Laboratorio 2 - Introduzione a MATLAB

Matrici in Matlab (primi comandi)

Per assegnare le matrici

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

Possiamo calcolare

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

ed estrarre gli elementi

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

L'operatore `*` esegue il prodotto righe per colonne:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 9 & 12 & 15 \\ 19 & 26 & 33 \end{bmatrix}$$

```
>> [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

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

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

Matrici di Vandermonde: sono matrici le cui colonne contengono potenze di un vettore di riferimento $x = [x_1, \ldots, x_n]$

$$\begin{bmatrix} x_1^{n-1} & \ldots & x_1^2 & x_1 & 1 \\ x_2^{n-1} & \ldots & x_2^2 & x_2 & 1 \\ x_3^{n-1} & \ldots & x_3^2 & x_3 & 1 \\ \ldots & \ldots & \ldots & \ldots & \\ x_n^{n-1} & \ldots & x_n^2 & x_n & 1 \end{bmatrix}$$

```
>> V=vander([1 2 3 4])
V =
     1     1     1     1
     8     4     2     1
    27     9     3     1
    64    16     4     1
```

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
```

Risolvere Esercizi 1 e 2

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 =
    28    80   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
```

Risolvere Esercizio 3

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 $\sum_{i=1}^{n} i \geq 45$.

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

Esempio 2: Calcolo del valor medio $M = \frac{1}{n}\sum_1^n x(i)$

Soluzione 1 (in uno *script-file*)

```
>>x=input('dammi x=');
  m=0;
  for i=1:length(x)
     m=m+x(i);
  end
  m=m/length(x)
```

Soluzione 2 (in uno *script-file*)

```
>>x=input('dammi x=');
  m=0;
  for xi=x
      m=m+xi;
  end
  m=m/length(x)
```

Soluzione 3 (in forma compatta)

```
>> m=sum(x)/length(x)
```

Soluzione 4 (utilizzando la funzione apposita di Matlab)

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

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

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

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
```

È 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` $\longrightarrow$ 1 se `a= b`, 0 altrimenti
  `a~=b` $\longrightarrow$ 1 se `a`$\neq$`b`, 0 altrimenti

- &, `||`, `~`, `xor`

a, b	a & b	a`||` b	xor(a,b)	$\sim a$
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

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

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))
```

Risolvere Esercizi 4 e 5.

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:

$$\underbrace{\texttt{function}}_{\text{parola chiave}} \underbrace{[\texttt{out1},\texttt{out2},\texttt{out3}]}_{\text{parametri in uscita}} = \underbrace{\texttt{nomefun}}_{\text{nome funzione}} \underbrace{(\texttt{in1},\texttt{in2},\texttt{in3})}_{\text{parametri in ingresso}}$$

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

Un file può contenere un unica funzione.

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 più corto vengano aggiunti tanti zeri in testa fino ad ottenere due vettori sommabili.

Soluzione 1

```
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
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
```

Risolvere Esercizio 6