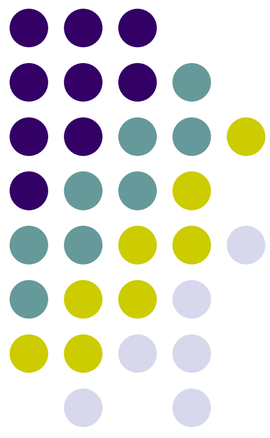
Numeri di macchina MATLAB

Un'introduzione



Dott. Francesca Incensi: francesca.incensi3@unibo.it

Problemi numerici, errori, numeri di macchina

Problemi matematici

Si consideri il seguente problema astratto: trovare *x* tale che

$$F(x;d) = 0 \qquad (*)$$

dove *d* è l'insieme dei dati da cui dipende la soluzione e *F* esprime la relazione funzionale tra *x* e *d.*

- Se *F* e *d* sono noti → PROBLEMA DIRETTO
- Se *F* e *x* sono noti → PROBLEMA INVERSO
- Se *d* e *x* sono noti → PROBLEMA DI IDENTIFICAZIONE

Problemi numerici

Un metodo numerico per la risoluzione del problema diretto (*) consiste, in generale, nel costruire una successione di problemi approssimati (*problemi numerici*)

$$F_n(x_n;d_n) = 0 \qquad n \geq 1$$

dipendenti da un certo parametro *n* con la speranza che $x_n \rightarrow x$ per $n \rightarrow \infty$.

Il problema numerico quindi costituisce l'approssimazione del problema matematico (*) che a sua volta modella un problema fisico.

Sorgenti di errori

In tale processo *l'errore globale e* esprime la differenza tra la soluzione effettivamente calcolata x^*_n e la soluzione fisica x_f di cui *x* fornisce un modello.

In tale prospettiva *e* può essere visto come la somma dell'errore e_m del modello matematico, espresso da $x-x_f$, con l'errore e_c del modello computazionale, cioè x^*_n-x

$$e = e_m + e_c$$

A sua volta e_m tiene conto dell'errore del modello matematico in senso stretto, e dell'errore sui dati, mentre e_c risulta essere combinazione dell'errore di discretizzazione numerica $e_n=x_n-x$ e dell'errore di arrotondamento e_a introdotto dal calcolatore.

Errore computazionale

ERRORI di TRONCAMENTO

dovuti al fatto che nel modello numerico le operazioni di passaggio al limite vengono approssimate con operazioni che richiedono un numero finito di passi.

ERRORI DI ARROTONDAMENTO

introdotti a causa della rappresentazione dei numeri al calcolatore: su un calcolatore in fatti può essere rappresentato solo un SOTTOINSIEME FINITO dell'insieme dei numeri reali.

Numeri reali: floating point

La notazione scientifica normalizzata:

32.213 ⟶ **0.32213×10^2** **normalizzata**

0.003221×10^4 **non normalizzata**

Fornisce una **rappresentazione univoca** dei numeri, fissata una base β:

$$x = \pm 0.a_1 a_2 .... \times \beta^{p} \qquad 0 \le a_i \le \beta - 1, \qquad a_1 \neq 0$$

$$x = \pm m \times \beta^{p}, \qquad \frac{1}{\beta} \le m < 1$$

Segno	***esponente* p**	***mantissa* m**

7

Numeri di macchina

I numeri reali si possono ben approssimare con numeri aventi una rappresentazione finita: INSIEME FINITO DI NUMERI A VIRGOLA MOBILE NORMALIZZATI (Numeri di macchina)

$$F(\beta, t, L, U) =$$

$$\left\{ x \in \Re = \pm (a_1 \beta^{-1} + a_2 \beta^{-2} + ... a_t \beta^{-t}) \beta^{p} \right\} \cup \{0\}$$

$$0 \le a_i \le \beta - 1, \qquad a_1 \neq 0 \qquad L \le p \le U, \qquad L < 0, U > 0$$

NON TUTTI I NUMERI REALI SONO RAPPRESENTABILI IN F.

8

Numeri di macchina

I. Esponente $L \le p \le U, \quad L < 0, U > 0$

Se $p \notin [L,U]$ p > U **overflow**

p < L **underflow**

II. Mantissa , t cifre disponibili

Se il numero di cifre nella mantissa è superiore a t:

I numeri non sono esattamente rappresentabili, occorre approssimarli: per **troncamento** o **arrotondamento**

Approssimazione per TRONCAMENTO

$$\overline{x} = \mathbf{fl}(x) = tronc(x) = \pm\left(a_1\beta^{-1} + a_2\beta^{-2} + \ldots + a_t\beta^{-t}\right)\beta^{p}$$

Esempio: base **β = 10** **t = 6**

y = 0.372145784 → troncamento → **ȳ = 0.372145**

Approssimazione per ARROTONDAMENTO

Si aggiunge $\frac{1}{2}\beta^{-t}$ alla mantissa e poi si tronca alla t-esima cifra

$$\overline{x} = \mathrm{fl}(x) = arr(x) =$$

$$= \pm tronc\left(a_1\beta^{-1} + a_2\beta^{-2} + \ldots + a_t\beta^{-t} + a_{t+1}\beta^{-t-1} + \frac{1}{2}\beta^{-t} \right)\beta^{p}$$

$$\overline{x} = \begin{cases} \pm\, 0.a_1a_2 \ldots a_t\beta^{\,p} & se \quad a_{t+1} < \dfrac{\beta}{2} \\ \pm\, 0.a_1a_2 \ldots (a_t + 1)\beta^{\,p} & se \quad a_{t+1} \geq \dfrac{\beta}{2} \end{cases}$$

Approssimazione per ARROTONDAMENTO

Esempio: base **β = 10** **t = 6**

y = 0.372145784 → arrotondamento → **ȳ = 0.372146**

y = 0.372145384 → arrotondamento → **ȳ = 0.372145**

Numeri di macchina

F(β, t, L, U) contiene $\mathbf{2(U-L+1)(\beta -1)\, \beta^{t-1}+1}$

F non è una perfetta simulazione di R

Non sono equispaziati sull'asse reale, ma si addensano in prossimità del più piccolo numero rappresentabile.

La loro densità decresce con l'aumentare del valore assoluto del numero

F(2,3,-1,2)

$0.100\ 2^{-1} = \frac{1}{2}\,\frac{1}{2} = \frac{1}{4}$

$0.101\ 2^{-1} = \left(\frac{1}{2} + \frac{1}{8}\right)\frac{1}{2} = \frac{5}{16}$

$0.110\ 2^{-1} = \left(\frac{1}{2} + \frac{1}{4}\right)\frac{1}{2} = \frac{3}{8}$

$0.111\ 2^{-1} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right)\frac{1}{2} = \frac{7}{16}$

$0.100\ 2^{0} = \frac{1}{2}\,1 = \frac{1}{2}$

$0.101\ 2^{0} = \left(\frac{1}{2} + \frac{1}{8}\right)1 = \frac{5}{8}$

$0.110\ 2^{0} = \left(\frac{1}{2} + \frac{1}{4}\right)1 = \frac{3}{4}$

$0.111\ 2^{0} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right)1 = \frac{7}{8}$

$0.100\ 2^{1} = \frac{1}{2}\,2 = 1$

$0.101\ 2^{1} = \left(\frac{1}{2} + \frac{1}{8}\right)2 = \frac{5}{4}$

$0.110\ 2^{1} = \left(\frac{1}{2} + \frac{1}{4}\right)2 = \frac{3}{2}$

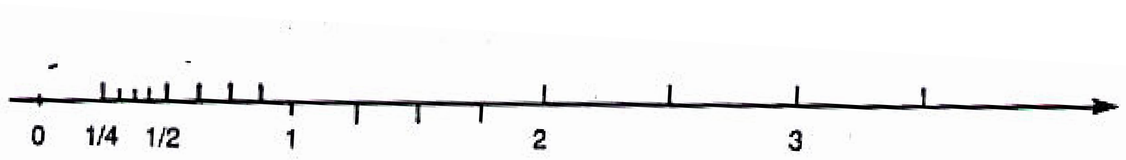
$0.111\ 2^{1} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right)2 = \frac{7}{4}$

$0.100\ 2^{2} = \frac{1}{2}\,4 = 2$

$0.101\ 2^{2} = \left(\frac{1}{2} + \frac{1}{8}\right)4 = \frac{5}{2}$

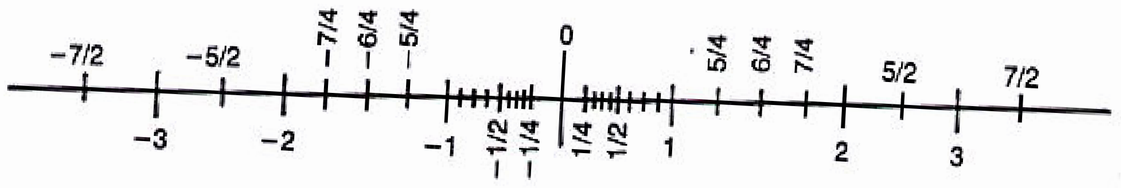
$0.110\ 2^{2} = \left(\frac{1}{2} + \frac{1}{4}\right)4 = 3$

$0.111\ 2^{2} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8}\right)4 = \frac{7}{2}$



Elementi positivi

F(2, 3, -1, 2) possiede 33 elementi



15

Errori assoluti e relativi

$$\textit{Sia} \quad x = m\beta^p, \quad \overline{x} = fl(x) = \overline{m}\,\beta^p$$

Errore assoluto: $\left| x - fl(x) \right|$

Errore relativo: $\dfrac{\left| x - fl(x) \right|}{\left| x \right|}$

16

Precisione di macchina

$$\frac{|x - fl(x)|}{|x|} \leq eps \qquad eps = \begin{cases} \beta^{1-t} & troncamento \\ \frac{1}{2}\beta^{1-t} & arrotondamento \end{cases}$$

È il più piccolo numero di macchina positivo tale che

fl(1+eps)>1

La formula fornisce una misura di quanto accuratamente i numeri reali possono essere "*approssimati*" da numeri finiti *F(β,t,L,U)* e fornisce quindi una misura della "*precisione del calcolatore*"

Precisione di macchina

Esempio: $\beta=10,\ t=5$

$x=1 \quad y=0.5\ 10^{-4}$ ⟵ **eps**

$arr(x+y)=fl(1.00005)=0.10001 \times 10^{1}$

$x=1 \quad y=0.4\ 10^{-4}$

$arr(x+y)= fl(1.00004)=0.10000 \times 10^{1}$

Lo Standard IEEE

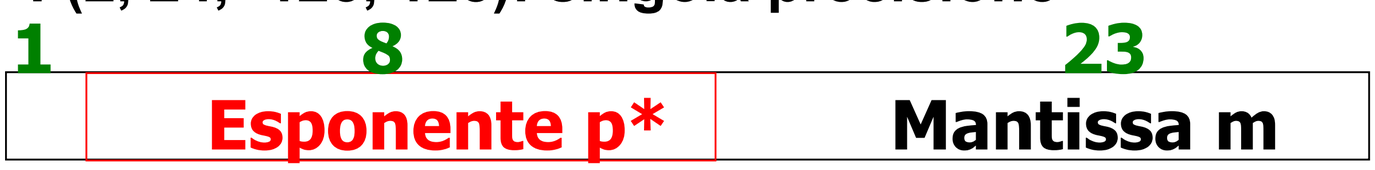
Lo standard IEEE 754 pubblicato nel 1985 definisce un sistema aritmetico floating point binario ed è adottato dalla maggior parte delle case costruttrici di elaboratori.

Esso adotta la tecnica di arrotondamento utilizzando il ROUND to EVEN e l'"hidden bit": poiché il primo bit di mantissa è sempre 1 possiamo fare a meno di memorizzarlo e guadagnare così un bit

Formati dello Standard IEEE per numeri floating point

Precisione semplice: 32 bit

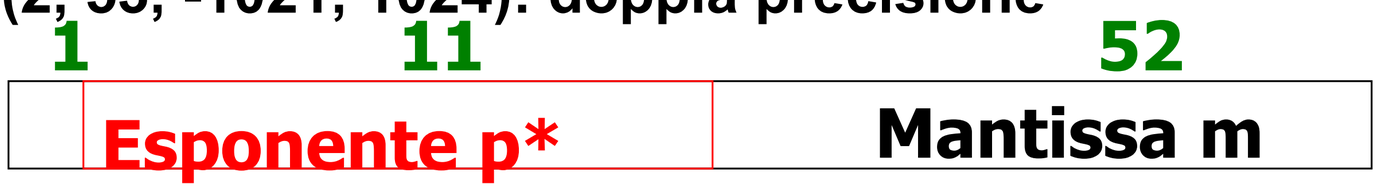
F(2, 24, -125, 128): singola precisione



Segno (0/1); $p^* = p + 127, \quad 0 \le p^* \le 255$; *m* ha 24 bit , l'1 iniziale è implicito

Precisione doppia: 64 bit

F(2, 53, -1021, 1024): doppia precisione



Segno (0/1); $p^* = p + 1023, \quad 0 \le p^* \le 2047$; *m* ha 53 bit , l'1 iniziale è implicito

Formati dello
Standard IEEE per floating point

Precisione semplice 32 bit base 2

$$eps = \frac{1}{2}2^{1-t} = \frac{2^{1-24}}{2} = \mathbf{5.9605e-008}$$

Precisione doppia 64 bit base 2

$$eps = \frac{1}{2}2^{1-t} = \frac{2^{1-53}}{2} = \mathbf{1.1102e-016}$$

Matlab usa aritmetica IEEE doppia precisione

Aritmetica finita (o floating point)

I risultati di operazioni aritmetiche tra numeri finiti generalmente non sono numeri finiti; pertanto in un calcolatore risulterà impossibile implementare correttamente le operazioni aritmetiche.

Operazioni in aritmetica floating point associano a due numeri finiti un terzo numero finito, ottenuto arrotondando l'esatto risultato dell'operazione aritmetica.

$$\bar{a} = fl(a), \quad \bar{b} = fl(b)$$
$$\bar{a} \quad op \quad \bar{b} = fl(\bar{a}\, op \bar{b}) = (\bar{a}\, op \bar{b})(1+\varepsilon)$$
$$op: \; +,-,*,/, \qquad con \quad |\varepsilon| \le eps$$

Cancellazione numerica

Nelle operazioni di sottrazione quando i due operandi sono "quasi uguali" si ha una ***perdita di cifre significative***

Es: **X1=0.14754326** **X2=0.14751742, t = 6, β = 10**

fl(X1)=0.147554 **fl(X2)=0.147252**

$fl(fl(X1)-fl(X2))=0.302000x10^{-3}$

mentre la vera differenza è

$X1-X2=0.302584x10^{-3}$

le ultime cifre della mantissa sono alterate!

Dovuto al fatto che dopo aver eseguito fl(x1)-fl(x2)=0.000302 la rappresentazione normalizzata ha introdotto 3 zeri alla fine della mantissa. Errore relativo ~ 10^{-3}

Cancellazione numerica: ESEMPIO

$a^2+b\,x + c = 0 \rightarrow$ $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

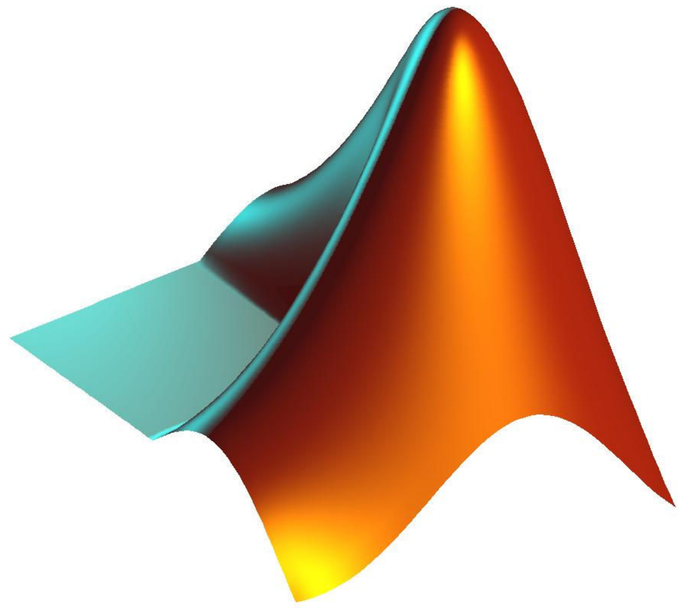
Errori di canc. num. in x_1 se $\sqrt{\Delta} \approx b$, o in x_2 se $\sqrt{\Delta} \approx -b$

Per eliminare il fenomeno, poiché:

$$x_1 \cdot x_2 = \frac{c}{a} \rightarrow x_2 = \frac{c}{x_1}$$

Se $b^2 \approx 4ac$ allora è bene usare maggior precisione

Introduzione a MATLAB



Tutorials

MATHWORKS web site:

http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml

Tutorial in italiano

http://guide.supereva.it/matlab/interventi/2001/11/78570.shtml
http://guide.supereva.it/matlab/interventi/2009/04/matlab-tutorial-5

Tutorial in inglese: MATLAB primer

http://math.ucsd.edu/~driver/21d-s99/matlab-primer.html
http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

Overview

- Introduzione e Overview
- Operazioni con matrici e funzioni comuni
- M Files: Script-Funzioni
- Programmazione
- Grafici 2D-3D, movie in MATLAB

History

- Il nome deriva da **MAT**rix **LAB**oratory
- MATLAB fu originariamente scritto da Dr. Cleve Moler, direttore scientifico a **The Math Works, Inc.,** per fornire un'interfaccia elementare ed immediata all'uso dei più diffusi pacchetti software per la risoluzione numerica di problemi di Algebra Lineare, quali le librerie LINPACK e EISPACK.
- La prima versione fu scritta nel **1970** per corsi di teoria delle matrici, algebra lineare e analisi numerica. MATLAB è quindi fondamentalmente costruito su un software sofisticato che ha come struttura di base una matrice che non richiede quindi un pre-dimensionamento.
- Fu utilizzato presto in molte università ed ebbe grande successo nella comunità di matematica applicata. **Jack Little**, un ingegnere, lo conobbe durante una visita a Moler presso la Stanford University nel 1983. Riconobbe le potenzialità commerciali del prodotto e si unì in società con Moler e Steve Bangert. Essi riscrissero MATLAB in C e fondarono **The MathWorks nel 1984** per continuarne lo sviluppo.

Foundation of Matlab

- ☺ *MATLAB è un linguaggio* ***case sensitive*** (la variabile di nome "**c**" è diversa da quella indicata con il nome "**C**")

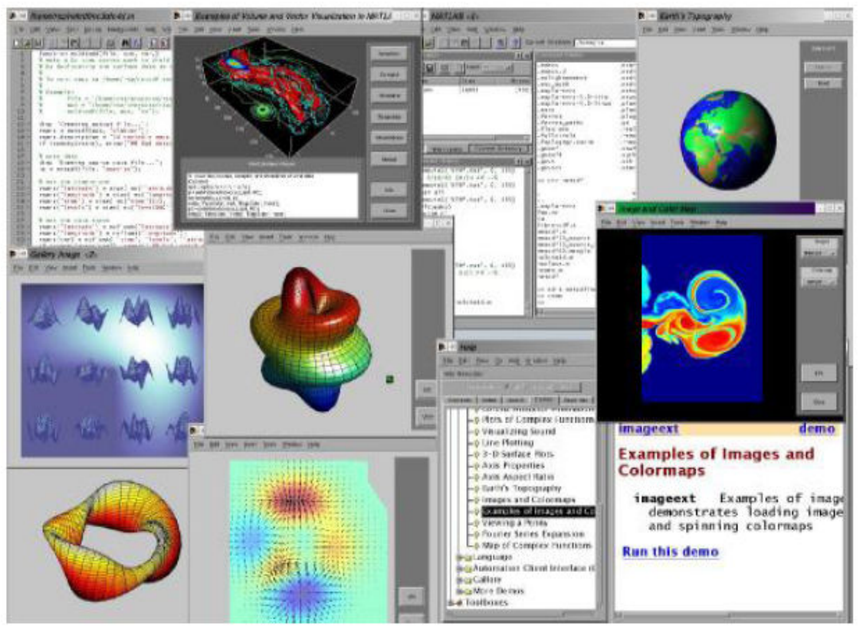
- ☺ MATLAB è un sistema interattivo. La struttura dati di base è costituita infatti dalla matrice: ciò significa che durante l'elaborazione ogni quantità viene trattata dall'ambiente di calcolo come una matrice di dimensione *mxn* (uno scalare è dunque memorizzato come una matrice *1x1*)

- ☹ Ci sono software gratuiti *open source* alternativi a MATLAB, in particulare **GNU Octave**, FreeMat, e **Scilab** che sono compatibili con il linguaggio MATLAB (ma non nell'ambiente MATLAB desktop).

Foundation of Matlab

MATLAB è un linguaggio interpretato, gli errori sono facili da trovare ma è molto lento in esecuzione.

- Grafica 2-D e 3-D per la visualizzazione di dati.
- Tools per costruire comode interfacce grafiche per l'utente.
- Funzioni per integrare gli algoritmi di MATLAB con applicazioni esterne e linguaggi, come C, C++, Fortran, Java, COM, e Microsoft Excel.



MATLAB Toolbox

- **Mapping Toolbox** fornisce gli strumenti e utilità per l'analisi di dati geografici e la creazione di mappe.
- **MATLAB Compiler** consente di convertire automaticamente i propri programmi MATLAB in applicazioni standalone e componenti software per condividerli con utenti finali. Le applicazioni e i componenti creati con il Compiler non necessitano di MATLAB per essere eseguiti.
- **Partial Differential Equation (PDE) Toolbox** contiene gli strumenti per lo studio e la soluzione di equazioni alle derivate parziali (PDE) in due dimensioni spaziali (2-D) e nel tempo. Un insieme di funzioni di comando e una interfaccia utente grafica consente di pre-elaborare, risolvere, e post elaborare PDE-generiche per una vasta gamma di applicazioni tecniche e scientifiche.
- **Symbolic/Extended Math Toolbox** fornisce gli strumenti per la risoluzione e la manipolazione di espressioni matematiche.
- **Image Processing Toolbox** fornisce una serie completa di algoritmi standard e strumenti grafici per l'elaborazione delle immagini, l'analisi, la visualizzazione e lo sviluppo di nuovi algoritmi. È possibile migliorare la qualità delle immagini con rumore o degradate, estrarre caratteristiche, analizzare forme e texture, e di registrare due immagini.

MATLAB ha le seguenti finestre di dialogo:

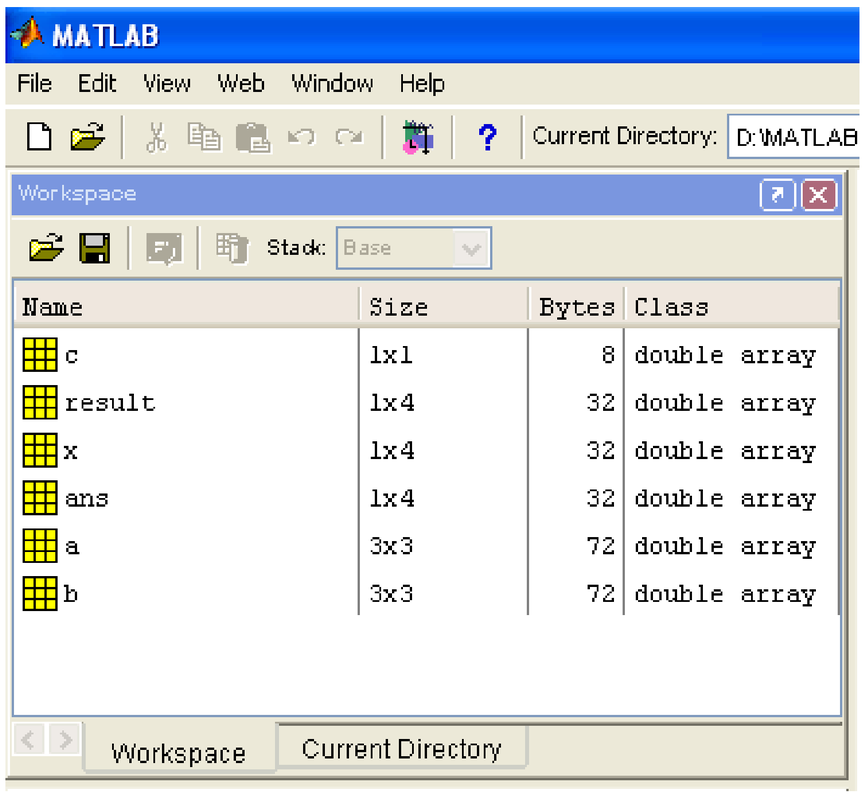
- Work-space

  Visualizza tutte le variabili definite.
- Command Window

  Per eseguire i comandi nell'ambiente MATLAB
- Command History

  Visualizza i comandi utilizzati

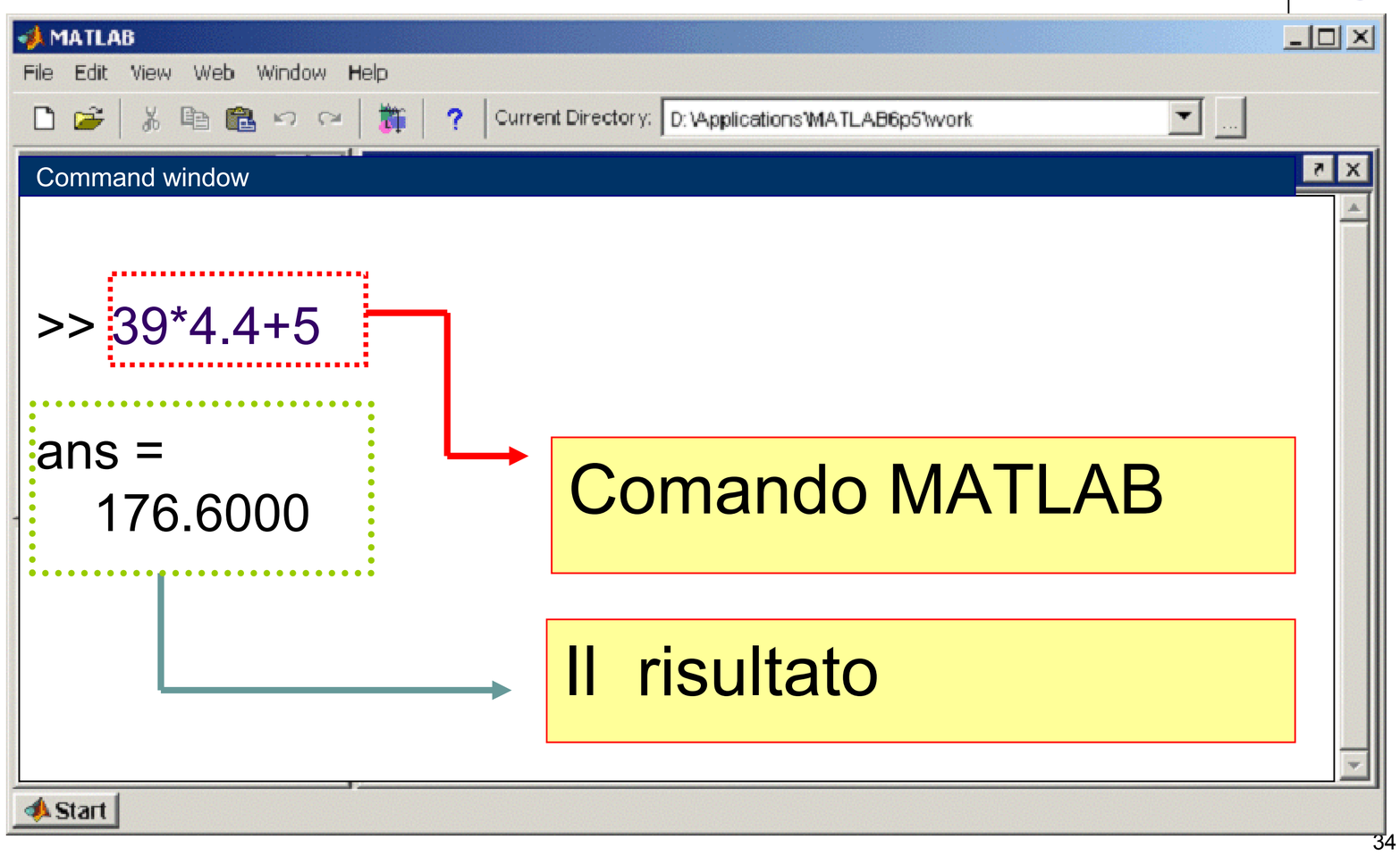
Matlab Workspace



Il Workspace di Matlab visualizza tutte le variabili definite insieme alla loro dimensione, occupazione di memoria e tipo.

33

Command Window



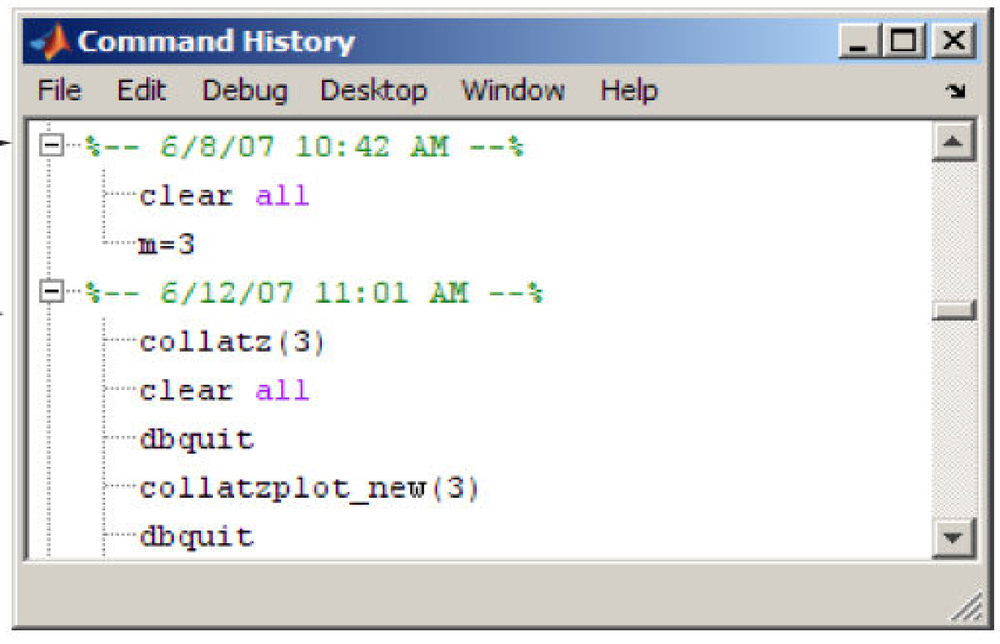
34

Command History

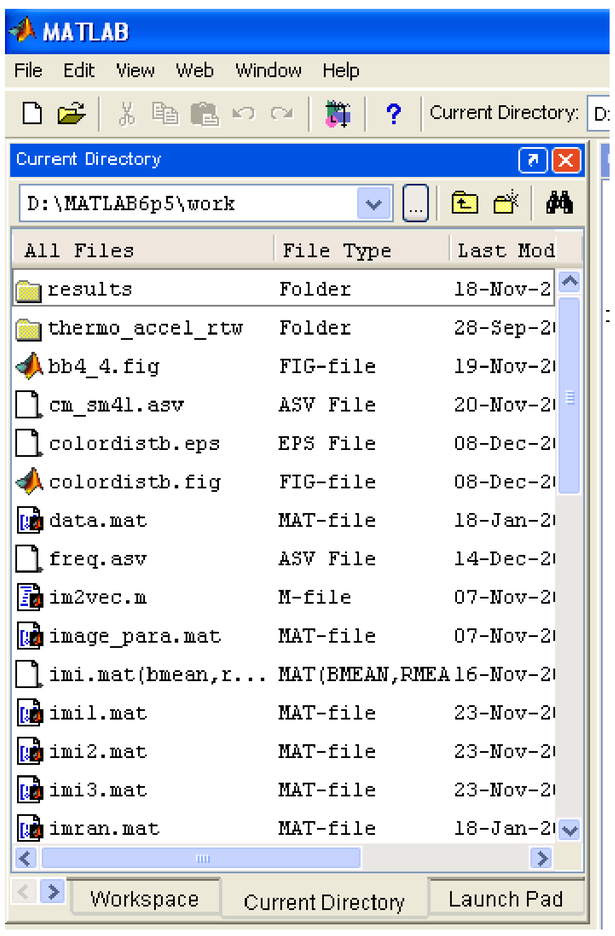
I comandi digitati nella Command Window sono memorizzati nella Command History. Dalla Command History, si possono vedere e cercare istruzioni utilizzate in precedenza, come pure copiare ed eseguire alcune di esse. Si può anche creare un M-file con alcune istruzioni selezionate.

Timestamp marks the start of each session.

Select one or more entries and right-click to copy, evaluate, or create an M-file from the selection.



Current Directory

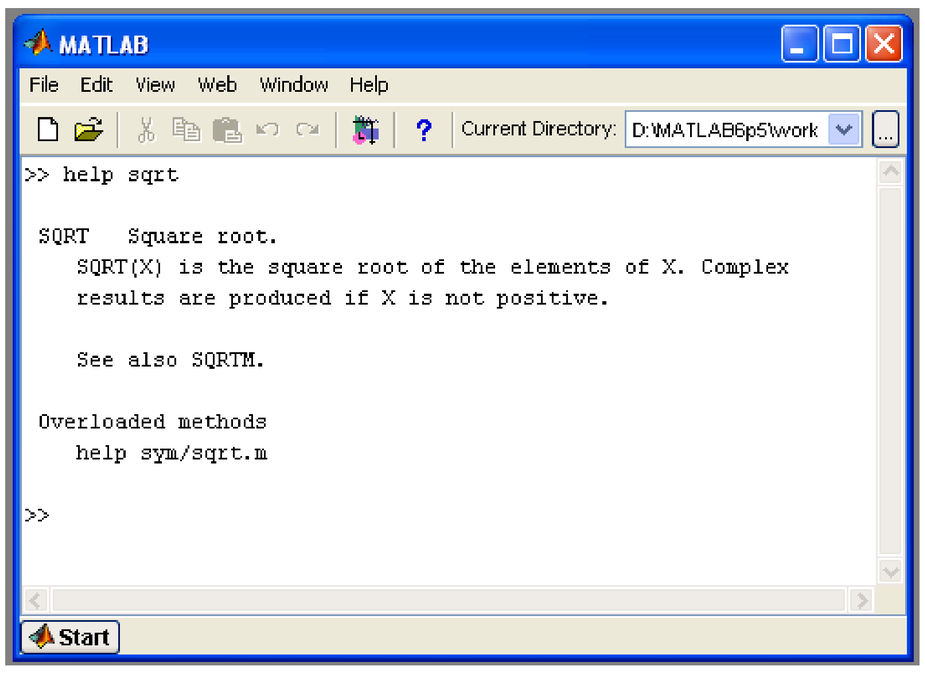


Fornisce un accesso rapido a tutti i files presenti nella directory scelta

Fornisce una breve descrizione (quando i files sono commentati) di ciascun M-file

Matlab Help

- L'Help di Matlab è uno strumento estremamente potente per imparare.
- L' Help non solo contiene informazioni teoriche ma mostra anche esempi per l'implementazione
- L'Help di Matlab può essere aperto usando il menu a tendina HELP



Matlab Programming

Due approcci

- Digitare i comandi nella Command Window
  - Buono per programmi corti
  - I comandi devono essere forniti di nuovo ogni volta che si vuole rieseguire una simulazione

- Programmare usando gli .m-files: **script**
  - Buono per programmi lunghi e complessi
  - Permettono all'utente di salvare tutti i comandi scritti negli .m-files

Operazioni di base

Simbolo	Operazione	MATLAB
^	Elevamento a potenza: a^b	**a^b**
*	Moltiplicazione: ab	**a*b**
/	Divisione a destra: a/b = $\frac{a}{b}$	**a/b**
\	Divisione a sinistra: a\b = $\frac{b}{a}$	**a\b**
+	Addizione: a + b	**a+b**
-	Sottrazione: a-b	**a-b**

1/5=5\1=0.2

39

Operatori relazionali e logici

>	Maggiore di
>=	Maggiore o uguale a
<	Minore di
<=	Minore o uguale a
==	uguale
~=	Diverso da

40

(per ~ in una tastiera italiana premere ALT e digitare 126 nel tastierino numerico)

Operatori relazionali e logici

&	AND
|	OR
~	NOT

Variabili

Una variabile è un simbolo usato per contenere un valore

MATLAB ha alcune regole per i nomi delle variabili:

- I nomi delle variabili sono **case sensitive:** BOB, Bob, bob, BoB sono tutti diversi!
- I nomi delle variabili possono contenere fino a 31 caratteri. Ogni carattere addizionale è ignotato.
- I nomi delle variabili devono **iniziare con una letterea**, seguiti da altre lettere, cifre o **underscores (no blank)**. Simboli di punteggiatura non sono consentiti, perchè molti di essi hanno un significato particolare in MATLAB

Variabili

- Alcuni nomi non possono essere usati come nome di variabile: *for, end, if, while, function, return, elseif, case, otherwise, switch, continue, else, global, break ……..*

- Se si tenta di usare una parola riservata come nome di variabile, MATLAB manda una segnalazione di errore

- Naturalmente, si possono usare parole simili semplicemente mettendo una o più lettere maiuscole

- MATLAB ha un numero di variabili e costanti speciali.

Variabili e costanti speciali

Nome	**Descrizione**
ans	Nome di Default usato per I risultati
eps	Precisione di macchina
i,j	Unità immaginaria, sqrt(-1)
Inf	infinito
NaN	Risultato numerico non definito
pi	π
realmax	Massimo numero reale f.p. nel computer
Realmin	Minimo numero reale f.p. nel computer

Istruzioni MATLAB

Istruzione MATLAB	**Note**
a1=5.66	*a1* è uno scalare
a1=[5.66]	Modo alternativo
A=[3 6.3, sqrt(2)]	A è una matrice 1X3 con elementi 3 , 6.3 e sqrt(2). La virgola o lo spazio sono usati per separare gli elementi in una riga
C=[1 4]	C è una matrice 2X1 con elementi 1e 4.
C = [1 ; 4]	Punto e virgola è usato per indicare la fine di una riga.
X=1:7	Equivale a X=[1 2 3 4 5 6 7]

45

Istruzioni MATLAB

Istruzione MATLAB	**Note**
V=[2 3 5 3 3 8]	$V = \begin{bmatrix} 2 & 3 & 5 \\ 3 & 3 & 8 \end{bmatrix}$
C=[1:3:11]	C=[1 4 7 10]
u=linspace(0,1,5)	u=[0 0.25 0.5 0.75 1]
Z=4\8	Z=2
V=eye(2)	$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
V = zeros(2,3)	$V = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

46

Gli elementi di una matrice

```
>> A= [ 1 2 3 ; 4  5  6 ;  7 8 9];

>> x = A ( 2, 3 )      %A(<riga>,<colonna>)
    x =
        6
>> y = A ( 2 , : )          % seleziona la 2nd riga
    y =
        4    5    6

>> z = A ( 1:2 , 1:3 )     % seleziona una sottomatrice
    z =
        1    2    3
        4    5    6
```

```
A =
    1    2    3
    4    5    6
    7    8    9
```

Come selezionare l'ultima colonna?

A(:,end)

Sintassi di base

Operatori matematici su array

Matematica	
+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
^	Potenza

Array	
+	Addizione
-	Sottrazione
.*	Moltiplicazione
./	Divisione
.^	Potenza

Attenzione alle dimensioni di una matrice nel calcolo matriciale

Esempio: A matrice 2x1 , B matrice 2x2

A*B → operazione non valida

B*A → operazione valida

Operazioni elemento per elemento

```
>> B = [ 1  2  ;  3  4 ] ;
B =
     1     2
     3     4
```

B

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
>> C = B * B        % equivalente a B^2
C =
     7    10
    15    22

>> D = B .* B       % o B.^2  il punto denota l'operazione elemento per elemento
D =
     1     4
     9    16
```

Che differenza tra *B / B* e *B ./ B* ?

Operatori e funzioni

Per vettori, sum(x) è la somma degli elementi di X. Per matrici, sum(x) è un vettore riga con la somma per colonne.

```
>> sum ( A )
          ans =
                    12    15    18
>>  sum ( ans )      % equivalente  a  sum(sum(A))
          ans =
                    45
>> A'                          % equivalente a  transpose(A)
     ans =
               1     4     7
               2     5     8
               3     6     9
>> diag(A)
          ans =
               1
               5
               9
```

```
A =
     1     2     3
     4     5     6
     7     8     9
```

Formato di output

I risultati possono essere visualizzati in modi diversi:

SHORT	Punto fisso, con 4 cifre dopo il punto decimale.
SHORT E	Formato Floating point, con 4 cifre dopo il punto decimale.
SHORT G	Il meglio tra punto fisso o floating point, con 4 cifre dopo il punto decimale.
LONG	Punto fisso, con 14/15 cifre dopo il punto decimale.
LONG E	Formato Floating point format, con 14/15 cifre dopo il punto decimale.
LONG G	Il meglio tra punto fisso o floating point,, con 14/15 cifre dopo il punto decimale.
RAT	Frazione di interi.

Esempio: >>y = 8/6

>format short
1.3333
>>format short e
1.3333E+000
>>format short g
1.3333
>>format long
1.33333333333333
>>format long e
1.33333333333333E+000
>>format long g
1.33333333333333
>>format rat
4/3

default : format short

Controllo Output

Funzione ***disp***: Utile per controllare l'output

disp: visualizza l'array.

disp(X) visualizza il contenuto dell'array, senza visualizzare il nome dell'array. Lo stesso risultato può essere ottenuto omettendo il punto e virgoal dopo il nome dell'array. Se X è una stringa, è visualizzato il testo.

ESEMPIO:

>> disp(X) visualizza il contenuto dell'array, senza visualizzare il nome dell'array

>> disp('The value of X is:') visualizza la stringa

The value of X is:

Controllo Input

Funzione ***input***: usata per un input interattivo.

R = input('How many apples') dà all'utente il prompt visualizzando il testo e attende l'input da tastiera.

L'input può essere ogni espressione MATLAB, che viene valutata, usando le variabili nel workspace corrente, e il risultato è memorizzato in R. Se l'utente preme il tasto return senza digitare nulla, INPUT restituisce una matrice vuota.

R = input('What is your name','s') dà all'utente il prompt visualizzando il testo e attende in input da tastiera una stringa. L'input digitato non viene valutato; i caratteri sono semplicemente memorizzati come una stringa MATLAB.

Controllo Input e Output

La funzione **Menu**: usata per un input interattivo scelto da un menù:

k = menu(‘title’,’option1’,’option2’,…)

ESEMPIO:

```
scelta=['A', 'B', 'C'];
k=menu(‘Cosa scegli?',scelta(1),scelta(2),scelta(3));
disp([‘Hai scelto: ',scelta(k)])
```


Caricare e salvare dati

Durante il lavoro con MATLAB, si può avere la necessità di salvare vettori e matrici definite nel programma. Per salvare il file nella directory di lavoro, digitare

save filename

dove "filename" è un nome scelto dall’utente.

Per recuperare I dati nel seguito, digitare

load filename

```
>>load data.txt;        % il nome dell’array è data
>>load(‘new.txt’);     % il nome dell’array è new
>>mydata =  load(‘proj3.txt’);  % il nome dell’array è mydata
```

Programmare in MATLAB

M-file: lista di comandi e istruzioni MATLAB eseguiti in ordine, memorizzata come file con l'estensione ".m", cioè ***filename.m***

- Ci sono due tipi di programmi MATLAB:

script files

```
%  script file
P=[1 3 2]
roots(P)
```

function files

```
function [y]=myfun(x)
%  function file myfun.m
y=x.^2-sin(x)
```

57

Script o function????

Script file

- Lista di istruzioni MATLAB
- Le variabili sono globali
- Si eseguono digitando il nome del file

Function file

- Inizia con ***function***
- Lista di istruzioni MATLAB
- Le variabili sono locali

58

Script o function????

Usare uno script file quando si ha una lunga sequenza di istruzioni per risolvere un problema

Si esegue il programma

- Digitando il nome nella command window
- Dai tools nell'editor window

Usare un function file quando una sequenza di istruzione si rende necessaria più volte, anche con diversi valori in ingresso

Sintassi

function *[out1, out2, ...] = funname(in1, in2, ...)*

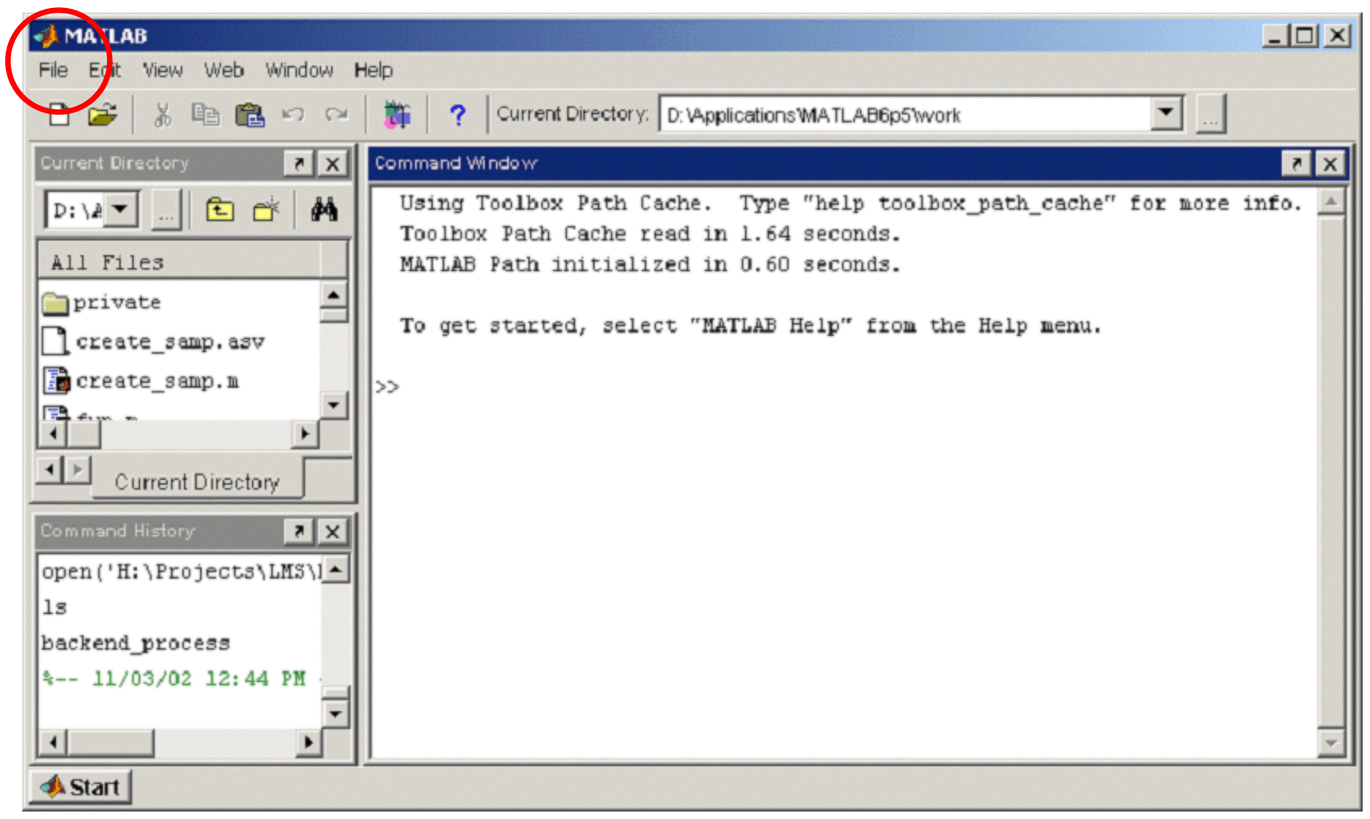
Descrizione

function [out1, out2, ...] = funname(in1, in2, ...) definisce function ***funname*** che accetta in ingresso ***in1***, ***in2***, …etc. e restituisce in uscita ***out1***, ***out2***,… etc.

59

Creare M-files

Selezionare FILE → OPEN → NEW → M-files



60

Esempio

Scrivere una function per calcolare la somma dei primi *n* numeri naturali

- Input: *n*
- Output: *somma*
- Function name: *sommanaturali*

Esempio

```
function [a,p] = polyGeom(s,n)
  % polyGeom  Calcola area e perimetro di un poligono  regolare %
  % Synopsis:  [a,p] = polyGeom(s,n)
  %
  % Input:  s = lunghezza di un lato del poligono
  %         n = numero dei lati del poligono
  %
  % Output: a = area del poligono
  %         p = perimetro del poligono
r = s/(2*tan(pi/n));   %  "raggio" del poligono
a = area(r,n);
p = perimeter(r,n);

% ============ function secondaria "area"
function a = area(r,n)
  % area  calcola l' area di un poligono con n lati e raggio r
a = n*r^2*sin(pi/n);

% ============ function secondaria "perimeter"
function p = perimeter(r,n)
  % perimeter calcola il perimetro di un poligono con n lati e raggio r
p = n*2*r*tan(pi/n);
```

Variabili globali

Se si vuole che più functions condividano una copia di una variabile, basta dichiarare la variabile come globale in tutte le functions. Fare la stessa cosa nella linea di comando se si vuole che la variabile sia presente nel workspace .

La dichiarazione global deve essere presente prima dell'uso nella function.

```
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

```
global GRAVITY
GRAVITY = 32;
y = falling((0:.1:5)');
```


Nomi di Function come argumenti

```
function y = humps(x)
y = 1./((x-.3).^2 + .01) + 1./((x-.9).^2 + .04) - 6;
```

Valutare questa function in un insieme di punti nell'intervallo
$0 \le x \le 1$ con

```
x = 0:.002:1;
y = humps(x);
```

Quindi eseguire il grafico con

```
plot(x,y)
```

Se si cerca lo zero della funzione: **z = fzero(@humps,.5)**

Se si vuole calcolare l'area delimitata dalla curva **Q = quadl(@humps,0,1)**

Function eval

Valuta function

Sintassi

[y1, y2, ...] = feval(fhandle, x1, ..., xn)

[y1, y2, ...] = feval(function, x1, ..., xn)

Descrizione

[y1, y2, ...] = feval(fhandle, x1, ..., xn) valuta la function, fhandle, usando gli argumenti da x1 a xn.

```
>> feval('sin',0.6*pi)
ans =
    0.9511
```

⟺

```
>> sin(0.6*pi)
ans =
    0.9511
```


Struttura If

L'istruzione *if* valuta una espressione logica ed esegue un gruppo di istruzioni quando l'espressione è vera. Le parole chiave *elseif* e *else* permettono di eseguire un gruppo alternativo di istruzioni corrispondenti al caso in cui l'espressione in esame sia falsa. La parola chiave *end*, che chiude l'if, termina l'ultimo gruppo di istruzioni.

Forma generale

```
istruzioni
if  condizione
    istruzioni
else

end
```

```
If (x>0)
    sign=1
elseif (x==0)
    sign=0
else
    sign=-1
end
```

Switch e case

L'istruzione *switch* esegue gruppi di istruzioni bassandosi sul valore di una variabile o di una espressione. Le parole chiave *case* e *otherwise* delimitano i gruppi. Ci deve sempre essere un *end* a chiudere lo switch iniziale.

Forma generale:

```
switch   switch_expr
 case   case_expr
      istruzioni
 case {case_expr1, case_expr2, ….}
      istruzioni
 otherwise
      istruzioni
end
```

```
switch (rem(n,4)==0) +
(rem(n,2)==0)
case 0
        M = odd_magic(n)
case 1
        M = single_even_magic(n)
case 2
        M =
double_even_magic(n)
otherwise
        error('This is impossible')
end
```

67

For loops

Il ciclo ***for*** ripete gruppi di istruzioni per un numero predeterminato di volte. Un ***end*** finale delimita le istruzioni del ciclo.

Forma generale:

```
for indice=inizio: incremento: fine
   istruzioni
end
```

```
s=0
for i=1:3:11
   s=s+i
end
```

68

While end

Il ciclo while ripete un gruppo di istruzioni un numero indefinito di volte sotto il controllo di una condizione logica. Un ***end*** finale delimita le istruzioni del ciclo.

Forma generale:

while condizione
    istruzioni
end

```
n=1;
while (prod(1:n) < 1.e10)
n=n+1;
end
disp(n )
```


Grafici in MATLAB

MATLAB permette di graficare in una finestra speciale definita con il comando ***figure***. Per creare un grafico è necessario definire un sistema di coordinate. Quindi, ogni grafico ha degli assi coordinati che contengono la figure.

Comando	**Descrizione**
plot(x,y)	Genera il grafico.
title('text')	Inserisce il titolo nel grafico.
xlabel('text')	Aggiunge un'etichetta all'asse orizzontale.
ylabel('text')	Aggiunge un'etichetta all'asse verticale.
legend('text')	Inserisce una legenda
figure (n)	Apre una nuova figura nella finestra numero n
hold on	Mantiene la stessa finestra di graficazione

Opzioni grafiche

Colori	
blu	b
nero	k
verde	g
rosso	r
giallo	y
bianco	w
ciano	c
magenta	m

Specif. linea	
Solida(default)	-
Tratteggiata	--
Punteggiata	:
Tratto punto	-.

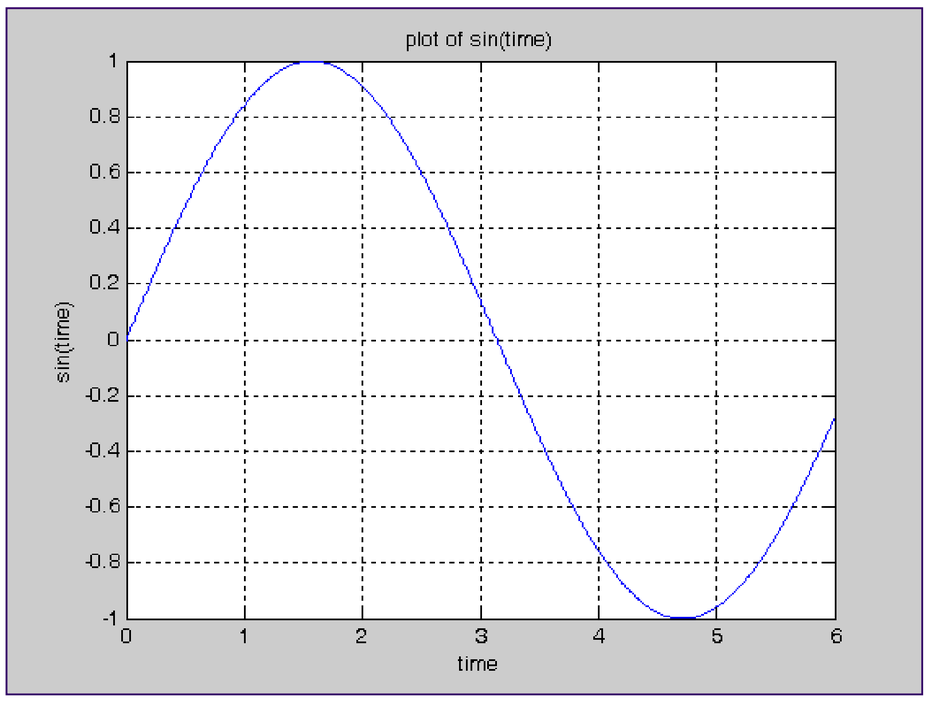
Marker	
Punto	.
Segno più	+
Stelletta	*
Cerchio	o
Croce	x

71

Esempio

time=[0:0.01:6]
Y=sin(time)

```
plot(time,Y)
xlabel('time')
ylabel('sin(time) ')
title(' plot of sin(time) ')
grid
```



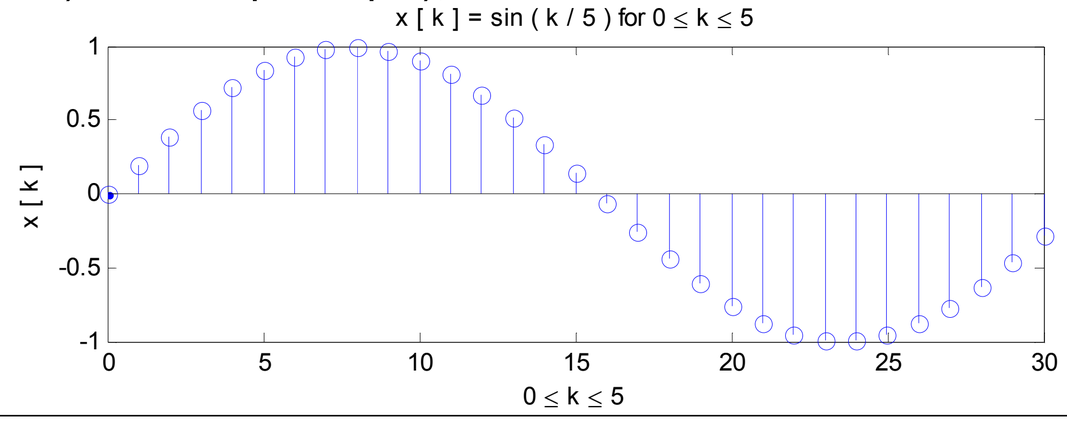
72

Grafici discreti: stem

La funzione *stem* è molto simile alla plot. E' usata per graficare sequenze temporali discrete.

Esempio:

```
>> k = [ 0 : 30 ] ;
>> x = sin ( k / 5 ) ;
>> stem ( k, x)
>> xlabel('0 \leq k \leq 5');
>> ylabel('x [ k ]');
>> title('x [ k ] = sin ( k / 5 ) for 0 \leq k \leq 5');
```

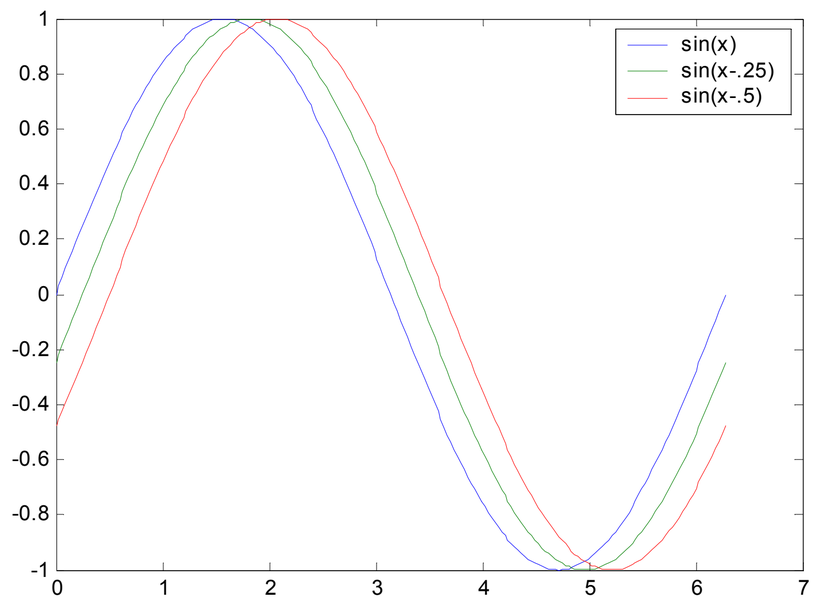


Più grafici in una finestra

Diverse coppie di dati x-y creano grafici multipli con una singola chiamata alla plot.

Esempio:

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> y2 = sin(x-.25);
>> y3 = sin(x-.5);
>> plot(x,y,x,y2,x,y3)
>> legend('sin(x)','sin(x-.25)','sin(x-.5)')
```



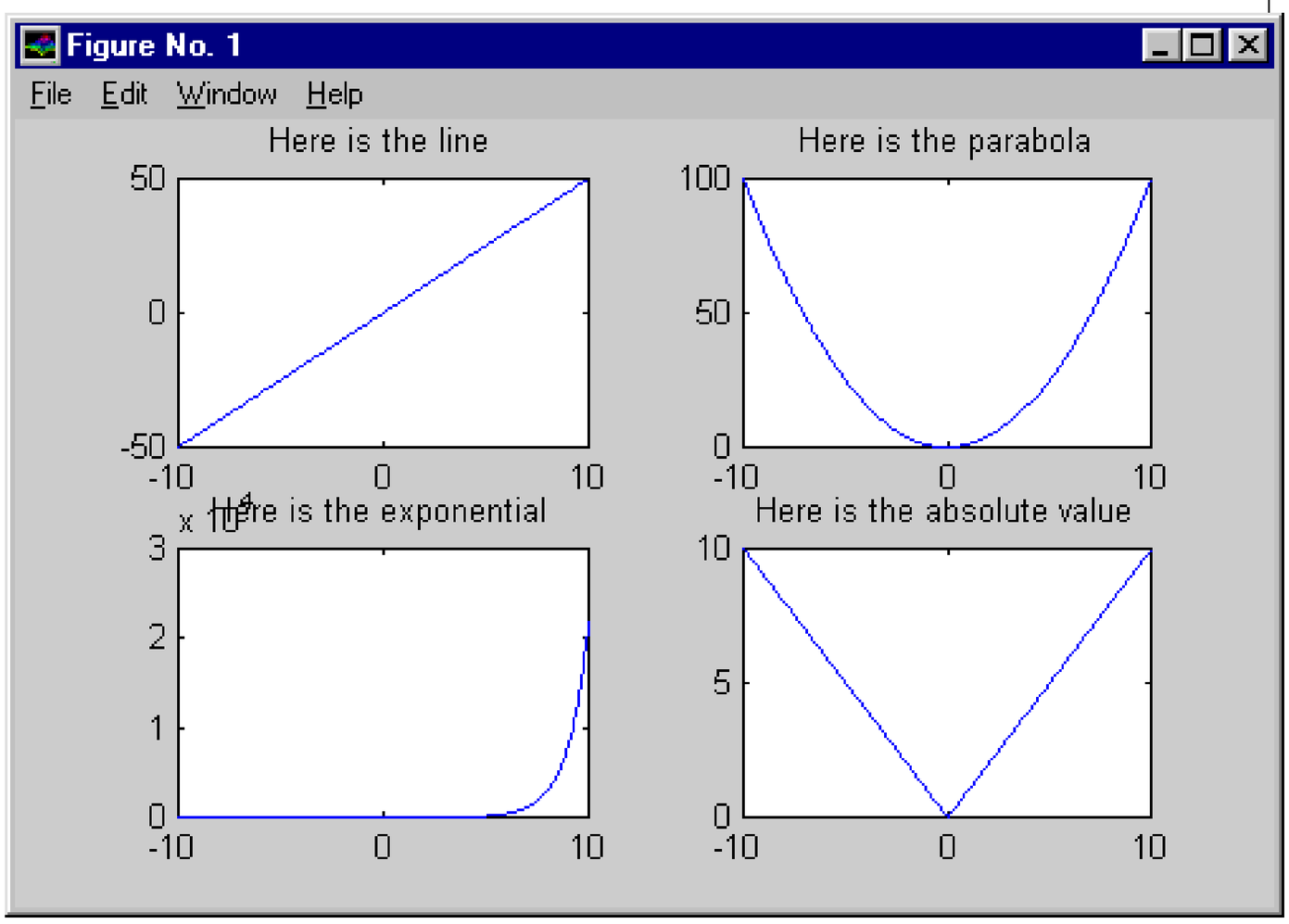
74

Subplot

```
>> t=-10:.01:10;
>> y1=t;
>> y2=t.^2;
>> y3=exp(t);
>> y4=abs(t);
>> subplot(221)
>> plot(t,y1),
>> title('Here is the line')
>> subplot(222)
>> plot(t,y2),
>> title('Here is the parabola')
>> subplot(223)
>> plot(t,y3),
>> title('Here is the exponential')
>> subplot(224)
>> plot(t,y4),
>> title('Here is the absolute value')
```

Il comando *subplot* permette di visualizzare più grafici distinti nella stessa finestra grafica.
Digitando *subplot(m,n,p)*
si suddivide la finestra in m x n finestre per grafici più piccoli e si seleziona la finestra p-esima per il grafico corrente.
I grafici sono numerati lungo la prima riga, poi lungo la seconda e così via.

Subplot



Esportare un grafico

Esportare un grafico significa creare un file standard nel formato per le immagini (come EPS o TIFF), che poi può essere importato in altri documenti e applicazioni come word processors, pacchetti software di graficazione, ecc.

Questo esempio esporta un grafico in un file EPS file con le seguenti caratteristiche:

• la dimensione dell'immagine quando importata in un documento word deve essere di 4 pollici (inches) di ampiezza (wide) e 3 inches in altezza.

• tutti i testi riportati nella figura devono avere un'ampiezza di 8 punti.

Specifica le dimensioni del grafico:

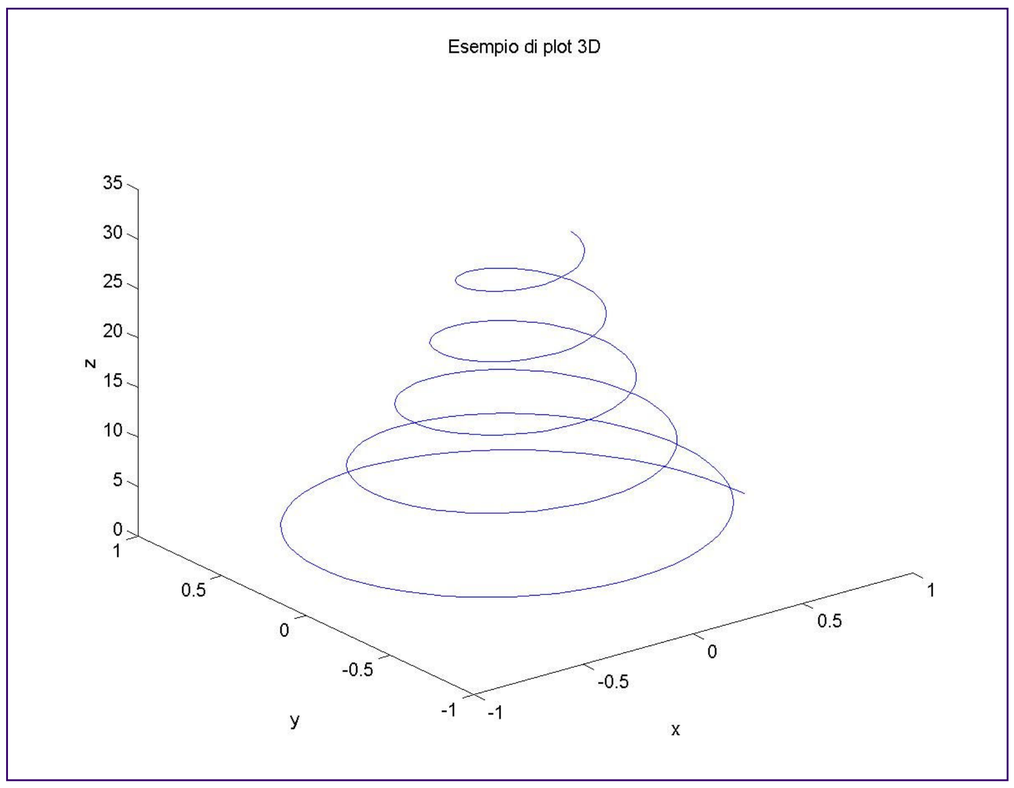
Per scegliere le dimensioni, usare la finestra di dialogo *Export Setup* (selezionare *Export Setup* dal menu File della figura). Poi selezionare *4* nella lista *Width* e *3* dalla lista *Height* .

Grafici 3D

Comando	Descrizione
plot3(x,y,z)	visualizza un grafico tridimensionale di un insieme di dati (una curva in forma parametrica)
mesh(X,Y,Z)	Crea una mesh
surf(X,Y,Z)	Crea una superficie 3D ombreggiata
[X,Y]=meshgrid(x,y)	Trasforma il dominio specificato da due vettori x, y in due matrici X e Y usate per valutare la funzione di due variabili. Le righe di X sono copie del vettore x e le colonne di Y sono copie del vettore y.

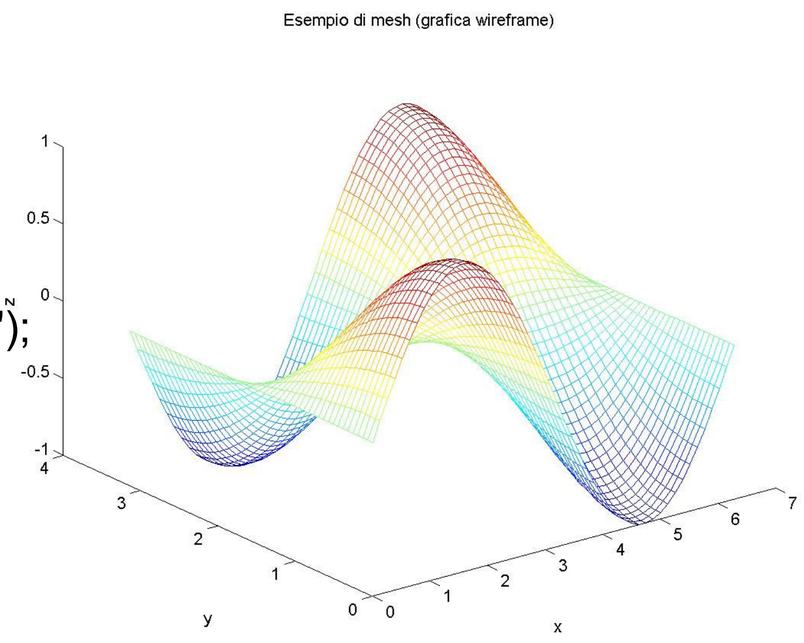
plot3

```
>>t=0:0.1:10*pi;
>>x=exp(-t/20).*cos(t);
>>y=exp(-t/20).*sin(t);
>>z=t;
>>plot3(x,y,z);
>>title('Esempio di plot 3D');
>>xlabel('x');
>>ylabel('y');
>>zlabel('z');
```



mesh

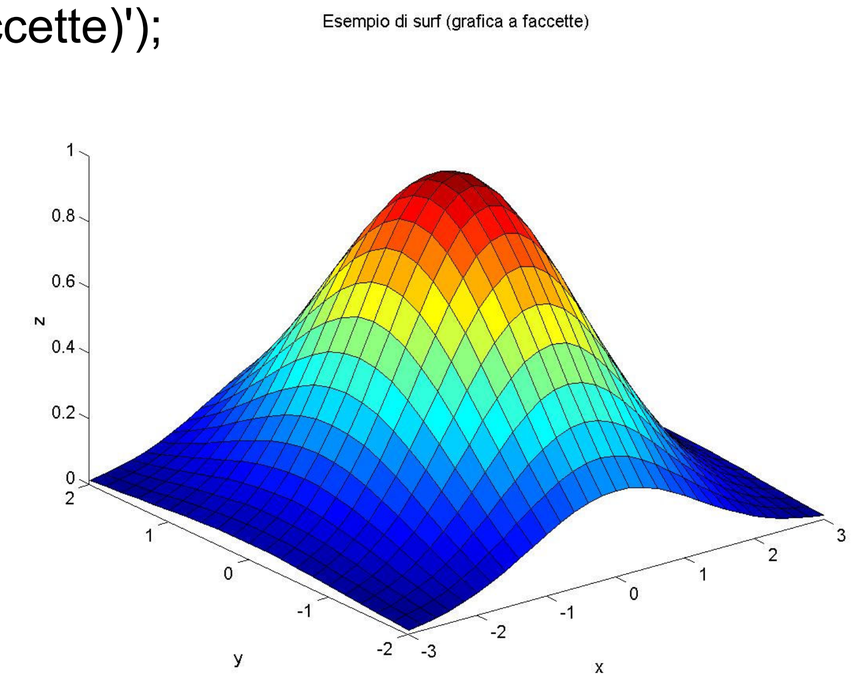
```
>>x=linspace(0,2*pi,50);
>>y=linspace(0,pi,50);
>>[X,Y]=meshgrid(x,y);
>>Z=sin(X).*cos(Y);
>>mesh(X,Y,Z);
>>title('Esempio di mesh (grafica wireframe)');
>>xlabel('x');
>>ylabel('y');
>>zlabel('z');
```



In informatica, **wireframe** o **wire frame model** (lett. *modello in fil di ferro*) indica un tipo di rappresentazione grafica da computer di oggetti tridimensionali, detta anche *vettoriale*. Con questo metodo vengono disegnati soltanto i bordi dell' oggetto, il quale di fatto resta trasparente al suo interno (sembrando, appunto, costruito con il fil di ferro). Questo metodo richiede calcoli molto più semplici rispetto alla rappresentazione di superfici, ed è quindi considerevolmente più veloce.

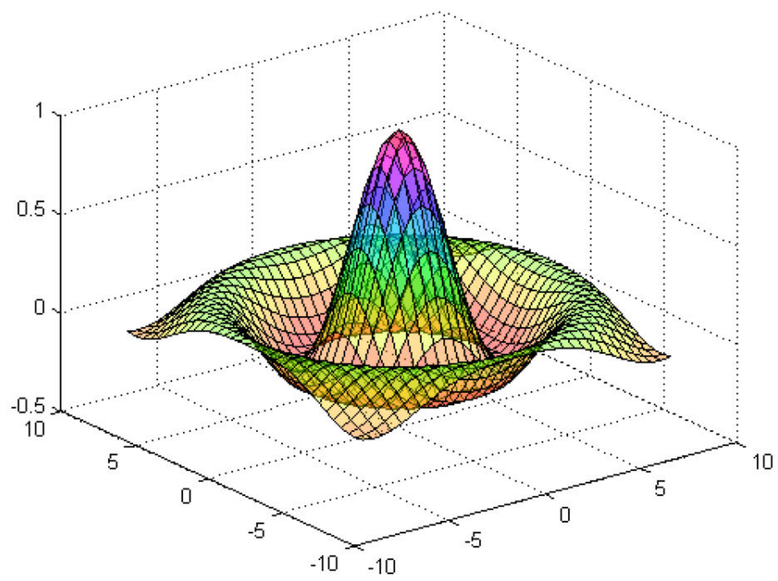
surf

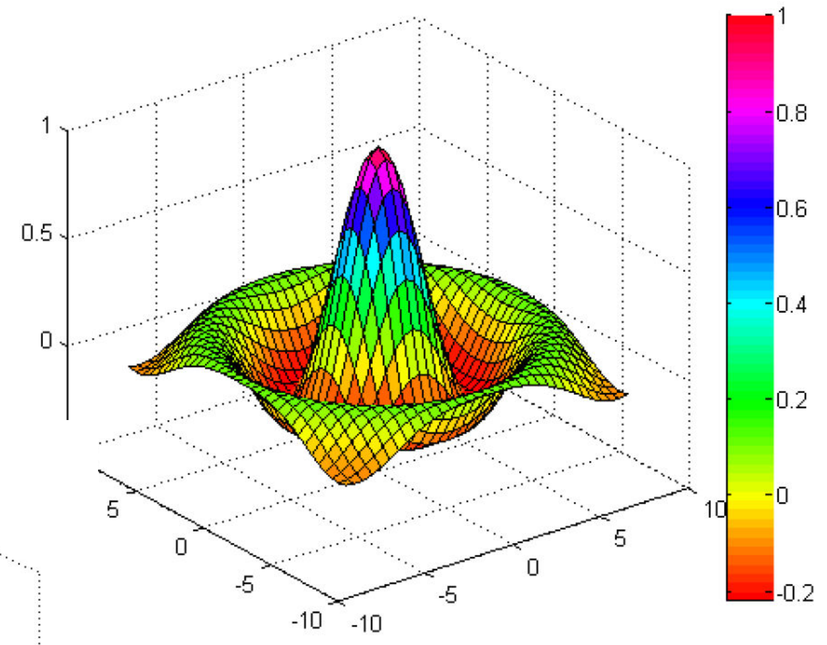
```
>>[X,Y]=meshgrid(-3:.2:3,-2:.2:2);
>>Z=exp(-(X.^2+Y.^2)/3);
>>surf(X,Y,Z);
>>title('Esempio di surf (grafica a faccette)');
>>xlabel('x');
>>ylabel('y');
>>zlabel('z');
```



surf

surf(X,Y,Z)
colormap hsv
colorbar

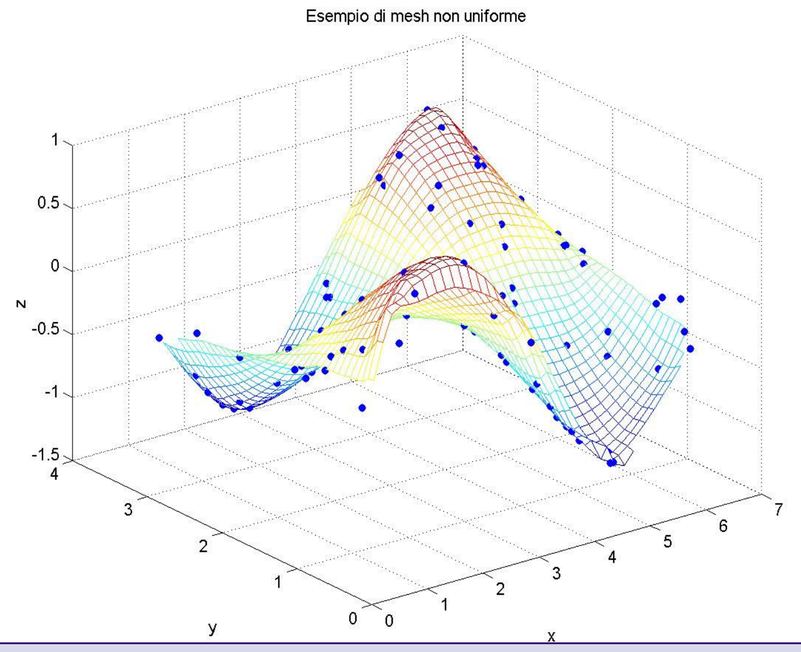




surf(X,Y,Z)
colormap hsv
alpha(.4)

Esempio

```
>>x=rand(100,1)*2*pi;
>>y=rand(100,1)*pi;
>>z=sin(x).*cos(y);
>>xlin=linspace(min(x),max(x),40);
>>ylin=linspace(min(y),max(y),40);
>>[X,Y]=meshgrid(xlin,ylin);
>>Z=griddata(x,y,z,X,Y,'cubic');
>>mesh(X,Y,Z);
>>title('Esempio di mesh non uniforme');
>>hold on;
>>plot3(x,y,z,'.','MarkerSize',15);
>>xlabel('x');
>>ylabel('y');
>>zlabel('z');
>>grid on;
```



Interpolazione di dati(x,y,z) non uniformi, Visualizza la funzione interpolante nei punti della griglia

Movie

Comando	Descrizione
getframe	Memorizza ogni singolo plot come un frame
movie()	Mostra i frame memorizzati

Esempio

```
fig1=figure(1);
numframes=8;
A=moviein(numframes,fig1);
x=linspace(0,1,101);
for i=1:numframes
   Y=x.^(i/4);
   plot(x,Y); % plot command
   % add axis label, legends, titles, etc. in here
   A(:,i)=getframe(fig1);
end
```

movie(fig1,A,4,3)

Formati del movie

Sfortunatamente Matlab salva il movie in un formato (.mat) che impegna tanta memoria: ogni volta che si salva un frame del movie, Matlab crea una mappa dei pixel della finestra del plot e la salva in una colonna della matrice del movie.

Si può convertire il movie nel formato *.avi,* per poterlo guardare al di fuori di Matlab, utilizzando il comando

>> movie2avi( )

Formati del movie

Un'altra possibilità è quelle di convertire il movie nel formato *.mpeg*
Per creare questo formato in Matlab è necessario utilizzare un programma ausiliario gratuito, *MPGWRITE* che si può scaricare dal sito di Matlab.
Dopo aver installato mpgwrite, per convertire un movie in .mpeg basta dare il comando:

```
>> mpgwrite(A, jet, 'movie.mpeg');
```

dove *A* è la matrice contenente il movie, *jet* (Red to Green to Blue) indica il set di colori da usare, mentre la stringa `*movie.mpg*' è il nome del file.mpeg

MEX-files

I mex files sono file che permettono di combinare subroutine scritte in C, C++, Fortran con file Matlab (MEX=Matlab EXECUTABLE)

Quando un MEX-file viene lanciato, i codici non scritti in Matlab vengono caricati come se fossero funzioni interne.

Matlab possiede diverse funzioni che permettono di trasferire le informazioni dai MEX-files e Matlba stesso

MEX-files: un esempio

Il seguente programma mostra la struttura fondamentale di una MEX-function:

```
#include "mex.h" /* Always include this */
void mexFunction(int nlhs, mxArray *plhs[ ],    /* Output variables */
int nrhs, const mxArray *prhs[ ])               /* Input variables */
{
mexPrintf("Hello, world!\n");
return;
}
```

Il codice va creato utilizzando l'Editor di Matlab e salvato in un file con estensione *.c* (hello.c)

MEX-files: un esempio

Successivamente avviene la compilazione: digitare in Command Window:

```
>> mex hello.c
```

Dopo la compilazione è possibile richiamare il file come un qualsiasi M-file di Matlab:

```
>> hello
```

Per una trattazione approfondita dell'argomento:

- http://www.mathworks.com/support/tech-notes/1600/1605.html
- http://www.math.ucla.edu/~getreuer/cmex.pdf