

Introduzione al MATLAB

MATLAB = “MATrix LABoratory” basato su LAPACK e BLAS

- ▶ svolge calcoli matriciali (soluzione sistemi lineari, autoval.-autovett., ...)
- ▶ è un linguaggio di programmazione (controllo flusso, funzioni, input/output) interpretato
- ▶ consente la visualizzazione 2D e 3D

Al *prompt* della *command window* si inseriscono i comandi (in particolare, si lanciano i propri programmi). Nella *command window* viene visualizzato il risultato dei comandi eseguiti.

< M A T L A B >

Copyright 1984-2002 The MathWorks, Inc.

Version 6.5.0.180913a Release 13

Jun 18 2002

To get started, select "MATLAB Help" from the Help menu.

>>

Alcuni esempi:

```
>> 2+3
```

```
ans =
```

```
5
```

```
>> sin(2)
```

```
ans =
```

```
0.9093
```

```
>> sin(pi)
```

```
ans =
```

```
1.2246e-16
```

```
>> [1 2; 3 4] + [10 20; 30 40]
```

```
ans =
```

```
11    22
```

```
33    44
```

```
>> a=2
```

```
a =
```

```
    2
```

```
>> b=3
```

```
b =
```

```
    3
```

```
>> a+b
```

```
ans =
```

```
    5
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
ans	1x1	8	double array
b	1x1	8	double array

```
Grand total is 3 elements using 24 bytes
```

```
>> A= [1 2; 3 4]; B= [10 20; 30 40]; C=A+B;
```

```
>> C
```

```
C =
```

```
    11    22
```

```
    33    44
```

```
>> whos
```

Name	Size	Bytes	Class
A	2x2	32	double array
B	2x2	32	double array
C	2x2	32	double array
a	1x1	8	double array
ans	1x1	8	double array
b	1x1	8	double array

```
Grand total is 15 elements using 120 bytes
```

```
>> vett_riga = [1 2 3]
vett_riga =
     1     2     3
>> vett_colonna = [1; 2; 3]
vett_colonna =
     1
     2
     3
>> vett_riga+vett_colonna
??? Error using ==> +
Matrix dimensions must agree.
>> vett_riga * vett_colonna
ans =
    14
```

Per "estrarre" sottomatrici si usa la notazione ":"

```
>> A=[2 4 6 ; 5 7 9; 2 3 4; 6 7 8]
```

```
A =
```

```
    2    4    6
    5    7    9
    2    3    4
    6    7    8
```

```
>> A(2,:) 
```

```
ans =
```

```
    5    7    9
```

```
>> A(2:3,1:2)
```

```
ans =
```

```
    5    7
    2    3
```

Matlab come linguaggio di programmazione.

I files che contengono il codice sono detti M-files (sono *.m). Ne esistono di due tipi:

- ▶ Scripts: non hanno variabili di input/output, tutte le variabili vengono create nella memoria principale (quella della *command window*)
- ▶ Functions: ricevono un input e restituiscono un output, creano variabili locali e interagiscono con la memoria principale solo tramite il passaggio di input e output. Sintassi:

```
function [output1, output2] = nomefunction (input1,input2)
    % commento (visualizzato nell'help)
    .....
    .....
    .....
    output1=.....
    output2=.....
return
```

Consideriamo lo script script_di_prova.m:

```
clear all
A= [1 2 3 ; 4 5 6 ; 7 8 9];
B= [ 0;1;2];
C=A*B
```

Quando eseguito, produce:

```
>> script_di_prova
C =
     8
    17
    26
>> who
Your variables are:
A B C
```

Consideriamo la function `function_di_prova.m`:

```
function somma=function_di_prova (Add1,Add2)
somma=Add1+Add2;
return
```

Quando eseguito, produce:

```
>> function_di_prova([1 2], [3 4])
ans =
     4     6
>> who
Your variables are:
A     B     C     ans
```

costrutti di flow control:

- ▶ if
- ▶ switch and case
- ▶ for
- ▶ while
- ▶ continue
- ▶ break

Esempio:

```
if I == J
  A(I,J) = 2;
elseif abs(I-J) == 1
  A(I,J) = -1;
else
  A(I,J) = 0;
end
```

Esercizio: quadratura (mediante p.to medio) in $[-1, 1]^2$

Si vuole scrivere una function `punto_medio` che calcoli una approssimazione dell'integrale di una generica f sul quadrato $[-1, 1]^2$ mediante la formula di quadratura

$$\int_{-1}^1 \int_{-1}^1 f(x, y) dx dy = \sum_{T \in \mathcal{T}_h} f(x_T, y_T) |T|$$

dove \mathcal{T}_h è una triangolazione del dominio, e (x_T, y_T) rappresenta le coordinate del baricentro del triangolo $T \in \mathcal{T}_h$.

Problemi:

- 1) come si costruisce \mathcal{T}_h ?
- 2) come si passa alla function la funzione integranda f ?

Costruzione di \mathcal{T}_h (mediante PDE Toolbox)

```
>> [p,e,t]=initmesh('square','hmax',2);
```

```
>> p
```

```
p =
```

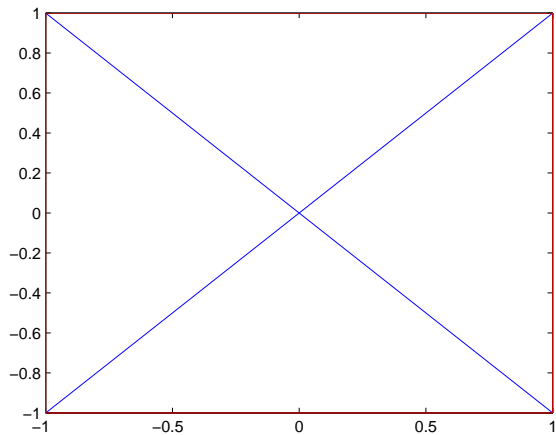
```
   -1    1    1   -1    0  
    1    1   -1   -1    0
```

```
>> t(1:3,:)
```

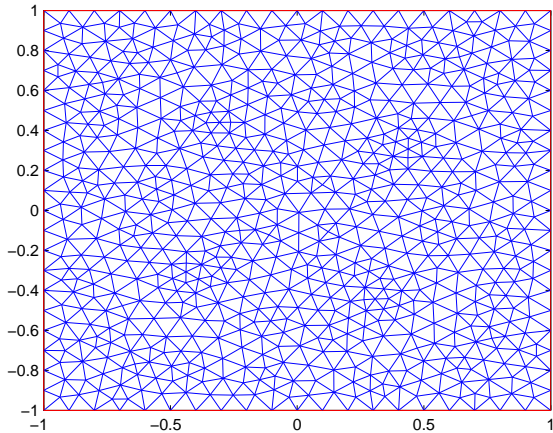
```
ans =
```

```
    2    3    4    1  
    1    2    3    4  
    5    5    5    5
```

```
>> pdemesh(p,e,t)
```



```
>> [p,e,t]=initmesh('squareg','hmax',.1);  
>> pdemesh(p,e,t)
```



```
function integrale=punto_medio(passo)
[p,e,t]=initmesh('squareg','hmax',passo);
integrale=0;
n_ele=size(t,2);
for i_ele=1:n_ele
    x_bar = sum(p(1,t(1:3,i_ele)))/3;
    y_bar = sum(p(2,t(1:3,i_ele)))/3;
    edge_1=[(p(:,t(3,i_ele))-p(:,t(2,i_ele)))); 0];
    edge_2=[(p(:,t(1,i_ele))-p(:,t(3,i_ele)))); 0];
    area_T=norm(cross(edge_1,edge_2))/2;
    integrale = integrale + f(x_bar,y_bar) * area_T;
end
return

function output = f(x,y)
    output = 1+x+y;
    return
```

```
>> punto_medio(1)
```

```
ans =
```

```
4.0000
```

```
>> format long
```

```
>> punto_medio(1)
```

```
ans =
```

```
4.0000000000000001
```

```
>> punto_medio(.05)
```

```
warning: Approximately 3199 triangles will be generated.
```

```
ans =
```

```
4.0000000000000009
```

integriamo ora:

```
function output = f(x,y)
    output = cos(pi/2*x).*cos(pi/2*y);
    return
```

si ottiene:

```
>> punto_medio(1)
ans =
    1.687681609571244
```

```
>> punto_medio(.1)
ans =
    1.622339784285813
```

```
>> 16/pi^2
ans =
    1.621138938277404
```

```
>> dblquad(@f,-1,1,-1,1,1e-10)
ans =
    1.621138938276451
```

Passaggio di f come argomento alla function

```
function integrale=punto_medio(fun,passo)
[p,e,t]=initmesh('squareg','hmax',passo);
integrale=0;
n_ele=size(t,2);
for i_ele=1:n_ele
    x_bar = sum(p(1,t(1:3,i_ele)))/3;
    y_bar = sum(p(2,t(1:3,i_ele)))/3;
    edge_1=[(p(:,t(3,i_ele))-p(:,t(2,i_ele)))); 0];
    edge_2=[(p(:,t(1,i_ele))-p(:,t(3,i_ele)))); 0];
    area_T=norm(cross(edge_1,edge_2))/2;
    integrale = integrale + fun(x_bar,y_bar) * area_T;
end
return
```

adesso possiamo passare (il puntatore a) la funzione integranda come argomento:

```
>> punto_medio(@f,.1)
ans =
    1.622339784285813
```

Studio dell'ordine di convergenza di punto_medio

```
clear all
int_es= 16/pi^2
for i=1:5
    h(i)=2^(-i)
    int_num= punto_medio(@f,h(i));
    err(i)=abs(int_es-int_num);
end
loglog (h,err)
xlabel('h')
ylabel('errore')
```

