Introduzione a MATLAB/OCTAVE

MATLAB =MAT(rix)-LAB(oratory) è un ambiente integrato per il calcolo scientifico utilizzabile sia in maniera interattiva che come linguaggio di programmazione.

In Matlab ogni quantità (variabile) viene trattata come matrice (tabella di valori). Un numero reale (scalare) è una matrice 1 × 1.

Sono predefinite numerose funzioni di uso generale (built-in func-tions), e raccolte di funzioni dedicate ad uno specifico argomento (toolboxes).

Per informazioni su Matlab: www.mathworks.com

Matlab è un software a pagamento. Octave è un software gratuito che ne riproduce buona parte delle funzioni fondamentali. Per info: www.octave.org.

Matlab in modalità interattiva

All’avvio di Matlab si accede ad una finestra di lavoro caratterizzata dal prompt

>>

Tutto quanto inserito dopo il prompt verrà eseguito dopo aver premuto il tasto “Enter”

Se Matlab riconosce il comando digitato produrrà un output in caso contrario segnalerà un errore. In ogni caso il sistema ripropone al termine il prompt in attesa di un nuovo comando.

Matlab si chiude con il comando quit

Alcuni comandi Matlab importanti da conoscere:

>> help

>> doc

permettono di ottenere informazioni dettagliate su qualsiasi comando. Il comando doc mostra anche quali pacchetti (toolboxes) siano installati nella versione in uso.

Ad esempio:

>> help sqrt >> doc sin

Per cercare il nome esatto di un comando:

>> lookfor cosine

cerca i comandi nella cui descrizione appare la parola cosine (Nota: la documentazione di Matlab è in inglese!)

Scalari in Matlab

Matlab valuta espressioni e le associa a variabili.
Assegnazione della variabile z:

```
>> z=6
z =
       6

>> 6
ans =
       6
```

se non specificato il valore 6 viene assegnato alla variabile ans che contiene sempre l'ultimo valore non esplicitamente assegnato ad una variabile.

```
>> a=3+2;
>> a
a =
      5
```

Il ";" alla fine dell'istruzione sopprime la visualizzazione a schermo del risultato (ma non l'esecuzione dell'operazione!).

Sono definite le operazioni elementari: +, −, ∗ , /, ∧ (elevamento a potenza).

```
>> a=3+2, b=5-3, d=3*4, e=3/2, f=2^3
```

Attenzione alle precedenze:

```
>> 3+2*4
ans=
        11
>> 3*2^4
ans=
        48
```

Per alterare l'ordine delle operazioni si utilizzano le parentesi tonde.

```
>> (3+2)*4
ans=
        20
>> (3*2)^4
ans=
        1296
```

```
>> (who) whos
```

(elenca le variabili attualmente attive in memoria) e d`a alcune informazioni importanti sulle loro caratteristiche (tipo di oggetto, dimensioni ...)

```
>> clear all
```

cancella il valore di tutte le variabili attive in memoria.

Funzioni matematiche predefinite

sqrt(x)	$\sqrt{x}$
exp(x)	e^x, exp(1) = e = Costante di Nepero 2.7182818284...
log(x)	ln(x)
sin(x)	sen(x)
cos(x)	cos(x)
tan(x)	tan(x)
asin(x)	arcsen(x)

. . .

Per vedere l'elenco:

```
>> help elfun
```

Attenzione: Se z è un numero negativo log(z) non da errore ma restituisce un numero complesso:

```
>> log(-1)

ans =

        0 + 3.1416i
```

infatti se z è un numero negativo o complesso, la funzione log restituisce il logaritmo complesso, ovvero se $z = re^{i\theta}$

$$\log(z) := \log(r) + i\theta$$

Variabili predefinite

- pi (pigreco),
- i,j (unità immaginarie, anche scritte come 1i, 1j),

Ogni variabile può essere sovrascritta (attenzione!). Per tornare indietro: clear.

```
>>pi
    3.1416
>>pi=5;
>> clear pi
>> pi
       3.1416
```

Vettori in Matlab

Assegnazione di un vettore riga:

```
>> w=[1 2 3]
w =

     1     2     3
```

Assegnazione di un vettore colonna:

```
>> v=[1; 2; 3]
w =

     1
     2
     3
```

Altri modi di generare vettori riga:

```
>> v=[1:8]
v =

     1     2     3     4     5     6     7     8
>> v=[1:.5:3]
v =

    1.0000    1.5000    2.0000    2.5000    3.0000
```

La sintassi generale è v=[valore iniz:passo:valore finale]. Il passo può essere anche negativo, ad es. v=[10:-.5:1];

Il comando linspace(valore iniziale, valore finale, N) genera N valori equispaziati fra valore iniziale e valore finale (estremi compresi). Ad esempio

```
>>   v=linspace(0,1,5)
            0       0.2500       0.5000       0.7500       1.0000
```

Per accedere alla componente di un vettore:

```
>> v(3)
ans =

       2
```

Attenzione: in Matlab l'indicizzazione inizia da 1 e non da 0!

Matlab produce un messaggio di errore quando si cerchi di accedere ad una componente non definita, ad esempio:

```
>> z=v(0)
??? Subscript indices must either be real positive integers
or logicals.
```

Nota: esiste in Matlab la parola chiave ”end” per accedere all’ultimo elemento di un vettore. Ad es., se v ha dieci elementi, v(end) equivale a v(10).

Per conoscere la lunghezza di un vettore v:

```
>> numel(v)
ans =

     5
```

Per controllare la dimensione di una variabile v:

```
>> size(v)
ans =

     1     5
```

Il comando zeros(n,1)(zeros(1,n)) produce un vettore colonna (riga) di lunghezza n con elementi tutti nulli.

Il comando ones(n,1) (ones(1,n)) genera un vettore colonna (riga) con tutte le componenti pari a 1.

Operazioni su vettori

Dati i vettori v=[1 2 3 4] e w=ones(1,4).

Trasposizione:

```
>> u=v'
u =
     1
     2
     3
     4

  ans =
    5.4772
```

Somma (sottrazione) algebrica tra vettori di ugual dimensioni

```
>>  v+w
ans =
     2     3     4     5
>>  v-w
ans =
     0123
```

Prodotto scalare $(v, w) = (v_1w_1 + v_2w_2 + ... + v_nw_n)$.

```
>> v*w' (oppure dot(v,w)) ans
=
      10
```

Prodotto vettoriale

$$(v \wedge w) = [x_2w_3 - x_3w_2,\ x_3w_1 - x_1w_3,\ x_1w_2 - x_2w_1] = \det \begin{matrix} i & j & k \\ x_1 & x_2 & x_3 \\ w_1 & w_2 & w_3 \end{matrix}$$

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = cross(a,b)
c =
        -36-3
```

Attenzione alle dimensioni dei vettori!

Operazioni su vettori “componente per componente”.

Consideriamo i vettori di ugual dimensione a e b appena definiti. il prodotto componente per componente genera il vettore: (a_1b_1, a_2b_2, a_3b_3)

```
>> z= a.*b
z =
      4      10      18
```

la divisione componente per componente genera il vettore: (a_1/b_1, a_2/b_2, a_3/b_3)

```
>> a./b
ans =
      0.25000.40000.5000
```

l’elevamento a potenza componente per componente genera il vettore: ($a^{b_1}{}_1$, $a^{b_2}{}_2$, $a^{b_3}{}_3$)

```
>>a.^b
ans =
       1      32   729
>>a.^3
ans =
       1       827
```

Manipolazione di sottoblocchi di vettori e concatenazione

Siano v=[1 2 3 4 5] e w=[100 200].

Per sostituire alle ultime due componenti di v le componenti di w:

```
>> v=[1 2 3 4 5]; w=[100 200];
>> v(end-1:end)=w
    v=
        1 2 3 100 200
```

Per eliminare da v la terza e la quarta componente usiamo il vettore vuoto []:

```
>> v=[1 2 3 4 5];
>> v(3:4)=[];
    v=
        1 2 5
```

Per concatenare i due vettori:

```
>> z=[v w]
      z=
         1 2 3 4 5 100 200
```

Altre funzioni predefinite sui vettori

```
>> v=[1,5,3];
>> sum(v)
ans =
        9
>> prod(v)
ans =
       15
>> max(v)
ans =
        5
>> min(v)
ans =
        1
>> sort(v)
ans =
        1
        3
        5
>> %diff(x) => [x(2)-x(1), x(3)-x(2),...x(n)-x(n-1)].
>> diff(v)
ans =
        4       -2
```

Programmare con Matlab: Script-files

Che cos'è uno script?

- È' un file con estensione .m (ad esempio: myfile.m).
- Contiene una sequenza di istruzioni Matlab, scritte come se fossero digitate in modalità interattiva.
- Digitando il nome di uno script-file a destra del prompt:

  >> myfile

  vengono eseguite in successione tutte le istruzioni contenute nel file.
- Le variabili assegnate in uno script-file sono visibili dall esterno, ovvero persistono in memoria al termine dell'esecuzione.

Alcune buone regole

- Il nome di uno script-file deve essere diverso dai nomi delle variabili che esso elabora e dai nomi delle variabili presenti in Workspace, altrimenti non verrà eseguito.
- Non assegnare ad uno script-file il nome di una funzione pre-definita di Matlab. Per verificare se un nome esiste già:

  >> exist('nome')

Esempio

Scrivere uno script-file che dato $n \in \mathbb{N}$, calcoli il fattoriale $n!$
$= 1 \cdot 2 \cdot 3 \cdot \ldots (n-1) \cdot n$

```
n=input('inserisci numero intero positivo')
fact=prod([1:n])
disp('fattoriale=')
disp(fact)
```

Matrici in Matlab (primi comandi)

Definiamo le matrici:

```
>> A=[1 2 3; 4  5 6];
>> B=ones(2,3);


>> C=A+B;
>> D=A-B;
```

ed estraiamo gli elementi

```
>> s=B(1,2)+D(2,3)
s =
     6
```

L'operatore * esegue il prodotto righe per colonne:

```
3 4     4  5 6    =   19 26 33
1 2     1  2 3         9  12 15
```

```
>> [1;  2 ;3; 4]*[ 1 2 3 4]

ans =
     1     2     3     4
     2     4     6     8
     3     6     9    12
     4     8    12    16

>> [ 1  2 3 4]*[ 1; 2; 3; 4]

ans =
    30
```

Operazioni elemento per elemento (come per i vettori):

```
>> A=[1 2; 3 4], B=[1 2;-1 1];
>> A.*B;
>> A./B;
>> A.^B;
```

Matrici particolari

Matrice identità

```
>> I=eye(4)
I =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Matrici di Hilbert

$$\begin{matrix} 1 & 1/2 & 1/3 & 1/4 & \dots & 1/n \\ 1/2 & 1/3 & 1/4 & 1/5 & \dots & 1/(n+1) \\ 1/n & 1/(n+1) & & & \dots & 1/(2n-1) \end{matrix}$$

```
>> H=hilb(3)
H =
    1.0000  0.5000  0.3333
    0.5000  0.3333  0.2500
    0.3333  0.2500  0.2000
```

Matrici di numeri casuali

- A=rand(3)

```
A =
    0.9501  0.4860  0.4565
    0.2311  0.8913  0.0185
    0.6068  0.7621  0.8214
```

Manipolazione di sottoblocchi di matrici e concatenazione

Sia A=eye(4) e B=hilb(2). Per sostituire alle ultime due righe e colonne di A la matrice B:

```
>> A=eye(4); B=hilb(2);
>> A(3:4,3:4)=B
A =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000    0.5000
         0         0    0.5000    0.3333
```

Per estrarre la quarta riga di A:

```
>> r=A(4,:)
r =
         0         0    0.5000    0.3333
```

Per eliminare una colonna usiamo il vettore vuoto []:

```
>> A(:,4)=[] A
  =
    1.0000         0         0
         0    1.0000         0
         0         0    1.0000
         0         0    0.5000
```

Per concatenare due matrici (attenzione alle dimensioni!):

```
>> A=eye(3,2); B=zeros(3,4);
>> C=[A,B]
C =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     0     0     0     0

>> D=[C;ones(1,6)]
D =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     0     0     0     0
     1     1     1     1     1     1
```

Altre funzioni predefinite su matrici e vettori

```
>> v=[1:4];
>> A=diag(v)
A =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4

>> M=[1 2 9; 7 5 6; 4 8 3];
>> v=diag(M)
v =
     1
     5
     3

>> sum(M)
ans =
    12    15    18

>> prod(M)
ans =
  2880   162
```

```
??? max(M)
ans =
        7        8        9

>> min(M)
ans =
        1        2        3

>> sort(M)
ans =
        1        2        3
        4        5        6
        7        8        9

>> B=[4 -1 1;-1 3 -2; 1 -2 3];
>> det(B)
ans =
      18

>> C=inv(B)
C =
      0.2778 0.0556   -0.0556
      0.0556 0.6111  0.3889
     -0.0556 0.3889 0.6111
```

```
>> C*B
ans =
    1.0000         0    0.0000
         0    1.0000         0
         0         0    1.0000
```

Matlab come linguaggio di programmazione

Cicli condizionati (comando while)

Sintassi generale:

```
while (condizione==true)
        istruzione
        ...
        aggiornamento condizione
end
```

Cicli con contatore (comando for)

Sintassi generale:

```
for contatore = inizio:passo:fine (oppure espressione)
      istruzione
      ...
      istruzione
end
```

Esempio 1: Determinare il primo intero n tale che $\text{sum}_{i=1}^{n} n_i \geq 45$.

```
>> n=1;
>> while sum(1:n)<45
            n=n+1;
    end
```

Esempio 2: Calcolo del valor medio di un vettore x

```
>> M=sum(x)/numel(x)
```

oppure

```
>> m=mean(x)
```

Esempio 3: Dato n, calcolare la matrice triangolare superiore A definita da

$$A_{i,j} = \begin{cases} i/(j+1) & \text{se } i < j \\ n & \text{se } i = j \end{cases}$$

Soluzione 1 (in uno script-file, realizzando un doppio ciclo)

```
>> clear A
>> n=input('inserisci dimensione matrice n=');
>> for i = 1:n
     for j=i+1:n
          A(i,j)=i/(j+1);
     end
   A(i,i)=n;
end
```

Soluzione 2 (in uno script-file usando operazioni vettoriali)

```
>> n=input('inserisci dimensione matrice n=');
>> A=eye(n)*n;
>> for i=1:n,  A(i,i+1:n)=i./[i+2:n+1]; end
```

E' buona regola prima di scrivere un ciclo vedere se è possibile evitarlo tramite un opportuno uso di istruzioni vettoriali.

Istruzioni condizionali (comando if)

Sintassi generale:

```
if (condizione1==true)
      istruzioni 1
elseif (condizione2==true)
      istruzioni 2
else
      istruzioni 3
end
```

Operatori relazionali e logici

Il valore 1 corrisponde ad una condizione vera, 0 ad una falsa.

>><, <=,>, >=, ==,~=

a==b −→ 1 se a= b, 0 altrimenti

a~=b −→ 1 se a6=b, 0 altrimenti

- &, ||, ~, xor

a, b	a & b	a|| b	xor(a,b)	~ a
0, 0	0	0	0	1
1, 0	0	1	1	0
0, 1	0	1	1	1
1, 1	1	1	0	0

Esempio: Dato n, costruire il vettore a di lunghezza n, definito da

$$a_i = \begin{cases} \left(\frac{1}{i}\right) & \text{se } i = 1, \text{ oppure } i = 3 \\ \frac{1}{(i-1)(i-3)} & \text{altrimenti} \end{cases}$$

Soluzione 1 (realizzando un ciclo)

```
>> clear a
>> n=input('inserisci dimensione vettore');
>> for i = 1:n
     if (i==1)||(i==3)
          a(i) = 1/i;
     else
          a(i) = 1/((i-1)*(i-3));
     end
end
```

Soluzione 2 (usando istruzioni vettoriali)

```
>> a=1./[1:n];
>> ind=[2,4:n];
>> a(ind)=1./((ind-1).*(ind-3))
```

Function in Matlab

Sono porzioni di codici scritte in un file indipendente che svolgono un determinato compito e comunicano con lo spazio di lavoro solo attraverso i parametri in ingresso ed in uscita.

L'intestazione di una function Matlab ha sempre la struttura:

```
function [out1, out2, out3] =      nomefun    (in1, in2, in3)
```

parola chiave — nome funzione — parametri in uscita — parametri in ingresso

La funzione nomefun deve essere salvata nel file nomefun.m.

Ogni function termina con la parola chiave return. Prima di essa, deve essere stato assegnato un valore a ciascuno dei parametri in uscita out1,out2,...

Le variabili nel blocco istruzioni interno alla function sono locali, ovvero vengono cancellate dalla memoria al termine della chiamata.

Per chiamare una function, ad esempio dallo spazio di lavoro:

```
>> [value1,value2,value3]=nomefun(in1,in2,in3);
```

Una funzione può richiamare o essere richiamata da altre.

Esempio: Scrivere una function Matlab tale che dati due vettori riga v1, v2 calcoli il vettore somma s in modo tale che se v1 e v2 non hanno la stessa lunghezza al vettore pi`u corto vengano aggiunti tanti zeri in testa fino ad ottenere due vettori sommabili.

Soluzione 1

```
function s=vsum(v1,v2)
%
1  s=vsum(v1,v2)
1  somma vettori riga eventualmente di lunghezza diversa
1
[mv1,nv1]=size(v1);
[mv2,nv2]=size(v2);
if ~((mv1==1)&(mv2==1))
     disp('errore dimensioni dei vettori incompatibili') return

end dif=abs(nv1-
nv2); if nv1>nv2

            s=v1+[zeros(1,dif) v2];
elseif nv1==nv2
            s=v1+v2;
else
            s=v2+[zeros(1,dif) v1];
end
return
```

Soluzione 2 (in forma compatta)

```
function s=vsum(v1,v2)
%
>>   s=vsum(v1,v2)
>>   somma vettori riga eventualmente di lunghezza diversa
>>
[mv1,nv1]=size(v1);
[mv2,nv2]=size(v2);
if ~((mv1==1)&(mv2==1))
     disp('errore dimensioni dei vettori incompatibili') return

end d=nv2-nv1;
s=[zeros(1,d),v1]+[zeros(1,-d),v2]; return
```