Introduzione a MATLAB (GNU Octave)

Laboratorio di Metodi Computazionali e Statistici (2022/23)

Fabrizio Parodi

Dipartimento di Fisica

October 18, 2023

Caratteristiche di MATLAB/GNU Octave

Forniscono entrambi un potente ambiente di calcolo e un linguaggio di programmazione di alto livello per i problemi del calcolo scientifico.

Caratteristiche:

- basati su C, C++ (inizialmente Fortran)
- disponibili su Windows, Unix, Mac

Per installare seguire il link `http://www.cedia.unige.it/matlab`, l'accesso avviene attraverso le proprie credenziali UNIGEPASS.

Vantaggi/svantaggi MATLAB/GNU Octave

Vantaggi

- User-friendly (ma non necessariamente programmer-friendly)
- Moltissime librerie e tools di grafica
- Ideale per la fase di prototipaggio

Svantaggi

- Software proprietario
- Inadatto per grandi molti di dati o per programmi dove le prestazioni in termini di velocità, gestione della memoria siano essenziali.
- Non è un vero e proprio linguaggio di programmazione ma uno strumento per accedere alle librerie (non OO, dipendente dalla IDE, regole sintattiche poco lineari e diversi dagli altri linguaggi)

Espressioni/assegnazioni

I comandi MATLAB/GNU Octave sono usualmente del tipo:

```
>> espressione
```

oppure

```
>> variabile=espressione
```

- espressione è un'espressione matematica o una funzione. In questo caso il risultato viene mostrato come `ans=`
- variabile è il nome di una variabile a cui viene assegnato (cioè in cui viene "memorizzato") il risultato dell'espressione. Se `espressione` è seguito da ; il risultato viene mostrato. Altrimenti no.

Help

- Il comando `help` fornisce una descrizione immediata di una funzione, un comando, un'operazione MATLAB/GNU Octave.
- Il comando `lookfor` testo identifica le funzioni nella cui descrizione compare l'argomento testo.
- Moltissima documentazione online.

MATLAB

MATLAB = **Mat**rix **lab**oratory

- in MATLAB/GNU Octave uno scalare è interpretato come una matrice 1×1

Operazioni aritmetiche tra scalari

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

Array

È un insieme di valori ordinati secondo uno o più indici.

- I vettori sono rappresentati da array ad un indice
- Le matrici sono rappresentate da array a 2 indici
- Gli scalari vengono considerati in MATLAB/GNU Octave come matrici 1×1

Vettori

- Dichiarazione

```
>> v = [ 1 2 3 4 ]
```

- Se i valori sono spaziati da spazi o `,` $\rightarrow$ vettore riga
- Se i valori sono spaziati `;` $\rightarrow$ vettore colonna
- Gli indici partono da 1 (al contrario di C, C++, Java, Python)
- Accesso a componente con parentesi tonde `v(2)`

Matrici

Creazione di una matrice n$\times$m (modi equivalenti)

- Gli spazio o le virgole separano gli elementi per colonna
- Il punto e virgola o l'esecuzione del tasto INVIO separano le righe

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

- Accesso a componente

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

- Il comando `size` ritorna le dimensioni della matrice, il comando `length` la dimensione di vettore o la dimensione maggiore per una matrice.

Accesso alle componenti

La sintassi di base per accedere alle componenti di una matrice (in senso ampio e quindi anche vettore) è, per ogni indice:

```
Inizio:Passo:Fine
```

- Inizio, Fine indicano l'elemento iniziale e l'elemento finale del vettore
- Passo è la distanza tra due successivi elementi. Se si omette è posto uguale ad 1.

Quindi, ad esempio, se A è una matrice:

- `A(i,:)` è la i-sima riga di `A`
- `A(:,j)` è la j-sima colonna di `A`
- `A(i:k,j:l)` è la sottomatrice di `A` che contiene le righe dalla i alla k, e le colonne dalla j alla l.

Funzioni per la costruzione di vettori/matrici

zeros	matrice con elementi tutti uguali a zero
ones	matrice con elementi tutti uguali a uno
eye	matrice identità
rand	matrice di numeri casuali
linspace	vettore di elementi equidistanti
logspace	vettore di elementi equidistanti in scala logaritmica

Ad esempio: la sintassi di base della funzione linspace è

```
Vettore=linspace(Inizio,Fine,Numero)
```

- Inizio, Fine indicano l'elemento iniziale e l'elemento finale del vettore dove
- Numero è il numero di elementi del vettore

Altre funzioni per vettori e matrici

diag(A,i)	vettore contenente la diagonale i-sima della matrice A
diag(b,i)	matrice che contiene nella diagonale i-sima il vettore b
tril(A,i)	matrice triangolare inferiore di A a partire dalla diagonale i
triu(A,i)	matrice triangolare superiore di A a partire dalla diagonale i
max(b), min(b)	massimo (minimo) valore degli elementi del vettore b
max(A), min(A)	vettore cont. il max (min) el. per ogni colonna della matrice A
sum(b), prod(b)	somma e prodotto degli elementi del vettore b
sum(A), prod(A)	vettore cont. somma e prodotto degli el. di A per ogni colonna
sort(b)	ordinamento crescente degli elementi del vettore b
det(A)	determinante
inv(A)	inversa
rank(A)	rango
eig(A)	autovettori ed autovalori

Operazioni tra matrici

'	creazione della trasposta
+	addizione
-	sottrazione
*	prodotto (righe per colonne)
^	elevamento a potenza (righe per colonne)
.*	prodotto (elemento per elemento)
./	divisione (elemento per elemento)
.^	elevamento a potenza (elemento per elemento)

Operazioni vettoriali

Calcolo vettoriale/matriciale nativo

Ad esempio:

$T_i = T_{i+1} + T_{i-1} - 2T_i$ che è la derivata seconda discretizzata, si scrive:

```
>> T(2:N-1) = T(3:N)+T(1:N-2)-2*T(2:N-1)
```

In questo caso specifico l'espressione non è corretta ai bordi.

Vettorializzazione di funzioni

- Molte funzioni predefinite in MATLAB/GNU Octave accettano come argomenti array a più indici.
- Per esempio la funzione *sin* può essere calcolata su un vettore di punti e restituire un vettore di valori.

```
>> x = linspace(0,pi,10);
>> y = sin(x);
```

Definizione e assegnazioni funzioni matematiche

Una funzione matematica del tipo

```
f(x) = espressione
```

può essere definita in MATLAB/GNU Octave:

- mediante anonymous function, utilizzando l'operatore di function handle :

  ```
  f=@(arg1, arg2,...,argn)[espressione]
  ```

- costruendo un'apposita function MATLAB/GNU Octave (lo vedremo in seguito)

La funzione verrà quindi valutata nel codice con l'espressione

```
f(arg1, arg2,...,argn)
```

Plot, fplot e superfici 3D

Plot 2D date ascisse e coordinate:

```
plot(Ascisse,Ordinate,Opzioni)
```

- Ascisse, Ordinate sono i vettori di dati (ascisse e ordinate dove dei punti)
- Opzioni è una stringa opzionale che definisce il tipo di colore, simbolo, linea usato nel grafico

Plot di funzioni:

```
fplot(funzione, [xmin xmax])
```

visualizza il grafico di una funzione (definita con), stabilendo automaticamente il numero di punti da utilizzare.

Superfici in 3D

```
surf(Ascisse,Ordinate,Matrice)
surf(Matrice)
```

- Disegna la superficie identificata da `Matrice` sulla griglia `Ascisse`x`Ordinate`
- Se `Ascisse` e `Ordinate` non sono specificate la griglia è creata sugli indici

Opzioni e comandi per plot

Colore		Simbolo		Linea	
`y`	giallo	`.`	punto	`-`	linea continua
`m`	rosa	`o`	circoletto	`:`	linea punteggiata
`c`	azzurro	`x`	per	`-.`	linea punto
`r`	rosso	`+`	più	`–`	linea tratteggiata
`g`	verde	`*`	asterisco		
`b`	blu	`s`	quadratino		
`w`	bianco	`d`	diamante		
`k`	nero	`v`	triangolo		

Funzione	Significato
`title`	inserisce un titolo nel grafico
`xlabel`	inserisce un nome per l'asse `x`
`ylabel`	inserisce un nome per l'asse `y`
`grid`	inserisce una griglia
`legend`	inserisce una legenda per ogni curva
`axis`	indica i valori min e max sugli assi

Gestione della grafica

- figure apre una nuova finestra grafica
- hold on consente di sovrapporre due o più grafici nella stessa figura
- hold off ritorna all'impostazione originale, in cui la finestra grafica viene ripristinata ad ogni nuovo grafico

M-files

La programmazione in MATLAB passa attraverso M-files (files con estensione .m)
File script

- Lavorano con le variabili del workspace
- Non richiedono variabili in input
- Non forniscono variabili in output
- Risultano utili quando si vuole automatizzare la ripetizione di una serie di operazioni che devono essere eseguite più volte

File function:

- Le variabili interne sono locali
- Accettano variabili in input
- Producono variabili in output
- Sono l'analogo dei programmi usualmente adottati in altri linguaggi di programmazione

script

```
% Calcolo della media e della varianza   Script file
% degli elementi di un vettore x
%
x=[1 2 3 4];
n=length(x);
media=sum(x)/n
varianza=sum(x.^2)/n-(media)^2
```

% segna l'inizio di un commento

`help` stampa il commento (che dovrebbe in genere contenere le istruzioni del programma)

function

La sintassi di una function richiede che la prima riga abbia la struttura:

```
function[Out1,Out2,...,Outn]=nomefunzione(In1,In2,...,Inn)
```

- In1,In2,...,Inn sono i parametri in ingresso Out1,Out2,...,Outn sono i parametri in uscita (se è uno solo, si possono omettere le parentesi quadre)
- La function deve essere salvata con il nome nomefunzione.m (cioè il file deve avere lo stesso nome della function)

function

```
function [med,var]=medvar(x)
%
% MEDVAR media e varianza di un vettore
% [med,var]=medvar(x)
% x = vettore in ingresso
% med = valor medio
% var = varianza
%
n=length(x);
med=sum(x)/n;
var=sum(x.^2)/n-(med)^2;
```

Operatori relazionali e logici

$<$	minore
$<=$	minore o uguale
$>$	maggiore
$>=$	maggiore o uguale
$==$	uguale
$\sim=$	diverso da
$\sim$	NOT
&	AND
—	OR

Struttura if/elseif/else

```
if EspressioneLogica1
  Blocco di istruzioni
elseif EspressioneLogica2
  Blocco di istruzioni
.
.
.
else
  Blocco di istruzioni
end
```

Loop

```
for Indice=Inizio: Incremento: Fine
  Blocco di istruzioni
end
```

- se `Fine` non viene specificato viene preso pari a 1
- `Inizio:Incremento:Fine` può essere sostituito da un `Vettore`

```
while Condizione
  Blocco di istruzioni
end
```

Come aggiungere componenti ad un vettore in MATLAB

Sfruttando concatenazione:

```
v = [];
for i=1:20
   v = [v i];
end
```

Più efficiente

```
v = [];
for i=1:20
   v(end+1) = i;
end
```

I/O

- Per acquisire x (scalare, vettore o matrice) si può richiederne il valore attraverso l'istruzione input:

```
x=input('Inserire il vettore:  ')
```

- L'output è gestito dall'istruzione `disp`

```
disp(a)
```

- Output formattato:

```
a = 1;
b = 2;
fileID = fopen('file.txt','w');
fprintf(fileID,'a e b valgono: %f e %f\n',a,b);
```

  - Gli specificatori di formato sono `%f` (float), `%d` (int) e `%s` (stringa); `\n` indica il fine linea
  - Se `fileId` viene omesso, stampa su schermo

```
fprintf('a e b valgono: %f e %f\n',a,b);
```

I/O da file

Salvare dati:

- Data la matrice x

```
x = [1.23 3.14 6.28; -5.1 7.00 0];
```

  il comando

```
save -ascii filename.dat x;
```

  produce il file filename.dat organizzato come segue

```
 1.2300000e+000 3.1400000e+000 6.2800000e+000
-5.1000000e+000 7.0000000e+000 0.0000000e+000
```

Caricare dati:

- il comando

```
load -ascii filename.dat
```

  carica nello spazio di lavoro tutte le variabili nel file. I dati saranno rappresentati da una matrice con il nome del file (senza estensione)
- Il file deve contenere dati separati da virgole o spazi

Esempio: propagazione di calore in una sbarra (impulso centrale)

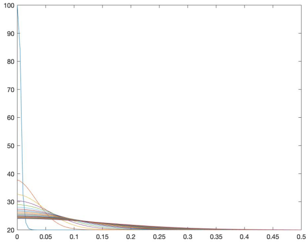
- impostazione del problema con metodo esplicito
- confronto con funzione teorica

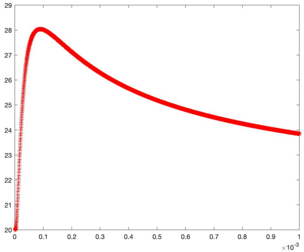
```
function HeatProp
  % Parametri fisici
  eta  = 0.2;  L    = 0.5; kappa = 10;
  % Delta x, Delta t
  N    = 101;  x   = linspace(0.0,L,N);
  dx   = x(2)-x(1);  dt  = (eta*dx^2)/kappa;
  % Impulso di calore iniziale e condizioni al contorno
  T0      = 20; DeltaT = 80;
  T       = ones(N,1)*T0;
  % ....
  % Evoluzione
  t    = 0; tend = 1e-4;
  while t<tend
    plot(x,T);
    % iterazione
    % ....
    drawnow
    t = t+dt
  end
```

Esercitazione

Esercitazione:

- Propagazione del calore lungo una sbarra termostatata ad un estremo con impulso di calore all'altro estremo (esperimento Lab3)
  - Metodo esplicito
  - Grafico dell'andamento, in funzione del tempo, della temperatura di un punto della sbarra.





Schema per la soluzione dell'equazione del calore

```
function HeatSchemeAsym
% Parametri fisici
  eta  = 0.2;  L    = 0.5; kappa = 10;
% Delta x, Delta t
  N    = 101;  x   = linspace(0.0,L,N);
  dx   = x(2)-x(1);  dt  = (eta*dx^2)/kappa;
% Impulso di calore iniziale
  T0     = 20; DeltaT = 80;
  T      = ones(N,1)*T0;
  T(1:2) = T(1:2) + DeltaT;
% Evoluzione
  t    = 0; tend = 1e-2;
  while t<tend
    plot(x,T);
    T(N) = T0;
%   T_left =
%   T(1)   =
    % Metodo esplicito
    % Salvataggio in vettore di T per un singolo punto
    drawnow %cercare parametri per rendere meno lenta la visualizzazione
    pause(0.0001)
    t = t+dt
  end
```

Propagazione del calore una sbarra con impulso al centro

Cranck-Nicholson

$$-T_{m-1,n+1} + \left(\frac{2}{\eta} + 2\right) T_{m,n+1} - T_{m+1,n+1} = T_{m-1,n} + \left(\frac{2}{\eta} - 2\right) T_{m,n} + T_{m+1,n}$$

$$\begin{bmatrix} \left(\frac{2}{\eta}+2\right) & -1 & 0 & \dots \\ -1 & \left(\frac{2}{\eta}+2\right) & -1 & \dots \\ 0 & -1 & \left(\frac{2}{\eta}+2\right) & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} T_{2,n+1} \\ T_{3,n+1} \\ T_{4,n+1} \\ \dots \\ \dots \\ T_{N-1,n+1} \end{bmatrix} = \begin{bmatrix} T_{1,n+1} + T_{1,n} + \left(\frac{2}{\eta} - 2\right) T_{2,n} + T_{3,n} \\ T_{2,n} + \left(\frac{2}{\eta} - 2\right) T_{3,n} + T_{4,n} \\ T_{3,n} + \left(\frac{2}{\eta} - 2\right) T_{4,n} + T_{5,n} \\ \dots \\ \dots \\ T_{N,n+1} + T_{N-2,n} + \left(\frac{2}{\eta} - 2\right) T_{N-1,n} + T_{N,n} \end{bmatrix}$$

Punto di partenza per la soluzione del'equazione del calore con CN

Matrice tri-diagonale con a, d, c e termine noto b

$$a = c = -1$$

$$d = \left(\frac{2}{\eta} + 2\right)$$

$$b_i = T_i + \left(\frac{2}{\eta} - 2\right) T_{i+1} + T_{i+2} \qquad b_1 = b_1 + T_1 \qquad b_m = b_m + T_n$$

Con $i \in [1, m = n - 2]$ (attenzione: l'array b ha due elementi in meno dell'array T, corrisponde alla sola parte centrale
L'indice spaziale si corre da 2 a $n - 1$.
Soluzione con:

$$h_i = \frac{c_i}{d_i + a_i h_{i-1}} \qquad p_i = \frac{b_i - a_i p_{i-1}}{d_i - a_i h_{i-1}}$$

e, infine,

$$x_i = p_i - h_i x_{i+1}$$

Schema per la soluzione del'equazione del calore

```
function HeatScheme
% Parametri fisici
  eta   = 0.2;  L    = 0.5; kappa = 10;
% Delta x, Delta t
  N     = 101;  x   = linspace(0.0,L,N);
  dx    = x(2)-x(1);  dt   = (eta*dx^2)/kappa;
% Impulso di calore iniziale
  T0             = 20;
  DeltaT         = 80;
  T              = ones(N,1)*T0;
  T((N-1)/2+1) = T0+DeltaT;
% Evoluzione
  t     = 0;
  tend = 1e-2
  while t<tend
    plot(x,T);
    T(N)  = T0;
    T(1)  = T0;
    % ...
    drawnow %cercare parametri per rendere meno lenta la visualizzazione
    pause(0.0001)
    t = t+dt
  end
```

Compito a casa: condensatore 2D

Condensatore 2D (soluzione eq. Laplace)

```
%
% Definisco griglia, posizione lastre e potenziale
%
for k=1:N  % N iterazioni
   Vold = V;
   % applico Jacobi, Gauss-Seidel o sovra-rilassamento
   %   iterando sulla matrice
end
figure(1)
surfc(V);
figure(2)
% calcolate gradiente (-> E) e disegnatelo
% funzione (quiver)
```

Compito a casa: corda vibrante

Punto di partenza per la soluzione della corda vibrante:

```
% parametri geometrici e fisici
% valori psi(x,t) a t=0
for i=1:1000
   plot(x,psi);
   psi(n) = 0;
   psi(1) = 0;
   % evoluzione
   drawnow;   % cercare parametri sul manuale
end
```