Introduzione a MATLAB/OCTAVE Parte 2

Plotting

Una importante funzione di Octave/Matlab è la possibilità esplorare i dati tramite funzioni di plotting

- Il problema è quello di produrre le strutture dati (array o matrici) che contengono le coordinate di punti da tracciare
- I sistemi di coordinate possono essere
  - Bidimensionali: grafici X → Y
  - Tridimensionali: grafici di superfici (X,Y) → Z

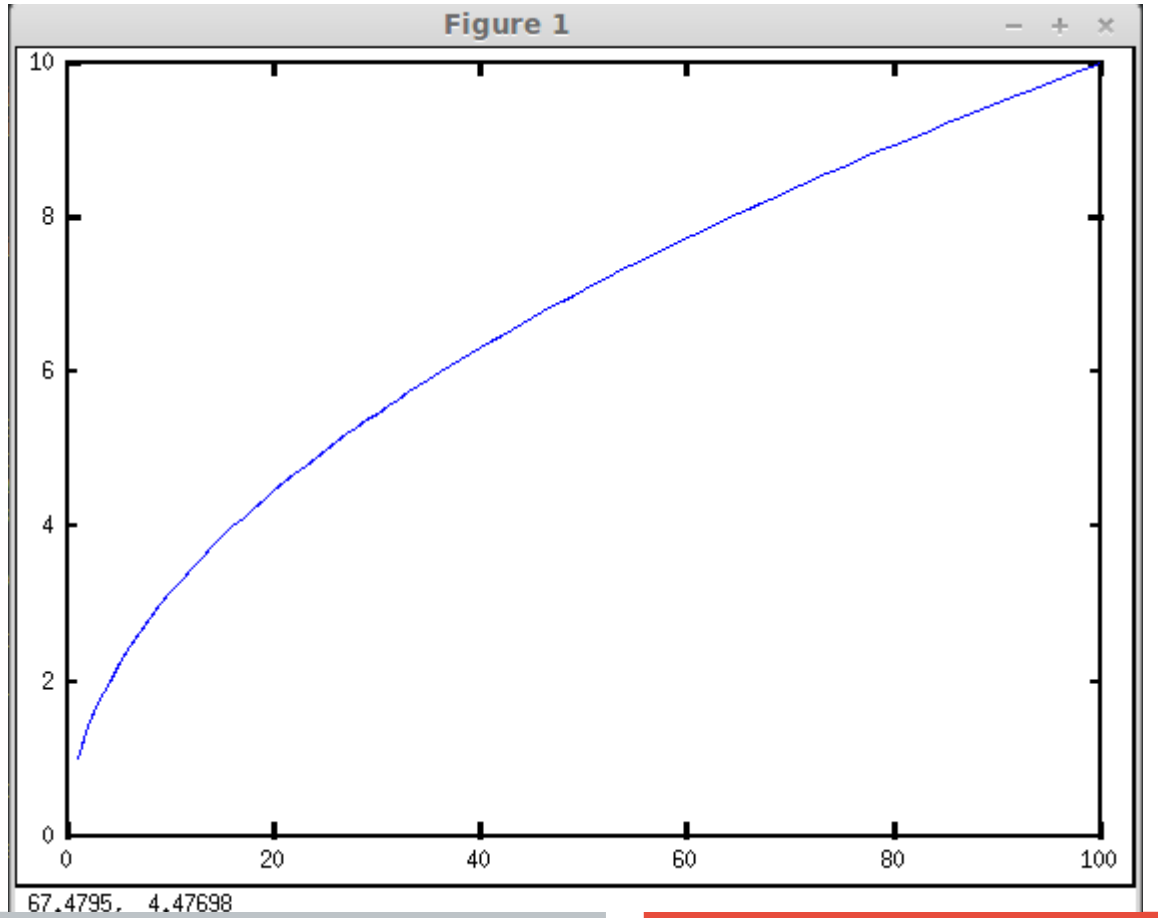
Plotting

Studi di funzione 2 dimensionali

- Raccogliere in un vettore N valori di X
- Raccogliere in un vettore gli N valori corrispondenti dell'ordinata Y
- Invocare la funzione plot passando i due vettori come argomenti

```
octave:1> x=[1:100];
octave:2> y=sqrt(x);
octave:3> plot(x,y)
```

Esempio: funzione `sqrt` (radice quadrata) per x=[0,100]



Plotting

La funzione plot accetta un numero variabile di argomenti

- Esegue il plot di diversi tracciati
- Il significato degli argomenti dipende dal numero di argomenti
- Anche in questo caso help plot vi da una descrizione completa
- Esempio

```
octave:1> x=[1:100];
octave:2> plot(x,sqrt(x),x,sqrt(2*x),x,sqrt(3*x))
```

Digitate `help plot` per avere informazioni sul comando di plotting che ha molte opzioni e diverse forme. In particolare

Gli oggetti grafici possono essere modificati e adattati per migliorare la resa grafica

Funzione per Generazione di Sequenze

E' comune che il problema sia invece posto così:

Dati due estremi x1 e x2 generare esattamente N punti nell'intervallo [x1, x2] (estremi compresi)

`linspace:` linearly spaced elements

```
x=linspace(0,1,100);
```

Plotting

Plot multipli:

Se il secondo argomento è una matrice allora plot usa le colonne come dati multipli per diversi grafici da tracciare rispetto al primo argomento x

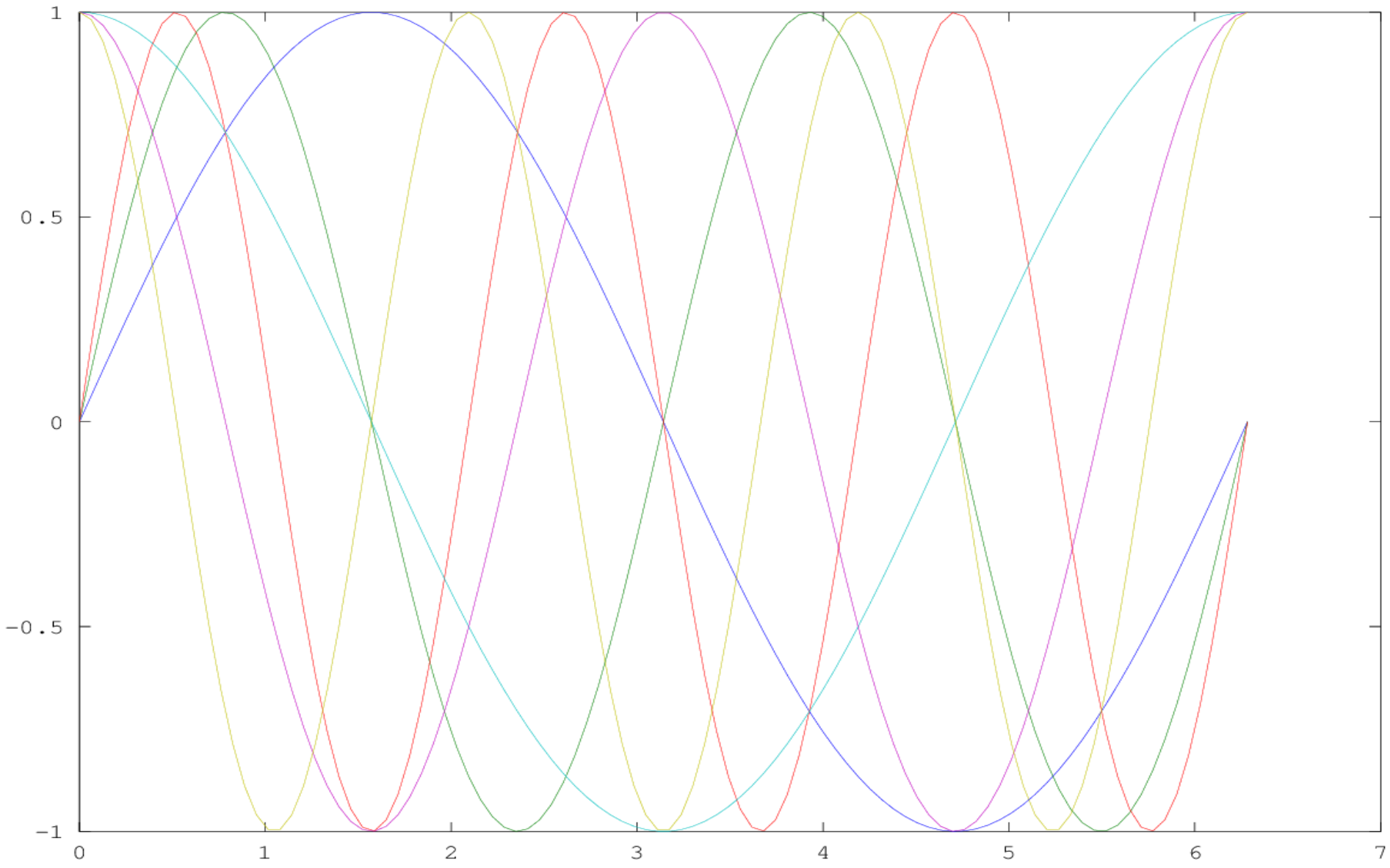
Esempio: tracciamento di 6 funzioni trigonometriche nell'intervallo [0, 2π]

```
x=linspace(0,2*pi,100)';
s=[sin(x) sin(2*x) sin(3*x) cos(x) cos(2*x) cos(3*x)];
plot(x,s)
```

Note:

- pi è una variabile 'built-in' che rappresenta il pi-greco
- Il carattere ; (punto e virgola) evita che il risultato venga stampato sullo schermo
- Esercizio: riprodurre questo esempio e usare il comando whos. Interpretare la dimensione delle variabili contenute nel workspace

Plotting



Plotting

Esercizio:

plot(x,s) funziona anche se il primo argomento x è una matrice di uguale dimensione di s.

Anche in questo caso plot lavora 'per colonne' disegnando il grafico prendendo i punti da ogni colonna corrispondente

Costruite una matrice

```
xm=[cos(3*x) cos(x) cos(2*x) sin(3*x) sin(x) sin(2*x)]
```

Quindi tracciate le curve con il comando plot(xm,s)

Plotting

Una delle forme della plot accetta un numero variabile di coppie di vettori

- In ogni coppia il primo array contiene le coordinate sull'asse x
- Il secondo contiene le coordinate sull'asse y

In ogni coppia le lunghezze dei vettori devono essere coincidenti

```
plot(X1,Y1,...,Xn,Yn)
```

Una forma di `plot` ammette un numero arbitrario di terne di argomenti

- Di ogni terna i primi due argomenti sono le coordinate x,y del grafico
- Il terzo argomento è una stringa di specificazione delle proprietà grafiche

```
plot(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn)
```

Plotting: line style specification

Una ‘line specification’ è una stringa che indica alcuni dei parametri grafici di ogni tracciato

Esempio: ‘--or’ significa

- ‘--’ il grafico è interpolato da una linea tratteggiata
- ‘o’ i punti del grafico sono marcati con un circolo
- ‘r’ il grafico deve essere rosso

```
>> pl = @(x) - 16*x.*x + 10*x - 2;
>> x=linspace(-1,1,32);
>> plot(x,pl(x),'--or')
```

Plotting: line specification

Parametri della linea di interpolazione

- ‘-’ linea continua
- ‘.’ linea punteggiata
- ‘.-’ punti e linee alternati

Parametri dei marker

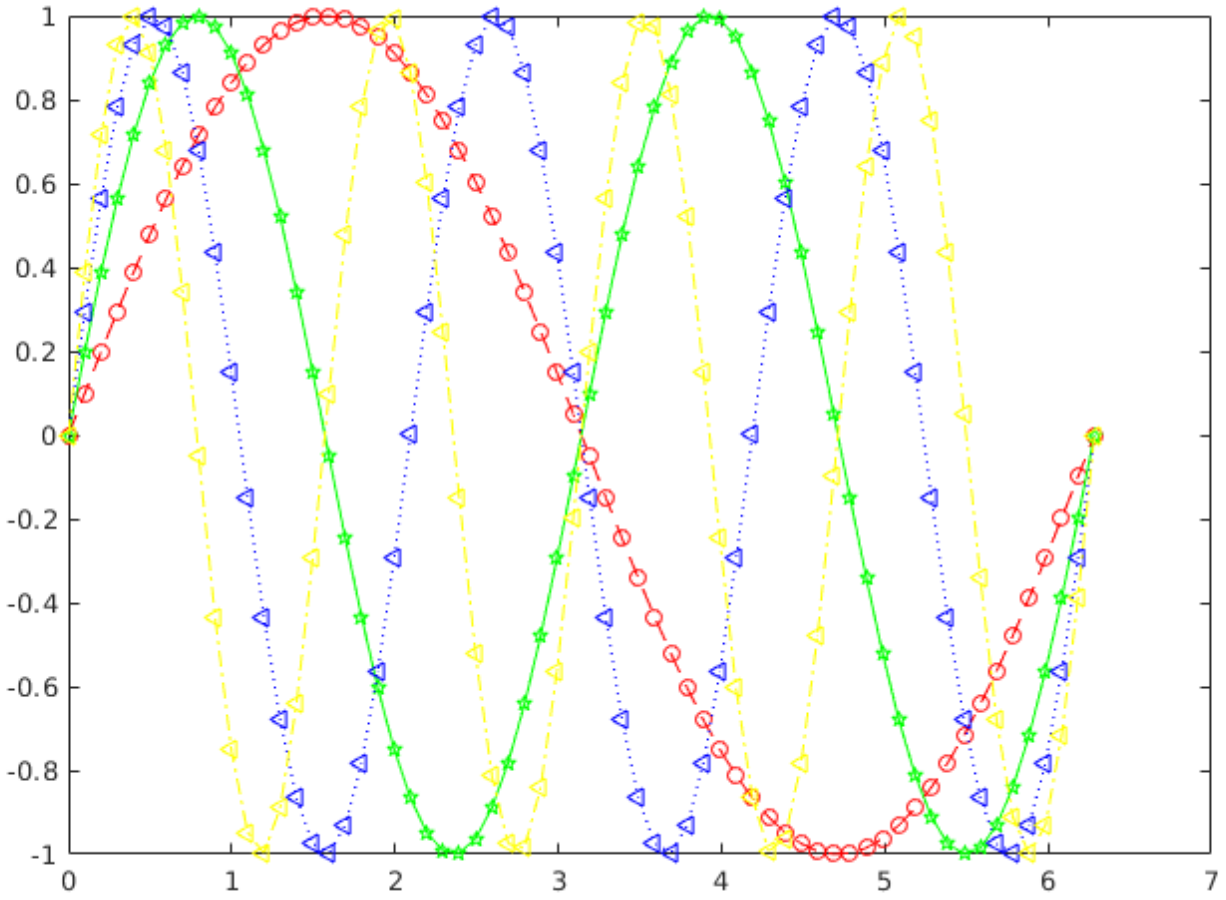
- ‘o’ circolo
- ‘p’ stella (pentagram)
- ‘x’ croce
- ‘>’ opp ‘<’ triangoli
- .....

Colori

- ‘r’ rosso
- ‘y’ giallo
- ‘g’ verde
- ‘b’ blu
- ...

```
% Generazione della variabile x
>> x=linspace(0,2*pi,64);

% Funzioni
>> s1=sin(x);
>> s2=sin(2*x);
>> s3=sin(3*x);
>> s4=sin(4*x);

% plot combinato con diversi marker, diversi stili di linea di
% interpolazione e colori

>> plot(x,s1,'--or',x,s2,'-pg',x,s3,':<b',x,s4,'-.<y')
```

Plotting: line specification



Scatterplots

La funzione 'scatter' è orientata a tracciare la dispersione di punti. Dal punto di vista del controllo grafico generalizza plot: accetta una lista di 'named arguments'

scatter(x,y):

traccia i punti le cui coordinate sono negli elementi corrispondenti di x,y. Quindi numel(x)==numel(y)

scatter(x,y,sz):

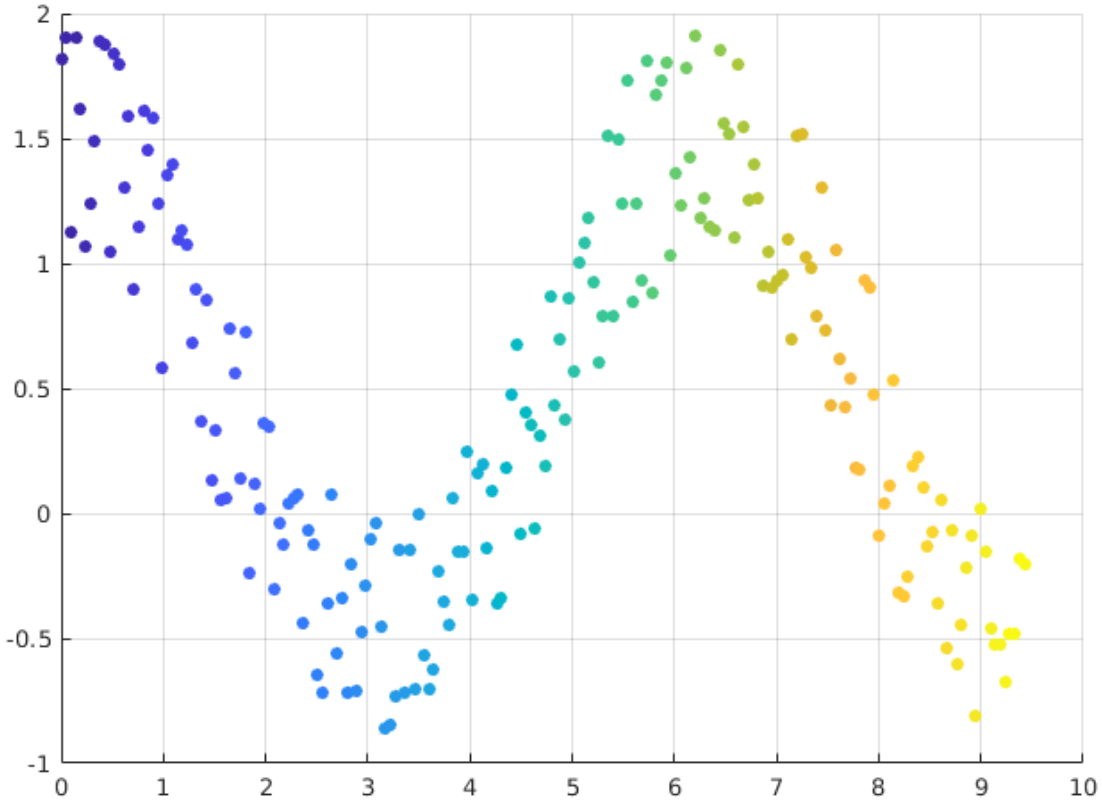
'sz' è la dimensione dei marker. Se 'sz' è uno scalare è la dimensione di tutti i marker. Se è un vettore deve essere numel(sz)==numel(x)==numel(y)

scatter(x,y,sz,c):

'c' è il colore del plot secondo il suo indice nella colormap corrente. Altrimenti può essere una tripletta di valore RGB. Se è un array o una matrice Nx3 (dove N=numel(x)) allora ogni punto ha un colore diverso

Scatterplots

```
>> x = linspace(0,3*pi,200);
>> y = cos(x) + rand(1,200);
>> c = linspace(1,10,length(x));
>> scatter(x,y,30,c,'filled')
>> grid
```



Scatterplots

scatter(x,y,sz,Name,Value)

- ‘Name’ è una stringa che determina un parametro grafico
- ‘Value’ il suo valore corrispondente
  - MarkerEdgeColor: colore del bordo del marker
  - MarkerFaceColor: colore di riempimento di un marker
  - LineWidth: spessore della linea di bordo del marker

Scatterplot

```
>> x = linspace(0,3*pi,200);
>> y = cos(x) + rand(1,200);
>> c = linspace(1,10,length(x));
>> scatter(x,y,30,'MarkerEdgeColor',[0 0.5 1],'MarkerFaceColor',[1 0.7 0])
>> grid
```

Subplot

`subplot(nrighe,ncolonne,indice)`

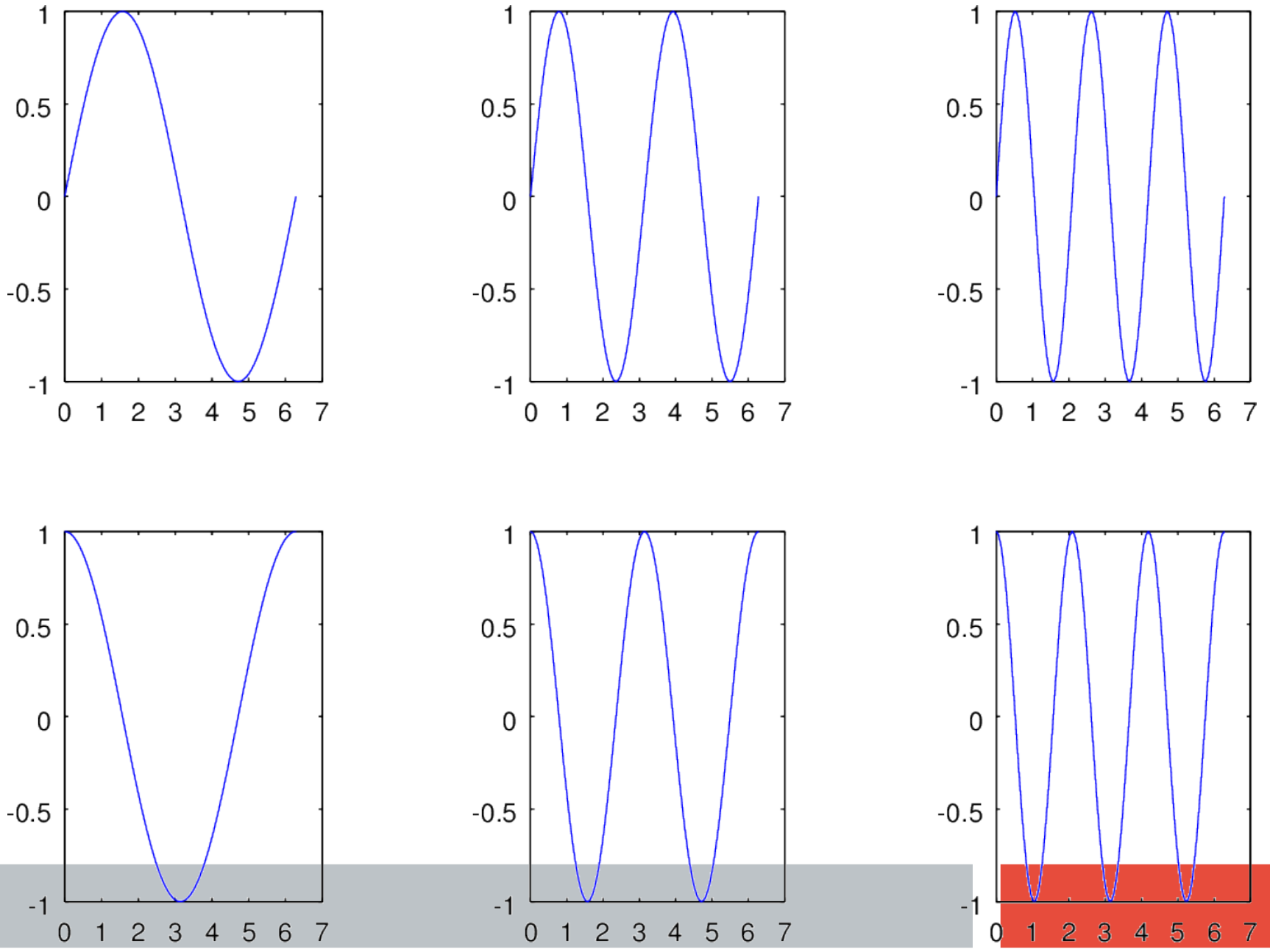
Suddivide lo spazio del plot in una griglia di `nrighe` e `ncolonne` e si prepara a disegnare il plot nella suddivisione avente identificata dal valore di `indice`

L'argomento `indice` è compreso tra 1 e `nrighe*ncolonne`

Subplot

```
x=linspace(0,2*pi,100)';
subplot(2,3,1)
plot(x,sin(x))

subplot(2,3,2)
plot(x,sin(2*x))

subplot(2,3,3)
plot(x,sin(3*x))

subplot(2,3,4)
plot(x,cos(x))

subplot(2,3,5)
plot(x,cos(2*x))

subplot(2,3,6)
plot(x,cos(3*x))
```

Subplot



Comando ‘`figure(n)`’

Senza argomenti apre una nuova finestra grafica

Ad una finestra grafica è associato automaticamente un numero intero

Se al comando viene passato come argomento un numero intero viene ‘attivata’ la finestra grafica associata al valore dell’argomento

Ogni output grafico automaticamente viene indirizzato sulla finestra grafica ‘attiva’

Traccia di punti nello spazio 3-dimensionale

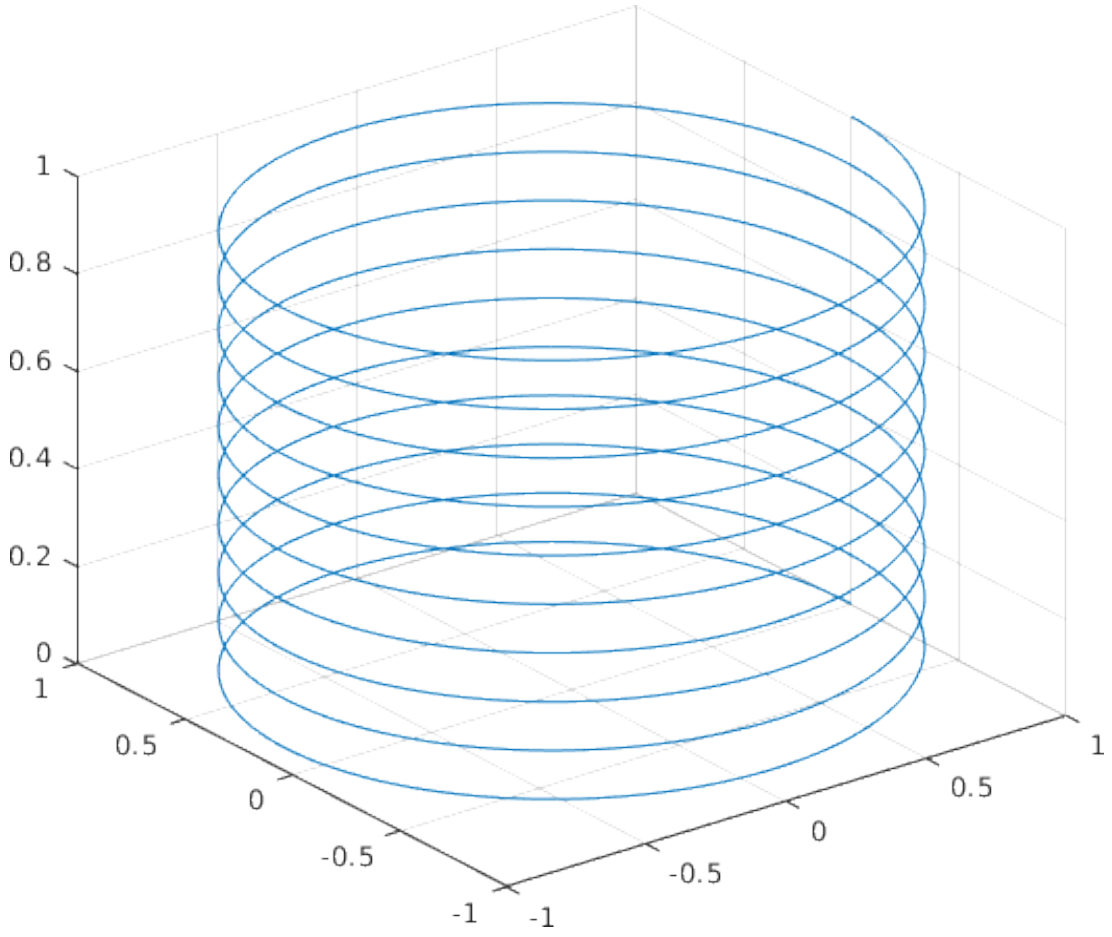
`plot3`: Generalizzazione a 3D della funzione plot

`plot3:` Usa come argomenti una successione di punti le cui coordinate sono memorizzate in array o matrici

3D Plotting

```
% creiamo un array con 4096 valori
% di un parametro t calcolati
% con spaziatura uniforme tra 0 e 2π
>> t=linspace(0,20*pi,16384);

% assegnamo alla variabile x i valori
% delle funzioni sin e cos per tutti
valori di t
>> y=sin(t);
>> x=cos(t);

% il valore della coordinata z è
proporzionale
% al valore del parametro t
>> z=t/(20*pi);

% tracciamo il grafico
>> plot3(x,y,z)

% tracciamo una griglia
>> grid
```

3D Plotting



Plot di superfici dove una coordinata (z) viene tracciata come funzione delle 2 variabili (x,y)

Procedura:

- Si deve generare 2 matrici
  - La prima contiene il valore di X per ogni punto da tracciare
  - La seconda contiene il valore di Y per ogni punto da tracciare
  - Si costruiscono con funzione `meshgrid`
  - La superficie si traccia con le funzione **`mesh`** o **`surf`**

3D Plotting

Il valore della variabile z è contenuto in una matrice con la stessa dimensione delle matrici prodotte da `meshgrid`

Si usa la funzione `mesh` per generare il plot 3D

Esercizio: disegnamo una superficie con funzione

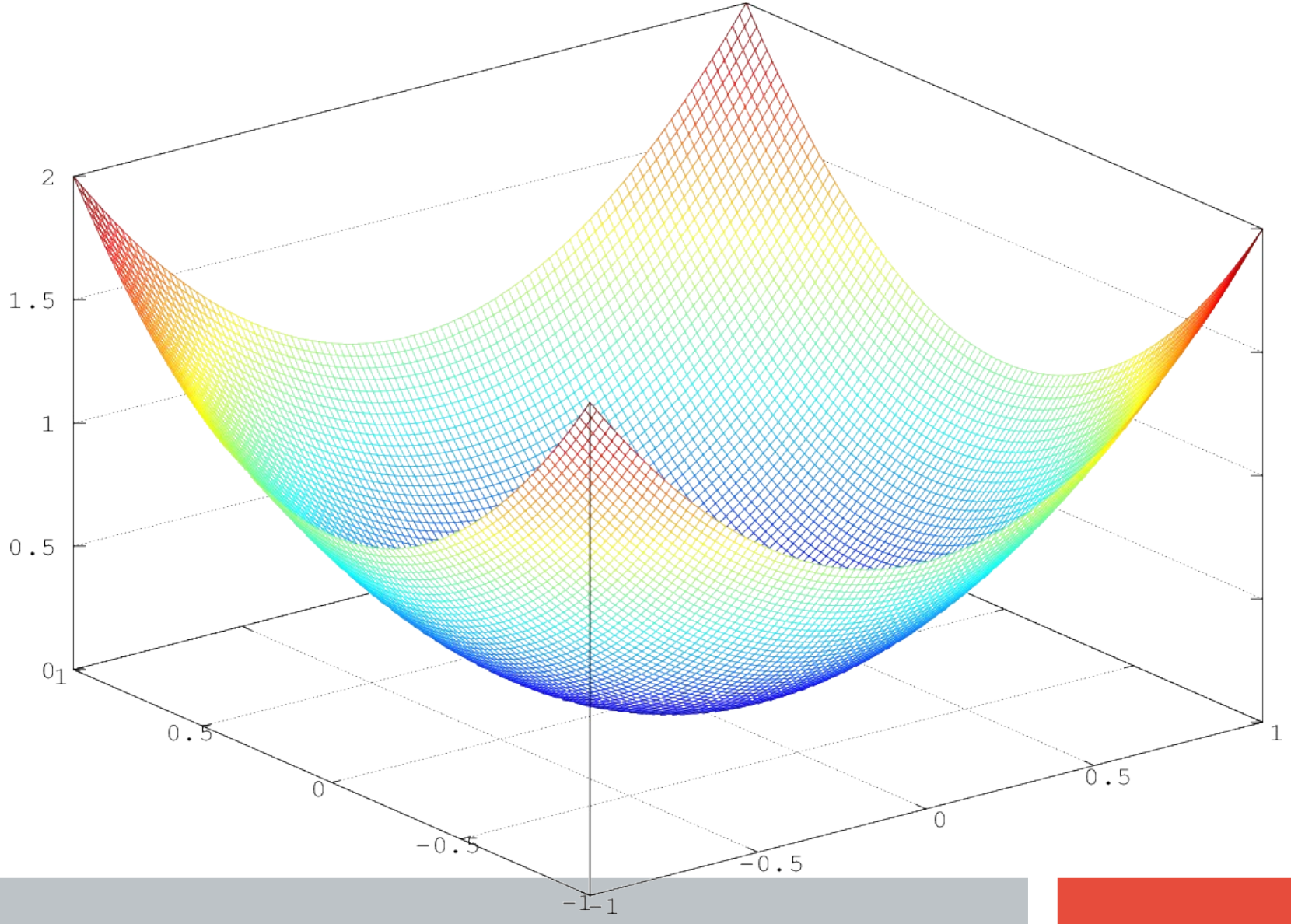
$$z = (x^2 + y^2)$$

Codice esempio

```
>> x = linspace(-1,1,512);
>> y = x;
>> [xx,yy] = meshgrid(x,y);
>> z = (xx.*xx + yy.*yy);
>> mesh(xx,yy,z)
```

- 1 Vegnono generate le coordinate x e y (100 punti compresi nell'intervallo [-1,1])
- 2 Le variabili xx e yy sono due matrici: contiengono rispettivamente il valore della coordinata x e y di ogni punto del piano delle variabili indipendenti
- 3 Si calcola la funzione z come somma dei quadrati dei valori delle variabili
- 4 La superficie viene tracciata

3D Plotting



Tentiamo un esempio più complesso

$$e^{-(x^2+y^2)/d} \cdot cos(2\pi\omega(x^2+y^2))$$

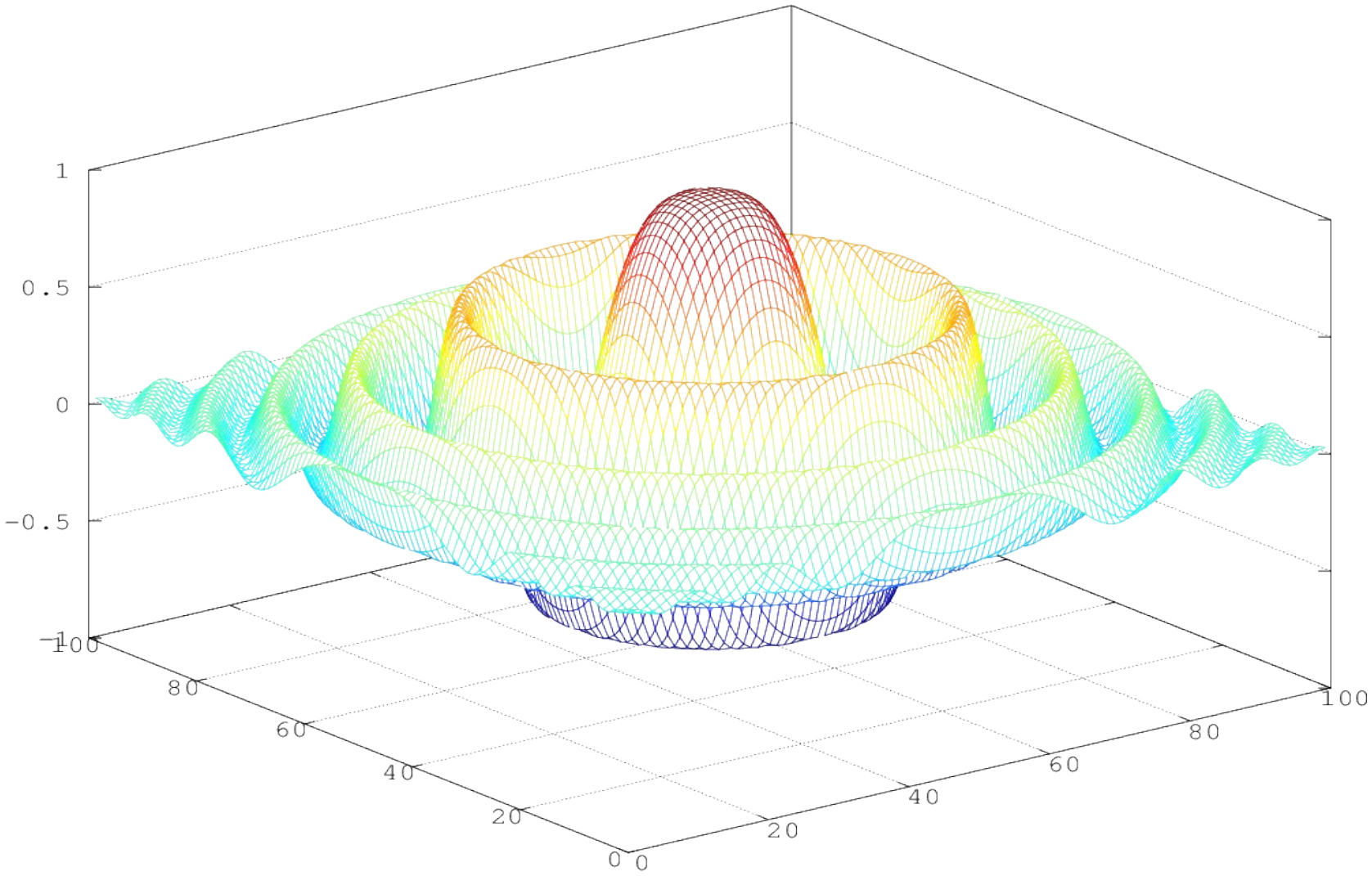
```
>> x=linspace(-1,1,100);
>> y=linspace(-1,1,100);
>> [xx,yy]=meshgrid(x,y);
>> z=(xx.*xx + yy.*yy);
>> d=0.5;
>> omega=3;
```

Tentiamo un esempio più complesso

$$e^{-(x^2+y^2)/d} \cdot cos(2\pi\omega(x^2+y^2))$$

```
>> x=linspace(-1,1,100);
>> y=linspace(-1,1,100);
>> [xx,yy]=meshgrid(x,y);
>> z=(xx.*xx + yy.*yy);
>> d=0.5;
>> omega=3;
>> f=exp(-z/d).*cos(2*pi*omega*z);
>> mesh(f)
```

3D Plot



3D Scatterplot

La funzione scatter3 generalizza la funzione bidimensionale

Esercizio: generare uno scatterplot dei colori di un'immagine RGB

```
>> rgbimg = imread('Tv16.tiff');
>> [X,map]=rgb2ind(rgbimg,16384);
>> scatter3(map(:,1),map(:,2),map(:,3),5,map)
```

Plotting di immagini

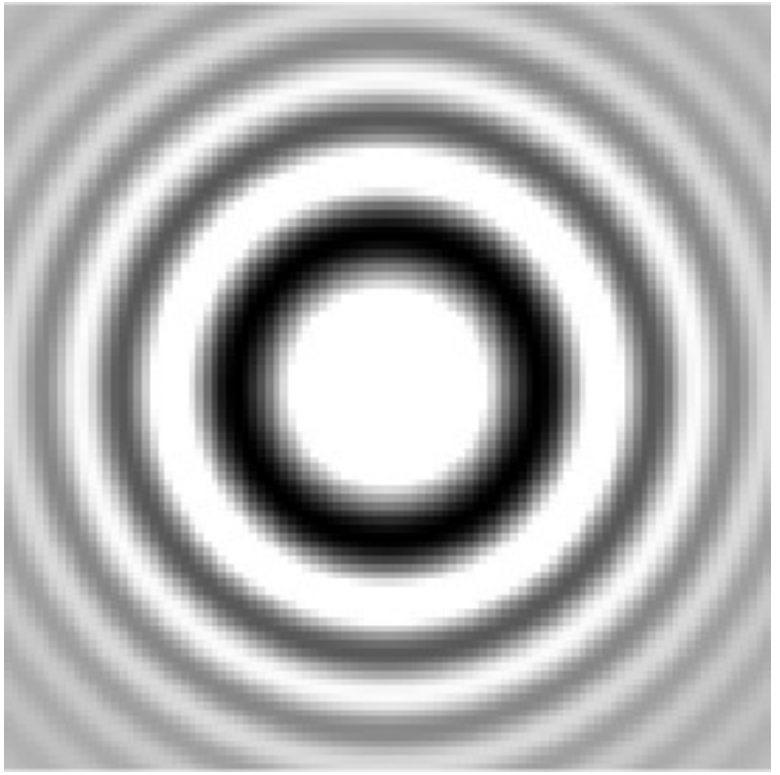
Nel nostro modello le immagini sono essenzialmente dei diagrammi 3D di intensità

- Ad un pixel sul piano x,y dell'immagine corrisponde un livello di intensità
- La scala delle intensità cambia a seconda della rappresentazione interna dei dati
- Per essere visualizzabile come immagine in ogni regione i valori sull'asse z devono essere ≥ 0

Plotting di Immagini

Visualizziamo la nostra superficie come differenza dell'altezza di un punto e del valore basso della superficie stessa

Calcoliamo il minimo di f



```
>> minimum = min(f(:));
>> imshow((f - min(f(:))/(max(f(:))-min(f(:)));
```

Lettura di Immagini da file

La funzione per leggere immagini da file è

```
imread('nome del file')
```

La funzione ritorna l'immagine in una matrice

```
>> riso = imread('riso.png');
>> class(riso)
ans = uint8
>> size(riso)
ans =
   1024    768      3
>> imshow(riso)
```

Fuzione zeros

La funzione zeros genera matrici di dimensione assegnata.

La sua forma più usata è

```
matrice_di_zeri=zeros(m,n);
```

Altre forme sono

```
matrice_di_zeri=zeros(m,n,p);
```

Per generare una matrice di zeri avente la stesse dimensione di una matrice data

```
matrice_di_zeri=zeros(size(matrice));
```

Per forzare la classe di rappresentazione si può accorciare la notazione usata nell'esempio

```
matrice_di_zeri=zeros(size(matrice),”uint8”);
```

```
...
>> red=uint8(zeros(size(riso)));
>> green=red;
>> blue=red;
>> red(:,:,1) = riso(:,:,1);
>> green(:,:,2) = riso(:,:,2);
>> blue(:,:,3) = riso(:,:,3);
>> subplot(1,3,1);
>> imshow(red);
>> subplot(1,3,2);
>> imshow(green);
>> subplot(1,3,3);
>> imshow(blue);
```

RGB Layers