



Laboratorio di Matlab

Rudy Manganelli

Dipartimento di Ingegneria dell'Informazione,
Università di Siena

manganelli@dii.unisi.it

<http://www.dii.unisi.it/~control/MatLab/LabMatlab.html>

Luglio 2008

Symbolic toolbox

Il *Symbolic Math Toolbox* è il toolbox che fornisce gli strumenti per gestire il calcolo simbolico (manipolazione di variabili e risoluzione di espressioni ed equazioni in cui sono trattate variabili di tipo simbolico) in ambiente Matlab

- **Calcolo:**

- derivate
- integrali
- limiti
- serie di Taylor

- **Algebra Lineare:**

- matrice inversa
- determinante
- autovalori
- matrice diagonale





- rango di una matrice
- null-space
- polinomio caratteristico

- **Semplificazioni di espressioni algebriche**

- **Soluzione di equazioni**

- **Trasformazioni integrali:**
 - trasformata di Laplace
 - trasformata di Fourier
 - trasformata Z

- **Grafica**

Per saperne di più ...

`help symbolic`



Funzioni di base

- Dichiarazione di variabili simboliche:

```
>> syms x y z
```

crea nel workspace le variabili simboliche `x y z`

Metodo alternativo per creare una variabile simbolica `'x'` :

```
>> x = sym('x');
```

- Passaggio da *double* a variabile simbolica:

```
>> sym(variable_name);
```

trasforma un *double* in una variabile simbolica (istanza di un oggetto simbolico)

Attenzione: c'è differenza tra l'inserimento della variabile tra apici o no

nell'utilizzo del comando `sym!!!`

```
>> A = zeros(4);
```

```
>> B = sym(A)
```

B=

[0, 0, 0, 0]

[0, 0, 0, 0]

[0, 0, 0, 0]

[0, 0, 0, 0]

- Passaggio da variabile simbolica a variabile numerica

se l'oggetto simbolico è composto solo da numeri si usa il comando `double` :

```
>> C = double(B)
```

C =

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

- Definizione di funzioni:

```
>> syms x y
```

```
>> f = 1/exp(sin(x^2+y^2))
```

f è ora un oggetto simbolico che rappresenta $f(x, y) = \frac{1}{e^{\sin(x^2+y^2)}}$



Calcolo

Derivate

Si utilizza il comando:

```
diff(function, der_variable, der_order)
```

Esempio 1:

```
>> syms x a
>> s = sin(a*x);
>> dsdx = diff(s, x, 1)
ans =
cos(a*x) * a
>> dsdx = diff(s)
ans =
cos(a*x) * a
```



Esempio 2:

```
>> f = 1/exp(sin(x^2+y^2))
>> diff(f,x,4)
ans =
16/exp(sin(x^2+y^2))*cos(x^2+y^2)^4*x^4+96/
exp(sin(x^2+y^2))*cos(x^2+y^2)^2*x^4*sin(x^2+y^2)
-48/exp(sin(x^2+y^2))*cos(x^2+y^2)^3*x^2+48/
exp(sin(x^2+y^2))*sin(x^2+y^2)^2*x^4-
144/exp(sin(x^2+y^2))*cos(x^2+y^2)*x^2*sin(x^2+y^2)
-64/exp(sin(x^2+y^2))*cos(x^2+y^2)^2*x^4+12/
exp(sin(x^2+y^2))*cos(x^2+y^2)^2-16/
exp(sin(x^2+y^2))*sin(x^2+y^2)*x^4+48/
exp(sin(x^2+y^2))*cos(x^2+y^2)*x^2+12/
exp(sin(x^2+y^2))*sin(x^2+y^2)
```

Integrali

- Calcolo di $I(x) = \int f(x) dx$: (*integrale indefinito*)

`fint = int(func, var_int);`

cerca di trovare la funzione `fint` t.c. `diff(fint, x) = func`

- Calcolo di $c = \int_a^b f(x) dx$: (*integrale definito*)

`c = int(func, var_int, a, b);`

Esempio 1:

```
>> syms x a
>> s = sin(a*x);
>> int(s, x)
ans =
-1/a*cos(a*x)
```

Esempio 2:

```
>> syms x a
>> s = sin(a*x);
>> int(s, x, 0, 2*pi/a)
ans =
0
```



Limiti

Per calcolare $\lim_{x \rightarrow x_0^{+/-}} f(x)$ si utilizza:

```
k = limit(func, var, val, dir);
```

calcola il limite della funzione `func` per la variabile `var` che tende al valore `val` da destra se `dir = 'right'`, da sinistra se `dir = 'left'`

Esempio:

```
>> syms x
```

```
>> limit(diff(abs(x)), x, 0, 'left')
```

```
ans =
```

```
-1
```

```
>> limit(diff(abs(x)), x, 0, 'right')
```

```
ans =
```

```
1
```



Algebra lineare

Matrice inversa

Data una matrice simbolica A quadrata, voglio determinare $B = A^{-1}$:

```
B = inv(A_symb);
```

Esempio:

```
>> syms a b c d
```

```
>> A = [a b; c d]
```

```
A =
```

```
[ a, b]
```

```
[ c, d]
```

```
>> B = inv(A)
```

```
B =
```

```
[ d/(a*d-b*c), -b/(a*d-b*c) ]
```

```
[ -c/(a*d-b*c), a/(a*d-b*c) ]
```

Determinante

Data una matrice simbolica A quadrata voglio calcolare il suo determinante:

```
d = det(A);
```

Esempio:

```
>> syms a b c d e f g h i
>> A = [a b c; d e f; g h i]
A =
[ a, b, c]
[ d, e, f]
[ g, h, i]
>> det(A)
ans =
i*a*e-a*f*h-i*d*b+d*c*h+g*b*f-g*c*e
```

Autovalori

Data una matrice simbolica A quadrata voglio calcolare i suoi autovalori:

`eig(A)` ;

Esempio:

```
>> syms a b c d
```

```
>> A = [a b; c d];
```

```
>> eig(A)
```

```
ans =
```

```
[1/2*a+1/2*d+1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
```

```
[1/2*a+1/2*d-1/2*(a^2-2*a*d+d^2+4*b*c)^(1/2)]
```



Matrice diagonale

Dato un vettore simbolico V di n elementi, voglio creare la matrice diagonale costituita dagli elementi di V :

`diag(V) ;`

Esempio:

```
>> syms a b c d
>> V = [a b c d]
V = [ a, b, c, d]
>> diag(V)
ans =
[ a, 0, 0, 0]
[ 0, b, 0, 0]
[ 0, 0, c, 0]
[ 0, 0, 0, d]
```

Rango di una matrice

Data una matrice simbolica A , voglio calcolare il suo rango:

$\text{rank}(A)$;

Esempio:

```
>> syms a b c d
>> A = [a b c; 2*a 2*b 2*c; a d b]
A =
[ a, b, c]
[ 2*a, 2*b, 2*c]
[ a, d, b]
>> rank(A)
ans =
2
```

Semplificazione di espressioni algebriche

- `f = simple(func)`: ricerca la forma più semplice di una espressione simbolica.
- `f = simplify(func)`: semplificazione standard.
- `f = subs(func, {var1 var2 ...}, {val1 val2 ...})`
sostituisce alle variabili `var1, var2, ...` dell'espressione `func` i valori corrispondenti `val1, val2, ...`

Esempio:

```
>> syms a b c d
>> A = [a b 0; c d 0; 0 0 3]
A =
[ a, b, 0]
[ c, d, 0]
[ 0, 0, 3]
```



```
>> v = subs(eig(A), {b c}, {0 0})  
v =  
[  
[1/2*a+1/2*d+1/2*(a^2-2*a*d+d^2)^(1/2)]  
[1/2*a+1/2*d-1/2*(a^2-2*a*d+d^2)^(1/2)]  
>> simple(v)  
ans =  
[3]  
[a]  
[d]
```

Soluzione di equazioni

```
z = solve(func, var)
```

Determina le radici dell'equazione $\text{func} = 0$ rispetto alla variabile var .

Esempio 1:

```
>> syms x a b c
```

```
>> f = a*x^2+b*x+c
```

```
f =
```

```
a*x^2+b*x+c
```

```
>> solve(f, x)
```

```
ans =
```

```
[ 1/2/a*(-b+(b^2-4*c*a)^(1/2)) ]
```

```
[ 1/2/a*(-b-(b^2-4*c*a)^(1/2)) ]
```

Esempio 2:

```
>> syms x a
>> f2 = log(x) - 1/5*x^2 - a
f2 =
log(x) - 1/5*x^2 - a
>> solve(f2, x)
ans =
1/2*(-2*lambertw(-2/5*exp(2*a)))^(1/2)*5^(1/2)
```



Grafica

ezplot

ezplot: disegna funzioni di una variabile $f(x) : \mathbb{R} \rightarrow \mathbb{R}$

`ezplot(fun, [min_val max_val]):`

disegna la funzione `fun` nel dominio `[min_val max_val]`

Esempio:

```
>> syms x
>> f = 5*sin(x)^2+x^2;
>> ezplot(f, [-pi pi])
```

ezplot3

`ezplot3(var_x, var_y, var_z, [t_min t_max]):`

traccia traiettorie $f(t) : \mathbb{R} \rightarrow \mathbb{R}^3$

Esempio 1:

```
>> syms t
```

```
>> ezplot3(sin(t), t, cos(t), [-5*pi 5*pi])
```

Esempio 2:

```
>> syms t
```

```
>> ezplot3(t, sin(t), cos(t)^2, [-3*pi 3*pi])
```

ezmesh, ezsurf

`ezmesh(fun, [x_min x_max y_min y_max])` : traccia griglie
superficiali per funzioni $f(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ sul dominio

`[x_min x_max y_min y_max]`

`ezsurf(fun, [x_min x_max y_min y_max])` : traccia superfici
per funzioni $f(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ sul dominio

`[x_min x_max y_min y_max]`

Esempio:

```
>> syms x y
```

```
>> ezmesh(sin(x+y))
```

```
>> ezsurf(sin(x+y), [-2*pi 2*pi -2*pi 2*pi])
```



Esercizio 9

1. definire la funzione $f(x, y, z) = e^{\sin(x+y)} + z^2$ come oggetto simbolico
2. Calcolare il gradiente $\nabla f(x, y, z)$ ed la matrice Hessiana $H(f(x, y, z))$
3. plottare $f(x, y) = f(x, y, z)|_{z=3}$
4. plottare $f(y, z) = f(x, y, z)|_{x=2y}$
5. plottare $f(x, y) = f(x, y, z)|_{x=y^2; z=\sqrt{x}}$



Simulink

Simulink è un pacchetto software di Matlab che permette di simulare ed analizzare sistemi in cui ingressi e uscite variano nel tempo.

Simulink si basa su una procedura in due passi:

- Creazione di un modello grafico del sistema tramite utilizzo dell'*editor di modelli Simulink*. In questa fase l'utente specifica le relazioni matematiche che intercorrono tra gli ingressi e le uscite del sistema.
- Simulazione delle proprietà del sistema in un intervallo temporale fissato dall'utente.



Operazioni di base

- Creazione di un nuovo modello
- Aggiunta dei blocchi necessari (*drag & drop*)
- Collegamento dei blocchi (*drag & drop*)
- Settaggio del tempo di simulazione
- Salvataggio del modello
- Lancio della simulazione



Blocchi standard

- Sources
- Sinks
- Continuous
- Discrete
- Math Operations
- Signal Routing
- User-Defined Functions
- Subsystems

Sources

- *Constant* : genera un segnale costante con valore regolabile.
- *Ramp* : genera il segnale rampa con pendenza regolabile.
- *Random Number* : genera un segnale random con distribuzione guassiana, con media e varianza regolabili.
- *Sine* : genera il segnale $A \sin(\omega t + \phi) + c$.
- *Step* : genera il segnale gradino con valore iniziale, finale ed istante di attivazione regolabili.
- *Clock* : genera in uscita il tempo di simulazione.
- *From Workspace* : genera in uscita un segnale a partire da una variabile definita nel workspace: `var_in = [sig_time signal]`, dove `sig_time` e `signal` sono due vettori colonna.

Esempio:

```
>> t = 1:0.01:10;  
>> var_in = [t' sin(t)'];
```

Sinks

- *Scope*: visualizza l'andamento temporale del segnale in ingresso.
- *To File*: dato un segnale in ingresso genera un file '.mat' costituito da due vettori: il vettore dei tempi di simulazione ed il segnale.
- *To Workspace*: dato un segnale in ingresso genera in uscita una variabile di tipo struttura od array contenente il segnale ed eventualmente il tempo di simulazione.
- *XY Graph*: genera la traiettoria di un punto in \mathbb{R}^2 .

Continuous

- *Derivative* : calcola la derivata del segnale di ingresso.
- *Integrator* : calcola l'integrale del segnale di ingresso.
- *Transfer Fcn* : simula il sistema specificato tramite la funzione di trasferimento $N(s)/D(s)$.
- *Zero-Pole* : simula il sistema specificato tramite una funzione di trasferimento definita a partire da zeri e poli.
- *State-Space* : simula il sistema specificato tramite la sua rappresentazione di stato.
- *Transport Delay* : applica un ritardo specificato al segnale di ingresso.



Discrete

- *Unit delay* : ritarda il segnale tempo-discreto in ingresso di un istante di campionamento.
- *Discrete Integrator* : genera l'integrale tempo-discreto del segnale di ingresso.
- *Discrete Fcn* : simula il sistema tempo-discreto specificato tramite una funzione di trasferimento $\frac{N(z)}{D(z)}$.
- *Discrete Z-P* : simula il sistema tempo-discreto specificato tramite una funzione di trasferimento definita a partire da zeri e poli.
- *Discrete S-S* : simula il sistema tempo-discreto specificato tramite la sua rappresentazione di stato.

Math Operation

- *Abs* : genera in uscita il valore assoluto del segnale in ingresso.
- *Gain* : genera in uscita il segnale di ingresso moltiplicato per un parametro (guadagno) k regolabile.
- *Sum* : genera come segnale di uscita la somma dei segnali ingressi, con numero di ingressi e segni regolabile.
- *Trigonometric* : genera come segnale di uscita la funzione trigonometrica del segnale di ingresso specificata.
- *Math Function* : genera in uscita una delle funzioni elementari predefinite in Matlab, calcolata sul segnale di ingresso.
- *Matrix Concaten.* : genera in uscita la concatenazione di più segnali in ingresso.
- *Matrix Gain* : genera in uscita il segnale in ingresso moltiplicato per una data matrice.



Signal Routing

- *Mux* : genera in uscita un segnale multiplo (bus), costituito dall'unione di più segnali di ingresso.
- *Demux* : genera più segnali in uscita a partire da un segnale multiplo (bus) in ingresso.

User-Defined Functions

- *Fcn* : genera in uscita una qualsiasi funzione del segnale di ingresso (u) definita dall'utente.
- *Matlab Fcn* : genera in uscita una qualsiasi funzione Matlab predefinita del segnale di ingresso (u).

Discontinuities

- *Saturation*: limita il segnale di ingresso in un intervallo fissato dall'utente.

Subsystems

- *Subsystem*: permette all'utente di costruire sotto blocchi simulink (utile con schemi simulink complessi).

Esercizio 10 - Simulink

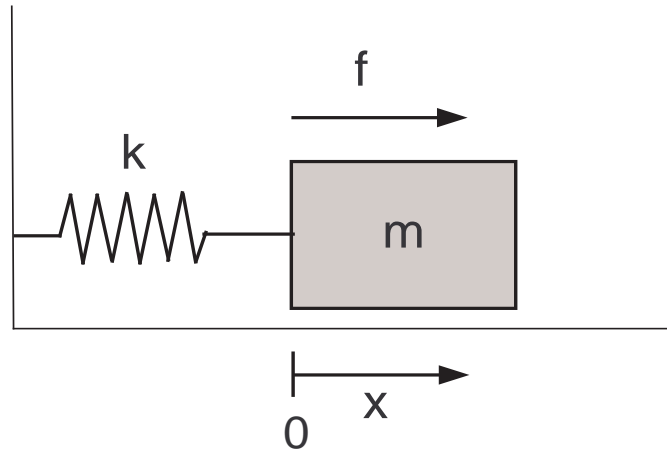
1. Creare uno schema a blocchi che:

- genera una rampa di pendenza 5, con display
- alla rampa sommare uno scalare k variabile nel workspace e farne il display congiuntamente con il precedente

2. Generare uno schema a blocchi che:

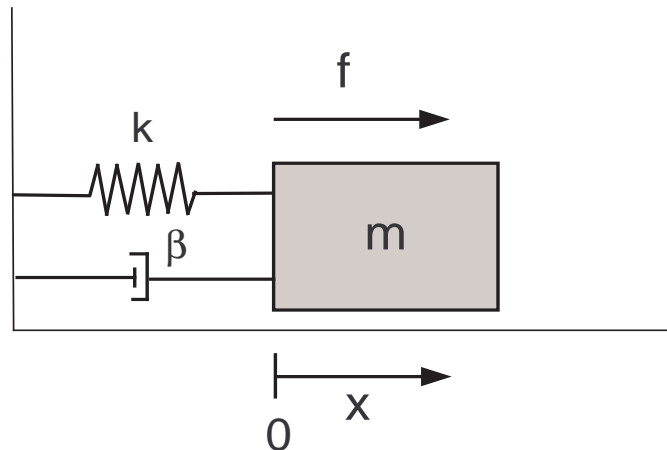
- genera il segnale $y(t) = \sin(2t)$ su un orizzonte $t \in [0, 4\pi]$
- genera il segnale $z_1(t) = \frac{d}{dt}y(t)$
- genera il segnale $z_2(t) = y(t + \frac{\pi}{2})$
- genera il segnale $z_3(t) = \int y(t)dt$

Esercizio 11 - Simulink



- Si consideri il *sistema massa-molla* riportato in figura. Supponendo che sia applicato in ingresso al sistema un gradino unitario, si simuli il sistema dinamico visualizzando l'andamento temporale della posizione x della massa.

Si supponga che la massa $m = 1.5$ Kg, la costante elastica della molla $k = 10^3$ N/m e $x(0) = \dot{x}(0) = 0$. (Suggerimento: $m\ddot{x} + kx = f$).



- Si consideri il *sistema massa-molla-smorzatore* riportato in figura. Supponendo che la costante di smorzamento della molla $\beta = 0.8$ Nms, si simuli il sistema dinamico visualizzando l'andamento temporale della posizione x della massa. Fissare gli altri parametri come nel punto precedente.
(Suggerimento: $m\ddot{x} + \beta\dot{x} + kx = f$).