Introduzione a MATLAB

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Padova

CHE COSA È MATLAB

MATLAB è:

•un **linguaggio di programmazione**

•un **ambiente di calcolo scientifico** con routines altamente specializzate

L'elemento base sono le **matrici**

dimensione M X N = matrice ad M righe ed N colonne.

Casi particolari:

M=1 ➔ vettore riga

N=1 ➔ vettore colonna

M=N=1 ➔ scalare

Fondamentale l'uso dell' **help !**

PERCHE' USARE MATLAB ?

Include migliaia di **funzioni** che possono essere

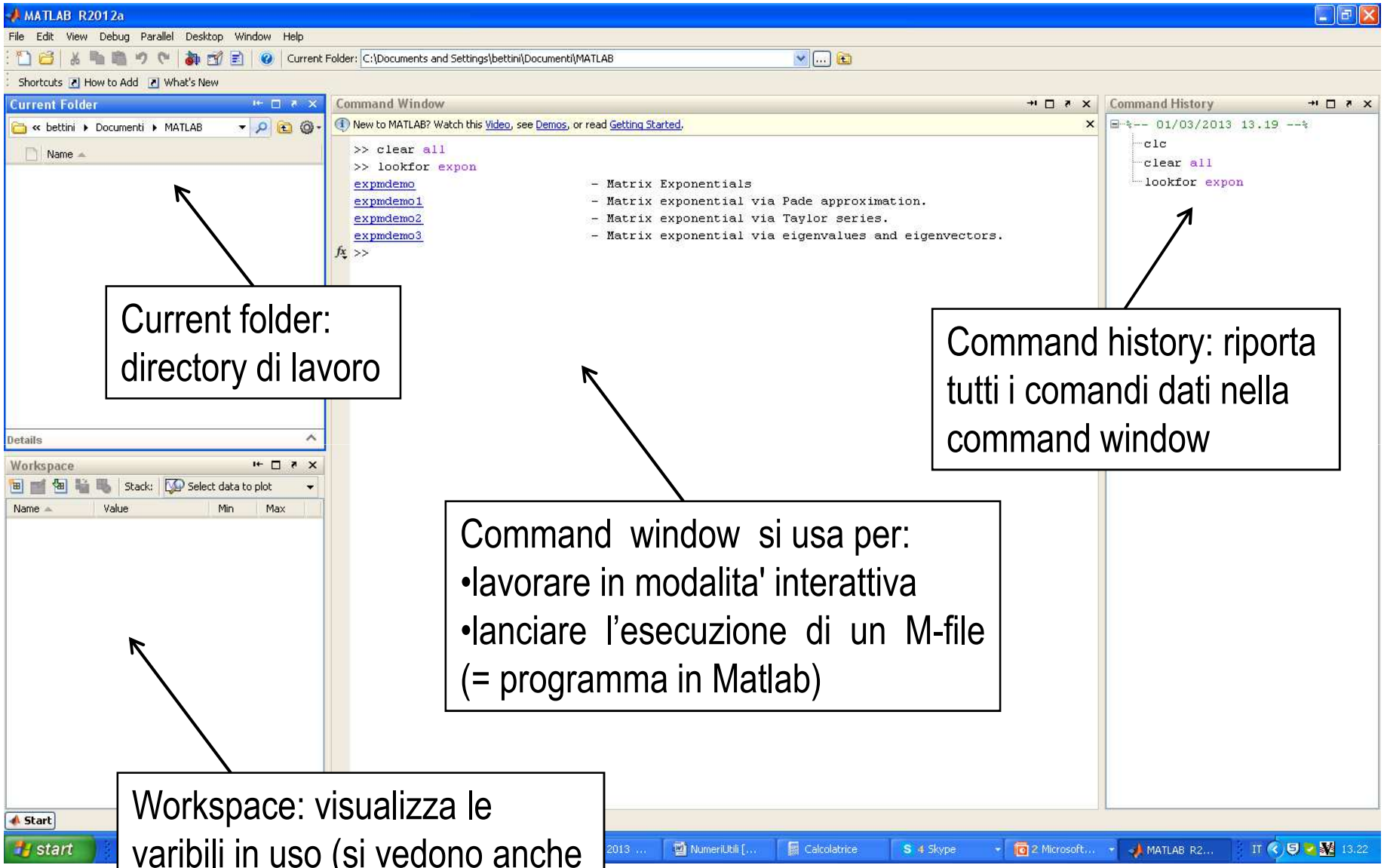
- **built-in** (es. *det*, *inv*, ...) o
- incluse in **toolbox** opzionali la cui licenza si acquista a parte
- scritte direttamente da noi

Tutte queste funzioni vengono poi direttamente richiamate nei programmi che andiamo a scrivere risultando così più **semplice** e **veloce** rispetto ad usare linguaggi come Fortran, Pascal, C, ...

Poichè il codice Matlab viene scritto in file di testo, è immediatamente trasferibile da una piattaforma ad un'altra (UniX, Mac, Windows,...).

Un codice Matlab viene di fatto interpretato, ma è anche possibile compilarlo tramite il **Matlab Compiler** (che però in questo corso non vedremo) rendendo così al contempo "chiuso" il sorgente e più veloce l'esecuzione.

Come si presenta Matlab:



Current folder: directory di lavoro

Command history: riporta tutti i comandi dati nella command window

Command window si usa per:
- lavorare in modalita' interattiva
- lanciare l'esecuzione di un M-file (= programma in Matlab)

Workspace: visualizza le varibili in uso (si vedono anche con i comandi **who e whos**)

Alcuni comandi utili dalla Command Window

>> <FrecciaSu>,<FrecciaGiu>	Richiama i comandi già dati dalla Command Window
>> help <NomeComando>	Consulta l'help in linea per il comando NomeComando Fare molta attenzione alle funzioni che suggerisce!
>> lookfor <NomeComando>	Ricerca nel manuale la parola chiave NomeComando e riporta tutte le funzioni che la contengono
>> ctrl C	Blocca l'esecuzione in corso
>> clc	Pulisce la command window
>> clear all	Pulisce il workspace
>> close all	Chiude tutte le finestre grafiche
>> demos	Dimostrazioni
>> exit	Chiude Matlab

N.B. - Tutte le istruzioni hanno effetto "eco" per eliminarlo bisogna concluderle con ";"
-Vicino al prompt (>>) c'è fx → richiama l'elenco delle funzioni build-in di Matlab

Dalla command window si possono fare operazioni come con una **calcolatrice**. All'enter Matlab fa le sue elaborazioni e fornisce i risultati memorizzati nella variabile “ans”, se non viene specificato un nome

Es:

```
>> 1253*5
ans =
      6265
>>
```

L’uso interattivo della command window è spesso poco conveniente e noioso (è difficile trovare gli errori, se voglio ripetere le operazioni più volte devo riscrivere tutto ogni volta...), ma a volte può essere utile, specie per iniziare.

E’ molto più utile utilizzare la command window per invocare **comandi** o **funzioni** (=programmi con argomenti)

LE VARIABILI IN MATLAB: matrici e vettori

Matlab non richiede una preassegnazione delle variabili, l'assegnazione è fatta direttamente o dalla command window o dal programma.

IMPORTANTE: Matlab è **case-sensitive**: a≠A

I **nomi delle variabili** sono a piacere (ma non possono cominciare con un numero, includere spazi e caratteri speciali, es. *, e non dovrebbero coincidere con nomi riservati di comandi e funzioni, es. pi, min, max…).

Come già detto l'elemento base di Matlab sono le **matrici** di dimensione MxN e si inseriscono per righe racchiuse fra parentesi [ ]:

```
A=[1 2 3
4 5 6];
```

oppure

```
A=[1 2 3; 4 5 6];
```

- i diversi elementi delle righe sono separati da spazio o ","
-le righe si inseriscono o andando a capo o con il ";"
-la funzione **size** fornisce la dimensione delle variabili:
size(A) → 2 x 3 (2 righe, 3 colonne)

LE VARIABILI IN MATLAB: matrici e vettori

I vettori sono particolari matrici fatte di una sola riga (1xN) o di una sola colonna (Mx1)

a=[1 2 3] vettore riga (size(a)= 1x3 (1 riga e 3 colonne))

b=a' vettore colonna (size(b)=3x1)

il simbolo " ' " corrisponde alla trasposizione!

La dimensione di un vettore si vede anche con il comando

length(a)

N.B. se l'argomento di length è una matrice restituisce max(size(matrice))

Es: length(A)=3

Si può sempre accedere ad un elemento della matrice o del vettore specificando tra parentesi ( ) rispettivamente gli indici o l'indice dell'elemento

A(2,2)=5

a(2)=2

b(3)=3

IMPORTANTE:
Gli indici sono **interi strettamente positivi:** cioè partono da **1**

Estrazioni di SOTTOMATRICI

B=[2 –2 10 3
7 1 -3 4
1 5 6 -4]

IMPORTANTE:
-l'operatore " **:** " significa tutte le righe o colonne
- l'operatore **end** indica l'ultimo elemento di riga o colonna

Per accedere ad un **elemento**:
x=B(3,4) ➔ x = –4
Per accedere ad **un'intera riga**, ad es. la 2
x=B(2,:) ➔ x = [7 1 -3 4]
Per accedere ad **un'intera colonna**, ad es. la 3
x=B(:,3) ➔ x = [10
-3
6]
Per accedere ad **una sottomatrice**, ad es.la 2X2 in basso a destra
x=B(2:3,3:4) ➔ x = [-3 4
6 -4] è equivalente a:
x=B(2:**end**,3:**end**)

Costruzioni di MATRICI

Le matrici si possono costruire:

- “affiancando” matrici piu' piccole (purchè le dimensioni siano compatibili);
- tramite un’allocazione dinamica: definisco un solo elemento di una matrice o definisco un nuovo elemento in una posizione che eccede la vecchia dimensione della matrice, gli altri elementi vengono definiti di imperio e posti uguali a zero:
  C(5,4)=4 ➔ crea una matrice C di 5 righe e 4 colonne fatta di tutti zeri tranne l’elemento (5,4) che vale 4
  A(3,3)=-1 ➔ fa diventare la nostra matrice A 3x3 con una terz riga [0 0 -1]
- tramite funzioni build-in di Matlab, ad es:

A=eye(10)	matrice identità 10x10
A=zeros(3,5)	matrice 3x5 con elementi tutti nulli
A=ones(3,5)	matrice 3x5 con elementi tutti pari a 1
A=diag([3, 5, 6])	matrice 3x3, con elementi sulla diagonale specificati

Elenco principali funzioni build-in per costruzione matrici speciali

zeros	matrix of zeros
ones	matrix of ones
eye	identity matrix
diag	Diagonal matrices and diagonals of a matrix
toeplitz	Toeplitz matrix
magic	Magic square
hilb	Hilbert matrix
invhilb	Inverse Hilbert matrix
vander	Vandermonde matrix
pascal	Pascal matrix
hadamard	Hadamard matrix
hankel	Hankel matrix.
rosser	Classic symmetric eigenvalue test problem
wilkinson	Wilkinson's eigenvalue test matrix

Costruzioni di VETTORI

Con elementi equispaziati:

-tramite il simbolo “ : ”

x=1:10 ➔ x=[1 2 3 4 ….. 10] vettore con passo 1

x=3:2:10 ➔ x=[3 5 7 9] vettore con passo 2

- tramite la funzione linspace:

y = linspace(Min, Max, N) genera un vettore di N (100 se non specificato) elementi equispaziati tra Min e Max

- tramite la funzione logspace

z= logspace(Min, Max, N) genera un vettore di N elementi logaritmicamente equispaziati tra le decadi 10^Min e 10^Max

Costruzioni di matrici e vettori casuali

A=rand(3,5)	matrice 3x5 con elementi casuali distribuiti uniformemente nell'intervallo [0,1]
A=randn(3,5)	matrice 3x5 con elementi casuali con distribuzione gaussiana con media nulla e deviazione standard 1, come $\mathrm{N}(0, 1)$
A=randi(n,N)	matrice NxN di elementi casuali **interi** distribuiti uniformemente nell'intervallo [1,n]

Sfruttando le trasformazioni lineari si possono creare varie distribuzioni di probabilita'

u=15+5*rand(50,1)	vettore colonna di lunghezza 50 con elementi tratti da una distribuzione uniforme tra nell'intervallo 15 - 20
v=10+2*randn(50,1)	vettore colonna di lunghezza 50 con elementi tratti da una distribuzione gaussiana con media 10 e deviazione standard 2

OPERAZIONI di base su MATRICI e VETTORI

•Trasposizione di matrice: Per la trasposizione di matrice si usa come in algebra lineare l'apice : " ' "

a=[1 2 3] b=a' ➔ b=[1
2
3]

•Prodotto per uno scalare e somma tra matrici: Matlab esegue in modo intuitivo le operazioni algebriche di base sulle matrici.

A=2*eye(2) → A=[2 0; 0 2]

B=ones(2) → B=[1 1; 1 1]

C=A+B → C=[3 1; 1 3]

Naturalmente, quando si richiede di eseguire somma, differenza e prodotto, le dimensioni delle matrici coinvolte devono essere compatibili.

OPERAZIONI di base su MATRICI e VETTORI

• Moltiplicazione tra matrici: esegue la moltiplicazione solo se si lavora con matrici di dimensione MxN e NxP (cioè nº colonne prima = nº righe seconda) ➔ risultato matrice MxP

• Divisione fra matrici: A/B=A*inv(B)

A\B=inv(A)*B

• Elevamento a potenza: si esegue tramite il simbolo " ^ "ed è definito solo per matrice quadrate, cioè X^2 = X*X (se si prova ad eseguirlo su un vettore si ottiene un messaggio di errore)

•Operazioni elemento per elemento: gli operatori * / \ ^ possono essere preceduti da " . " diventando operazioni elemento per elemento per cui le regole delle dimensioni cambiano drasticamente.

a=[1 2 3] ➔ a.*a=[1 4 9]

a.^3=[1 8 27]

Le funzioni sulle matrici sono infinite e riportarle tutte è impossibile , ma vi si accede facilmente cliccando su “fx” vicino al prompt di Matlab.
Solo per citare qualche sottocategoria:

→MATLAB

→Mathematics

→Arrays and Matrices

→Basic Information (ad es.: disp, display, min, max…)

→Elementary Matrices and Arrays

→Elementary Math

→Trigonometric (ad es.: cos, sin, cosh, sinh, tan, tanh….)

→Exponential

→Complex

→Rounding and Remainder

→Data Analysis

→Descriptive Statistics (mean, std, median,….)

OSSERVAZIONI:

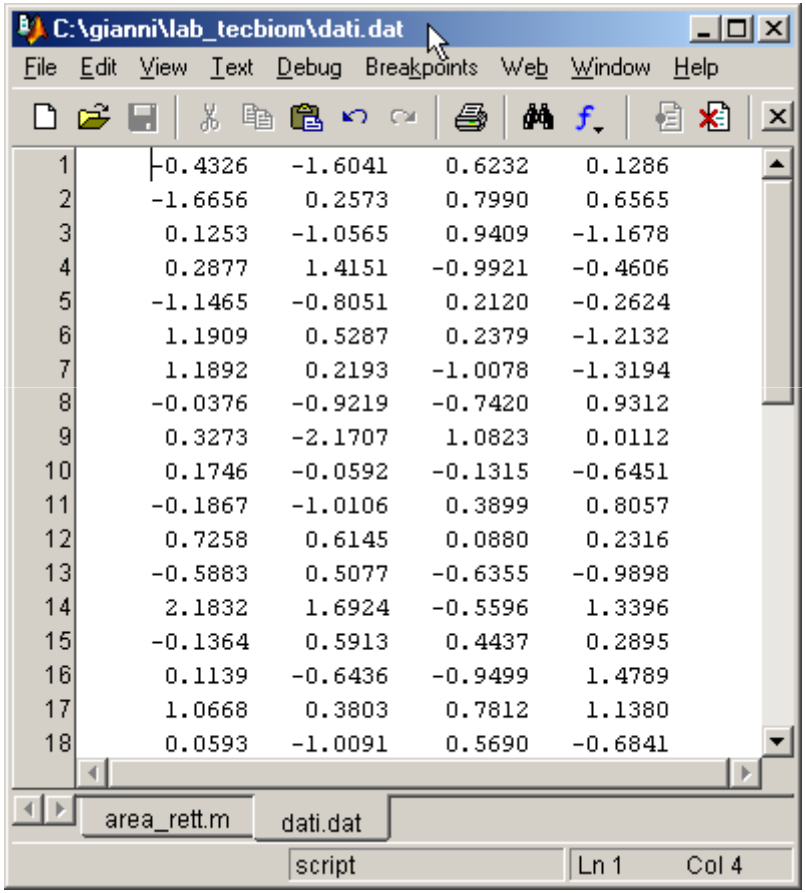
• In Matlab “i” e “j” sono le costanti che rappresentano l’unità immaginaria quindi, se si lavora con numeri complessi è bene evitare nel codice l’uso di i e j come variabili (anche se è consentito));

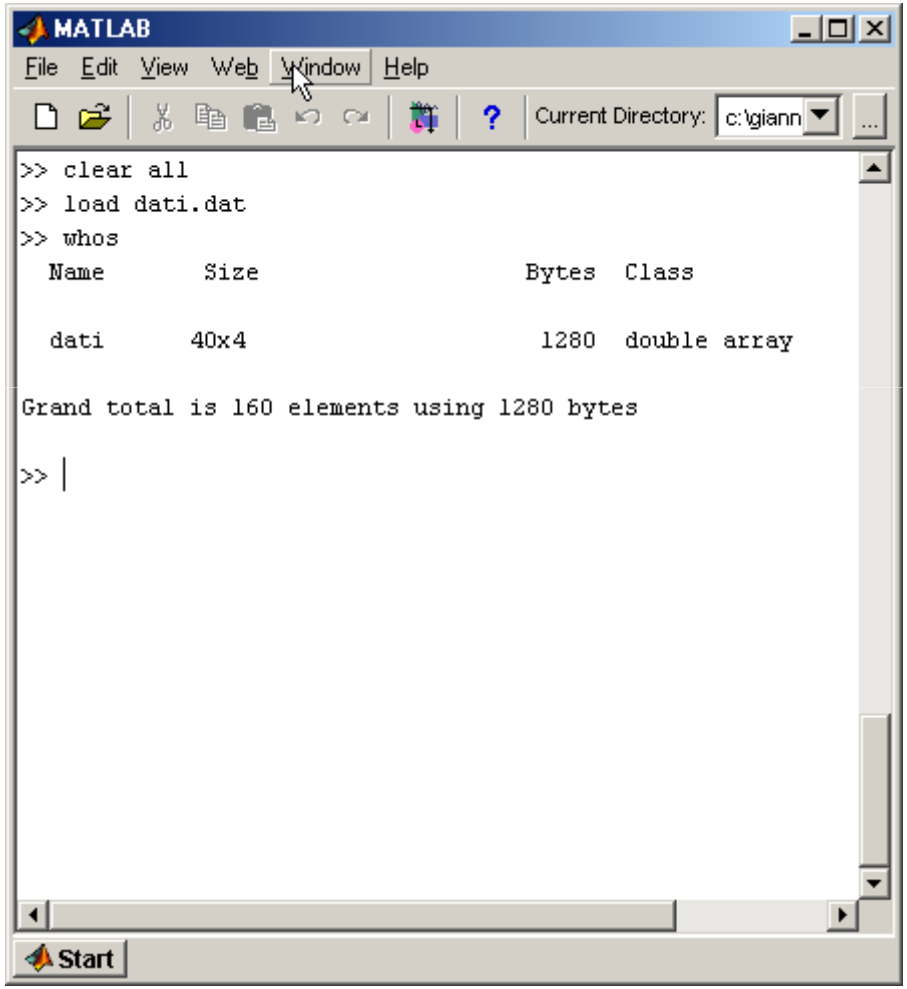
• pi è una variabile predefinita che vale π

•Quando Matlab viene chiuso il workspace viene perso, per salvare e ricaricare le variabili:

save <File>	salva nel file File.mat tutte le variabili del workspace
load <File>	carica nel workspace tutte le variabili presenti in File.mat
save <File> <Variabili>	salva nel file File.mat le variabili in Variabili
load <File> <Variabili>	carica nel workspace le variabili Variabili del file File.mat

Una matrice molto grande come ad es. **dati** (40 X 4) nell'esempio sotto può essere salvata in un file di testo es. **dati.dat** e richiamabile nel workspace con il comando **load dati.dat**





LE VARIABILI IN MATLAB: le stringhe

Le variabili in Matlab possono essere di tipo stringa. Per dichiarare una variabile come stringa è sufficiente racchiuderla fra apici:

S1='pippo'

Più stringhe possono essere concatenate come si fa con vettori e matrici

S2='pluto'

S=[S1,S2] ➔ S=pippopluto

S=[S1,' e ',S2] ➔ S=pippo e pluto

Funzioni che gestiscono le stringhe:

blanks	crea una stringa di spazi
str2num	converte una stringa in numero
num2str	converte un numero in una stringa
ischar	vero se la variabile e' una stringa o carattere

LE VARIABILI IN MATLAB: i record

Le variabili in Matlab possono essere dei record.
Ogni nuovo elemento del record si inserisce inserendo un punto ‘ . ‘
Es:
Studente.nome=‘pippo’
Studente.cognome=‘pluto’
Studente.eta=20

La variabile “studente” è un record con due campi stringa e un campo numerico

LE VARIABILI IN MATLAB: i polinomi

Possono essere gestiti anche polinomi,
Un polinomio di grado N viene inserito in Matlab tramite un vettore di lunghezza N+1 i cui elementi sono i coefficienti del polinomio in ordine decrescente degli esponenti:
X^3+2*x^2-x+10 ➔ P=[1, 2, -1, 10]

Funzioni:
polyval(P,x) : calcola il valore del polinomio in x
roots(p) : radici del polinomio.
poly(r) : determina il polinomio le cui radici sono r.

OPERATORI RELAZIONALI

Gli operatori relazionali più comuni sono:

== uguale
~= diverso da
< minore di
<= minore o uguale etc.

Il risultato di un'operazione relazionale vale

0 se falsa
1 se vera

x=2
x==0 ➔ ans=0
x==2 ➔ ans=1

Gli operatori relazionali possono essere applicati anche alle matrici

OPERATORI RELAZIONALI

Possono essere usati anche come funzioni.

eq	- Equal	==
ne	- Not equal	~=
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	<=
ge	- Greater than or equal	>=

Es:

A=[1 2]

B=[1 -2]

C=eq(A,B) ➔ C=[1, 0]

OPERATORI LOGICI

Gli operatori logici più comuni sono:

& and logico

| or logico

~ not logico

Esempi:

```
>> x=1; y= -1;
>> x>0 & y>0          (questa relazione è falsa)
ans =
     0
>> x>0 | y>0          (questa relazione è vera)
ans =
     1
```

Gli M-files

Una sequenza ordinata di comandi può essere scritta in un M-file (=file testo con estensione m).

Per scrivere M-files ci si può servire di un comune text-editor (es. notepad) o dell'editor interno di Matlab

Per far eseguire un M-file dalla Command Window, è sufficiente scrivere il nome dell'M-file e battere "Invio", oppure si può cliccare sull'icona di esecuzione

Nella esecuzione di un M-file, Matlab si comporta come un interprete.

Gli M-files

La scrittura di un M-file rispetto all'esecuzione di comandi dalla Command Window permette di:

•Sperimentare un algoritmo, senza dover reintrodurre da tastiera, ad ogni variazione dello stesso, una lunga lista di comandi

•Ottenere programmi che possono essere riutilizzati, per esempio cambiando solo i dati

•Scambiare programmi con altri utenti

•Ottenere una documentazione permanente per un lavoro

TIPI DI M-FILES

Scripts: sono files di comandi. Non hanno variabili in entrata e in uscita e operano sulle variabili del workspace

```
% Questo file calcola la radice degli elementi di
% una matrice a, se a>0, altrimenti stampa un messaggio di errore
if a>=0
  a=sqrt(a)
else
  disp('errore')
end
```

Attenzione: nel workspace deve essere stata definita una variabile a

Functions: sono files di comandi con argomenti in entrata e in uscita. Le variabili interne a questi programmi non influenzano le variabili del workspace

```
function a=radfunz(x)
% RADFUNZ(X) calcola la radice degli elementi di X
%            se X>=0, altrimenti stampa un messaggio di errore
%
if x>=0
  a=sqrt(x)
else
  disp('errore')
end
```

Attenzione: Questo file deve essere salvato come radfunz.m

Per cominciare ci concentreremo sugli scripts

DA RICORDARE

Quando dalla command window digitiamo “pippo” + Enter, Matlab:

1. Controlla nel workspace se pippo è una **variabile** ed eventualmente ce ne restituisce il valore

2. Controlla se esiste una **function built-in** di nome pippo ed eventualmente cerca di eseguirla

3. Controlla se esiste nella current directory un **M-file** di nome pippo.m ed ed eventualmente cerca di eseguirlo

4. Controlla se nell’insieme delle cartelle presenti nel **matlabpath** (toolbox + quello che abbiamo inserito noi) esiste una function di nome pippo ed eventualmente cerca di eseguirla

USO DEL % (COMMENTI)

•E’ buona abitudine sia negli scripts che nelle functions inserire dei commenti

•I commenti sono segnalati da %: Matlab ignora tutti i caratteri **dell’intera riga** dopo il %

•Le prime righe di commento di uno script o di una function diventano parte dello help online