

Università degli Studi di Ferrara

Corso di Laurea in Matematica - A.A. 2018 - 2019

Programmazione Lezione 15 – Grafica in MATLAB

Docente: Michele Ferrari - michele.ferrari@unife.it

Nelle lezioni precedenti

- MATLAB: Vettori e Matrici
- Analisi dei tipi di dato in MATLAB
- Creazione di Vettori e Matrici
- Manipolazione di Vettori e Matrici
- Operazioni di base
- Operatori puntuali

In questa lezione

- Funzioni e operatori vettoriali
- Conversione di strutture dati
- Grafici in MATLAB
- Funzioni e operatori per personalizzare i grafici

Funzioni e operatori vettoriali

Abbiamo visto un insieme di operatori che sono in grado di agire direttamente su intere strutture dati.

Analizziamo il problema di visualizzare una tabella di valori di una funzione matematica così definita:

$$f(x) = \frac{15120 - 6900x^2 + 313x^4}{15120 + 660x^2 + 13x^4}$$

nell'intervallo $[0, \pi]$.

Funzioni e operatori vettoriali

Tale funzione è una particolare approssimazione della funzione $\cos(x)$.

Per costruire una tabella di 5 valori potremmo:

- Utilizzare `linspace` per generare un vettore adeguato x
- Scrivere la funzione utilizzando correttamente gli operatori vettoriali
- Aggiustare la struttura dati del risultato

Funzioni e operatori vettoriali

```
>> x=linspace(0,pi,5);  
>> y=(15120-6900*x.^2+313*x.^4)./(15120+660*x.^2+13*x.^4);  
>> [x' y']
```

ans =

0	1.0000
0.7854	0.7071
1.5708	0.0000
2.3562	-0.7057
3.1416	-0.9821

Funzioni e operatori vettoriali

Notiamo come utilizzando le operazioni vettoriali i valori della funzione in corrispondenza della successione di punti definita dal vettore x sono creati **con una singola istruzione vettoriale** ed assegnati al vettore y

Conversione di strutture dati

Nel caso si avesse la necessità di convertire dati memorizzati un certo tipo di array in un array organizzato diversamente (mantenendo lo stesso numero di elementi) possiamo utilizzare la funzione **reshape**

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

```
A =
```

```
 1  2  3
 4  5  6
 7  8  9
10 11 12
```

```
>> B = reshape(A,2,6)
```

```
B =
```

```
 1  7  2  8  3  9
 4 10  5 11  6 12
```

Conversione di strutture dati

Sintassi:

```
reshape(Matrice, Righe, Colonne)
```

dove i parametri Righe e Colonne si riferiscono alla nuova matrice prodotta come risultato

Esiste inoltre una forma molto compatta di conversione matrice → vettore colonna utilizzando la notazione due punti

Conversione di strutture dati

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

```
>> x = A(:)
```

```
x =
```

```
1
```

```
4
```

```
7
```

```
2
```

```
5
```

```
8
```

```
3
```

```
6
```

```
9
```

Conversione di strutture dati

Chiaramente qualora si abbia la necessità di un vettore riga basterà considerare il trasposto della precedente espressione.

Notiamo che la notazione $x(:)$ genererà sempre un vettore colonna, a prescindere dalla struttura dati originale, a differenza dell'operatore di trasposizione x' che può cambiare i vettori colonna in vettori riga e viceversa

Grafica in MATLAB

MATLAB possiede numerose capacità grafiche che consentono di rappresentare **molto velocemente** dati memorizzati in vettori e matrici.

Ove in altri linguaggi avremmo dovuto preoccuparci di:

- Quali librerie grafiche utilizzare
- Eventuale compatibilità nativa con il motore grafico del sistema operativo
- Come ragionano queste librerie nel disegnare finestre ed elementi

In MATLAB l'ambiente fornisce supporto nativo e funzioni immediate per la grafica, in particolare per rappresentazioni di strutture dati e grafici

Grafica in MATLAB

Consideriamo l'esempio di voler ottenere un grafico bidimensionale per rappresentare la variazione di una variabile rispetto ad un'altra, ovvero il grafico di una funzione $y=f(x)$

Ipotizziamo quindi di voler rappresentare una successione di valori della funzione coseno sull'intervallo $[0,2\pi]$

Come per l'esempio precedente ed i numerosi esercizi svolti a lezione la nostra prima preoccupazione deve essere quella di generare le strutture dati su cui lavorare, pertanto devo ottenere un vettore x contenente i valori nell'intervallo indicato ed un vettore y con i corrispondenti valori della funzione

Grafica in MATLAB

```
>> n=21;
```

```
>> x=linspace(0,2*pi,n);
```

```
>> y=cos(x);
```

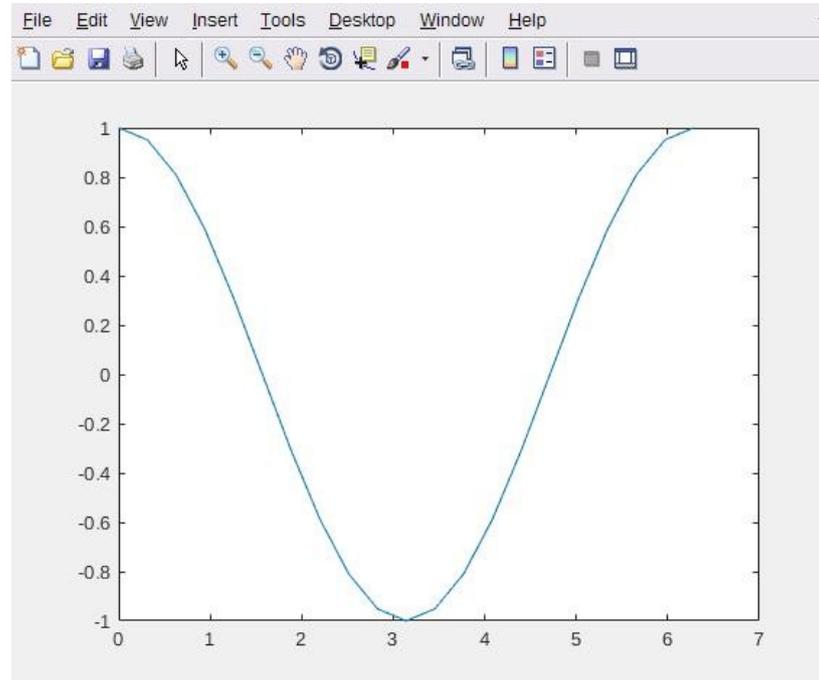
Ovvero:

- Genero x come vettore di 21 punti equispaziati nell'intervallo $0, 2\pi$
- Genero y come funzione coseno dei valori di x

La funzione plot

A questo punto, avendo i dati, posso generare il grafico utilizzando la funzione **plot**:

```
>>plot(x,y)
```



La funzione plot

In questo modo otteniamo una rappresentazione grafica della tabella di valori ottenuta semplicemente raccordando con segmenti di retta nel piano xy i vertici $(x(i), y(i))$ in modo ordinato al variare di i da 1 a n .

La scelta della scala di visualizzazione è automatica.

Potremmo voler evidenziare i vertici della poligonale così costruita, in tal caso ci viene in aiuto la possibilità di passare opzioni alla funzione `plot`, la cui sintassi completa è:

```
plot(Vettore1, Vettore2, Opzioni)
```

La funzione plot

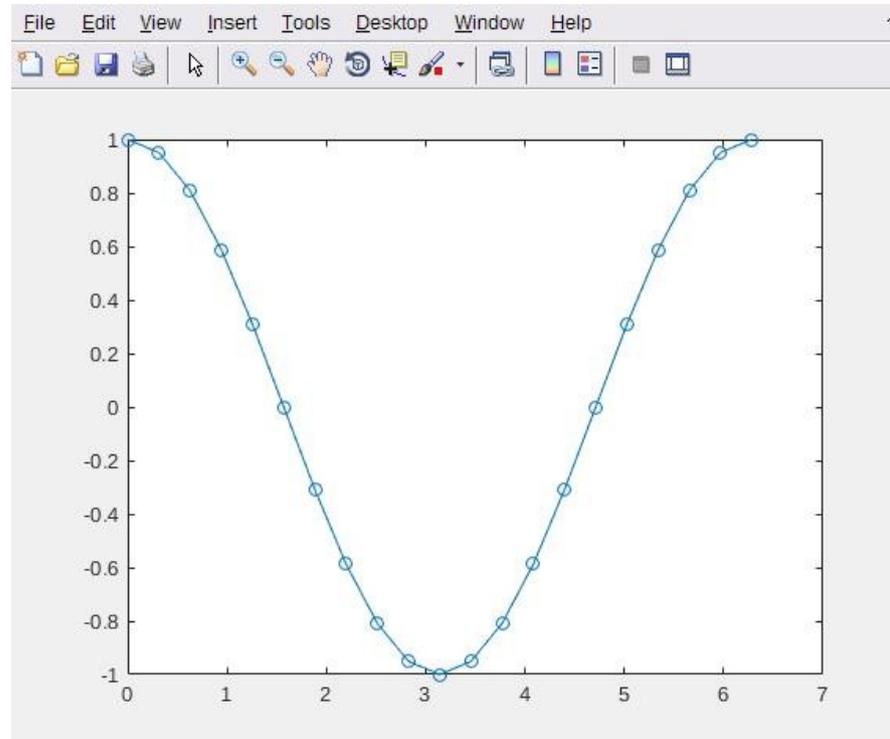
Dove:

- Vettore1 e Vettore2 contengono le ascisse e le ordinate dei punti
- Opzioni è una stringa (si usa tra apici) che definisce il tipo di colore, di simbolo e di linea utilizzati nel grafico.

La funzione plot

Vediamo dunque il risultato di una chiamata alla funzione plot fatta nel seguente modo:

```
>> plot(x,y,'-o')
```



La funzione plot

Notiamo infine che l'utilizzo del comando plot nella semplice forma

```
>> plot(vettore)
```

Realizzerà un grafico del vettore rispetto ai propri indici.

Personalizzare un grafico

Possiamo inoltre commentare il grafico aggiungendo le istruzioni

```
>> title('Grafico della funzione cos(x)')
```

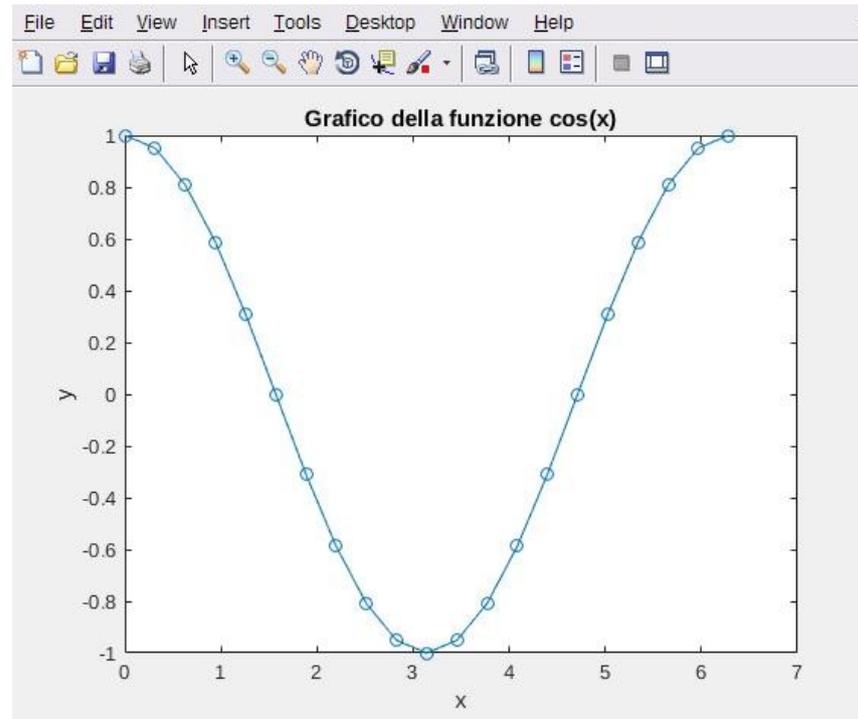
```
>> xlabel('x')
```

```
>> ylabel('y')
```

che consentono di inserire informazioni sul titolo del grafico e sugli assi x e y.

Personalizzare un grafico

Ottenendo così:



Se volessimo ottenere un risultato che emuli meglio una curva continua basterà dimensionare n , aumentandone i punti in modo da minimizzare i segmenti di raccordo.

Personalizzare un grafico

Vediamo alcune opzioni del comando plot:

Colore	Simbolo	Tipo di linea
y giallo	. punto	- linea continua
m magenta	o circoletto	: linea punteggiata
c ciano	x per	-. linea punto
r rosso	+ più	- - linea tratteggiata
g verde	* asterisco	
b blu	s quadratino	
w bianco	d diamante	
k nero	v triangolo	

Riscalare i grafici

Una caratteristica importante è quella di potere riscalare i grafici tramite la funzione **axis**:

```
>> x = linspace(0,2*pi);
```

```
>> y = cos(x);
```

```
>> plot(x,y);
```

```
>> a = axis
```

```
a =
```

```
    0    7   -1    1
```

```
>> axis([a(1) pi a(3) a(4)])
```

per ottenere il grafico soltanto sull'intervallo $[0, \pi]$.

La funzione

```
axis([xmin xmax ymin ymax])
```

impone che l'intervallo di valori di x sia compreso tra $xmin$ e $xmax$ e quello di y tra $ymin$ e $ymax$.

`axis` usato da solo riporta alla scala automatica e restituisce come risultato un vettore contenente i valori attuali della scala.

Esercitazione

- Generare un vettore x di 60 punti equispaziati tra 0 e 2π
- Generare una tabella di valori per la funzione seno di x
- Visualizzare un grafico con opportune etichette e titolo, evidenziare i punti di raccordo e utilizzare un colore personalizzato
- Ripetere l'esercizio per una funzione esponenziale ed una logaritmica

Grazie per l'attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

Per la parte introduttiva di MATLAB:

L. Pareschi, G. Dimarco “Introduzione a MATLAB”, corso di Laboratorio di Calcolo Numerico 2006