Università degli Studi di Ferrara

Dipartimento di Ingegneria

Corso di Laurea in Ingegneria Civile e Ambientale

LISTATI ED ESERCIZI PER IL CORSO MATLAB

Tutorato di Analisi Matematica 1

A.A. 2018/2019

Dott. Ing. Yuri Taddia

✉ yuri.taddia@unife.it

Versione aggiornata al 28 agosto 2018

Parte I

Installazione di MatLab

In questa breve sezione sono riportate le istruzioni per scaricare ed installare MatLab sul proprio PC fisso o portatile.

Le istruzioni sono le stesse riportate anche sul sito del Dipartimento di Ingegneria dell'Università degli Studi di Ferrara al link https://de.unife.it/it/didattica/servizi-agli-studenti/fornitura-software-matlab-tah.

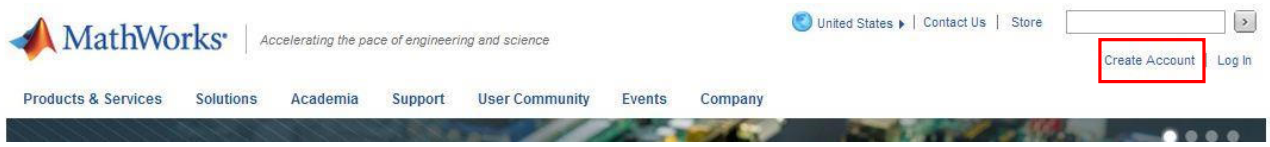
Ogni studente dovrà registrarsi **solo ed esclusivamente con l'indirizzo email di ateneo** (nome.cognome@student.unife.it) affinché la procedura di attivazione possa andare a buon fine.

In caso di problemi con l'installazione si prega di contattare direttamente i tecnici del laboratorio di informatica.

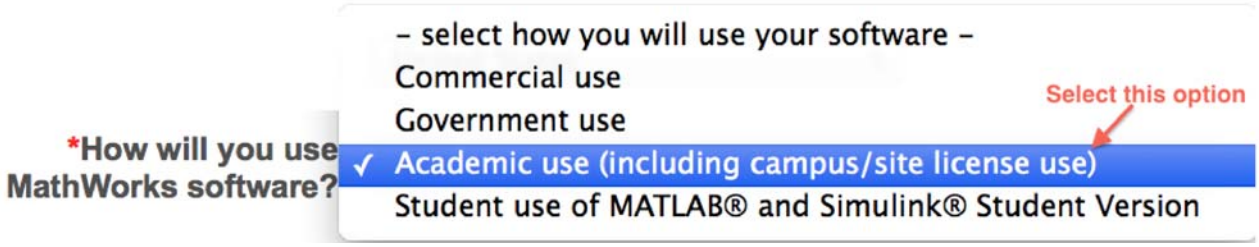
How to install and activate your personal copy of MATLAB (Student Copy)

MathWorks Account Creation

- Go to http://www.mathworks.com
- Create a new MathWorks account



- You should use your university email (`unife.it`) and select the option "Academic use"



Associate with the license

- As soon as you are logged in your MathWorks account click on "Manage Licenses" and then on the button "Add License":

- Select "Activation Key" and click Continue
- Enter the Activation Key:
- `38126-92284-76765-40650-43058`
  and click Continue.

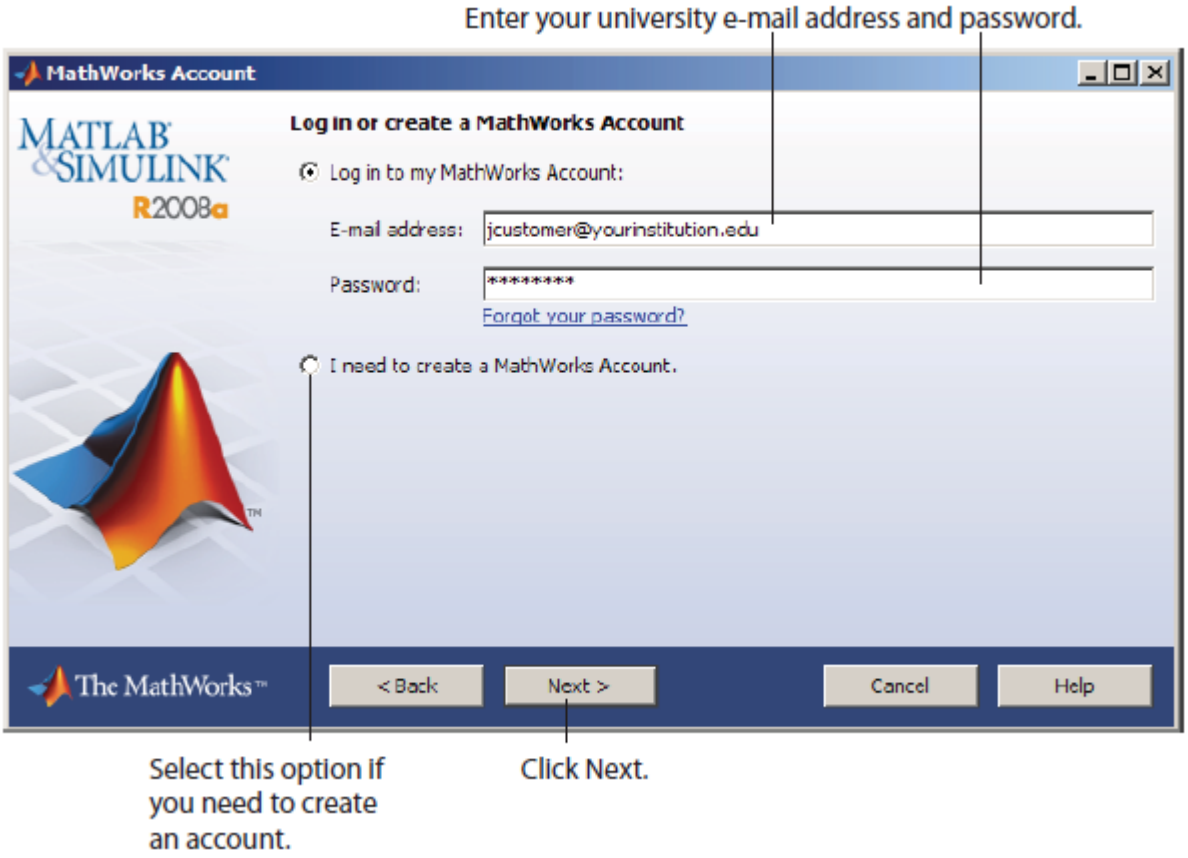
Download the products

- On your account page you can click on "Get Licensed Product and Updates" to download the installation files.

Install

- Select the option: "Install automatically using the Internet"

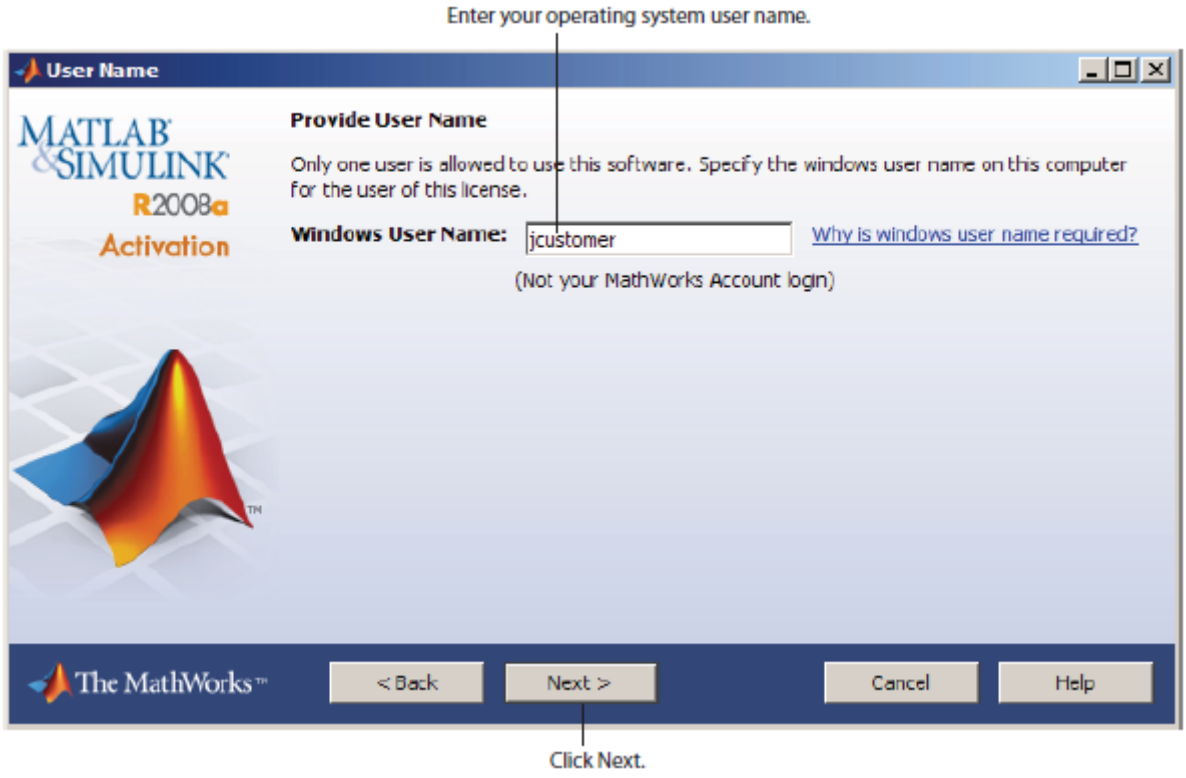
- Review the License Agreement and click next.
- Login to your MathWorks Account



- Select the license … and click “Next”.
- Select the Typical installation, specify your installation folder and start the installation procedure.

Activate

- At the end of the installation process proceed with activation.
- Select license … and Click “Next”
- Enter your Activation Key:
- 38126-92284-76765-40650-43058
  And click “Next”
- Specify the user name that you use to login into your PC.



- Click Next and then Activate.

If you need installation help please open a Service Request by clicking on “Create new Request” on your MathWorks Account. Fill in a Technical Support -> Installation request to receive direct assistance by MathWorks Installation Support Team.

ADDITIONAL RESOURCES:

- Online Documentation:
  http://www.mathworks.com/help/
- MATLAB Answers:
  http://www.mathworks.com/matlabcentral/answers/
- TAH Resource Kit:
  http://www.mathworks.com/academia/tah-support-program/campus.html

Parte II

Listati delle lezioni

In questa sezione sono riportati i listati visti a lezione e forniti agli studenti durante le lezioni stesse.

Non ci sono figure, ma sono presenti tutti i comandi necessari per ottenerle.

Per gli approfondimenti si rimanda:

- alla dispensa completa scaricabile al seguente link http://www.unife.it/ing/civile/insegnamenti/analisi-matematica-I/materiale-didattico;
- agli esercizi proposti nella parte III;
- ai libri di introduzione a MatLab, alcuni dei quali sono disponibili nella biblioteca scientifico-tecnologica.

```
% ========================================================================== %
%% CAPITOLO 1: INTRODUZIONE A MATLAB
% ========================================================================== %

% Tutte le istruzioni di questo capitolo sono da eseguire direttamente a COMMAND
% WINDOW (tranne le parti dopo il simbolo %) per cominciare a familiarizzare con
% alcuni semplici comandi in MATLAB.

% -------------------------------------------------------------------------- %
%% LE QUATTRO OPERAZIONI
% Cominciamo con qualcosa di facile.

2+3         % il risultato dell'operazione è assegnato alla variabile predefinita ans
2*3         % il valore della variabile ans è sovrascritto
2/3
3 + 4       % gli spazi non contano (ma a volte sì...)
a = 2       % assegniamo un valore ad una variabile
A = 4       % Matlab è un programma "case sensitive"
b = a*2
c = a+b;    % notazione ";" , memorizza senza mostrare il risultato
c           % visualizziamo il valore della variabile

% Le variabili create vengono memorizzate da Matlab all'interno del workspace, il cui
% contenuto può essere visualizzato a cw con il comando:

whos

% Ricordiamo che la divisione per zero non ha senso algebrico; Matlab ci avvisa di
% questo facendo comparire l'espressione Inf (Infinity) se il dividendo è diverso da
% zero (è un po' come se facesse il limite) e NaN (Not a Number) se il dividendo è
% zero.

7/0 % Inf
0/0 % NaN
0/7 %0, ovviamente

% -------------------------------------------------------------------------- %
%% PULIZIA
% Le variabili che abbiamo definito sono mostrate nella finestra workspace. Per
% cancellare le variabili dal workspace, chiudere le figure (quest'ultimo sarà utile
% in seguito) e ripulire la cw, si usano i comandi:

clear       % per pulire il workspace ...
clf         % ... chiudere la figura corrente ... ( CLear current Figure )
close all   % ... chiudere TUTTE le figure ...
clc         % ... e ripulire la cw ( CLear Command window )

% -------------------------------------------------------------------------- %
%% MATRICI
% Introduciamo ora alcune semplici operazioni con le matrici, sempre da effettuarsi a
% cw.

a = [0 1 2 3]       % un vettore riga ...
size(a)             % size: la sua dimensione
max(a)              % l'elemento massimo di a
min(a)              % l'elemento minimo di a
b = [7;5;3;1]       % un vettore colonna
length(b)           % length: il numero di elementi di un vettore
max(b)              % max e min funzionano anche per i vettori colonna
A = [1 2 3; 4 5 6]  % la matrice A
size(A)             % la sua dimensione
A'                  % la matrice trasposta di A
z = A(2,1)          % estraiamo un elemento dalla matrice A...
b(3,1) = z          % ... e lo sostituiamo nel vettore b

% Si noti la sintassi degli ultimi due comandi: ciò che si definisce va a sinistra
% del segno di uguaglianza mentre la quantità nota va a destra.
```

```
% ------------------------------------------------------------------------------ %
%% VETTORI

v = [1:10]
w = [10:10:100]
z = [7: -2:1]
x = linspace(0,1,5)
c = [a x]           % concatenazione di vettori
d = [a';x']
e = [b',z]
A = [a;b']          % creiamo una matrice concatenando vettori
D = A(:,1)          % estraiamo la prima colonna di A
D = A(1:1,:)        % estraiamo la prima riga di A ( altra notazione )
D = A(:,1:2)        % estraiamo le prime due colonne di A
D = A(:,1:3)        % estraiamo le prime tre colonne di A
B = A(:,1:end-1)    % sintassi equivalente : uso la parola chiave "end"
E = A(:,2:4)        % estraiamo le ultime tre colonne di A
F = A(1:2,2:end)    % estraiamo una sottomatrice di A

% ------------------------------------------------------------------------------ %
%% OPERATORI RELAZIONALI
% Può essere talvolta utile confrontare due matrici A e B (aventi le stesse
% dimensioni!) per vedere se esse hanno gli stessi elementi. Per questo Matlab usa
% l'operatore relazionale == che crea una matrice della stessa dimensione delle
% matrici A e B i cui elementi sono 1 o 0, a seconda che gli elementi delle matrici A
% e B coincidano o meno.

D == B % infatti le due matrici coincidono !

% Altri operatori relazionali sono i seguenti: == (uguale), ~= (diverso),
% > (maggiore), < (minore), >= (maggiore o uguale), <= (minore o uguale).
% Facciamo un semplice esempio del loro uso:

X = [5 6 7 8]
Y = [5 4 8 10]
X <= Y % operatore relazionale
Z = [5;4;8;10]

% Provare: X <= Z... le matrici devono avere le stesse dimensioni!

% ------------------------------------------------------------------------------ %
%% OPERAZIONI CON MATRICI

B+E     % somma algebrica
B.*E    % prodotto
3*B     % prodotto di uno scalare per una matrice
3.*B
B./E    % divisione
B.^2    % elevamento a potenza

% ------------------------------------------------------------------------------ %
%% MATRICI SPECIALI
% Ecco infine alcune matrici un po' speciali:

C = ones(2,3)    % matrice (2X3) di soli uno
Z = zeros(2,3)   % matrice (2X3) di zeri
R = rand(3,4)    % matrice (3X4) di numeri casuali

% Il comando rand crea una matrice di numeri casuali compresi tra 0 e 1. Provare a
% ripetere più di una volta il comando rand a riga di comando.
```

```
% ========================================================================== %
%% CAPITOLO 2: GLI SCRIPT E LA GRAFICA IN MATLAB
% ========================================================================== %

% Ogni esempio andrebbe scritto all'interno di uno script file diverso.

% REGOLE PER IL NOME DI UNO SCRIPT FILE:
% 1) Deve iniziare con una lettera e non deve contenere il trattino (-) né spazi;
% 2) Non deve avere lo stesso nome di variabili, comandi, funzioni di MATLAB.

% È buona norma anteporre sempre le tre istruzioni seguenti (clc, clear, close all):

clc          % pulisce la command window
clear        % cancella tutte le variabili dal workspace
close all    % chiude tutte le figure eventualmente aperte

%% Esempio 1
x = linspace(-2,2);
plot(x,x.^2)       % plottiamo f(x)=x.^2

%% Esempio 2
x = -2:0.5:2;
plot(x,x.^2)       % plottiamo ancora f(x)=x.^2, ma con meno punti...

%% Un po' di documentazione...
help graph2d       % un po' di aiuto per i grafici 2D...
doc linespec       % un po' di documentazione per personalizzare l'aspetto dei plot

%% Esempio 3
x = -2:0.5:2;
plot(x,x.^2,'or')  % plottiamo ancora f(x)=x.^2, ma con pallini rossi

%% Esempio 4
x = linspace(-2,2);
plot(x,x.^2,'r--')                      % grafico in rosso , tratteggiato
xlabel('x')                             % il nome dell' asse x
ylabel('y')                             % il nome dell' asse y
grid on                                 % la griglia
title('Grafico della funzione x^2')     % il titolo della figura

%% Esempio 5
x = linspace ( -2 ,2);
plot(x,x.^2,'linewidth',4)          % linewidth : spessore di linea
hold on                             % "congelo" il grafico, altrimenti lo sovrascrivo!
plot(0,0,'or','markersize',14)      % markersize : dimensione del pallino
xlabel('x','fontsize',12)           % fontsize : dimensione del carattere
ylabel('y','fontsize',12)
```

```
% ========================================================================== %
%% CAPITOLO 3: SUCCESSIONI
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% LA SUCCESSIONE GEOMETRICA {q^n}
% - se |q| < 1  , allora  q^n --> 0;      % - se  q  = 1  , allora  q^n = 1;
% - se  q  > 1  , allora  q^n --> +inf;   % - se  q <= -1 , allora  q^n è indefinito;

% Verifico il caso |q| < 1:
n  = 0:20;  % definisco il vettore n (con il ';' nascondo l'output a cw)
q  = 3/4;   % assegno un valore < 1 alla variabile q
an = q.^n;  % calcolo gli an (punto per il calcolo componente per componente)

% visualizzo il risultato nella figura 1, uso dei cerchi rossi
figure(1)
plot(n,an,'or')                % plot della successione
hold on; grid on               % congelo il grafico e disegno la griglia
plot([0 20],[0 0],'b')         % disegno il limite in blu
axis([-1 21 -0.2 1.2])         % specifico i limiti degli assi [xmin xmax ymin ymax]
title('La successione geometrica')  % metto un titolo

%+--------------------------------------+
    % Un comando alternativo: SCATTER

    figure(2)
    scatter(n,an,'r')

    figure(3)
    scatter(n,an,20,'r','filled')
%+--------------------------------------+

% -------------------------------------------------------------------------- %
%% SUCCESSIONI DEFINITE PER RICORRENZA: il ciclo FOR
% Considero la successione  a(n+1) = 1/(1+a(n))
n = 1:10;      % scelgo il numero di passi da compiere
a(1) = 1;      % assegno il primo elemento della successione

% uso il ciclo for per il calcolo degli a_i
for i = 2:length(n)       % per "i" che va da 1 al numero di elementi di "n"
    a(i) = 1/(1+a(i-1));  % scrivo la formula dell'i-esimo valore di "a"
end

% visualizzo il vettore "a" costruito nel ciclo con pallini blu
figure(4), plot(n,a,'bo')

% NB: l'utilizzo del comando length permette di cambiare il numero di elementi della
% successione cambiando un singolo elemento del codice!!

% All'interno del ciclo for il vettore "a" viene sottolineato. Matlab vi segnala un
% problema (un "warning"): il codice potrebbe essere più efficiente se il vettore "a"
% avesse una dimensione nota a priori.
% Meglio allora preallocare a (è una questione di memoria allocata...)

% Riscrivo lo stesso codice dichiarando il vettore "a" per verificare l'assenza di
% warning.

clear a n   % cancello le variabili a, n dal workspace. Se faccio un nuovo script in
            % un nuovo file oppure modifico direttamente quello che ho scritto prima,
            % non ho bisogno di usare clear.

n = 1:10;                   % scelgo il numero di passi da compiere
a = zeros(1,length(n));     % creo un vettore non n zeri
a(1) = 1;                   % assegno il primo elemento della successione

% inizio il ciclo for per il calcolo degli a_i
for i = 2:length(n)         % per "i" che va da 1 al numero di elementi di "n"
    a(i) = 1/(1+a(i-1));    % scrivo la formula dell'i-esimo valore di "a"
end
```

```
% visualizzo "a" in fig. 4 e disegno il limite (utilizzo un plot multiplo)
figure(4), hold on
plot(n,a,'g.',...   % ridisegno il vettore a con punti verdi (vado a capo con ...)
[n(1) n(end)],[(sqrt(5)-1)/2,(sqrt(5)-1)/2],'r--')  % disegno il limite in rosso
                                                    % tratteggiato
title('Una successione definita per ricorrenza','fontsize',14)

% ------------------------------------------------------------------------------- %
%% LA SUCCESSIONE CHE HA PER LIMITE e {(1+1/n)^n con n-->Inf}

clc, clear, close all

n = 1:100;
bn = (1+1./n).^n ;

figure(5)
plot(n,bn,'*b')
title('Successione con limite e')

% Disegno il limite
e = exp(1);
hold on
plot([0 n(end)],[e e],'r')
text(85,2.74,'Limite = e')  % scrivo il valore del limite con il comando text,
                            % specificando la posizione x,y del testo e il suo
                            % contenuto mediante una stringa

% ------------------------------------------------------------------------------- %
%% SUCCESSIONI INFINITE E INFINITESIME DI ORDINE DIVERSO

clc, clear, close all

% Inizio con lo studio di successioni INFINITE di ordine diverso.
n = 1:20;
an = log(n);
bn = sqrt(n);
cn = n.^2;
dn = 2.^n;

figure(6)
% Voglio creare più plot in una stessa finestra grafica (stessa "figure")
subplot(1,2,1)  % comincio con il primo, ma devo subito decidere come suddividere
                % lo spazio nella finestra grafica (help subplot...)
plot(n,an,'sr',n,bn,'ob',n,cn,'*k',n,dn,'^g')
grid on
legend('a_n','b_n','c_n','d_n') % inserisco la legenda, ne specifico le singole voci
title('Confronto tra successioni infinite')
ylim([0 10]); % impongo un limite solo all'asse y

% Ora passo allo studio di successioni INFINITESIME di ordine diverso.
% Calcolo nuove successioni costituite da elementi che sono semplicemente
% i reciproci di quelle precedenti:
An = 1./an;
Bn = 1./bn;
Cn = 1./cn;
Dn = 1./dn;

subplot(1,2,2) % voglio cambiare grafico, passando ad un secondo
plot(n,An,'sr',n,Bn,'ob',n,Cn,'*k',n,Dn,'^g') % faccio un plot multiplo
grid on
legend('A_n','B_n','C_n','D_n') % inserisco la legenda, ne specifico le singole voci
title('Confronto tra successioni infinitesime')
```

```
% ========================================================================== %
%% CAPITOLO 4: SERIE
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% RICHIAMO DEI COMANDI MAX, MIN e ABS: vediamo l'HELP di Matlab
% ... se ho un vettore come trovo il massimo valore che contiene?
% ... e se invece ho una matrice??? (eseguire la riga seguente e rispondere)
help max; help min; help abs;

% RISPOSTE: il massimo valore di un vettore v è dato da max(v), mentre il massimo
% valore di una matrice A è max(max(A)), infatti il solo comando max(A) restituisce
% un vettore riga contenente il valore massimo di ciascuna colonna. Analogamente per
% il comando min.

% Esempio pratico:
% --> definisco un vettore "x" e calcolo max(x), min(x) e abs(x)
x = [-1 2 3 -4]
max(x), min(x), abs(x)
% --> definisco una matrice "A" e calcolo max(A) e max(max(A))
A = [7 4 3;8 5 6;1 2 9]
max(A)          % max(A) mi restituisce il vettore [8 5 9] costituito dagli elementi
                % massimi di ciascuna colonna!...
max(max(A))     % ... mentre max(max(A)) grazie al doppio uso di max mi restituisce
                % il massimo del vettore max(A)=[8 5 9], ovvero max(max(A))=9, che
                % effettivamente coincide con il massimo elemento della matrice A.

% -------------------------------------------------------------------------- %
%% I COMANDI SUM, CUMSUM, PROD e CUMPROD

% Il comando sum(x) somma semplicemente gli elementi di "x", restituendo uno scalare.
% Il comando cumsum(x) fa invece la somma cumulativa, restituendo un vettore.
% Se assimiliamo x ad una successione (composta da un numero finito di termini),
% sum(x) ne fornisce la somma, mentre cumsum(x) restituisce la successione delle
% somme parziali della successione.
sum(x), cumsum(x)
% ...cosa succede se x è una matrice? leggete l'HELP e provate con x = ones(4)

% In modo analogo, i comandi prod e cumprod permettono di calcolare i fattoriali
n = [1 2 3 4 5 6 7];
cumprod(n)              % se li vogliamo tutti
prod(1:7)               % se ne vogliamo uno
% Più semplicemente esiste:
factorial(7)

% -------------------------------------------------------------------------- %
%% LA SERIE GEOMETRICA: s_n = cumsum(q^n), con n = 0,1,2,3,...,+inf
% - se |q| <  1   , allora  sum(q^n) = 1/(1-q);
% - se  q  >= 1   , allora  sum(q^n) = +inf;
% - se  q  <= -1  , allora  sum(q^n) è indeterminata;
clc, clear, close all

n = 0:20;           % provate a cambiare il valore massimo di n...
q = 4/5;            % e se cambia q, invece?
an = q.^n;
sn = cumsum(an)     % manca il ";" e visualizzo nella cw i valori

% mostro in figura 1 l'andamento della serie
figure(1), plot(n,sn,'or')
grid on                     % personalizzare l'aspetto del grafico
xlabel('n'), ylabel('s_n')  % aiuta a comprenderne il contenuto

% eseguo un controllo sul valore assoluto della variabile "q"
if abs(q) < 1               % istruzione IF: se |q|<1 (controllare help if )
    hold on
    somma = 1/(1-q);        %  --> aggiungo al grafico il limite a cui tende
    plot(n,somma,'b*')      %      la serie geometrica
end
```

```
% ------------------------------------------------------------------------------ %
%% LA SERIE ARMONICA: s_n = cumsum(1./n), con n = 1,2,3,...,+inf
% Come noto, la serie armonica DIVERGE, ovvero per ogni M > 0 esiste un numero N tale
% che s_n >= M per ogni n >= N. Pertanto sum(1./n) --> +inf

clc, clear, close all
spz = 0;                % inizializzo la variabile spz: somma parziale
k = 0;                  % inizializzo il contatore k
M = 8;                  % fisso un numero M a piacere

% Ora vado a cercare N:
while spz < M           % quando spz (somma parziale) supera M il ciclo si ferma
    k = k+1;            % altrimenti , incrementiamo k di 1...
    spz = 1/k + spz ;   %... e addizioniamo un termine alla somma parziale spz
end
disp(['Il numero N dei termini è: ',num2str(k)])
disp(['La somma parziale dei primi N termini è: ',num2str(spz)])

% Il comando disp serve a visualizzare (diplay) a command window una STRINGA di
% output: è possibile concatenare più stringhe creando un vettore con [str1 str2] i
% cui elementi siano necessariamente stringhe di testo. A tale scopo è utile il
% comando num2str che converte valori numerici in stringhe di testo.

% ------------------------------------------------------------------------------ %
%% SERIE A TERMINI DI SEGNO VARIABILE
% Studiamo ora una serie a termini di segno alterno (la più semplice...):
% s_n=cumsum((-1).^n./n), con n = 1,2,3,...,+inf

clc, clear, close all
n = 1:50;
an = (-1).^n./n;
sn = cumsum(an);

% Plottiamo la successione an:
subplot(1,2,1)
plot(n,an,'^-b'), grid on, xlabel('n'), ylabel('a_n')

% Plottiamo la successione delle somme parziali sn:
subplot(1,2,2)
plot(n,sn,'r'), grid on, xlabel('n'), ylabel('s_n')

% Tale serie converge, anche se non assolutamente...
% Per avere una stima del limite, andiamo a valutare la semisomma L degli ultimi due
% termini della successione delle somme parziali a patto che la loro differenza d non
% sia superiore ad una certa soglia (altrimenti sarà necessario aumentare n oppure
% usare un ciclo WHILE per stimare il limite...):
d = sn(end)-sn(end-1);
L = (sn(end)+sn(end-1))/2;
if d < 0.1
    disp(['Il limite approssimato è: ',num2str(L)])
    gtext(['Il limite è circa ',num2str(L,2)]) % quale differenza con text?
end

% ------------------------------------------------------------------------------ %
%% LA SERIE ARMONICA GENERALIZZATA: s_n=cumsum(1./(n.^alfa)), con n = 1,2,3,...,+inf
% - se alfa  >  1  -->  CONVERGE
% - se alfa <=  1  -->  DIVERGE

clc, clear, close all
n = 1:50;
alfa = double(input('Digitare un valore di alfa: '));
sn = cumsum(1./(n.^alfa));
plot(n,sn,'.r')
grid on
xlabel('n')

if alfa > 1
    title(['La serie converge: \alpha = ',num2str(alfa)])
else
    title(['La serie diverge: \alpha = ',num2str(alfa)])
end
```

```
% ========================================================================== %
%% CAPITOLO 5: GRAFICI DI FUNZIONE
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% LA FUNZIONE SIN(x)./x
% Come noto, la funzione sin(x)./x non è definita per x=0, anche se il limite per
% x-->0 è 1.
% Vediamo cosa succede se facciamo il plot con MatLab:
x = -30:0.01:30; % attenzione!: lo zero fa parte del dominio
y = sin(x)./x;
figure(1)
plot(x,y)
grid on
% Zoomando molto nel punto (0,1) si vede un "buco" nel grafico, ovvero una vera e
% propria discontinuità... Cerchiamo di vedere che valore ha nel vettore delle y
% l'elemento y(x=0):
j = find(x==0); % cerco l'indice dell'elemento 0 nel vettore x, ovvero per
                % quale j è vero che x(j)=0
y(j) % visualizzo il valore che matlab ha calcolato per "sin(0)./0"
     % Ovviamente ottengo un "Not a Number", con cui MatLab mi avverte che
     % il risultato è una operazione matematicamente indefinita (0/0 in
     % questo caso). NaN è uno standard internazionale, digitare help nan...

% -------------------------------------------------------------------------- %
%% LA FUNZIONE TAN(x)
% La funzione tan(x) non è definita per x = pi/2 + k*pi.
% In questo caso, tuttavia, il limite sinistro e quello destro non coincidono e sono
% rispettivamente uguali a +Inf e -Inf.
% Vediamo come si comporta MatLab:
x = 0:pi/180:pi; % attenzione! pi/2 fa parte del dominio...
k_colon = find(x==pi/2)          % cerchiamo i valori dell'indice j per cui
                                 % l'elemento x(j) è uguale a pi/2
x(k_colon)=NaN;   % sostituisco il valore pi/2 con "Not a Number"
% In questo modo non verrà disegnato nulla in corrispondenza di
% x=pi/2 e ci sarà una discontinuità nel grafico.
figure(2)
subplot(1,2,1)
plot(x ,tan(x))
axis([0 pi -50 50])
x = linspace(0,pi); % attenzione!: lo zero NON fa parte del dominio
k_linspace = find(x==pi/2)       % cerchiamo i valori dell'indice j per cui
                                 % l'elemento x(j) è uguale a pi/2 (nessuno!)
subplot(1,2,2)
plot(x,tan(x))
axis([0 pi -50 50])

% Prestare dunque la massima attenzione ai punti che non appartengono al dominio:
% MATLAB potrebbe infatti creare linee indesiderate.

% -------------------------------------------------------------------------- %
%% SIMMETRIA RISPETTO ALL'ASSE y (e rispetto all'asse x e all'origine)
% Vediamo ora come "specchiare" il grafico di una funzione rispetto all'asse delle
% ordinate.
% Banalmente (come visto a lezione...) si tratta di plottare sia f(x) che f(-x);
% quest'ultima rappresenta infatti la "riflessione" di f(x) rispetto all'asse y.

x = linspace(-3,3);
% Prendiamo, ad esempio, la funzione e^x
y = exp(x);
% Disegniamo in nero f(x) e in rosso f(-x)
figure(3)
plot(x,y,'k',-x,y,'r')
% potrei anche fare così: plot (x,y,'k',x,exp(-x),'r')
grid on, legend('f(x)','f(-x)')

% E se volessi fare la simmetria rispetto all'asse x? Cosa dovrei scrivere?
figure(4)
plot(x,y,'k',x,-y,'r')
```

```
% potrei anche fare così: plot (x,y,'k',x,-exp(x),'r')
grid on, legend('f(x)','-f(x)')

% E se volessi fare la simmetria rispetto all'origine degli assi?
figure(5)
plot(x,y,'k',-x,-y,'r')
% potrei anche fare così: plot (x,y,'k',x,-exp(-x),'r')
grid on, legend('f(x)','-f(-x)')

% ------------------------------------------------------------------------------- %
%% IL VALORE ASSOLUTO
% Proviamo a disegnare il grafico del modulo di una funzione.
% Prendiamo, ad esempio, il polinomio x*(x - 1)*(x + 1)*(x - 2)*(x + 3/2)
x = -2:0.01:2;
p = x.*(x-1).*(x+1).*(x-2).*(x+3/2); % un polinomio, p

% Plottiamo in una stessa figura 2 grafici:
% Nel primo disegniamo il polinomio p...
figure(6)
subplot(1,2,1)
plot(x,p,'k')
grid on
xlabel('x') % titolo asse x
ylabel('p') % titolo asse y
axis([min(x) max(x) min(p) max(abs(p))]) % il comando axis imposta i limiti
title('Il polinomio p')

% ... e nel secondo il suo modulo.
subplot(1,2,2)
plot(x,abs(p),'r')
xlabel('x')
ylabel('|p|')
grid on
axis([min(x) max(x) min(p) max(abs(p))])
title('La funzione |p|') % ovviamente non è più un polinomio...

% ------------------------------------------------------------------------------- %
%% FUNZIONI DEFINITE A TRATTI
% Molto spesso risulta comodo definire una funzione a tratti... vediamo come fare
% utilizzando MatLab.

% PRIMO METODO: Separando il dominio. Il più breve.
x = -2:0.01:0;
figure(7)
subplot(2,2,1)
plot(x,2*x)
hold on
x = 0:0.01:2;
plot(x,x.^2)
title('1° Metodo')

% SECONDO METODO: Usare le concatenazioni. Per avere un solo dominio.
x1 = -2:0.01:0; % dominio della prima funzione
y1 = 2*x1;      % prima funzione
x2 = 0:0.01:2;  % dominio della seconda funzione
y2 = x2.^2;     % seconda funzione
x = [x1 x2];    % concateniamo i domini
y = [y1 y2];    % concateniamo le funzioni
subplot(2,2,2)
plot(x,y)
title('2° Metodo')
% NOTA: sarebbe più corretto scrivere x=[x1 x2(2:end)]; y=[y1 y2(2:end)];

% TERZO METODO: Una combinazione dei comandi FOR e IF (lentissimo...!!!)
x = -2:0.01:2;
for i = 1:length(x)
    if x(i) <= 0
        w(i) = 2*x(i);
    else
        w(i) = (x(i)).^2;
    end
end
```

```
subplot(2,2,3)
plot(x,w)
title('3° Metodo')

% QUARTO METODO: Gli operatori relazionali, che al solito restituiscono il valore 1
% se la relazione è vera oppure 0 se è falsa. Il più logico.
x = -2:0.01:2;
z = 2*x.*(x>=-2 & x<=0)+ x.^2.*(x>0 & x<=2);
subplot(2,2,4)
plot(x,z)
title('4° Metodo')

% ------------------------------------------------------------------------- %
%% FARE PIU' GRAFICI
% Per fare più grafici è possibile:

% 1) Usare un ciclo FOR
x = linspace(0,1);
figure(8)
subplot(1,2,1)
for k = 1:10
    plot(x,x.^k)
    hold on
end

% 2) utilizzare il comando MESHGRID
x = 0:0.1:1;        % vettore delle x
n = [1 2 3 4 5];    % vettore delle n

[X,N] = meshgrid(x,n);
% Il comando meshgrid genera due matrici: una matrice X che ha tante righe quanti
% sono gli elementi del vettore n e ogni riga di X è uguale a x ed una matrice N che
% ha tante colonne quanti sono gli elementi del vettore x e ogni colonna di N è
% uguale a n.
% Per convincersene, visualizzare X e N:
X, N

subplot(1,2,2)
plot(x,X.^N) %(x,X^1) (x,X^2) ...
grid on, legend('n=1','n=2','n=3','n=4','n=5','location','northwest')

% ------------------------------------------------------------------------- %
%% CERCARE MINIMI E MASSIMI

% I comandi min e max possono essere usati anche per trovare i minimi e i massimi di
% una funzione, mentre il comando find è utile per trovare i punti di minimo e quelli
% di massimo.

x = linspace(0,3);
f = x.*exp(-2*x);
figure(9)
plot(x,f)
grid on

M = max(f)       % massimo, valore a cw;  %%%oppure metto ; e aggiungo display (M)%%%
i = find(f==M); % indice del massimo
xM = x(i)        % punto di massimo, valore a cw

% Ma attenzione!...
clc
x = linspace(-2*pi,2*pi,100);
f = cos(x);
plot(x,f)
grid on
axis([min(x) max(x) -1 1])
m = min(f)           % minimo, il valore a cw
i = find(f==m);      % indice del minimo
xm = x(i)            % punto di minimo, valore a cw

% Così trovo solo UN punto di minimo, anche se in realtà sono 2...
% Provare con x = linspace(-2*pi,2*pi,1000) e vedere cosa succede.
```

```
% ========================================================================== %
%% COMPLEMENTI AL CAPITOLO 5: TRASLAZIONI E RISCALAMENTI
% ========================================================================== %

% -------------------------------------------------------------------------- %
% COMANDI NUOVI IN QUESTA SEZIONE:

% --> Istruzione IF sia con ELSEIF che con ELSE (già introdotta nel Cap. 5);
% --> il carattere & come operatore logico AND (vedere help AND);
% --> ERROR: come far apparire un errore personalizzato a command window;
% -------------------------------------------------------------------------- %

% Inizializzo il codice di calcolo:
clc, clear, close all

% Vediamo come cambia il grafico della gaussiana f(x)=e^(-x^2) se sostituiamo alla
% variabile x rispettivamente le variabili (x-a) e k*x, ovvero se operiamo una
% traslazione orizzontale ed un riscalamento.
% Ovviamente se a>0 il grafico di f(x) verrà traslato verso destra, mentre se a<0 la
% traslazione avverrà verso sinistra. Analogamente se k>1 si avrà una compressione
% orizzontale, mentre se 0<k<1 si otterrà una dilatazione.

x = linspace(-3,3,1000);
y = exp(-x.^2); % una gaussiana

% Una traslazione orizzontale...
a = input('Digitare il valore di a: ');     % input : per assegnare il valore
                                            % alla variabile da cw
z = exp(-(x-a).^2);
if a > 0
    s = 'Traslazione a destra' ; % definiamo una stringa di testo
elseif a < 0
    s = 'Traslazione a sinistra' ;
else
    s = 'Nessuna traslazione' ;
end

% ... e un riscalamento.
k = input ('Digitare il valore di k: ');    % input : per assegnare un valore
                                            % alla variabile k da cw
w = exp(-(x.*k).^2);
if k > 1
    t = 'Compressione orizzontale' ;
elseif k == 1
    t = 'Nessun riscalamento' ;
elseif k > 0 & k < 1 % prendiamo k > 0 ... usiamo & come operatore logico AND
    t = 'Dilatazione orizzontale' ;
else
    error ('k deve essere maggiore di 0!') % error : errori a cw
end

% GRAFICI:
figure('name','Operazioni con le funzioni') % name : nome finestra

subplot(3,1,1) % subplot: grafico 1...
plot(x,y,'k')
title('e^{-x^2}')
grid on

subplot(3,1,2) % ... grafico 2...
plot(x,z,'r')
grid on
title([s,': a = ',num2str(a)])

subplot(3,1,3) % ... e grafico 3.
plot(x,w,'b')
grid on
title([t,': k = ',num2str(k)])
```

```
% ========================================================================== %
%% CAPITOLO 6: GRAFICI DI FUNZIONE: APPLICAZIONI
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% SCALE LOGARITMICHE E SEMILOGARITMICHE

% Talvolta, in alcune applicazioni ingegneristiche, può essere utile rappresentare
% grafici con gli assi in scala logaritmica (base 10), ovvero avere
% 10^0,10^1,10^2,10^3,... al posto dei valori 0,1,2,3,...
% Ciò risulta particolarmente utile quando i valori delle grandezze da graficare
% abbiano un range estremamente ampio e/o qualora si vogliano mettere in evidenza
% certi comportamenti (nelle distribuzioni dei dati, nelle leggi fisiche che li
% governano,...) che altrimenti non sarebbe possibile apprezzare.

% In MatLab esistono i comandi loglog, semilogx, semilogy:
help loglog, help semilogx, help semilogy;

% -------------------------------------------------------------------------- %
%% LE POTENZE IN SCALA LOGARITMICA
% Vediamo cosa succede se facciamo il plot con loglog di una potenza:

x = linspace(0,3);
y1=x.^2; y2=x.^3; y3=x.^4;

figure(1)
subplot(1,2,1)
plot(x,y1,'b',x,y2,'g',x,y3,'r')
grid on
axis([min(x) max(x) 0 max(y3)])
title('Potenze in scala lineare')
legend('x^2','x^3','x^4','location','northwest')

subplot(1,2,2)
loglog(x,y1,'b',x,y2,'g',x,y3,'r')
grid on % si noti com'è la griglia in scala logaritmica!
axis([min(x) max(x) 0 max(y3)])
title('Potenze in scala logaritmica')
legend('x^2','x^3','x^4','location','northwest')

% Il grafico di una qualsiasi funzione potenza con entrambi gli assi in scala
% logaritmica è una RETTA. Il coefficiente angolare di tale retta è direttamente
% correlato all'esponente della potenza.

% -------------------------------------------------------------------------- %
%% FUNZIONI ESPONENZIALI IN SCALA y-SEMILOGARITMICA
% Vediamo cosa succede se facciamo il plot con semilogy di un fz. esponenziale:

x = linspace(0,3);
y1=2.^x; y2=3.^x; y3=4.^x;

figure(2)
subplot(1,2,1)
plot(x,y1,'b',x,y2,'g',x,y3,'r')
grid on
axis([min(x) max(x) 0 max(y3)])
title('Esponenziali in scala lineare')
legend('2^x','3^x','4^x','location','northwest')

subplot(1,2,2)
semilogy(x,y1,'b',x,y2,'g',x,y3,'r')
grid on % si noti com'è la griglia in scala semilogaritmica!
axis([min(x) max(x) 0 max(y3)])
title('Esponenziali in scala y-semilogaritmica')
legend('2^x','3^x','4^x','location','northwest')

% Il grafico di una qualsiasi funzione esponenziale con l'asse delle ordinate in
% scala y-semilogaritmica è una RETTA. Il coefficiente angolare di tale retta è
% direttamente correlato alla base della funzione esponenziale.
```

```
% ========================================================================== %
%% CAPITOLO 7: POLINOMI
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% I POLINOMI IN MATLAB

% In MatLab esistono molti comandi (funzioni) dedicati ai polinomi:
help polyfun
% Noi ci concentreremo solo su alcuni...

% -------------------------------------------------------------------------- %
%% I COMANDI POLYVAL, ROOTS e POLY

% POLYVAL serve a valutare un polinomio in una serie di punti che viene specificata
% attraverso un vettore x.

% In MatLab un polinomio va specificato SEMPRE indicandone TUTTI i coefficienti,
% partendo da quello di grado massimo fino al termine noto (se una potenza di x non
% compare, allora il suo coefficiente è zero!!!)

% ROOTS invece serve a ricavare le radici del polinomio, ovvero quei valori di x per
% cui p(x)=0.

% POLY infine serve a ricavare i coefficienti del polinomio, note le sue radici.

x = -2:0.1:2;        % punti su cui valutare il polinomio
p = [1 2 -1 -2];     % vettore dei coefficienti del polinomio
y = polyval(p,x);    % polyval
figure(1)
plot(x,y)
hold on
r = roots(p)         % radici, a cw
plot(r,0,'or')
grid on
poly(r)              % poly ritorna i coefficienti del polinomio

% -------------------------------------------------------------------------- %
%% I COMANDI CONV e DECONV

% Per effettuare il prodotto e la divisione tra polinomi, esistono CONV e DECONV.

% Consideriamo i polinomi A(x)=x^3 e B(x)=x^2+1
A = [1 0 0 0];     % A=x^3
B = [1 0 1];       % B=x^2+1

% CONV che esegue il prodotto, la sintassi è conv(p1,p2) dove p1 e p2 sono i vettori
% dei coefficienti dei due polinomi dei quali si vuol calcolare il prodotto.
C = conv(A,B)      % il risultato sono i coefficienti C=[1 0 1 0 0 0] ovvero x^5+x^3

% DECONV che esegue la divisione, la sintassi è deconv(p1,p2) dove p1 e p2 sono i
% vettori dei coefficienti dei due polinomi dei quali si vuol calcolare il quoziente,
% in modo che p1=Q*p2+R.
[Q,R] = deconv(A,B)    % A = Q*B + R; si ottiene Q=x e R=-x

% -------------------------------------------------------------------------- %
%% IL COMANDO POLYFIT

% POLYFIT serve ad ottenere quel polinomio, di un determinato grado, che meglio
% interpola una serie di dati assegnata.

X = [0 1 2 3];       % ascisse dei punti da approssimare
Y = [0 -1 0 1];      % ordinate dei punti da approssimare
P = polyfit(X,Y,3)   % coefficienti del polinomio a cw
x = linspace(0,3);
p = polyval(P,x);    % creiamo il polinomio con polyval
figure(2)
plot(x,p,X,Y,'or')   % cerchietti rossi nei punti da approssimare
grid on
```

```
% ========================================================================== %
%% CAPITOLO 8: FUNCTIONs
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% LA MIA PRIMA FUNCTION
% Iniziamo definendo la funzione y=exp(-x.^2)+exp(-(x+3).^2)+exp(-(x-3).^2) in un
% file che chiameremo fork.m attraverso una sintassi molto particolare, tipica di
% ogni FUNCTION.
% Adesso richiamiamo la funzione appena definita all'interno di uno script file e ne
% facciamo il grafico:
x = linspace(-6,6,100);
figure(1), plot(x,fork(x))
% Digitiamo help fork:
help fork

% -------------------------------------------------------------------------- %
%% CALCOLARE QUOZIENTE E RESTO DELLA DIVISIONE TRA DUE NUMERI INTERI POSITIVI
% Iniziamo definendo la function divis in un file che chiameremo divis.m
x = input('Inserire il valore di a : ')
y = input('Inserire il valore di b : ')
[t,u] = divis(x,y);
disp(['Il valore del quoziente è: ',num2str(t)])
disp(['Il valore del resto è: ',num2str(u)])

% -------------------------------------------------------------------------- %
%% ZERI DI UNA FUNZIONE: FZERO
% Passiamo ora allo studio degli zeri di una funzione: in MatLab esiste il comando
% "fzero" che ha varie sintassi. Bisogna sempre specificare un valore iniziale x0 da
% cui MatLab deve partire per trovare lo zero "più vicino".

% 1) fzero(@fun, x0) nel caso di funzioni interne a MatLab (built-in functions):
    fzero(@cos, 1)

% 2) fzero(@(x) fun(x), x0) nel caso di funzioni non interne a MatLab:
    fzero(@(x)exp(x)-2, 1)  % funzione anonima
    % oppure
    fzero('exp(x)-2', 1)     % stringa

% 3) Nel caso di funzioni non interne, ma "complicate", può essere utile definirle
    % attraverso una function da richiamare:
    % Definiamo la function rad nel file rad.m e richiamiamola in uno script:
    x = linspace(0,1);
    r = rad(x);
    figure(2), plot(x,r)     % plotto la funzione rad
    grid on, hold on
    z = fzero(@rad, 1)       % calcolo gli zeri(funziona solo c'è function M-file)
    plot(z,0,'or')           % evidenzio lo zero sul grafico
    fzero('rad', 1)          % alternativa (funziona solo c'è function M-file...)

% FZERO serve anche a risolvere equazioni: f(x)=g(x) <==> f(x)-g(x)=0

% Risolviamo x*exp(x)=1
x = linspace(-1,1,1000);
y = x.*exp(x)-1;
figure(3), plot(x,y), grid on, hold on
z = fzero('x.*exp(x)-1', -0.5) % il valore a cw
plot(z,0,'or')

% Risolviamo log(x)=exp(-x)
x = linspace(0.5,2.5,1000);
y = log(x);
z = exp(-x);
d = y-z ;
figure(4), plot(x,y,'k',x,z,'b',x,d,'r'), grid on, hold on
legend('ln(x)','e^{-x}','ln(x)-e^{-x}','location','southeast'), hold on
int = fzero('log(x)-exp(-x)', 1.5) % il valore a cw
plot(int,log(int),'ok')
```

```
% ------------------------------------------------------------------------------ %
%% MINIMI E MASSIMI DI UNA FUNZIONE: FMINBND
% Passiamo ora ai massimi e minimi di una funzione. La sintassi del comando fminbnd
% (bounded minimization), che consente il calcolo dei punti di minimo locali di una
% funzione simbolica, è analoga a quella del comando fzero.
% Vediamola con un esempio: definiamo innanzitutto la function polinomio.
clc, clear, close all
a = linspace(-2,4); b = polinomio(a);
% La sintassi è [xmin min] = fminbnd(fun,a,b)
[xmin,min] = fminbnd(@polinomio,-2,4)    % valori a cw
% Poiché max(f)=-min(-f(x)), definiamo anche -f(x) nella function polinomiom.
[xmax,max] = fminbnd(@polinomiom,-2,4); % trovo min(-f)
max = -max; xmax, max,                  % calcolo max(f)=-min(-f)
figure(5), plot(a,b,'k',xmin,min,'*r',xmax,max,'*b'), grid on

% Come evitare di creare una function ad hoc? Con "@(x)fun(x)", cioè usando le
% FUNZIONI ANONIME, la cui classe non è né double né sym, ma function_handle:
[xmin min] = fminbnd(@(x)-cos(x),-2*pi,2*pi)
% Nel nostro caso:
f = @polinomio  % creiamo una funzione anonima f "uguale" alla function esterna
                % "polinomio". Nel workspace compare f come function_handle.
a = linspace(-2,4); b = f(a);
% La sintassi è [xmin min] = fminbnd(fun,a,b)
[xmin,min] = fminbnd(f,-2,4)      % stavolta f non è più una stringa!
% Poiché max(f) = -min(-f(x)), definiamo anche -f(x) come funzione anonima:
meno_f = @(x)-f(x)    % nuova funzione anonima...
[xmax,max] = fminbnd(meno_f,-2,4);          % trovo min(-f)
max = -max; xmax, max,                      % calcolo max(f)=-min(-f)
figure(6), plot(a,b,'k',xmin,min,'*r',xmax,max,'*b'), grid on

% ------------------------------------------------------------------------------ %
% ----------------------------- LISTA FUNCTIONs -------------------------------- %

% fork.m
function y = fork(x)
    % FORK: function che, dato un vettore x, restituisce i punti y del grafico (che
    % ricorda una forchetta).
    % x deve essere un vettore di classe double; l'output è un vettore di classe
    % double delle stesse dimensioni di x
    y = exp(-x.^2)+exp(-(x+3).^2)+exp(-(x-3).^2);
end

% divis.m
function [q,r] = divis(a,b)
    if a < b
        error ( 'a deve essere maggiore di b ! ' )
    elseif or(a<=0, b<=0)
        error ( 'a e b devono essere numeri interi e positivi ! ' )
    end
    q = 1; % poiché a >= b
    r = a-b ;
    while r >= b
        q = q+1;
        r = r-b ;
    end
end

% rad.m
function y = rad(x)
    y = 1-x.^(1/2)+x.^(2/3)-2*x.^(3/4);
end

% polinomio.m
function y = polinomio(x)
    p =[1 -3 -11 27 10 -24];
    y = polyval(p,x);
end

% polinomiom.m
function y = polinomiom(x)
    y = -polinomio(x);
end
```

```
% ========================================================================== %
%% CAPITOLO 9: INTRODUZIONE AL CALCOLO SIMBOLICO
% ========================================================================== %

% Tutte le istruzioni di questo capitolo si possono eseguire direttamente a COMMAND
% WINDOW. Innanzitutto inizializzo il codice di calcolo:
clc, clear, close all

% Il Symbolic Toolbox di Matlab definisce gli oggetti simbolici (di classe sym).
% Digitando "help symbolic" si ha un elenco delle funzioni simboliche di Matlab.
% LO SCOPO DEL CALCOLO SIMBOLICO è quello di FARE CALCOLI "ANALITICI" E NON NUMERICI.
% Ad esempio:

pi              % è la variabile numerica "pi greco" = 3.14 (...) approssimato
p = sym('pi')   % creo la variabile simbolica p pari al "vero" numero irrazionale
a = sin(pi/4)   % restituisce un risultato numerico
b = sin(p/4)    % fornisce un risultato simbolico (e quindi analitico)

% con il comando double() posso passare da variabile simbolica a numerica:
c = double(b)
class(b), class(c)  % verifica della trasformazione

% ------------------------------------------------------------------------- %
%% MANIPOLARE ESPRESSIONI

clc, clear, close all
syms x y

% Il comando EXPAND sviluppa potenze, espressioni trigonometriche, esponenziali
% e logaritmiche.
b = (x+y)^3           % definisco un binomio di terzo grado
expand(b)             % espando il binomio

s = sin(x+y)          % analogamente posso usare funzioni trigonometriche
expand(s)             % --> ricavo la forma di addizione per il seno
expand(sin(2*x))      % --> ricavo la formula di duplicazione per il seno

p = exp(x+y)  % sfrutto le proprietà delle potenze.
expand(p)

% Il comando FACTOR fattorizza gli elementi dell'espressione o fattorizza un numero
% (intero non negativo) in numeri primi.
t = x^2+5*x+6         % NB: t è una variabile simbolica
c1 = factor(t)        % --> factor mi restituisce una variabile di tipo sym
class(c1)             % Il risultato è simbolico

c2 = factor(56)       % NB: 56 è un numero di tipo double
prod(c2)              % --> factor mi restituisce una variabile di tipo double
class(c2)             % Il risultato è numerico (double)

% Il comando PRETTY visualizza in maniera più leggibile l'espressione:
m = (x^2-y^2)/(x^4-4*x^2*y^2)
pretty(m)

% Il comando COLLECT raccoglie i coefficienti a fattore.
p = x^2*(3*y-1)+y^2*(x-y)
collect(p,x)    % rispetto a x
collect(p,y)    % rispetto a y

% Il comando SIMPLIFY semplifica le espressioni
simplify(sin(x)^2+cos(x)^2)

% symplify può essere utile anche per le proprietà di funzioni log/exp:
a = sym('14641')    % 14641=11^4
l = log(a)          % log(14641)
simplify(l)         % 4*log(11)

% Il comando SOLVE permette di trovare gli zeri di un'espressione simbolica
syms a b c x
p = a*x^2+b*x+c
solve(p,x) % risolve simbolicamente p=0 rispetto alla variabile "x"
solve(p,a) % risolve simbolicamente p=0 rispetto alla variabile "a"
```

```
% ------------------------------------------------------------------------------ %
%% GRAFICI E LIMITI
clc, clear, close all
% Il plot simbolico è FPLOT, acronimo di "Plot expression or function".
% Di default questo comando rappresenta il grafico nell'intervallo [-5,5].
% Altrimenti devo esplicitare il dominio fplot(fun,[xmin,xmax])
syms a b c x
p = a*x^2+b*x+c
p1 = subs(p,[a b c],[2 4 3]) % sostituisco: a = 2, b = 4, c = 3 --> p1=2*x^2+4*x+3
figure(1), fplot(p1)         % grafico della funzione simbolica p1

syms x          % facciamo un altro esempio
y = sin(x)/x
subs(y,x,[-3/2*pi:pi:3/2*pi]) % calcolo alcuni valori di y(x)...
figure(2), fplot(y)          % ...e plotto y(x)
% NB: la funzione compare come titolo della figura

clf                % pulisco la figura corrente (clear current figure)
fplot(y,[0,2*pi])  % e specifico [xmin,xmax]

% LIMIT fa il limite di una funzione simbolica per x che tende ad un valore
limit(y,x,0) % --> il limite di y=sin(x)/x per x che tende a 0 è 1

% cerchiamo i limiti della funzione f(x) = 1/x
f = 1/x
figure(3), fplot(f)
limit(f,x,0)            % NaN , non esiste !
limit(f,x,0,'left')     % -Inf , limite sinistro
limit(f,x,0,'right')    % Inf , limite destro
limit(f,x,-inf)         % 0

% Esempio: script che disegna il grafico di y=sin(1/(log(x))) e mostra a cw il limite.
syms x
y = sin(1/(log(x)));                % definisco la funzione
figure(4)
fplot(y,[0,3],'r','linewidth',2)    % plotto la funzione simbolica
grid on
l = limit(y,x,0,'right')    % limite destro simbolico
L = double(l)               % limite destro numerico
% visualizzo un messaggio a cw:
disp(['Il limite per x che tende a 0^{+} è:',num2str(L)])  % num2str(l) non va!


% ============================================================================== %
%% COMPLEMENTI AL CAPITOLO 9: CONVERGENZA DELLE SERIE NUMERICHE
% ============================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all
% Per studiare la convergenza delle serie numeriche esiste il comando SYMSUM.
% Cominciamo con la serie geometrica.
syms q n
s = symsum(q^n,n,0,inf)
% Si ottiene: s = piecewise([1 <= q, Inf], [abs(q) < 1, -1/(q - 1)]), dove piecewise
% serve a "spezzare" il dominio e definire una "soluzione a tratti"

% Proviamo ora a sostituire qualche valore di q con il comando SUBS, la cui sintassi
% è subs(s,old,new); subs sostituisce all'interno di s ogni valore "old" con il nuovo
% valore "new", poi valuta l'espressione così ottenuta...vediamolo con un esempio:
subs(s,q,0.5)   % 2, corretto
subs(s,q,2)     % Inf, corretto (diverge...)
subs(s,q,-2)    % NaN, corretto (è indeterminata!)
% Oppure direttamente:
s = symsum(3^n,0,inf)       % Inf, corretto (diverge...)
s = symsum((1/2)^n,0,inf)   % 2, corretto

% Infine vediamo qualche serie armonica generalizzata:
a = symsum(1/n,n,1,inf)         % Inf, è la serie armonica...
r = symsum(1/n^2,n,1,inf)       % pi^2/6... (Analisi II)
s = symsum(1/sqrt(n),n,1,Inf)   % Cosa accade?!
```

```
% ========================================================================== %
%% CAPITOLO 10: CALCOLO DIFFERENZIALE NUMERICO E SIMBOLICO
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
% La derivazione numerica in Matlab è realizzata con la funzione diff(x) che
% calcola le differenze tra valori adiacenti del vettore x, generando un nuovo
% vettore con un valore in meno. ESEMPIO (scrivere a cw):
x = [0 2 6 4], dx = diff(x)

% -------------------------------------------------------------------------- %
%% CALCOLO DELLA DERIVATA NUMERICA DI f(x) = sin(x)
clc, clear
n = 20;                 % n piccolo per vedere la differenza
x = linspace(0,2*pi,n); % vettore di n valori
f = sin(x);             % vettore di n valori
Dfex = cos(x);          % derivata esatta
% calcolo la derivata numerica
dx = diff(x);           % vettore di n-1 valori !
df = diff(f);           % vettore di n-1 valori !
Dfapp = df./dx;         % derivata numerica CHE HA n-1 VALORI!

% QUALI VALORI DI x ASSOCIO ALLA DERIVATA NUMERICA??? PROVIAMO:
Xapp1 = x(1:end-1); Xapp2 = x(2:end);

figure(1)
plot(x,f,'k',x,Dfex,'b'), hold on
plot(Xapp1,Dfapp,'r',Xapp2,Dfapp,'g')
legend('f','Df esatta','Df approx (X1)','Df approx (X2)'), grid on
axis([0 2*pi min(f) max(f)])
title('Grafici di f(x) = sin(x) e della sua derivata')

% calcolo l'errore commesso usando la derivata approssimata (come cambia con n?)
err = max(abs(Dfex(1:end-1) - Dfapp));
disp(['L''errore massimo è: ',num2str(err)])

% -------------------------------------------------------
% IN QUALI PUNTI STO REALMENTE CALCOLANDO LA DERIVATA?
X_medio = (Xapp1+Xapp2)/2; Dfex = cos(X_medio);
err1 = max(abs(Dfex - Dfapp));
disp(['L''errore massimo è: ',num2str(err1)])
figure(1), hold on
plot(X_medio,Dfapp,'co','DisplayName','Punti')

% -------------------------------------------------------------------------- %
%% DERIVATA SECONDA NUMERICA:
d2f_num = diff(f,2);          % vettore di n-2 valori !
dx2_num = dx(1:end-1).^2;     % vettore di n-2 valori !
d2_num = d2f_num./dx2_num;    % vettore di n-2 valori !

figure(2)
plot(x,f,'k',X_medio,Dfapp,'b',x(2:end-1),d2_num,'r')
% Potrei anche usare: plot(x,f,'k',x(1:end-1),Dfapp,'b',x(1:end-2),d2_num,'r') se non
% volessi "centrare" i punti in cui calcolo le derivate numeriche...
legend('f(x) = sin(x)','Derivata prima numerica = cos(x)',...
    'Derivata seconda numerica = -sin(x)'), grid on

% -------------------------------------------------------------------------- %
%% DERIVATA TERZA NUMERICA:
d3f_num = diff(f,3);          % vettore di n-3 valori !
dx3_num = dx(1:end-2).^3;     % vettore di n-3 valori !
d3_num = d3f_num./dx3_num;    % vettore di n-3 valori !

hold on
plot(X_medio(2:end-1),d3_num,'g','DisplayName',...
    'Derivata terza numerica = -cos(x)')
```

```
% ------------------------------------------------------------------------- %
%% CALCOLO DELLA DERIVATA SIMBOLICA DI f(x) = log((x^2+4)/(x^2-4))
syms x
f = log((x^2+4)/(x^2-4));      % definisco la funzione simbolica
disp('Data f: ')               % ...e la stampo a cw
pretty(f)
Df = diff(f);                  % calcolo la sua derivata con il comando diff,
sDf = simplify(Df);            % ...la semplifico
disp('Df è: ')                 % ...e la stampo a cw
pretty(sDf)


% ------------------------------------------------------------------------- %
%% DERIVATA SIMBOLICA CON PIU' VARIABILI f(x,y) = x^y e DERIVATA SECONDA SIMBOLICA
syms x y
f = x^y;                   % definisco la funzione simbolica
disp('Data f: '), pretty(f)
Dfx = diff(f,'x');         % derivo rispetto ad x
sDfx = simplify(Dfx);
disp('La sua derivata rispetto a x è: ')
pretty(sDfx)
Dfy = diff(f,'y');         % derivo rispetto ad y
sDfy = simplify(Dfy);
disp('La sua derivata rispetto a y è: ')
pretty(sDfy)
D2fx = diff(f,'x',2);      % derivata seconda rispetto ad x
sD2fx = simplify(D2fx);
disp('La sua derivata seconda rispetto a x è: ')
pretty(sD2fx)

% ------------------------------------------------------------------------- %
%% DERIVATA SIMBOLICA DI UNA FUNZIONE PARAMETRICA f(x) = a/(5+4*cos(a*x))
syms a x
f = a/(5+4*cos(a*x));
i = input('Inserire un valore numerico per a:'); % inserire a cw il valore di a
y = subs(f,a,i);      % assegno il valore alla variabile simbolica

% rappresento graficamente la funzione
figure('name','Grafico di una funzione simbolica')
fplot(y,[-6,6]), grid on  % grafico della funzione simbolica tra -6 e 6

% calcolo il limite della funzione per x che tende a zero e a +infinito
l_0 = limit(y,x,0); l_inf = limit(y,x,+inf);
disp('Il limite per x che tende a 0 è:')
double(l_0)
disp('Il limite per x che tende a +inf è:')
double(l_inf)

% calcolo la derivata della funzione e la visualizzo a cw
Dy = diff(y);
disp('La derivata prima della funzione f è:')
pretty(Dy)
% rappresento graficamente la funzione:
figure(4),
fplot(Dy,[-6,6]), grid on

% ------------------------------------------------------------------------- %
%% APPROFONDIMENTO SULLA FRAGILITA' DEL CALCOLO SIMBOLICO
% Il simbolico è fragile. Il codice ingenuo lo distrugge:
syms x
e = exp(1)
f = e^(-(x^2))
%... mentre il codice corretto lo preserva
syms x
f = exp(-(x^2))

syms x
e = exp(sym('1'))
f = e^(-(x^2))
```

```
% ========================================================================== %
%% CAPITOLO 11: CALCOLO INTEGRALE NUMERICO E SIMBOLICO
% ========================================================================== %

% Inizializzo il codice di calcolo:
clc, clear, close all

% -------------------------------------------------------------------------- %
%% CALCOLO APPROSSIMATO DELL'INTEGRALE CON "RETTANGOLI"

% Calcoliamo l'area sottesa dal grafico di sin(x) tra 0 e pi:
n = 5
dx = pi/n;
x = 0:dx:pi;                        % attenzione: x quante componenti ha? (n+1)
xm = 0+dx/2:dx:pi-dx/2;             % attenzione: xm quante componenti ha? (n)
y = zeros(1,n);
Area = 0;

for i = 1:n
   y(i) = sin(xm(i));
   Area = Area + y(i)*dx;
end

Area_VERA = 2;
Errore = abs(Area - Area_VERA)
subplot(1,2,1)
bar(xm,y,'r','BarWidth',1), hold on, fplot(@(x)sin(x),[0 pi])
scatter(xm,zeros(size(xm)),'k'), scatter(xm,sin(xm),'b')
title('Stima dell''integrale con rettangoli')
text(pi/2,1.05,['L''errore è ',num2str(Errore,2)],'horizontalalignment','center')

% -------------------------------------------------------------------------- %
%% CALCOLO APPROSSIMATO DELL'INTEGRALE CON "TRAPEZI"

% Calcoliamo l'area sottesa dal grafico di sin(x) tra 0 e pi:
subplot(1,2,2)
y = sin(x);
Area_TRAPZ = trapz(x,y)
Errore = abs(Area_TRAPZ - Area_VERA)
area(x,y,'FaceColor',[0 1 0]), hold on, fplot(@(x)sin(x),[0 pi])

for i = 1:length(x)
    plot([x(i),x(i)],[0,y(i)],'k')
end

scatter(x,zeros(size(x)),'k'), scatter(x,y,'b')
title('Stima dell''integrale con trapezi (TRAPZ)')
text(pi/2,1.05,['L''errore è ',num2str(Errore,2)],'horizontalalignment','center')

% -------------------------------------------------------------------------- %
%% CALCOLO ESATTO (SIMBOLICO) DELL'INTEGRALE

% Calcoliamo l'area sottesa dal grafico di sin(x) tra 0 e pi:
clear
syms x
AREA = int(@(x)sin(x),x,0,pi)

% Calcoliamo l'integrale indefinito (in realtà una primitiva...) di log(x):
syms x, f=log(x), int(f); pretty(ans)

% Calcoliamo l'integrale indefinito (in realtà una primitiva...) di z^2:
syms z, f=z^2, int(f); pretty(ans)

% Fin qui tutto bene... vediamo ora qualcosa di diverso:

% Calcoliamo l'integrale indefinito (in realtà una primitiva...) di e^(-x^2):
syms x, f=exp(-x^2), int(f); pretty(ans)
help erf    % cosa è "erf"?

% Calcoliamo inoltre l'integrale generalizzato di e^(-x^2) tra -Inf e +Inf:
syms x, int(@(x)exp(-x^2),x,-inf,inf); pretty(ans)
```

```
% ------------------------------------------------------------------------------- %
%% FUNZIONE INTEGRALE

% Determiniamo la funzione definita come l'integrale di sin(t) in dt tra 0 e x:
syms x t
F = int(@(t)sin(t),t,0,x)
figure(4)
fplot(@(t)sin(t),'k'), hold on
fplot(F,'b'), ylim([-1 2]), pause(1)
fplot(@(x)-cos(x)+1,'r'), ylim([-1 2])

% ------------------------------------------------------------------------------- %
% Determiniamo la funzione definita come l'integrale di e^(-t^2) in dt tra a=0 e x
% nell'intervallo [0,3].

% Con il calcolo numerico:
a=0; b=3; n=1000;
[F,x] = Fint('exp(-t.^2)',a,b,n);
f = exp(-x.^2);
figure(2)
plot(x,f,'k',x,F,'r','linewidth',2)
legend('f(x) = e^{-x^2}','F = primitiva'), grid on

% Con il calcolo simbolico:
syms t
figure(3)
fplot(exp(-t^2),[0,3],'k'), hold on
fplot(int(exp(-t^2)),[0,3],'r'), ylim([0 1])
legend('f(x) = e^{-x^2}','F = primitiva'), grid on

% ------------------------------------------------------------------------------- %
% ------------------------------ LISTA FUNCTIONs -------------------------------- %

% Fint.m
function [g,x] = Fint(fun,a,b,n)
    fun = fcnchk(fun);  % grazie a fcnckh (function check) basta definire la funzione
                        % fun direttamente tramite una stringa nello script file
                        % all'interno del comando Fint... diversamente non si
                        % potrebbero che usare solo funzioni fun che siano interne a
                        % MatLab (built-in) oppure appositamente definite in un
                        % M-file (le function esterne viste nel capitolo 8)!
    dx = (b-a)/n;
    x = a:dx:b;
    xm = a+dx/2:dx:b-dx/2;
    y = fun(xm);
    g = [0 cumsum(y)*dx];
end
```

Parte III

Esercizi

In questa sezione sono riportati gli esercizi proposti al termine di ciascuna lezione.

Gli asterischi (*, **, ***) indicano il livello di difficoltà di ciascun esercizio.

Le soluzioni vengono fornite, durante il corso, dopo una settimana dall'assegnazione degli esercizi stessi e comprendono non solamente i listati, ma anche i risultati e le figure.

Per una più semplice comprensione i listati sono racchiusi in appositi riquadri con sfondo grigio e sono formattati allo stesso modo che nell'editor di MatLab, colori compresi.

Si raccomanda la massima partecipazione da parte degli studenti, tentando di risolvere almeno qualche esercizio dopo ogni lezione, per acquisire dimestichezza con i principali comandi sin dalle prime lezioni e sfruttare le lezioni successive per eventuali chiarimenti.

Matrici, Script e Grafica

** Esercizio 1.1

Definire le matrici $\mathbf{A} = \begin{bmatrix} 1 & 7 & 4 \\ 3 & 9 & 2 \end{bmatrix}$ e $\mathbf{B} = \begin{bmatrix} 8 & 1 \\ 2 & 5 \\ 0 & 6 \end{bmatrix}$.

Effettuare, se possibile, le seguenti operazioni ($\mathbf{A}^\mathsf{t}$ è $\mathbf{A}$ trasposta):

- $\mathbf{A} + \mathbf{B}$;
- $\mathbf{A}^\mathsf{t} + \mathbf{B}$;
- $\mathbf{A} + \mathbf{B}^\mathsf{t}$;
- definire la matrice $\mathbf{C}$ di dimensioni 3×3, avente come prima riga la seconda riga di $\mathbf{A}$ e come seconda e terza riga le due colonne di $\mathbf{B}$;
- calcolare il prodotto elemento per elemento della prima riga di $\mathbf{A}$ per la seconda colonna di $\mathbf{B}$;
- elevare ogni elemento di $\mathbf{A}$ per il rispettivo elemento di $\mathbf{B}^\mathsf{t}$ (significa calcolare gli elementi $(a_{ij})^{b_{ji}}$);
- creare il vettore $\mathbf{v} = \begin{bmatrix} 2 & 5 & 9 \end{bmatrix}$ ed inserirlo sia in $\mathbf{A}$ che in $\mathbf{B}$, rispettivamente come terza riga e terza colonna;
- estrarre da $\mathbf{A}$ l'elemento $a_{1,2}$ e sostituirlo in $\mathbf{B}$ rimpiazzando l'elemento $b_{3,1}$;
- calcolare infine $\mathbf{D} = \mathbf{A} + \mathbf{B} + \mathbf{C}$ e verificare (con l'operatore `==`) che l'elemento $d_{3,3}$ sia pari al doppio dell'elemento $d_{1,1}$.

** Esercizio 1.2

Generare:

- un vettore riga $\mathbf{v}$ di 20 elementi casuali compresi tra 0 e 7;
- un vettore colonna $\mathbf{w}$ formato da 10 elementi casuali compresi tra -2 e 4;
- un vettore (riga) $\mathbf{a}$ di 50 elementi "linearmente equidistanti gli uni dagli altri" (cioè con passo costante...) in cui il primo termine valga 5 e l'ultimo valga 25; verificare quindi che il vettore contenga effettivamente 50 elementi;
- un vettore (riga) $\mathbf{b}$ il cui primo elemento sia 5, l'ultimo sia 25 ed il passo sia pari a 0.05; verificare anche in questo caso il numero effettivo di elementi di $\mathbf{b}$.

* Esercizio 1.3

Scrivere uno script che disegni il grafico della funzione $y = \sqrt{x}$ nell'intervallo $x \in [0, 10]$.

** Esercizio 1.4

Scrivere uno script che disegni in un unico grafico le funzioni $\sqrt{x},\ x,\ x^2$ nell'intervallo $x \in [0, 2]$. Provare ad utilizzare colori e stili di linea personalizzati per ciascuna funzione.

Rispondere quindi ai seguenti quesiti:

- È sempre vero che $x^2 > x$?
- È sempre vero che $x^2 > \sqrt{x}$?

Successioni e ciclo For

** Esercizio 2.1

Data la successione

$$a_n = \frac{n^\alpha}{k^n}$$

- calcolarne i termini per $n = 0, 1, 2, ..., 10$ con $\alpha = 1, 2, 3$ e $k = 5$, quindi rappresentarli in un unico grafico, con colori diversi al variare di α;
- calcolarne i termini per $n = 0, 1, 2, ..., 10$ con $\alpha = 2$ e $k = 1, 2, 3$, quindi rappresentarli in un unico grafico, con colori diversi al variare di k (restringere l'asse y all'intervallo $[0, 10]$ per non perdere il dettaglio sulle successioni con $k > 1$).

** Esercizio 2.2

Data la successione a_n definita per ricorrenza

$$a_n = \begin{cases} \pi & n = 1 \\ 9 - \sqrt{a_{n-1} - 3} & n \geq 2 \end{cases}$$

scrivere uno script MatLab che ne calcoli i primi 20 termini e disegni il grafico di a_n.

Determinare sia graficamente che analiticamente il valore a cui tende la successione e dire come sarebbe cambiato il limite se al posto di $a_1 = \pi$ fossero stati usati rispettivamente i valori $a_1 = 10$ ed $a_1 = e$.

** Esercizio 2.3

Data la successione a_n definita per ricorrenza

$$a_n = \begin{cases} 2 & n = 1 \\ 1 + \dfrac{1}{a_{n-1}^2 + \sqrt{a_{n-1}}} & n \geq 2 \end{cases}$$

scrivere uno script MatLab che:

- ne calcoli i primi 10 termini;
- visualizzi a *command window* il vettore `an` costituito dai 10 valori così calcolati;
- disegni (usando dei "pallini") il grafico di a_n.

** Esercizio 2.4 (ciclo For)

Disegnare un quadrato "per punti" (usando degli asterischi) utilizzando diverse volte il ciclo `for`.

Suggerimento: si cominci scrivendo ed eseguendo le seguenti righe di codice, poi aggiungere...

```
clc,clear,close all,
n=20;

for i=-n/2:1:n/2        % disegno il primo lato del quadrato...
    x=i;
    y=n/2;
    plot(x,y,'*'), drawnow, hold on,    % drawnow aggiorna istantaneamente
    axis([-n/2-1 n/2+1 -n/2-1 n/2+1]), axis square
    pause(0.2)
end
```

Serie, ciclo While e istruzione If

* Esercizio 3.1

Date le successioni, entrambe *divergenti*,

$$a_n = \log \sqrt{n}, \quad s_n = \sum_{k=1}^{n} \log\Big(\log\big(\log k\big)\Big)$$

scrivere uno script MatLab che, attraverso l'uso di opportuni cicli `while`:

- calcoli per quale $n \in \mathbb{N}$ la successione a_n supera il valore 5 e disegnare il grafico dei primi n termini di a_n;
- calcoli per quale $n \in \mathbb{N}$ la successione s_n delle somme parziali della serie $\displaystyle\sum_{k=1}^{\infty} \log\Big(\log\big(\log k\big)\Big)$ supera il valore 10 e disegnare il grafico dei primi n termini della serie s_n;

* Esercizio 3.2

Data la serie

$$\sum_{k=1}^{\infty} \frac{1}{k^4}$$

scrivere uno script MatLab che:

- plotti i primi 20 termini della successione $a_k = \dfrac{1}{k^4}$;
- plotti i primi 20 termini della successione delle somme parziali della serie $s_n = \displaystyle\sum_{k=1}^{n} \frac{1}{k^4}$;
- fornisca una stima della somma della serie, attraverso l'ultimo elemento della successione s_n.

Confrontare la stima ottenuta con il numero π^4, calcolando il rapporto tra i due. Provare ad aumentare il numero di termini considerati, passando da 20 a 100. Cosa se ne può dedurre? Qual è dunque la somma della serie? (Potrete calcolare voi stessi questo limite sfruttando il calcolo simbolico: lo faremo nell'esercizio 6.6.)

* Esercizio 3.3

Verificare graficamente come la successione s_n delle somme parziali della serie $\displaystyle\sum_{k=0}^{\infty} \frac{1}{k!}$ converga molto più rapidamente ad e rispetto alla successione $a_n = \left(1 + \dfrac{1}{n}\right)^n$, plottando in uno stesso grafico i primi 10 termini sia di a_n che di s_n ed evidenziando con una retta il limite pari ad e.

** Esercizio 3.4

Scrivere uno script MatLab che fornisca una stima della somma della serie

$$\sum_{n=0}^{\infty} \frac{\sqrt{n+1}}{2^n}$$

Poiché la serie converge piuttosto rapidamente, la stima può essere effettuata considerando il k–esimo termine della serie, indicato con s_k, tale che la differenza (in modulo) tra s_k e s_{k-1} sia inferiore ad una prefissata tolleranza T (ovvero deve essere $|s_k - s_{k-1}| < T$). Si assuma per T il valore 10^{-3}.

Visualizzare infine a *command window* la stima della somma e disegnare in un grafico i primi k termini della serie.

** Esercizio 3.5 (Istruzione IF con uso di *elseif* + ciclo FOR e ciclo WHILE)

Si vuole scrivere uno script in grado di rendicontare i risultati di un esame, fornendo il numero di studenti che:

- hanno ottenuto la *lode* (voto > 30);
- sono risultati *sufficienti* (voto ≥ 18);
- sono risultati *insufficienti* ($12 \leq$ voto < 18);
- sono risultati *gravemente insufficienti* (voto < 12);

Generare un vettore **v** contenente 100 valori casuali e interi (usare `randi` oppure `rand` all'interno di `round`) compresi tra 0 e 31, che simulerà i voti.

Impostare un'istruzione `if` all'interno di un ciclo `for` per effettuare un conteggio degli elementi di **v** inizializzando a 0 il valore delle seguenti variabili:

`lodi`, `suff`, `insuff`, `grav_insuff`

Terminato l'esercizio, modificare il listato per usare un ciclo `while` al posto del ciclo `for`.

SUGGERIMENTO: iniziare così...

```
clc,clear,close all
v=randi(31,1,100)'    % randi genera valori interi... help randi
suff=0; insuff=0; ... % inizializzare tutto!
for i=...             % qual è il contatore?!
    if v(i)...        % prima condizione
        suff=...;     % se la prima condizione è verificata, allora...
        lodi=...;
    elseif v(i)...    % e così via...!
        suff=suff+1;
    elseif v(i)...
        ...
    else
        ...
    end
end
% VISUALIZZO i risultati del conteggio:
suff, insuff, grav_insuff, lodi
```

Grafici di funzioni

* Esercizio 4.1

Data la funzione

$$f(x) = \operatorname{cotg}\left(2x + \frac{\pi}{4}\right)$$

disegnarne il grafico nell'intervallo $0 \leq x \leq 2\pi$ evitando l'apparizione di linee verticali "indesiderate" e restringendo gli assi del grafico agli intervalli $0 \leq x \leq 2\pi$ e $-5 \leq y \leq 5$.

SUGGERIMENTO: Essendo richiesto il grafico ristretto a $-5 \leq y \leq 5$, allora se, ad esempio, $|f(x)| > 10$ è lecito porre $f(x) =$ `NaN`.

** Esercizio 4.2

Data la funzione $f(x) = \log\left(\sqrt{2^x + 4} \cdot \left(2 \operatorname{sen} x + 3\right)\right) - 2$, disegnare, nell'intervallo $x \in [-10, 10]$, i grafici di

$$f(x),\ -f(x),\ f(-x),\ -f(-x),\ |f(x)|,\ f(x+2)$$

Disporre i grafici all'interno di una stessa figura usando il comando `subplot`. Mantenere anche il grafico di $f(x)$, con linea rossa tratteggiata, in ognuno dei grafici precedenti (aggiungere `hold on, plot(x,y,'--r')`).

** Esercizio 4.3

Data la funzione $f(x)$ definita a tratti nell'intervallo $x \in [-3, 2\pi]$

$$f(x) = \begin{cases} 2x^2 + 4x & -3 \leq x < 0 \\ \operatorname{sen} x & 0 \leq x \leq 2\pi \end{cases}$$

determinarne il *minimo globale* ed il *punto di minimo globale* attraverso un opportuno script in MATLAB, disegnare quindi il grafico di $f(x)$ ed evidenziare con un "pallino" rosso tale minimo sul grafico stesso. Infine visualizzare a *command window* il messaggio «`Il minimo globale è f(x) = ...  nel punto x = ...`».

** Esercizio 4.4 Applicazione: uso di `semilogx`

L'*analisi granulometrica* consiste nella *vagliatura*, attraverso una serie di appositi setacci con apertura decrescente, di un campione di terreno. Vengono pesati il campione e la frazione trattenuta da ciascun setaccio. Da tale analisi è possibile costruire un grafico, detto *curva granulometrica*, riportante in ascissa l'apertura dei setacci e in ordinata la percentuale di passante: tale grafico è caratterizzato dal fatto di avere le ascisse in scala logaritmica. È inoltre possibile calcolare alcuni parametri, tra i quali i valori D_{50}, D_{10} e D_{60} corrispondenti ai valori di ascissa per i quali la percentuale di passante è rispettivamente uguale a 50%, 10% e 60% e il *coefficiente di uniformità* definito come $U = \dfrac{D_{60}}{D_{10}}$.

Si supponga di dover analizzare i risultati di un'*analisi granulometrica* condotta su un campione di sabbia. Il peso totale del campione analizzato è $P_{tot} = 435$ g; gli altri pesi sono illustrati in tabella 4.1.

Tabella 4.1: Risultati dell'analisi granulometrica

Apertura setaccio [mm]	Peso trattenuto [g]
4.76	2
2	39.3
0.84	100.9
0.42	162.7
0.253	96.2
0.075	27

Si richiede di:

- calcolare la percentuale di passante, come $\%_{passante} = \frac{P_{tot} - P_{trattenuto}}{P_{tot}} \cdot 100$;
- disegnare la curva granulometrica (ascisse in scala logaritmica!);
- valutare (sul grafico, per semplicità) i valori D_{50}, D_{10} e D_{60} e calcolare il *coefficiente di uniformità*.

** Esercizio 4.5 Applicazione: uso di `semilogy`

Un qualunque evento sismico viene spesso associato alla sua *magnitudo*.

Senza entrare nel merito, essa è correlata alla quantità di *energia* liberata durante l'evento stesso.

In tabella 4.2 è fornita tale correlazione "per punti".

Si chiede di plottare due grafici affiancati, con la *magnitudo* in ascissa e l'*energia* in ordinata, utilizzando in quello di sinistra scale lineari per entrambi gli assi, mentre in quello di destra il solo asse y dovrà essere logaritmico.

Cosa se ne deduce? Di che tipo è la relazione tra *energia* e *magnitudo*?

Tabella 4.2: Magnitudo vs Energia.

Magnitudo	**Energia**
0	$63 \cdot 10^3$ J
1	$2 \cdot 10^6$ J
1.5	$11 \cdot 10^6$ J
2	$63 \cdot 10^6$ J
2.5	$355 \cdot 10^6$ J
3	$2 \cdot 10^9$ J
3.5	$11 \cdot 10^9$ J
4	$63 \cdot 10^9$ J
4.5	$355 \cdot 10^9$ J
5	$2 \cdot 10^{12}$ J
5.5	$11 \cdot 10^{12}$ J
6	$63 \cdot 10^{12}$ J
6.5	$354 \cdot 10^{12}$ J
7	$2 \cdot 10^{15}$ J
7.5	$11 \cdot 10^{15}$ J
8	$63 \cdot 10^{15}$ J
8.5	$355 \cdot 10^{15}$ J
9	$2 \cdot 10^{18}$ J
9.5	$11 \cdot 10^{18}$ J
10	$63 \cdot 10^{18}$ J

Polinomi

* Esercizio 5.1

Determinare le radici del polinomio:

$$p(x) = x^5 + 17x^4 - 462x^3 - 230x^2 + 18413x - 42315$$

* Esercizio 5.2

Determinare l'espressione (i coefficienti) del polinomio avente le seguenti radici:

$$x_1 = -3,\ x_2 = -7/3,\ x_3 = -3/5,\ x_4 = 5,\ x_5 = 12$$

** Esercizio 5.3

Una serie di prove sperimentali ha fornito i risultati riportati in tabella 5.1.

Tabella 5.1: Serie sperimentale di dati.

x	14	15	100	16	0	66	63	46	58	96
y	625.51	715.00	30210.26	809.76	10.30	13210.37	12042.84	6449.49	10217.54	27849.86

Si vuol ricercare se esista un polinomio di grado 2 in grado di descrivere il legame tra y e x.

Una volta determinati i coefficienti di tale polinomio, si riportino in un unico grafico sia il polinomio che la serie di dati.

** Esercizio 5.4

Assegnati i polinomi $p_1(x)$, $p_2(x)$, $p_3(x)$, $p_4(x)$ e $p_5(x)$:

- $p_1(x) = x^3 - 2x^2 - 5x + 6$
- $p_2(x) = x^3 + x^2 - 8x - 12$
- $p_3(x) = x^5 - 15x^3 - 10x^2 + 60x + 72$
- $p_4(x) = 4x^3 + 2x^2 - 7x + 5$
- $p_5(x) = x^2 - 3$

determinare i seguenti polinomi $p_6(x)$, $Q_7(x)$, $R_7(x)$, $Q_8(x)$ e $R_8(x)$:

- $p_6(x) = p_1(x) \cdot p_2(x)$
- $Q_7(x)$, $R_7(x)$ rispettivamente *quoziente* e *resto* della divisione $p_3(x) : p_2(x)$
- $Q_8(x)$, $R_8(x)$ rispettivamente *quoziente* e *resto* della divisione $p_4(x) : p_5(x)$

*** Esercizio 5.5 *Earthquake-Resistant Building Design* (*William J. Palm III*)

Questo esempio è tratto da *Introduction to* MatLab *for Engineers (Third Edition)* di *William J. Palm III*. Si vuol semplicemente fornire un'utile applicazione di MatLab ai futuri ingegneri civili, senza entrare nel merito della *dinamica delle strutture*, disciplina complessa e oggetto dei corsi dell'ultimo anno della *laurea magistrale*.

Buildings designed to withstand earthquakes must have natural frequencies of vibration that are not close to the oscillation frequency of the ground motion. A building's natural frequencies are determined primarily by the masses of its floors and by the lateral stiffness of its supporting columns (which act as horizontal springs). We can find these frequencies by solving for the roots of a polynomial called the structure's *characteristic* polynomial. The Figure 5.2 shows the exaggerated motion of the floors of a three-story building. For such a building, if each floor has a mass m and the columns have stiffness k, the polynomial is

$$(\alpha - f^2)[(2\alpha - f^2)^2 - \alpha^2] + \alpha^2 f^2 - 2\alpha^3$$

where $\alpha = \frac{k}{4m\pi^2}$.

The building's natural frequencies in cycles per second are the positive roots of this equation. Find the building's natural frequencies in cycles per second for the case where $m = 1000$ kg and $k = 5 \cdot 10^6$ N/m.

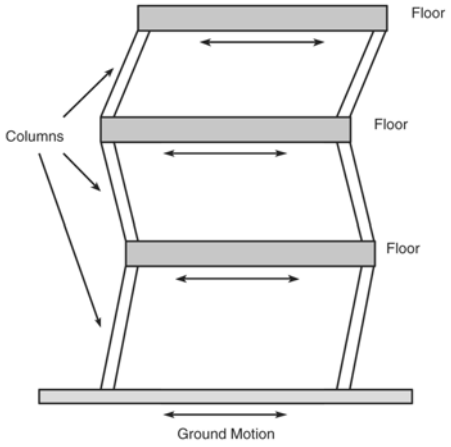


Figura 5.2: Simple vibration model of a building subjected to ground motion.

Soluzione dell'esercizio 5.5

The characteristic polynomial consists of sums and products of lower-degree polynomials. We can use this fact to have MatLab do the algebra for us. The characteristic polynomial has the form

$$p_1(p_2^2 - \alpha^2) + p_3 = 0$$

where

$$p_1 = \alpha - f^2 \; p_2 = 2\alpha - f^2 \; p_3 = \alpha^2 f^2 - 2\alpha^3$$

The MatLab script file is

```
clc, clear, close all,
k = 5e+6;
m = 1000;
alpha = k/(4*m*pi^2);
p1 = [-1,0,alpha];
p2 = [-1,0,2*alpha];
p3 = [alpha^2,0,-2*alpha^3];
p4 = conv(p2,p2)-[0,0,0,0,alpha^2];
p5 = conv(p1,p4);
p6 = p5+[0,0,0,0,p3];
r = roots(p6)
```

The resulting positive roots and thus the frequencies, rounded to the nearest integer, are 20, 14 and 5 Hz.

Calcolo simbolico

* Esercizio 6.1

Scomporre in fattori primi la seguente espressione:

$$x^9 + 9x^7 + 2x^6 + 12x^5 + 24x^4 - 80x^3 + 96x^2 - 192x + 128$$

* Esercizio 6.2

Esplicitare le formule di duplicazione del coseno e della tangente facendo in modo che siano "ben leggibili".

* Esercizio 6.3

Data la seguente espressione:

$$a^3 - a^2b + 4a^2 - 4ab + 4a - 4b$$

raccogliere rispetto alla sola variabile a e rispetto alla sola variabile b, infine scomporla in fattori primi.

* Esercizio 6.4

Risolvere simbolicamente le seguenti equazioni:

- $\text{sen}(2x) = \cos(x)$
- $\sqrt{x} = e^{x-1}$

SUGGERIMENTO: Se $\text{sen}(2x) = \cos(x)$, allora $\text{sen}(2x) - \cos(x) = 0$ ...

* Esercizio 6.5

Disegnare il grafico dell'espressione simbolica $\dfrac{x^2+4}{x-3}$. Impostare l'asse x con valori compresi tra -25 e +25.

** Esercizio 6.6

Ricordate l'esercizio 3.2? Calcolate ora la somma della serie $\displaystyle\sum_{k=1}^{\infty} \frac{1}{k^4}$ con il calcolo simbolico. Cosa ottenete?

** Esercizio 6.7

Calcolare simbolicamente la somma delle seguenti serie: $\displaystyle\sum_{k=0}^{\infty} \frac{3^k}{k!}$, $\displaystyle\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}$

** Esercizio 6.8

Calcolate simbolicamente: $\displaystyle\lim_{x\to+\infty} \left(\sqrt{x^2+2x} - \sqrt[3]{x^3-2x^2}\right)$

** Esercizio 6.9

Calcolate simbolicamente $\displaystyle\lim_{x\to0^-} xe^{-\frac{1}{x}}$ e $\displaystyle\lim_{x\to0^+} xe^{-\frac{1}{x}}$. Dire se esiste $\displaystyle\lim_{x\to0^\pm} xe^{-\frac{1}{x}}$.

Functions

** Esercizio 7.1

In MatLab non esiste un comando per calcolare il logaritmo di un numero reale in una base arbitraria. Definire quindi, tramite una *function*, la funzione `logbase` che restituisca il logaritmo di un numero $x \in \mathbb{R}, x > 0$ in base $b \in \mathbb{R}$, con $b > 0, b \neq 1$, sfruttando la *formula del cambiamento di base*.

Richiamare tale *function* in uno *script* file per calcolare $\log_{11}(161051), \log_7(823543), \log_5(1220703125)$.

** Esercizio 7.2

Calcolare zeri, minimi e massimi della funzione $g(x) = x \cdot \text{sen}(x-1) - \frac{1}{3}$ nell'intervallo [-5,1].

Evidenziarli inoltre sul grafico di $g(x)$ utilizzando "pallini" di colore diverso a seconda che il punto considerato sia uno zero, un minimo o un massimo.

** Esercizio 7.3

Definire tramite una *function* un'unica funzione `cilindro` che restituisca sia il volume che la superficie *totale* (somma dunque della superficie *laterale* e di quella delle due *basi*) di un cilindro di cui siano noti il raggio di base e l'altezza.

Usare quindi tale *function* per calcolare il volume e la superficie totale di un cilindro avente $r = 7\,\text{cm}$ e $h = 23\,\text{cm}$.

*** Esercizio 7.4

Definire due *function* denominate rispettivamente `tempF2C` e `tempC2F` tali che la prima converta temperature da gradi Fahrenheit (°F) a gradi Celsius (°C) e la seconda converta temperature da gradi Celsius (°C) a gradi Fahrenheit (°F).

Testare le due funzioni attraverso uno *script* file che converta i valori 32 °F e 212 °F in gradi Celsius e i valori 0 °C e 100 °C in gradi Fahrenheit.

Si ricorda che valgono le seguenti relazioni:

$$\text{T[°C]} = \frac{5}{9} \cdot (\text{T[°F]} - 32) \quad \rightleftarrows \quad \text{T[°F]} = \frac{9}{5} \cdot \text{T[°C]} + 32$$

*** Esercizio 7.5

Definire tramite una *function* la funzione

$$f(x) = \begin{cases} \log(10 + \sqrt{-x}) \cdot \text{sen}(x) & x < 0 \\ 2\,\text{sen}(x) \cdot e^{-\frac{x}{2}} & x \geq 0 \end{cases}$$

Richiamarla quindi in uno *script* file per:

- disegnarne il grafico limitatamente all'intervallo $x \in [-10, 10]$;
- calcolarne uno zero in prossimità di $x_0 = 3$;
- evidenziare lo zero appena trovato sul grafico di $f(x)$, con un "pallino" rosso;
- calcolarne i minimi e i punti di minimo negli intervalli $-2 < x < -1$ e $4 < x < 5$;
- calcolarne i massimi e i punti di massimo negli intervalli $-6 < x < -4$ e $0 < x < 2$;
- evidenziare tali minimi e massimi sul grafico di $f(x)$.

Calcolo Differenziale

* Esercizio 8.1

Data la funzione

$$f(x) = x^2 \operatorname{sen}(x)$$

calcolare sia la derivata prima *simbolica* che *numerica* (limitarsi per quest'ultima all'intervallo $-10 \leq x \leq 10$).

Sovrapporle quindi all'interno di uno stesso grafico, disegnando la prima in blu e la seconda in rosso, sempre con $x \in [-10, 10]$. Provare a "raffittire" la derivata *numerica* aumentando `n` in `linspace(-10,10,n)`.

** Esercizio 8.2

Data la funzione

$$f(x) = \operatorname{sen}\left(2x - \frac{\pi}{2}\right)$$

calcolare le sue prime tre derivate *simboliche*, quindi riportare sia $f(x)$ che le sue derivate in un unico grafico (simbolico) assegnando un colore diverso a ciascuna delle quattro funzioni (ad esempio $f(x)$ in nero, $Df(x)$ in blu, $D^2f(x)$ in rosso e $D^3f(x)$ in verde).

Inserire una legenda ed un titolo opportuni.

** Esercizio 8.3

Data la funzione

$$g(x) = e^{-x^4}$$

calcolare le sue prime tre derivate *numeriche*, quindi riportare $g(x)$ e le sue derivate rispettivamente in quattro distinti grafici affiancati all'interno di un'unica figura (usare `subplot`). Limitarsi all'intervallo $x \in [-2, 2]$.

Inserire titoli opportuni per ciascun grafico.

** Esercizio 8.4

Avvalendosi esclusivamente del *calcolo simbolico*, data la funzione

$$f(x) = x^a \operatorname{sen}(x)$$

plottare in due figure distinte rispettivamente:

- $f(x)$ per $a = 1, 2, 3, 4, 5, 6$;
- $\dfrac{\mathrm{d}f}{\mathrm{d}x}$ per $a = 1, 2, 3, 4, 5, 6$.

** Esercizio 8.5

Senza avvalersi del *calcolo simbolico*, data la funzione

$$f(x) = x^a \operatorname{sen}(x)$$

plottare in due figure distinte rispettivamente:

- $f(x)$ per $a = 1, 2, 3, 4, 5, 6$ con $x \in [-2\pi, 2\pi]$;
- $\dfrac{\mathrm{d}f}{\mathrm{d}x}$ per $a = 1, 2, 3, 4, 5, 6$ con $x \in [-2\pi, 2\pi]$.

Calcolo Integrale

* Esercizio 9.1

Calcolare sia *numericamente* che *simbolicamente* il seguente integrale definito:

$$\int_0^{\frac{\pi}{2}} \operatorname{sen}(x)\cos(x)\,\mathrm{d}x$$

** Esercizio 9.2

Calcolare una primitiva della funzione:

$$f(x) = \frac{x+4}{x^2+4x+4}$$

Si consideri ora la primitiva $F(x)$ ottenuta ponendo $c = -1$, dove c rappresenta la *costante di integrazione*:

- fare il grafico sia di $f(x)$ che di $F(x)$, usando due colori diversi ed inserire una legenda;
- trovare le coordinate del punto di intersezione (attorno a $x_0 = 3$) tra $f(x)$ e $F(x)$ ed evidenziarlo sul grafico con un "pallino".

** Esercizio 9.3

Determinare, se possibile, le espressioni delle funzioni $F(x)$ e $G(x)$ definite rispettivamente come

$$F(x) = \int_0^x te^{-t^2}\,\mathrm{d}t\ ,\ G(x) = \int_0^x \frac{\operatorname{sen}(t)}{t}\,\mathrm{d}t$$

e disegnarne i grafici.

Calcolare inoltre gli integrali generalizzati

$$\int_0^\infty te^{-t^2}\,\mathrm{d}t\ ,\ \int_{-\infty}^\infty \frac{\operatorname{sen}(t)}{t}\,\mathrm{d}t$$

** Esercizio 9.4

Senza avvalersi del *calcolo simbolico*, calcolare il valore dell'area compresa tra le curve di equazione $f(x) = x^n$ e l'asse delle ascisse nel solo intervallo $x \in [0,1]$ e per $n = 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5$.

Memorizzare i risultati in un vettore `Aree` e plottarne gli elementi in funzione di n, con "pallini".

** Esercizio 9.5

Avvalendosi esclusivamente del *calcolo simbolico*, calcolare il valore dell'area compresa tra le curve di equazione $f(x) = x^n$ e l'asse delle ascisse nel solo intervallo $x \in [0,1]$ e per $n = 1/5, 1/4, 1/3, 1/2, 1, 2, 3, 4, 5$.

Memorizzare i risultati in un vettore `Aree` e plottarne gli elementi in funzione di n, con "pallini".

SUGGERIMENTO: "numericamente" e "senza avvalersi del *calcolo simbolico*" significa usare *vettori* (`linspace`, operazioni con il punto `.* ./ .^`) e `trapz`, mentre "simbolicamente" e "avvalendosi esclusivamente del *calcolo simbolico*" significa usare *variabili simboliche* e/o *funzioni anonime*, `subs` e `int` ...