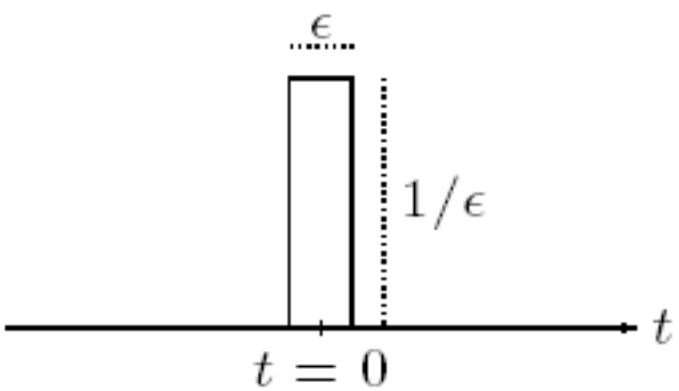
Segnali Impulsivi: SINTESI

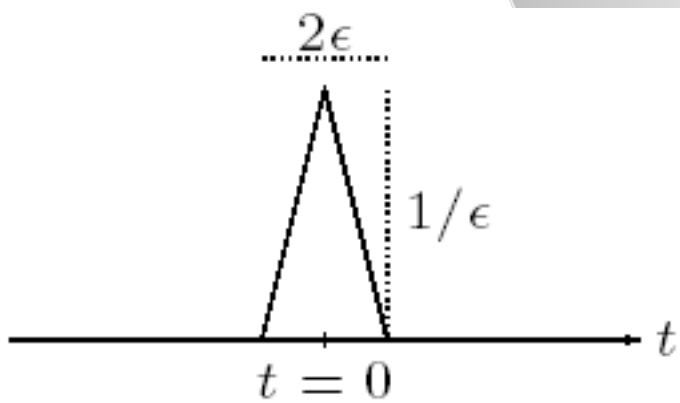
La funzione delta (di Dirac) o **impulso** δ é una *idealizzazione* di un segnale che

- è molto grande per *t quasi* = 0
- è molto piccola lontano da $t = 0$
- ha integrale = 1

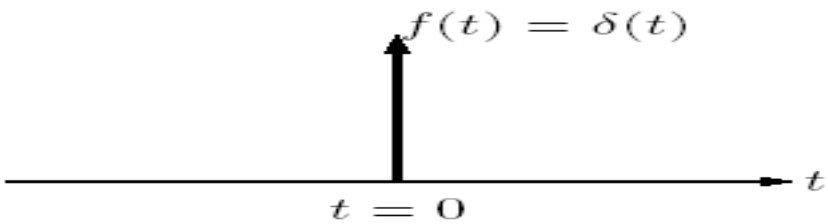
La forma esatta della funzione non ha importanza

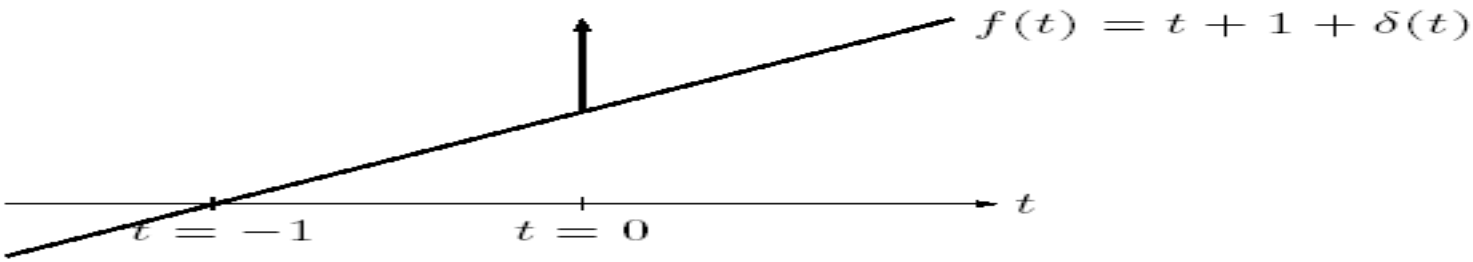
- ε *è generalmente piccolo*





Modalità di rappresentazione grafica di δ





Proprietà Formali di δ

- formalmente **definiamo** δ tramite la proprietà:

$$\int_a^b f(t)\,\delta(t)\,dt = f(0)$$

- Essendo $a < 0$, $b > 0$, ed *f continua per* $t = 0$
- **idea:** δ agisce su un intervallo di tempo piccolissimo e si ha che $f(t) \approx f(0)$
- $\delta(t) = 0$ per $t \neq 0$
- *Per* $a < 0$, $b > 0$
- *Per* $a > 0$, $b < 0$
- $\delta(0)$ non è definita

$$\int_a^b \delta(t)\,dt = 1$$

$$\int_a^b \delta(t)\,dt = 0$$

$$\int_a^b \delta(t)\, dt = 0$$

È **ambigua** se a o b sono =0

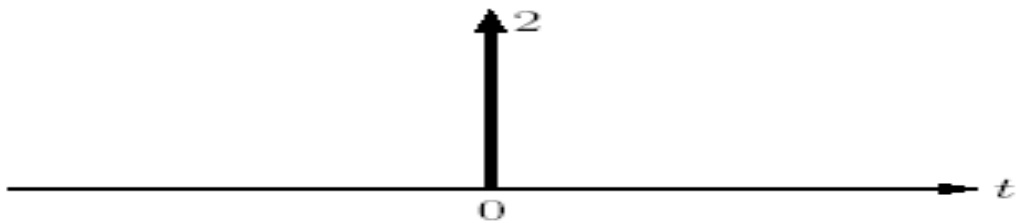
- Per evitare confusione usiamo i limiti a- o b+ per indicare se includiamo l'impulso oppure no. Esempio:

$$\int_{0+}^{1} \delta(t)\, dt = 0, \quad \int_{0-}^{1} \delta(t)\, dt = 1, \quad \int_{-1}^{0-} \delta(t)\, dt = 0, \quad \int_{-1}^{0+} \delta(t)\, dt = 1$$

Impulsi scalati

- $\alpha\delta(t - T)$ rappresenta un impulso al tempo T con *magnitudine* α. Si ha per a<T<b ed f continua in T:

$$\int_a^b \alpha\delta\ (t - T)\ f(t)\,dt = \alpha f(T)$$



Proprietà di Shift

- Il segnale $u(t) = \delta(t - T)$ è una funzione impulso all'istante $t = T$
  con $a < T < b$, ed f continua a $t = T$, si ha:

$$\int_a^b f(t)\,\delta(t - T)\,dt = f(T)$$

Esempio:

$$\int_{-3}^{2} f(t)(2 + \delta(t+1) - 3\delta(t-1) + 2\delta(t+3))\,dt$$

$$= 2\int_{-3}^{2} f(t)dt + \int_{-3}^{2} f(t)\delta(t+1)dt - 3\int_{-3}^{2} f(t)\delta(t-1)\,dt$$

$$+ 2\int_{-3}^{2} f(t)\delta(t+3)\,dt = 2\int_{-3}^{2} f(t)dt + f(-1) - 3f(1)$$

Interpretazione Fisica

- Le funzioni impulso sono usate per modellare segnali fisici che agiscono su intervalli di tempo brevi ed il cui effetto dipende dall'integrale del segnale.
- **esempio:** urto di una massa all'istante $t = 2$ sec:la forza f agisce sulla massa m tra $t = 1.999$ sec e $t = 2.001$ sec

$$\int_{1.999}^{2.001} f(t)\,dt = I$$

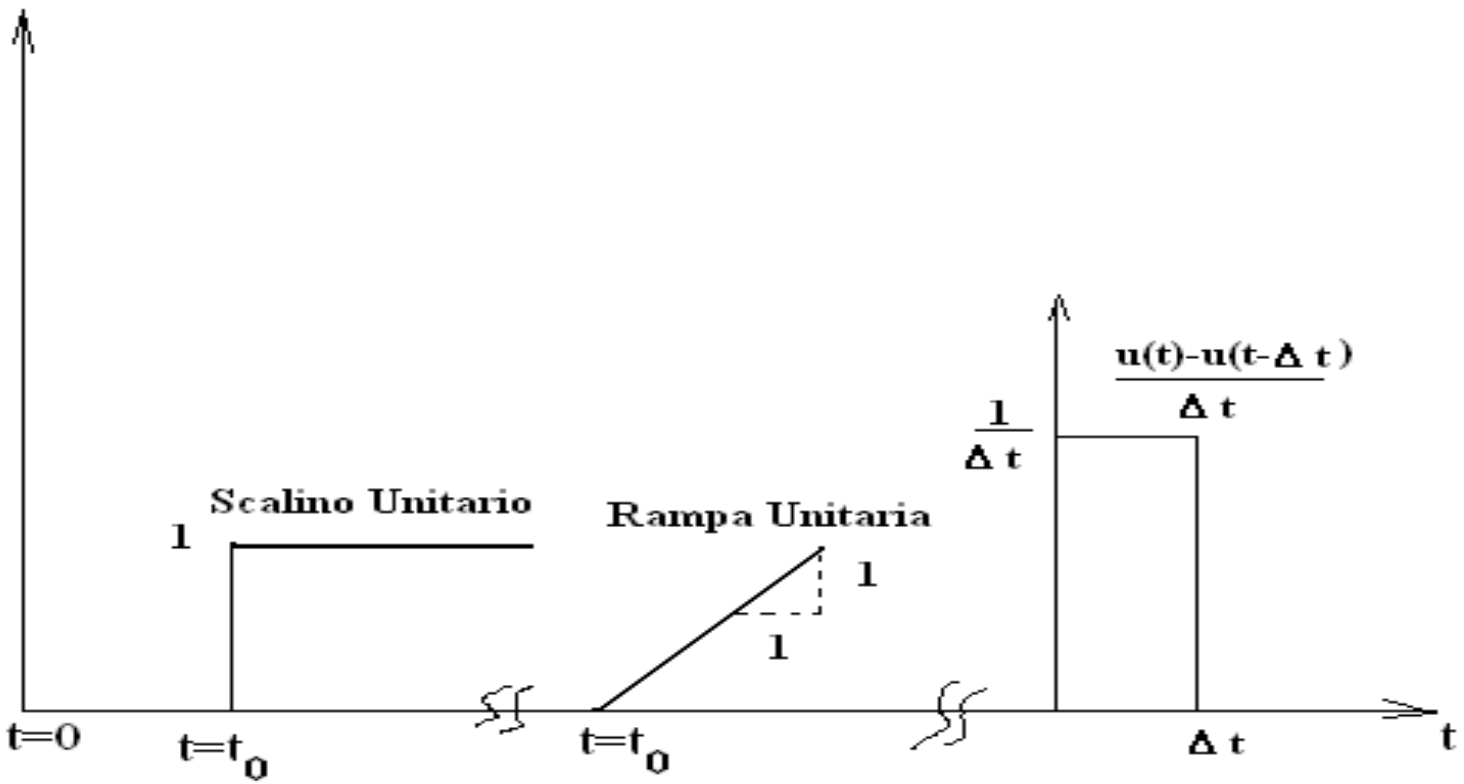
La variazione di velocità sarà:

$$v(2.2001) - v(1.999) = \frac{1}{m}\int_{1.999}^{2.001} f(\tau)\,d\tau = \frac{I}{m}$$

Per molti casi si può modellare la forza come un impulso (Es. t=2) di grandezza I

La risposta allo scalino unitario o *Risposta Indiciale* rappresenta l'uscita y(t) quando l'ingresso e' u(t) e tutte le condizioni iniziali sono nulle.

La risposta alla rampa unitaria e' rappresentata dall'uscita y(t) quando l'ingresso e' x(t)=t per t > 0 ; x(t)= 0 per t < 0 e tutte le condizioni iniziali sono nulle.



Sistemi del secondo ordine

Nello studio dei sistemi di controllo, le equazioni differenziali lineari a coefficienti costanti del secondo ordine del tipo:

$$\frac{d^2y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = \omega_n^2 x$$

sono importanti perché molti sistemi di ordine superiore si possono approssimare con sistemi del secondo ordine.

La costante ζ prende il nome di **smorzamento** e la costante ω_n e' **la pulsazione naturale del sistema**.

Di notevole importanza e' la determinazione della **risposta forzata** di questa equazione agli **ingressi canonici** definiti in precedenza. Infatti **la risposta forzata** ad un **impulso**, o a uno **scalino unitario** o a una **rampa unitaria** corrispondono alla **risposta impulsiva**, alla **risposta indiciale** o alla **risposta alla rampa unitaria** di un sistema rappresentato da questa equazione.

Nell'ipotesi che $0 \leq \zeta \leq 1$, l'equazione caratteristica dell'equazione differenziale in studio e' data da:

$$D^2 + 2\omega_n \zeta D + \omega_n^2 = (D + \omega_n \zeta - j\omega_n \sqrt{1-\zeta^2})(D + \omega_n \zeta + j\omega_n \sqrt{1-\zeta^2}) = 0$$

Quindi le radici sono:

$$\begin{cases} D_1 = -\omega_n \zeta + j\omega_n \sqrt{1-\zeta^2} \equiv -\alpha + j\omega_d \\ D_2 = \omega_n \zeta - j\omega_n \sqrt{1-\zeta^2} \equiv \alpha - j\omega_d \end{cases}$$

dove $\alpha \equiv \zeta\omega_n$ e' detto coefficiente di smorzamento e

$\omega_d \equiv \omega_n \sqrt{1-\zeta^2}$ prende il nome di pulsazione naturale smorzata.

α e' l'inverso della costante di tempo $\tau = 1/\alpha = 1/(\zeta\omega_n)$.

La funzione peso dell'equazione in studio e' quindi data da:

$$w(t) = \frac{1}{\omega_d} e^{-\alpha t} \sin(\omega_d t)$$

Esercizio per ricavare la funzione peso dell'equazione in studio

Nell'ipotesi che $0 \leq \zeta \leq 1$, l'equazione caratteristica dell'equazione differenziale in studio e' data da:

$$D^2 + 2\omega_n \zeta D + \omega_n^2 = (D + \omega_n \zeta - j\omega_n \sqrt{1-\zeta^2})(D + \omega_n \zeta + j\omega_n \sqrt{1-\zeta^2}) = 0$$

Quindi le radici sono:

$$D_1 = -\omega_n \zeta + j\omega_n \sqrt{1-\zeta^2} \equiv -\alpha + j\omega_d$$
$$D_2 = \omega_n \zeta - j\omega_n \sqrt{1-\zeta^2} \equiv \alpha - j\omega_d$$

Un sistema fondamentale e':

$$y_1 = e^{-\alpha t + j\omega_d t} = e^{-\alpha t} e^{j\omega_d t}$$
$$y_2 = e^{-\alpha t - j\omega_d t} = e^{-\alpha t} e^{-j\omega_d t}$$

w(t) si puo' scrivere come combinazione lineare di y_1 ed y_2:

$$w(t) = c_1 e^{-\alpha t} e^{j\omega_d t} + c_2 e^{-\alpha t} e^{-j\omega_d t}$$

c_1 e c_2 sono coefficienti incogniti

Procedura per determinare c_1 e c_2

Elaboriamo un po' w(t) $$w(t) = e^{-\alpha t}\left(c_1 e^{j\omega_d t} + c_2 e^{-j\omega_d t}\right)$$

Ricordandoci che : $e^{j\omega dt} = \cos\omega_d t + j\sin\omega_d t$ e

$e^{-j\omega dt} = \cos\omega_d t - j\sin\omega_d t$ w(t) diventa:

$$w(t) = e^{-\alpha t}\left[c_1 \cos\omega_d t - jc_1 \sin\omega_d t + c_2 \cos\omega_d t + jc_2 \sin\omega_d t\right]$$

$$w(t) = (c_1 + c_2)e^{-\alpha t}\cos\omega_d t + j(c_2 - c_1)e^{-\alpha t}\sin\omega_d t$$

$$w(t) = Ae^{-\alpha t}\cos\omega_d t + Be^{-\alpha t}\sin\omega_d t$$

essendo $A \equiv c_1 + c_2$ e $B \equiv j(c_2 - c_1)$ coefficienti incogniti
Determinabili mediante le condizioni iniziali.

$$w(0)=0,\left.\frac{dw}{dt}\right|_{t=0}=0,\ldots\ldots,\left.\frac{d^{n-2}w}{dt^{n-2}}\right|_{t=0}=0,\left.\frac{d^{n-1}w}{dt^{n-1}}\right|_{t=0}=1$$

$$w(0)=\left[Ae^{-\alpha t}\cos\omega_d t+Be^{-\alpha t}\sin\omega_d t\right]_{t=0}=A=0$$

$$\left.\frac{dw}{dt}\right|_{t=0}=Be^{-\alpha t}\left[\omega_d\cos\omega_d t-\alpha\sin\omega_d t\right]_{t=0}=\omega_d B=1$$

pertanto essendo A=0 e B=$1/\omega_d$ si ha in definitiva:

$$w(t)=Ae^{-\alpha t}\cos\omega_d t+Be^{-\alpha t}\sin\omega_d t=\frac{e^{-\alpha t}}{\omega_d}\sin\omega_d t$$

Per esercizio calcoliamoci la risposta al gradino unitario:

$$\frac{\mathbf{d^2y}}{\mathbf{dt^2}} + 2\zeta\omega_{\mathbf{n}} \frac{\mathbf{dy}}{\mathbf{dt}} + \omega_{\mathbf{n}}^2 \mathbf{y} = \omega_{\mathbf{n}}^2$$

La risposta allo scalino unitario sara':

$$y_u(t) = \int_0^t w(t-\tau)\omega_n^2 d\tau = 1 - \frac{\omega_n e^{-\alpha t}}{\omega_d} \sin(\omega_d t + \phi)$$

$$\phi \equiv \tan^{-1}(\omega_d / \alpha)$$

$$y_u(t) = \int_0^t w(t-\tau)\omega_n^2 d\tau = \int_0^t \frac{e^{-\alpha(t-\tau)}}{\omega_d} \sin\omega_d(t-\tau)\omega_n^2 d\tau$$

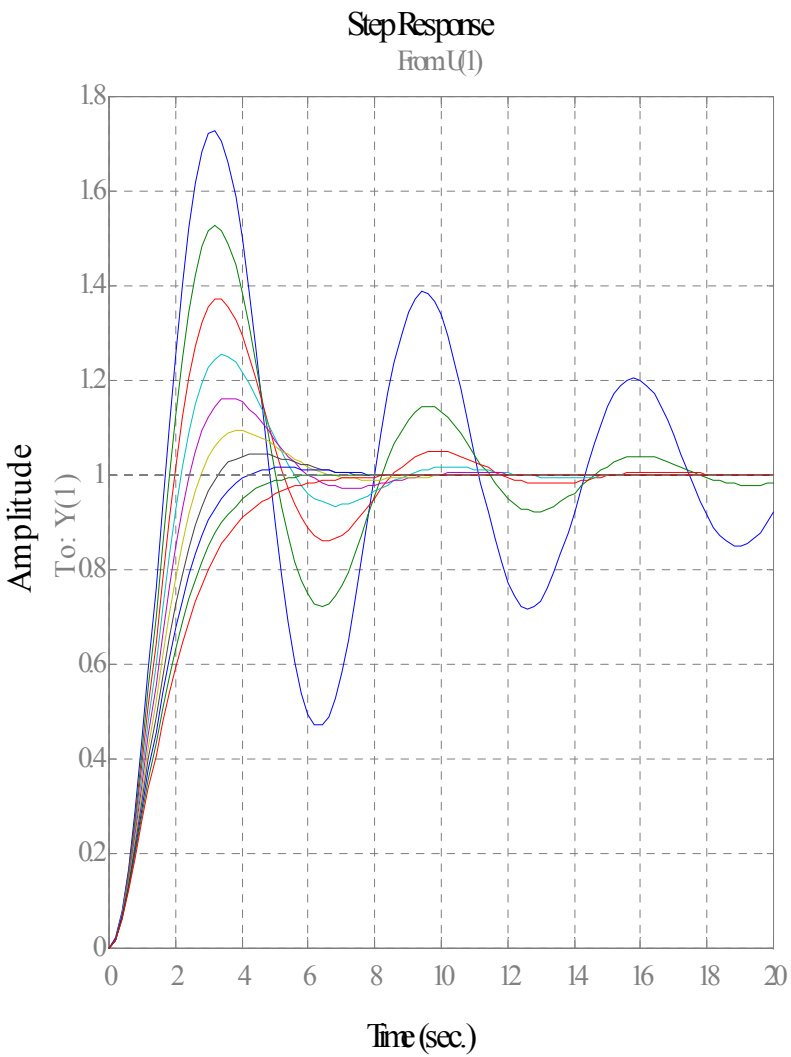
Per esercizio proseguite, tenendo conto che l'integrale si puo' risolvere mediante integrazione per parti…..

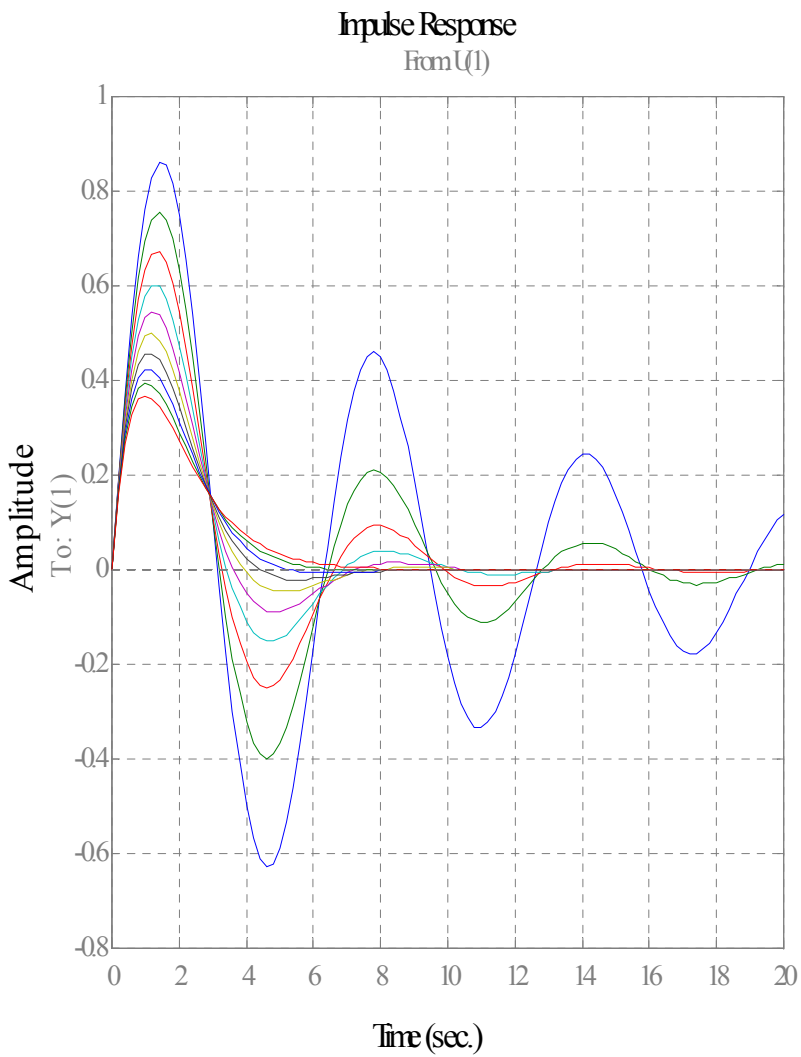
Con Matlab vediamo come e' possibile calcolare le funzioni risposta allo scalino unitario e all'impulso. Con la control toolbox vediamo anche come e' possibile avere la rappresentazione di queste risposte con le funzioni impulse e step. Segue un programmino "Naive" per ottenere lo scopo.

```
clear
num=[1];
figure
grid
zita=0.1:0.1:1;
for i=1:10
   hold on
   den=[1 2*zita(i) 1];
sys=tf(num,den);
impulse(sys,20)
end
hold off
```

```
figure
grid
for i=1:10
   hold on
   den=[1 2*zita(i) 1];
sys=tf(num,den);
step(sys,20)
end
```

```
num=1 ; den=[1 2*zita 1]; sys=tf(num,den)
```






```
impulse(sys,20)
```

$\omega_n=1$

```
step(sys,20)
```

INTRODUZIONE A MATLAB

A cosa serve questa presentazione

•Scopi di questo materiale:

• fornire le informazioni necessarie per l'uso di Matlab e Simulink in relazione agli esercizi del Corso

• dare una panoramica generale (tutt'altro che esauriente) delle potenzialita' di Matlab per la formulazione e la soluzione di problemi numerici .

Dove trovare altre informazioni?

Sito web di Mathworks:
www.mathworks.com
seguendo i link alla voce “support” e’ possibile
trovare i manuali di Matlab in formato pdf.
(http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml)

Un testo in italiano di introduzione a matlab e Simulink:
Guida Operativa a MATLAB, SIMULINK e Control Toolbox
Alberto Cavallo, Roberto Setola, & Francesco Vasca
Liguori Editore, 1994

INDICE DEL MATERIALE

- Descrizione generale di Matlab (v. 6.5)
- Quadro delle funzioni predefinite
- Definizione di matrici e vettori
- Definizione di polinomi
- Rappresentazione di sistemi dinamici lineari
- Analisi di sistemi di controllo
- Rappresentazione grafica dei dati
- L’ambiente di simulazione Simulink

Corso Introduttivo a MATLAB

- **Elementi fondamentali**
- Operazioni
- Polinomi e Grafica
- Programmazione
- Simulink

Elementi Fondamentali

- Introduzione
- Ingresso da tastiera
- Ingresso da files esterni
- Manipolazione di matrici
- Esercizi

Elementi Fondamentali

- **Introduzione**
- Ingresso da tastiera
- Ingresso da files esterni
- Manipolazione di matrici
- Esercizi

Introduzione

- MATrix LABoratory (ver. 6)
- *Computation*, *Visualization*, *Programming*
- "Learning by using"
- Dove trovare utili informazioni: http://www.mathworks.com

Elementi Fondamentali

- *Introduzione*
- **Ingresso da tastiera**
- Ingresso da files esterni
- Manipolazione di matrici
- Esercizi

Ingresso da tastiera

- Definizione di uno scalare e di un vettore

```
>> A=2 <CR>

>> a=[1 2 3 4] <CR>
```

- Cancellazione di un vettore (uso di ‘a’ e ‘A’)

```
>> clear A <CR>
```

- Definizione di una matrice (uso di ‘,’ e ‘;’)

```
>> b=[1 2 3 4 ... <CR>
5 6 7 8] <CR>

>> c=[1 2 3 4; 5 6 7 8]; <CR>
```

- … tanto per cominciare a “giocare”...

```
>> b=c <CR> >> b==c <CR>

>> a’ >> sum(b) >> diag(c)
```

Elementi Fondamentali

- *Introduzione*
- Ingresso da tastiera
- **Ingresso da files esterni**
- Manipolazione di matrici
- Esercizi

Ingresso da files esterni

- Un file ASCII "IN.TXT" col contenuto:

1.00 2.01 3.98 -5.67 2.1 6

>> load IN.TXT >> IN

- Le opzioni di formato

format short, long, short e, long e, hex, bank, rat,

- Salvataggio in un file

>> save *nomeFile nomeVar1 nomeVar2*

- Comandi di utilità :

>> help >> help matlab\general

>> help matlab\ops >> clc

Manipolazione di matrici

Elementi Fondamentali

- *Introduzione*
- Ingresso da tastiera
- Ingresso da files esterni
- **Manipolazione di matrici**
- Esercizi

- Matrici speciali

>> eye(3) >> eye(3,4)

>> zeros(2,3) >>ones(1,2)

>> diag([4 5 6 7])

- Stringhe di caratteri

>> a='asta' >> b='un''asta'

>>findstr(a,'a') (occorrenze di 'a' in a)

>>int2str(123) >> num2str(1.23)

>>str2num('1.23')

- Da stringa a matrice

>>A=str2mat('oggi','non','piove')

>> A(:,1) >> A(:,5)

Elementi Fondamentali

- *Introduzione*
- Ingresso da tastiera
- Ingresso da files esterni
- Manipolazione di matrici
- **Esercizi**

Esercizi

Dato x=(1,2,3,4), si costruisca y(1,3,4,5)

Dato x=(1,2,3,4,5,20), si costruisca

y=(1,2,3,4,5,20,20,5,4,3,2,1) usando ':' e 'fliplr'

Dati a=(1,2,3,4) e b=(7,8,9,10) si costruisca c=(1,7,2,8,3,9,4,10)

Si segua il Tutorial di Matlab sull'uso: si trova in Help➔Examples and Demos

Continua

- √ Elementi fondamentali
- **Operazioni**
- Polinomi e Grafica
- Programmazione
- Simulink

Operazioni

- Operazioni aritmetiche
- Funzioni
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Operazioni

- **Operazioni aritmetiche**
- Funzioni
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Operazioni aritmetiche

• Variabili predefinite (att.ne all'unità immaginaria!)
ans, eps, pi, i, j Inf, NaN, clock, cputime, date,flops, realmax, realmin, nargin, nargout

• Operazioni aritmetiche su scalari
+, -, *, /, ^, sqrt, \

```
>> 2/3 >> 2\3 >> 2*1/3
>> 1/2*3
```

• Operazioni aritmetiche su vettori e l'operatore '.*'

```
>>sqrt([1 2
3])>>[1:3]*[1:3]
>> [1:3]*[1:3]'
>>[1:3].*[1:3]
>> [1:3]^2 >> [1:3].^2
>> [1:3].^[1:3]
```

Operazioni

- **Operazioni aritmetiche**
- Funzioni
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Operazioni aritmetiche

- Operazioni algebriche su matrici

>> [1+j,1-2*j]' >> [1+j,1-2*j].'

>> A+3 >> 2*A >> B=inv(A) >> B=A^(-1)

>> X=A/B *(esegue $A*B^{-1}$)*

>> X=A\B *(esegue $A^{-1}*B$))*

>>B=pinv(A) ➔*($B=(A^TA)^{-1}A^T$)*•

La soluzione del problema Ax =b- se length(x)>length(b), cioè più incognite che equazioni, esistono infinite soluzioni (o meglio se rank(A)=rank([A b])); due possibili soluzioni sono:>>x=pinv(A)*b *(soluzione a min norma)* >> y=A\b *(soluzione con minor numero di zeri)*- se length(x)<length(b), cioè più equazioni che incognite, nesssuna soluzione (o meglio se rank(A)<rank([A b])); una soluzione approssimata è>> x=pinv(A)*b *(soluz. a min norma d'errore)*

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

• Arrotondamento

- **round** arrotonda all'intero più vicino

>> round(1.5) >>round(1.49…9)

(15 o 16 c.d.)

- **fix** arrotonda verso lo 0

>> fix(1.9) >> fix(1.1) >> fix(-2.1)

- **floor** arrotonda per difetto all'intero più vicino

>> foor(1.9) >> floor(-2.6) >> floor(-2.1)

- **ceil** arrotonda per eccesso all'intero più vicino>> ceil(1.9) >> ceil(1.1) >> ceil(-2.1)

• Approssimazioni razionali

- **rem** resto di una divisione intera

>>rem(3,2) >>rem(2,3)>>rem(2,0)

- **mod** risultato della a mod b

>> mod(3,7) >> mod(7,3)

- **rats** approssimazione razionale

>> rats(9.22) >> rats(9.22,5)

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

• Due esercizi:

1 Per ognuna delle coppie (a,b) riportate in tabella, si risolva l'equazione ax+b=0 con il vincolo che x sia intero. Se l'equazione non ammette soluzione si risolva il problema

min(x intero) |ax+b|

a = 2, 5, 6, 13, 5, 0.1

b = 4, 3, 28, -33, 72, sqrt(2)

Si calcoli (x mod 3) per x=(1, 5, 312, 22, 64), senza usare il comando mod

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

- Aritmetica complessa

- **real** parte reale
- **imag** parte immaginaria
- **conj** coniugato complesso
- **abs** valore assoluto o modulo complesso
- **angle** angolo di fase

- Esempi

```
>> 2+3i >> 2+3j >> 2+3*i
>> clear i >> a=3,z=2+ai
```

- Esercizio. Si tabelli il modulo e la fase della funzione razionale fratta riportata, per s=jω ed ω∈[10^(-2):10^2]

$$\frac{3s^2+5s+7}{s^3+5s^2+7s+12}$$

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

- Funzioni esponenziali e logaritmiche

- **pow2** esponenziale in base 2

- **exp** esponenziale in base e

- **log** logaritmo naturale

- **log2** logaritmo in base 2

- **log10** logaritmo in base 10

- Esempio

```
>> x=(1:.1:5)'; y=log(x);
>> [x y]
>> plot(x,y)
```

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

- Funzioni trigonometriche

- **sin** seno

- **cos** coseno

- **tan** tangente

- **asin** arcoseno

- **acos** arcocoseno

- **atan** arcotangente

- **atan2** arcotangente a quattro quadranti

- **cart2pol** da coordinate cartesiane a polari

- L'esempio della circonferenza

```
>> teta=-pi:.1:pi; x=cos(teta);
y=sin(teta);

>> [fase,modulo]=cart2pol(x,y)

>> plot(x,y) >> plot(modulo,fase)
```

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

• Funzioni su matrici

- **max** funzione di massimo (e min di minimo):>>max(x)>>max(A)>>max(max(A)

>> [Y,I]=max(A) >> max(A,B)

>> [m1,i1]=max(A); [m,k]=max(m1); h=i1(k); h,k

- **sort** e **sortrows** ordinamento

>> R=[1,2;5,1;3,3;2,4]; A=sort(R)

>> [S,i1]=sort(R); B=R(i1,:)

- **sum, rank, det, poly, trace, norm, eig, mean, median, std, cov, corrcoef**

- **expm, logm, sqrtm**

>> A=[0 1 1;0 0 1;0 0 0]; >> E1=exp(A);

>> E2=expm(A); E3=eye(3)+A+A^2/2; … E4= eye(3)+A+A.^2/2;

>> E1, E2, E3, E4

Operazioni

- Operazioni aritmetiche
- **Funzioni**
- Operazioni di relazione
- Operazioni logiche
- Esercizi

Funzioni

• Minimo e zero di una funzione

function b=prova(v);

*b=(v+1).*exp(-v.^2);*

```
>> a=fmin('prova',-2,2)
>> a=fzero('prova',-2)
```

• Minimo di una funzione di più variabili

function b=prova2(v);

x=v(1); y=v(2); z=v(3);

*b=x^2+2.5*sin(y)-z^2*x^2*y^2;*

```
>> a=fmins('prova2',[-0.6 -1.2 0.135])
```

Operazioni

- Operazioni aritmetiche
- Funzioni
- **Operazioni di relazione**
- Operazioni logiche
- Esercizi

Operazioni di relazione

- Operatori di relazione

- < minore - <= minore o uguale

- > maggiore - >= maggiore o uguale

- == uguale - ~= diverso

- Operatori logici

- & and - | or

- xor or esclusivo - ~ not

- Esempi

```
>> 2>3
>> 2+2~=4
>> P=(rem(A,2)==0)
```

(gli elementi di A divisibili per 2)

Operazioni

- Operazioni aritmetiche
- Funzioni
- Operazioni di relazione
- **Operazioni logiche**
- Esercizi

Operazioni logiche

• Come sempre gli operatori su matrici agiscono per colonne

- **any** dà 1 se almeno un elemento è diverso da 0

- **all** dà 1 se tutti gli elementi sono diversi da 0

>> A=[0 1 1;0 0 1;0 0 0]; any(A), all(A)

>> all(A<0) >> all(A>=0) >> any(A>0.5)

- **find** trova gli indici il cui argomento è diverso da 0

>> find(A) >> [h,k]=find(A)

>> t=0:.005:20; y=sin(t);

(trovare i valori di t per cui y=0.5)

>> tolleranza=0.05; i=find(abs(y-0.5)<tolleranza)

Operazioni

- Operazioni aritmetiche
- Funzioni
- Operazioni di relazione
- Operazioni logiche
- **Esercizi**

Esercizi

1 Dati x=3+j5, y=-2+j4, z=j3, si calcolino

x+y, x-z, (x+y)z, |x|, 1/y, z^2, log(x), e^y, |x/y|

2 Si risolva il sistema di equazioni

$x^2+y^2=a$ $x/y=b$

per a=(1, 4, 3) e b=(1, sqrt(3), 0.5)

3 Si verifichi, per vari valori di z, la validità della formula di Eulero

$e^z = e^{x+jy} = e^x (\cos y + j \sin y)$

4 Si rappresenti la funzione $e^{3t}\sin 5\pi t$ per $t\in[-2,2]$ con 45 punti

5 Si verifichi che la trasformazione bilineare

w=(z+2)/(z+4) trasforma cerchi in cerchi

Operazioni

- Operazioni aritmetiche
- Funzioni
- Operazioni di relazione
- Operazioni logiche
- **Esercizi**

Esercizi

6 Si rappresentino 5 periodi della funzione |sin t|

7 Si definisca il vettore x contenente 31 valori

dell'intervallo $[1/2, 2^5]$ ottenuti con spaziatura logaritmica in base 2

8 Definito un opportuno vettore x si valutino

L=(x>=2), L=(x<3), L=(x>3),

L=(x<4 & x>-4)

9 Dato il vettore x=(1, 34, 2, -12, 56, 7, 0, 9)

visualizzare i valori maggiori di 5.

- √ Elementi fondamentali
- √ Operazioni
- Polinomi e Grafica
- Programmazione
- Simulink

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Polinomi e Grafica

- **Operazioni su polinomi**
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Operazioni su polinomi

- Definizione di un polinomio

>> p=[1 0 2 -1]; *(definisce il polinomio s^3+2s-1)*

- Le radici di un polinomio

>> roots([1 2 1]) >> roots([1 0 1])

- Dalle radici al polinomio

>> radici=[-3-2j,-3+2j,-5]; poly(radici)

>> radici=[-3-2j,-3-2j,-5]; poly(radici)

- Il prodotto di polinomi

>> a=[1 0 1]; b=[1 1]; c=conv(a,b)

- Il rapporto di polinomi *$a(s)=q(s)b(s)+r(s)$*

>> [q,r]=deconv(a,b) >> [q,r]=deconv(b,a)

- Il valore di un polinomio in un punto

```
>> polyval(a,3)
```

>> w=-10:10; s=jw; num1=s.^2+3.*s+1

>> num2=polyval([1 3 1],s)

Polinomi e Grafica

- **Operazioni su polinomi**
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Operazioni su polinomi

• Derivata di un polinomio

```
>> p=[1 0 2 -1]; q=polyder(p)
>> conv(a,b); polyder(a,b)
```

• Espansione in fratti semplici di rapporti di polinomi

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots\ldots + \frac{r_n}{s-p_n} + k(s)$$

```
>> b=1; a=poly([-1-j,-1+j,3]);
>> [r,p,k]=residue(b,a)
>> [b1,a1]=residue(r(2:3),p(2:3),[])
>> c=[1 1]; d=poly([1 1 1 3]; residue(c,d)
```

$$\frac{r_{k1}}{s-p_k} + \frac{r_{k2}}{(s-p_k)^2} + \ldots\ldots + \frac{r_{kk}}{(s-p_k)^k}$$

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Interpolazione

Il comando **polyfit**

```
>> x=-1:.1:1; y=2*x+1+2*rand(1,length(x));
>> plot(x,y)
>> p=polyfit(x,y,1); y1=polyval(p,x);
>> plot(x,y,x,y1)
>> p=polyfit(x,y,5); y5=polyval(p,x);
>> p=polyfit(x,y,10); y10=polyval(p,x);
>> plot(x,y,x,y1,x,y5,x,y10)
```

Il comando **spline** (interpolazione con cubiche)

```
>> x=-1:.1:1; y=2*x+1+2*rand(1,length(x));
>> xx=-1:.1/2:1; yy=spline(x,y,xx);
>> plot(x,y,'o',xx,yy)
```

Il comando **interpft** (interpolazione più fitta con FFT)

```
>> yfitto=interpft(y,100);
>> plot(x,y,'o',xx,yy,-1:2/100:1-2/100,yfitto);
```

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Interpolazione

• Il significato del comando **spline**
-il significato fisico (la spline usata per disegnare)- si infittisce l'intervallo tra i campioni, tra questi si cerca un'approssimazione polinomiale e si assicura la differenziabilità fino ad un certo ordine nei punti di giunzione (la cosiddetta "pp-form"):

$$x \in [a,b], x \in [a = \xi_1, \xi_2, ..., \xi_{h+1} = b]$$

$$p_j(x) = \sum_{i=1}^{k} (x - \xi_j)^{k-i} / (k-i)!, \quad j = 1.....h$$

```
» x=0:pi/100:4*pi;y=sin(x)+sin(2*x)+sin(5*x);
» plot(x,y); lun=length(x); xc=x(1:10:lun);
» yc= x(1:10:lun);
 pp=spline(x,y), ppc=spline(xc,yc)
» pol=polyfit(xc,yc,4);yp=polyval(pol,x);
» plot(x,y,xc,yc,'*',x,yp,'r')
» xs=x(1:2:lun);ys=spline(xc,yc,xs);
» plot(x,y,xc,yc,'*',xs,ys,'ro')
```

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Interpolazione

Il significato del comando **interpft**

- effettua una Fast Fourier Transform sulla sequenza di dati assegnata, assunta periodica

$$X(k) = \sum_{n=1}^{N} x(n) e^{-j2\pi(k-1)\left(\frac{n-1}{N}\right)}, \quad 1 \le k \le N$$

$$x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{j2\pi(k-1)\left(\frac{n-1}{N}\right)}, \quad 1 \le n \le N$$

- per ottenere un campionamento più fitto si "riempie" x(n) con zeri, mantenendo la stessa periodicità della sequenza iniziale

```
» x1=-1:.1:1; x2=1.1:.2:2; x3=2.2:.3:3; x=[x1 x2 x3]
» y=2*x+1+2*rand(1,length(x));
» xx=-1:.1:3; yy=spline(x,y,xx);
» yf=interpft(y,100); plot(x,y,xx,yy,'*',-
1:4/99:3,yf,'r')
»xf=[-1:2/(33-1):1, 1.1:0.9/(33-1):2, 2.2:.8/(34-1):3];
» plot(x,y,xx,yy,'*',xf,yf,'r')
```

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Interpolazione

• Il comando **interp1** per l'interpolazione monodimensionale

```
>> x=-1:.1:1;
y=2*x+1+2*rand(1,length(x));
>> xi=-1:.03:1; yi=interp1(x,y,xi,'nearest')
>> plot(x,y,'o',xi,yi)
>> yil=interp1(x,y,xi,'linear');
plot(x,y,'o',xi,yi,xi,yil)
```

o altre opzioni come 'spline' e 'cubic'.

• Interpolazione bidimensionale: dati X ed Y monotoni

ZI = **interp2**(X, Y, Z, XI, YI, method)

```
>> [x,y]=meshgrid(-3:1:3)
>> z=x.*exp(-x.^2-y.^2)
>> surf(x,y,z);
>> [xi,yi]=meshgrid(-3:.25:3);
>> zi=interp2(x,y,z,xi,yi); surf(xi,yi,zi)
```

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Esercizi sui polinomi

- Calcolare i punti estremali e gli eventuali punti di flesso delle funzioni f(x)=x^4+5x^3+1 e g(x)=(x+1)/(x^3+3x+1).
- La densità dell'aria r (in kg/m3) varia con la quota h (in km) secondo la tabella riportata. Determinare un polinomio interpolatore di ordine n=1, 2, 3, …, 6 e la norma dell'errore commesso. Si calcolino poi i valori della densità dell'aria ai valori di quota 10, 20, 30, 40,50 e 60.

h=7, 10, 15, 21, 27, 34, 39, 43, 47, 51, 55, 59, 61

r=556, 369, 191,75, 26.2, 9.9,4.4,2.3, 1.4,.8, .5,.33,.25

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Grafici bidimensionali

• Rappresentazione di un vettore di dati e uso del **plot**

>> plot(rand(10)) >> plot(rand(1,10))
>> plot(rand(1,10),'r*:')
>> help plot
>> x=[1:2:10,13:-3:-1]; y=rand(1,length(x));
>> plot(x,y)

• Più grafici nella stessa finestra

>> sublot(2,2,1); plot(rand(1,10))
>> sublot(2,2,3); plot(3*rand(1,10))

• Scalatura degli assi

>> axis([-1 5 -2 Inf])

• Altri comandi: xlabel, ylabel, title, grid, gtext, ginput

>> title('\it{alfa si scrive} \alpha')
>> t=0:.1:2*pi; plot(t,sin(t);
>> text(3*pi/4,sin(3*pi/4),
'\leftarrowsin(t)=0.707')

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Grafici bidimensionali

- Le proprietà di una figura

```
>> a=plot(rand(1,10)); set(a)
```

- Le opzioni sulla finestra di figura

- File --> Save As - File --> Export
- Tools --> Properties - Tools --> Legend
- Tools --> Add … - Tools --> Zoom

- Grafici multipli

```
>> plot(x1,y1,'k',x2,y2)
>> plot([1 2 3],[-1 -2 -3])
>> plot([1 2 3;-1 -2 -3])
>> teta=-pi:.1:pi;
plot(teta,[sin(teta);cos(teta)]);
>> legend('seno','coseno')
```

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Esercizi su 2D

• Verificare che la funzione cos(x) nell'intorno dell'origine è approssimabile con le somme parziali del suo sviluppo in serie di Mc Laurin, tracciando i grafici della funzione cos(x) e delle prime 5 somme parziali.

$$\cos(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$$

• Verificare a quale dente di sega converge la serie

$$\pi - \sum_{n=0}^{\infty} \frac{\sin(kx)}{k}$$

Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Grafici tridimensionali

• Un primo esempio

```
>> t=0:pi/10:10*pi; plot3(sin(t),cos(t),t)
>> x=0:.1:4; y=-2:.1:1; [X,Y]=meshgrid(x,y);
>> Z=sin(X).*cos(Y); plot3(X,Y,Z);
>> mesh(X,Y,Z); surf(X,Y,Z); surfc(X,Y,Z);
```

• Il colore per i grafici

```
>> [X,Y]=meshgrid(-
8:.5:8);R=sqrt(X.^2+Y.^2)+eps
>> Z=sin(R)./R; surf(Z);
```

• La mappa dei colori Red Green Blue (RGB)

```
nero 0 0 0
bianco 1 1 1
giallo 1 1 0
magenta 1 0 1
cyan 0 1 1
grigio 0.5 0.5 0.5
>> t=-pi:pi/10:pi; fi=(-pi/2:pi/20:pi/2)';
>> X=cos(fi)*cos(t); Y=cos(fi)*sin(t);
>> Z=sin(fi)*ones(size(t)); surf(X,Y,Z)
>> colormap([0 0 0;1 1 1]); C=rand(size(Z));
>> surf(X,Y,Z,C)
```

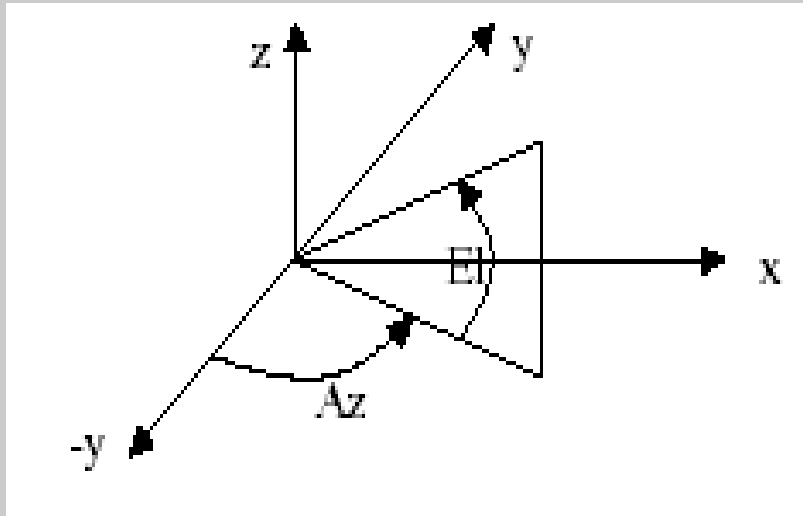
Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Grafici tridimensionali

- Le curve di livello

```
>> t=-2:.2:2; [X,Y,Z]=peaks(n);
mesh(X,Y,Z);
>> figure(2); contour(X,Y,Z,20);
>> hold on; [U,V]=gradient(Z,.2);
>> quiver(X,Y,U,V); hold off
>> [C,h]=contour(Z,10); clabel(C,h)
>> figure(3); mesh(X,Y,Z), hold on;
>> contour3(X,Y,Z,[1 1],'k');
>> Az=180;El=0; view([Az El])
```



Polinomi e Grafica

- Operazioni su polinomi
- Interpolazione
- Esercizi sui polinomi
- Grafici bidimensionali
- Esercizi su 2D
- Grafici tridimensionali
- Esercizi su 3D

Grafici tridimensionali

• I solidi di rotazione ottenuti col comando **cylinder**

- due possibili rotazioni della retta y=x:

```
» cylinder(0:.1:2)
» cylinder(-2:.1:2)
```

- le possibili rotazioni di un coseno:

```
» teta=0:pi/10:2*pi;
»
[X,Y,Z]=cylinder(4*cos(teta));
» subplot(2,2,1), mesh(X)
» subplot(2,2,2), mesh(Y)
» subplot(2,2,3), mesh(Z)
» subplot(2,2,4), mesh(X,Y,Z)
```

- √ Elementi fondamentali
- √ Operazioni
- √ Polinomi e Grafica
- Programmazione
- Simulink

Programmazione

- Le strutture fondamentali
- Script files
- Funzioni
- Esercizi di programmazione

- Programmazione
- Le strutture fondamentali
- Script files
- Funzioni
- Esercizi sulla programmazione

Le strutture fondamentali

- La struttura if-then-else

if *condizione*,
    istruzioni
elseif *condizione*,
    istruzioni
else
    istruzioni
end

- L'iterazione for

for i=1:n,
    istruzioni
end

- L'iterazione while

while *condizione*,
    istruzioni
end

- Programmazione
- • Le strutture fondamentali
- • Script files
- • Funzioni
- • Esercizi sulla programmazione

Script files

• Un esempio di M-file

```
% Un M-file per il calcolo dei numeri
di Fibonacci
f=[1 1]; i=1
while f(i)+f(i+1)<1000
f(i+2)=f(i)+f(i+1);
i=i+1
end
f
disp('premi un tasto per andare al
disegno'),
pause,
plot(f)
```

- Programmazione
- • Le strutture fondamentali
- • Script files
- • Funzioni
- • Esercizi sulla programmazione

Funzioni

• La struttura delle “function”

```
function [t,y]=seno(teta)
% Nella prima riga va l’help di directory
% Nelle righe successive il completamento all’help
t=teta; c=‘y=sin(teta)’; eval(c);
» [u,v]=seno(pi/3)
```

• Un esempio con lo switch

```
function [multipli,non_multipli]=mult(base)
% Nei primi 100 interi, i numeri multipli della base data
n=2; non_multipli=0; multipli=1;
while n<=100
switch rem(n,base)==0
case 0
non_multipli=[non_multipli n];
case 1
multipli=[multipli n];
end
n=n+1; end
```

- Programmazione
- • Le strutture fondamentali
- • Script files
- • Funzioni
- • Esercizi sulla programmazione

Esercizi sulla programmazione

• Usando il comando while si calcoli il valore della funzione sin(x) per x=0.01, 0.02, …, 1 rad usando la serie

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$$

considerando solo i termini sufficienti a garantire un'accuratezza fino alla quarta cifra decimale.
Si confronti quindi il risultato con quello fornito dal comando sin.

- Programmazione
- • Le strutture fondamentali
- • Script files
- • Funzioni
- • Esercizi sulla programmazione

Esercizi sulla programmazione

• Una delle mappe più studiate nella teoria del caos è la mappa quadratica:

$$x_{n+1} = x_n^2 + c$$

Per essa, le orbite asintotiche a partire da $x_0=0$ variano qualitativamente al variare di c. Assumendo i fenomeni transitori esauriti dopo 100 iterazioni, si tracci il diagramma delle orbite in funzione di c, con la seguente logica:

10 valori per $-0.75 < c < 0.25$
10 valori per $-1.25 < c < -0.75$
50 valori per $-1.4 < c < -1.25$
10 valori per $-1.75 < c < -1.4$
25 valori per $-1.78 < c < -1.76$
10 valori per $-2 < c < -1.78$

L'ambiente Simulink

- Simulink è un ambiente grafico per l'analisi e la simulazione di sistemi dinamici complessi.
- Quali sono le potenzialità rispetto a Matlab?

1. Permette di simulare sistemi complessi, composti da numerosi blocchi interconnessi tra loro;
2. Spesso taluni blocchi presentano comportamenti nonlineari o tempo-varianti;
3. Puo' essere necessario integrare nello stesso schema blocchi a tempo continuo e discreto .

Principio di funzionamento

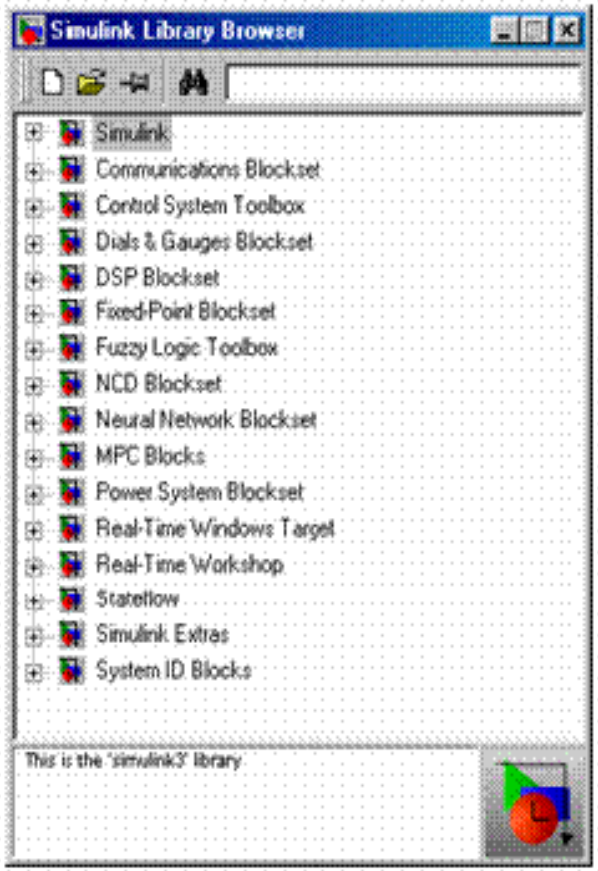
- Simulink contiene una libreria di blocchi elementari che descrivono elementi statici e dinamici;
- L'utente compone lo schema a blocchi del sistema da simulare mediante drag & drop di blocchetti elementari dalle librerie e la loro connessione tramite fili;
- Simulink genera automaticamente le equazioni sottostanti il modello e risolve il problema numerico di simulazione desiderato.

Principio di funzionamento

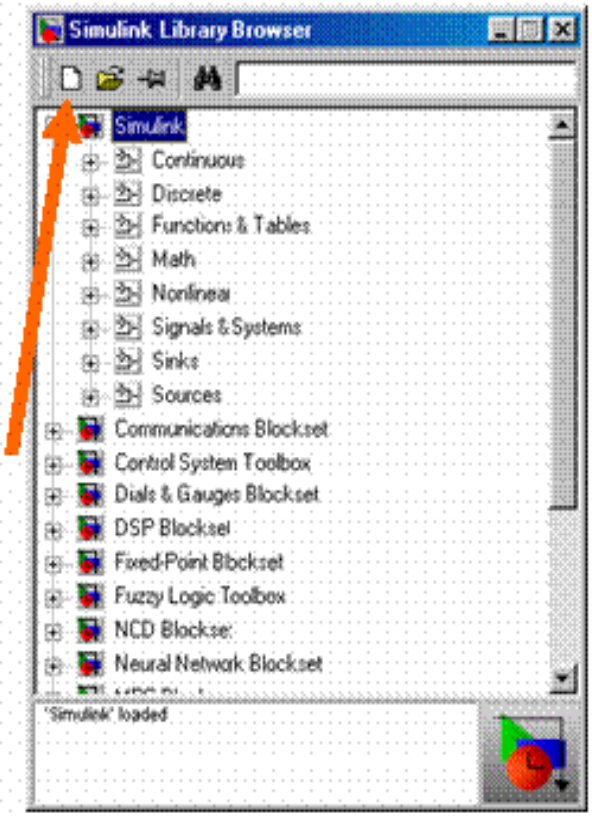
- Simulink interagisce con Matlab attraverso il Workspace => i modelli Simulink possono contenere variabili dichiarate nel Workspace;
- Allo stesso modo il risultato delle simulazioni puo' essere esportato nel Workspace e analizzato con Matlab.

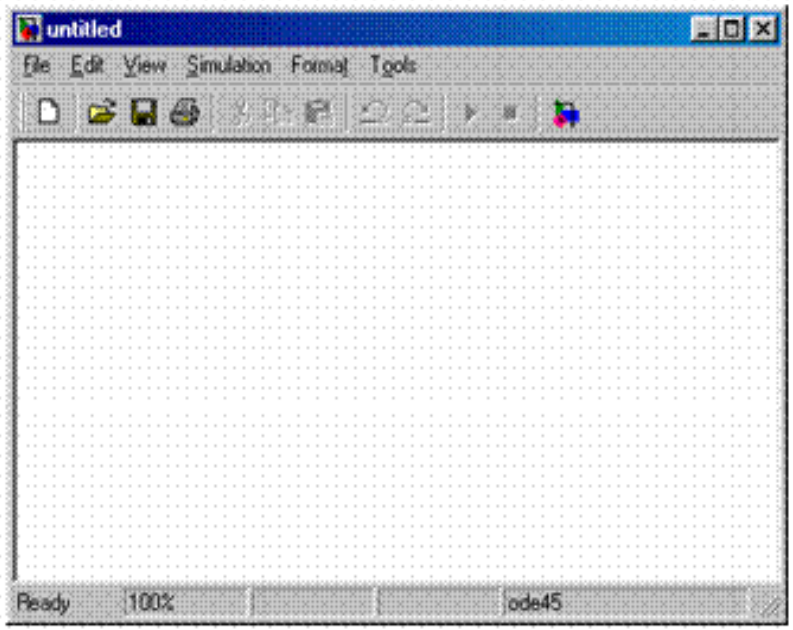
L'interfaccia grafica

- Digitando 'simulink' al prompt di Matlab si apre la libreria dei modelli.
- Da qui e' possibile creare un nuovo modello (foglio bianco) e comporre il sistema da simulare mediante i diversi blocchi.



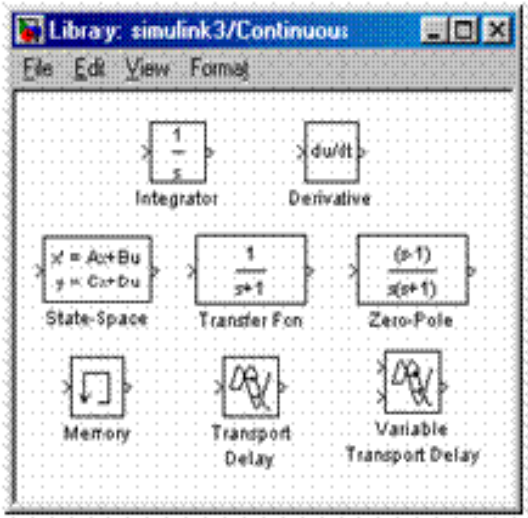
Nuovo modello



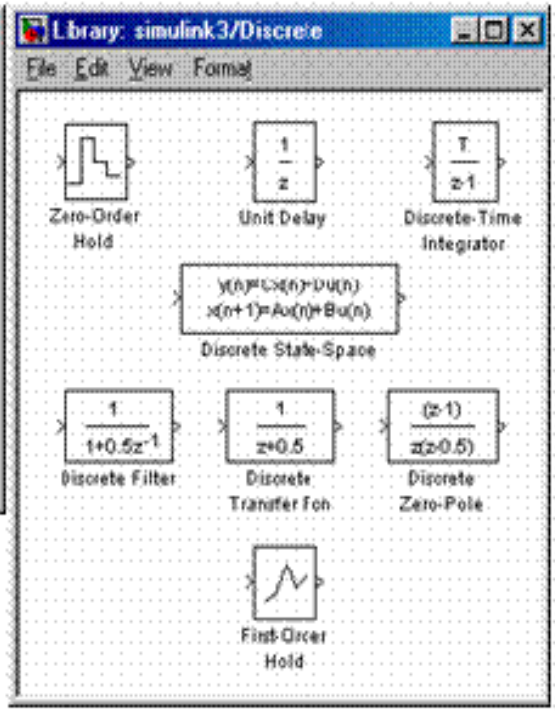


Ogni modello viene salvato in Simulink come file *.mdl

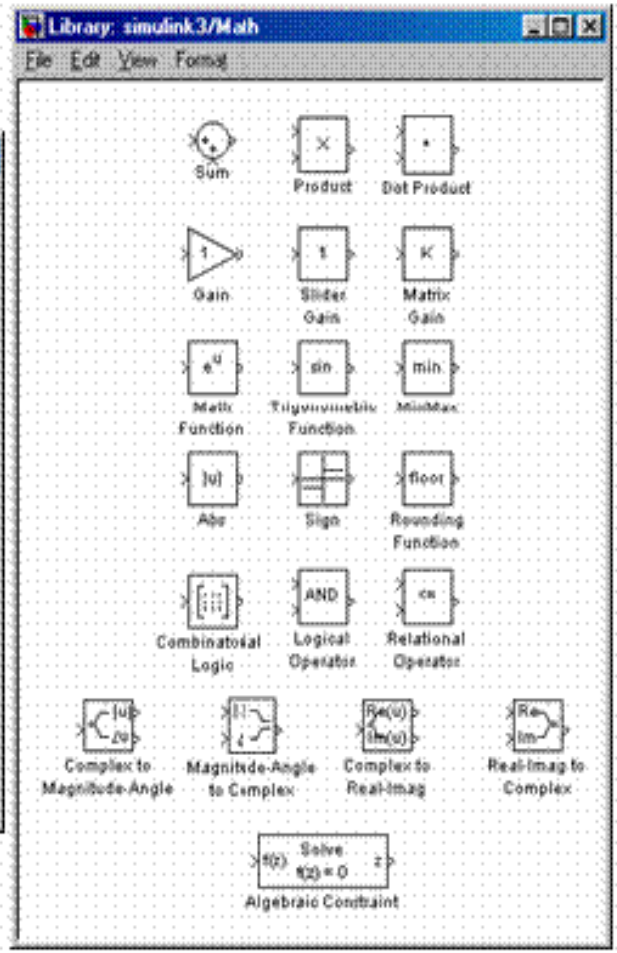
Principali librerie di Simulink (1)



Continuous

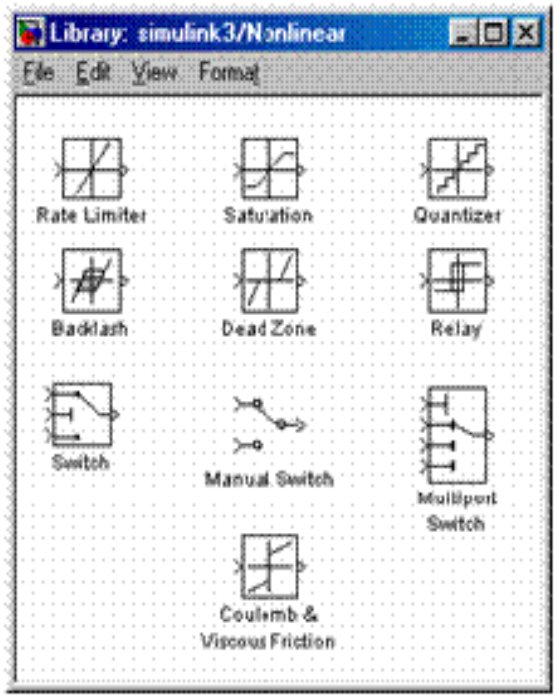


Discrete

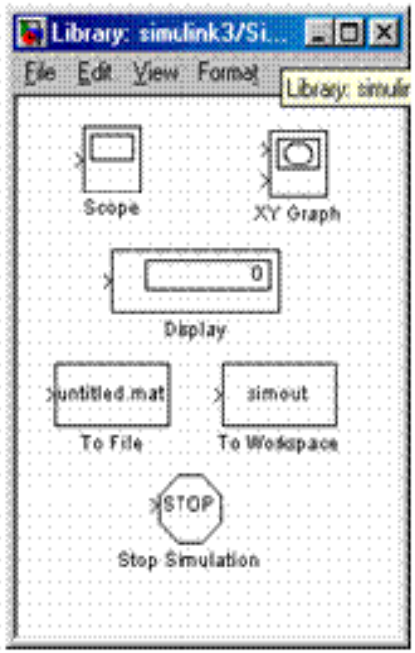


Math

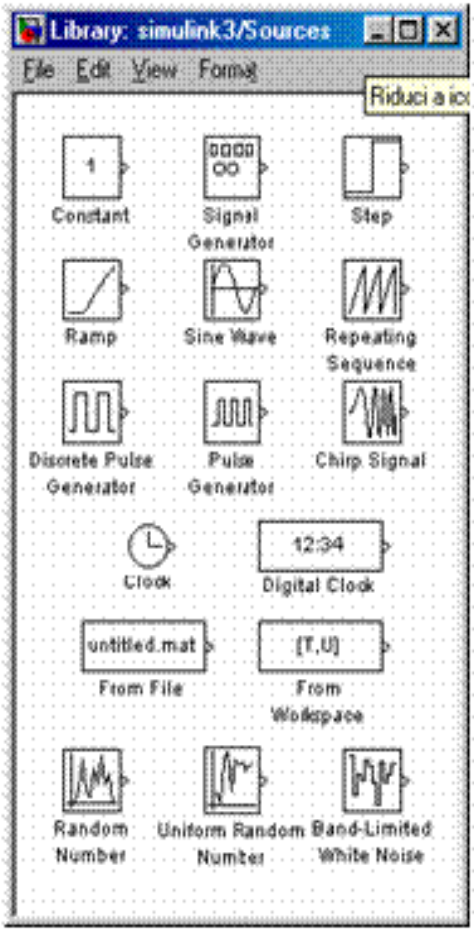
Principali librerie di Simulink (2)



Nonlinear



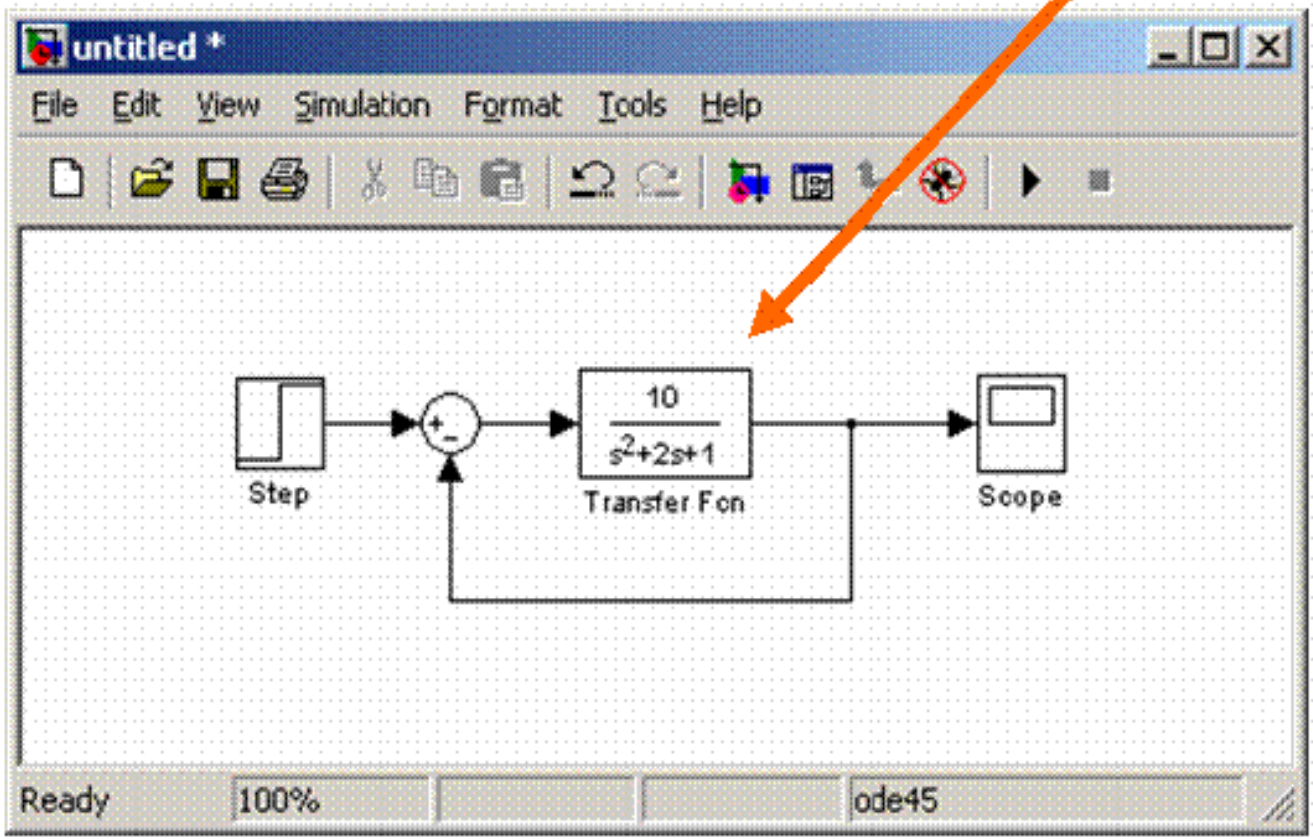
Sinks



Sources

Esempio

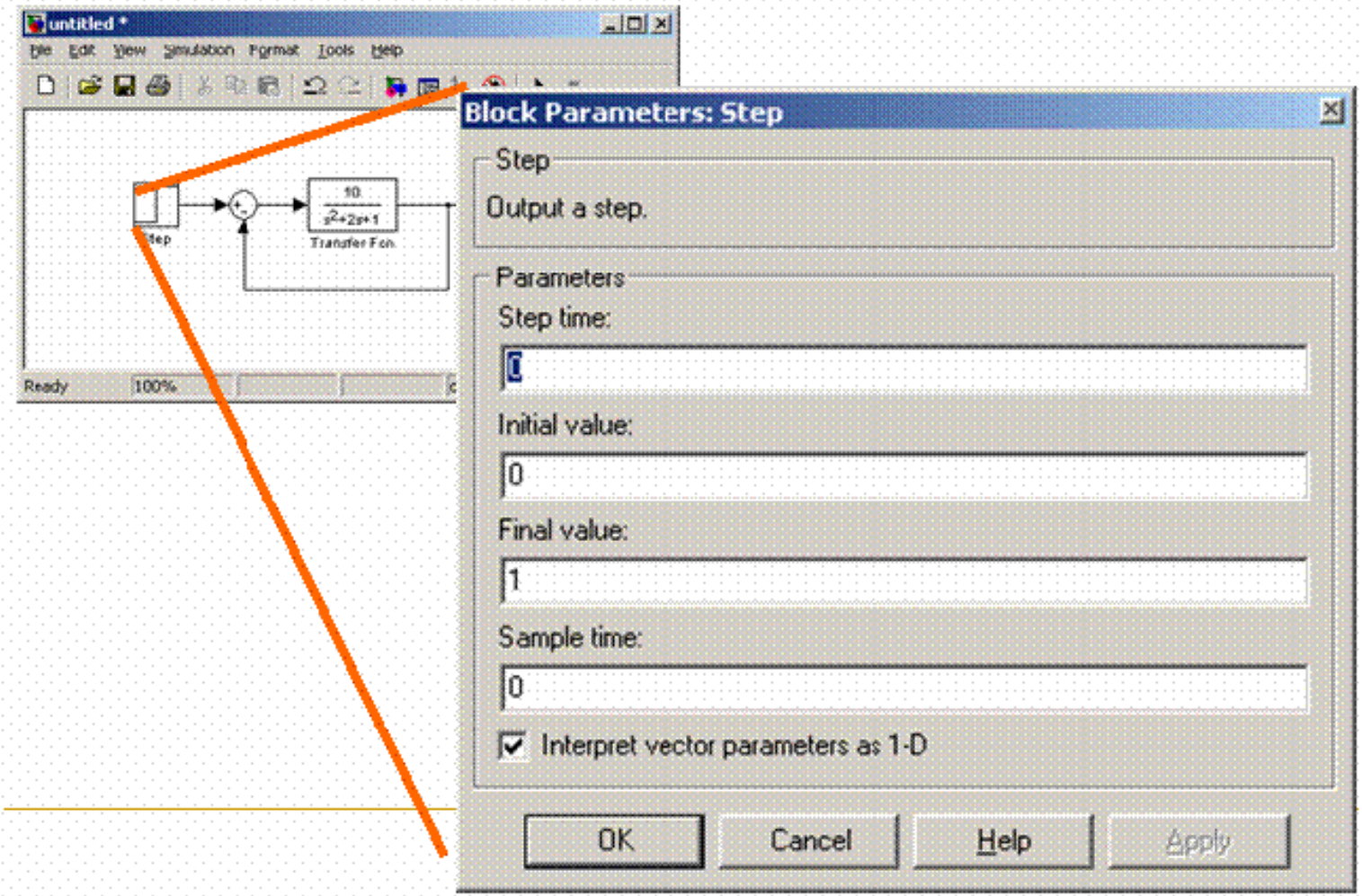
$$L(s) = \frac{10}{(1+s)^2}$$



Blocchi da usare

- Per la creazione del sistema dinamico:
  Continuous => TransferFcn
- Il gradino in ingresso si trova in:
  Sources => Step
- Il blocco di visualizzazione dei risultati si trova in:
  Sinks => Scope

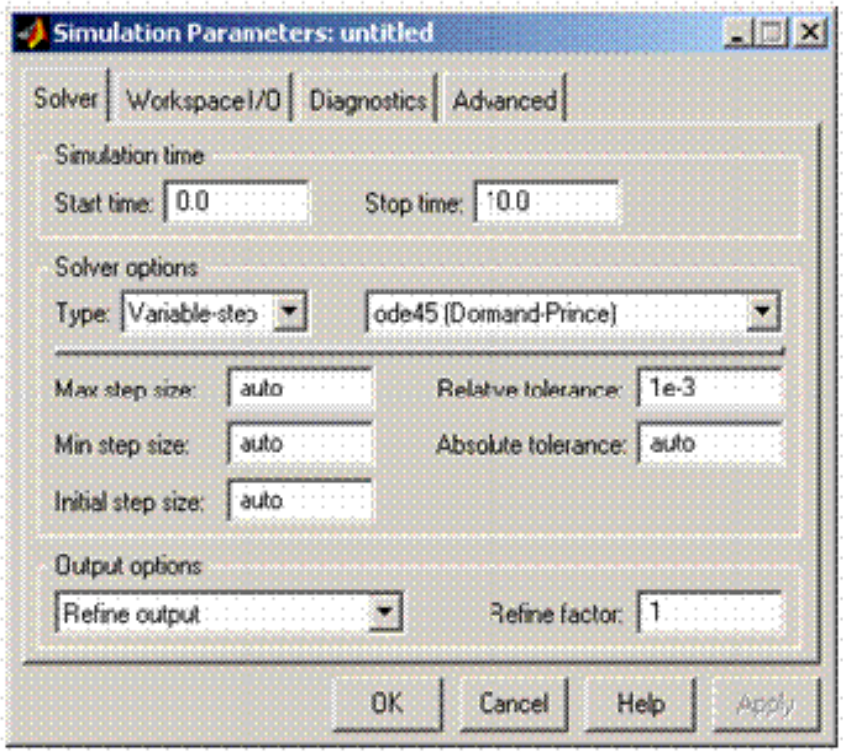
Modello e parametri



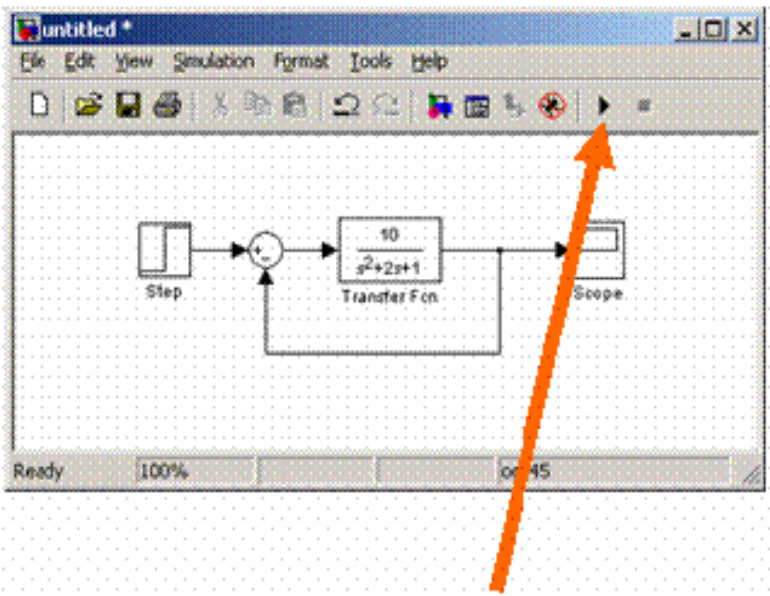
Parametri della simulazione

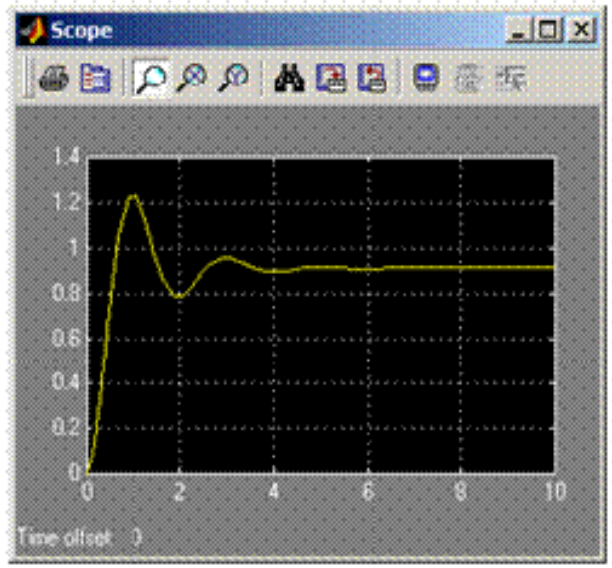
- L'utente deve definire:

1. Istanti di inizio e fine della simulazione;
2. Tipo di solutore numerico (se il problema richiede metodi particolari);
3. Parametri del solutore (in genere i default vanno bene…).



Simulazione e risultati



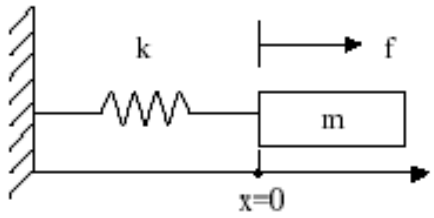


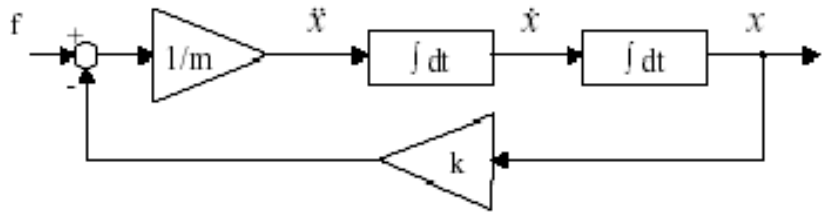
- Simulink
- • Come costruire uno schema
- • Sources e Sinks
- • La simulazione
- • Esercizi

Come costruire uno schema

sistema massa-molla

$$m\ddot{x} = -kx + f \quad \Rightarrow \quad \ddot{x} = \frac{1}{m}(-kx + f)$$





- Lo schema Simulink corrispondente

- Simulink
- • Come costruire uno schema
- • Sources e Sinks
- • La simulazione
- • Esercizi

Esercizi

• Costruire lo schema Simulink ed analizzare la risposta al gradino unitario del sistema:

$$\ddot{x} - 0.5\dot{x} + 2x - x^2 = u$$

• Il classico modello preda-predatore:

$$\dot{N}_1 = 3.5N_1 - 0.1N_1N_2, \quad \dot{N}_2 = -4N_2 + 0.06N_1N_2$$

con N_1 prede ed N_2 predatori. Si analizzi il comportamento del sistema al variare delle condizioni iniziali ($N_1(0)=100$, $N_2(0)=20$).