Calcolo Numerico

A.A. 2014-2015

Lab n. 2

22-10-2014

Grafica: plot 2D

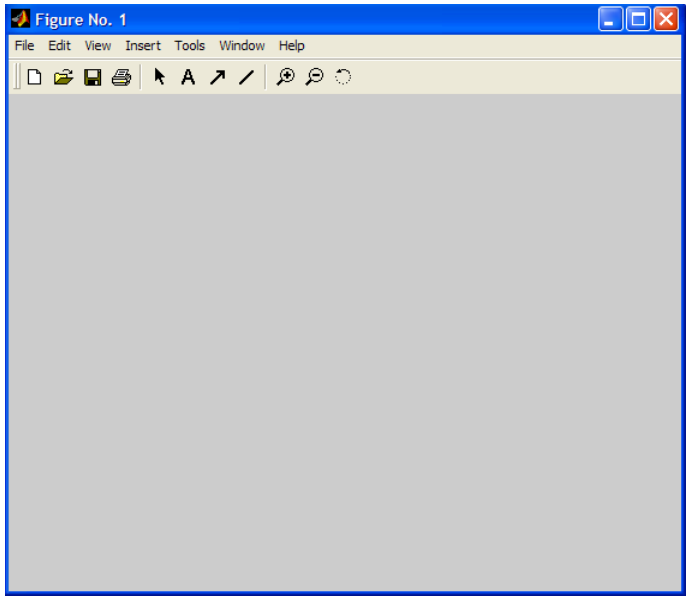
Grafica

- MATLAB è molto più di un semplice software in grado di eseguire calcoli numerici (anche se in forma molto sofisticata). Con MATLAB è possibile realizzare **grafici di funzioni** anche in più dimensioni e realizzare veri e propri programmi. In MATLAB è possibile
  - disegnare funzioni in 2D e 3D
  - rappresentare dati memorizzati in vettori e matrici in molti modi differenti
- Il comando **`plot(x,y)`** si usa
  - per **rappresentare punti nel piano**
  - per disegnare il **grafico di una funzione** **`y=f(x)`**
    - **`x e y`** devono essere vettori di ugual misura

Figure

- **`figure`**
  - apre la finestra grafica al quale viene associato un numero
- **`figure(n)`**
  - **`n`** è il numero associato alla finestra grafica
- **`id = figure`**
  - ad ogni figura è associato un identificativo **`id`** che consente di gestirne le proprietà grafiche

```
>> figure(1)

>> id = figure;
```



Plot - sintassi

```
plot(vettore_x, vettore_y, 'opzioni')
```

- **vettore_x** e **vettore_y** devono avere lo stesso numero di componenti, sono i vettori dei dati (ascisse e ordinate dei punti)
- **opzioni**: è una stringa opzionale che definisce il tipo di colore, di simbolo, di linea usato nel grafico, …

Esempi colore: **m** magenta, **r** rosso, **g** verde, **b** blu, **w** bianco, **k** nero, **y** giallo, **c** ciano

Esempi tipo di linea: **-** continua (default), **--** tratteggiata, **:** punteggiata, **-.** punto-linea

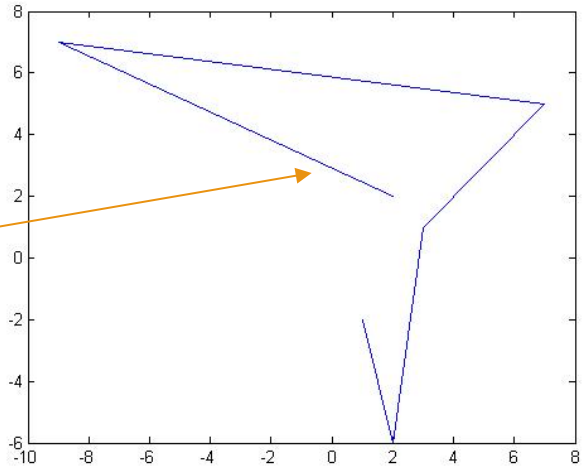
Esempi simbolo: **+** croce, **o** cerchietto, ***** asterisco, **x** ics, **.** punto, **s** quadratino, **d** diamante, **v** triangolo

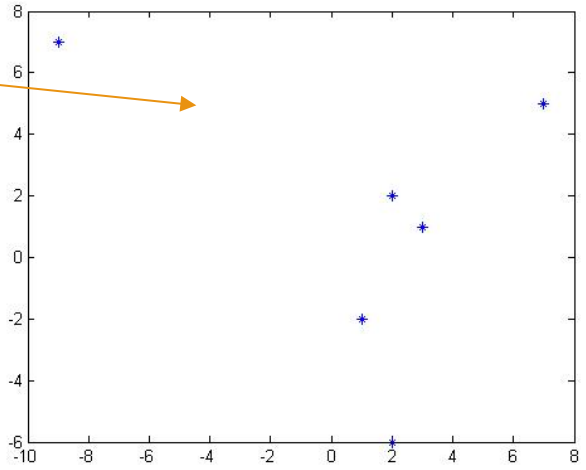
help plot per vedere quali sono le varie opzioni

Rappresentazione dei punti

- Per rappresentare dei punti nel piano

```
>> x = [1 2 3 7 -9 2];
>> y = [-2 -6 1 5 7 2];
>> plot(x,y)
>> figure(2)
>> plot(x,y,'*')

% realizza il grafico del
% vettore y rispetto ai propri
% indici
>> plot(y)
```





Rappresentazione dei punti

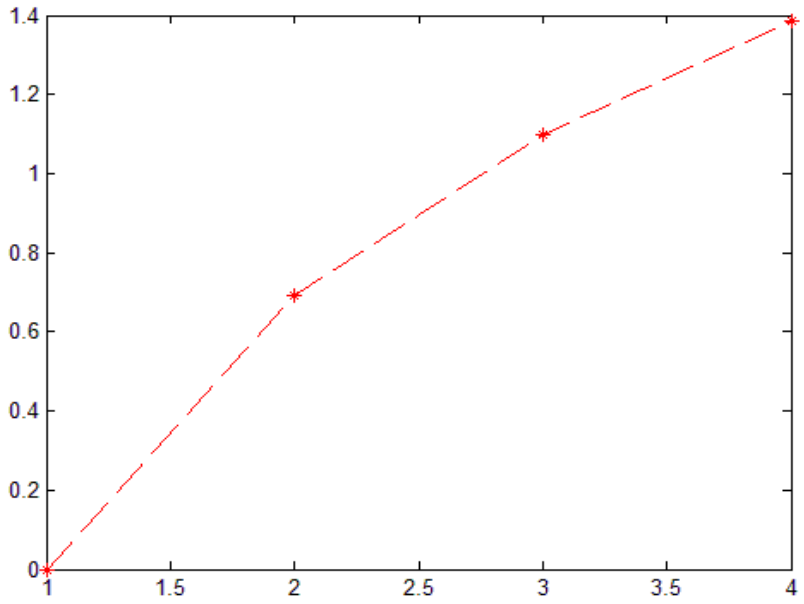
- La funzione **`plot(x,y)`** rappresenta graficamente la tabella di valori ottenuta raccordando con **segmenti di retta** nel piano cartesiano **`Oxy`** i vertici **`(x(i),y(i))`** in modo ordinato al variare di i da **`x(1)`** a **`x(end)`.**

- Se vogliamo evidenziare i **vertici della poligonale** possiamo utilizzare l'istruzione **`plot(x,y,'-g')`** che disegna i punti **`(x(i),y(i))`** in verde. La funzione non cambia, sono i parametri di ingresso che cambiano, il terzo parametro aggiunto serve a specificare lo stile della linea del grafico (pallini in verde)

Grafici di funzione

- Esempio: Siano **`x = [1 2 3 4]`** e **`y = log(x)`**. Disegnare **`y`** in funzione di **`x`** usando una linea tratteggiata rossa e marcando i punti della curva con asterischi
  - si creano due vettori x e y contenenti rispettivamente la successione di valori nell'intervallo ed i corrispondenti valori della funzione

```
>> x = 1:4;
>> y = log(x);
>> plot(x,y, 'r--*')
```

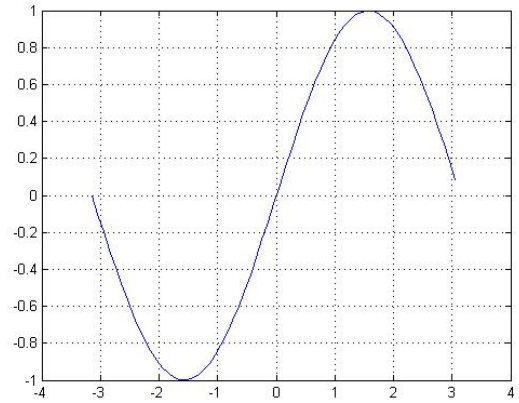


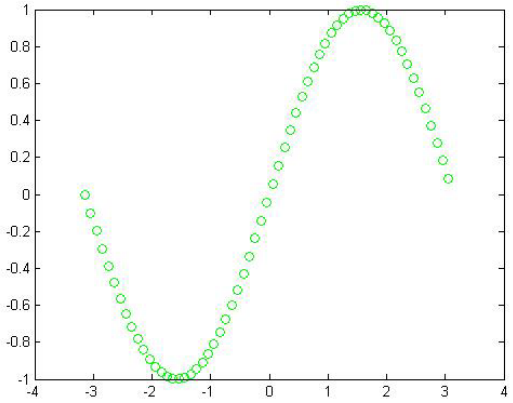
Se volessimo produrre un grafico privo di spigoli è sufficiente aumentare il numero di punti in modo che i segmenti di raccordo siano cosi piccoli da dire l'idea di una linea continua

Esempio

Per plottare la funzione **y=sin(x)**

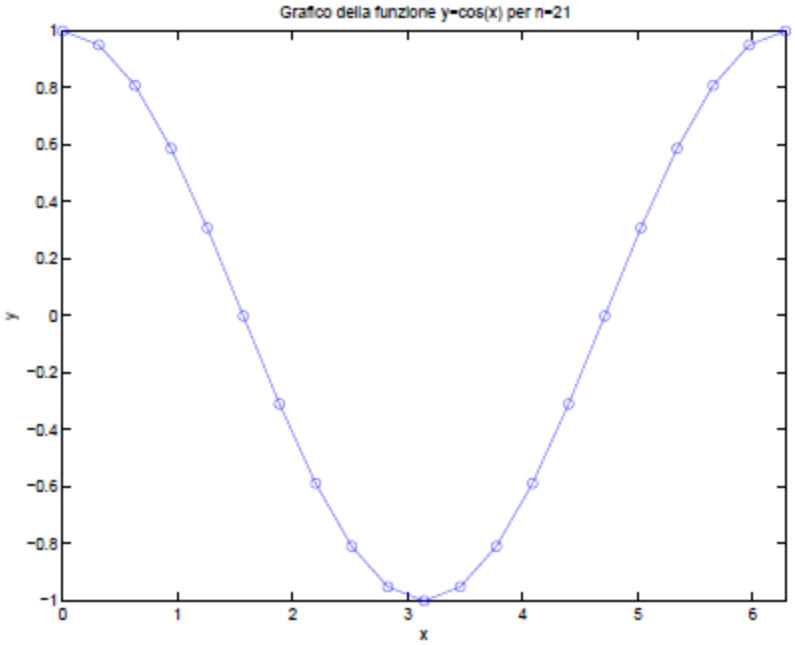
```
% definiamo l'intervallo in cui
% vogliamo disegnare la funzione
x = [-pi:.01:pi];

% definiamo la funzione
y = sin(x);

% disegniamo la funzione
plot(x,y)

% con un terzo parametro di input
figure(2)
plot(x,y, '°g')
```

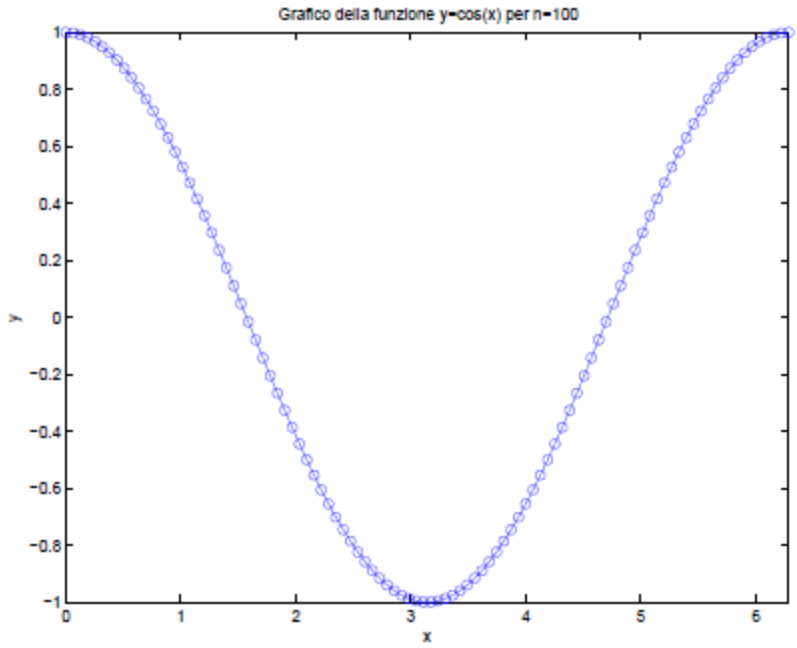




Esempio

Rappresentazione grafica della funzione **`y=cos(x) in [0, 2π]`**





Grafici sovrapposti

- Sia **x** il vettore delle ascisse e **y1 e y2** due vettori aventi la stessa lunghezza di **x**. Per visualizzare sulla **stessa finestra** si usa il comando **hold on** che funziona come un interruttore acceso/spento; tale comando fa sovrapporre tutti i grafici successivi nella stessa finestra grafica fino a quando non si digita il comando **hold off**

```
>> figure
>> plot(x,y1)
>> hold on
>> plot(x,y2)
>> hold off
```

- Equivalentemente si può usare la seguente istruzione che usa in modo automatico linee tipo differenti per i diversi grafici. (le opzioni si possono omettere)

```
>>  plot(x,y1,'opzioni',x,y2,'opzioni')
```

Grafici – etichette

- Esistono molte possibilità per **personalizzare** un grafico

`title('stringa')` titolo del grafico

`xlabel('stringa')` etichetta per l'asse delle ascisse (asse x)

`ylabel('stringa')` etichetta dell'asse delle ordinate (asse y)

`grid` inserisce una griglia nel grafico

`legend('stringa1', 'stringa2',…)` legenda

`axis([xmin xmax ymin ymax])` regola la dimensione degli assi coordinati (determina il rettangolo nel quale si vogliono visualizzare i dati).

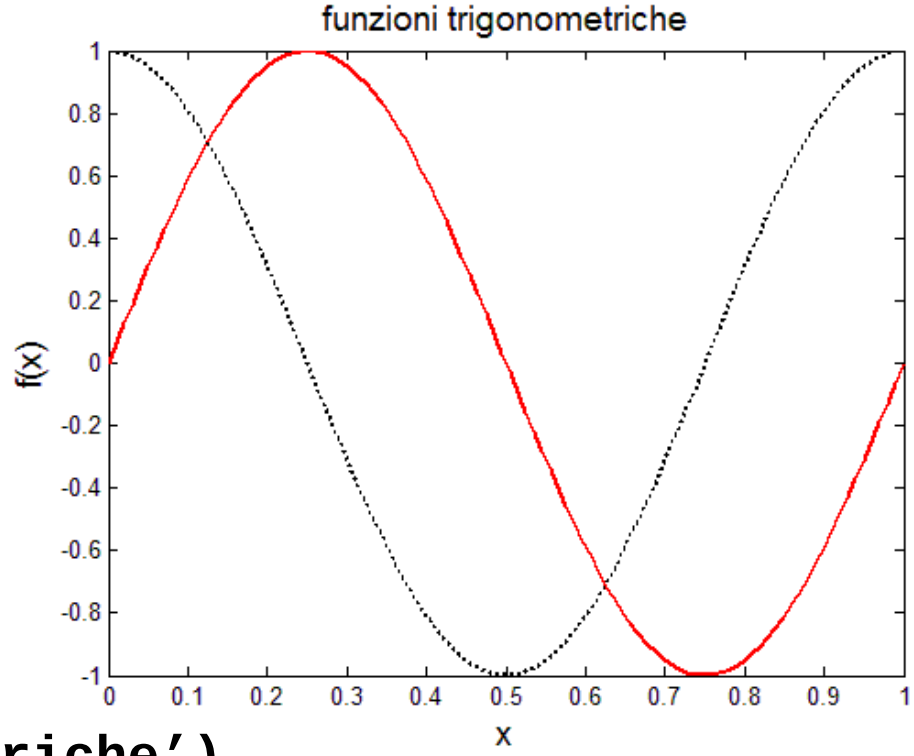
`axis('equal')` usa la stessa scala sulle ascisse e le ordinate

`set, get`

Esempio

Esempio: Sia x un vettore di 10 elementi equidistanti e contenuti nell'intervallo [0,1]. Definire i vettori y = sin(2πx) e z = cos(2πx). Disegnarli sullo stesso grafico usando tratti diversi e colori diversi nel rettangolo [0 1]x[-1 1]. Etichettare gli assi rispettivamente con 'x' e 'f(x)' e dare il seguente titolo 'funzioni trigonometriche'.

```
>> x = linspace(0,1,10);
>> y = sin(2*pi*x);
>> z = cos(2*pi*x);
>> figure
>> plot(x,y,'r',x,z,'b:')
>> axis([0 1 -1 1])
>> xlabel('x')
>> ylabel('f(x)')
>> title('funzioni trigonometriche')
```



Sottografici

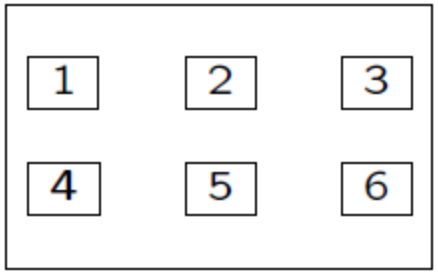
Spesso ci si pone il problema di **disegnare diversi grafici separati in una stessa finestra**. L'obiettivo può essere raggiunto facilmente utilizzando la funzione **`subplot`** la cui sintassi è

```
subplot(Righe, Colonne, Sottofinestra)
```

`subplot(m, n, k)` suddivide la finestra dei grafici in una matrice mxn e attiva la k-ma sottofinestra. Il successivo comando **`plot`** disegna il grafico nella finestra attivata

Esempio: **`subplot(2,3,4)`**

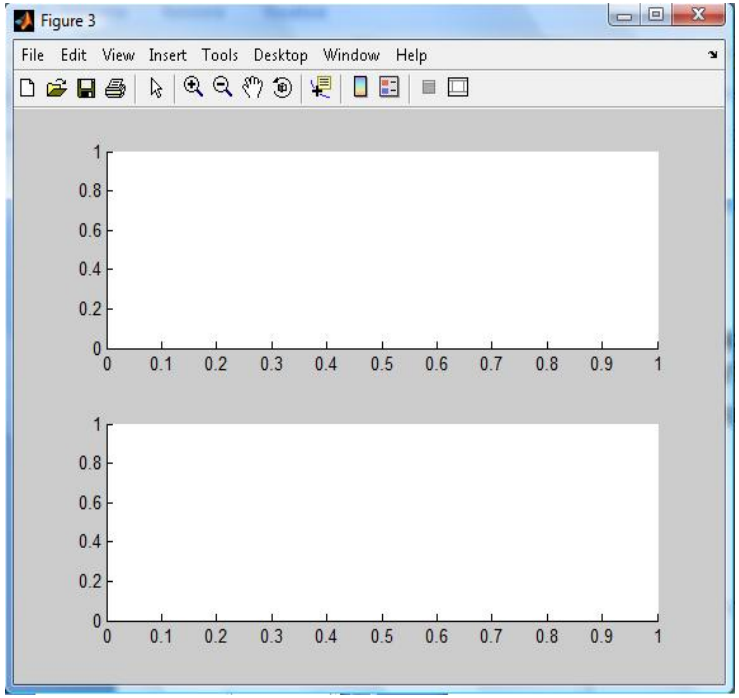
Tale istruzione suddivide la finestra in una matrice 2 × 3 di sottofinestre ed attiva la quarta sottofinestra grafica. Le sottofinestre sono numerate come segue



Sottografici

Esempio

```
>> subplot(2,1,1)
% divide la finestra grafica in 2
% sottofinestre poste una sotto
% l'altra e attiva la prima
>> plot(x1,y1);
>> title('grafico1')
>> subplot(2,1,2);
% attiva la seconda finestra
>>plot(x2,y2); title('grafico2')
```



fplot

```
fplot(funzione,limiti_assi,'opzioni')
```

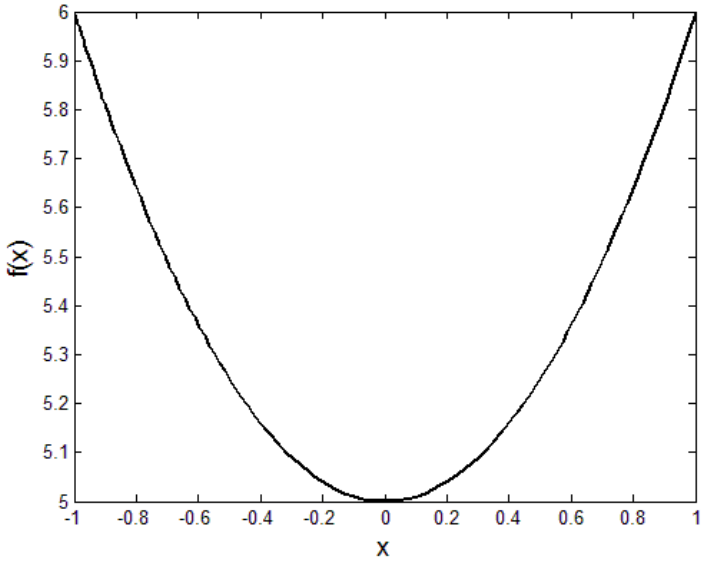
- **`funzione`** puntatore ad una funzione

```
>> f = @(x)[x^2+5]
    f =
        @(x)[x^2+5]
```

- **`limiti_assi : [xmin xmax]`** è un vettore contenente i limiti inf e sup dell'intervallo in cui si vuole visualizzare la funzione;

Esempio:

```
>> fplot(f,[-1 1])
>> xlabel('x'), ylabel('f(x)')
```



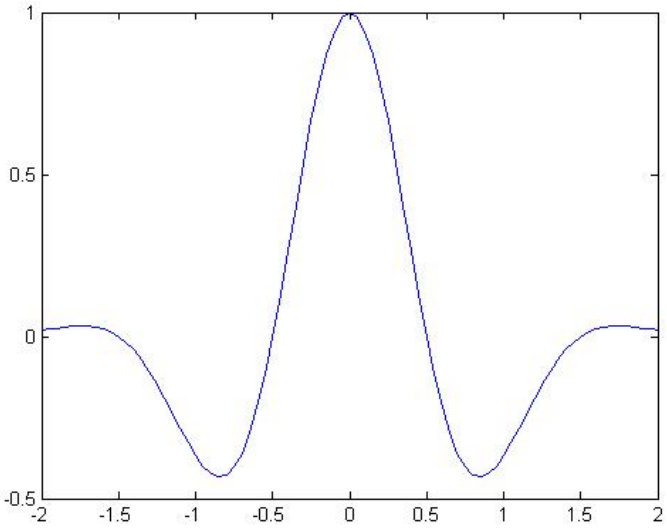
Max

- Trovare il max della funzione

$$f(x) = e^{-x^2}\cos(\pi x)$$

nell'intervallo [-2,2]

```
>> x = [-2:.05:2];
>> y = exp(-x.^2).*cos(pi*x);
>> plot(x,y)
>> [maximo p_max]= max(y);
```



Esercizio

- Consideriamo le due forme equivalenti (da un punto di vista algebrico) di uno stesso polinomio:

$$p(x) = (x-1)^7 =$$
$$= x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$$

  - calcolare numericamente i due polinomi nell'intervallo [0.9998 1.0002]
  - usare la funzione **`fplot`**
  - cosa accade?
  - disegnare i polinomi nella stessa finestra grafica abbellendo i grafici con varie etichette

Esercizi

- Disegnare la parabola $y = -x^2+3x+2$ nell'intervallo [-10,10] e calcolarne le coordinate del vertice

- Disegnare tra 0 e 2π la funzione sin(x) e le sue approssimazioni di Maclaurin del 1° e 5° grado
  - le 3 curve devono essere rispettivamente a tratto continuo, tratteggiata e marcata con cerchietti

- Disegnare un cerchio

- Sia **`x=1:10`**, **`y1=atan(x)`** **`e`** **`y2=log(x)`**. Disegnare y1 e y2 marcandoli opportunamente e abbellire con varie etichette i grafici

Script, elementi di programmazione

Files .m

- Al posto di eseguire i comandi direttamente da linea di comando, possiamo memorizzare la successione dei comandi in un file di testo, salvarli e successivamente eseguirli; file di questo tipo sono detti **m-file** (files `.m`)
- Un file `.m` può essere generato con qualsiasi editor di testo ASCII (ad es. notepad di windows o l'editor di Matlab)

- Il **path** di Matlab è un insieme di directory sul computer locale in cui vengono cercate le funzioni o gli script che vengono chiamati dalla linea di comando
- Per visualizzare l'attuale path
  - dalla GUI di Matlab
  - dal prompt digitare il comando **`path`**

Path

- Se creiamo un nuovo file `.m` possiamo
  - salvare il file in una directory contenuta nel path
  - aggiungere la directory, in cui è salvato il file, al path
  - cambiare la directory di lavoro (Current Directory)

- **Per aggiungere una directory al path**
  - Menù **File** `->` **Set Path** apre il **Path Browser**
    - da Path Browser: selezionare la directory con il pulsante **Add Folder** (o **Add with subfolders**)
    - questa operazione è consigliata solo per inserire nuovi pacchetti o quando si vuole aggiungere una nuova funzione ai toolbox esistenti. Negli altri casi è consigliato lavorare nella directory di lavoro o sotto-directory
  - con il comando **`addpath('nome_dir')`**

Script e funzioni

- **Script files**: **`nome_file.m`**
  - eseguono una lista di istruzioni
  - non prevedono parametri di ingresso
  - utilizzano il workspace di MATLAB, **le variabili usate sono messe nella memoria di lavoro di MATLAB**
- **Funzioni: `nome_funzione.m`**
  - si possono passare parametri in ingresso ed ottenerne in uscita
  - sintassi **`function [y1,…,yn] = nome_funzione(x1,…,xn)`**
    - **`y1,…,yn`** -> parametri in uscita
    - **`x1,…,xn`** –> parametri in entrata
  - **le variabili usate all'interno sono locali**

Creazione di m-files

- **Per creare un nuovo m-file**
  - Menù **File** -> **New** ->**M-File**
- Per lanciare uno **script** (e quindi eseguire i comandi in esso contenuti)
  - **se siamo nella stessa directory** dove è salvato il file digitare il nome dello script nella linea di comando
  - **se siamo in una directory** diversa rsipetto a quella in cui è salvato lo script
    - digitare dal prompt **`run .../file.m`** **`...`** indica il percorso dalla cartella dove stiamo lavorando alla cartella in cui il file è salvato
    - è possibile cambiare la directory di lavoro utilizzando il comando **`cd`** (si digiti **`help cd`** per maggiori chiarimenti) oppure utilizzando le apposite icone nella barra dei comandi.

Script

- Tutte le variabili utilizzate nello script durante l'esecuzione dell' M-file vengono automaticamente messe nella **memoria di lavoro** di MATLAB
  - vedremo come questo non valga nel caso in cui si crei una funzione

- Per una minima manipolazione dei file su disco, MATLAB mette a disposizione alcuni comandi

```
dir, delete, cd, pwd, mkdir, copyfile, ls, cp
```

- **Esempio:** disegniamo una retta e una parabola

Commenti

- Il carattere **%** serve per introdurre un commento all'interno dello script, MATLAB ignora il contenuto alla destra del carattere **%** fino alla linea successiva
  - **CTRL R**(**CTRL T**) per commentare (eliminare il commento da) una riga
- Il commento all'inizio dello script file è particolarmente importante in MATLAB, infatti richiamando il comando help seguito dal nome dello script otteniamo come risposta il commento inserito all'inizio dello script stesso

```
>> help disegna
disegna.m
Disegna una retta e una parabola
```

- Una caratteristica degli script è quella di non avere parametri in ingresso modificabili. Ad esempio se vogliamo modificare i valori di n dobbiamo modificare ogni volta lo script.

Esempio

- Creiamo il file disegna.m e scriviamo la lista di comandi

```
% Disegna una retta e una parabola
%
n = 5;
x = linspace(-n,n);
y1 = x;
y2 = x.^2+5*x+1;

figure, hold on
plot(x,y1)
plot(x,y2,'r')
```

- Per eseguire il file, dal prompt

```
>> disegna
```

N.B. Il file deve essere salvato nella directory di lavoro che è quella in cui ci troviamo!!!

Cancellazione numerica

- La cancellazione numerica è la perdita di cifre significative
- E' un fenomeno che si verifica durante l'operazione di **sottrazione** tra due numeri "quasi uguali"
  - se due numeri sono quasi uguali, dove uguali s'intende a meno della precisione macchina, allora è possibile il verificarsi della cancellazione numerica.
- Siano *x1 e x2* due numeri reali. Se *x=x1-x2* è "molto piccolo", l'errore relativo

$$\delta_x = \left| \frac{fl(x1 - x2) - x}{x} \right|$$

può essere molto grande e ciò produce una perdita di cifre significative nel calcolo di *fl(x1-x2)*

- E' sempre preferibile evitare la sottrazione tra numeri macchina "quasi uguali"

Esercizio

- Scrivere uno script **`cancellazione.m`** in cui si calcolino numericamente le soluzioni dell'equazione di secondo grado

$$ax^2 + bx + c = 0$$

con le seguenti formule

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \qquad x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a} \qquad x_2 = \frac{2ac}{-b - \sqrt{\Delta}}$$

per i valori di

  - *a=1, b=206.5, c=0.01021*
  - *a=1, b=50000, c=0.01*

- Calcolare il valore dell'equazione per tali valori e il prodotto *x1x2*