

Laboratorio di Analisi Numerica

Introduzione a Matlab/Octave

Ángeles Martínez Calomardo
amartinez@units.it

Laurea Triennale in Intelligenza Artificiale e Data Analytics
A.A. 2021–2022

Matlab

- Prodotto commerciale che fornisce sofisticati strumenti di calcolo.
- È distribuito da The MathWorks (si veda il sito www.mathworks.com).
- La sua principale caratteristica è la manipolazione di matrici, come viene sottolineato dall'acronimo MATLAB che deriva da MATrix LABoratory, vale a dire “laboratorio matriciale”.
- Calcolatrice scientifica evoluta.
- Linguaggio di programmazione ad alto livello.

Per avviare Matlab in ambiente Unix basta digitare il comando `matlab` seguito dal tasto di invio.

Octave

- Anche Octave è un ambiente integrato per il calcolo scientifico e la visualizzazione grafica come Matlab.
- È distribuito gratuitamente dalla GNU (si veda il sito www.octave.org).
- Matlab e Octave presentano delle differenze ma sono sufficientemente compatibili da permettere alla maggior parte di programmi Matlab di essere eseguiti senza modifiche in ambiente Octave e viceversa.

Per avviare Octave in ambiente Unix basta digitare il comando `octave` seguito dal tasto di invio.

**LE INFORMAZIONI CONTENUTE IN QUESTI LUCIDI SONO VALIDE
PER ENTRAMBI I PROGRAMMI**

Interfaccia grafica di Matlab

L'interfaccia grafica di Matlab è costituita da 4 ambienti:

- **Workspace.** Una finestra che mostra il contenuto del workspace (variabili memorizzate e loro valore).
- **Current directory.** Una finestra sulla cartella in cui si sta lavorando, che mostra i files presenti nella cartella stessa.
- **Command history.** Contiene una lista di tutti i comandi digitati.
- **Command window.** Finestra nella quale vengono inseriti i comandi.

L'interfaccia grafica di Octave è costituita degli stessi 4 ambienti con nomi diversi:

- **Variables' list.** Variabili memorizzate e loro valore.
- **Navigator.** Una finestra sulla cartella in cui si sta lavorando, che mostra i files presenti nella cartella stessa.
- **Command List.** Contiene una lista di tutti i comandi digitati.
- **Octave Terminal.** Finestra nella quale vengono inseriti i comandi.

Command window / Octave Terminal

- La **Command window/Octave Terminal** permette di interagire con l'ambiente di calcolo di Matlab/Octave che si presenta come una linea di comando detta *prompt*.
- Permette di eseguire programmi (script e function) presenti in Matlab/Octave, ma anche programmi costruiti dall'utente usando il linguaggio Matlab/Octave e salvati su un file di testo con estensione .m (m-file).
- Per eseguire uno script costruito dall'utente occorre scrivere dopo il prompt il nome del file **senza l'estensione .m**.
- Un programma Matlab/Octave non deve essere compilato: premuto il tasto enter le istruzioni vengono interpretate.
- Per essere eseguiti i programmi devono essere presenti nella cartella di lavoro (current directory).
- Per capire qual è la cartella attuale esiste il comando **pwd**.
- Il comando **what** lista i file .m presenti nella cartella di lavoro, mentre il comando **ls** lista tutti i files presenti nella stessa.

Variabili e assegnazione

- In Matlab/Octave non occorre dichiarare le variabili: l'assegnazione coincide con la dichiarazione.

```
a = 2/3
a =    0.66667
b = 3/2
b =    1.5000
a*b
ans =    1
```

- Matlab/Octave crea le variabili a e b nel momento in cui viene loro assegnato un valore. Se il risultato di un'espressione non viene assegnato a nessuna variabile definita dall'utente, viene assegnato alla variabile di default `ans`.
- I nomi delle variabili possono essere lunghi al massimo 19 caratteri e devono iniziare con un carattere alfabetico (distingue tra maiuscole e minuscole).

Esercizio

Assegnare alla variabile A il valore 1, e scrivere a dopo il prompt. Si osservi che A ed a sono due variabili distinte.

Variabili e assegnazione

- Il comando `who` permette di sapere quali sono le variabili dell'utente attualmente in memoria.
- Il comando `whos` ne mostra anche la dimensione e l'occupazione di memoria (numero di bytes).

`whos`

Variables in the current scope:

Attr	Name	Size	Bytes	Class
	A	1x1	8	double
	a	1x1	8	double
	<code>ans</code>	1x1	8	double
	b	1x1	8	double

Total is 4 elements using 32 bytes

- Le variabili possono essere cancellate utilizzando il comando `clear`.
- Ci sono variabili predefinite come l'unità immaginaria i o il numero π .

Operazioni aritmetiche e funzioni matematiche predefinite

+ addizione
− sottrazione
* prodotto
/ divisione
^ elevamento a potenza

Funzione	function MATLAB
sin	sin
cos	cos
tan	tan
arcsin	asin
arccos	acos
arctan	atan
exp	exp
ln	log
\log_2	log2
\log_{10}	log10
.	abs
$\sqrt{\cdot}$	sqrt

L'istruzione `format`

- Permette di modificare il formato di visualizzazione dei risultati ma **non modifica la precisione con cui i calcoli vengono eseguiti**.
- Tutti i calcoli vengono effettuati in Matlab/Octave utilizzando i numeri in virgola mobile in doppia precisione, secondo lo standard IEEE-754r.
- I principali formati di visualizzazione dei risultati si ottengono digitando `help format`.

Dato il numero $1/7$, alcuni formati comunemente usati sono

<code>format short</code>	produce	0.1429
<code>format short e</code>	produce	1.4286e-01
<code>format short g</code>	produce	0.14286
<code>format long</code>	produce	0.142857142857143
<code>format long e</code>	produce	1.428571428571428e-01
<code>format long g</code>	produce	0.142857142857143

- Gli stessi formati sono disponibili in Octave e forniscono risultati con lievi discrepanze.

Matrici e vettori

- Le variabili per Matlab/Octave hanno una struttura di tipo **matriciale**.
 - ▶ Gli scalari sono considerati matrici 1×1 .
 - ▶ I vettori riga sono matrici $1 \times n$.
 - ▶ I vettori colonna sono matrici $n \times 1$.
- Per definire una **matrice** se ne possono innanzitutto assegnare direttamente gli elementi riga a riga. Ad esempio digitando

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

si produce

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

- Notiamo che il punto e virgola separano righe diverse.

Matrici e vettori

- L'elemento in riga i e colonna j di A si accede con $A(i,j)$. Per la matrice A dell'esempio precedente

```
>> A(2,3)
ans = 6
```

Esercizio

Costruire una matrice 2×3 con i primi sei numeri interi come coefficienti. Azzerare gli elementi $A(1,1)$ e $A(2,2)$.

- Soluzione.

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

```
>> A(1,1) = 0;
>> A(2,2) = 0;
>> A
```

```
A =
     0     2     3
     4     0     6
```

Estrarre parti di una matrice

Si può fare tramite l'uso del carattere due punti :

Ad esempio, per individuare la seconda riga di A , basta scrivere

```
>> A(2,:) 
```

Per individuare la terza colonna di A si scrive invece

```
>> A(:,3) 
```

- Per estrarre una intera sottomatrice da una matrice assegnata, basta specificare un insieme di righe e colonne. Esempio:

$A =$

```
1    1    1
-2    3    1
4   -6    7
```

```
>> M=A(2:3,:) 
```

$M =$

```
-2    3    1
4   -6    7
```

- Per scrivere una matrice vuota A , invece, si usa l'istruzione $A = []$.

Matrici

Matlab allarga una matrice quanto basta per sistemare un elemento dato.

```
>> A=[1 1 1; -2 3 1; 4 -6 7]  
A =
```

```
1 1 1  
-2 3 1  
4 -6 7
```

```
>> A(5,5)=2  
A =
```

```
1 1 1 0 0  
-2 3 1 0 0  
4 -6 7 0 0  
0 0 0 0 0  
0 0 0 0 2
```

```
>> A=A(1:3,1:3)  
A =
```

```
1 1 1  
-2 3 1  
4 -6 7
```

```
>> A(3,6)=9  
A =
```

```
1 1 1 0 0 0  
-2 3 1 0 0 0  
4 -6 7 0 0 9
```

Comandi predefiniti che operano su matrici

- Comandi che generano matrici:

`rand(m,n)` matrice $m \times n$ con coefficienti random.

`eye(n)` matrice identità di ordine n .

`ones(n)` matrice di ordine n con coefficienti tutti uguali ad 1.

`zeros(n)` matrice di ordine n con coefficienti tutti uguali a 0.

- Altri comandi importanti che operano con matrici:

`inv(A)` calcola l'inversa della matrice A ;

`[n,m] = size(A)` restituisce il numero di righe e di colonne di A ;

`det(A)` calcola il determinante di A .

`eig(A)` calcola gli autovalori di A .

Esercizio

Creare una matrice quadrata A di ordine 4 con tutti gli elementi uguali a 1 e calcolarne il determinante.

Che cosa succede se proviamo a calcolare l'inversa di A ?

Comandi predefiniti che operano su matrici

Soluzione dell'esercizio

```
>> A = ones(4)
A =
```

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

```
>> det(A)
```

```
ans = 0
```

```
>> inv(A)
```

```
warning: inverse: matrix singular to machine precision , rcond = 0
```

```
ans =
```

```
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
Inf Inf Inf Inf
```

Operazioni tra matrici

- Essendo A, B, C matrici con coefficienti reali ed s uno scalare, si definiscono le operazioni:

$C = s * A$ prodotto di una matrice per uno scalare.

$C = A'$ trasposizione di una matrice.

$C = A + B$ somma di due matrici di dimensione $m \times n$.

$C = A - B$ sottrazione di due matrici $m \times n$.

$C = A * B$ prodotto di A (m righe e n colonne) per B (n righe e p colonne).

$C = A .* B$ $c_{ij} = a_{ij} \cdot b_{ij}$ (prodotto di due matrici componente a componente).

- Esempio:

```
>> A = [2  -1; 3  4; -2  7]
```

```
A =
```

```
    2    -1
    3     4
   -2     7
```

```
>> B = A'
```

```
B =
```

```
    2     3    -2
   -1     4     7
```


Operazioni tra matrici: esempi

Riportiamo un esempio di somma, prodotto, e prodotto componente a componente di due matrici.

```
>> A = [4 -1 0; -1 4 -1; 0 -1 4];  
>> B = [1 2 3; -4 -5 -6; 0 1 2];
```

Il carattere **;** usato alla fine di qualunque istruzione sopprime l'output a video.

```
>> C = A+B
```

```
C =
```

```
    5     1     3  
   -5    -1    -7  
    0     0     6
```

```
>> C = A*B
```

```
C =
```

```
     8     13     18  
   -17    -23    -29  
     4      9     14
```

```
>> C = A.*B
```

```
C =
```

```
     4     -2      0  
     4    -20      6  
     0     -1      8
```

Vettori

- Matlab/Octave tratta i vettori come casi particolari di matrici.
- Per memorizzare il vettore riga $x = [1, 2, 3, 4, 5]$ occorre digitare

```
x = [1 2 3 4 5];
```

- mentre

```
y = [-2; 4 ; 12];
```

produce il vettore colonna

```
y =  
    -2  
     4  
    12
```

- `zeros(1,n)` crea un vettore riga di dimensione n con tutti gli elementi nulli;
`zeros(n,1)` idem per vettori colonna.
- Per creare un vettore riga (colonna) con elementi uguali ad 1 usiamo
`ones(1,n)` (`ones(n,1)`).

Vettori

- La componente i -esima di un vettore si identifica con $x(i)$. Per esempio la terza componente del precedente vettore y sarà:

```
y(3)
ans =
    12
```

- un vettore colonna si trasforma nel corrispondente vettore riga mediante trasposizione:

```
x = [-2; 4; 12] '
x =
    -2     4    12
```

- Si può creare un vettore vuoto (cioè con zero componenti) con il comando $x = []$.
- La norma euclidea di un vettore x si definisce come $\|x\| = \sqrt{x^T x}$. In Matlab/Octave si implementa con il comando $\text{norm}(x)$.

Operazioni tra vettori

- Essendo z, u, v vettori (riga o colonna) ed s uno scalare, si definiscono le operazioni:

$z = s*u$ prodotto di un vettore per uno scalare.

$z = u+v$ somma di due vettori di dimensione n .

$z = u-v$ sottrazione di due vettori n .

$z = u.*v$ $z_i = u_i \cdot v_i$ (prodotto tra due vettori componente a componente).

$z = u./v$ $z_i = u_i / v_i$ (divisione tra due vettori componente a componente).

- Il prodotto scalare tra due vettori colonna x e y di dimensione n :

$s = x^T y = \sum_{i=1}^n x_i y_i$ si esegue in Matlab/Octave come

$s = x'*y$

Esercizio

Si definisca il vettore colonna u di componenti $(1,2)$ e il vettore colonna v di componenti $(3,4)$; si calcoli $u+v$, $u-v$, il prodotto scalare di u per v e il prodotto componente a componente.

Prodotto matrice per vettore

- Se A è una matrice $n \times m$ e u un vettore colonna $m \times 1$, allora $A * u$ è l'usuale prodotto matrice per vettore.

Esercizio

Sia $A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ -1 & 3 & 0 \end{pmatrix}$, $u = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$. Calcolare $w = A \cdot u$.

- Soluzione:

```
>> A = [1 3 5; 2 4 6; -1 3 0];  
>> u = [1 1 2]  
u =  
    1    1    2
```

```
>> w = A*u'  
w =  
    14  
    18  
     2
```

- Se avessimo scritto $A * u$, avremmo ottenuto un messaggio di errore!

Vettori con elementi equispaziati

Notazione ":"

In Matlab/Octave si possono creare vettori riga con elementi equispaziati con la notazione ":" la cui sintassi è

`vettore = [inizio:incremento:fine]`

dove inizio è il primo elemento del vettore, e incremento è un parametro opzionale che indica la spaziatura tra gli elementi (se omesso incremento=1).

Esempio:

```
>> u = [1:2:10]
u =
     1     3     5     7     9
```

```
>> v = [5:-1:1]
v =
     5     4     3     2     1
```

```
>> w = [0:0.2:1]
w =
 0.00000  0.20000  0.40000  0.60000  0.80000  1.00000
```

Crea gli elementi
 $\text{vettore}(i) = \text{inizio} + (i-1) \cdot \text{incremento}$
fino a quando
 $\text{vettore}(i) \leq \text{fine}$

Vettori con elementi equispaziati

Comando `linspace`

- Un'alternativa ai due punti è il comando:

`linspace`(*inizio*,*fine*,numero di punti)

- Il comando `linspace` genera un vettore riga con un numero prefissato di punti equispaziati compresi tra *inizio* (primo elemento) e *fine* (ultimo elemento del vettore).
- Se il numero di punti è omesso se ne creano 100.
- Esempi:

```
>> u = linspace(0,8,5)
```

```
u =
```

```
    0    2    4    6    8
```

```
>> v = linspace(-5,5,6)
```

```
v =
```

```
   -5   -3   -1    1    3    5
```

```
>> v = linspace(-5,5,5)
```

```
v =
```

```
 -5.00000  -2.50000   0.00000   2.50000   5.00000
```

Vettori con elementi equispaziati

Esercizio

Creare il vettore u di componenti $u_i = -3 + 0.5 \cdot i$, $i = 0, \dots, 6$, utilizzando sia la notazione `:` che il comando `linspace`.

- Soluzione.

```
>> u = [-3:0.5:0]  
u =
```

```
   -3.00000   -2.50000   -2.00000   -1.50000   -1.00000   -0.50000  
         0.00000
```

```
>> u = linspace(-3,0,7)  
u =
```

```
   -3.00000   -2.50000   -2.00000   -1.50000   -1.00000   -0.50000  
         0.00000
```

- Un'istruzione Matlab/Octave di uso molto comune è `length` che calcola il numero di elementi di un vettore.

```
>> length(u)  
ans = 7
```


Uso vettoriale delle funzioni elementari

- Le funzioni matematiche predefinite in Matlab/Octave sono vettoriali.

Esempi:

```
>> u = [1:1:6]
u =
     1     2     3     4     5     6
>> log(u)
ans =
     0.00000     0.69315     1.09861     1.38629     1.60944     1.79176
>> exp(u)
ans =
     2.7183     7.3891    20.0855    54.5982   148.4132   403.4288
```

- Per le operazioni prodotto e divisione bisogna usare il punto per lavorare vettorialmente.

```
>> u = -1:0.5:1
u =
    -1.0000    -0.5000     0.0000     0.5000     1.0000
>> u.^2
ans =
     1.00000     0.25000     0.00000     0.25000     1.00000
>> 1./u
ans =
    -1     -2   Inf     2     1
```

Uso vettoriale delle funzioni elementari

Esercizio

Calcolare il seno e il coseno dei seguenti angoli: $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi\}$.

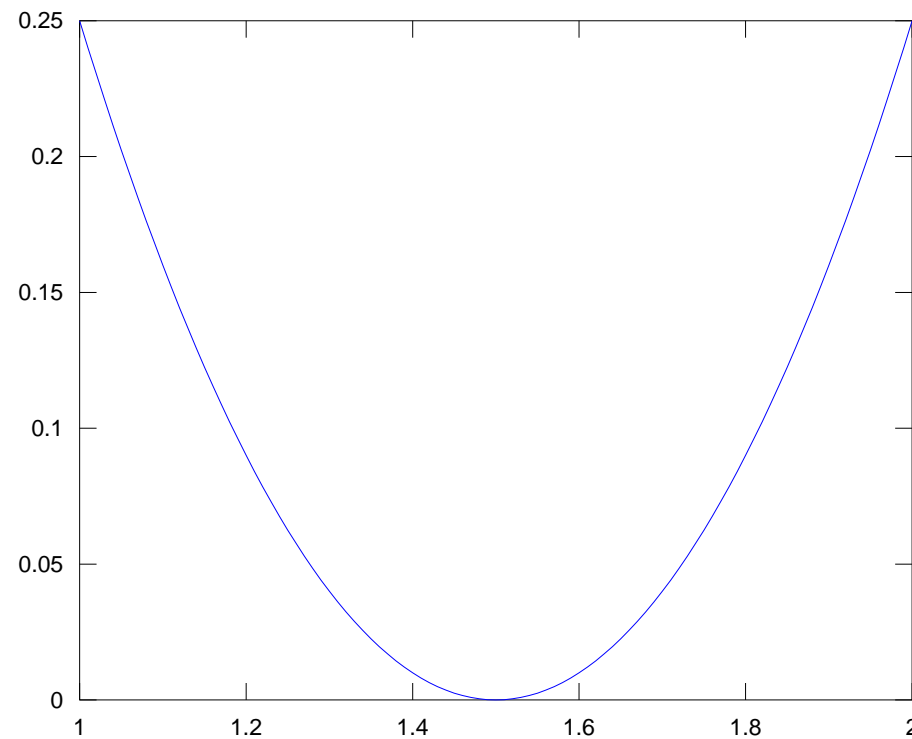
Grafica in Matlab/Octave: comando `plot`

- Il comando `plot(x,y)` traccia il grafico di una serie di dati contenuti in due vettori x (il vettore delle ascisse) e y (il vettore delle ordinate) di lunghezza n .
- I due vettori x e y devono avere **la stessa lunghezza**: il grafico viene disegnato unendo tali punti con dei segmenti.
- Si possono utilizzare tanti comandi opzionali con cui modificare le caratteristiche del grafico:
 - ▶ aggiungere un titolo (`title`),
 - ▶ inserire etichette sugli assi (`xlabel`, `ylabel`)
 - ▶ selezionare tipo di linea, colore e spessore, ecc
- L' `help plot` fornisce tutte le indicazioni a tal proposito.

Grafica in Matlab/Octave: comando `plot`

Esempio 1

```
x = linspace(1,2,101);  
y = (x-1.5).^2;  
plot(x,y)
```



Grafica in Matlab/Octave: comando `plot`

Esempio 2

```
x=linspace(0,2*pi,100);
```

```
plot(x, sin(x), 'r-');
```

```
hold on;
```

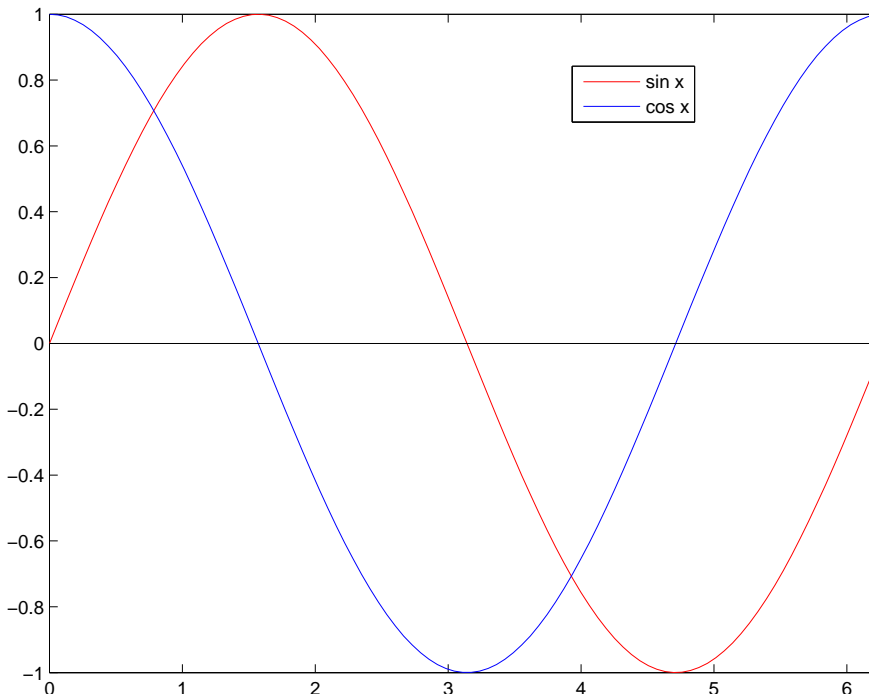
```
plot(x, cos(x), 'b-');
```

```
plot([0 2*pi],[0,0], 'k-');
```

```
hold off
```

```
legend('sin x', 'cos x');
```

```
axis([0 2*pi -1 1])
```

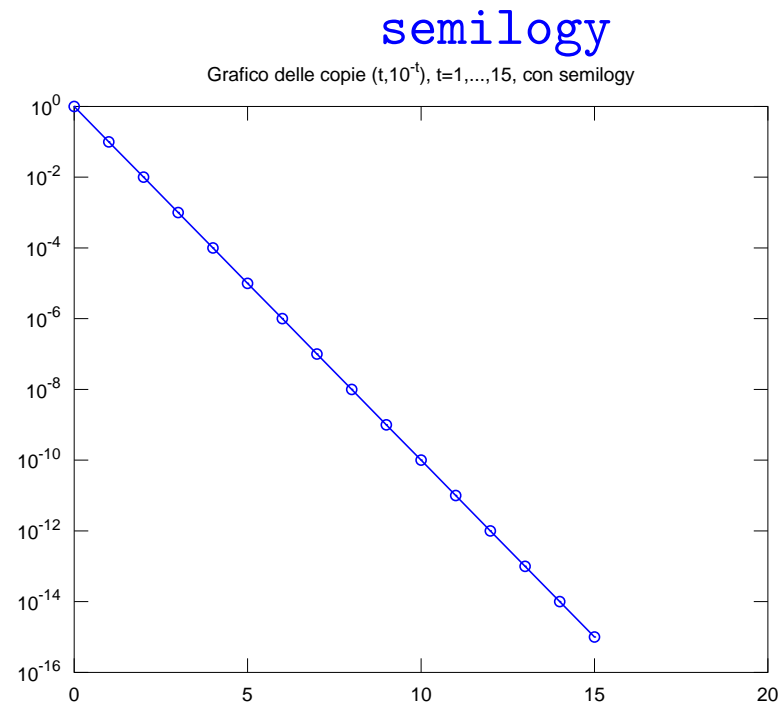
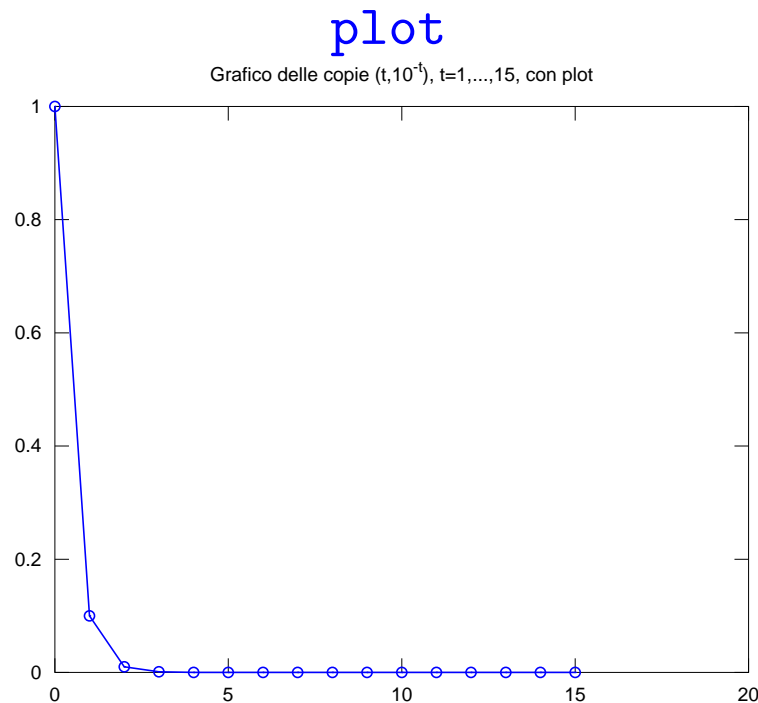


Grafici in scala semilogaritmica

- In molte aree scientifiche vengono usati grafici in scala semilogaritmica.
- Matlab/Octave fornisce tre comandi a tale proposito:
 - `semilogy`: equivalente a `plot` ma con l'asse delle ordinate in scala logaritmica
 - `semilogx`: idem con l'asse delle ascisse in scala logaritmica
 - `loglog`: entrambi gli assi in scala logaritmica
- Noi ricorreremo ai grafici in scala semilogaritmica, mediante il comando `semilogy`, nella realizzazione dei grafici che ci serviranno a studiare gli errori commessi dai metodi numerici.

Grafici in scala semilogaritmica: comando `semilogy`

- Supponiamo di voler plottare le coppie $(t, 10^{-t})$ per $t = 0, \dots, 15$.
- Il comando `plot` non mostra adeguatamente la differenza tra i valori poiché hanno ordini di grandezza troppo diversi.
- Questa differenza è invece palese nel grafico ottenuto con `semilogy`.



Matlab/Octave come linguaggio di programmazione

- Matlab/Octave può essere considerato un linguaggio di programmazione alla stregua di Fortran, di C, ecc.
- Non viene compilato ma interpretato (poco efficiente per calcoli intensivi).
- Un programma Matlab/Octave deve essere salvato in un m-file (file avente estensione .m).
- I programmi Matlab/Octave possono essere di due tipi:
 - `script`
 - `function`
- Strutture di programmazione basilari in Matlab/Octave:
 - Istruzione condizionale (`if else end`)
 - Cicli (`for` e `while`)

Il costrutto for

- La sintassi del costrutto for è la seguente:

```
for k = vettore  
    istruzioni  
end
```

- I comandi che si trovano tra **for** e **end** sono eseguiti per tutti i valori di **k** che sono nell'**vettore**.
- Esempio: calcolare la somma dei primi 10 numeri interi positivi usando un ciclo **for**.

```
somm=0;  
for n=1:10  
    somm=somm+n;  
end
```

- Possiamo usare più cicli for innestati.

Matlab/Octave come linguaggio di programmazione

Operatori logici e di relazione in Matlab/Octave

Operatori logici

& &	AND
	OR
~	NOT

Operatori di relazione

==	uguale
~=	diverso
<	minore
>	maggiore
<=	minore o uguale
>=	maggiore o uguale

- Il valore restituito dagli operatori può essere vero o falso e MATLAB utilizza il numero 1 per indicare il valore vero e 0 per il valore falso.
- Se, ad esempio, poniamo $x=5$; e $y=1$ e scriviamo la proposizione $x < y$, MATLAB risponde con `ans = 0` indicando che il confronto esprime una condizione falsa.

Il costrutto `while`

- Per il ciclo `while` la sintassi è data da:

```
while espressione logica
    istruzioni
end
```

- Questo ciclo è usato quando le istruzioni devono essere ripetute fino a quando rimane vera l'espressione logica (numero di volte indeterminato a priori).
- **Esempio:**

```
f=1; j=1;
while j < 10
    f=f*j;
    j=j+1;
end
```

- Il codice precedente calcola il fattoriale di 9!

Esercizio

Calcolare la somma dei primi n numeri interi positivi utilizzando un ciclo `while`.

Esempio di uso del costrutto `while`

Per trovare la soluzione dell'equazione $x = \cos x$, scriviamo le istruzioni:

```
x = 1;
dif = 100;
while dif > 1e-8
    xnew = cos(x);
    dif = abs(xnew-x);
    x = xnew;
end
xnew
```

che producono il risultato

```
xnew = 0.739085136646572
```

Il costrutto if-else-end

```
if espressione logica
    istruzioni
end
```

Esempio

```
if a > b
    maxval = a
end
```

```
if espressione logica
    istruzioni
else
    istruzioni
end
```

Esempio

```
if x > 0
    a = sqrt(x)
else
    a = 0
end
```

```
if espressione logica 1
    istruzioni
elseif espressione logica 2
    istruzioni
...
else
    istruzioni
end
```

Esempio di if-else-end e for

Il seguente codice costruisce una matrice

```
for m = 1:k
    for n = 1:k
        if m == n
            a(m,n) = 2;
        elseif abs(m-n) == 2
            a(m,n) = 1;
        else
            a(m,n) = 0;
        end
    end
end
```

Per $k = 5$ si avrebbe:

```
a =
    2  0  1  0  0
    0  2  0  1  0
    1  0  2  0  1
    0  1  0  2  0
    0  0  1  0  2
```

Programmi in Matlab/Octave: script

- È una semplice raccolta di istruzioni o comandi Matlab/Octave senza interfaccia di input/output.
- Ad esempio, l'insieme di istruzioni

```
a=1; b=-3; c = 2;  
delta = b^2-4*a*c;  
if delta < 0  
    disp('radici complesse')  
else  
    x1 = (-b-sqrt(delta))/(2*a)  
    x2 = (-b+sqrt(delta))/(2*a)  
end
```

una volta salvato in un m-file, di nome `eq2grado.m` diventa uno script.

- Per eseguirlo, è sufficiente scrivere dopo il prompt il nome senza estensione:

```
>> eq2grado  
x1 = 1  
x2 = 2
```

Programmi in Matlab/Octave: `function`

- Come lo script si definisce in un m-file, ad esempio `nomefun.m`
- La sua definizione inizia con la parola chiave `function`:

```
function [out1, ..., outn] = nomefun(in1, ..., inm)
```

- `out1, ..., outn` sono i parametri di output (opzionali);
- `in1, ..., inm` sono i parametri di input.
- Le variabili all'interno della `function` sono locali, il loro valore viene perduto al termine dell'esecuzione.
- Una funzione può esser invocata o da command window o da uno script.
- La `function` termina o all'ultima sua istruzione oppure quando si incontra per la prima volta il comando `return`.

Programmi in Matlab/Octave

Esempio

```
function [x1,x2,err] = radici(a,b,c)
err = 0;
delta = b^2-4*a*c;
if delta < 0
    err = 1; x1=0; x2=0;
    return
else
    x1 = (-b-sqrt(delta))/(2*a);
    x2 = (-b+sqrt(delta))/(2*a);
end
```

file `radici.m`

```
a=1; b=-3; c = 2;
[x1,x2,err] = radici(a,b,c)
```

file `scriptradici.m`

```
>> scriptradici
x1 = 1
x2 = 2
err = 0
```

```
>> delta
error: 'delta' undefined near line 99 column 1
```

Gestione dell'output su video

Comando `disp`

- Il comando `disp` serve per visualizzare una **stringa** di caratteri (testo racchiuso tra apici), o una **variabile** senza che ne venga visualizzato il nome.

●

```
>> x=1:2:19;  
>> disp(x)  
1      3      5      7      9     11     13     15     17     19
```

```
>> disp('Questa e'' una stringa');  
Questa e' una stringa
```

- Si possono visualizzare più dati in un unico comando `disp`:
 - ▶ Stringhe e variabili numeriche insieme

```
>> disp(['Convergenza in ', num2str(iter), ' iterazioni']);  
Convergenza in 23 iterazioni
```

- Il comando `num2str` converte un numero in una stringa.
 - ▶ Più variabili numeriche

```
>> disp([val, err, iter])  
2.1099e+00    1.0000e-10    2.3000e+01
```

- L'output del comando `disp` finisce sempre con un avanzamento di linea.

Gestione dell'output su video

Comando `fprintf`

- Per visualizzare un insieme di dati di output con un certo formato si usa il comando `fprintf` con i descrittori di formato:

Descrittore	Significato
<code>%f</code>	formato decimale (virgola fissa)
<code>%e</code>	notazione esponenziale
<code>%i</code> o <code>%d</code>	notazione per interi con segno
<code>%g</code>	la notazione piú compatta tra <code>%f</code> ed <code>%e</code>
<code>%s</code>	stringa di caratteri
<code>\n</code>	avanzamento di linea
<code>\t</code>	tabulazione
<code>\b</code>	backspace

- Tra `%` e il tipo di formattazione è possibile precisare il numero minimo di caratteri da stampare e il numero di cifre decimali dopo il punto.

Valore	<code>%6.3f</code>	<code>%6.0f</code>	<code>%6.3e</code>	<code>%6.3g</code>	<code>%6.3d</code>	<code>%d</code>
2	2.000	2	2.000e+000	2	002	2
0.02	0.020	2	2.000e-002	0.02	000	0
200	200.000	200	2.000e+002	200	200	200
<code>sqrt(2)</code>	1.414	1	1.414e+000	1.41	001	1

Come misurare la durata di un programma

- Per confrontare due programmi che risolvono lo stesso problema è utile misurare il tempo **totale** di CPU impiegato per eseguirli.
- In Matlab/Octave questo tempo si misura in secondi con il comando: `cputime`.

Esempio:

```
>> A = rand(5000); t = cputime; det(A); tfin=cputime; cpu=tfin-t  
cpu = 22.89
```

Esercizio

*Si crei una matrice quadrata random A di dimensione variabile da $n = 200$ a $n = 8000$ (con passo 200), e un vettore v lungo n . Misurare il tempo di esecuzione del prodotto $A * v$.*

Prodotto matrice vettore

Soluzione

Scriviamo lo script `matvet.m`

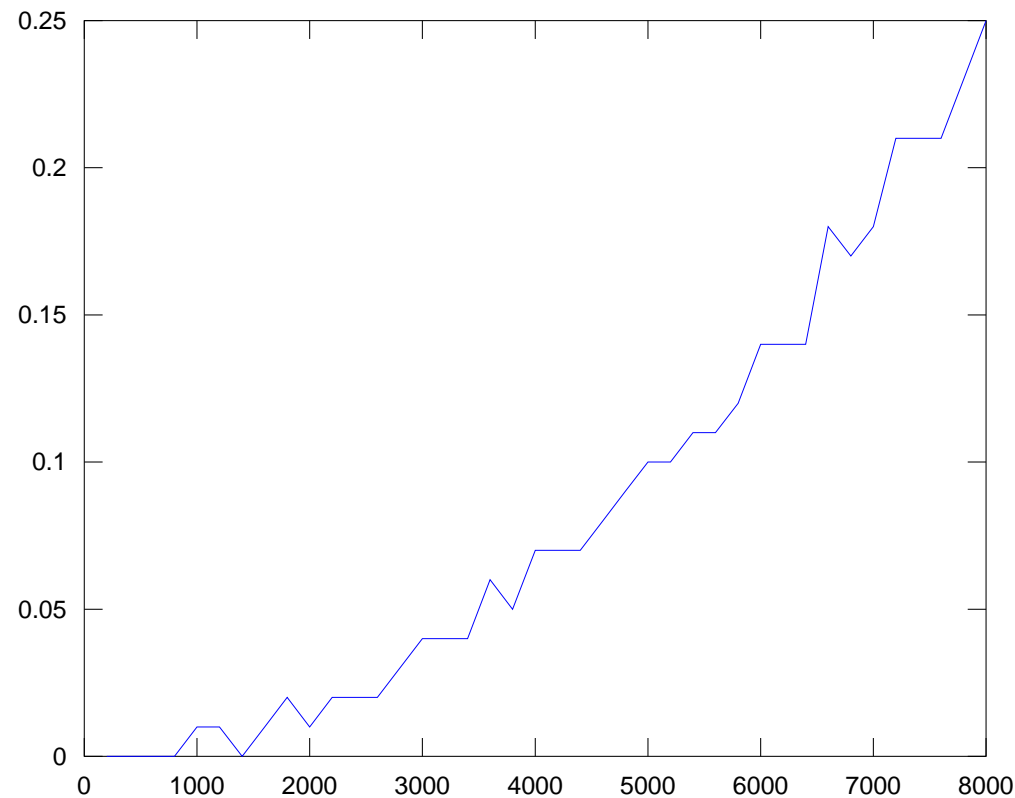
```
n=8000; step=200;
A=rand(n,n);
v=rand(n,1);
T=[ ];
sizea=[ ];
for k=200:step:n
    AA=A(1:k,1:k);
    vv=v(1:k);
    t=cputime;
    bb=AA*vv;
    tt=cputime-t;
    T=[T; tt];
    sizea=[sizea; k];
end
```

Grafico dei tempi

Usando il comando

```
>> plot(sizea ,T)
```

otteniamo il grafico del tempo di esecuzione dell'algoritmo in funzione della dimensione della matrice.



Come misurare la durata di un programma (2)

Comandi `tic` e `toc`

- Per misurare il tempo impiegato da Matlab per eseguire un determinato programma si può usare il comando `tic` seguito dal comando `toc`.
- Il tempo misurato è espresso in secondi.

Esempio:

```
>> A = rand(5000); t = tic; det(A); time=toc(t);  
time =  
    23.416
```