Introduzione ad Octave

Andrea Marin

Università Ca' Foscari di Venezia
Dipartimento di Informatica
Corso di Probabilità e Statistica

2009

Presentation outline

1. Introduzione al corso
2. Una breve introduzione a Matlab / Octave

Informazioni generali

- Corso di Probabilità e Statistica
- Parte teorica: prof. Andrea Torsello
- Esercitazioni di laboratorio: Andrea Marin
- Ore di lezione preveiste: 16 (8 incontri)
- Giorno ed ora: martedì dalla 15 alle 17
- Aula: Laboratorio 5
- Il calendario delle settimane in cui si tiene il laboratorio è disponibile sulla pagina web del corso: siete pregati di tenerlo d'occhio per eventuali variazioni

Informazioni su di me

- Nome: Andrea Marin
- Ufficio: Stanza 3 primo piano
- Email: `marin@dsi.unive.it`
- Pagina web: `http://www.dsi.unive.it/~marin`
  - Da qui si può seguire il link per il corso di probabilità

Modalità di esame

- Corso congiunto con la parte teorica
- È richiesto di superare entrambi i moduli
- Durante il laboratorio saranno assegnate 8 esercitazioni con difficoltà diverse
- Ne vanno consegnate almeno sei
- I tempi di consegna saranno fissati di volta in volta

Introduzione ad Octave

- Nasce come strumento per la didattica nell'Università del Texas
- Octave è il nome di un professore di Ingegneria delle reazioni chimiche
- Il software è distributo con licenza GNU
  - Per distribuzione Debian-like è a disposizione nei repository
  - Il software è gratuito
- La sintassi di Octave è MOLTO simile a quella di altri pacchetti commerciali

A cosa serve?

- Octave consente di effettuare operazioni numeriche
- Dispone di un linguaggio di programmazione che facilita la scrittura di elaborazioni complesse
- È in grado di effetture input/output da file
- È in grado si stampare grafici
- Tratta agevolmente matrici, vettori, sistemi lineari ecc. . .
- . . . e molto altro . . .

Primi passi

- In queste slides faremo riferimento alla versione per Linux
- Dalla shell, a partire dalla cartella di lavoro, l'ambiente viene avviato digitando:
  `$ octave`
- Se il software è correttamente installato compare un prompt
- Digitando:
  `> exit`
  si esce da Octave
- Per ottenere aiuto su un comando:
  `> help` *`nomecomando`*

I primi comandi: 1+1=?

- Digiatiamo nel prompt:

```
> 1+1
```

ottenendo:

```
ans = 2
```

- Dichiarare una variabile:

```
> a = 3
a = 3
> a = a + 5
a = 8
> a
a = 8
> a+2
ans = 10
```

Array, Matrici e Vettori

- Gli array sono gli elementi fondamentali per lavorare con OCTAVE
- Gli scalari sono visti come array 1x1
- Gli indici degli array cominciano da 1 (e non da 0 come in C)
- Gli array possono essere classificati in *vettori* (una dimensione) e *matrici* (più di una dimensione)
- Nel caso di array bi-dimensionali la dimensione è data dal numero di righe e di colonne (con le righe messe prima) (*convenzione...*)

Esempi

$\begin{bmatrix} 10 & 5 \\ 2 & 12 \\ 1 & 0 \end{bmatrix}$	Array 3×2
	`> A = [10 5; 2 12; 1 0];`
	`> A(3,1)`
	`ans = 1`
	`> A(2,:)`
	`ans = 2 12`

$[1\ 5\ 0\ 2]$	Vettore riga 1×4
	`> A = [1 5 0 2];`
	`> A(2)`
	`ans = 5`

$\begin{bmatrix} 3 \\ 7 \end{bmatrix}$	Vettore colonna 2×1
	`> A = [3; 7];`
	`> A(2)`
	`ans = 7`

Variabili

- Una variabile è una regione di memoria che contiene un array
- Ad una variabile è associato un identificatore
- Il valore di una variabile può essere cambiato in ogni momento
- Gli identificatori devono cominciare con una lettera
- OCTAVE è case sensitive

Inizializzare un array

- Si utilizzano parentesi quadre e punti e virgola
- Gli elementi sono elencati per riga
- In ciascuna riga gli elementi sono elencati da sinistra a destra
- Elementi nella stessa riga sono separati da uno spazio o una virgola
- Le righe sono separate da *new line* o da ;

Tipi di dato fondamentali

- Double: 64 bit floating point a doppia precisione

```
> A = 3.15E-10;
```

- Stringhe 2 byte per carattere

```
> ''pippo''
ans = pippo
```

Operatore :

- L'operatore : può essere utilizzato per inizializzare array molto grandi
- Sintassi: *first*:*increment*:*last*
- Esempio:

```
> A = 1:2:10
A = 1 3 5 7 9
> B = (0:  0.25 :  1)*pi
B = 0.00000 0.78540 1.57080 2.35619 3.14159
```

- Se l'incremento non è specificato, lo si assume uguale ad 1

Trasposizione di matrici

- L'operatore ' traspone una matrice
- Esempio:

```
> F = [1:4]';
> G = 1:4;
> H = [F G'];
```

- `H` è la matrice:

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \end{bmatrix}$$

Inizializzatori predefiniti

- zeros(n): inizializza una matrice quadrata $n \times n$ di tutti zeri
- zeros(n,m): inizializza una matrice rettangolare $n \times m$ di tutti zeri
- ones(n)
- ones(n,m)
- eye(n): inizializza una matrice Identit $n \times n$

Acquisire una variabile in input

- La funzione *input* legge un valore in fase di esecuzione
- Sintassi:

```
>in1 = input(''Inserisci il valore:  '')
Inserisci il valore:
```

- Il valore acquisito viene inserito nella variabile `in1`

Costanti e keywords

- pi: π
- i,j: $\sqrt{-1}$
- Inf: ∞
- NaN: not a number
- eps: la più piccola differenza tra due numeri
- ans: memorizza il risultato dell'ultima espressione

Realizzare degli script Matlab/Octave

- Con un editor di testo scrivere una sequenza di istruzioni
- Salvare il file con estensione `.m`
- Lanciare Octave dalla cartella in cui si è salvato lo script
- Digitare il nome del file (senza estensione)
- Esempio:

`esempio.m`

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

```
> esempio
> C
C = 14
    32
```

Sottoprogrammi in Matlab/Octave

- Ogni file `.m` può contenere una funzione
- Il nome della funzione deve essere uguale al nome del file che la contiene
- Esempio prima riga del file deve essere:

`nomefunzione.m`

```
function[a, b,
c]=nomefunzione(par1, par2,
par3, par4);
```

Stampare l'output su file

- C-like output
- `outfile = fopen(''prova.dat'',''w'')` apre un file
- `fwrite(outfile, format, espressione1, espressione2, ...)` scrive su file. La strina `format` segue la stessa sintassi di C
- `fclose(outfile)` chiude un file

Strutture di controllo: *if*

- Sintassi:

```
if (condition)
  then-body
else
  else-body
endif
```

- Esempio:

```
if (rem (x, 2) == 0)
  printf ("x is even");
else
  printf ("x is odd");
endif
```

Strutture di controllo: *while*

- Sintassi:

```
while (condition)
    body
endwhile
```

- Esempio:

```
fib = ones (1, 10);
i = 3;
while (i <= 10)
  fib (i) = fib (i-1) + fib (i-2);
  i++;
endwhile
```

Strutture di controllo: *for*

- Sintassi:

```
for var = expression
  body
endfor
```

- Esempio:

```
fib = ones (1, 10);
for i = 3:10
   fib (i) = fib (i-1) + fib (i-2);
endfor
```

Esercizi

1. Estrarre 1000 coppie di numeri compresi tra 0 e 1. Interpretare ogni coppia come un punto della regione di piano interna al quadrato $(0, 0)$, $(1, 1)$. Usare `gnuplot` per stampare i punti estratti.
2. Considerare i punti estratti nell'esercizio precedente. Estrarre un'altra coppia di punti e si assuma che questi siano i vertici opposti di un rettangolo i cui lati sono paralleli agli assi cartesiani. Contare quanti punti ricadono in questo rettangolo, e verificare che la proporzione di punti sulla superficie del rettangolo e i punti complessivi è *vicina* alla proporzione tra l'area del rettangolo e quella del quadrato di lato unitario.