Uso avanzato di MATLAB

Miscione Giuseppe

g.miscione@virgilio.it

Indice

- Funzioni matematiche
- Funzioni di arrotondamento
- Disegnare funzioni
- Creare script
- Controllo del flusso di esecuzione

Funzioni matematiche - 1

- **sin(x)**
- **cos(x)**
- **tan(x)**
- **asin(x)**
- **acos(x)**
- **atan(x)**
- **sqrt(x)**
- **exp(x)**
- **log(x)**
- **log10(x)**
- **log2(x)**

Digitare il comando “help elfun” per avere un elenco completo delle funzioni implementate

Funzioni matematiche - 2

- **Tutte le funzioni matematiche possono essere applicate sia a scalari che a vettori e matrici;**
- **In questi ultimi casi la funzione è applicata ad ogni elemento del vettore o della matrice.**

Funzioni di arrotondamento

- **round(x)** : arrotonda *x* nella maniera usuale
  - round(8.75) = 9 ; round(-8.75)= -9 ;
- **fix(x)** : arrotonda *x* muovendosi verso 0
  - fix(8.75) = 8 ; fix(-8.75)= -8 ;
- **floor(x)** : arrotonda *x* muovendosi verso -∞
  - floor(8.75) = 8 ; fix(-8.75)= -9 ;
- **ceil(x)** : arrotonda *x* muovendosi verso +∞
  - ceil(8.75) = 9 ; ceil(-8.75)= -8 ;

Disegnare funzioni - 1

- Uno dei campi in cui si può utilizzare il MATLAB è l'analisi dei dati;
- Una delle tecniche per eseguire questa analisi è l'utilizzo di grafici che riproducono l'andamento dei dati;
- Il MATLAB propone, per questo, delle funzioni per disegnare grafici.

Disegnare funzioni - 2

- Prendiamo i vettori:

  x = [x1 x2 x3 x4 x5 x6]

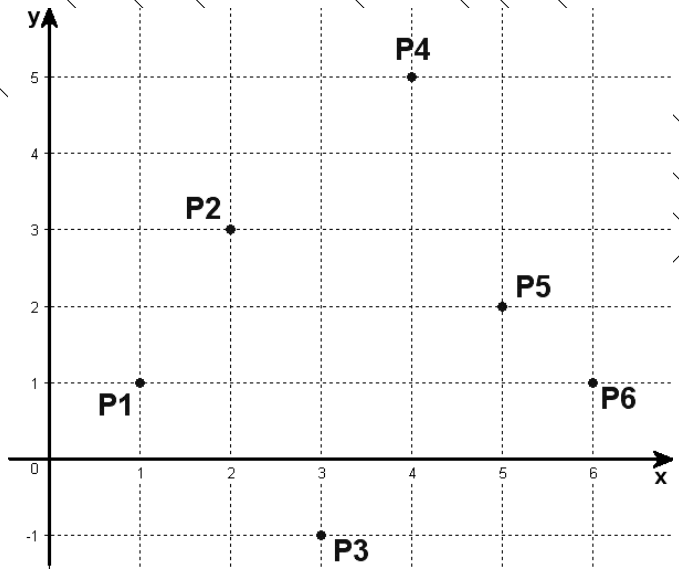
  y = [y1 y2 y3 y4 y5 y6]
- Definiamo i punti nel piano cartesiano:

  P1(x1,y1) P2(x2,y2)

  P3(x3,y3) P4(x4,y4)

  P5(x5,y5) P6(x6,y6)

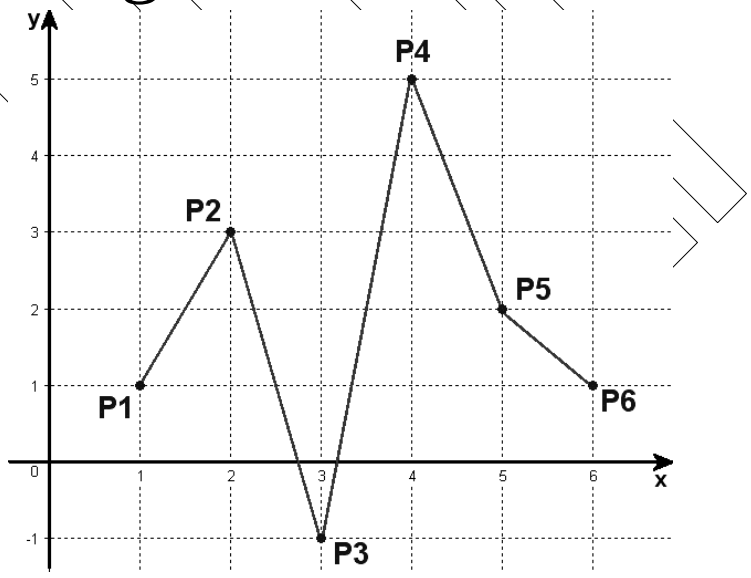
Disegnare funzioni - 3



Disegnare funzioni - 4

- Colleghiamo con una linea retta P1 con P2, poi P2 con P3, P3 con P4 e così via;
- In questo modo otteniamo un'approssimazione del grafico della funzione che lega *y* con *x*.

Disegnare funzioni - 5



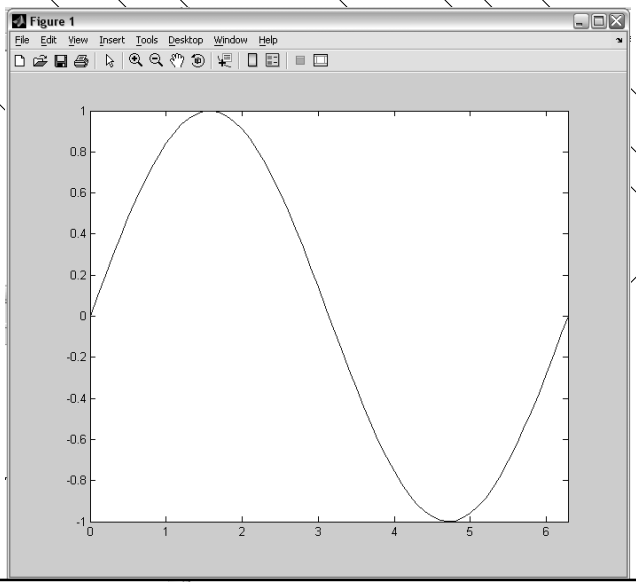
Disegnare funzioni - 6

- **plot(x, y)** : disegna la funzione che lega *y* a *x* seguendo la procedura appena descritta; *x* e *y* devono essere vettori con le stesse dimensioni ed essere entrambi vettori riga o vettori colonna.

Disegnare funzioni - esempio

```
x = linspace(0,2*pi,60);
y = sin(x);
plot(x, y)
```

Disegnare funzioni - esempio

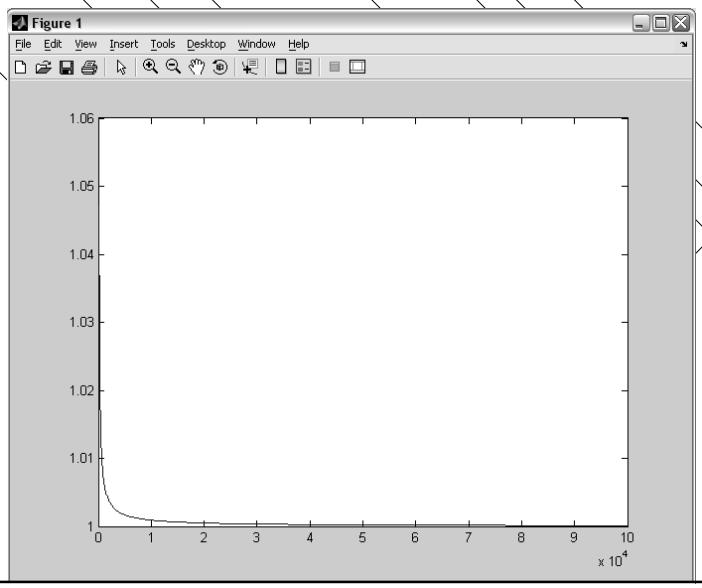


Disegnare funzioni - esempio

$$\lim_{n\to+\infty} \sqrt[n]{n} = \lim_{n\to+\infty} (n)^{1/n}$$

```
n = [0:100:100000];
n(1) = 1;
lim = n.^(1./n);
plot(n,lim)
```

Disegnare funzioni - esempio

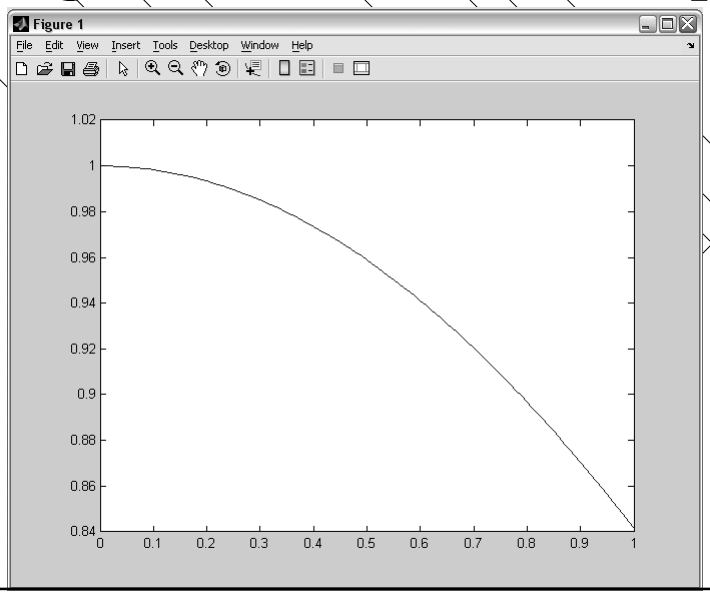


Disegnare funzioni - esempio

$$\lim_{x \to 0} \frac{\sin(x)}{x}$$

```
x = [1:-0.01:0.00001];
lim = sin(x)./x;
plot(x,lim)
```

Disegnare funzioni - esempio



Disegnare funzioni - 7

- Spesso è utile disegnare in una stessa figura più funzioni, in modo da poterle confrontare;
- Ogni volta che viene usato, il comando "**plot**" dapprima cancella la figura attuale e poi disegna la nuova;
- Quindi usandolo più volte consecutivamente otterremmo solo il grafico dell'ultima funzione.

Disegnare funzioni - 8

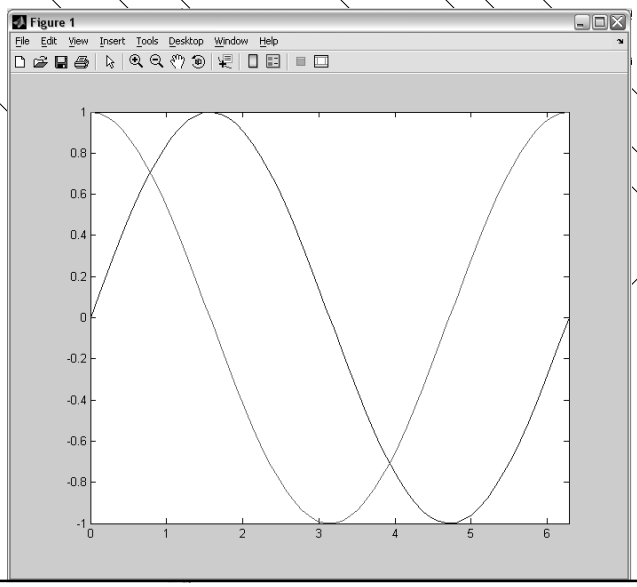
- Per evitare questo problema si può usare la sintassi:

  plot(x1, y1, x2, y2, …)

- Ad esempio:

```
x = linspace(0,2*pi,60);
y1 = sin(x);
y2 = cos(x);
plot(x, y1, x, y2)
```

Disegnare funzioni - esempio



Funzioni accessorie per i disegni - 1

- **figure(n)** : apre e rende attiva la figura numero *n*. Dopo l'esecuzione di questo comando tutte le operazioni di disegno verranno effettuate nella finestra scelta;
- **grid on** : disegna una griglia sulla figura attiva;
- **grid off** : cancella la griglia dalla figura attiva;

Funzioni accessorie per i disegni - 2

- **xlabel('asse x')** : inserisce l'etichetta 'asse x' sull'asse delle ascisse;
- **ylabel('asse y')** : inserisce l'etichetta 'asse y' sull'asse delle ordinate;
- **title('titolo')** : inserisce il titolo 'titolo' in alto, sulla figura;
- Ovviamente, le etichette e il titolo possono essere cambiati a piacimento!

Funzioni accessorie per i disegni - 3

- **axis([xmin xmax ymin ymax])** : imposta i valori minimi e massimi da disegnare sugli assi cartesiani;
- **legend('funzione1', 'funzione2', …)** : inserisce una legenda nella figura per poter riconoscere più facilmente il significato delle varie linee disegnate. In input abbiamo tante stringhe quante sono le funzioni disegnate con il comando "**plot**";

Funzioni accessorie - esempio

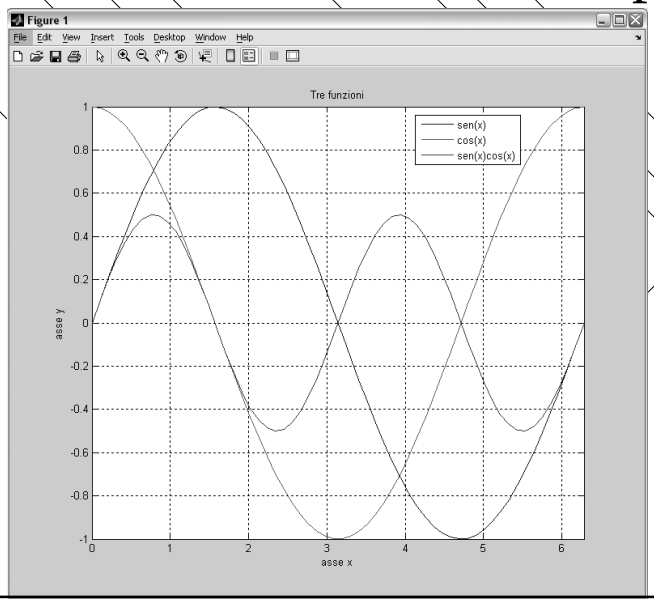
```
x = linspace(0,2*pi,60);
y1 = sin(x);
y2 = cos(x);
y3 = sin(x).*cos(x);
```

Funzioni accessorie - esempio

```
plot(x, y1, x, y2, x, y3)
axis([0  2*pi  -1  1])
grid on
xlabel('asse x')
ylabel('asse y')
title('Tre funzioni')
legend('sen(x) ', 'cos(x) ', 'sen(x)cos(x)')
```

Abbiamo usato tre stringhe perché abbiamo disegnato tre funzioni

Funzioni accessorie - esempio



Creare script - 1

- Uno script è un file di testo con estensione “.m” che contiene un insieme di istruzioni MATLAB che risolvono un determinato problema;
- Usando gli script è possibile risolvere problemi che richiedono un gran numero di istruzioni, oppure risolvere più volte lo stesso problema cambiando solo un dato o un parametro, senza dover riscrivere ogni volta, una per una, tutte le istruzioni.

Creare script - 2

- Per aprire l’editor di script, nella finestra principale del MATLAB si può cliccare sul menu “File” e scegliere la voce “New | M-file”;
- Alternativamente si può digitare il comando “**edit**” nella Command Window.

Creare script - 3

- Come in ogni altro linguaggio di programmazione è possibile inserire dei commenti negli script;
- In MATLAB i commenti iniziano con il carattere “%” e terminano alla fine della riga. Ad esempio:

```
%Questo è un commento
A=[1 2;3 4]; %Un altro commento
```

Creare script - 4

- Una volta completata la scrittura dello script, con il pulsante evidenziato nell’immagine sopra è possibile eseguirlo;
- I risultati verranno mostrati nella Command Window.

Controllo di flusso

- In uno script è possibile usare istruzioni di controllo del flusso di esecuzione (if…else…, for…, while…), come in altri linguaggi di programmazione (ad esempio C++, Java, ecc.);
- Un salto oppure un ciclo vengono eseguiti se una determinata condizione risulta vera;
- Per descriverla si ricorre all'uso di operatori di confronto e logici.

Controllo di flusso – operatori di confronto

- == : uguaglianza;
- ~= : disuguaglianza;
- < , <= : minore, minore o uguale;
- > , >= : maggiore, maggiore o uguale.

Controllo di flusso – operatori logici

- **&** : AND logico (A & B è vero se sia A che B sono veri);
- **|** : OR logico (A | B è vero se o A o B sono veri);
- **~** : negazione logica (~A è vero se A è falso e viceversa);

Controllo di flusso – if

if <*condizione*>

<istruzioni eseguite se la condizione è vera>

else

<istruzioni eseguite se la condizione è falsa>

end

Controllo di flusso – esempio

Riprendiamo l'esempio guida della lezione precedente e risolviamo con uno script il seguente sistema lineare:

$$\begin{bmatrix} 1 & -1 & 0 & -6 \\ 2 & 0 & -3 & 2 \\ 5 & 1 & -2 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 2 \\ -1 \end{bmatrix}$$

Controllo di flusso – esempio

$\det(A) \neq 0 \Rightarrow$ **Il sistema può avere una ed una sola soluzione**

$rango(A) = rango([A|b]) \Rightarrow$ **Il sistema ha una ed una solo soluzione**

$x = A^{-1} \cdot b \Rightarrow$ **Si trova la soluzione**

Controllo di flusso – esempio

Cancelliamo il workspace e ripuliamo la Command Window

```
clear
clc
b = [ -10; 3; 2; -1 ];
A = [1 -1 0 -6; 2 0 -3 2; 5 1 -2 0; 0 0 1 -1];
Ab = [ A b ];
```

Controllo di flusso – esempio

```
if(det(A)~=0 & rank(A)==rank(Ab))
  disp('Il sistema è risolubile')
  x = A\b % equivale a: x = inv(A)*b
else
  disp('Il sistema non è risolubile');
end
```

Controllo di flusso – while

while <*condizione*>

<istruzioni eseguite finchè la condizione è vera>

end

Controllo di flusso – for

for k=a:delta:b

<istruzioni eseguite per k=a, k=a+delta, k=a+2delta, k=a+3delta, …, k=b>

end

All’indice del ciclo (k nell’esempio sopra) può essere assegnato ogni nome di variabile valido.

Controllo di flusso – for

- L’esecuzione di cicli nel codice MATLAB è piuttosto lenta;
- Se possibile, l’uso di cicli va evitato usando una o più funzioni predefinite che permettono di arrivare allo stesso risultato.

Controllo di flusso – esempio

```
clc;
b=[];
for k=1:70000
  b=[b 0];
end
disp('Finito');
```

Crea un vettore vuoto

Questo ciclo impiega circa 13 secondi per eseguirsi mentre b=zeros(1,70000) è eseguito quasi immediatamente.

Controllo di flusso – Successione di Fibonacci

La successione di Fibonacci è così definita:

$$\{a_k\}, \forall k \geq 1$$

$$a_1 = 1$$

$$a_2 = 1$$

$$a_k = a_{k-1} + a_{k-2}, \forall k \geq 3$$

Controllo di flusso – Successione di Fibonacci

```
clear;
clc;
a=[1 1];
for k=3:10
   a=[a   a(k-1)+a(k-2)];
end
disp('Finito');
a
```