Università degli Studi di Ferrara

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

Programmazione
Lezione 8 – Grafica in MATLAB

Lorenzo Caruso - lorenzo.caruso@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,π].

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 in 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 [0,2π]

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 contente i valori nell'intervallo indicato ed un vettore y con i corrispondenti valori della funzione

Grafica in MATLAB

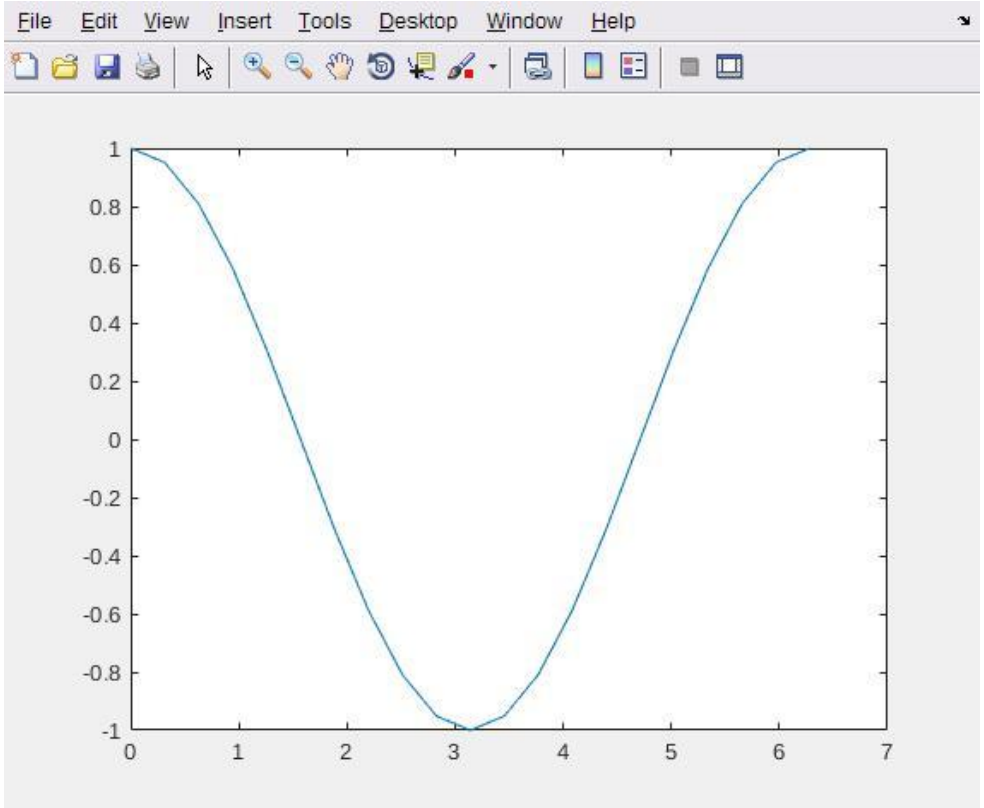
```
>> n=21;
>> x=linspace(0,2*pi,n);
>> y=cos(x);
```

Ovvero:

- Genero x come vettore di 21 punti equispaziati nell'intervallo 0, 2π
- 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)

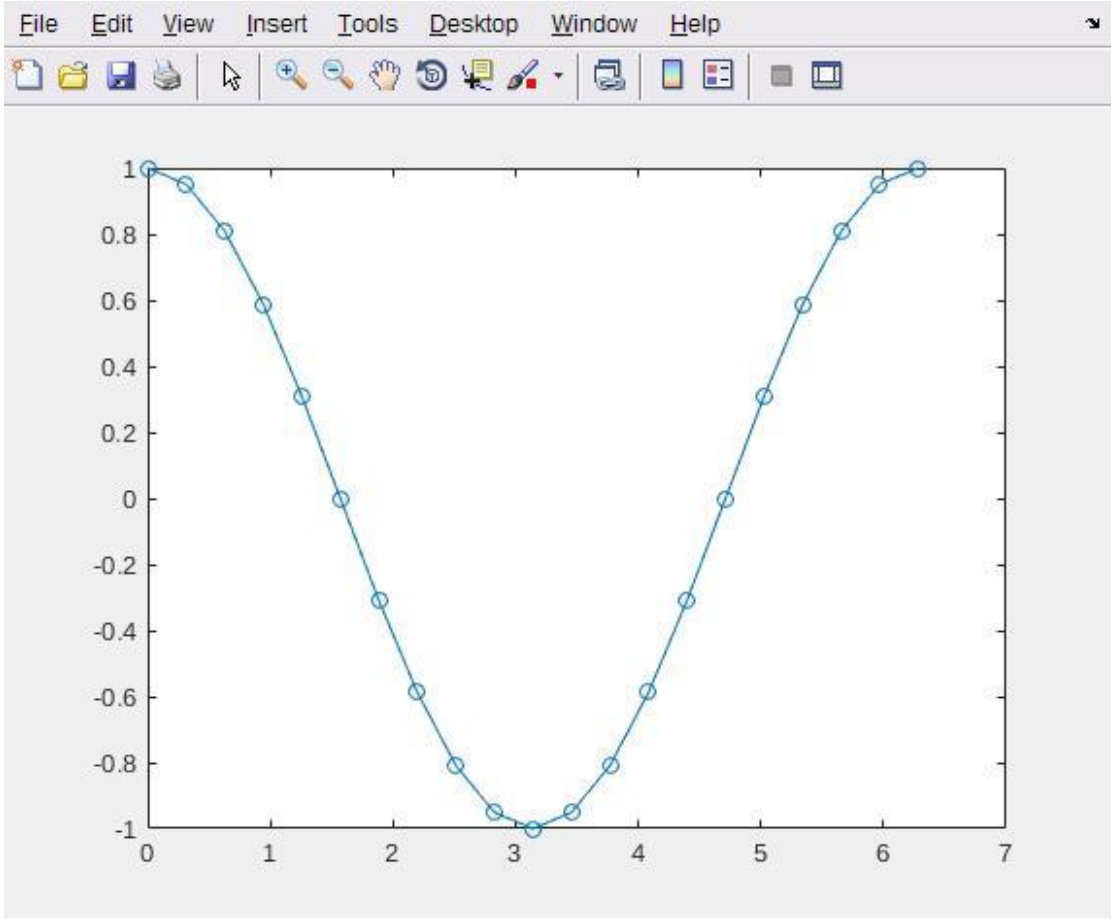
La funzione plot

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

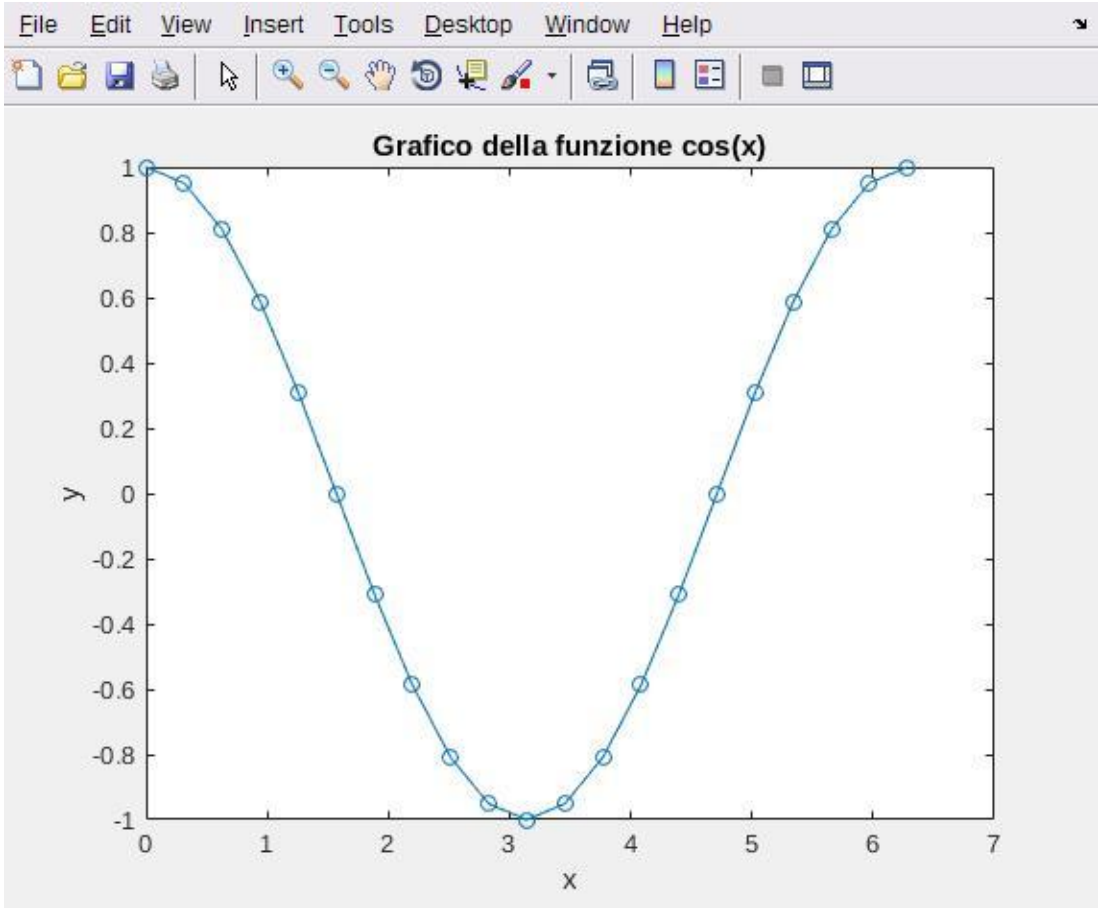
Possiamo inoltre commentare il grafico aggiungendo le istruzioni

```
>> 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, π].
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 contenete i valori attuali della scala.

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

Università degli Studi di Ferrara

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

Programmazione
Lezione 9 – Grafica in MATLAB
Seconda parte

Lorenzo Caruso - lorenzo.caruso@unife.it

Nelle lezioni precedenti

- Funzioni e operatori vettoriali
- Conversione di strutture dati
- Grafici in MATLAB: la funzione **plot**
- Funzioni e operatori per personalizzare i grafici

In questa lezione

- Grafici Sovrapposti
- Commentare un grafico
- Sottografici
- Grafici di superfici

Grafici Sovrapposti

Ci poniamo il problema di realizzare i grafici delle funzioni seno e coseno sull'intervallo [0, 2π] nella stessa finestra grafica.

Per fare questo abbiamo due possibilità:

- I comandi **hold**
- Un particolare utilizzo della funzione **plot**

Grafici Sovrapposti

Per prima cosa, come sempre, è necessario costruire le strutture dati con cui lavorare:

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

I comandi hold

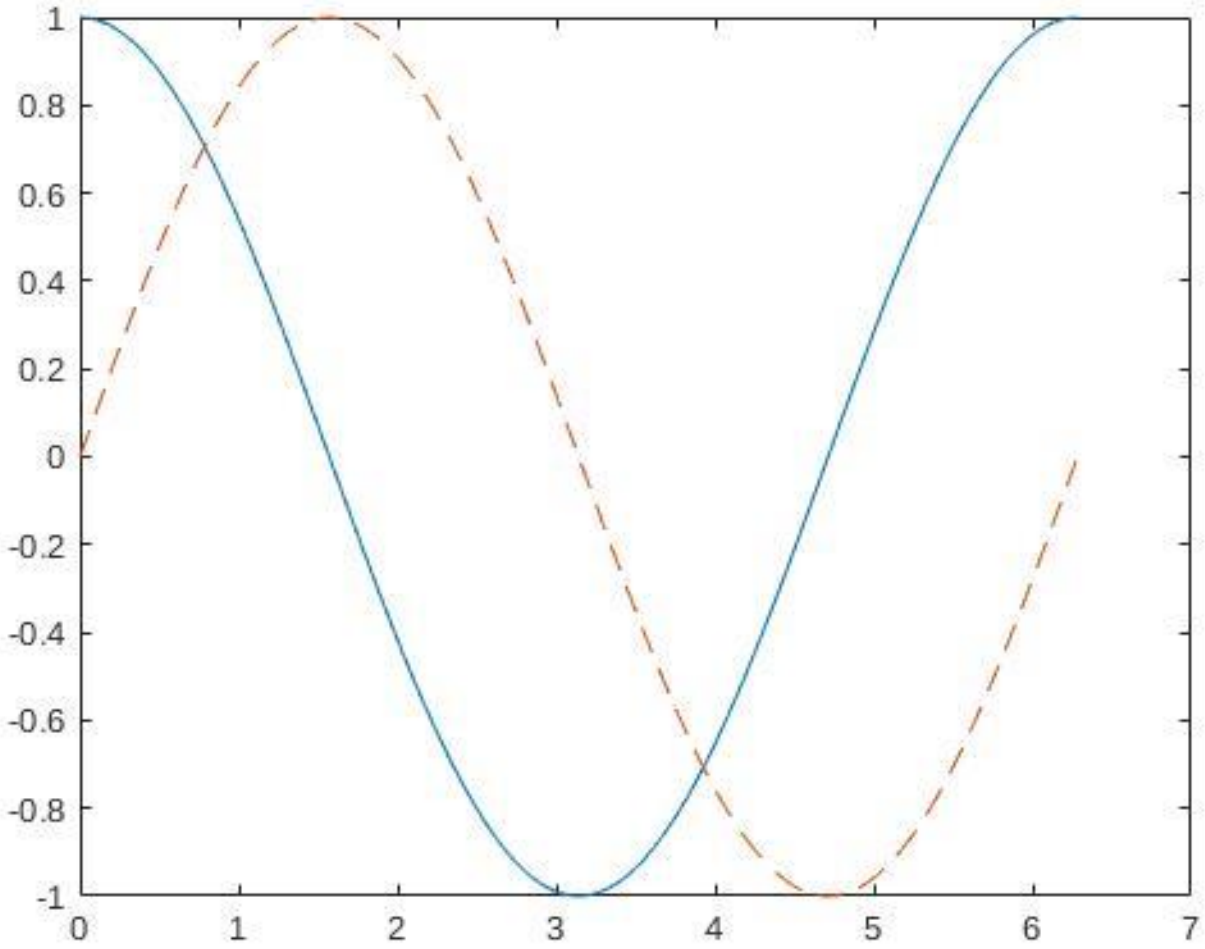
Utilizzando il primo metodo possiamo costruire grafici sovrapposti nel seguente modo:

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

- Il comando **hold on** fa sovrapporre tutti i grafici successivi al comando nella finestra grafica
- Il comando **hold off** ritorna all'impostazione originale

Notiamo come abbiamo dovuto preoccuparci di distinguere le linee dei due grafici sovrapposti utilizzando le opzioni di personalizzazione della funzione **plot** “- -” e “-”

I comandi hold: risultato

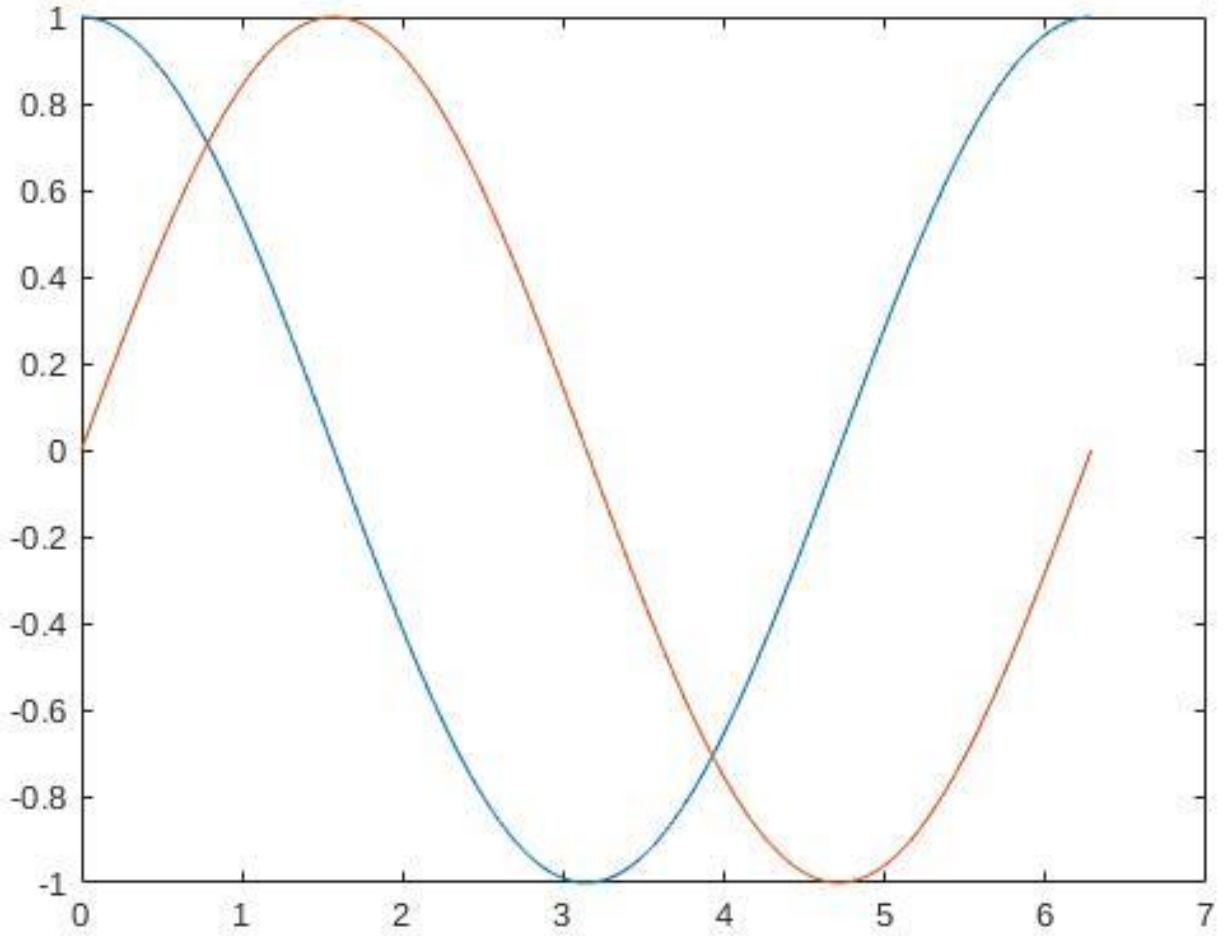


Grafici sovrapposti con plot

Fornendo a plot i corretti parametri è possibile ottenere la sovrapposizione dei grafici con una distinzione automatica delle linee:

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

La funzione plot: risultato



Funzione plot e personalizzazione

Proviamo ad integrare il comando precedente con qualche miglioramento:

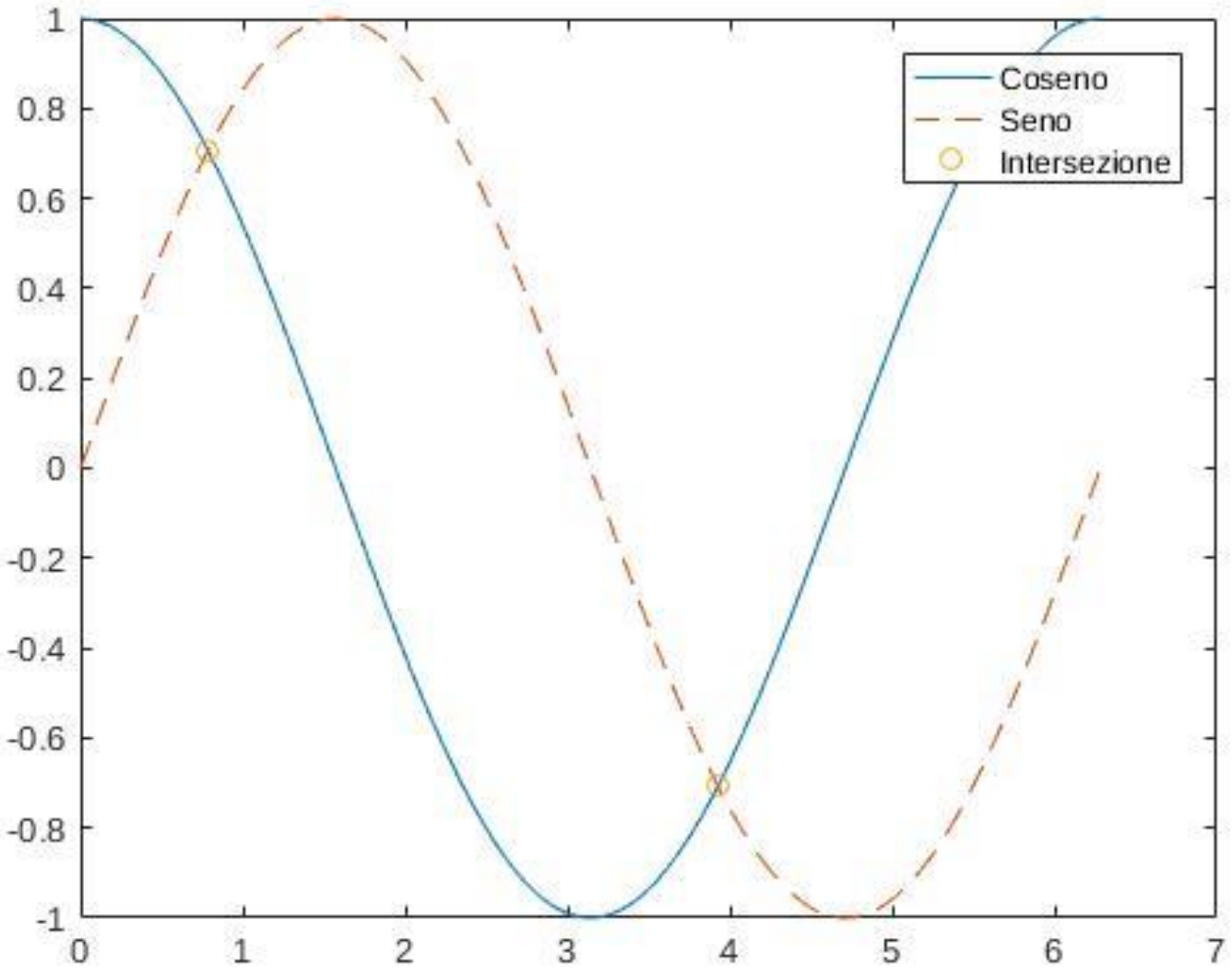
```
>>  plot(x,y1,'-',x,y2,'--',[1 5]*pi/4,[1 -1]/sqrt(2),'o')
>> legend('Coseno', 'Seno', 'Intersezione')
```

Ovvero:

- Disegna con una linea continua il grafico x, y1
- Disegna con una linea tratteggiata il grafico x, y2
- Disegna un cerchietto nelle due intersezioni
- Disegna una legenda con le diciture indicate

Nota: La funzione legend inserisce una legenda nel grafico che identifica le curve disegnate con linee e simboli differenti

Funzione plot e personalizzazione



Commentare un grafico

Funzione	Significato
axis	Permette di impostare i valori minimi e massimi sugli assi x e y
title	Inserisce un titolo nel grafico
xlabel	Nome per l'asse x
ylabel	Nome per l'asse y
zlabel	Nome per l'asse z
grid	Disegna una griglia sugli assi x e y
legend	Aggiunge una legenda
text	Inserisce una stringa di testo in una posizione specifica

Possono essere trovate ulteriori funzioni utilizzando il comando:
>> help graph2d

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)

dove Righe e Colonne definiscono la struttura della matrice di sottofinestre grafiche all'interno della finestra grafica principale e Sottofinestra indica il numero della sottofinestra grafica attiva.
Consideriamo a titolo di esempio l'istruzione
>> 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:

1	2	3
4	5	6

Grafici in sottofinestre: esempio

Vogliamo disegnare in sottofinestre grafici di diverse funzioni, esempio:

$$f(x) = exp^{-x^2} cos(kx)$$

Nell' intervallo [−2, 2] al variare di k intero. Possiamo scrivere:

```
>> x=linspace(-2,2);
>> y= exp(-x.^2).*cos(pi*x);
>> subplot(2,2,1);
>> plot(x,y); title('k=1');
>> y= exp(-x.^2).*cos(2*pi*x);
>> subplot(2,2,2);
>> plot(x,y); title('k=2');
>> y= exp(-x.^2).*cos(3*pi*x);
>> subplot(2,2,3);
>> plot(x,y); title('k=3');
>> y= exp(-x.^2).*cos(4*pi*x);
>> subplot(2,2,4);
>> plot(x,y); title('k=4');
```

Grafici in sottofinestre: esempio

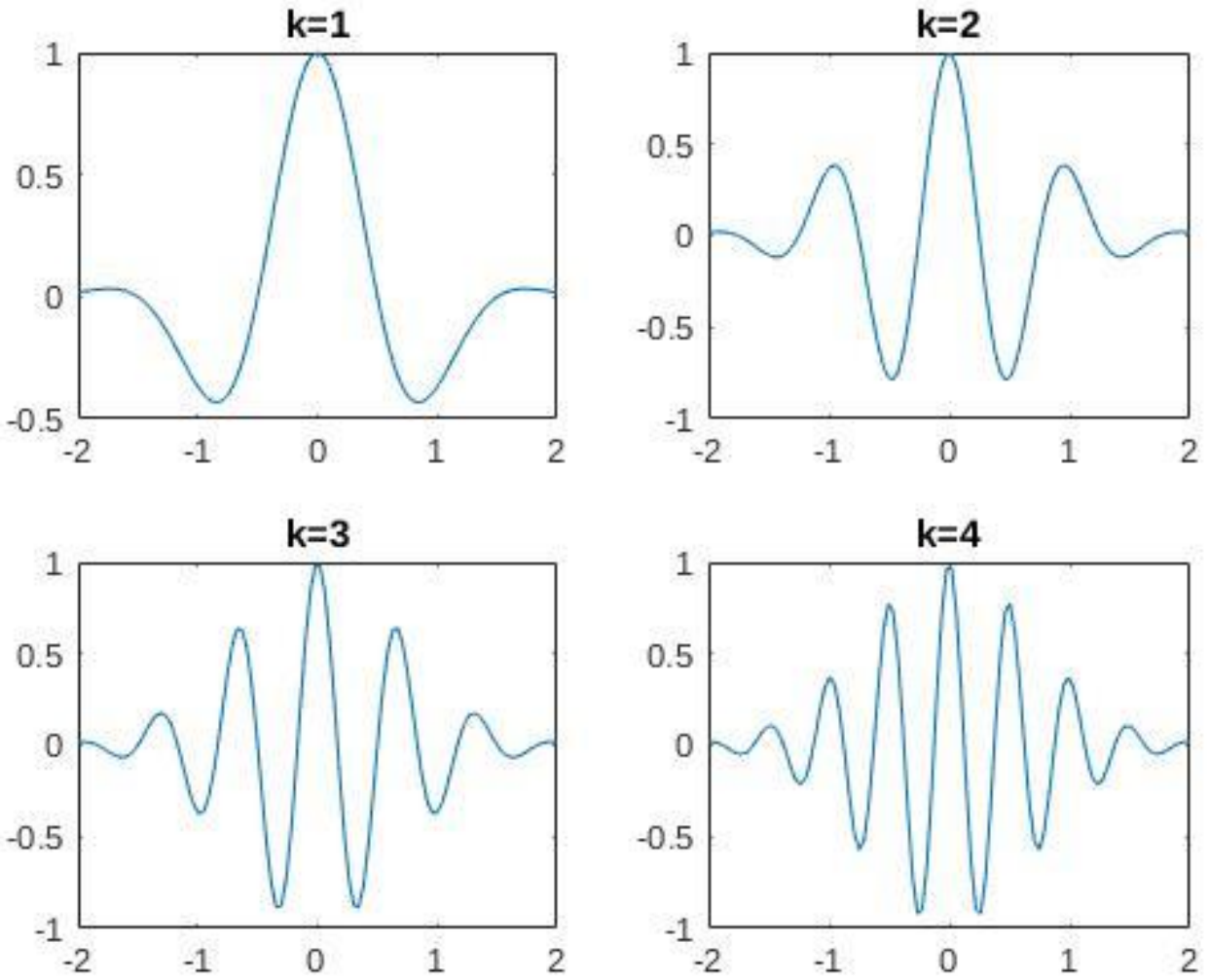


Grafico di superficie

Le capacità grafiche di MATLAB consentono di produrre grafici di funzioni in più variabili, ossia rappresentazioni grafiche di array a più indici.

Così come una funzione di una variabile y= f(x) definisce una curva nel piano, una funzione di due variabili indipendenti z= f(x,y) definisce una superficie nello spazio tridimensionale.

Grafico di superficie

Consideriamo la funzione

$$z = x(1 - x)y(1 - y)$$

nel dominio rettangolare $0 \leq x \leq 1$, $0 \leq y \leq 1$.

Per costruirne il grafico della superficie dobbiamo innanzitutto definire la griglia di valori (x, y) nei quali valuteremo la nostra funzione $z = f(x, y)$.

La funzione meshgrid

Per costruire la griglia di valori possiamo utilizzare la funzione **meshgrid** nella forma:

```
>> n=5;m=5;
>> x=linspace(0,1,n);
>> y=linspace(0,1,m);
>> [X,Y]=meshgrid(x,y)
```

La funzione meshgrid

La funzione costruisce due matrici MxN X e Y, restituisce pertanto:

X =

0	0.2500	0.5000	0.7500	1.0000
0	0.2500	0.5000	0.7500	1.0000
0	0.2500	0.5000	0.7500	1.0000
0	0.2500	0.5000	0.7500	1.0000
0	0.2500	0.5000	0.7500	1.0000

Y =

0	0	0	0	0
0.2500	0.2500	0.2500	0.2500	0.2500
0.5000	0.5000	0.5000	0.5000	0.5000
0.7500	0.7500	0.7500	0.7500	0.7500
1.0000	1.0000	1.0000	1.0000	1.0000

La funzione meshgrid

Le risultati matrici vengono così costruite:

X=

x(1)	x(2)	...	x(n)
x(1)	x(2)	...	x(n)
...	...	...	...
x(1)	x(2)	...	x(n)

Y=

y(1)	y(1)	...	y(1)
y(2)	y(2)	...	y(2)
...	...	...	...
y(m)	y(m)	...	y(m)

>> help meshgrid
meshgrid Cartesian grid in 2-D/3-D space

[X,Y] = meshgrid(xgv,ygv) replicates the grid vectors xgv and ygv to produce the coordinates of a rectangular grid (X, Y). The grid vector xgv is replicated **numel(ygv)** times to form the columns of X. The grid vector ygv is replicated numel(xgv) times to form the rows of Y.

>> help numel
 numel Number of elements in an array or subscripted array expression.
 N = numel(A) returns the number of elements, N, in array A

Grafico di superficie

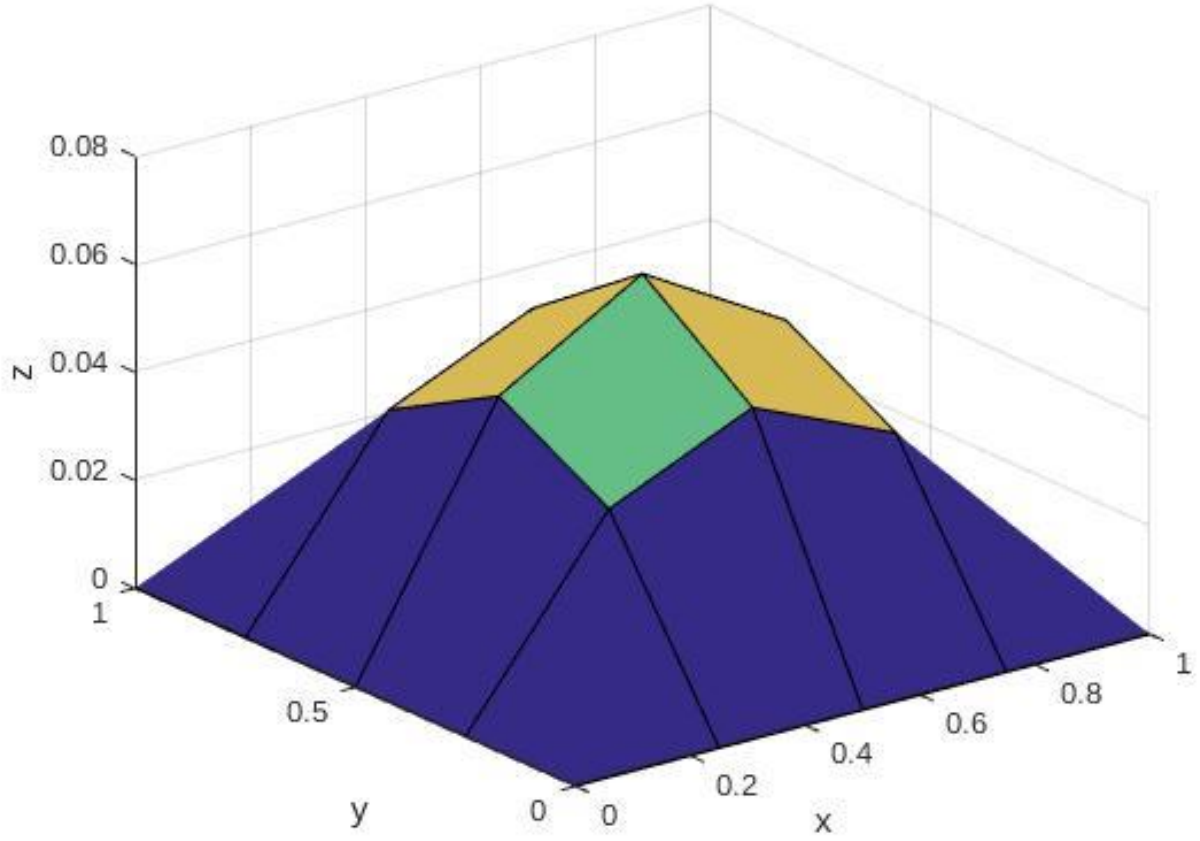
Ora che abbiamo costruito le strutture dati su cui lavorare possiamo scrivere la funzione e ottenerne il grafico:

```
>> Z = X.*(1-X).*Y.*(1-Y);
>> surf(X,Y,Z);
>> xlabel('x'); ylabel('y'); zlabel('z');
```

Si noti che la matrice Z avrà la proprietà che Z(i, j) = f (X(i, j), Y (i, j)).

Grafico di superficie

Otteniamo così:



La funzione surf

La funzione principale è quindi surf che ha la seguente sintassi

surf(Matrice1, Matrice2, Matrice3)

dove Matrice1 e Matrice2 sono matrici quadrate contenenti i valori delle variabili x e y, mentre Matrice3 è una matrice quadrata tale che l'elemento di indici i, j è dato da f (x(i), y(j)).
Esistono inoltre altre forme per utilizzare la funzione e si rimanda al solito all'help in linea.
Il grafico di superfici può essere personalizzato in diversi modi.
Oltre alle opzioni precedentemente discusse, ossia axis, title, xlabel, ylabel e zlabel, è possibile cambiare il colore, l'angolo di visualizzazione e realizzare grafici di tipo differente.

Grafico di superficie: esercizio

Come già notato in esempi precedenti per ottenere un grafico con un aspetto meno “spigoloso” è sufficiente aumentare i punti nell'ambito dell'inizializzazione delle strutture dati.
Proviamo a ricostruire il grafico ottenuto aumentando il numero di punti (15-20).
Ricordiamo:
clear all (puliamo la memoria)

- Definiamo m ed n
- Costruiamo x e y con linspace
- Costruiamo X e Y con meshgrid
- Costruiamo Z con la funzione
- Disegniamo il grafico

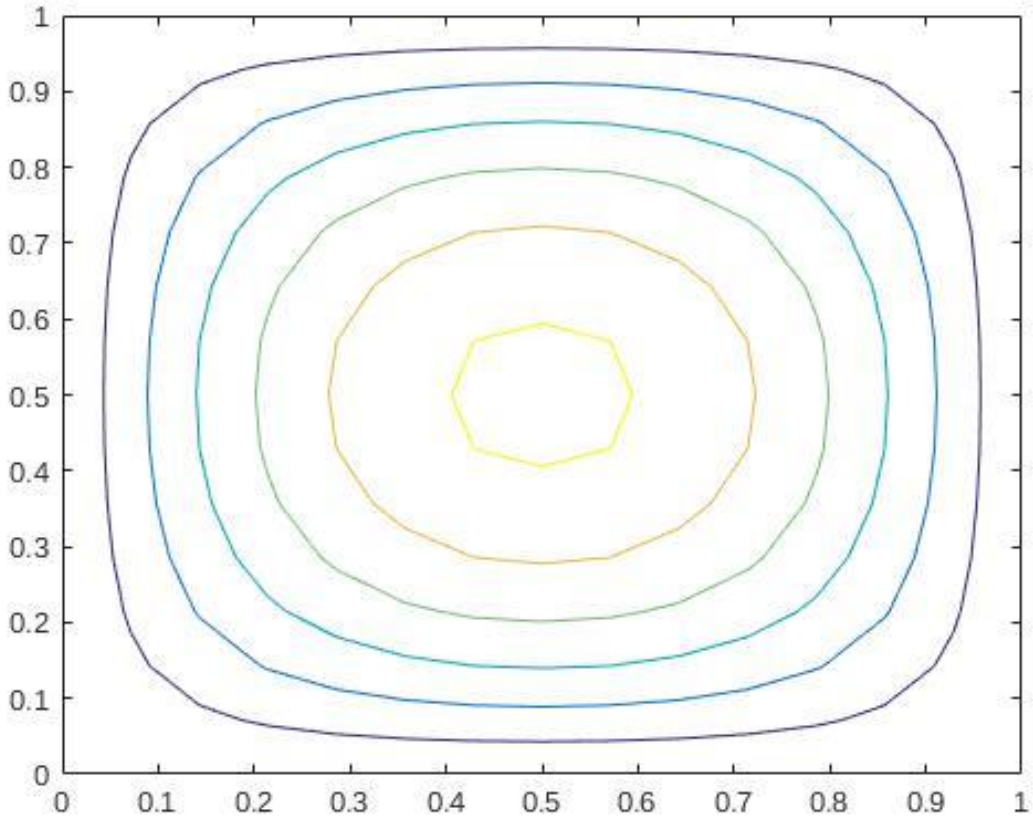
La funzione contour

Spesso risulta utile avere una rappresentazione bidimensionale della superficie, analoga a quella comunemente utilizzata nella realizzazione di mappe topografiche.
Utilizziamo le matrici X, Y e Z costruite in precedenza (ma aumentando m ed n a 15) con la funzione:

```
>> contour(X,Y,Z);
```

La funzione contour

Otteniamo così:



In Figura possiamo confrontare il risultato grafico ottenuto con le funzioni surf e contour. Il grafico mostra le curve di livello della funzione, ossia le curve sulle quali z = f (x, y) risulta costante.

Grafico di superficie: alcune funzioni

Funzione	Significato
view	cambia l'orientamento del grafico
colormap	cambia il colore del grafico
shading	cambia l'ombreggiatura del grafico
mesh	disegna un grafico a griglia
contour	disegna un grafico a curve di livello
surf	disegna un grafico di superficie

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