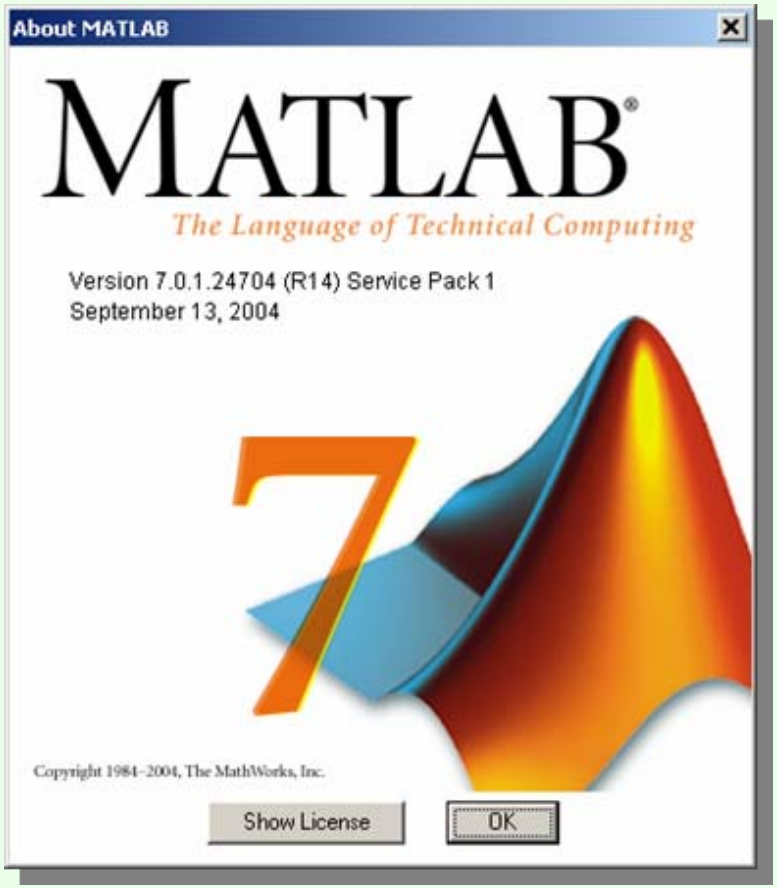
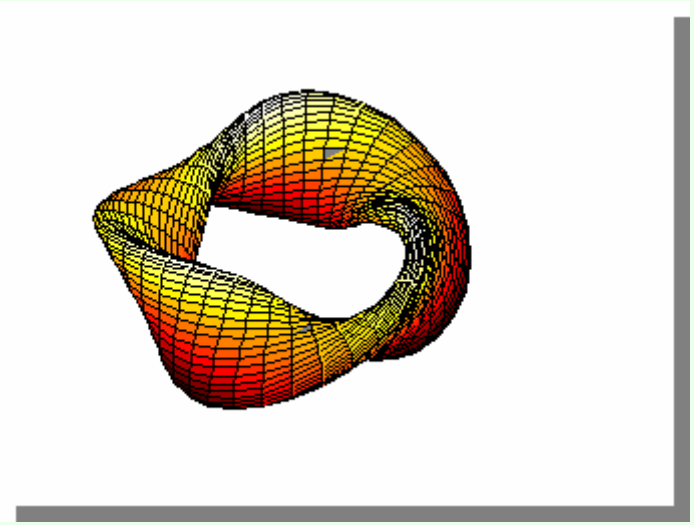
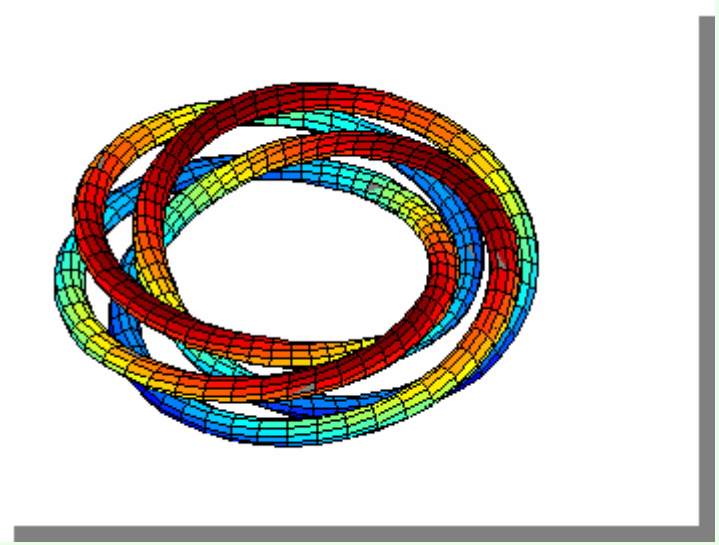
TM

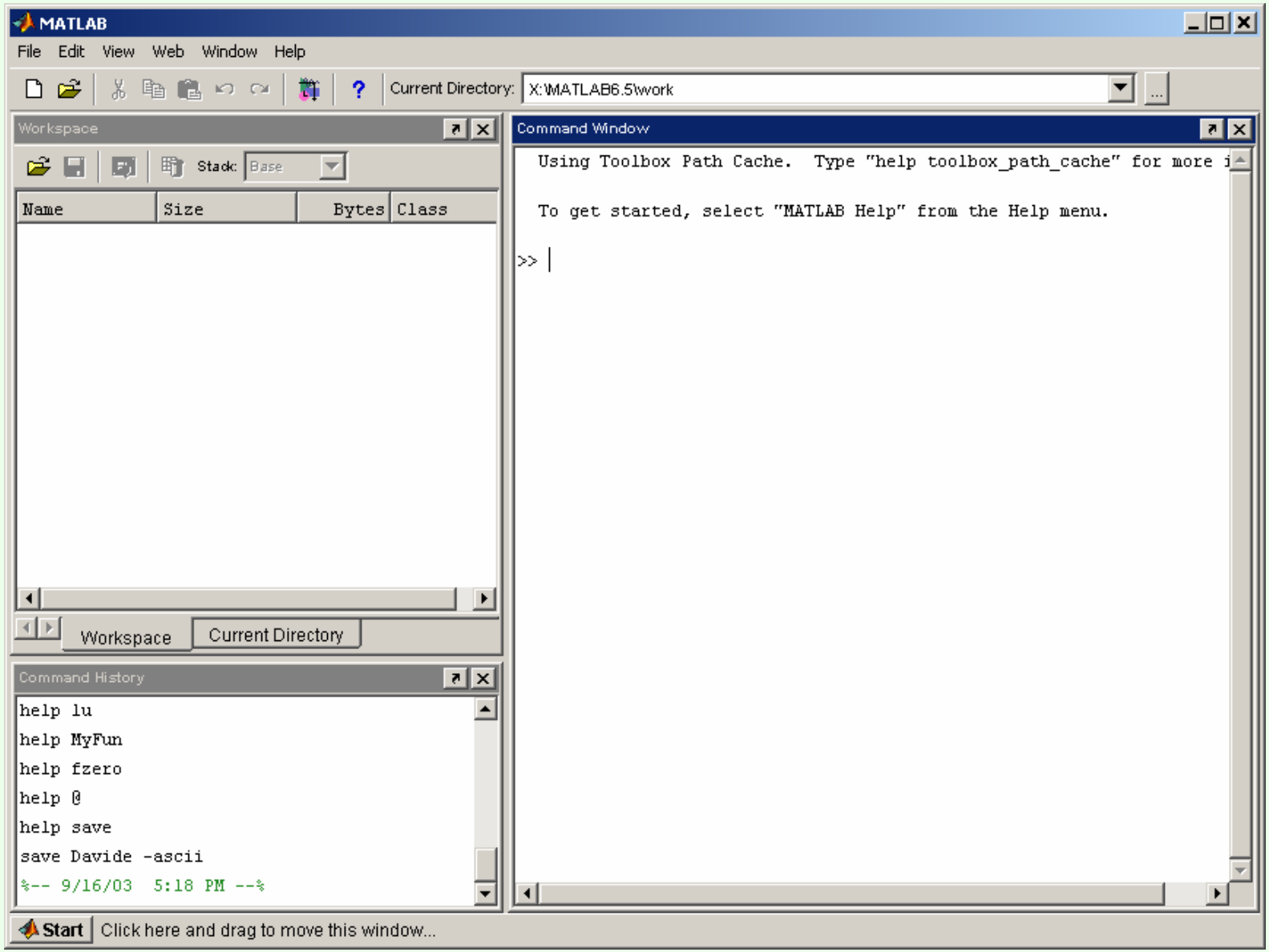
Tutorial di Matlab



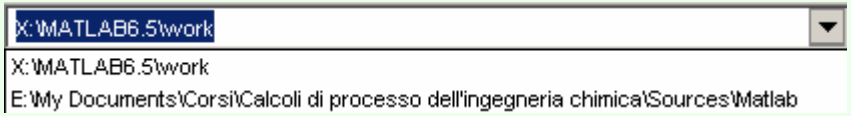


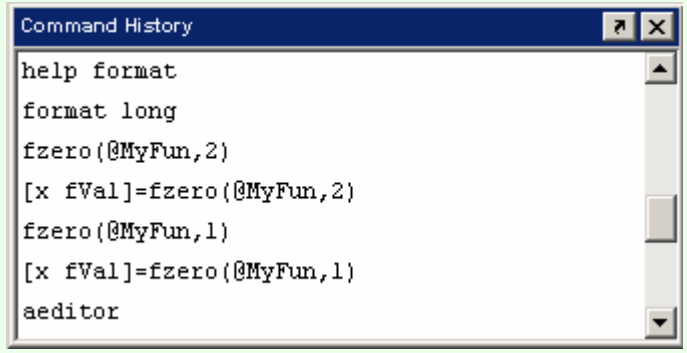


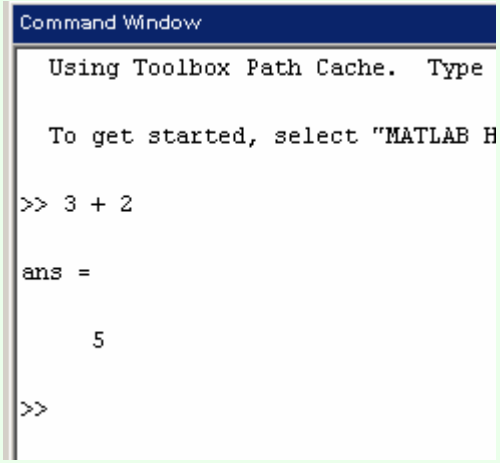
Avvio di Matlab

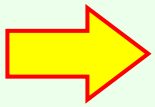


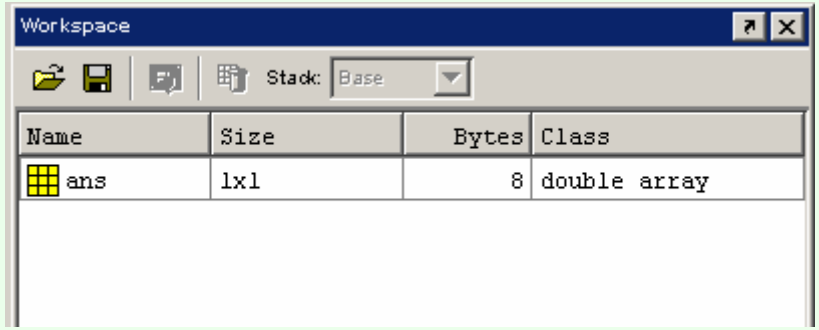
Avvio di Matlab

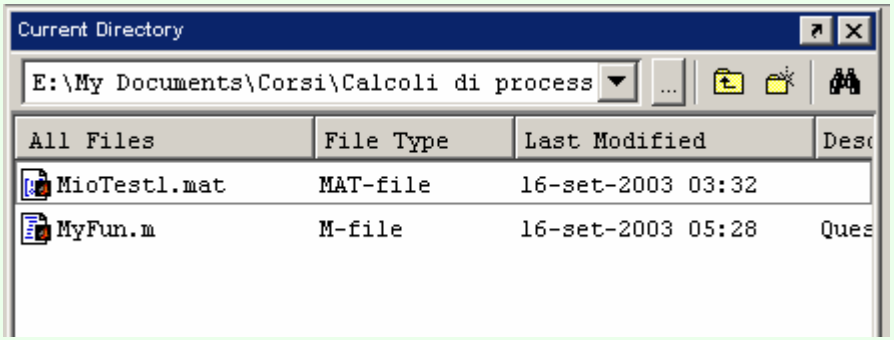








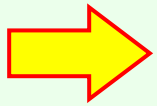




Si noti che Matlab esegue i calcoli e memorizza le variabili (scalari, vettoriali o matriciali) in **doppia precisione**

Matrici

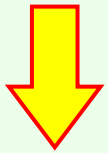
```
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9]

A =

     1     2     3
     4     5     6
     7     8     9

>>
```



Workspace

Stack: Base

Name	Size	Bytes	Class
A	3x3	72	double array
ans	1x1	8	double array



Array Editor: A

File Edit View Web Window Help

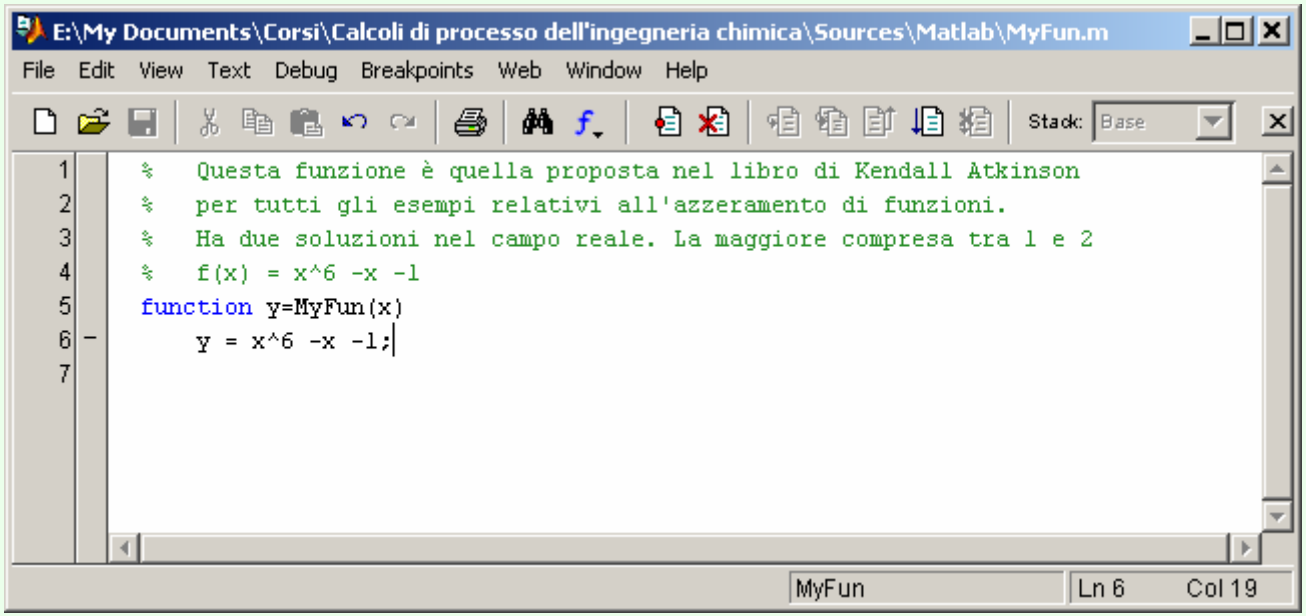
Numeric format: shortG Size: 3 by 3

	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9

Per lanciare l'Array Editor è sufficiente cliccare due volte sull'oggetto matrice (**A** nel nostro caso) della finestra Workspace

Editor di Matlab

Cliccando sull'icona "New M-file" della barra strumenti in alto a sinistra viene lanciato l'editor di Matlab con cui si possono scrivere e salvare dei file funzione o degli script (sequenze di comandi Matlab) richiamabili in un secondo momento.



Il direttorio di default dove Matlab salva i file dell'utente è: \Work.

Aggiungendo un nuovo direttorio alla lista dei direttori, Matlab ricerca i file funzione o di script anche in tale direttorio. È buona norma evitare di salvare i propri file nel direttorio di default \Work al fine di mantenere distinti i singoli progetti ed argomenti di ricerca.

Vettori

```
>> b = [3, 2, 1]

b =

     3     2     1
```

```
>> c = b'

c =

     3
     2
     1
```

```
>> d = c*b

d =

     9     6     3
     6     4     2
     3     2     1
```

```
>> e = b * c

e =

    14
```

Il comando **`clear`** cancella tutte le variabili presenti nel Workspace (area di lavoro in memoria dove Matlab pone le variabili create dall'utente).

N.B.: quando si inizia a risolvere un nuovo problema è buona norma cancellare tutte le variabili presenti nel Workspace per evitare inutili errori di riutilizzo o sovrapposizione di variabili già definite precedentemente.

```
>> clear
>> a = [1 2 3];
>> b = [5 6 7];
>> c = a .* b

c =

     5    12    21
```

Per effettuare un prodotto elemento per elemento tra vettori di uguali dimensioni si utilizza l'operatore: **`.*`**

Se alla fine di un'istruzione si aggiunge il segno **`;`** allora Matlab non presenta a video il risultato dell'istruzione anche se la esegue e ne memorizza il valore.

Altri comandi di Matlab

Il comando **`clc`** cancella invece soltanto il contenuto della finestra di Comando. Le variabili fino a quel punto introdotte **NON** vengono cancellate.

```
>> who

Your variables are:

a  b  c
```

```
>> whos
  Name        Size                    Bytes  Class

  a           1x3                        24  double array
  b           1x3                        24  double array
  c           1x3                        24  double array

Grand total is 9 elements using 72 bytes
```

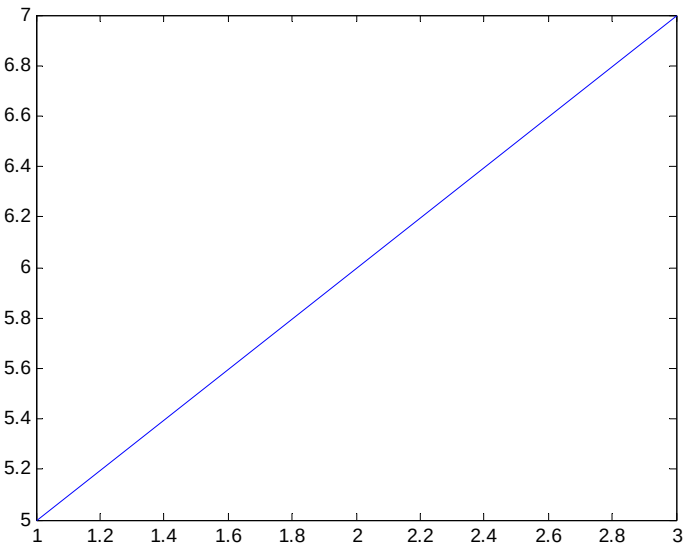
```
>> y = 0 : 2 : 10

y =

     0     2     4     6     8    10
```

```
>> z = 0 : 3 : 10

z =

     0     3     6     9
```

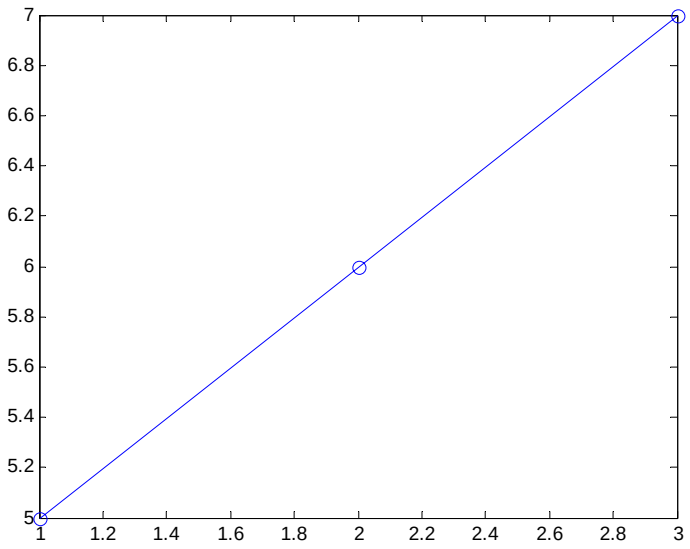
```
>>
>> x = 0.1 : 0.2 : 1.0

x =

    0.1000    0.3000    0.5000    0.7000    0.9000
```

Diagrammi

```
b =

     5     6     7

>> plot(b)
```




```
b =

     5     6     7

>> plot(b,'-o')
>>
```

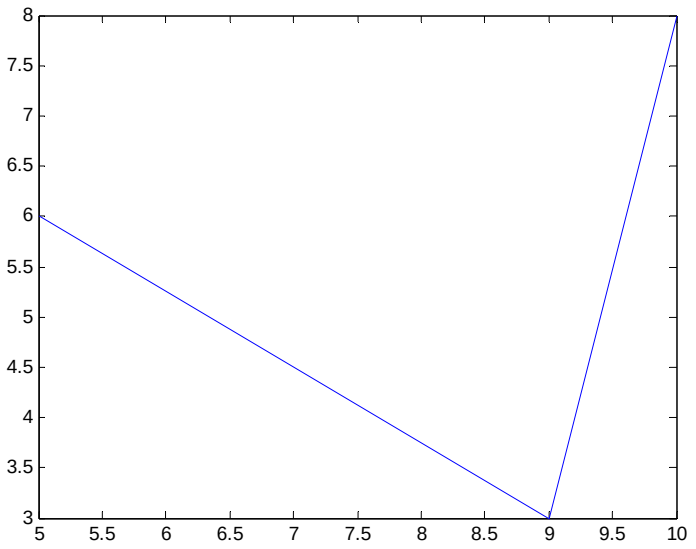


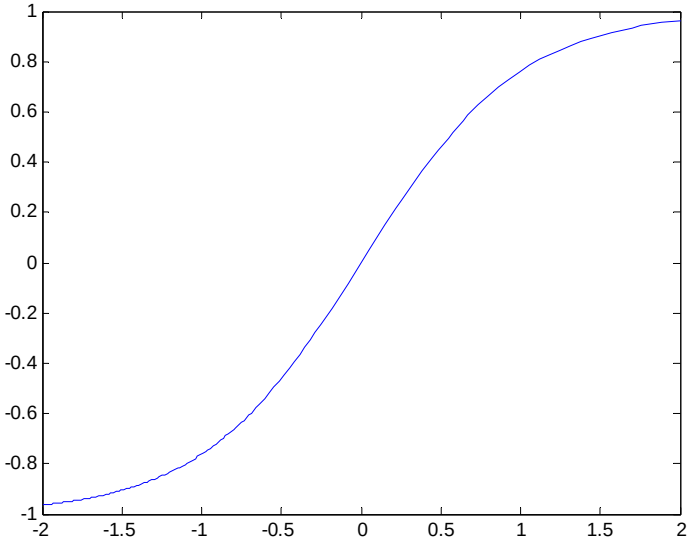
Diagrammi

```
x =

     5     9    10

>> y

y =

     6     3     8

>> plot(x, y)
```

fplot(@FunDaDiagrammare,[xLow xUp])

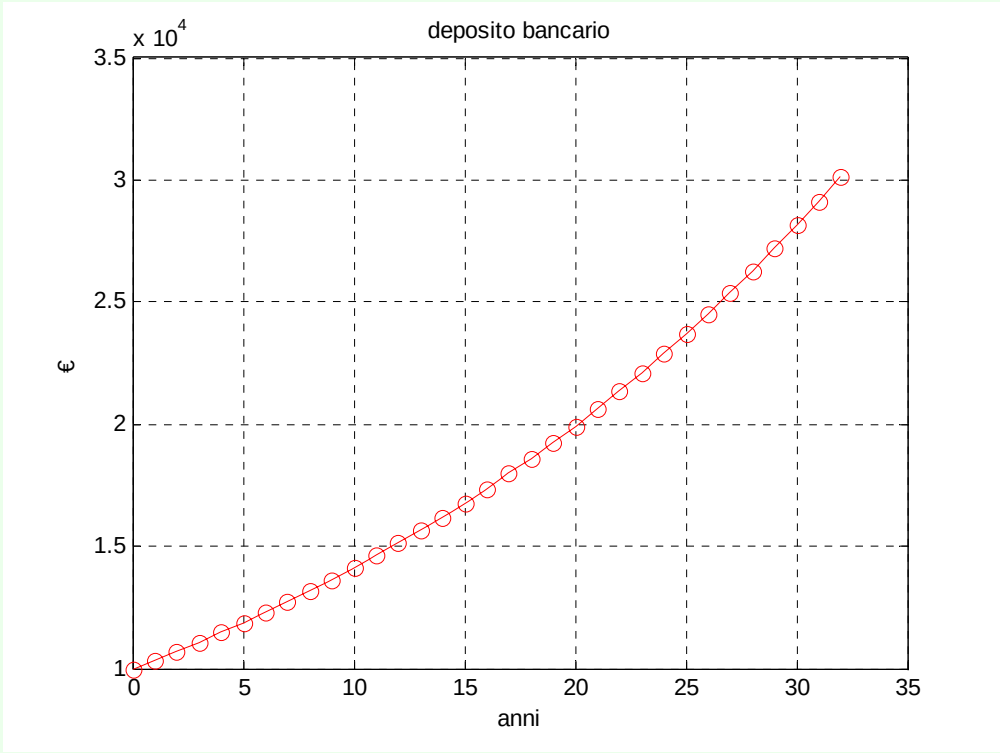
fplot('tanh',[-2 2])





Diagrammi

```
%  anno e cap sono i vettori con i valori delle ascisse ed ordinate
%  'r-o' stile della curva: "r"=red, "-"=linea continua, "o"=pallini per ogni dato
%  grid = griglia per entrambi gli assi
%  xlabel, ylabel = descrizione per gli assi delle ascisse e ordinate
%  title = titolo del grafico
plot(anno,cap,'r-o'),grid,xlabel('anni'),ylabel('€'),title('deposito bancario');
```



Help di Matlab

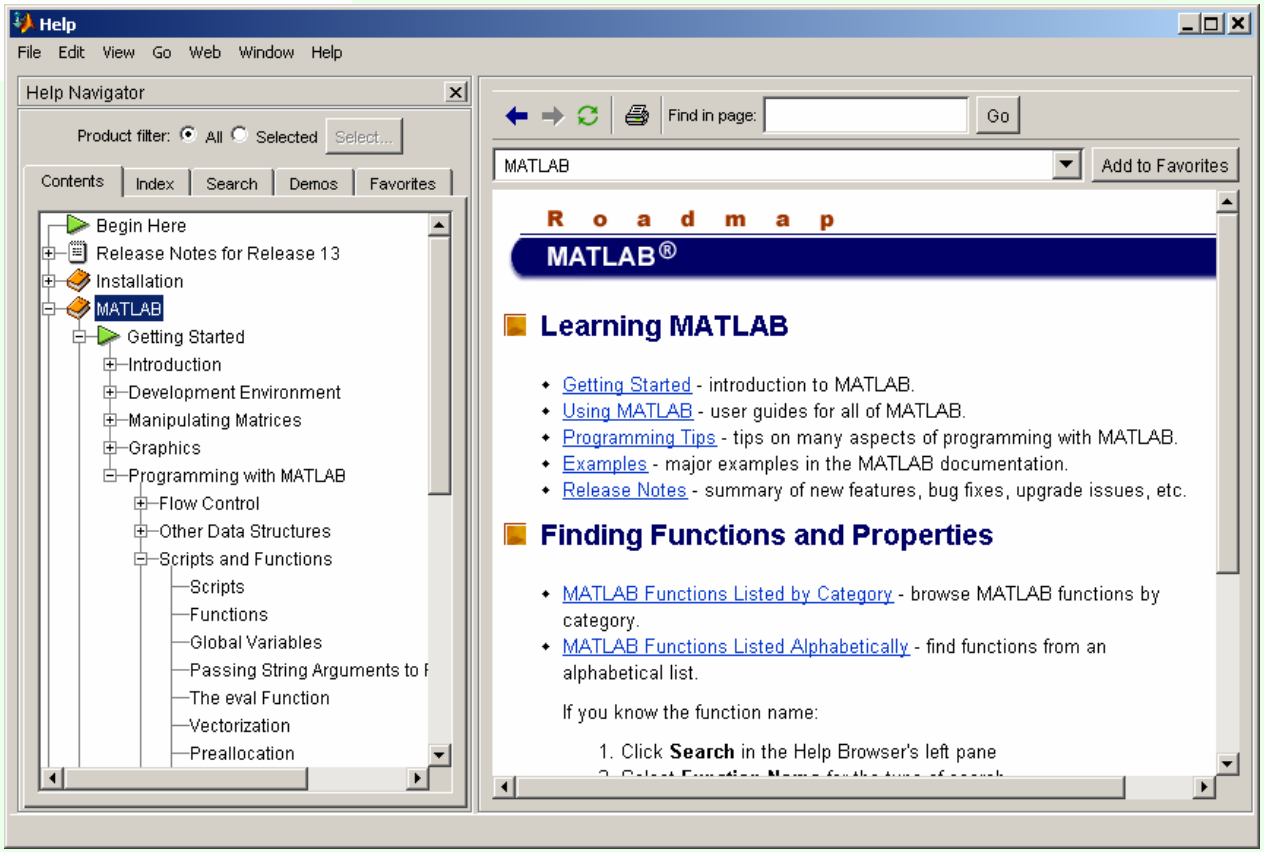
```
>> help inv

 INV    Matrix inverse.
    INV(X) is the inverse of the square matrix X.
    A warning message is printed if X is badly scaled or
    nearly singular.

    See also SLASH, PINV, COND, CONDEST, LSQNONNEG, LSCOV.

 Overloaded methods
    help sym/inv.m
```

```
>> help politecnico

politecnico.m not found.

>>
```

Help → MATLAB Help

Vedi anche:

Demos



Diagrammi di funzioni

E:\My Documents\Corsi\Calcoli di processo dell'ingegneria chimica\Sources\M...

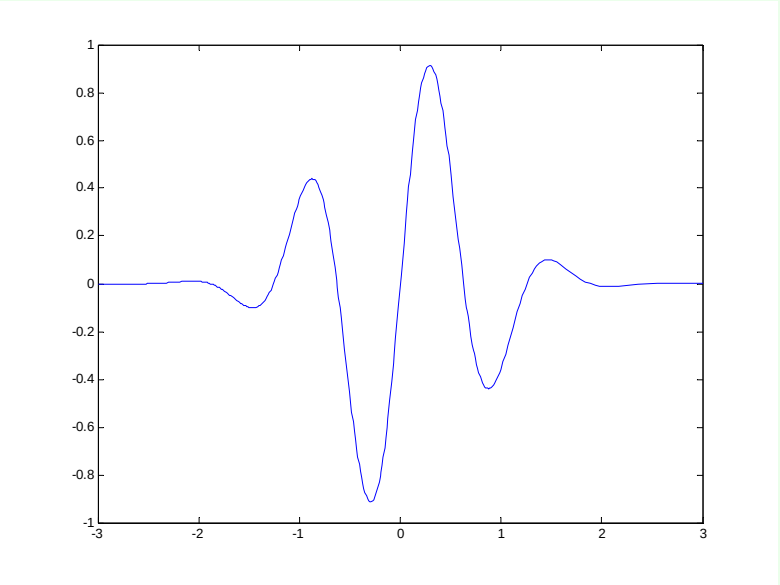
File Edit View Text Debug Breakpoints Web Window Help

```
1  %   Funzione di esempio per produrre un grafico con Matlab
2  function y = DvdSinExp(x)
3  -     y = exp(-x^2) * sin(5.*x);
4
```

DvdSinExp Ln 4 Col 1

```
>> z = DvdSinExp(0.9)

z =

    0.4251
```

```
>>
>> fplot(@DvdSinExp,[-3,3])
>>
```



Variabili e funzioni predefinite (intrinseche)

`i, j`	unità immaginaria
`pi`	pigreco
`Inf`	infinito (ad es.: `3/0`)
`NaN`	not a number (as es.: `0/0`, `Inf/Inf`)
`sin, cos, tan, asin, acos`	funzioni trigonometriche e inverse (in radianti)
`sinh, cosh, asinh, acosh`	funzioni iperboliche e loro inverse
`sqrt, log, log10, exp`	funzioni varie
`rem`	resto della divisione intera (divisione–modulo)
`mod`	resto della divisione intera (divisione–modulo)

N.B.: **`rem(x,y)`** restituisce un risultato avente lo stesso segno di **`x`** mentre **`mod(x,y)`** produce un risultato con lo stesso segno di **`y`**. Se **`x`** e **`y`** hanno lo stesso segno **`rem`** e **`mod`** forniscono il medesimo risultato. Se **`x`** e **`y`** hanno segno diverso i risultati prodotti da **`rem`** e **`mod`** sono diversi, non solo a livello di segno. Provare ad esempio: **`rem(-16,3)`** e **`mod(-16,3)`**.

Costrutti if for while

```
if x > 0
   y = sqrt(x);
elseif x == 0
   y = 0;
else
   y = NaN;
end
```

```
s = 0.;
for k = 1:100
   s = s + 1./k;
end


for k = 1:3:100
   ...
end
```

```
x = 3.;
while x < 25.
   x  = x + 2.;
end
disp(x)
```

Operatori di confronto: < <= > >= == ~=

Operatori logici: ~ [NOT], && [AND] , || [OR]

Costrutto switch

```
d = floor(3*rand) + 1
switch d
   case 1
      disp('È un 1')
   case 2
      disp('È un 2')
   otherwise
      disp('È un 3')
end
```

```
z = floor(10*rand)
switch z
   case {1, 3, 5, 7, 9}
      disp('Dispari')
   case {2, 4, 6, 8}
      disp('Pari')
   otherwise
      disp('Zero')
end
```

N.B.: per produrre il simbolo: **{** premere ALT + 123 (tastierino numerico)

per produrre il simbolo: **}** premere ALT + 125 (tastierino numerico)

per produrre il simbolo: **|** premere ALT + 124 (tastierino numerico)

per produrre il simbolo: **~** premere ALT + 126 (tastierino numerico)

Output formattato su video

```
>> fprintf('Pigreco = %g ',pi)
Pigreco = 3.14159
>> fprintf('Pigreco = %15.2f ',pi)
Pigreco =            3.14
>> fprintf('Pigreco = %15.4f ',pi)
Pigreco =          3.1416
>> fprintf('Pigreco = %15.8f ',pi)
Pigreco =      3.14159265
>> fprintf('Pigreco = %15.10f ',pi)
Pigreco =    3.1415926536
>> fprintf('Pigreco = %20.16f ',pi)
Pigreco =      3.1415926535897931
>> fprintf('Pigreco = %15.2e ',pi)
Pigreco =       3.14e+000
>> fprintf('Pigreco = %20.16e ',pi)
Pigreco = 3.1415926535897931e+000
```

Output formattato su file

```
unit = fopen('NomeFile.ext', 'w');
fprintf(unit, '\nPigreco = %15.10f ',pi);
... ... ...
fclose(unit);
```

Utilizzando i comandi **fwrite** e **fread** è anche possibile scrivere, **'w'**, o leggere, **'r'**, su o da file binari.

Si veda anche (utilizzando l'help) i comandi: **save** e **load**.

Buone norme di programmazione

Qualora si debba scrivere un numero reale è opportuno nei vari linguaggi di programmazione aggiungere il punto decimale.

Esempio: `(1./4.)` anziché `(1/4)`

Infatti, anche se Matlab perdona l'utente sbadato, non altrettanto avviene in altri linguaggi di programmazione quali il Fortran o il C++.

Programmando in Fortran ad esempio: `1/4 = 0` in quanto la divisione viene eseguita tra due cifre intere ed il risultato è automaticamente posto uguale ad un intero.

Attenzione quindi: `Pr^(1/3) = 1.` qualsiasi sia il valore di `Pr`.

La scrittura corretta è viceversa: `Pr^(1./3.)`.

Buone norme di programmazione

Altro rischio programmativo è rappresentato dagli operatori:

`*` oppure `.*`

`/` oppure `./`

`^` oppure `.^`

Gli operatori: `*`, `/`, `^` operano a livello vettoriale–matriciale in linea con l'analisi classica.

Viceversa gli operatori: `.*`, `./`, `.^` operano sui singoli elementi dei vettori o matrici.

Per effettuare un prodotto tra matrici in senso classico (prodotto righe per colonne) si utilizza l'istruzione: `C = A * B;`

Viceversa per effettuare il prodotto elemento per elemento: `C = A .* B;`

N.B.: Per evitare ambiguità tra l'utilizzo del punto come separatore decimale oppure come operatore congiunto ai simboli: `*`, `/`, `^` è opportuno mantenere gli operatori: `.*`, `./`, `.^` ben separati dalle variabili sulle quali essi operano. Esempio: `1. / 4. * pi * a ^ 2`

N.B.: elevando a potenza intera **non** si deve aggiungere il punto. Esempio: `a^2`, `b^5`

HOW TO

HT1

Domanda: come produrre dei punti equispaziati secondo n intervalli comprendendo gli estremi?

Risposta: innanzitutto evitare assolutamente il costrutto Matlab™: `x = [xLow:delta:xUp]`.

Ad esempio la seguente istruzione è scorretta: `x = [0.1: 0.01: 1.2]` a causa della precisione limitata della CPU, anche operando in doppia precisione.

Viceversa è corretto il seguente algoritmo:

Input: `xLow, xUp, nIntervalli`

Algoritmo corretto:

```
delta = (xUp – xLow) / nIntervalli;
for i = 1: nIntervalli + 1
   x(i) = xLow + delta * (i-1);
next i
```

HOW TO

HT1 continua

Il seguente algoritmo è **meno corretto** in quanto accumula gli errori di troncamento legati all'operazione di somma. I termini **`x(i)`** sono cioè meno precisi.

Algoritmo poco preciso:

```
delta = (xUp – xLow) / nIntervalli;
x(1) = xLow;
for i = 1, nIntervalli
   x(i+1) = x(i) + delta;
next i
```

HOW TO

HT2

Domanda: come produrre n punti equispaziati che comprendano anche gli estremi di un intervallo?

Risposta: innanzitutto evitare assolutamente il costrutto Matlab™: `x = [xLow:delta:xUp]`.

È corretto il seguente algoritmo:

Input: `xLow, xUp, nPunti`

Algoritmo:

```
delta = (xUp – xLow) / (nPunti – 1);
for i = 1: nPunti
   x(i) = xLow + delta * (i-1);
next i
```

HOW TO

HT3

Domanda: come funziona esattamente il costrutto **`while`** ?

Risposta: contrariamente a quanto qualcuno possa pensare, il costrutto **`while`** continua ad eseguire le istruzioni in esso contenute **MENTRE** la sua condizione è **VERA**.

Esempio: data la serie armonica (somma degli inversi dei numeri naturali) per sapere quanti termini sono necessari prima di raggiungere almeno il valore 5 si scriverà:

```
n = 0;
somma = 0.;
while somma < 5.
   n = n + 1;
   somma = somma + 1. / n;
end
disp([' n = ', num2str(n)]);
```

HOW TO

HT3 continua

Sarebbe un grave errore utilizzare l'espressione:

```
while somma = 5.
```

Parimenti è un'imprecisione utilizzare l'espressione:

```
while somma <= 5.
```

in quanto il problema chiede chiaramente: "prima di raggiungere **almeno** il valore 5". Ciò sta a significare che il ciclo **`while`** deve terminare non appena è stato raggiunto o superato il valore 5. Supponiamo, cosa non vera, che la somma dei primi 83 reciproci dei numeri naturali conducesse esattamente al valore 5. Se si utilizzasse l'istruzione:

```
while somma <= 5.
```

la procedura **`while`** sarebbe autorizzata a iterare ancora una volta per sommare il termine 84-esimo che quindi andrebbe oltre il valore 5 richiesto dal problema.

HOW TO

HT4

Domanda: come determinare il numero di righe e colonne di una matrice?

Risposta: occorre utilizzare l'istruzione **`size`**.

Esempio:

```
A = zeros(3,5)
nRows = size(A,1);
nCols = size(A,2);
disp(['nRows = ',num2str(nRows),'   nCols = ',num2str(nCols)]);
```

Output:

```
A =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
nRows = 3   nCols = 5
```

HOW TO

HT5

Domanda: come determinare il numero di dimensioni di una matrice?

Risposta: occorre utilizzare le istruzioni **`size`** e **`length`**.

Esempio:

```
A = zeros(3,5);

nRowsCols = size(A)

nDimensions = length(nRowsCols)
```

Output:

```
nRowsCols =

     3     5


nDimensions =

     2
```

HOW TO

HT6

Domanda: come si crea una matrice di numeri random?

Risposta: occorre utilizzare l'istruzione **`rand`**.

Esempio:

```
A = rand(4)     %  crea una matrice 4x4 di numeri random
B = rand(3,5)   %  crea una matrice 3x5 di numeri random
```

N.B.: i numeri random generati hanno una distribuzione uniforme ed appartengono all'intervallo 0,...1. Ogniqualvolta si ripeta il comando **`rand`** si ottiene una nuova matrice contenente numeri diversi dalla volta precedente.

HOW TO

HT7

Domanda: come si estrae la diagonale di una matrice e la si memorizza in un vettore?

Risposta: occorre utilizzare l'istruzione **`diag`**.

Esempio:

```
A = rand(4)    %  crea una matrice 4x4 di numeri random

b = diag(A)    %  estrae la diagonale della matrice
```

Output:

```
A =

    0.6491    0.1996    0.0990    0.3467
    0.8555    0.3462    0.3092    0.4739
    0.0465    0.9669    0.8400    0.3614
    0.6771    0.2056    0.9913    0.6677

b =

    0.6491
    0.3462
    0.8400
    0.6677
```

HOW TO

HT8

Domanda: come si effettua il prodotto degli elementi di un vettore?

Risposta: o con un semplice ciclo **`for`** oppure con l'istruzione **`prod`**.

Esempio:

```
vet = [3.1 4.4 -7.12 2.8];
ris1 = 1.;
for i = 1: length(vet)
   ris1 = ris1 * vet(i);
end
ris1
ris2 = prod(vet)
```

N.B.: attenzione a non usare l'istruzione **`size(vet)`** anziché **`length(vet)`** in quanto se si ha un vettore riga l'istruzione **`size`** ritorna come primo elemento il numero di righe dell'array, coincidente con **`1`**.

Così facendo il ciclo **`for`** verrebbe eseguito soltanto una volta operando sul primo elemento del vettore **`vet`** producendo così **`ris1 = 3.1`** anziché **`-271.9270`**.

Output:

```
ris1 =
 -271.9270
ris2 =
 -271.9270
```

HOW TO

HT9

Domanda: come si calcola il determinante di una matrice triangolare?

Risposta: se una matrice è triangolare destra o sinistra il suo determinante coincide con il prodotto degli elementi della diagonale. Si utilizzano le istruzioni: **`prod`** e **`diag`**. In alternativa l'istruzione generale **`det`**.

Esempio:

```
R = zeros(3);
for i = 1: 3
   for j = i: 3
      R(i,j) = 100. * rand;
   end
end
R
det1 = prod(diag(R))
det2 = det(R)
```

Output:

```
R =
   34.3592   11.6499   52.5129
         0   46.4761   51.6449
         0         0   11.9602
det1 =
  1.9099e+004
det2 =
  1.9099e+004
```

HOW TO

HT10

Domanda: come salvare uno script (sequenza di istruzioni) in Matlab™ ?

Risposta: innanzitutto occorre comprendere su quale direttorio Matlab sta puntando tramite il comando: **`pwd`**.

Salvando il file nel direttorio riportato da Matlab™ tramite il comando pwd sarà poi possibile eseguirlo dalla linea di comando digitando il solo nome del file stesso.

Esempio:

```
>> pwd

ans =

X:\MATLAB7.0\work
```

Se si crea un nuovo file e lo si salva con il nome: **`prova.m`**, sarà sufficiente lanciare il comando: **`prova`** per eseguirlo.

Se si vuole cambiare direttorio dove salvare e far eseguire i file di Matlab™ è possibile utilizzare il comando **`chdir nomedir`**. Esempio: **`chdir c:\myfiles\ese1\`**

HOW TO

HT11

Domanda: come disegnare con Matlab™ quattro diagrammi in un'unica finestra?

Risposta: si utilizza l'istruzione **`subplot(n,m,i)`** che permette di disegnare **`n`** righe ed **`m`** colonne di diagrammi puntando nella fattispecie sul diagramma **`i`**-esimo contando da sinistra a destra e dall'alto in basso. Il diagramma è poi ottenuto tramite la solita istruzione **`plot`**.

Esempio:

```
subplot(2,2,1);    % 4 diagrammi (2 x 2), inizio a puntare al #1
plot(x1,y1),title 'diagramma 1';
subplot(2,2,2);    % punto al #2 (in alto a destra)
plot(x2,y2),title 'diagramma 2';
subplot(2,2,3);    % punto al #3 (in basso a sinistra)
plot(x3,y3),title 'diagramma 3';
subplot(2,2,4);    % punto al #4 (in basso a destra)
plot(x4,y4),title 'diagramma 4';
```

HOW TO

HT12

Domanda: come si determina la parte intera di un numero in virgola mobile?

Risposta: provare i seguenti comandi di Matlab™: **`fix, round, floor, ceil`**.

Esempio:

```
x = [-5.8, -3.2, -2.5, 0., 0.5, 2.1, 3.9]
fixX = fix(x)
roundX = round(x)
floorX = floor(x)
ceilX = ceil(x)
```

Output:

```
x      =  -5.8000   -3.2000   -2.5000   0     0.5000    2.1000    3.9000
fixX   =    -5       -3        -2       0     0         2         3
roundX =    -6       -3        -3       0     1         2         4
floorX =    -6       -4        -3       0     0         2         3
ceilX  =    -5       -3        -2       0     1         3         4
```

HOW TO

HT13

Domanda: come si fa a sapere in quale direttorio Matlab™ sta operando?

Risposta: si utilizza il comando: pwd

Esempio:

```
>> pwd

ans =

E:\MyDocuments\Corsi\CDPDIC\Sources\Matlab\Ese3
```

HT14

Domanda: come si fa a sapere quali sono i file presenti nel direttorio attuale?

Risposta: si utilizza il comando: dir

Esempio:

```
>> dir

DvdSolveL.m   Ese34.m   Ese35.m   Ese38.m   UseDvdSolveA.m
```

HOW TO

HT15

Domanda: come si fa a sapere a quale direttorio appartiene una certa funzione?

Risposta: si utilizza il comando: which FUN (oppure anche which FUN –ALL)

Esempio:

```
>> which fsolve
C:\MATLAB\toolbox\optim\fsolve.m
```

HT16

Domanda: come si fa a cambiare direttorio tramite linea di comando?

Risposta: si utilizza il comando: cd

Esempio:

```
>> cd c:\windows\system32
```

HOW TO

HT17

Domanda: come si fa a calcolare l'integrale analitico di una funzione?

Risposta: si utilizzano i comandi: syms e int

Esempio:

```
>> syms x
>> int(x^2)
ans =
1/3*x^3
```

HT18

Domanda: come si fa a calcolare la derivata analitica di una funzione

Risposta: si utilizzano i comandi: syms e diff

Esempio:

```
>> syms x
>> diff(x^3)
ans =
3*x^2
```

HOW TO

HT19

Domanda: come si fa a specificare i limiti inferiore e superiore di un asse in un diagramma prodotto con il comando **`plot`** ?

Risposta: si utilizzano le istruzioni: **`XLim`** e/o **`YLim`** e il comando **`set`**

Esempio:

```
plot(xF,yF,'r-'),grid,title('Funzione di Lagrange');
set(gca,'YLim',[-0.5 1.5]);
```

Spiegazione: **`gca`** è il puntatore alla finestra attuale dove vengono presentati i grafici. L'istruzione **`YLim`** indica che si vuole specificare un intervallo per l'asse delle ordinate. Il vettore successivo **`[-0.5  1.5]`** definisce i valori minimo e massimo per tale asse.

N.B.: in alternativa si clicca due volte sull'asse delle ascisse o delle ordinate del grafico prodotto dal comando **`plot`** e si assegnano in modo interattivo gli estremi del grafico.

HOW TO

HT20

Domanda: come si fa a mostrare i caratteri speciali nelle scritte (ad esempio titolo ed assi di un diagramma) ?

Risposta: si utilizzano i caratteri speciali TeX che premettono un carattere di backslash “\”

Esempio:

```
plot(xF,yF,‘r-'), title('Funzione \alpha’); % lettera greca alfa
```

Altri esempi:

```
\beta   \gamma  \sim    \infty  \circ   \Gamma  \Delta  \pm     \forall
\exists \approx \neq
```

HOW TO

HT21

Domanda: se durante l'esecuzione del codice si incontra un problema insormontabile e si desidera bloccare il programma come si fa?

Risposta: si utilizza il comando **`exit`**

Esempio:

```
if portata < 0.
   disp('La portata è negativa. Impossibile proseguire...');
   exit
end
```

Spiegazione: l'istruzione **`exit`** blocca l'esecuzione del programma, esce dallo script o dalla funzione in cui Matlab stava operando e torna al prompt dei comandi.

N.B.: se l'istruzione **`exit`** è all'interno di un file di comandi (non una funzione) Matlab viene abortito. Per evitare ciò utilizzare l'istruzione: **`error('……')`** come spiegato in seguito.

HOW TO

HT21 continua

Lo stesso risultato delle due istruzioni appena proposte:

```
disp('La portata è negativa. Impossibile proseguire...')
exit
```

è ottenibile tramite la singola istruzione:

```
error('La portata è negativa. Impossibile proseguire...')
```

Questa istruzione dopo aver scritto a video il messaggio di errore, blocca l'esecuzione del programma. È fondamentale utilizzarla all'interno di file di comando (ovvero file che non siano funzioni. In caso contrario l'istruzione **`exit`** abortisce Matlab.

HOW TO

HT22

Domanda: come eseguire una serie di istruzioni iterativamente fintantoché una delle due condizioni non è più vera?

Risposta: si utilizza il comando **`while`** con l'operatore di confronto **`&&`**

Esempio:

```
while (b-a) < EPSI   &&   iConto < MAX_ITER
   iConto = iConto + 1;
   ...  %  serie di istruzioni iterative
end
```

Spiegazione: il ciclo **`while`** viene eseguito mentre le condizioni poste sono vere. Se ad esempio si vuole proseguire con le iterazioni, fintantoché l'intervallo **`(b-a)`** è maggiore di **`EPSI`** o fintantoché il numero massimo di iterazioni **`MAX_ITER`** non è stato raggiunto, il ciclo **`while`** riportato nell'esempio è ciò che fa al caso nostro.

HOW TO

HT23

Domanda: esiste un modo per memorizzare una informazione in una variabile di tipo stringa?

Risposta: Sì. È molto semplice. Basta assegnare la stringa alfanumerica tra virgolette semplici ad una nuova variabile.

Esempio:

```
unaStringa = 'Soluzione non accettabile';

messaggio = 'Il programma abortisce per un grave errore.';

disp(unaStringa);

disp(messaggio);
```

HOW TO

HT24

Domanda: come è possibile studiare il comportamento analitico di una funzione effettuando numerose operazioni quali calcolo della sua derivata, integrale, traslazione, inversa, …, diagrammando comodamente i risultati?

Risposta: È molto semplice. Basta eseguire da riga di comando l'istruzione: **`funtool`**.

Esempio:

```
>> funtool
```

Automaticamente Matlab™ lancia una finestra interattiva con numerosi bottoni e caselle di testo di facile comprensione. In più vengono visualizzate due finestre grafiche su cui appaiono in tempo reale i risultati richiesti dall'utente. La scrittura della funzione è semplicissima e rispetta la sintassi Matlab™.

HOW TO

HT25

Domanda: è possibile lavorare con Matlab™ in singola o in doppia precisione ?

Risposta: Dalla versione 7.0 di Matlab™ sì, utilizzando i comandi **`single`** e **`double`**.

Esempio:

```
>> format long
>> single(3.14)
ans =
   3.1400001
>> double(3.14)
ans =
   3.14000000000000
```

Si noti che utilizzando il formato esteso di rappresentazione dei numeri la singola precisione non è in grado di rappresentare il numero razionale 3.14 e quindi lo approssima con **`3.1400001`**. Solo la doppia precisione descrive correttamente il numero 3.14.

HOW TO

HT25 continua

Esempio:

```
>> y = 1.e20 * 1.e30
y =
   1e+050
>> x = single(1.e20)* single(1.e30)
x =
   Inf
```

Se non specificato **y** è per default in doppia precisione in Matlab™. Al contrario **x** eccede il massimo valore rappresentabile in singola precisione. Avremmo ottenuto lo stesso risultato con **x = single(y)**.

HOW TO

HT26

Domanda: come si fa a capire se un numero intero è pari o dispari ?

Risposta: tramite la funzione intrinseca **`mod(x,y)`** che restituisce il resto della divisione tra interi

Esempio:

```
if mod(n,2) == 0
   %  il numero è pari (avendo ipotizzato che n sia un intero)
else
   %  il numero è dispari
end
```

HOW TO

HT27

Domanda: come si fa a disegnare più grafici di funzione su di uno stesso diagramma

Risposta: tramite il comando **`hold on`** dopo la prima istruzione **`plot`** o **`fplot`** e prima della successiva.

Esempio:

```
fplot('sin(x)',[-pi pi])

hold on

fplot('cos(x)',[-pi pi])
```

HOW TO

HT28

Domanda: se su di una stessa figura giacciono più diagrammi come è possibile mostrare una legenda per ognuna delle curve?

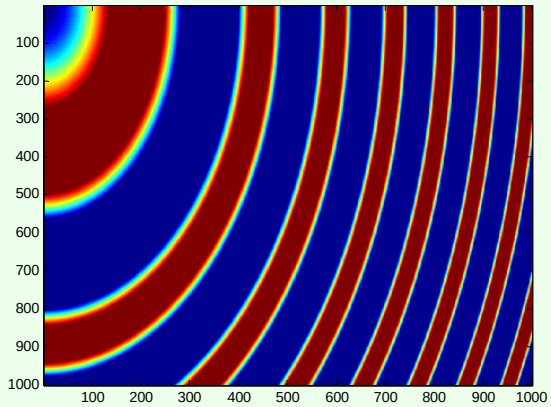
Risposta: tramite l'istruzione **`legend`**.

Esempio:

```
plot(x1,y1,'b-',x2,y2,'r-',x3,y3,'k:')

legend('prima curva','seconda curva','terza curva')
```

HOW TO

HT29



Domanda: come si ottiene una mappa colorata di una matrice?

Risposta: tramite l'istruzione **`image`**.

Esempio:

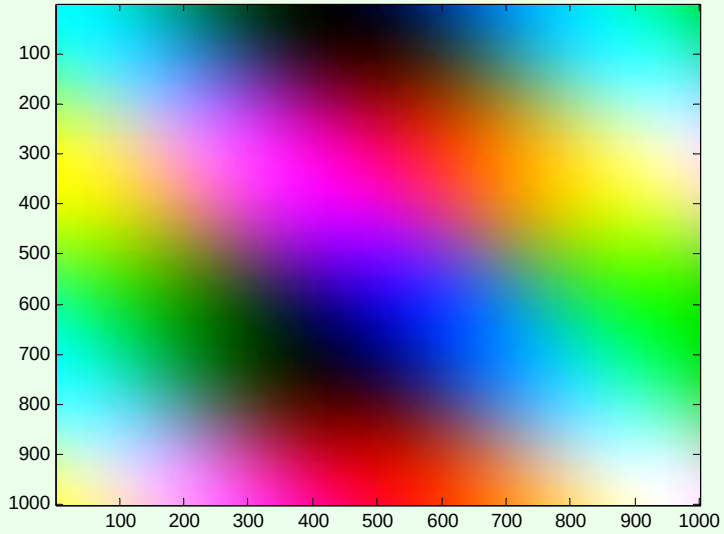
```
nDim = 1000;
A = zeros(nDim);
for i = 1: nDim
   for j = 1: nDim
      A(i,j) = 100.*sin((pi*i/nDim)^2 + (2.*pi*j/nDim)^2);
   end
end
image(A)
```

Risposta: i valori dei coefficienti della matrice debbono essere sufficientemente diversi l'uno dall'altro pena l'appiattimento dei colori.

HOW TO

HT29 continua

In alternativa, sempre utilizzando l'istruzione **image** è possibile fornire le componenti RGB (red, green, blue) di ogni elemento della matrice. Tali componenti debbono appartenere all'intervallo 0.,...1. In tale caso si lavora con una matrice a tre dimensioni dove la terza dimensione, composta di tre elementi, memorizza i dati RGB.



Esempio:

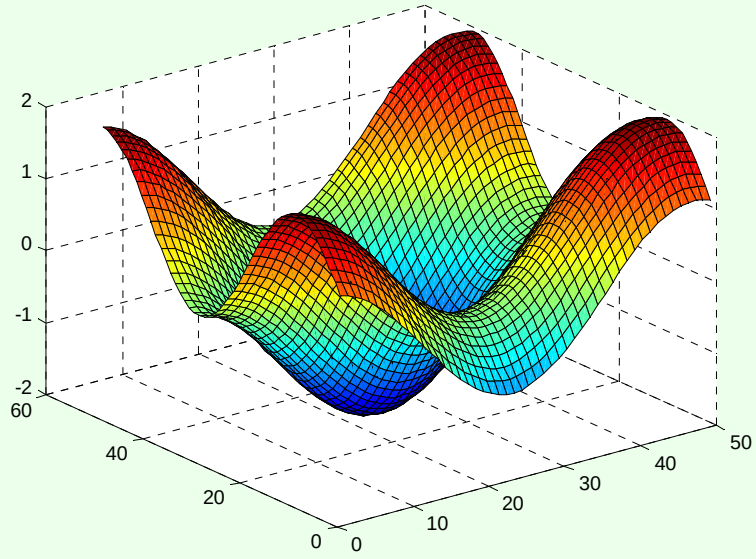
```
nDim = 1000;
A = zeros(nDim,nDim,3);
for i = 1: nDim
   for j = 1: nDim
      A(i,j,1)=abs(sin(1.5*pi*i/nDim)^2);
      A(i,j,2)=abs(cos(1.1*pi*j/nDim)^2);
      A(i,j,3)=abs(cos(1.3*pi*(i-j)/nDim)^2);
   end
end
image(A)
```

HOW TO

HT30

Domanda: come si ottiene una superficie colorata dai dati di una matrice?

Risposta: tramite l'istruzione **`surf`**.



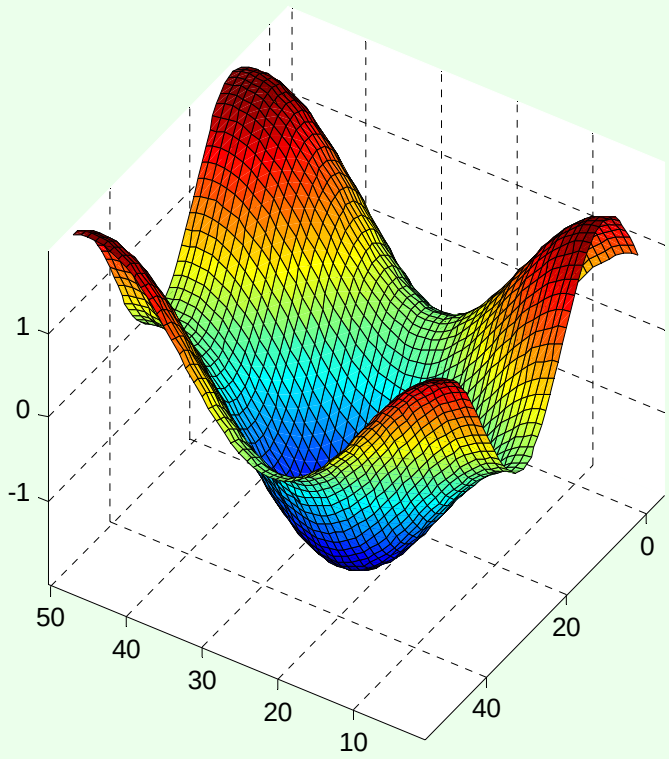
Esempio:

```
nDim = 50;
A = zeros(nDim);
for i = 1: nDim
   for j = 1: nDim
      A(i,j) = sin(2.5 * pi * i / nDim) + ...
         cos(2.1 * pi * j / nDim);
   end
end
surf(A)
```

HOW TO

HT30 continua

Aggiungendo l'istruzione **`cameratoolbar`** alla fine dello script riportato alla *slide* precedente è possibile ruotare l'immagine in tre dimensioni utilizzando il *mouse*.



HOW TO

HT31

Domanda: come si ottiene un suono di beep?

Risposta: tramite l'istruzione **beep**.

Esempio:

```
A = rand(1000);
b = rand(1000,1);
x = A \ b;
beep
disp('Ho finito di risolvere il sistema lineare...')
```

HOW TO

HT32

Domanda: come si fa a cronometrare il tempo di una certa sequenza di istruzioni?

Risposta: tramite le istruzioni **`tic`** e **`toc`**.

Esempio:

```
tic
A = rand(2000);
b = rand(2000,1);
x = A \ b;
toc
```

Output:

```
Elapsed time is 3.250000 seconds.
```

N.B.: la precisione nel conteggio del tempo di calcolo non è elevatissima.

In alternativa è possibile utilizzare il comando **`cputime`** che restituisce il tempo di calcolo impiegato da Matlab™ fino a quel momento.

In tal caso per conoscere il tempo di calcolo impiegato da una sequenza di istruzioni è sufficiente scrivere:

```
tIni = cputime;
...;    % sequenza istruzioni
cputime - tIni
```

HOW TO

HT33

Domanda: come si fa a suonare un vettore?

Risposta: tramite l'istruzione **sound**.

Esempio1:

```
x = rand(10000,1)
sound(x)
```

Esempio2:

```
load handel
sound(y)
```

N.B.: gli elementi del vettore x vengono riprodotti con una frequenza di compionamento di 8192 elementi al secondo (8192 Hz) se altrimenti non specificato.

HOW TO

HT34

Domanda: come si fa a sospendere l'esecuzione di un programma?

Risposta: tramite l'istruzione **pause**.

N.B.: se si utilizza la semplice istruzione **pause** il programma si ferma in attesa che l'utente prema un qualsiasi tasto.

In alternativa **pause(xSec)** blocca il programma per **xSec**. Ad esempio **pause(0.2)**.

HOW TO

HT35

Domanda: come si fa a bloccare l'esecuzione di un programma che sia entrato in un loop infinito o che comunque non risponda più a causa di un calcolo lunghissimo?

Risposta: premendo i tasti **`Ctrl+c`**.

N.B.: i tasti **`Ctrl+c`** debbono essere premuti contemporaneamente dalla finestra di comando di Matlab™. Dopo la pressione di tali tasti il programma o lo script in esecuzione viene bloccato e non è possibile riprendere l'esecuzione.

HOW TO

HT36

Domanda: è possibile incrementare il numero di cifre significative mostrate nel comando **`disp`** tramite la funzione **`num2str`** ?

Risposta: Sì. È sufficiente aggiungere alla funzione **`num2str`** un descrittore circa il formato desiderato per il numero. Ad esempio:

Esempio:

```
disp(['x1 = ',num2str(x1,'%12.8f'),   ...
            '  x2 = ', num2str(x2,'%20.16e’]);
```

Bibliografia

- http://www.eece.maine.edu/mm/matweb.html
- http://spicerack.sr.unh.edu/~mathadm/tutorial/software/matlab/
- http://www.engin.umich.edu/group/ctm/basic/basic.html
- http://www.mines.utah.edu/gg_computer_seminar/matlab/matlab.html
- http://www.math.ufl.edu/help/matlab-tutorial/
- http://www.indiana.edu/~statmath/math/matlab/
- http://www.ciaburro.it/matlab/matlab.pdf

Altre risorse presso:

- http://www.chem.polimi.it/homes/dmanca/CDPDIC/index.htm
- http://www.chem.polimi.it/homes/dmanca/CDPDIC/tutorial/tutorial.htm
- http://www.chem.polimi.it/homes/dmanca/CDPDIC/Video/video.htm