MATLAB

Informatica A - Prof. Perego Paolo, Ph.D

Introduzione

Matlab è un software di programmazione scritto in C sviluppato per il calcolo numerico e l'analisi statistica. Matlab permette di eseguire script scritti in un linguaggio di programmazione proprio: è un linguaggio interpretato e non compilato.

Matlab come calcolatrice

Matlab può essere utilizzato come calcolatrice. Se non specificato il risultato è inserito automaticamente nella variabile ans. Se la riga finisce con un ';' (punto e virgola), l'operazione viene eseguita ma il risultato non è mostrato.
Esempi:

Somma
Operazione con risultato nella variabile ans

```
>> 5+3
ans =
     8
```

Operazione con risultato nella variabile a

```
>> a = 5+3
a =
   8
```

Operazione con visualizzazione manuale della variabile a

```
>> a = 5+3;
>>
>> a
a =
   8
```

Elevamento a potenza

```
>> 5^3
ans =
     125
```

Radice quadrata

```
>> sqrt(144)
ans =
     12
```

Divisione Destra

```
>> 5/3
ans =
     1.6667
```

Divisione Sinistra

```
>> 5\3
ans =
     0.6000
```

Valore Assoluto

```
>> abs(-123)
ans =
     123
```

Pi greco

```
>> pi
ans =
    3.1416
```

Attenzione, non occorre dichiarare le variabili come nel C, basta assegnare un valore per dichiarare automaticamente la variabile. Tuttavia non si possono utilizzare variabili che non siano mai state assegnate.

Matrici e Vettori

Vettore riga

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

Vettore colonna

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

Trasposizione da riga a colonna e viceversa

```
>> A
A =
    1     2     3     4     5
>> A'
A =
     1
     2
     3
     4
     5
```

Accedere ad una posizione nella matrice X = riga Y = colonna A(X,Y) -> Valore di A alla riga X e colonna Y eg.: riga 2 e colonna 3

```
>> A = [1 2 3 4 5;4 5 6 7 8; 9 10 11 12 13]
A =
     1     2     3     4     5
     4     5     6     7     8
     9    10    11    12    13
>> A(2,3)
ans =
     6
```

Selezionare i numeri da un punto del vettore in poi

```
>> A = [1  2  3  4  5]
A(2:end)
ans =
      2      3      4      5
```

Per creare una matrice possono essere combinati anche vettori

```
>> A = [1  2  3  4  5];
>> B = [A; A.^2; A.^5]
ans =
           1           2           3           4           5
           1           4           9          16          25
           1          32         243        1024        3125
```

Il punto davanti all'operatore di elevamento a potenza indica a Matlab di effetturare l'operazione sui singoli valori del vettore.

Creare matrici di zeri - zeros(n) crea una matrice quadrata di dimensione n - zeros(n,m) crea una matrice di n righe e m colonne

```
>> A = zeros(5)
A =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
```

allo stesso modo possono essere create matrici di uno con la funzione **ones**. Moltiplicando la matrice ottenuta per un numero n, si ottiene una matrice di soli n.

```
>> A = ones(3,2)
A =
     1     1
     1     1
     1     1
>> A = 5*ones(3)
A =
     5     5     5
     5     5     5
     5     5     5
```

diag permette di generare una matrice diagonale o estrarre la diagonale da una matrice

```
>> diag([1  2  3  4])
ans =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4

>> A = [1  2  3  4; 5  6  7  8;9  10  11  12]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12

>> diag(A)
ans =
     1
```

```
     6
    11

>> diag(diag(A))
ans =
     1     0     0
     0     6     0
     0     0    11
```

Per concatenare due matrici/vettori si possono usare i comandi **horzcat** e **vertcat**

```
A = [1  2  3]
A =
     1     2     3
>> B = [4  5  6]
B =
     4     5     6
>> C = [A B]
C =
     1     2     3     4     5     6
>> D = horzcat(A,B)
D =
     1     2     3     4     5     6
>> E = vertcat(A,B)
E =
     1     2     3
     4     5     6
```

Occorre tenere sempre in considerazione la dimensione delle matrici/vettori.

```
>> A = [1  2  3  4]
A =
     1     2     3     4
>> B = [5  6  7]
B =
     5     6     7
>> vertcat[A,B]
 vertcat[A,B]
        |
Error: Unbalanced or unexpected parenthesis or bracket.
```

Per calcolare il determinante di una matrice si usa la funzione **det**

```
>> A = [6  4  0; 4  2  7; 1  2  8]
A =
     6     4     0
     4     2     7
     1     2     8
>> det(A)
ans =
  -88.0000
```

L'operatore :
Per creare un vettore riga contenente numeri tra 3 e 13:

```
>> A = 3:13
A =
     3     4     5     6     7     8     9    10    11    12    13
```

Per creare un vettore riga contenente un numero ogni 4 da tra 3 e 23

```
>> A = 3:4:23
A =
     3     7    11    15    19    23
```

Per creare un vettore colonna contenente 7 numeri da 11,5 a 14,2

```
>> A = 11.5:(14.2-11.5)/6:14.2
A =
   11.5000   11.9500   12.4000   12.8500   13.3000   13.7500   14.2000
```

la stessa operazione può essere eseguita grazie alla funzione **linspace**.

```
>> A =linspace(11.5,14.2,7)
A =
   11.5000   11.9500   12.4000   12.8500   13.3000   13.7500   14.2000
```

L'operatore ':' duepunti viene utilizzato anche per identificate l'intera riga quando si vuole selezionare una parte della matrice.
Es. Crare il vettore riga contente la seconda riga e il vettore riga contenente la terza colonna.

```
>> A = [1 2 3 4 5 6;7 8 9 10 11 12; 13 14 15 16 17 18; 19 20 21 22 23 24; 25 26 27 28 29 30]
A =
     1     2     3     4     5     6
     7     8     9    10    11    12
    13    14    15    16    17    18
    19    20    21    22    23    24
    25    26    27    28    29    30

>> riga = A(2,:)
riga =
     7     8     9    10    11    12

>> riga2 = A(:,3)
riga2 =
     3
     9
    15
    21
    27
```

Numeri casuali e modifica di matrici/vettori

Per creare numeri casuali, Matlab usa la funzione rand(n) o rand(n,m).
rand genera una matrice nxn o nxm contenente numeri casuali tra 0 e 1.
Per creare numeri casuali interi si utilizza invece la funzione **randi([min max],[n m]**.
randi([min max],[n m] genera una matrice di n righe e m colonne contenente numeri interi casuali da min a max.

```
>> rand(5)
ans =
    0.2769    0.3171    0.7655    0.6463    0.6551
    0.0462    0.9502    0.7952    0.7094    0.1626
    0.0971    0.0344    0.1869    0.7547    0.1190
    0.8235    0.4387    0.4898    0.2760    0.4984
    0.6948    0.3816    0.4456    0.6797    0.9597

>> rand(2,3)
ans =
    0.3404    0.2238    0.2551
    0.5853    0.7513    0.5060
```

Genera un numero casuale da 0 a 8

```
>> randi(8)
ans =
     5
```

Genera un numero casuale da 3 a 10

```
>> randi([3  10])
ans =
     8
```

Genera una matrice 2x2 con numeri casuali tra 0 e 5

```
>> randi(5,2)
ans =
     1     4
     2     3
```

Genera una matrice 5x2 con numeri casuali tea 3 e 10

```
>> randi([3  10],[5,2])
ans =
     5     5
     9     9
     7     9
     7     6
    10     7
```

Per conoscere le dimensioni di una matrice si usa la funzione **size**.
size(A) restituisce le due dimensioni della matrice:

```
a =
     5     5
     9     9
     7     9
     7     6
    10     7
>> size(a)
ans =
     5     2
```

size(A,1) restituisce il numero di righe
size(A,2) restituisce il numero di colonne
Se A è un vettore riga o colonna, size(A) resistuisce sempre due valori, di cui uno di valore 1. Per ottenere la dimensione di un vettore si può utilizzare la funzione **lenght(A)**.

```
>> A = [1  2  3  4  5]
A =
     1     2     3     4     5
>> size(A)
ans =
     1     5
>> A=A'
A =
     1
     2
     3
     4
     5
>> size(A)
ans =
     5     1
>> length(A)
ans =
     5
```

Calcolo della media e somme

La media può essere calcolata per mezzo della funzione **mean**.
Se A è una matrice, l'operazione **mean(A)** restituisce la media delle colonne.

```
>> A = [6  4  0;  4  2  7;  1  2  8]
A =
```

```
     6     4     0
     4     2     7
     1     2     8
>> mean(A)
ans =
    3.6667    2.6667    5.0000
```

mean(A,1) equivale a size(A) perché calcola la media delle colonne
mean(A,2) calcola invece la media delle righe.

```
>> mean(A,1)
ans =
    3.6667    2.6667    5.0000
>> mean(A,2)
ans =

    3.3333
    4.3333
    3.6667
```

Per ottenre la media dell'intera matrice si usa **mean(mean(A))**

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

Se occorre invece sommare il contenuto di vettori o matrici si usa la funzione **sum**
sum(A) e sum(A,1) somma i numeri della matrice A sulle colonne
sum(A,2) somma i numeri della matrice A sulle righe
Per sommare Altre funzioni utili possono essere:

rem(A,B) calcola il resto della divisione.
La funzione si comporta in particolari condizioni come segue:

```
rem(x,0) vale NaN
rem(x,x) per x diverso da zero vale zero
rem(x,y) per x diverso da y e y diverso da zero, ha lo stesso segno di x.
```

mod(A,B) calcola il resto della divisione.
La funzione mod è del tutto simile a rem ma si comporta in modo differente:

```
mod(x,0) vale x
mod(x,x) vale 0
mod(x,y) per x diverso da y e y diverso da zero, ha lo stesso segno di y.
```

Entrambe queste funzioni possono essere utilizzate per verificare se un numero è pari/dispari o per controllare se divisibile per n:

```
>> rem(A,2)==0
```

Ritorna 1 se il numero A è pari, 0 altrimenti. Può essere utilizzata come condizione per scegliere solamente i numeri pari. "rem" può essere benissimo sostituito con mod. Se A è una matrice, l'operazione restituisce una matrice della stessa dimensione di A, contenente 1 se in quella posizione il numero nella matrice A è pari, 0 se dispari.

```
>> mod(A,n)==0
```

Ritorna 1 se il numero A è divisibile per n, 0 altrimenti. Se A è una matrice, l'operazione restituisce una matrice con la stessa dimensione dio A, contente 1 ove il numero è divisibile per n, 0 altrimenti.

L'operatore **find** permete di ricercare dei valori all'interno di una matrice.

```
i=find(X)
```

ritorna un indice (tramite un vettore colonna) dove vengono travati in X valori non zero.

```
i=find (X,k)
```

come sopra, ma si ferma al k-esimo valore.

```
[i,j] = find(X)
```

ritorna le posizioni righe / colonne dove vengono trovati valori non zero.

Se all'interno delle parentesi tonde di find si inserisce una condizione, è possibile ricercare la riga e la colonna nella quale la condizione è verificata.
Prendiamo per esempio la matrice A cosi' composta:

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

Per ottenere gli indici di A che contengono il numero 3 si procede come segue:

```
>> [i,j] = find(A==3)
i =
     1
j =
     3
```

Per ottenere gli indici di A che contengono un numero divisibile per 3 si procede come segue:

```
>> [i,j]=find(rem(A,2)==0)
i =
     2
     1
     3
     2
j =
     1
     2
     2
     3
```

Cicli e condizioni

A differenza del C, in matlab ogni condizione o ciclo inizia con IF/FOR... e termina sempre con END.

CONDIZIONE IF
La struttura dell'IF è così costituita:

```
if I == J
   A(I,J) = 2;
elseif abs(I-J) == 1
   A(I,J) = -1;
else
   A(I,J) = 0;
end
```

CICLO FOR
La struttura del FOR è così costituita:

```
for i = 1:N
   %Istruzioni da ripetere
end
```

CICLO WHILE
La struttura del WHILE è così costituita:

```
while a > 0
   %Istruzioni da ripetere
end
```

Le funzioni

Matlab permette di creare delle funzioni. Queste vanno create in un file .m che abbia lo stesso nome della funzione. Per creare il file l'istruzione è:

```
edit funzione.m
```

La struttura della funzione è la seguente:

```
function [out1, out2, ...] = nomeFunzione(in1,in2...)
   %Contenuto della funzione
end
```

Come si può notare, per i parametri di ingresso non occorre specificare il tipo di variabile. Anche per i parametri di uscita non occorre specificarne il tipo e, a differenza del C, si possono avere anche più parametri. Non esiste il passaggio per puntatore. Se si vogliono modificare gli stessi parametri di ingresso, o si sovrascrivono passandoli sia in ingresso che in uscita, o si usano variabili globali.

ESERCIZI

All'interno degli esercizi sono presenti alcuni dei temi d'esame degli anni precedenti.

1. Creare una matrice 4x5 contenente solo 3

```
A = 3*ones(4,5)
>> A = 3*ones(4,5)
A =
     3     3     3     3     3
     3     3     3     3     3
     3     3     3     3     3
     3     3     3     3     3
```

2. Inserire valori da 1 a 5 nella riga 2

```
>> A(2,:) = 1:5
A(2,:) = 1:5
A =
     3     3     3     3     3
     1     2     3     4     5
     3     3     3     3     3
     3     3     3     3     3
```

3. sostituire le colonne dispari con soli numeri 10

```
>> A(:,1:2:end) = 10*ones(4,3)
```

oppure

```
>> A(:,1:2:end) = 10*ones(size(A(:,1:2:end)))
A =
     3     3     3     3     3
     1     2     3     4     5
     3     3     3     3     3
     3     3     3     3     3
```

4. sostituire i numeri dispari con 0

```
>> A(rem(A,2)~=0) = 0
A =
    10     0    10     0    10
    10     2    10     4    10
    10     0    10     0    10
    10     0    10     0    10
```

5. Radice quadrata della somma di A

```
>> sqrt(sum(sum(A)))
ans =
   11.2250
```

6. Eliminare le righe dispari

```
>> A(1:2:end,:) = []
A =
    10     2    10     4    10
    10     0    10     0    10
```

7. Eliminare le colonne con media >0

```
A =
    -9     9   -10    -4    -5
    -9    -8    -3     1     3
     1     1    -7    -7     4
     6    -1     6     2     5
>> A(:,mean(A)>0)=[]
A =
    -9   -10    -4
    -9    -3     1
     1    -7    -7
     6     6     2
```

N.B: Non può essere eliminata la singola cella di una matrice. Può solo essere annullata.

8.Eliminare la colonna 3

```
>> A = [A(:,1:2) A(:,4:end)]
```

oppure

```
>> A(:,3) = []
```

9. Inserire, dopo la seconda colonna, una colonna contenente valori casuali da -11 a 37

```
>> A
A =
    -9     9   -10    -4    -5
    -9    -8    -3     1     3
     1     1    -7    -7     4
     6    -1     6     2     5
>> A = [A(:,1:2) randi([-11 37], [4 1]) A(:,3:end)]
A =
    -9     9    -4   -10    -4    -5
    -9    -8    29    -3     1     3
     1     1    15    -7    -7     4
     6    -1    37     6     2     5
```

10. Chiedere un numero all'utente e stampare solamente i numeri di cui il numero inserito è un divisore

```
>> n = input('Inserisci un numero: ')
n =
     4
>> A(rem(A,n)==0)
```

11. Creare una matrice B che contiene i valori di A meno la media dei soli valori negativi

```
>> B = A - mean(A(A<0))
```

12. Scrivere una funzione matlab che calcoli la somma dei valori interi da 1 fino a n, con n inserito dall'utente. Stampare a video solamente il valore della somma.

```
function s = somma(n)
   if (n==0)
     s = 0;
   else
     s = n + somma(n-1);
   end
end
```

13. Scrivere una funzione ricorsiva per il calcolo nella serie di fibonacci

```
function n = fibonacci(x)
   if (x==1)
     n=0;
   else if(x==2)
     n=1;
   else
     n = fibonacci(x-1)+fibonacci(x-2);
   end
end
```

14. Creare la matrice M con dimensione di righe e colonne casuali (diverse) da 5 a 10 contenente solo valori pari a 1; (max 1 riga)

```
>> M = ones(randi([5 10],[1 2]));
```

15. Inserire nelle righe dispari valori casuali da 1 a 10 (max 1 riga)

```
>> M(1:2:end,:) = randi([1,10],size(M(1:2:end,:)))
```

16. Calcolare la media delle celle con valore maggiore di 3

```
>> mean (M(M>3))
```

17. Scrivere una funzione che presa in ingresso la matrice M, crei una nuova matrice "specchio" N con la prima colonna in ultima posizione, la seconda in penultima... (max 8 righe)

```
function [N] = funzione(M)
    for i=1:1:size(M,2)
        N(:,size(M,2)-i+1)) = M(:,i);
    end
end
```

18. Creare un vettore colonna A e un vettore colonna B contenente rispettivamente 6 e 5 valori casuali differenti tra loro con valori da 1 a 10; (max 2 righe)

```
>> A = randi([1 10],[6 1]);
>> B = randi([1 10],[1 5]);

oppure

>> A = randi([1 10],[6 1]);
>> B = randi([1 10],[5 1]);

Devo porre attenzione al fatto che siano entrambi vettori colonna
```

19. A partire dai vettori A e B, creare una matrice C di 6 righe e 5 colonne.

```
Se una colonna e una riga
>> C = A*B;
>> C = A.*B;

Se entrambe colonne:

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

20. Cancellare le righe che hanno come primo valore un numero inferiore a 10 (max 1 riga)

```
>> C(C(:,1)<10,:)=[]
```

21. Calcolare la somma di tutte le colonne dispari (max 1 riga)

```
>> sum(sum(C(:,1:2:end)))
```

22. Scrivere una funzione che presa in ingresso la matrice C, crea una nuova matrice "cornice", ponendo cioè a zero tutti gli elementi non appartenenti alla prima e ultima riga/colonna. (max 8 righe)

```
function [out] = funzione(in)
    out = in;
    for i=2:1:size(in,1)-1
        for j=2:1:size(in,2)-1
           out(i,j) = 0;
        end
    end
end
```

23. Scrivere una funzione che presi in ingresso il numero di righe ed il numero di colonne, crea la matrice M(righe,colonne) contenente i primi N = (righe x colonne) numeri dispari(max 12 riga)

```
function out = luglio5_19(row,column)
    c = 1;
    for (i=1:row)
        for (j=1:column)
            out(i,j) = c;
            c = c+2
        end
    end
end
```

24. Sostituire in una colonna a caso, tutti valori pari a uno (NB.: l'istruzione deve essere la più generale possiible, in modo da poter essere utilizzata con matrici di dimensini differenti) (max 1 riga)

```
>> A(:,randi(size(A,2))) = ones(size(A,1),1)
>> A(:,randi(size(A,2))) = ones(1,size(A,1))
```

25. Eliminare la riga con media più bassa (max 2 riga).

```
>> M(mean(M,2)==min(mean(M,2)),:)=[]
```

26. Moltiplicare tutte le celle per un numero casuale intero tra 5 e 18 (max 1 riga)

```
>> M = M * randi([5,18],1)
>> M = M * randi([5,18])
```

27. Moltiplicare per -1 tutte le celle che contengono numeri divisibili per 7 (max 1 riga)

```
>> M(rem(M,7)==0) = M(rem(M,7)==0)*-1
>> M(mod(M,7)==0) = M(mod(M,7)==0)*-1
```

Calcolo simbolico

Matlab può essere utile anche per semplificare espressioni e risolvere equazioni attraverso quello che viene denominato calcolo simbolico.
Occorre prima di tutto definire le variabili utilizzate nell'espressione:

```
>> x = sym('x')
x =
x
```

COLLECT: raccoglie i coefficienti con la stessa potenza

```
>> x = sym('x');
>> E = (x-1)*(x-2)*(x-3);
>> collect(E)
ans =
x^3 - 6*x^2 + 11*x - 6
```

```
>> x = sym('x');
>> y = sym('y');
>> E = (x-5)^2+(y-3)^2;
>> collect(E,y)
ans =
y^2 - 6*y + (x - 5)^2 + 9
```

EXPAND: applica regole algebriche per espandere l'espressione

```
>> x = sym('x');
>> y = sym('y');
>> E = cos(x+y);
>> expand(E,y)
ans =
cos(x)*cos(y) - sin(x)*sin(y)
```

FACTOR: esprime l'espressione come prodotto di polinomi

```
>> x = sym('x');
>> y = sym('y');
>> E = x^3-6*x^2+11*x-6;
>> factor(E)
ans =
[ x - 3, x - 1, x - 2]
```

POLY2SYM: converte coefficienti del vettore p in un polinomio simbolico

```
>> P = [2 6 4];
>> poly2sym(P)
ans =
2*x^2 + 6*x + 4
```

SYM2POLY: fa il contrario

```
>> V = 2*x^2 + 6*x + 4;
>> sym2poly(V)
ans =
     2     6     4
```

PRETTY: visualizza l'espressione in forma matematica

```
>> E = 2*x^2 + 6*x + 4;
>> pretty(E)
```

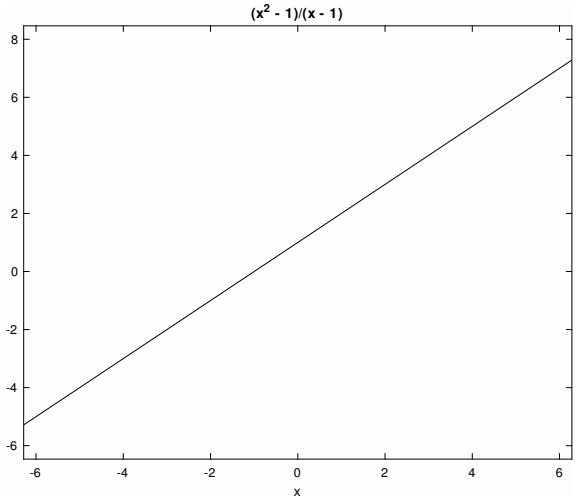
$2\,x^2 + 6\,x + 4$

SIMPLIFY: semplifica l'espressione

```
>> E = (1-x^2)/(1-x)
>> simplify(E)
ans =
x + 1
```

EZPLOT: disegna l'equazione ad una variabile

```
>> ezplot(E)
```



SOLVE: risolve equazioni e sistemi

```
>> E = '6*x+2==0';
>> solve(E)
ans =
-1/3
```

DIFF: derivata

```
>> diff(sin(x)^2)
ans =
2*cos(x)*sin(x)
```

INT: integrale

```
>> int(x^n)
ans =
piecewise(n == -1, log(x), n ~= -1, x^(n + 1)/(n + 1))
```

LIMIT: integrale

```
>> limit(sin(a*x)/x)
ans =
a
>> limit(1/x,x,0,'right')
ans =
Inf
>> limit(1/x,x,0,'left')
ans =
-Inf
```

SYMSUM: restituisce la somma

```
>> symsum(x^2,1,4)
ans =
30
```

TAYLOR: polinomio di taylor

```
>> taylor(exp(x))
ans =
x^5/120 + x^4/24 + x^3/6 + x^2/2 + x + 1
```

DSOLVE: risolve equazioni differenziali