```
%come scrivere vettori

v=[1 7 8 9 5]; %vettore riga solamente con le virgole o con spazi
u=[1;69;75;9;15]; %vettore colonna coi punti e virgola

%vettore trasposto
v_trasposto= v';

%MATRICE INVERSA, solamente con matrici del tipo NxN
Y= rand(7,7);
Y1=inv(Y);

%array index

v(1:3); %estraiamo gli elementi da 1 a 3 del vettore riga v
idx1=[1:3]; v(idx1); %prima definisco il vettore idx con gli indici degli elementi che
%voglio estrarre e poi di fianco metto v(idx) che mi dà il vettore degli
%elementi che ho selezionato
idx2=[2,4]; v(idx2); %prendo secondo e quarto elemento di v

%range di numeri
f=[0.5:0.2:1]; %vettore riga con elementi da a=0.5 a b=1 spaziati con intervallo
%di 0.2

%come accedere agli elementi di una matrice
A=rand(6,6);
A(:,1); %elementi della prima colonna (fisso l'indice di colonna ma non quello
%di riga)
A(:,1:3); %estraggo le prime tre colonne di A
A(:,[1 3 6]); %creo una nuova matrice incollando le colonne 1,3,6 della A
A(3,:); %comando per accedere alla terza riga della matrice

%estrazione della sottomatrice
A([1 2], [1:2]);
%combinazione degli intervalli
A=rand(6,6);
idx_righe=[1 3 5]; %indici delle righe che voglio estrarre
idx_colonne=[2 4 6]; %indici delle colonne che voglio estrarre
C=A(idx_righe,idx_colonne); %mi fa la combinazione degli elementi che soddisfano
%le mie richieste (sono nella riga 1,3,5 e nella colonna 2,4,6)
%IMPORTANTE: PRIMA DELLA PARENTESI SCRIVO LA LETTERA DELLA MATRICE DA CUI
%VOGLIO ESTRARRE

%operazioni tra vettori e tra matrici IMPORTANTE: METTO SEMPRE IL PUNTO
%QUANDO VOGLIO MOLTIPLICARE TUTTI GLI ELEMENTI DI UNA MATRICE PER TUTTI GLI
%ELEMENTI DI ALTRO



%moltiplico elemento per elemento
x=[1 2 3 4];
y=[5 6 7 8];
ele_ele= x.*y %prodotto elemento per elemento
```

```
sum(ele_ele) %ho ottenuto la somma dei vari prodotti elemento per elemento
%e quindi ho fatto il prodotto scalare

%PRODOTTO SCALARE TRA VETTORI= non posso fare prodotto scalare tra vettore
%riga e vettore colonna
%mediante ciclo for
x=[1 2 3 4];
y=[5 6 7 8];
N=4;
pr_scalare=0;
for i=1:N
   pr_scalare= pr_scalare + x(i)*y(i);
end
disp(pr_scalare);

%PRODOTTO TRA VETTORI E MATRICI: posso farlo solamente se, data una matrice
%A(5,7) il vettore è del tipo (7,1) ossia vettore colonna di righe pari
%alle colonne della matrice
M=4;
N=3;
A=rand(M,N);
t=rand(N,1);
pr_matrvett=zeros(M,1); %inizializzo sempre le variabili
for i=1:M;
   for k=1:N;
      pr_matrvett(i)= pr_matrvett(i) + A(i,k)*t(k);
   end
end
disp(pr_matrvett- A*t)
disp(pr_matrvett);
%modo semplice
pr_mv= A*t;
disp(pr_mv);

%PRODOTTO MATRICE PER MATRICE
%1)mediante ciclo for
N=2;
M=4;
P=5;
D=rand(M,N);
E=rand(N,P);
C=zeros(M,P);
for i=1:M
   for k=1:N
      for j=1:P
         C(i,j)= C(i,j) + D(i,k)*E(k,j);
      end
   end
end
disp(C-D*E);
disp(C);

%modo semplice prodotto matrice matrice
```

```
C1= D*E;
disp(C1);

%STUDIO DELLA SERIE GEOMETRICA
%q^n, forniamo in ingresso i dati di q e n
q=0.5;
n=10;
S=0;
for i=0:n-1 %vanno da 0 a n-1
   S= S+ q^i;
end
S_inf= 1/(1-q);
disp([S S_inf, abs(S_inf - S)]);
%calcolo della serie mediante il limite pari a 1/(1-q) ed è S_inf

%SOLUZIONE DEI SITEMI LINEARI DEL TIPO Ax=b
A=rand(6,6);
b=rand(6,1);
%primo metodo
x=inv(A)*b;
%secondo metodo
x1=A\b;
disp(x);
disp(x1);
disp(x-x1);

%TEST PROPRIETA' AUTOVETTORI E AUTOVALORI
S=rand(4,4);
%come prima cosa devo rendere simmetrica la matrice, la sommo alla inversa
%e divido per 2
S_simm=(S+S')/2;
[X Lambda]= eig(S_simm);
%X matrice degli autovettori, Lambda matrice diagonale degli autovalori,
%già ordinati in ordine crescente

%scrivi il massimo autovalore
massimo= max(abs(eig(A)));
disp(massimo);



%PRODOTTO ELEMENTO PER ELEMENTO
x=[1 2 3 4];
y=[5 6 7 8];
ele_ele= x.*y %prodotto elemento per elemento
sum(ele_ele) %ho ottenuto la somma dei vari prodotti elemento per elemento
%e quindi ho fatto il prodotto scalare

%anche nel prodotto tra matrici, significa che moltiplico ogni elemento per
%il suo corrispondente
Z=[1 2; 3 4];
Q=[5 6; 7 8];
%prodotto ELEMENTO PER ELEMENTO della matrice
```

```
W=Z.^2 %!!!! sono diversi
W1=Z*Z

%COME SCRIVERE LE FUNZIONI
%devo scrivere le funzioni usando il simbolo elemento per elemento "." in
%modo da scrivere correttamente i prodotti

%ESEMPIO DALLA SIMULAZIONE Riportare il grafico della funzione f(x)
%(definita sotto) per -0.5 < x < 1, con passo ?x = 0.1. Utilizzare il comando
% plot con la modalità di tracciamento 'ko-'. Limitare l'estensione dei valori
%rappresentati sull'asse y nell'intervallo [-1 1]. Si suggerisce di salvare
%la figura in formato pdf utilizzando il menu della finestra in cui Matlab
%mostra il grafico a seguito del comando plot.

clear all; close all; clc;
x=[-0.5:0.1:1]; %intervallo di valori delle x
y=(1+sin(x))./(x.^2+2*x-1) %scrivo la funzione coi punti ogni volta che faccio
%operazioni lunghe come divisione ed elevamento a potenza

plot(x,y,'ko-'); %ogni volta che faccio plot metto sia la x che la y che la
%modalità con cui voglio il plot, a pallini sui vari punti e nero
ylim([-1,1]); %valori limite delle y in doppia parentesi e SEMPRE DOPO IL PLOT

%FUNZIONE ROOTS= si tratta di un comando Matlab per il calcolo degli zeri
%del polinomio, devo mettere dentro una doppia parentesi dopo il comando
%roots i coefficienti dei termini del polinomio
disp('gli zeri sono');
roots([1 2 -1])


%SOMMA DI ELEMENTI DI UNA SERIE
a=0.0;
for n=1:10
   a= a+((-1)^n)*2.5*(3)^(-n);
end
disp(a);
%scrivo la forma della serie senza punti, ciclo for inizializzando la
%variabile e faccio il display solamente della forma finale, che è già la
%sommatoria

clear all; close all; clc;
%COMANDO SORT: permette di ordinare gli elementi in ordine crescente,
%tramite invece [SA, IDX]= sort(A) otteniamo sia l'ordine crescente ma
%anche gli indici. A(IDX) significa che leggiamo gli elementi di A secondo
%gli indici e quindi li leggiamo in ordine crescente
C=rand(6,1);
sort(C);
[SA, IDX]= sort(C);

%MATRICE ESPONENZIALE PER LA RISOLUZIONE DI SISTEMI EQUAZIONI DIFFERENZIALI
%DI PRIMO GRADO (decadimento reattivo)

%È assegnato il sistema lineare di equazioni differenziali al primo ordine
```

```
%(riportato in calce in forma matriciale). Utilizzando la notazione di Matlab,
%la matrice A è definita in questo modo:
A(1,:) = [ -1.0 0.3 0.0 ];
A(2,:) = [ 0.0 -0.3 0.2 ];
A(3,:) = [ 1.0 0.0 -0.2 ];
%Ricavare il valore del vettore colonna n(t) per t = 0.75 a partire dal
%vettore colonna delle condizioni iniziali (al tempo t = 0)
t=0.75;
n0 = [ 1 0 0 ]';
%la formula da usare è n(t)=expm(A*t)n(0)
nt=expm(A*t)*n0

%CALCOLO DELLA MEDIA E DELLA DEVIAZIONE STANDARD SU MATLAB

%È data la sequenza di 7 misure indipendenti di una certa grandezza fisica
%(con due cifre significative dopo la virgola):
w=[14.00 13.51 17.95 17.48 18.35 17.95 16.57];
%Rispondere alle due domande seguenti (a) Determinare la media e la deviazione
% standard e riportarla con la stessa precisione con cui sono forniti i dati
media=mean(w);
deviazione_standard= std(w);
%COME SCRIVERE SOLAMENTE UN DETERMINATO NUMERO DI CIFRE
disp(round(media,2)) %il numero dopo indica il numero di cifre decimali che
%voglio dopo la virgola
disp(round(deviazione_standard,2))

%(b) Considerare la tabella che riporta il valore t* che definisce un intervallo
%di confidenza del 95% relativo alla distribuzione di Student con un numero
%di gradi libertà nu variabile.
variabili=[3 4 5 6 7 8];
t_asterisco=[3.1824 2.7764 2.5706 2.4469 2.3646 2.3060];
%gradi di libertà=variabili N-1
valore_t=2.4469;

%Sulla base del valore corretto di t* ottenuto dalla tabella, determinare
%l'intervallo di confidenza del 95% entro cui ci aspettiamo che cada il valore
%atteso della grandezza fisica. Riportare il risultato nella forma ? = <x> ± ?x
%(il valore medio <x> è stato definito durante il corso come "x soprassegnato (barrato)",
%ma su Forms non è semplice inserire questo carattere)
dx=(deviazione_standard*valore_t)/sqrt(length(w))

%COME TROVARE GLI INTERVALLI DI CONFIDENZA
%come prima cosa una volta fornita la probabilità che vogliamo ottenere, ad
%esempio intervallo di confidenza, allora ricavo la alpha dalla formula P*=1-2alpha
%trovata la alpha i casi si dividono a seconda che la variabile che sto
%studiando abbia distribuzione normale o distribuzione gaussiana.
%CASO DELLA GAUSSIANA, devo trovare la variabile z
    %z_alpha=norminv(alpha)
%CASO DELLA DISTRIBUZIONE DI STUDENT
    %t_alpha=tinv(alpha, nu)
%alpha=(1-P*)/2
```

```
%FUNZIONE NORMCDF

%È nota la distribuzione Gaussiana delle misure di una certa grandezza x,
%identificata dal valore vero ? = 2.5 e dalla deviazione standard ? = 0.5.
%eterminare la probabilità P che il risultato di una successiva misura sia
%compreso tra 2.4 e 2.6. NOTA: si consideri che la funzione che calcola la
%probabilità cumulata di una distribuzione normale esiste in Matlab ed è
%denominata “normcdf”. In questa espressione G(z) è la distribuzione Gaussiana
%standardizzata, ovvero con media nulla e deviazione standard unitaria;
%verificare la possibilità di specificare in normcdf valori diversi di media
%e deviazione standard.
sigma=0.5;
mu=2.5;
P= normcdf(2.6,mu,sigma)-normcdf(2.4,mu,sigma)

%FUNZIONE TCDF
%Sono note 6 misure di una certa grandezza, caratterizzate dal valore medio
%?? = 1.5 e deviazione standard s = 0.2. Determinare la probabilità P
%che il risultato di una successiva misura sia compreso tra 1.4 (xb) e 1.6 (xa).
%come prima cosa calcolo da xa e xb i valori di ta e tb
ta=1.2247;
tb=-1.2247;
P=tcdf(ta,5)-tcdf(tb,5)
```

ESEMPIO UTILIZZO CICLO WHILE

Il ciclo while può essere utilizzato per trovare l'errore che si commette se si considera per via numerica la somma di N termini per approssimare il valore della serie ($\sum_{k=0}^{N-1} q^k$) rispetto al calcolo della serie tramite limite $\frac{1}{1-q}$.

Ad esempio:

```
%scelta del generico valore di q (valore assoluto minore di uno per poter effettuare il calcolo a limite) e del
valore di target desiderato 1.0e-3

se la differenza tra il valore S_current e S_analitico diventa minore di 1.0e-3 si è raggiunta la convergenza
desiderata e si ferma il ciclo. Significa che, per il valore di k che verrà letto grazie al comando disp nella
command window si avrà la differenza tra valore calcolato mediante limite e quello calcolato
algebricamente minore o uguale al valore target scelto. Nel caso in questione la convergenza si otterrà per
un numero di k minore di k_max e pari a 10, quindi al di sotto del valore del k_max. Per k = 10 infatti
S_current pari a 1.9990 e S_analitico a 2.0000, avendo ottenuto precisamente un errore di convergenza
pari al target ossia di 1.0e-3)

q= 0.5;
target_epsilon =1.0e-3;

%inizializzazione delle variabili
```

```
S_analitico=1.0/ (1.0 -q);
k_max = 200;
k=0;
S_current = q^k;
while (abs(S_current- S_analitico) >= target_epsilon);
   k=k+1;

  S_current=S_current + q^k;
   if (k>k_max)
      break;
   end
end
disp([S_current S_analitico k]);
```

%I cicli whiles sono potenzialmente infiniti. Infatti se le ipotesi del ciclo continuano a essere soddisfatte (assumono il valore true) allora il ciclo continua all'infinito. Per interromperlo ed essere sicuri di ovviare al problema di ciclo potenzialmente infinito è necessario inserire al suo interno un ciclo if + break che permette di bloccare il ciclo while prima del tempo (nel caso le ipotesi continuino a essere verificate) fino a un valore massimo a scelta (nel caso in questione stabilito pari a k_max=200. Il comando break in questo caso viene inserito perché, nonostante sia noto che la serie geometrica converge e che quindi a un certo punto si arriverà alla condizione di errore desiderata, questa potrebbe avvenire per un numero elevato di cicli.)

COSTRUTTO WHILE

```
%% CICLO WHILE

while (condizione)
   blocco_istruzioni;
   if (condizione_secondaria)
      break;
   end
end
 %finchè la prima condizione viene verificata si continua a eseguire il
 %primo blocco di codice. nella sezione di codice è possibile ricorrere
 %all'utilizzo della funzione break per uscire prematuramente dal ciclo
 %(prima che la condizione iniziale risulti falsa). tale condizione di
 %break viene regolata mediante una condizione specificata dall'if.
```

COSTRUTTO ED ESEMPIO IF

```
% COSTRUTTO IF
if (condizione1)
   %blocco1 (esegue ed esce dal costrutto if)
   %questo blocco viene eseguito se condizione1 risulta vera,
   %indipendentemente dalla condizione2
```

```
else if (condizione2)
    %blocco2 (esegue ed esce dal costrutto if)
    %questo viene eseguito se condizione1 è falsa e se condizione2 è
    %vera
  else
    %blocco3 (esegue ed esce dal costrutto if)
    %che viene eseguito se sono false le condizioni 1 e 2

  end
end

%%ESEMPIO, utilizzo del comando if per verificare la correttezza del prodotto tra matrici effettuato non
mediante il comando for ma direttamente mediante funzione matlab
B=magic(3);
C=magic(3)+1; %sto usando un abuso di notazione, sommo a ciascun elemento della matrice 1

A=zeros(3,3);
for i=1:3
  for k=1:3
    for j=1:3
    A(i,k)= A(i,k) + B(i,j)*C(j,k);
    end
  end
end

disp(A-B*C)

x=sum(sum(abs(A-B*C)));

if (x>1.0e-3);
  disp('discrepanza')
else
  disp('risultato corretto')
end
```

COSTRUTTO FOR

```
%il ciclo for esegue per un numero prefissato di volte un corpo di codice.  Il numero di volte che verrà
ripetuto il codice è stabilito determinando il valore della variabile k del ciclo. Prima del ciclo è fondamentale
inizializzare a zero le variabili presenti nel corpo
%inizializzazione
for k=1:10
    %corpo
end
%ESEMPIO, utilizzo del ciclo for come metodo per il calcolo del prodotto tra matrici. È fondamentale
ricordarsi di inizializzare le variabili (in questo caso A) prima di entrare nel ciclo for
B=magic(3);
C=magic(3)+1; %sto usando un abuso di notazione, sommo a ciascun elemento della matrice 1

A=zeros(3,3);
for i=1:3
  for k=1:3
    for j=1:3
    A(i,k)= A(i,k) + B(i,j)*C(j,k);
```

```
    end
  end
end

disp(A-B*C)
```

COSTRUTTO SWITCH

```
%% SWITCH
switch espressione
  case valore1
    blocco1;
  case {valore3, valore4, valore2}
    blocco2;
  otherwise
    blocco3;
end
```

%questo costrutto permette l'esecuzione di blocchi specifici di codice a seconda del valore di una variabile o di una espressione numerica, scritta subito dopo il comando switch. Il costrutto viene letto in questo modo: nel caso in cui vi sia corrispondenza tra il valore assunto dall'espressione e valore1 si procede all'esecuzione esclusivamente del blocco1. Una volta eseguito si passa direttamente al comando end. Nel caso in cui il valore assunto dall'espressione sia uno tra valore2, valore3, valore4 verrà eseguito il blocco2 (una volta eseguito si passa direttamente a end). Infine se nessuna delle precedenti condizioni di corrispondenza tra espressione e valore viene soddisfatta si passa direttamente al blocco3 che segue il comando otherwise.

```
%ESEMPIO
d=floor(3*rand)+1
switch d
  case 1
    disp('vale 1');
  case 2
    disp('vale 2');
  case 3
    disp('vale 3');
end
```