

MATLAB-SIMULINK

Creazione di grafici 2D e 3D

Ing. Alessandro Pisano

apisano@unica.it

```
clear all  
s='set(gcf, 'position', [0,0,400,350]);';
```

Parte terza

Grafici avanzati








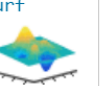





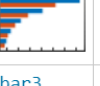




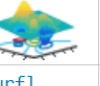





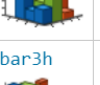




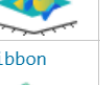







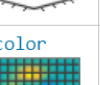



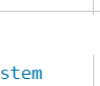

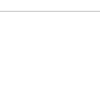










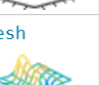



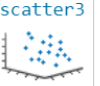



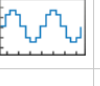











Alla pagina

https://www.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html

è riportato l'elenco completo delle varie tipologie di grafici avanzati che possono essere realizzati in Matlab.

Nel seguito uno screenshot da tale pagina. Cliccando su un tipo di grafico si accede ad una pagina dedicata con spiegazioni di dettaglio ed esempi di costruzione.

There are various functions that you can use to plot data in MATLAB®. This table classifies and illustrates the common graphics functions.

Line Plots	Data Distribution Plots	Discrete Data Plots	Geographic Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots	Volume Visualization	Animation	Images
<code>plot</code> 	<code>histogram</code> 	<code>bar</code> 	<code>geobubble</code> 	<code>polarplot</code> 	<code>contour</code> 	<code>quiver</code> 	<code>surf</code> 	<code>streamline</code> 	<code>animatedline</code> 	<code>image</code> 
<code>plot3</code> 	<code>histogram2</code> 	<code>barh</code> 	<code>geoplot</code> 	<code>polarhistogram</code> 	<code>contourf</code> 	<code>quiver3</code> 	<code>surfc</code> 	<code>streamslice</code> 	<code>comet</code> 	<code>imagesc</code> 
<code>stairs</code> 	<code>pie</code> 	<code>bar3</code> 	<code>geoscatter</code> 	<code>polarscatter</code> 	<code>contour3</code> 	<code>feather</code> 	<code>surf1</code> 	<code>streamparticles</code> 	<code>comet3</code> 	
<code>errorbar</code> 	<code>pie3</code> 	<code>bar3h</code> 		<code>compass</code> 	<code>contourslice</code> 		<code>ribbon</code> 	<code>streamribbon</code> 		
<code>area</code> 	<code>scatter</code> 	<code>pareto</code> 		<code>ezpolar</code> 	<code>fcontour</code> 		<code>pcolor</code> 	<code>streamtube</code> 		
<code>stackedplot</code> 	<code>scatter3</code> 	<code>stem</code> 					<code>fsurf</code> 	<code>coneplot</code> 		
<code>loglog</code> 	<code>scatterhistogram</code> 	<code>stem3</code> 					<code>fimplicit3</code> 	<code>slice</code> 		
<code>semilogx</code> 	<code>spy</code> 	<code>scatter</code> 					<code>mesh</code> 			
<code>semilogy</code> 	<code>plotmatrix</code> 	<code>scatter3</code> 					<code>meshc</code> 			
<code>fplot</code> 	<code>heatmap</code> 	<code>stairs</code> 					<code>meshz</code> 			
<code>fplot3</code> 	<code>wordcloud</code> 						<code>waterfall</code> 			
<code>fimplicit</code> 	<code>parallelplot</code> 						<code>fmesh</code> 			

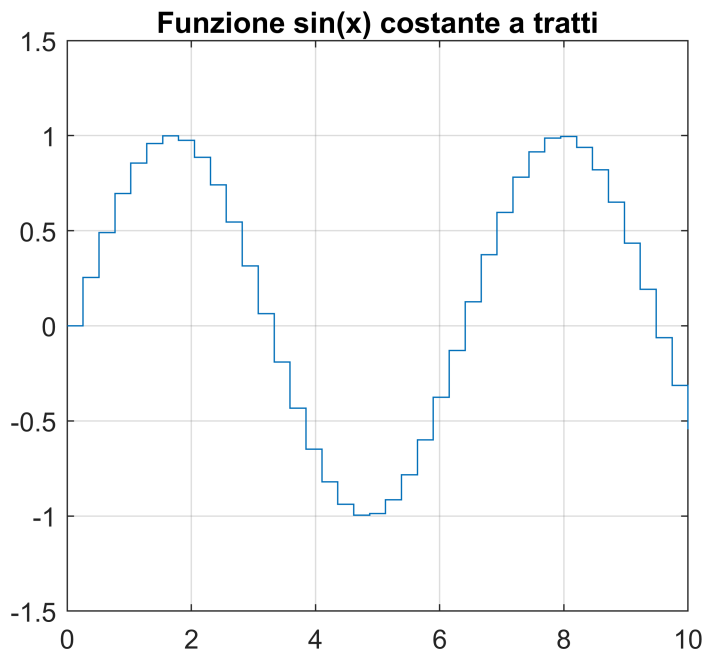
Vediamo alcuni esempi di creazione di alcune tipologie di grafici di più frequente utilizzo.

STAIRS

La funzione `stairs` ha modalità di impiego in tutto e per tutto analoghe a quelle della funzione `plot`. La differenza è che la funzione `stairs` interpola i punti attraverso una curva costante a tratti, e non attraverso una spezzata come la funzione `plot`

Vediamo un esempio

```
x=linspace(0,10,40);  
stairs(x,sin(x)), grid  
eval(s)  
ylim([-1.5 1.5])  
title('Funzione sin(x) costante a tratti')
```

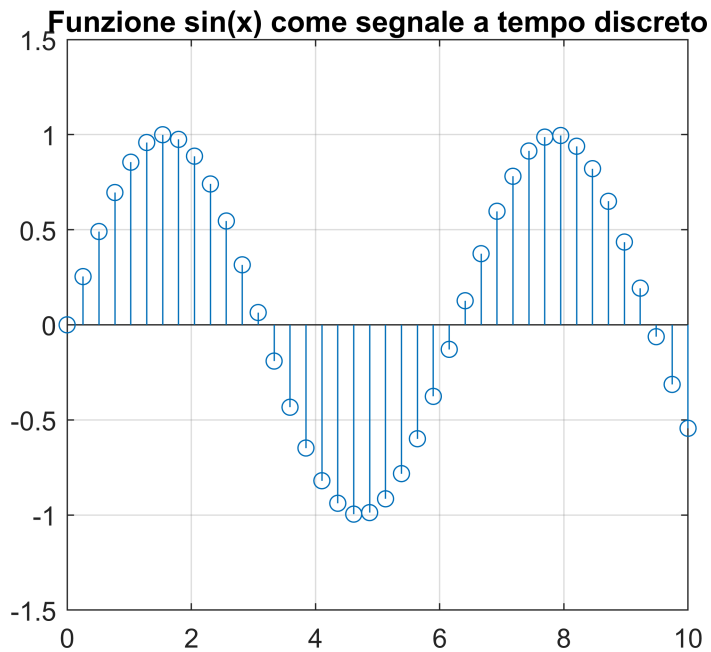


Tutto quanto visto in precedenza in merito alla aggiunta di label, titolo, legenda, ed altre personalizzazioni quali dimensione dei font, colore e tipo di linea, creazione di grafici multipli, ecc ecc continua a valere con modalità inalterate rispetto alla funzione `plot`.

STEM

La funzione `stem` colloca in corrispondenza dei punti un marker (di default un cerchietto) e inserisce un segmento verticale che unisce ciascun punto con l'asse delle ascisse. Utile per rappresentare segnali a tempo discreto.

```
x=linspace(0,10,40);  
stem(x,sin(x)), grid  
ylim([-1.5 1.5])  
title('Funzione sin(x) come segnale a tempo discreto')
```



SEMILOGX, SEMILOGY, LOGLOG

Queste tre funzioni consentono di realizzare dei grafici in cui l'asse delle ascisse, quello delle ordinate, o entrambi, siano graduati in scala logaritmica.

Si consulti l'articolo:

<https://smarcell1961.blogspot.com/2020/10/uno-strumento-del-demonio-la-scala.html>

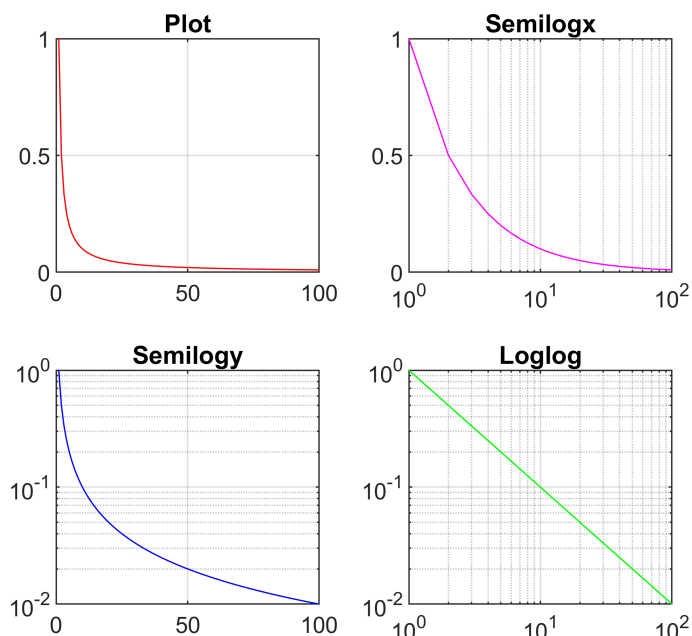
di cui segue uno screenshot.



Il seguente codice genera un grafico multiplo in cui la funzione $f(x) = \frac{1}{x}$ viene graficata nell'intervallo

$x \in [1, 1000]$ utilizzando la scala naturale in entrambi gli assi (comando plot), la scala logaritmica nell'asse delle ascisse (comando semilogx), la scala logaritmica nell'asse delle ordinate (comando semilogy), e la scala logaritmica in entrambi gli assi (comando loglog)

```
x=1:100; y=1./x;
subplot(2,2,1), plot(x,y,'r'), title('Plot'),grid
subplot(2,2,2), semilogx(x,y,'m'), title('Semilogx'),grid
subplot(2,2,3), semilogy(x,y,'b'), title('Semilogy'),grid
subplot(2,2,4), loglog(x,y,'g'), title('Loglog'), grid
```



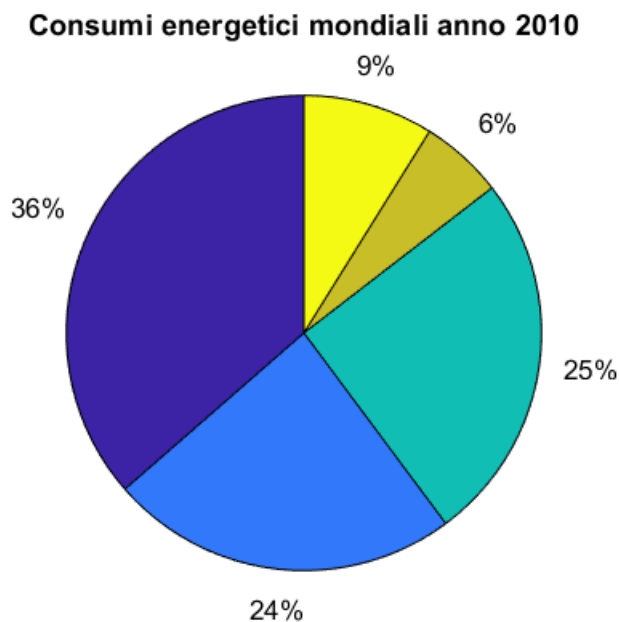
Diagrammi a torta (funzioni PIE e PIE3)

Vediamo alcuni esempi di creazione di diagrammi a torta prendendo spunto dai seguenti dati inerenti i consumi energetici mondiali suddivisi per fonte energetica di provenienza (in Mtep)

	2003	2010	2020	2030
Petrolio	4085	4677	5312	6025
Gas naturale	2497	3052	3934	4785
Carbone	2530	3246	4034	4927
Nucleare	668	728	829	874
Rinnovabili	824	1139	1338	1572
Totale mondo	10602	12844	15447	18184

Costruiamo in particolare un diagramma a torta relativo ai consumi dell'anno 2010

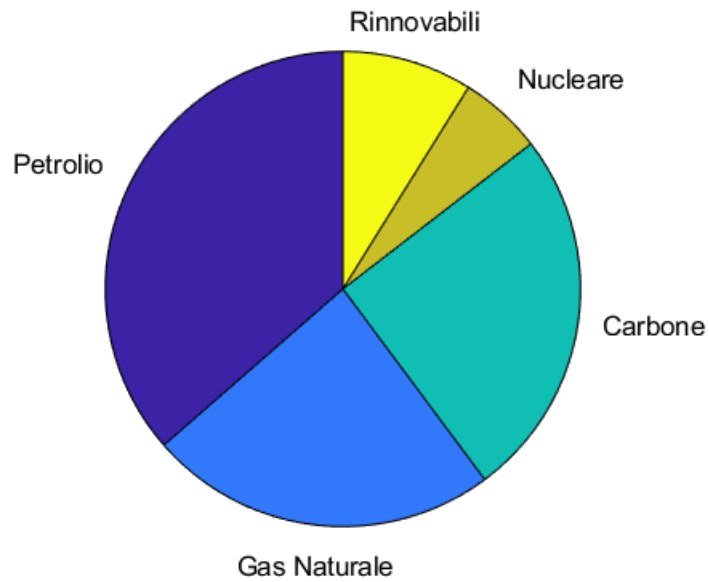
```
x=[4677 3052 3246 728 1139];  
figure  
pie(x), title('Consumi energetici mondiali anno 2010')  
eval(s)
```



Il diagramma mostra affianco alle "fette" della torta la relativa percentuale sul totale. Il seguente codice fa comparire affianco alle fette una stringa descrittiva

```
x=[4677 3052 3246 728 1139];  
labels = {'Petrolio', 'Gas Naturale', 'Carbone', 'Nucleare', 'Rinnovabili'};  
pie(x,labels), title('Consumi energetici mondiali anno 2010')
```

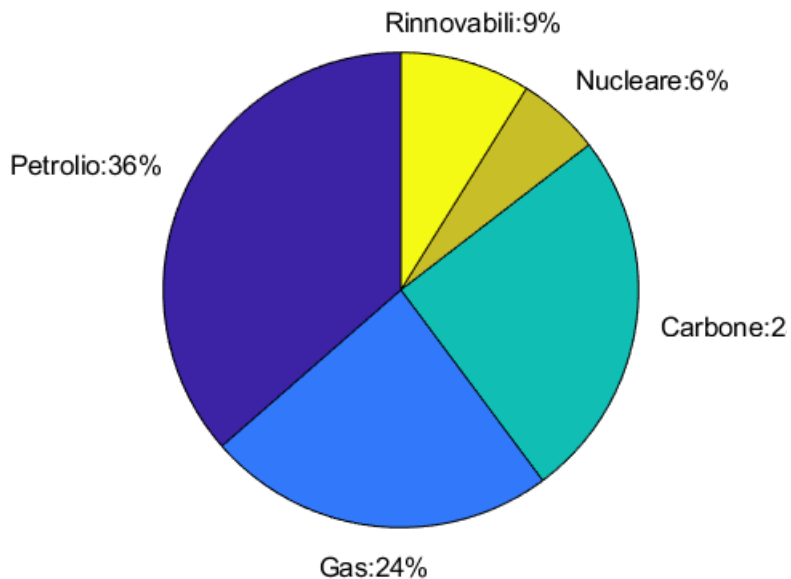
Consumi energetici mondiali anno 2010



Il seguente codice permette di visualizzare affianco alle fette sia la stringa che la relativa percentuale

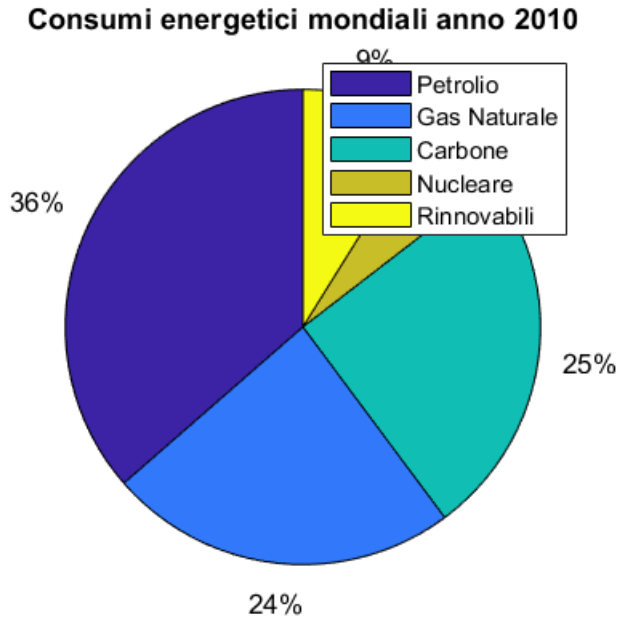
```
x=[4677 3052 3246 728 1139];  
h=pie(x);  
textObjs = findobj(h,'Type','text');  
oldStr = get(textObjs,{'String'});  
Names = {'Petrolio:','Gas:','Carbone:','Nucleare:','Rinnovabili:'};  
newStr = strcat(Names,oldStr);  
set(textObjs,{'String'},newStr);  
title('Consumi energetici mondiali anno 2010')
```

Consumi energetici mondiali anno 2010



In alternativa, si può abbinare al diagramma una legenda

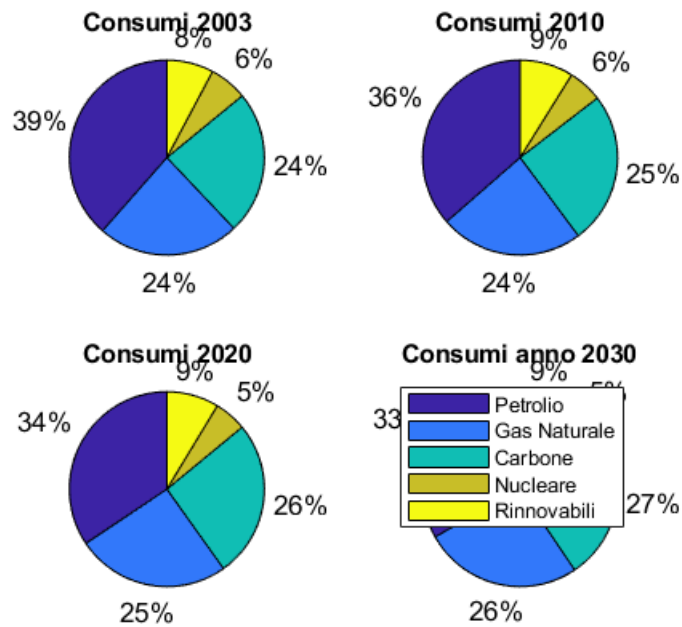
```
x=[4677 3052 3246 728 1139];  
figure  
pie(x)  
eval(s)  
legend('Petrolio','Gas Naturale','Carbone','Nucleare','Rinnovabili');  
title('Consumi energetici mondiali anno 2010')
```



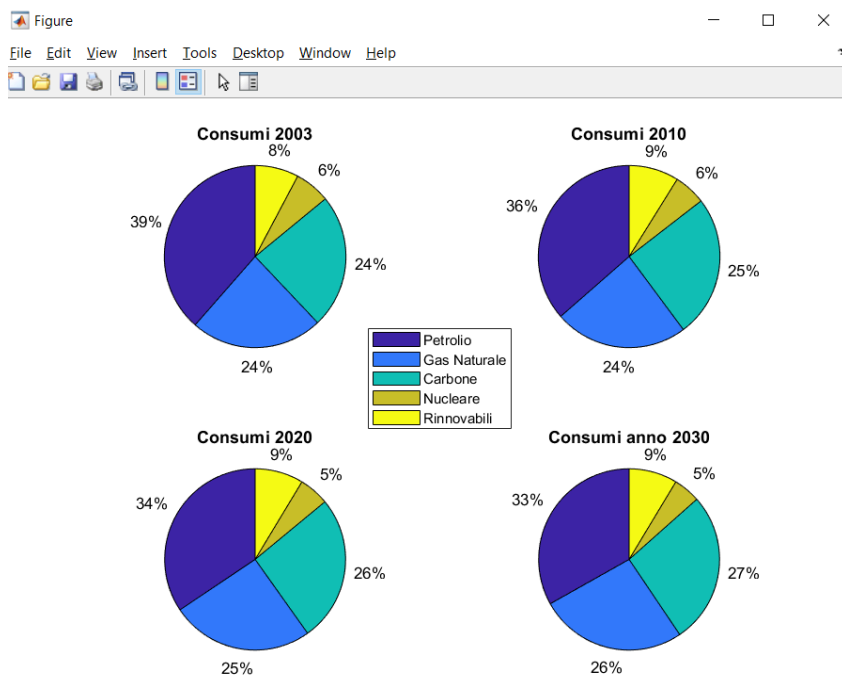
Costruendo il diagramma in uno script (o esportandolo in una finestra esterna con il relativo pulsante) risulta molto semplice spostare la legenda trascinandola con il mouse in modo che non si sovrapponga al diagramma (provare).

Costruiamo un grafico multiplo che confronti i dati delle 4 annualità di riferimento

```
y2003=[4085 2497 2530 668 824];  
y2010=[4677 3052 3246 728 1139];  
y2020=[5312 3934 4034 829 1338];  
y2030=[6025 4785 4927 874 1572];  
  
subplot(2,2,1)  
pie(y2003), title('Consumi 2003')  
subplot(2,2,2)  
pie(y2010), title('Consumi 2010')  
subplot(2,2,3)  
pie(y2020), title('Consumi 2020')  
subplot(2,2,4)  
pie(y2030), title('Consumi anno 2030')  
legend('Petrolio','Gas Naturale','Carbone','Nucleare','Rinnovabili');
```

Esportando il diagramma in una finestra esterna con il relativo pulsante risulta molto semplice ridimensionarlo e spostare la legenda in modo da creare un grafico maggiormente leggibile come quello riportato nel seguito

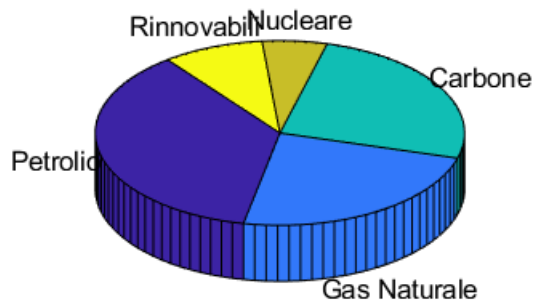


Con l'istruzione `pie3` si genera una versione 3D del diagramma

```
x=[4677 3052 3246 728 1139];
labels = {'Petrolio', 'Gas Naturale', 'Carbone', 'Nucleare', 'Rinnovabili'};
figure
```

```
pie3(x,labels),
title('Consumi energetici mondiali anno 2010')
eval(s)
```

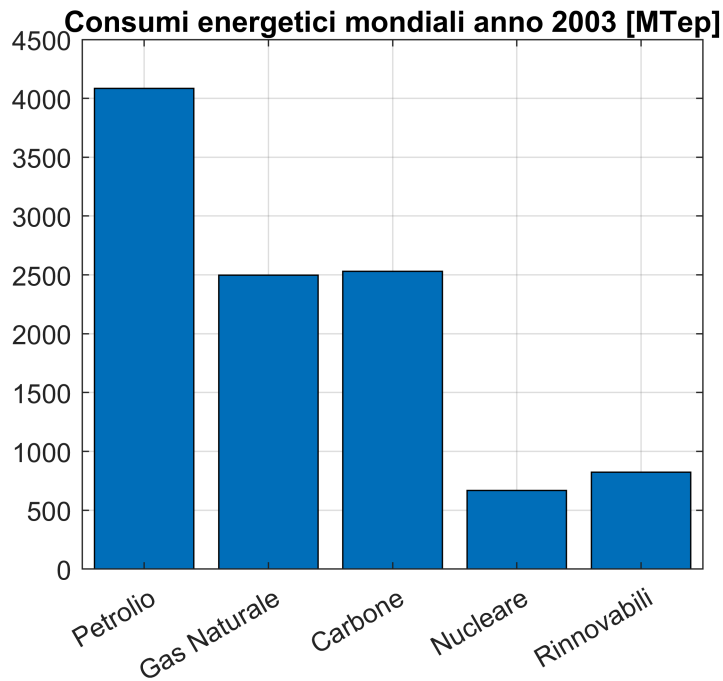
Consumi energetici mondiali anno 2010



Diagrammi a barre

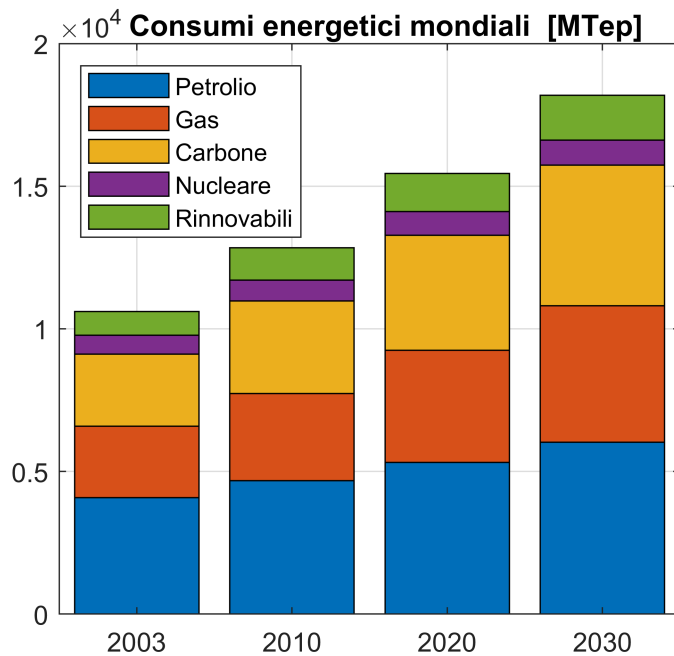
Realizziamo un primo diagramma riferito ai dati del 2003.

```
X = categorical({'Petrolio','Gas Naturale','Carbone', 'Nucleare', 'Rinnovabili'});
X = reordercats(X,{'Petrolio','Gas Naturale','Carbone', 'Nucleare', 'Rinnovabili'});
y2003=[4085 2497 2530 668 824];
bar(X,y2003), grid
title('Consumi energetici mondiali anno 2003 [Mtep]')
```



Ora realizziamo un diagramma aggregato che copra le 4 annualita di riferimento e per ciascuna annualita mostri con colori differenti le quote di consumo energetico relative alle varie fonti.

```
X = categorical({'2003','2010','2020','2030'});
X = reordercats(X,{'2003','2010','2020','2030'});
y2003=[4085 2497 2530 668 824];
y2010=[4677 3052 3246 728 1139];
y2020=[5312 3934 4034 829 1338];
y2030=[6025 4785 4927 874 1572];
Y=[y2003;y2010;y2020;y2030];
bar(X,Y,'stack'), grid
legend('Petrolio','Gas','Carbone','Nucleare','Rinnovabili','Location','NorthWest')
title('Consumi energetici mondiali [MTep]')
```



Grafici 3D

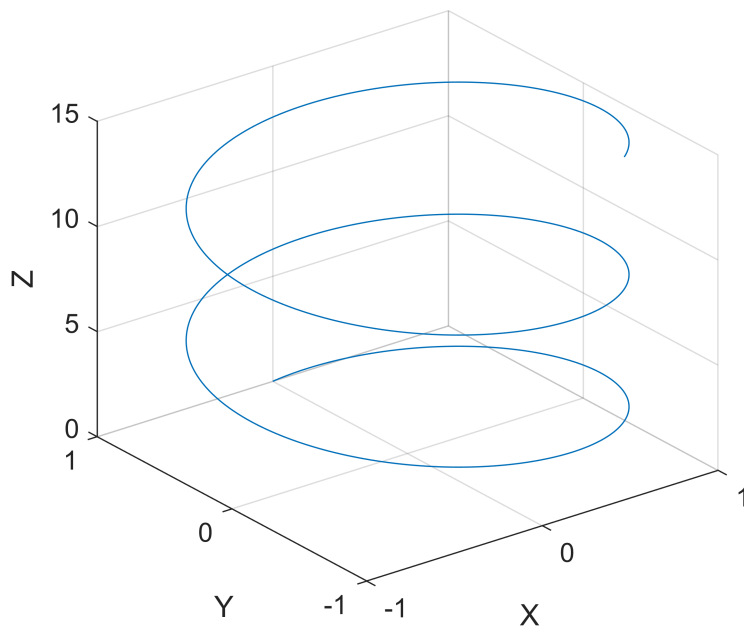
Curve parametriche in 3D

L'analogo 3D della funzione `plot` è la funzione `plot3`. L'impiego standard di tale funzione prevede che si passino come argomenti di ingresso 3 vettori x , y e z di pari dimensione N . La funzione `plot3` genera una spezzata che interpola i punti aventi coordinate $(x(i), y(i), z(i))$, $i = 1, 2, \dots, N$.

Generiamo un grafico 3D della curva parametrica

$$\begin{aligned} x &= \sin(t) \\ y &= \cos(t) \quad 0 \leq t \leq 15 \\ z &= t \end{aligned}$$

```
t = 0:0.01:15;
x=sin(t); y=cos(t); z=t;
plot3(x,y,z), grid
eval(s)
xlabel('X')
ylabel('Y')
zlabel('Z')
```



Grafici di superfici

Consideriamo come problema generale la visualizzazione del grafico di una superficie $z = f(x, y)$ con le variabili x ed y appartenenti ad un dominio rettangolare

$$x_{\min} \leq x \leq x_{\max} \quad y_{\min} \leq y \leq y_{\max}$$

Sviluppiamo un esempio di creazione grafico considerando la superficie $z = \text{sinc}(\sqrt{x^2 + y^2}) = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$

Il passo preliminare è creare due vettori x ed y che "discretizzino" (o se si preferisce "campionino") gli intervalli $x_{\min} \leq x \leq x_{\max}$ e $y_{\min} \leq y \leq y_{\max}$ in modo analogo a quanto fatto per la creazione dei grafici 2D.

Cio può essere fatto con una sintassi del tipo

```
x=linspace(xmin,xmax,Nx)
y=linspace(ymin,ymax,Ny)
```

avendo cura che N_x ed N_y siano sufficientemente grandi in rapporto alle dimensioni degli intervalli, in modo che la discretizzazione avvenga in maniera sufficientemente "densa". Fatto cio, si creano due matrici X ed Y utilizzando la seguente sintassi:

```
[X,Y]=meshgrid(x,y)
```

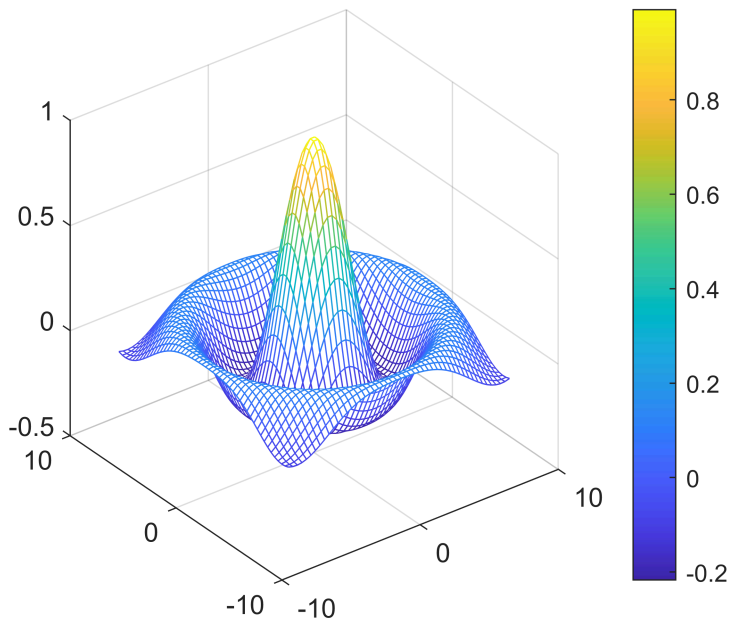
Se i vettori x ed y sono identici (cioè se il dominio rettangolare è simmetrico rispetto all'origine del piano xy) si può utilizzare la seguente sintassi semplificata.

```
[X,Y]=meshgrid(x)
```

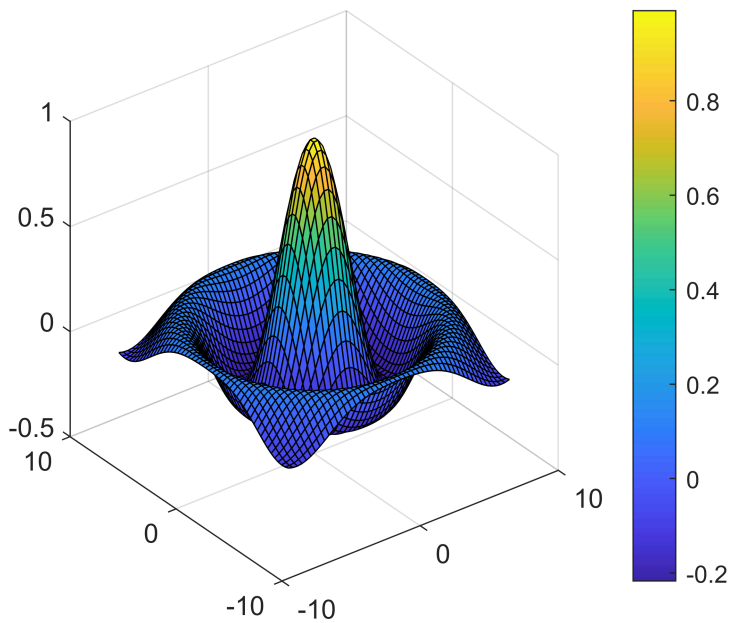
Si costruisce successivamente una matrice Z replicando l'espressione analitica della funzione con le matrici X ed Y come argomenti ($Z = f(X, Y)$) e si crea il grafico utilizzando a scelta la funzione mesh(X,Y,Z) o la funzione surf(X,Y,Z) .

Sviluppiamo il codice Matlab riferito alla superficie $z = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$, che desideriamo graficare nel dominio rettangolare simmetrico $-8 \leq x \leq 8$, $-8 \leq y \leq 8$

```
N=50;  
x=linspace(-8,8,N);  
[X,Y] = meshgrid(x);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)  
colorbar
```

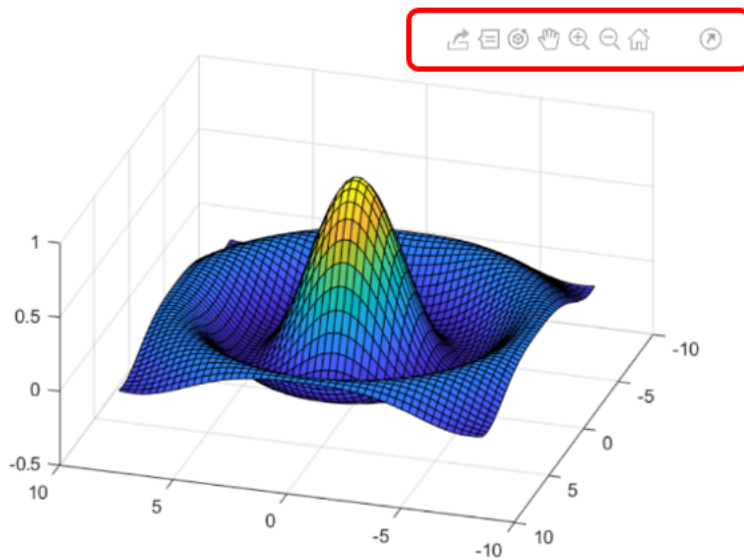


```
surf(X,Y,Z)  
colorbar
```

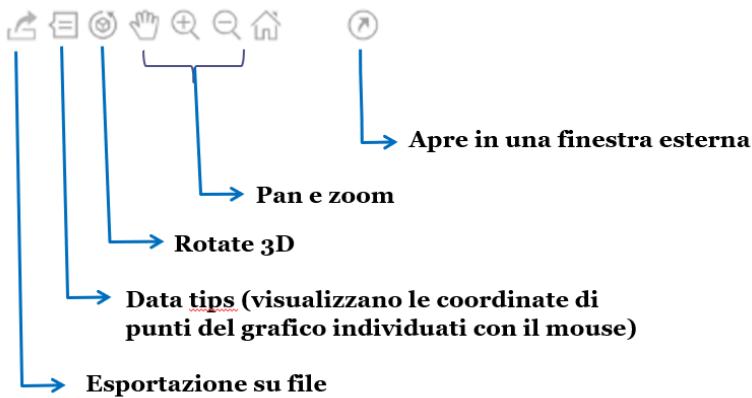


I grafici 3D generati sono abbastanza simili, La curva viene colorata in modo che a valori diversi della funzione corrispondano tonalita differenti. Le due funzioni si differenziano nel modo in cui inseriscono tali tonalita: la funzione mesh nelle linee che interpolano i punti, la funzione surf nelle regioni individuate dalle linee. La funzione mesh non colora le regioni, mentre la funzione surf traccia le linee interpolanti di default in colore nero. Se si aumenta la risoluzione incrementando N da 50 a 200 si osserva come risulti piu opportuno usare il comando mesh.

Se si porta il mouse sopra una figura vengono visualizzati alcuni pulsanti (si veda la seguente immagine)

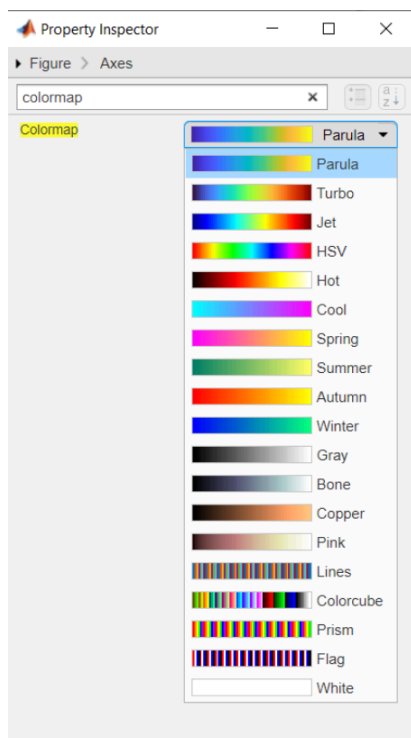


Funzioni dei pulsanti (testarli)



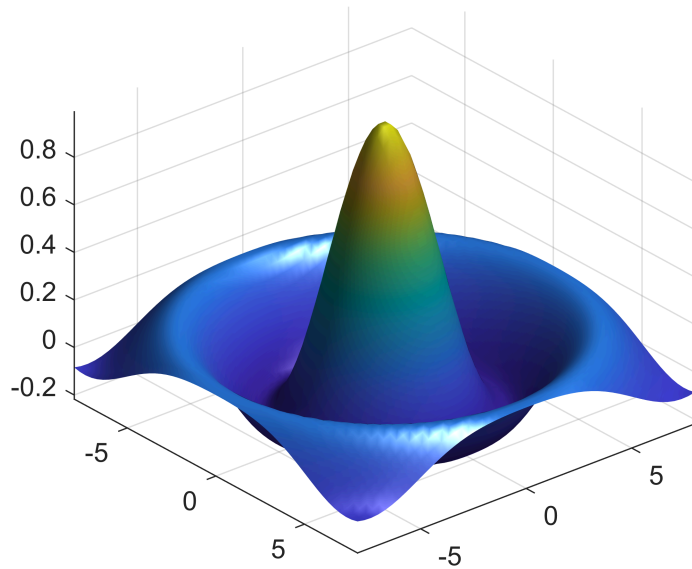
Sosituendo mesh e surf con meshc e surfc vengono aggiunte al grafico le curve di livello nel piano xy (provare).

Per modificare le tonalità di colore aprire l'immagine in una finestra esterna, selezionare Property Inspector dal menu View, e nella casella di ricerca scrivere colormap. Si accede ad una ampia varieta di scelta (v figura).



Mostriamo un codice di esempio che impiega opzioni avanzate di configurazione della funzione surf per produrre un rendering di qualita superiore.

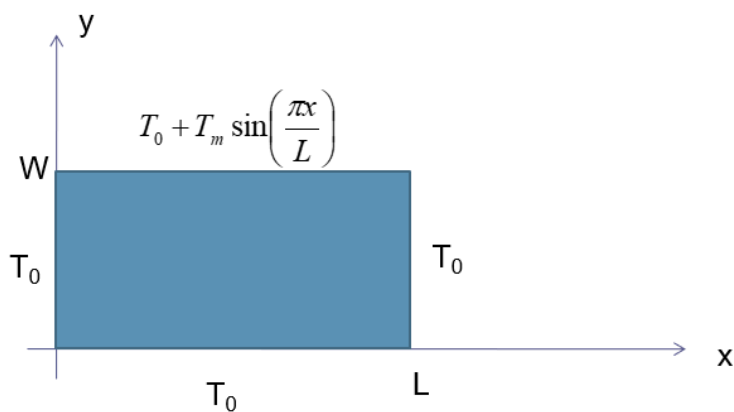
```
N=50;
x=linspace(-8,8,N);
[X,Y] = meshgrid(x);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
surf(X,Y,Z,'Facecolor','interp','EdgeColor','none','FaceLighting','phong')
axis tight
view(50,35)% angolo di ripresa (AZ/EL)
```

Concludiamo con la realizzazione di un grafico che visualizzi in 3D la distribuzione di temperatura a regime nella generica sezione di una sbarra filiforme di lunghezza infinita soggetta a particolari condizioni al contorno in termini di temperatura ai bordi della sezione. Si veda la dispensa "Function files e anonymous functions".



Sezione generica:



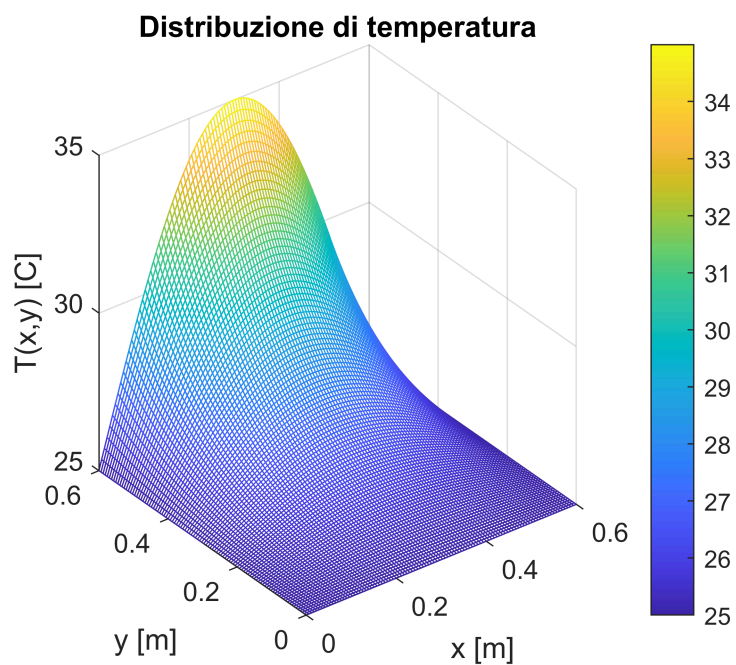
L'espressione analitica della distribuzione di temperatura a transitorio esaurito è la seguente:

$$T(x, y) = T_0 + T_m \sin\left(\pi \frac{x}{L}\right) \frac{\sinh\left(\pi \frac{y}{L}\right)}{\sinh\left(\pi \frac{W}{L}\right)}$$

Il seguente codice crea il grafico 3D desiderato con il comando mesh

```
W=0.6; L=0.6;
T0=25; Tm=10;

N=100;
x=linspace(0,L,N);
y=linspace(0,W,N);
[X,Y] = meshgrid(x,y);
T=T0+Tm*sin(pi*X/L).*sinh(pi*Y/L)/sinh(pi*W/L);
mesh(X,Y,T)
title('Distribuzione di temperatura')
xlabel('x [m]')
ylabel('y [m]')
zlabel('T(x,y) [C]')
colorbar
```



Impieghiamo in alternativa il comando surf

```
W=0.6; L=0.6;
T0=25; Tm=10;

N=100;
x=linspace(0,L,N);
y=linspace(0,W,N);
[X,Y] = meshgrid(x,y);
Tfun=@(x,y)(T0+Tm*sin(pi*x/L).*sinh(pi*y/L)./sinh(pi*W/L));
```

```
T= Tfun(X,Y);  
surf(X,Y,T)  
title('Distribuzione di temperatura')  
xlabel('x [m]')  
ylabel('y [m]')  
zlabel('T(x,y) [C]')  
colorbar
```

