MATLAB

1

MATLAB

- **MATLAB: MAtrix LABoratory**
- **Cosa è MATLAB ?**
  - MATLAB® è un linguaggio di programmazione facile da usare adatto ad ambiti tecnici dove I problemi sono espressi in notazione matematica.
- **Usi tipici:**
  - Matematica e calcolo
  - Sviluppo di algoritmi
  - Acquisizione di dati
  - Modellizzazione, simulazione e prototipaggio
  - Analisi esplorazione e visualizzazione dei dati
  - Grafici scientifici
  - Sviluppo di applicazioni
- **MATLAB è un sistema interattivo il cui elemento di base è la matrice che non richiede di essere dimensionata**

2

Ambiente MATLAB

- L'ambiente ***MATLAB*** è composto da cinque parti:
  - **Ambiente di Svilupo**: Un insieme di tool che facilitano l'uso di MATLAB. Questo include il Desktop MATLAB, la Command Window, la Command History, un editor con debugger, un browsers per l'help, il "workspace".
  - **La libreria di funzioni matematiche.** Una vasta collezione di algoritmi che spaziano dalle funzioni matematiche elementari (sum, sine, cosine) fino a quelle più complesse (calcolo di matrici inverse, calcolo degli autovalori, funzioni di Bessel, FFT).
  - **Il Linguaggio di programmazione.** Un linguaggio di alto livello con controllo di flusso, funzioni, strutture, input ed output. Permette di programmare rapidamente (create quick and dirty) o realizzare programmi complessi.
  - **Supporto per i grafici.** Un numero coesistente di facilities per realizzare grafici a due o tre dimensioni ma anche Image Processing, animazioni, e presentazioni.
  - **MATLAB API.** Una libreria che permette di integrare funzioni MATLAB con altri linguaggi (C,C++, Fortran). Permettono di chiamare routine MATLAB (dynamic linking), leggere e scrivere i MAT-file.

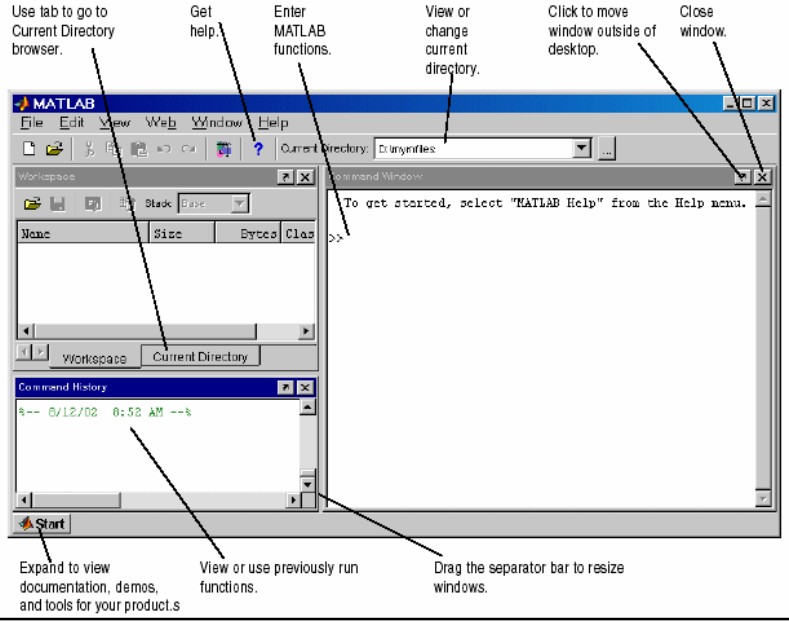
3

Organizzazione del Corso

- **Ambiente di sviluppo**
  - Introduzione all'ambiente MATLAB, i tool ed il *MATLAB desktop*.
- **Manipolazione delle Matrici**
  - Introduzione alla programmazione matriciale
- **Realizzare grafici**
  - Introduzione alle funzioni che realizzano i grafici.
- **Programmare in MATLAB**
  - Introduzione al linguaggio di programmazione di MATLAB per realizzare script e funzioni, manipolare strutture dati.

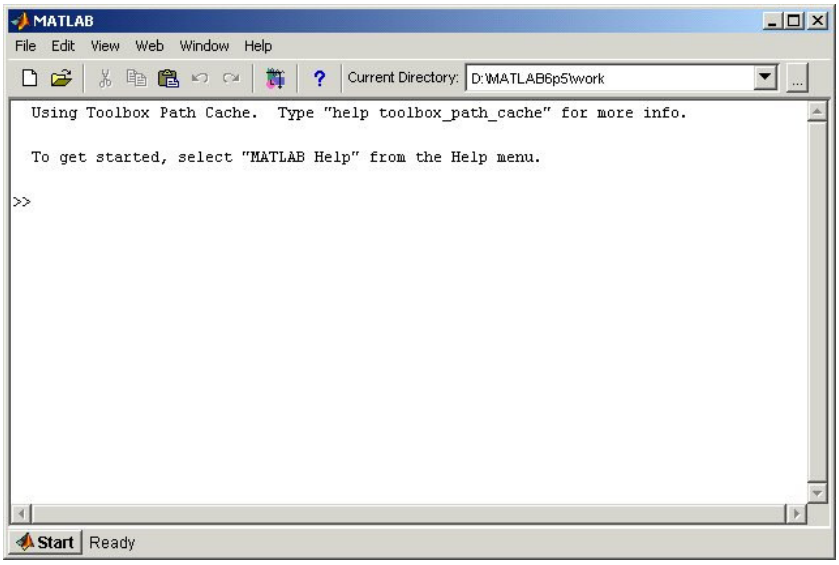
4

MATLAB Desktop



5

MATLAB Desktop - Classico



6

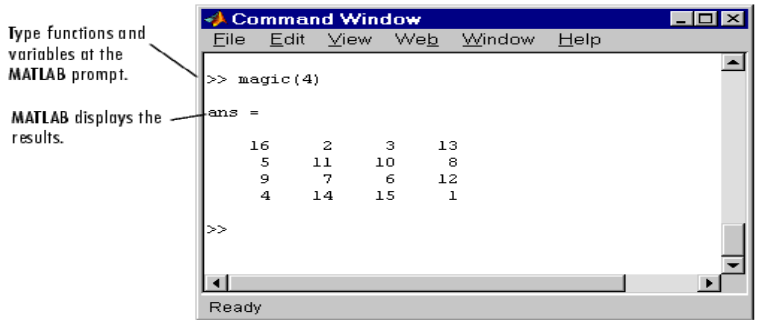
Desktop Tools

- Tool principali :
  - **Command Window**
  - **Command History**
  - (*Start Button*)
  - **Help Browser**
  - **Current Directory Browser**
  - **Workspace Browser**
  - (Array Editor)
  - **Editor/Debugger**
  - (Profiler)

7

Command Window

- Ricorda la shell unix (supporta i comandi shell)
- È usata per inserire varibili eseguire script (M-file)
- È controllata dall'utente

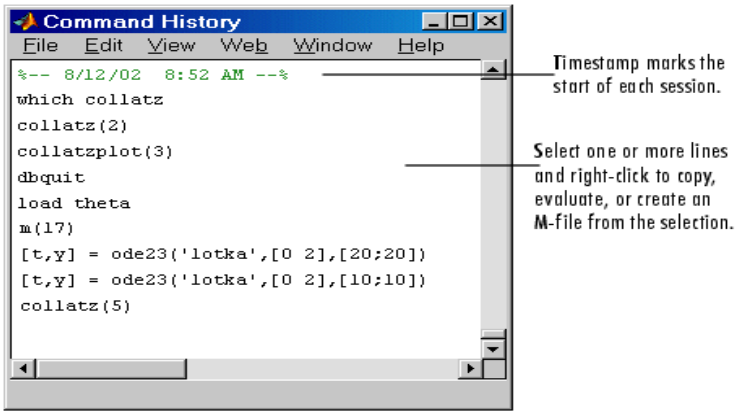


- **Eseguire Programmi esterni senza uscire da MATLAB**

8

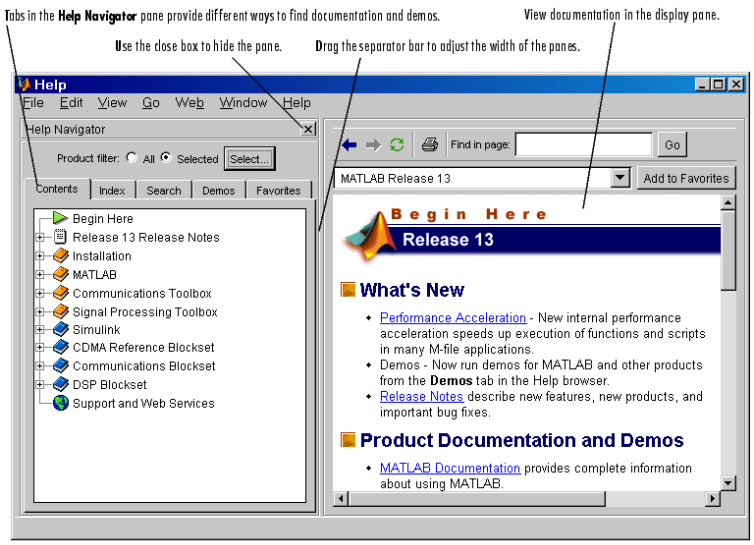
Command History

- I comandi eseguiti nella Command Window sono memorizzati nella Command History.



9

Help Browser



10

Command Window HELP

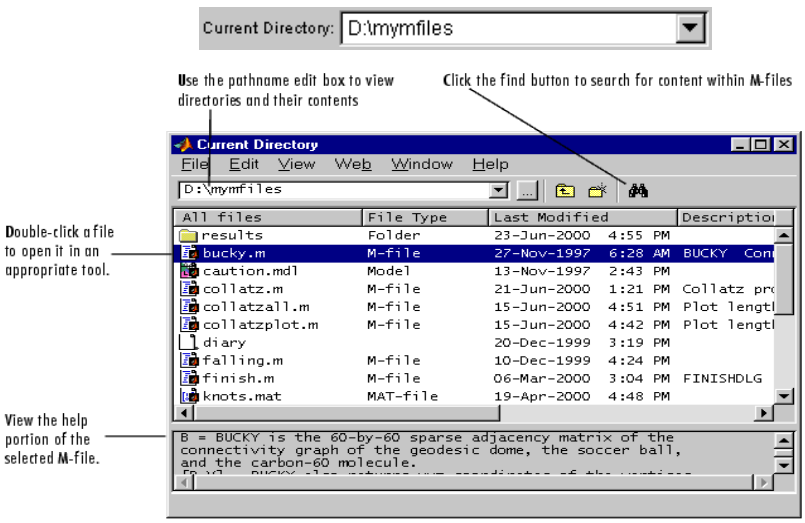
- Scrivendo “help <TOPIC>" vi viene visualizzato nella CW l’help.
- Il Topic può essere:
  - Il nome di un comando
  - Una directory

- Esempi
  - help sum
  - help matlab/ops

11

Current directory browser

- MATLAB esegue le operazioni (script, funzioni) nella directory corrente e nel search path.



12

Search PATH

- MATLAB usa un ***search path*** per trovare gli M-files e gli altri file. Le directory sono quelle del vostro sistema.
- Ogni file che si vuole eseguire deve risiedere nella "current directory" od in una directory che è nel search path.
- Aggiungete le directory che contengono i vostri file al search path.
- Default, I file (librerie) di MATLAB sono incluse nel search path.
- Per visualizzarle:
  - Scrivete path nella CW
  - Menu>File>Set Path…

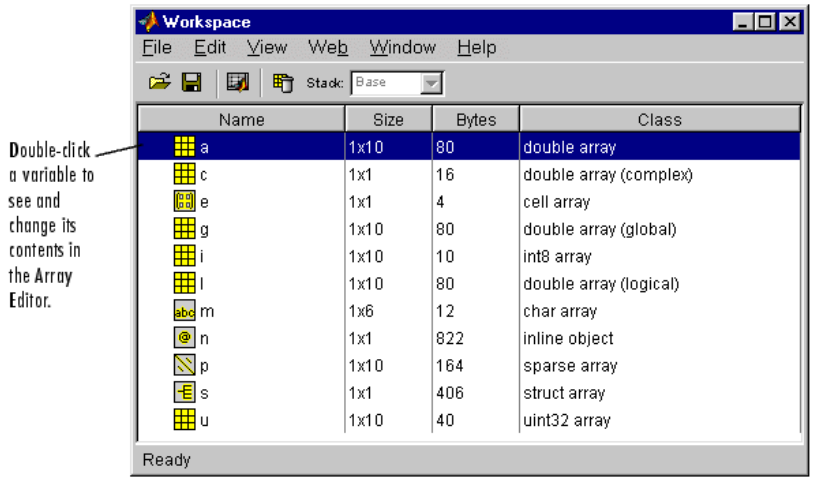
13

Workspace Browser

- Il workspace è l'insieme delle variabili (tipicamente array) allocate durante la sessione.
- Aggiungere varibili al workspace si fa eseguendo funzioni, M-files, e caricando (load) sessioni passate.
- Per vedere il contenuto del WS si usa il Workspace Browser o nella CW le funzioni who e whos.
- Per cancellare variabili dal WS:
  - Graficamente con **Delete**
  - Oppure usare clear.
- Il WS si cancella alla chiusura di MATLAB.
- Salvare il WS
  - **File>Save Workspace As**
  - Funzione save
  - Viene salvato un file binario .mat

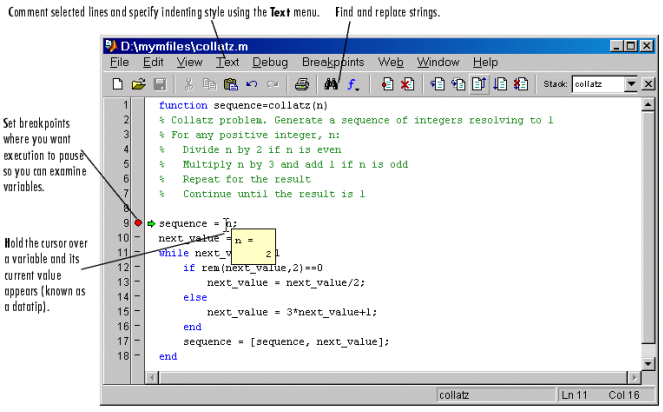
14

Workspace Browser



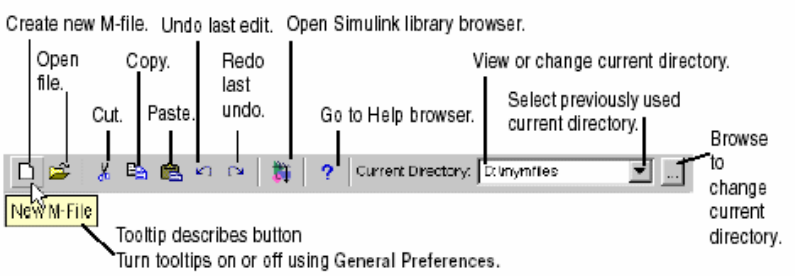
Editor/Debugger

- Usato per creare e debuggare M-files
- Gli M-File i programmi di MATLAB (script o funzioni)
- GUI per modificare o creare gli m file,



- Eseguito dalla WC con: "*edit (nomefile)*"

Desktop Toolbar



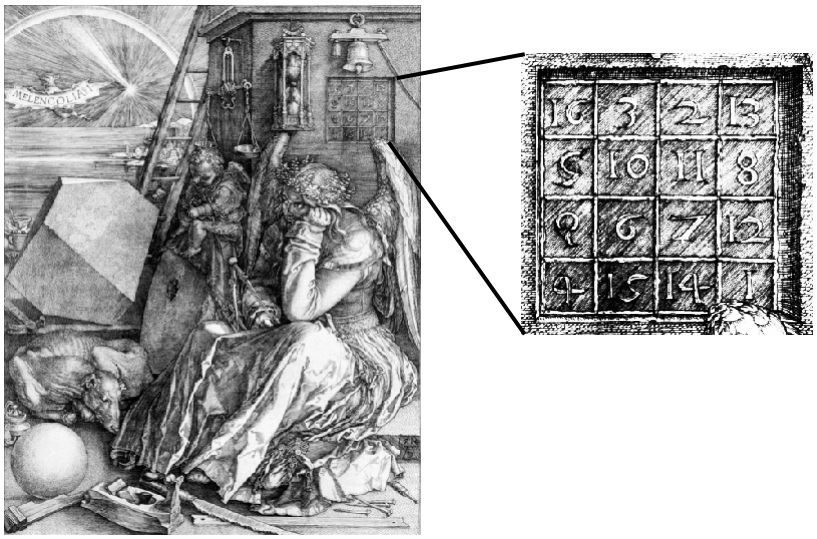
17

Calcolo Matriciale

- Le Matrici sono array rettangolari di numeri.
- Un significato speciale va attribuito alle matrici 1x1 cioé gli scalari e 1xN o Nx1 cioè ai vettori riga o colonna
- MATLAB lavora direttamente con le matrici (array)
  - Differentemente dagli altri linguaggi o programmi

18

Dürer's matrix - magic



19

Creare matrici

- Molti modi, eccone alcuni:
  - Creare una lista di elementi
  - Load da file
  - Crearle con le funzioni di MATLAB
  - Crearle con le funzioni scritte da voi.
- Iniziamo con la matrice di Dürer's:
  - Separare gli elementi su una riga con spazi o virgole
  - Usare il punto e virgola per delimitare le righe
  - Circondare la lista di elementi con [ ].

20

MATLAB – Entering matrices - Example

» A = [16 3 2 13; 5 10 11 8;9 6 7 12;4 15 14 1]

- MATLAB vi mostra la matrice che avete inserito.

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

- E la memorizza nel workspace

21

Alcune operazioni sulle matrici

- **sum**

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

→

```
sum(A)
MATLAB replies with
ans =
    34    34    34    34
```

- Se non fornite una variabile a cui assegnare il valore, MATLAB assegna il risultato alla varibile “ans” (che sta per *answer*) .
- Usando “sum” avete calcolato un vettore riga contenete la somma delle colonne di A.

22

Ancora operazioni

- **trasposto**

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

→

```
A'
produces
ans =
    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1
```

- **Esempio:**

```
sum(A')'
```

produces a column vector containing the row sums

```
ans =
    34
    34
    34
    34
```

23

Ancora operazioni

- **diag**

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

→

```
diag(A)
```

produces

```
ans =
    16
    10
     7
     1
```

- **Esempio:**

```
sum(diag(A))
```

produces

```
ans =
    34
```

24

Indici

- **Uso degl indici**
  - L'elemento nella riga *i* e colonna *j* di A si indica con A(*i*,*j*).
  - Ex: A(4,2)

- **Esempio** : calcolare la somma degli elementi della quarta colonna

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
```

This produces

```
ans =
    34
```

Ancora matrici

- È possibile riferirsi ad un elento della matrice usando un solo indice, A(k) ossia come se ci riferissimo ad un vettore NxM (dimesioni della matrice)
- Il vettore è visto come un unico vettore colonna formato dalle colonne della matrice
  - $k=pN+m$

  Dove N è il numero di righe, p,m sono due interi tali che $p=0,1,...,M$ and $m=0,1,...,N-1$
- Esempio: A(8) è anche A(4,2).

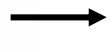
```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

26

Ancora indici

- Se fate riferimento ad un elemento che “fuori” dalla matrice vi da un errore: Example:

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```



```
t = A(4,5)
Index exceeds matrix dimensions.
```

- Se aggiungete un valore fuori dal size della matrice MATLAB aumenta il size per inserire il nuovo arrivato.

Ex:

```
X = A;
X(4,5) = 17

X =
    16     3     2    13     0
     5    10    11     8     0
     9     6     7    12     0
     4    15    14     1    17
```

27

I due punti!!

1)

```
1:10
```

is a row vector containing the integers from 1 to 10

```
1     2     3     4     5     6     7     8     9    10
```

2)

To obtain nonunit spacing, specify an increment. For example,

```
100:-7:50
```

is

```
100    93    86    79    72    65    58    51
```

and

```
0:pi/4:pi
```

is

```
0    0.7854    1.5708    2.3562    3.1416
```

28

Ancora :

- Le espressioni relative agli indici con i ":" indicano parti della matrice.

```
A(1:k,j)
```

- ":" indica tutta la riga o la colonna
- end indica l'ultima riga o colonna

```
A(:,end)
```

29

Espressioni

- MATLAB fornisce espressioni matematiche, ma differentemente dagli altri linguaggi, queste coinvolgono direttamente le matrci.
- I building blocks delle espressioni sono
  - Variabili
  - Numeri
  - Operatori
  - Funzioni

30

Variabili

- Non devono essere dichiarate
  - Trova il nome di una nuova varibile e la crea
  - Se la varibile esiste cambia il suo contenuto (e se necessario alloca altra memoria)

```
num_students = 25
```

- Le variabili iniziano con lettere dell'alfabeto e possono contenere numeri, underscores, etc.
- MATLAB usa solo I primi 31 caratteri per il nome della variabile.
- MATLAB è case sensitive
- Per vedere il contenuto di una varibile digitate il suo nome

31

Numeri

- MATLAB usa la notazione decimale convenzionale, con il punto per i decimali opzionale e il segno opzionale.
- *Notazione scientifica* usa la lettera e per specificare la scala.
- *Numeri immaginari* usa i o j come suffisso

```
3              -99            0.0001
9.6397238      1.60210e-20    6.02252e23
1i             -3.14159j      3e5i
```

32

Operatori

- + Addition
- - Subtraction
- * Multiplication
- / Division
- ^ Power

33

Funzioni

- Funzioni matemetiche elementari:
  - abs, sqrt, exp, e sin.
  - *help elfun*
- Funzioni avanzate o specifiche per matrici
  - *help specfun*
  - *help elmat*

- Alcune funzioni sono integrate
- sqrt, sin, etc
- Altre sono definite come M-file
  - gamma, sinh, etc
  - Potete vedere il codice e modificarle

34

Funzioni

- Alcune funzioni forniscono delle costanti

`pi`	3.14159265...
`i`	Imaginary unit, $\sqrt{-1}$
`j`	Same as `i`
`eps`	Floating-point relative precision, 2^{-52}
`realmin`	Smallest floating-point number, 2^{-1022}
`realmax`	Largest floating-point number, $(2-\varepsilon)2^{1023}$
`Inf`	Infinity
`NaN`	Not-a-number

- Infinito è generato dividendo un numero per 0 (massimo numro realmax)
- Not-a-number viene da espressioni tipo 0/0 or Inf-Inf
- Le funzioni possono essere sovrascritte
  - Ex: eps=1e-6
- Se la cancello ripristino la varibile di default (**clear eps**)

35

Esempi

```
rho = (1+sqrt(5))/2
rho =
    1.6180

a = abs(3+4i)
a =
     5

z = sqrt(besselk(4/3,rho-i))
z =
   0.3730+ 0.3214i
```

36

Esempi

zeros	All zeros
ones	All ones
rand	Uniformly distributed random elements
randn	Normally distributed random elements

```
Z = zeros(2,4)
Z =
     0     0     0     0
     0     0     0     0
```

```
N = fix(10*rand(1,10))
N =
     4     9     4     4     8     5     2     6     8     0
```

```
F = 5*ones(3,3)
F =
     5     5     5
     5     5     5
     5     5     5
```

```
R = randn(4,4)

R =
    1.0668    0.2944   -0.6918   -1.4410
    0.0593   -1.3362    0.8580    0.5711
   -0.0956    0.7143    1.2540   -0.3999
   -0.8323    1.6236   -1.5937    0.6900
```

37

Esempi - Concatenazione

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

↓

```
B = [A  A+32; A+48  A+16]
```

↓

```
B =
    16     3     2    13    48    35    34    45
     5    10    11     8    37    42    43    40
     9     6     7    12    41    38    39    44
     4    15    14     1    36    47    46    33
    64    51    50    61    32    19    18    29
    53    58    59    56    21    26    27    24
    57    54    55    60    25    22    23    28
    52    63    62    49    20    31    30    17
```

38

Esempi

- **Cancellare righe o colonne**
  - Uso della matrice (vettore) vuoto [ ]
  -

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
X = A;
```

Then, to delete the second column of X, use

```
X(:,2) = []
```

This changes X to

```
X =
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

39

Esempi

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
A + A'

ans =
    32     8    11    17
     8    20    17    23
    11    17    14    26
    17    23    26     2
```

```
A'*A

ans =
   378   212   206   360
   212   370   368   206
   206   368   370   212
   360   206   212   378
```

```
d = det(A)

d =
     0
```

```
X = inv(A)
```

you will get a warning message

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.175530e-017.
```

40

Ancora Matrici

- Al di fuori dell'algebra lineare le matrici sono vettori bidimensionali
- **Le operazioni aritmetiche sono eseguite elemento per elemento**

- + e – sono invarianti
- Per le altre è necessario indicare con il punto il tipo di operazione.

+	Addition
-	Subtraction
.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Unconjugated array transpose

41

Esempi

```
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

→ `A.*A` →

```
ans =
   256     9     4   169
    25   100   121    64
    81    36    49   144
    16   225   196     1
```

```
n = (0:9)';
```

→ `pows = [n  n.^2  2.^n]` →

```
pows =
     0     0     1
     1     1     2
     2     4     4
     3     9     8
     4    16    16
     5    25    32
     6    36    64
     7    49   128
     8    64   256
     9    81   512
```

42

Rappresentazione grafica dei dati in MATLAB

43

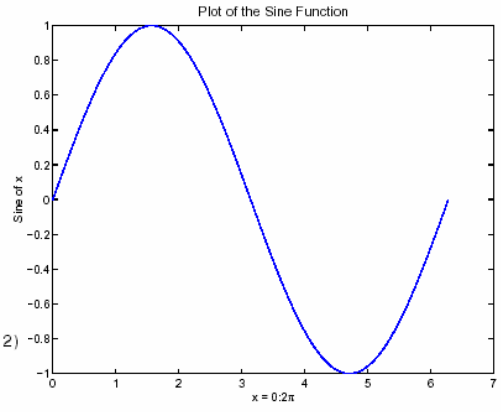
Creare un plot

- Si usa la funzione plot. (È polimorfa a seconda degli argomenti in ingresso)
- Se y è un vettore, *plot*(*y*) produce un grafico lineare degli elementi *y* verso il loro indice.
- Se sono due I vettori in ingresso plot(x,y) produce il grafico di y verso x.
- EX:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

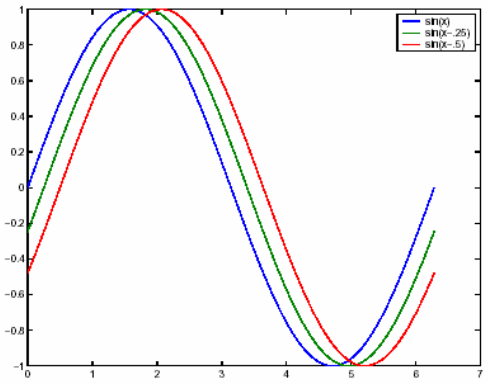
- label

```
xlabel('x = 0:2\pi')
ylabel('Sine of x')
title('Plot of the Sine Function','FontSize',12)
```



Multiple Data Sets

```
y2 = sin(x-.25);
y3 = sin(x-.5);
plot(x,y,x,y2,x,y3)
```




```
legend('sin(x)','sin(x-.25)','sin(x-.5)')
```

45

Stili e Colori

```
plot(x,y,'color_style_marker')
```

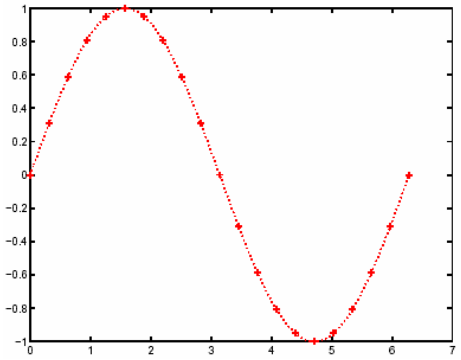
- *color_style_marker* è una stringa con al massimo 4 caratteri
  - a color, a line style, and a marker type
- Colori: 'c', 'm', 'y', 'r', 'g', 'b', 'w', and 'k'.
  - cyan, magenta, yellow, red, green, blue, white, and black
- Stile della linea
- '-' solid, '--' dashed, ':' dotted, '-.' dash-dot
- Marker
  - '+', 'o', '*', 'x', 's', 'd', '^', 'v' , '>' , '<', 'p' ,'h'
- Se specificate il marker e non la linea questa non viene disegnata

46

Esempi

- **Esempio:** `plot(x,y,'ks')`
  disegna quadrati neri per ogni punto, ma non la linea.
- **Esempio:** `plot(x,y,'r:+')`
  disegna una linea puntinata rossa ed I marker sono +.

```
x1 = 0:pi/100:2*pi;
x2 = 0:pi/10:2*pi;
plot(x1,sin(x1),'r:',x2,sin(x2),'r+')
```



Plot di numeri complesi

- plot ingnora la parte immaginaria.
- Se l'argomento complesso è uno
  - plot disegna la parte reale vs quella immaginaria:

Z array di numeri complessi

`plot(Z)` ⟷ `plot(real(Z),imag(Z))`

48

Ancora su plot

- Chiamando il comado plot viene generata una nuova figura
- hold permette di aggiungere plot ad una figura esistente
- Se scrivete "*hold on*" matlab non genera una nuova figura (se ne esiste una)
  - Se ce ne sono diverse viene selezionata la corrente
  - Per far diventare una figura la figura corrente:
    - Cliccare con il mouse sulla figura
    - Comando figure
- Se scrivete "*hold off"* ritorna normale

49

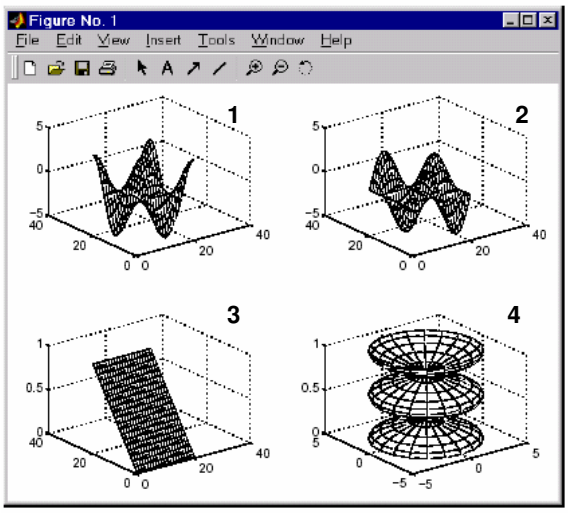
Plot multipli

- Si usa il comando subplot
- subplot(n,m,p)
- Divide la figura in una matrice nxm selezionando il p-esimo come corrente.
- La nuerazione è per righe

50

Esempio

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



Gli assi della figura

- Comado axis
- **Serve ad**
  - **impostare i limiti**
    - Altrimenti li calcola MATLAB
  - Reimpostare la selezione automatica
- Il comando grid mostra la griglia

2D `axis([xmin xmax ymin ymax])`

3D `axis([xmin xmax ymin ymax zmin zmax])`

`axis auto`

`grid on` `grid off`

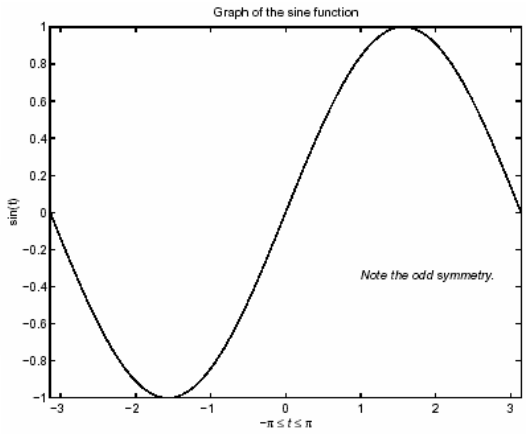
52

Label e titolo

- xlabel, ylabel, zlabel
- title
- text

- Esempio:

```
t = -pi:pi/100:pi;
y = sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel('-\pi \leq {\itt} \leq \pi')
ylabel('sin(t)')
title('Graph of the sine function')
text(1,-1/3,'{\itNote the odd symmetry.}
```



53

PROGRAMMAZIONE in MATLAB

54

Gli m-file

- I file che contengono codice sono gli **M-files**.
- Si creano con un text editor
- Si usano come le funzioni matlab
- Sono di due tipi:
  - **Scripts**: non accettano ingressi, non ritornano valori. Operano sui dati del workspace.
  - **Functions**, accettano ingressi, non ritornano uscite. Le variabili che usano sono locali.
- Si possono organizzare ed inserire nel path (come librerie)
  - Se I nomi delle funzioni sono duplicati conte l'odine del search path

55

Scripts

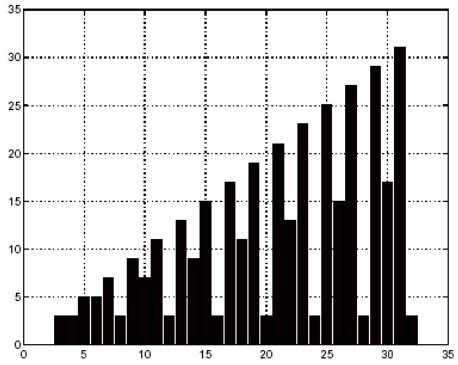
- Se invocate uno script, vengono eseguiti I comandi nel file
- Operano sul WS e possono creare nuove variabili nel WS
- Non ritornano nulla
- Le variabili create restano nel WS e posso essere usate dopo
- Possono generare plot

56

Esempio

- File *magicrank.m*

```
% Investigate the rank of magic squares
r = zeros(1,32);
for n = 3:32
   r(n) = rank(magic(n));
end
r
bar(r)
```



57

Funzioni

- Accettano input e tronano un output
- Il nome del file e della funzione deve essere lo stesso
- Operano su varibili locali
  - Non vedono il WS

58

Esempio

- File *rank.m*
  - *toolbox/matlab/matfun*"

```
function r = rank(A,tol)
%   RANK Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

s = svd(A);
if nargin==1
   tol = max(size(A)') * max(s) * eps;
end
r = sum(s > tol);
```

- La prima liena
  - inizia con la parola function.
  - Definisce il nome della funzione e gli argomenti
- Le altre (fino alla prima bianca forniscono l'help
  - "*help rank*"

59

Esempio

- Il resto è codice eseguibile
- Le varibili *s r*, *A tol*, sono locali;
  - Non sono visibili nel WS.
- Argomenti va

```
rank(A)
r = rank(A)
r = rank(A,1.e-6)
```

  - nargin e nargout contano gli argomenti

60

Veribili Globali

- global definisce una variabile globale
- Deve essere ripetuta in ogni funzione e prima dell'uso della variabile
- Esempio:

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

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

61

Vettorizzazione

- Dove usereste un ciclo (for, do, while)

```
x = .01;
for k = 1:1001
   y(k) = log10(x);
   x = x + .01;
end
```

- Cercate scriverlo in modo vettoriale

```
x = .01:.01:10;
y = log10(x);
```

62

Preallocazione

- Se non potete vettorizzare il calcolo cercate di preallocare le varibili usate nel ciclo
- Example:

```
r = zeros(32,1);
for n = 1:32
    r(n) = rank(magic(n));
end
```

- Altrimenti MATLAB opera N allocazioni
- La preallocazione velocizza i cicli.

63

Controllo di flusso

- if
- switch / case
- for
- while
- continue
- break

64

Esempio - if

```
if rem(n,2) ~= 0
   M = odd_magic(n)
elseif rem(n,4) ~= 0
   M = single_even_magic(n)
else
   M = double_even_magic(n)
end
```

65

if – con matrici

- *if A == B*, ...
  - È legale, è fa ciò che ci si aspetta se A e B sono scalari.
  - A e B matrici, A == B non controlla l'uguaglianza, controlla quali elementi sono uguali;
    - Il risulatato è una matrice di 0 ed 1.
  - se A e B non hanno la stessa dimensione da errore
- Modo corretto:

  if isequal(A,B)

- Altre funzioni utili:

  isequal, isempty , all , any

66

Switch

```
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
```

- Differentemente dal C lo switch di MATLAB con continua in mancanza del break

67

For e While

```
for n = 3:32
   r(n) = rank(magic(n));
end
r
```

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

68

MATLAB - Break

- The break statement lets you exit early from a for or while loop. In nested loops, break exits from the innermost loop only.
- Example:

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
   x = (a+b)/2;
   fx = x^3-2*x-5;
   if fx == 0
      break
   elseif sign(fx) == sign(fa)
      a = x; fa = fx;
   else
      b = x; fb = fx;
   end
end
x
```

69

Altre strutture dati

- MATLAB include altre strutture dati quali
  - Multidimensional Arrays
  - Cell Arrays
  - Characters and Text
  - Structures

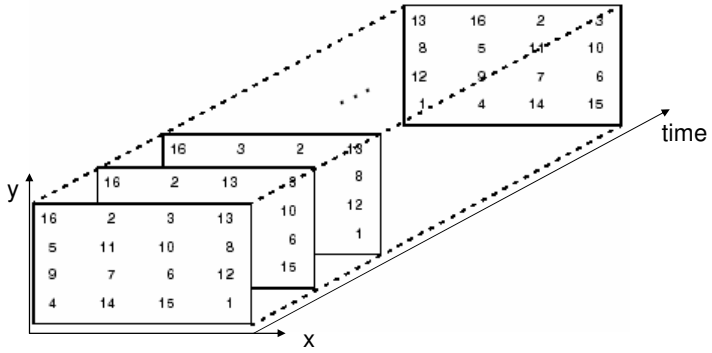
70

Multidimensional Arrays

- Sono array con più di due indici
- Li creiamo con zeros, ones, rand, or randn con più di due argomenti.

- Esempio:
```
R = randn(3,4,5);
```
crea un array 3x4x5 con alementi random.

- Un modo grafico per rappresentarli è:



71

Cell arrays

- Sono collezioni di multidimensional arrays.
- Si creano con la funzione cell.
- Spesso, i cell array sono creati racchiudento colleezioni di cose fra le parentesi graffe, { }.
- Uso le graffe con un indice per accedere al contenuto della cell
- Esempio:
```
C = {A sum(A) prod(prod(A))}
```

- Visualizzando C ottenete
```
C =
    [4x4 double]    [1x4 double]    [20922789888000]
```

72

Ancora Cell arrays

- I Cell arrays contengono copie degli array non puntatori
  - Se cambiate il A il contenuto dell Cell Array non cambia
- NOTA
  - I Multidimensional Array contengono matrici della stassa dimensione
  - Un cell array contine matrici di dimensionedifferente
- Esempio:

```
M = cell(8,1);
for n = 1:8
   M{n} = magic(n);
end
M
```

73

Characters and text

- Il testo è contenuto fra le virgolette singole.
  - Esempio: *s = 'Hello'*
    - *s* è un array di caratteri (dimensione 5)
- Si possono concatenare
  - Esempio: *h = [s, ' world']*
    - h = Hello world
  - Esempio: *v = [s; 'world']*

```
v =
   Hello
   world
```

74

Ancora Text and characters

- Per manipolare strighe di dimensione differente ci sono due alternative
  - Padding
  - cell array of strings.
- La funzione char accetta un arbitrario numero di stringhe
  - Aggiunge degli spazi per renderle della stessa lunghezza
  - Ogni array è contenuto in una riga

```
S = char('A','rolling','stone','gathers','momentum.')
```

```
S =
A
rolling
stone
gathers
momentum.
```

- Alternatively, you can store the text in a cell array.

```
C = {'A';'rolling';'stone';'gathers';'momentum.'}
```

```
C =
    'A'
    'rolling'
    'stone'
    'gathers'
    'momentum.'
```

75

Structures

- Structures sono multidimensional arrays con elementi a cui si accede mediante indici testuali
- Esempio:

```
S.name = 'Ed Plum';
S.score = 83;
S.grade = 'B+'
```

```
S =
     name: 'Ed Plum'
    score: 83
    grade: 'B+'
```

76

Ancora Structure

- Possiamo aggiungere elementi ad una struttura

```
S(2).name = 'Toni Miller';
S(2).score = 91;
S(2).grade = 'A-';
```

- La funzione struct aggiunge un intero record

```
S(3) = struct('name','Jerry Garcia',...
              'score',70,'grade','C')
```

- Accesso alle strutture

Esempi: `S.score` ↔ `[S(1).score, S(2).score, S(3).score]`

`[S.score]` ↔

```
ans =
    83    91    70
```

77

File I/O

- **load**
  - Memorizza variabili nel workspace
  - load
    - load filename
    - load filename X Y Z
    - load filename –ascii
    - load filename –mat
    - S = load(...)
- *load* memorizza tutte le variabili prese in *matlab.mat*, se esiste
- *load filename* memorizza tutte le variabili da filename
  - Se l'estensione è .mat lo tratta come tale altrimenti pensi sia ASCII.
- *load filename X Y Z* memorizza le variabili X Y Z da filename MAT-file.

78

File I/O

- **save**
  - Salva le variabili del workspace
    - save
    - save filename
    - save filename var1 var2 ...
    - save ... *option*
    - save('filename', ...)
  - *save* stores salva tutte le variabili del WS nel file matlab.mat.
  - *save filename* salva tutte le variabili del WS nel file filename
  - *save filename var1 var2 ...* salva le variabili var1 … del WS in filename.

79