Introduzione a Matlab

Gianna M. Del Corso

Dipartimento di Informatica, Università di Pisa, Italy

6 Marzo 2015

Introduzione

- MATrix LABotary
- Ambiente di calcolo scientifico: *Computation, Visualization, Programming*
- Linguaggio di programmazione ad alto livello
- Corredato da una famiglia di applicazioni specifiche Toolbox: signal processing, statistica, optimization, neural networks, etc, ...

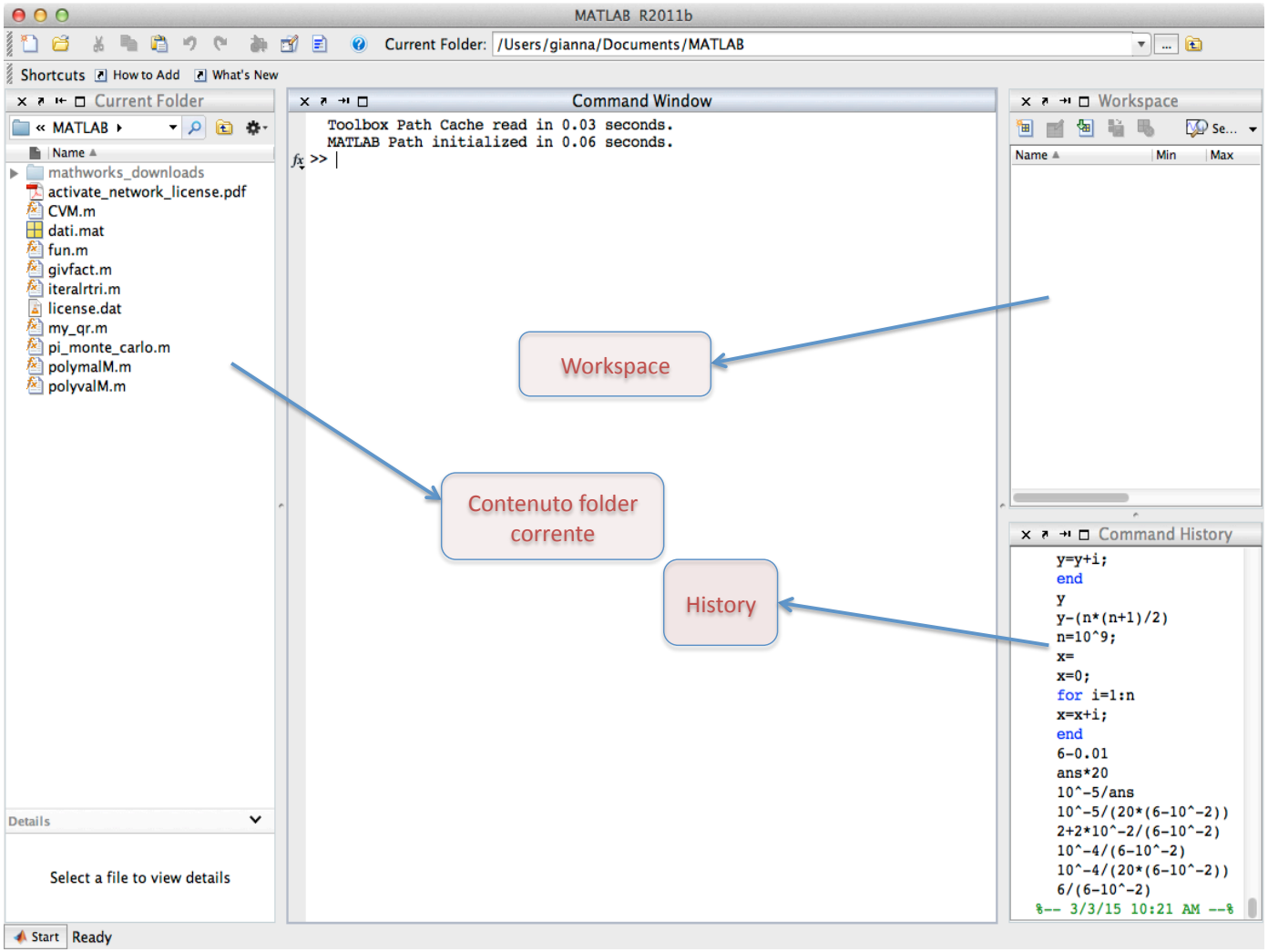
MATrix LABotary

- MATLAB è nato principalmente come programma destinato alla gestione di matrici.È un interprete di comandi in cui l'unità base dei dati è un vettore o una matrice.
- È un software proprietario, marchio proprietario di MathWorks Inc.
- Octave è un software GNU (quindi gratis..) compatibile con MATLAB...
- I comandi possono essere forniti interattivamente o contenuti in files su disco (m files)

MATrix LABotary

- Comprende un vasto set di funzioni predefinite e i toolbox possono essere ampliati.
- Ha una buona potenzialità grafica
- Esistono versioni sia di Octave che di Matlab per Unix/Linux, Windows, MAC e i file creati sono portabili da un sistema all'altro.

Avvio


MATLAB R2011b
Current Folder: /Users/gianna/Documents/MATLAB
Shortcuts How to Add What's New
Current Folder
« MATLAB
Name
mathworks_downloads
activate_network_license.pdf
CVM.m
dati.mat
fun.m
givfact.m
iteralrtri.m
license.dat
my_qr.m
pi_monte_carlo.m
polymalM.m
polyvalM.m
Details
Select a file to view details
Command Window
Toolbox Path Cache read in 0.03 seconds.
MATLAB Path initialized in 0.06 seconds.
>>
Workspace
Name Min Max
Command History
y=y+i;
end
y
y-(n*(n+1)/2)
n=10^9;
x=
x=0;
for i=1:n
x=x+i;
end
6-0.01
ans*20
10^-5/ans
10^-5/(20*(6-10^-2))
2+2*10^-2/(6-10^-2)
10^-4/(6-10^-2)
10^-4/(20*(6-10^-2))
6/(6-10^-2)
%-- 3/3/15 10:21 AM --%
Start
Ready
Workspace
Contenuto folder corrente
History

Potenzialita' di Matlab

- Come calcolatrice (doppia precisione)
- Calcolo Matriciale,
- Soluzione di equazioni,
- Derivate di funzioni (calcolo simbolico)
- Grafici 2D
- Grafici 3D: plot di funzioni, superfici, istogrammi, diagrammi, etc
- Programmazione

Introduzione

- La linea di comando è indicata da un prompt >> che indica che il sistema è in attesa di comandi da interpretare.
- I comandi previsti da Matlab sono accompagnati da una guida esaustiva (*help nome_del_comando*)
- Dalla linea di comando si possono dichiarare variabili, espressioni e chiamare funzioni sia predefinite che quelle definite dall'utente.
- Le funzioni non sono altro che file di testo e sono eseguite semplicemente digitando il nome con la giusta sequenza di parametri.

Introduzione

Il tipo di dato fondamentale in MATLAB è la matrice: uno scalare è infatti una semplice matrice 1×1 mentre un vettore è una matrice $1 \times n$ o $n \times 1$.
Le matrici possono essere inserite in diversi modi:

1. da tastiera;
2. caricate da file esterni;
3. generate da funzioni interne;
4. generate da m-file creati dall'utente

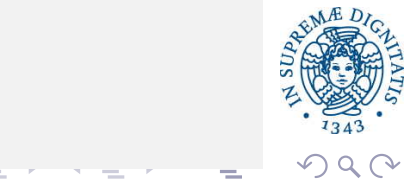
Primi calcoli in virgola mobile

Matlab utilizza la doppia precisione (8 byte per ogni numero).

```
>> realmin
ans = 2.2251e-308
>> realmax
ans = 1.7977e+308
>> eps
ans = 2.2204e-16
```

Matlab come una calcolatrice:

```
>> 1+1
ans = 2
>> 10^10
ans = 1.0000e+10
>> 1e10
ans = 1.0000e+10
```



Primi calcoli in virgola mobile

```
>> (1e10)^2
ans = 1.0000e+20
```

Se c'è un punto e virgola alla fine della linea, Matlab esegue il calcolo ma non scrive il risultato

```
>> 1+1;
>>
```

Perdita di precisione da alcuni calcoli:

```
>> a=1e10
a = 1.0000e+10
>> b=1e4
b = 10000
>> c=(a+b)^2
c = 1.0000e+20
```

Primi calcoli in virgola mobile

```
>> format long % scrive piu' cifre
>> c
c = 1.00000200000100e+20
>> c - a^2 - 2*a*b - b^2
ans = 7936
```

Principalmente da sottrazioni tra due numeri grossi e molto vicini, (*errori di cancellazione*), ma anche da moltiplicazioni:

```
>> a=98;
>> 1 - a*(1/a)
ans = 1.1102e-16
>> a=97;
>> 1 - a*(1/a)
ans = 0
```

Vettori e Matrici

Matlab è pensato per lavorare con vettori e matrici; pertanto, ha una sintassi specifica e parecchi comandi dedicati, che rendono molto più semplice lavorare con i vettori rispetto a un linguaggio generico come il C.

Creare vettori e matrici

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

   1 2 3
   4 5 6
```

```
>> zeros(3,2)
ans =

   0 0
   0 0
   0 0
```

```
>> ones(3,2)
ans =
```

Il range operator :

Con la sintassi `a:t:b` creiamo un vettore (riga) che contiene gli elementi `a`, `a+t`, `a+2t`...fino a `b` (o fino all'ultimo che sia minore o uguale a `b`). Se `t=1`, può essere omesso.

```
>> 1:0.5:4
ans =
   1.0000 1.5000 2.0000 2.5000 3.0000 3.5000 4.0000
>> 1:10
ans =
   1 2 3 4 5 6 7 8 9 10
>> 1:2:10
ans =
   1 3 5 7 9
```

Accedere agli elementi

```
>> A=ones(2,3)
A =
   1  1  1
   1  1  1
>> A(1,2)=2
A =
   1  2  1
   1  1  1
>> A(1,2)
ans = 2
>> A(5,10)
error: invalid row index = 5
error: invalid column index = 10
```

Accedere agli elementi

```
>> A(5,10)=7
A =
   1  2  1  0  0  0  0  0  0  0
   1  1  1  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  0
   0  0  0  0  0  0  0  0  0  7
```

Se cerco di *leggere* un elemento che non esiste (perché la matrice è troppo piccola), ottengo un errore. Se cerco di *scrivere* un elemento che non esiste, la matrice viene automaticamente ingrandita.

Operazioni su vettori

```
>> a=1:3
a =
  1 2 3
>> b=4:6
b =
  4 5 6
>> a+b
ans =
   5 7 9
>> sin(a)
ans =
   0.84147 0.90930 0.14112
>> 2*a+1
ans =
   3 5 7
```

Operazioni su vettori

```
>> a.*b %operazioni elemento per elemento
ans =
    4 10 18

>> c=a' %matrice trasposta
c =
   1
   2
   3
>> C=a'*b %prodotto matrice-matrice
C =
    4 5 6
    8 10 12
   12 15 18
```

Operazioni su vettori

```
>> length(a) %lunghezza di un vettore
ans = 3
>> size(C) %dimensioni di una matrice - (righe, colonne)
ans =
   3 3
```

Operazioni su vettori

Utilizzando l'operatore `:`, in Matlab è possibile selezionare un'intera sottomatrice di una matrice:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
   1 2 3
   4 5 6
   7 8 9
>> A(1:2,2:3)
ans =
   2 3
   5 6
>> A(2:end,1:end-2)
ans =
   4
   7
```

Operazioni su vettori

```
>> A(1:end,1:end)
ans =
   1 2 3
   4 5 6
   7 8 9
>> A(1,:)
ans =
   1 2 3
```

La sintassi `a:b` seleziona tutte le righe/colonne comprese tra `a` e `b` (estremi inclusi). Il valore `end` viene sostituito con il massimo indice disponibile. Il solo `:` è un'abbreviazione per `1:end`.

Operazioni su matrici

Possiamo anche assegnare un valore a una sottomatrice selezionata in questo modo:

```
>> A(1:2,1:2)=eye(2)
A =
  1  0  3
  0  1  6
  7  8  9
```

Ovviamente le dimensioni devono essere compatibili: non posso selezionare una sottomatrice 2×2 e assegnarle il valore `eye(3)`!
Il seguente comando ritorna la *matrice minore* di (i, j) in A, cioè la matrice che si ottiene eliminando la i-esima riga e la j-esima colonna di A.

Operazioni su matrici

```
>> i=2; j=3;
>> B=A([1:i-1, i+1:end], [1:j-1, j+1:end])
>> B=
   1  0
   7  6
```

Abbiamo giè visto che se `X,Y,Z,W` sono numeri, la riga di codice `B=[X Y; Z W]` crea la matrice

$$\begin{bmatrix} X & Y \\ Z & W \end{bmatrix};$$

ora vediamo che la stessa sintassi funziona anche se `X,Y,Z,W` sono matrici di dimensioni "compatibili" e crea la matrice formata accostando i quattro blocchi.

Esercizio: Creare una matrice in cui la prima riga sia composta dai numeri da 1 a 10, la seconda riga composta dai numeri da 11 a 20 e la e la terza dai numeri da 21 a 30. Modificare la seconda riga in modo da annullarne gli elementi.

```
>> A=[1:10; 11:20; 21:30]
>> A =
     1  2  3  4  5  6  7  8  9 10
    11 12 13 14 15 16 17 18 19 20
    21 22 23 24 25 26 27 28 29 30
>> A(2, :)=0
A =
     1  2  3  4  5  6  7  8  9 10
     0  0  0  0  0  0  0  0  0  0
    21 22 23 24 25 26 27 28 29 30
```

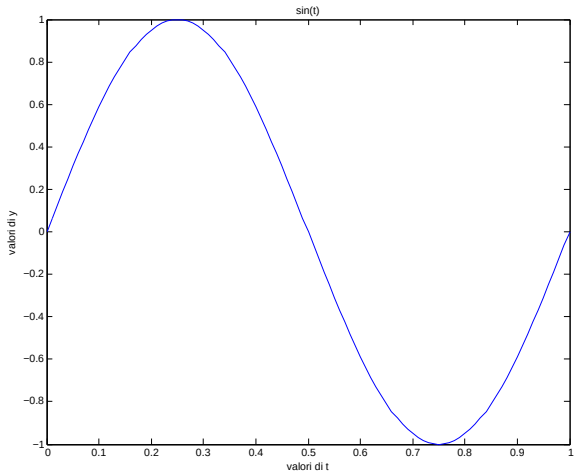
Esempi: Operazioni tra matrici

- In Matlab è possibile memorizzare e richiamare dati utilizzando i comandi `load` e `save`
- La sintassi del comando `save` è la seguente:
  `save nomeFile nomevar1 nomevar2 nomevar3`
- Il comando `save` salva le variabile nomevar1, nomevar2, nomevar3, in un file nomeFile.MAT in formato proprietario. È possibile richiamarle in memoria utilizzando il comando `load`.
  `load nomeFile`
  in questo caso vengono ripristinate in memoria le variabili nomevar1, nomevar2, nomevar3,
- È possibile anche salvare i dati in formato ASCII usando l'opzione `-ascii`.
  `save nomeFile.Estensione nomevar1 nomevar2 nomevar3 -ascii`

Grafica

Ci sono molte funzioni per fare grafici.

```
>> t=0:0.01:1;
>> y= cos(2*pi*t);
>> plot(t, y);
>> xlabel('valori di t');
>> ylabel('valori di y');
>> title('sin(t)');
```



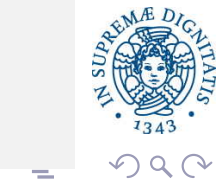
Grafica

Si possono dare specifiche di linea, su colore, simbolo e tipo di linea.
Si possono disegnare più grafici sugli stessi assi

```
>> x=linspace(0, pi);
>> y1=cos(x); y2=sin(x);
>> plot(x, y1, x, y2);
```

I colori utilizzati sono distinti!
Possiamo utilizzare il comando `hold` per "congelare" il grafico

```
>> x=linspace(0, pi);
>> y1=cos(x);
>> y2=sin(x);
>> plot(x, y1, 'linewidth', 2);
>> hold on
>> plot(x, y2, 'linewidth', 2);
>> hold off
```



Esercizio Disegnare un cerchio di raggio 1 e centro $(0, 0)$

```
>> theta=[0:0.01:2*pi];
>> plot(1*cos(theta), 1*sin(theta));
```

Grafica

Si possono fare grafici semilogaritmici

```
>> semilogx(x, y);
>> semilogy(x, y);
>> loglog(x, y);
```

Con il comandi `subplot(n, m, k)`, si può partizionare la finestra grafica in $n \times m$ parti e fare un plot nel $k - esimo$ riquadro.
Si possono disegnare grafici a barre (con il comando `bar`), a torta (con il comando `pie`)

Grafici di funzioni

Per visualizzare le funzioni matematiche ci sono diverse funzioni

- Come già visto possiamo usare il comando `plot(x, y)` con `x` un campionamento dell'intervallo sul quale vogliamo disegnare la funzione e con y il valore che la funzione assume nei punti `x`.
- Possiamo usare `ezplot(f)` o `fplot(f, [a, b])`;

Come dobbiamo scrivere la funzione da dare come argomento a questi due comandi?

```
>> ezplot(cos)
Error using cos
Not enough input arguments.
```

facendo invece

```
>> f=@cos;
>> ezplot(f)
```

non abbiamo problemi.

Grafici di funzioni

Non stiamo passando il nome della funzione ma una "handle" alla funzione. Cio'e le funzioni devono essere "anonimizzate" prima di passarle a `ezplot(f)` o `fplot(f, [a, b])`

- `ezplot(f)` disegna f nell'intervallo $[-2 * \pi, 2 * \pi]$, a meno che il dominio della funzione non escluda qualche punto.
- `ezplot(f, [a, b])` disegna f nell'intervallo $[a, b]$
- `fplot(f, [a, b])` disegna f nell'intervallo $[a, b]$ che va sempre assegnato

Attenzione! Le funzioni devono essere vettorizzate. (esempi)

Grafica 3-D

Per tracciare una linea $(x, y, z = f(t))$ in un diagramma tridimensionale usiamo la funzione`plot3`

```
>> t = [0:pi/50:10*pi];
>> plot3(exp(-0.05*t).*sin(t), exp(-0.05*t).*cos(t),t),
>> xlabel('x'),ylabel('y'),zlabel('z'),grid
```

che disegna la curva
$x = \exp(-0.05t)\sin(t), y = \exp(-0.05t)\cos(t), z = t$

Grafica 3-D

per disegnare superfici dobbiamo usare il comando
`[X, y]=meshgrid(a:h:b)` che genera una griglia di punti.

```
>> [X, Y]=meshgrid(-2:0.1:2);
>> Z=X.*exp(-((X-Y.^2).^2+Y.^2));
>> mesh(X, Y, Z)
```

Al posto di `mesh` potevamo usare `surf` per ottenere una superficie con i colori sfumati.
Si possono fare anche cose più complicate... es MRI.m

Programmazione in Matlab

- In MATLAB si possono scrivere m-file, ovvero file di testo conteneti sequenze di comandi e strutture di controllo che vengono interpretate.
- I file possono essere generati o mediante un qualsiasi editore di testo o quello interno e salvati con estensione .m
- Devono essere contenuti nella directory corrente o contenuta nel path di MATLAB
- Gli m-file sono di due tipi

  1) Script
  2) Function

Programmazione in Matlab

- Le variabili non vanno dichiarate
- I tipi di dato non vanno specificati
- Non occorre allocare spazio in memoria per le variabiloi
- Non sono creati file eseguibili

Gli Script

Gli script sono sequenze di comandi da eseguire nell'ambiente di chiamata. Al suo interno possono essere chiamate delle funzioni di MATLAB o definite dall'utente
Se scriviamo uno script `esempio.m` poi lo si esegue scrivendo `esempio` nella finestra dei comandi, cioè il nome del file senza estensione.
Importante. Gli script usano le variabili presenti al momento dell'esecuzione dei comandi e modificano l'ambiente di lavoro.

Le function

A differenza degli script le function hanno parametri di input e di output

`function [variabili_di_output]=nomefunction(variabili_di_in`

dove `nomefunction` è il nome che diamo alla funzione e il nome dell'm-file in cui è memorizzata.