Wolfram Mathematica
Il software di riferimento per la Didattica, la Ricerca e lo Sviluppo

ADALTA
Distributore ufficiale per l'Italia di Wolfram Research
www.adalta.it/wolfram

WebSeminar Mathematica

Lezione 7

La grafica : funzioni e opzioni

Crescenzio Gallo – Università di Foggia
crescenzio.gallo@unifg.it

Note:

- Il materiale visualizzato durante questo seminario è disponibile per il download all'indirizzo http://www.crescenziogallo.it/unifg/seminario-mathematica-2014/
- Il materiale utilizzato è tratto dai webinar pubblicati da Adalta e prodotti dal dott. Roberto Cavaliere (*Mathematica* Technic Sales Manager, *r.cavaliere@adalta.it*)

Agenda

Introduzione

- Il concetto di visualizzazione grafica in *Mathematica*
- L'oggetto grafico come espressione
- I tipi di grafici

Function visualization

- Plot, Plot3D, LogPlot, ContourPlot, ecc.: visualizzare le funzioni
- Uso delle opzioni

Data Visualization

- ListPlot, DiscretePlot, ListPlot3D, ecc.: visualizzare i dati

Drawing tool: come modificare i grafici manualmente

Qualche caso particolare di visualizzazione

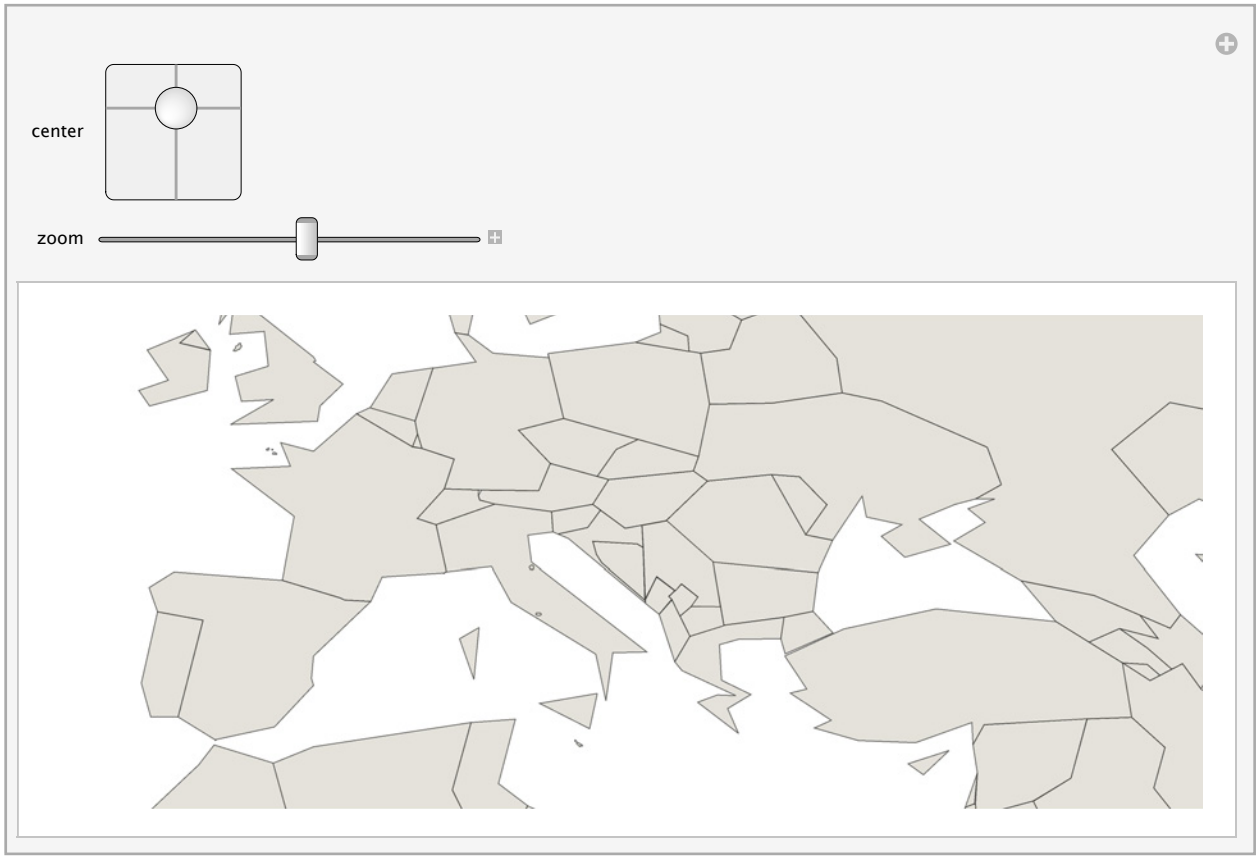
Conclusioni

Introduzione: il concetto di visualizzazione grafica in *Mathematica*

Con la versione 6 di *Mathematica* rilasciata nel 2008 si introdusse un concetto di grafica completamente rinnovato. Da una par si era convertito l'intero impianto della grafica da Postscript-like a vettoriale, con un notevole guadagno in qualità e versatili nelle rappresentazioni grafiche. Dall'altra parte, l'introduzione del concetto di elementi dinamici ed interattivi aveva anche modif cato il modo di pensare alle rappresentazioni grafiche dei dati. Dunque, l'oggetto grafico, divenuto una vera e propria espressior di *Mathematica*, assume un ruolo completamente diverso, non più semplice rappresentazione di una serie di dati ma ancora più uno strumento vero e proprio per l'esplorazione delle caratteristiche dei fenomeni oggetto di studio.

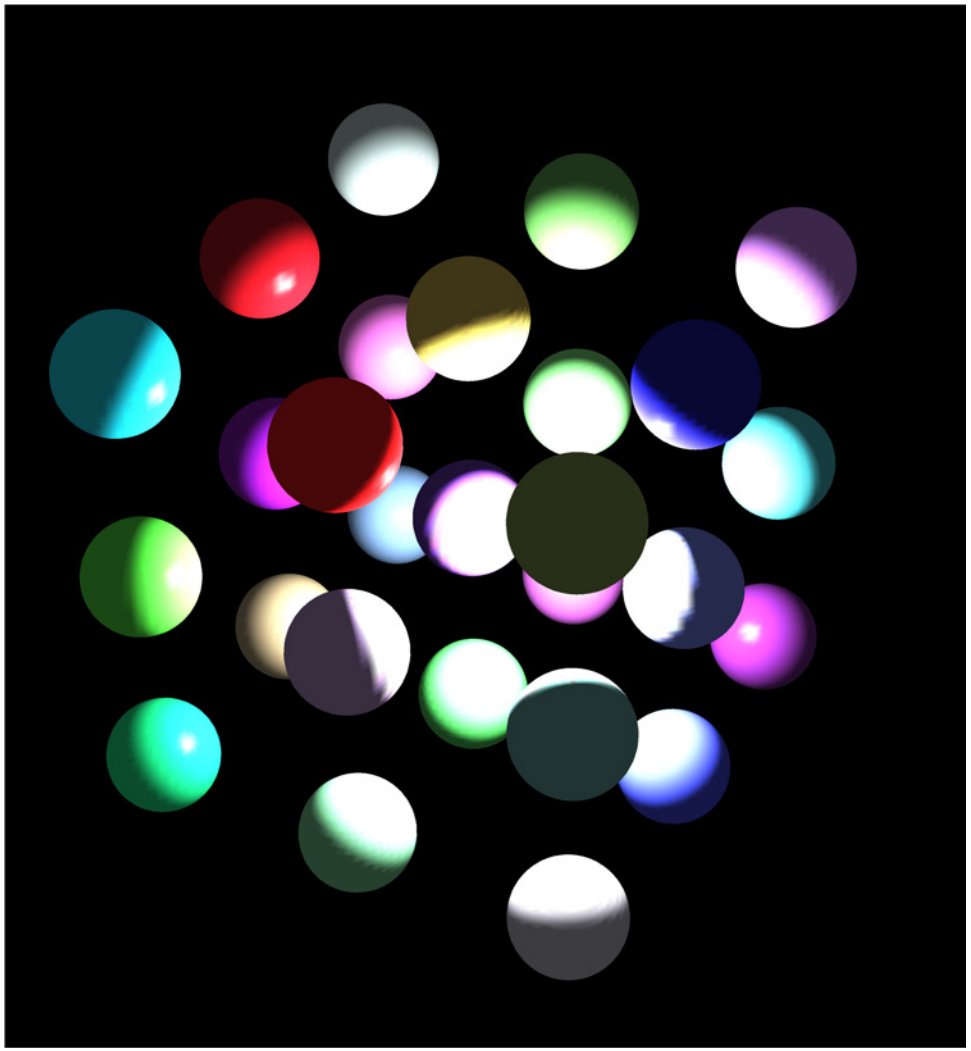
Esempi di grafici con proprietà gestibili interattivamente.

```
DynamicModule[
 {worldGraphic = CountryData["World", "Shape"], xMin, xMax, yMin, yMax, xSize, ySize},
 {{xMin, xMax}, {yMin, yMax}} = FullOptions[worldGraphic, PlotRange];
 xSize = xMax - xMin;
 ySize = yMax - yMin;
 Manipulate[
  Deploy[Show[CountryData["World", "Shape"],
    PlotRange → Dynamic[{{center[[1]] - (xSize / zoom) / 2, center[[1]] + (xSize / zoom) / 2},
       {center[[2]] - (ySize / zoom) / 2, center[[2]] + (ySize / zoom) / 2}}], ImageSize → 600]],
  {{center, {0, 0}}, {xMin, yMin}, {xMax, yMax}},
  {{zoom, 1}, 1, 10}
 ]]
```



```
Framed@DynamicModule[{
    contents = {}
  }, EventHandler[Graphics[{PointSize[0.1], Point[Dynamic[
        contents = Map[If[#[[1, 2]] ≥ 0, {#[[1]] - #[[2]], #[[2]] + {0, 0.001}}, EmitSound[Sound[Sound
              -17, .1, "Percussion"]]]; {{#[[1, 1]], 0}, {1, -0.8} #[[2]]}] &, contents];
        Map[First, contents]]
      ]}, PlotRange → {{0, 1}, {0, 1}}],
    "MouseDown" :→ (AppendTo[contents, {MousePosition["Graphics"], {0, 0}}])]]

myspheres = Table[{RGBColor[Random[], Random[], Random[]], Specularity[White, 128],
    Sphere[{x, y, z}, 1]}, {x, 0, 10, 4}, {y, 0, 10, 4}, {z, 0, 10, 4}];
Manipulate[Graphics3D[{White, PointSize[.02], Point[{t[[1]], t[[2]], 5}], myspheres},
  Background -> Black, Boxed -> False,
  Lighting -> {RGBColor[.3, .3, .3], {White, {{{t[[1]], t[[2]], 5}, {0, 0, 0}}, 2}}},
  PlotRange -> {{-1, 10}, {-1, 10}, {-1, 10}}, ImageSize -> 500],
 {t, {-15, -15}, {20, 20}}, SaveDefinitions -> True]
```

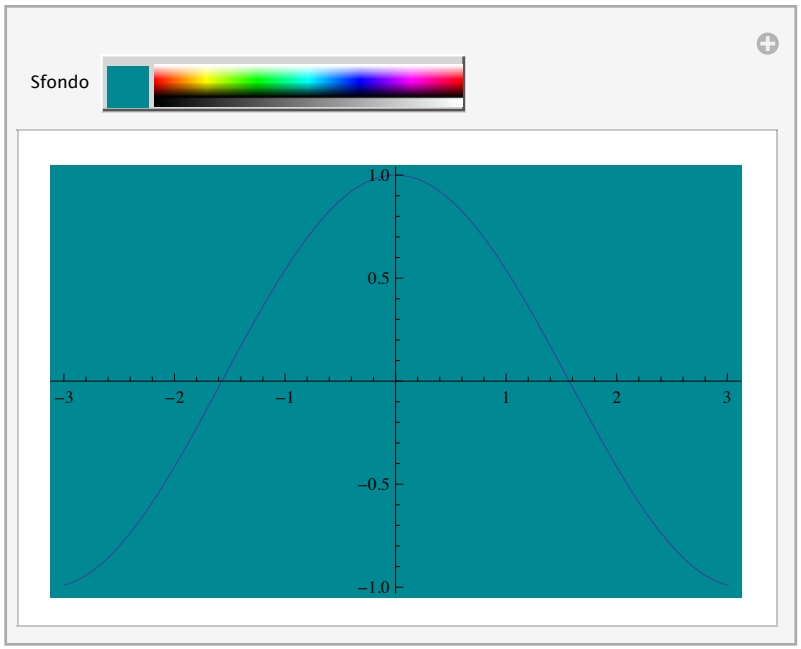
t



Introduzione: l'oggetto grafico come espressione

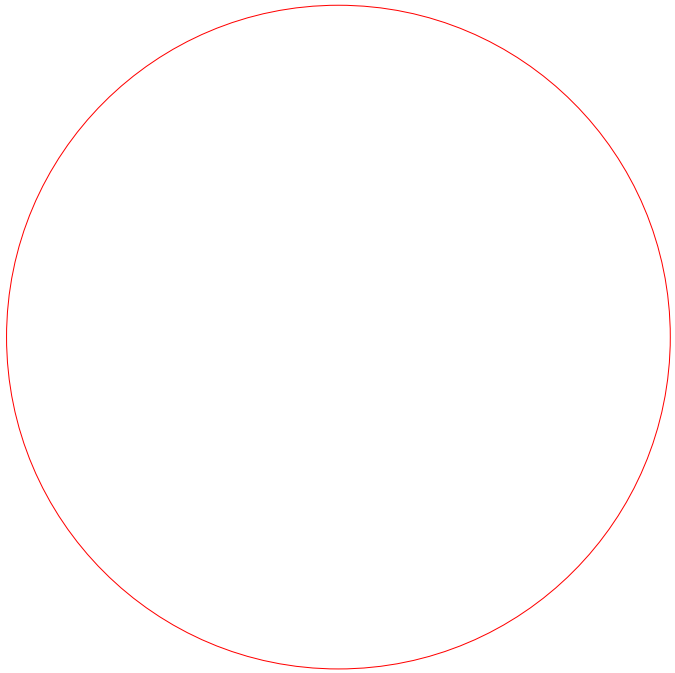
Quanto visto precedentemente è reso possibile dal fatto che i grafici sono rappresentati internamente esattamente come ogni alt espressione di *Mathematica*, dunque qualsiasi loro proprietà può essere gestita interattivamente. Vediamo un esempio del stesso concetto ma estremamente più elementare.

```
m = Manipulate[Plot[Cos[x], {x, -3, 3}, Background → color], {{color, Red, "Sfondo"}, Blue}]
```

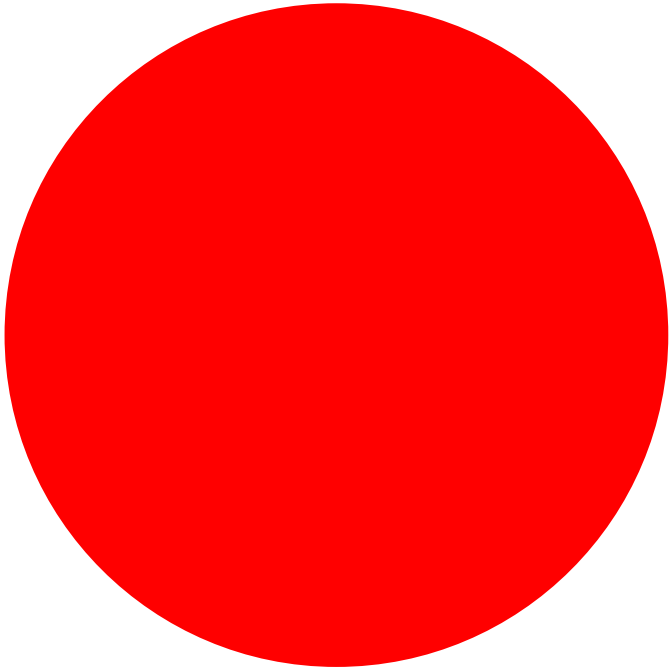


Altro vantaggio di avere i grafici rappresentati come espressioni è che vi si possono apportare modifiche anche dopo che il grafi è stato realizzato.

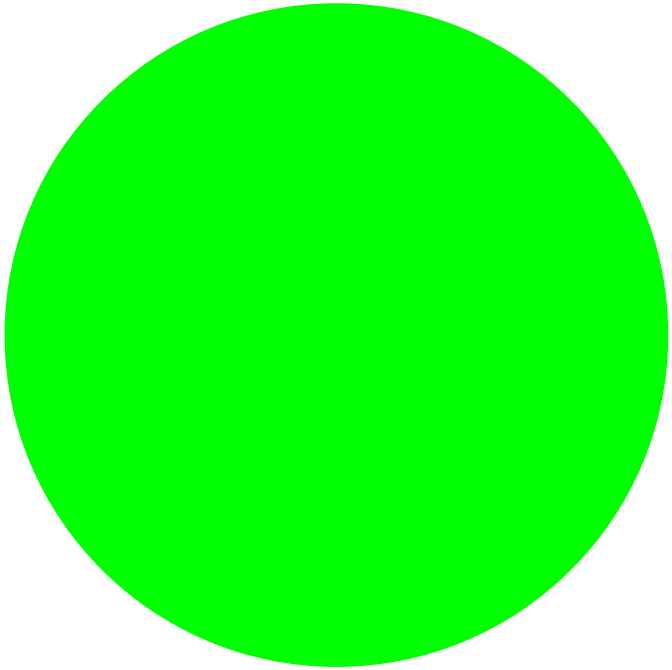
```
gr = Graphics[{Red, Circle[]}]
```



```
ReplaceAll[gr, Circle -> Disk]
```

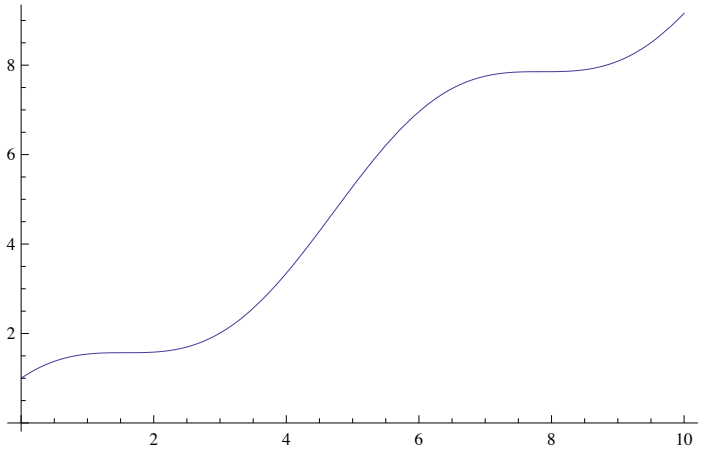


```
ReplaceAll[gr, {Circle -> Disk, Red → Green}]
```

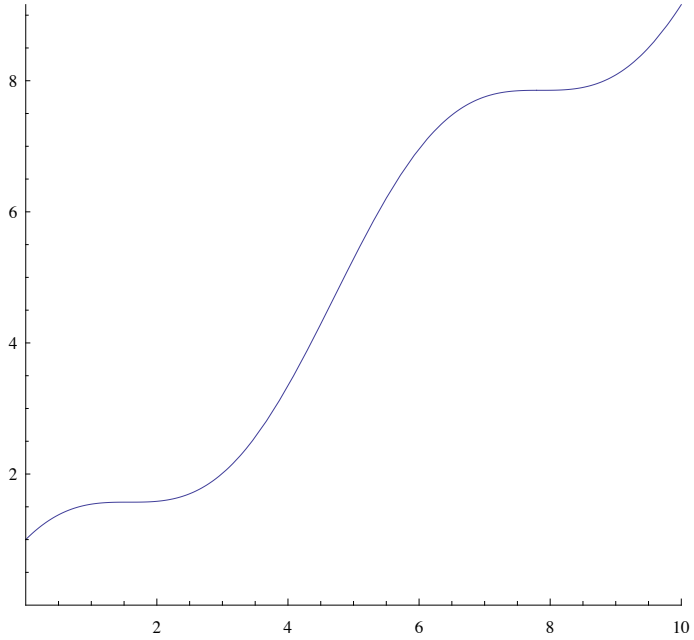


Un altro esempio di modifica post eleborazione.

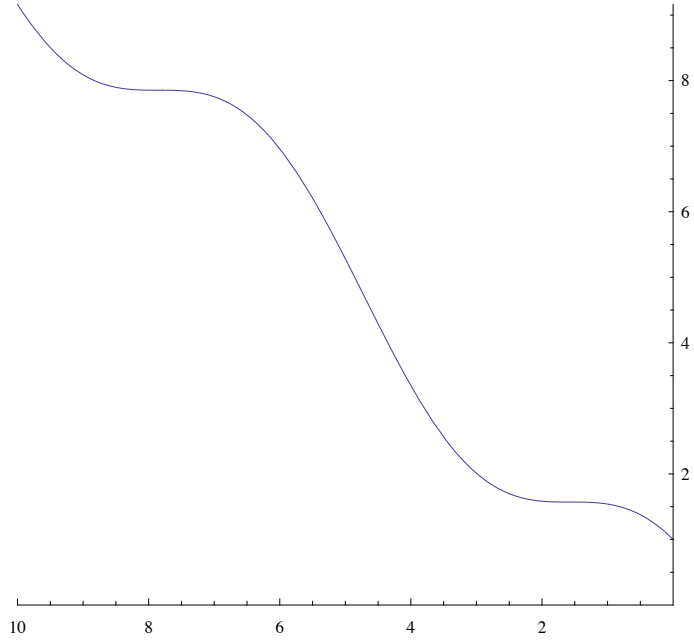
```
Plot[Cos[x] + x, {x, 0, 10}]
```




```
s = FullGraphics[Plot[Cos[x] + x, {x, 0, 10}]]
```



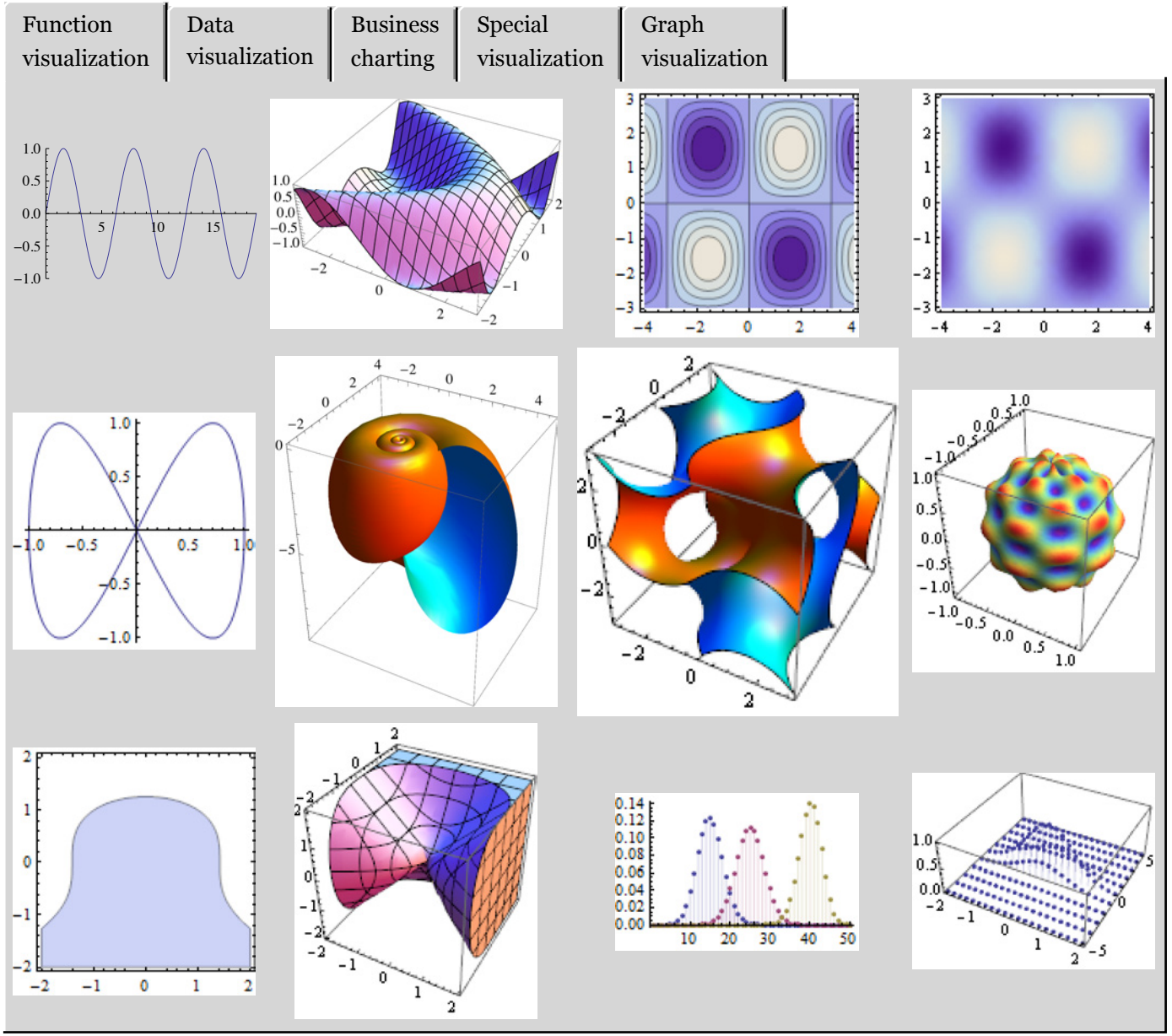

```
ReplaceAll[s, {x_Real, y_Real} → {-x, y}]
```



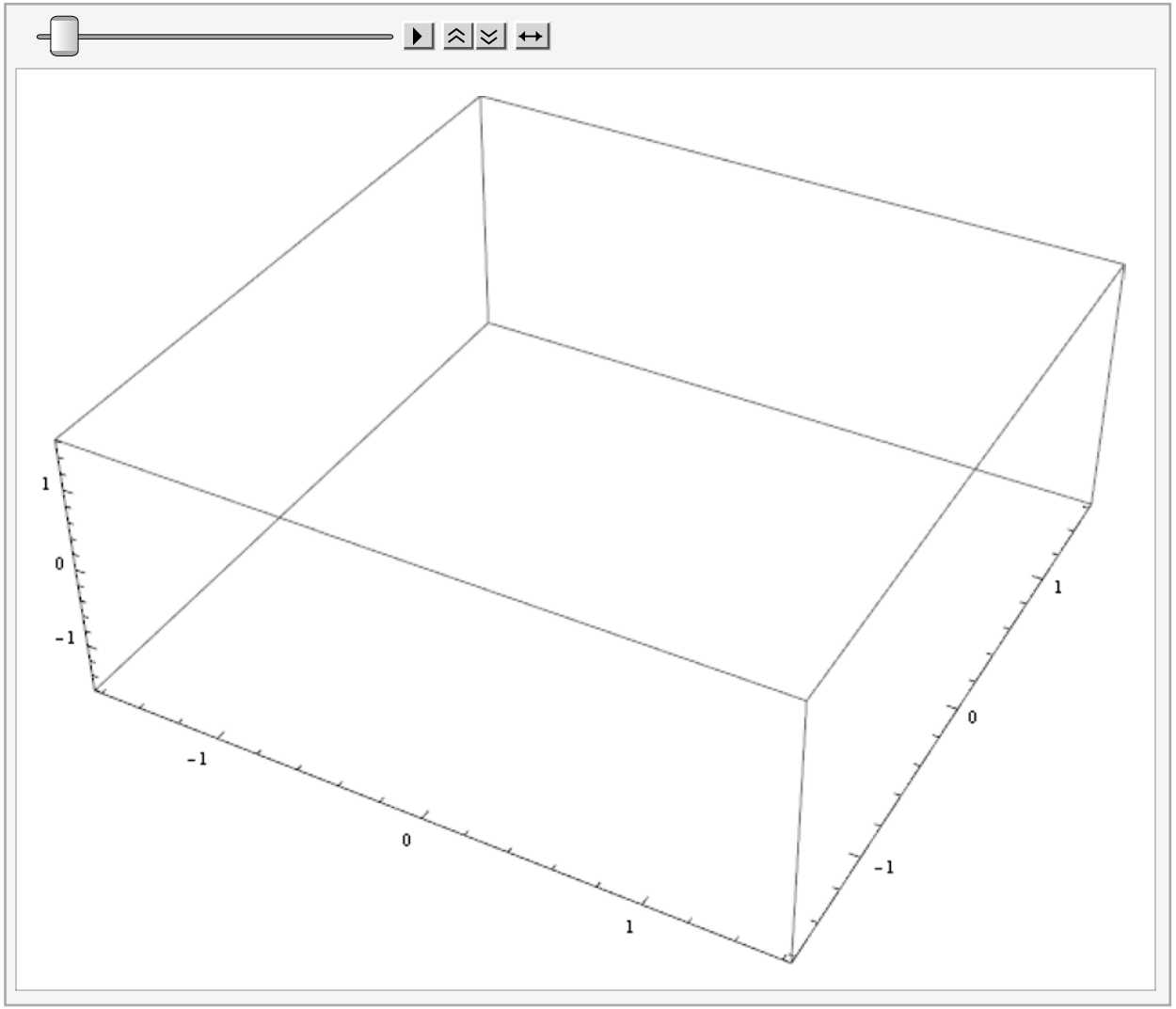
Introduzione: i tipi di grafici

Riepilogo principali funzioni

Mathematica dispone di centinaia di funzioni dedicate espressamente alla grafica in due e tre dimensioni.


Function visualization
Data visualization
Business charting
Special visualization
Graph visualization

La qualità


1
0
-1
-1
0
1
1
0
-1

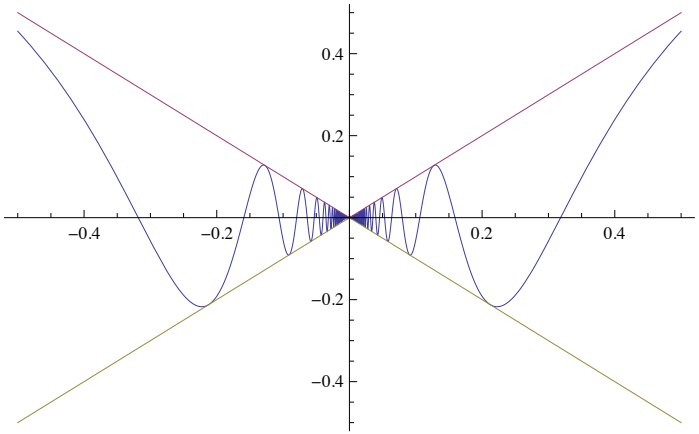
Function visualization: Plot, Plot3D, LogPlot, ContourPlot, ecc.: visualizzare le funzioni

Quando si conosce l'espressione analitica di una funzione si possono utilizzare comandi quali **Plot**, **Plot3D** e tantissimi alt (Function Visualization). Vediamo qualche esempio.

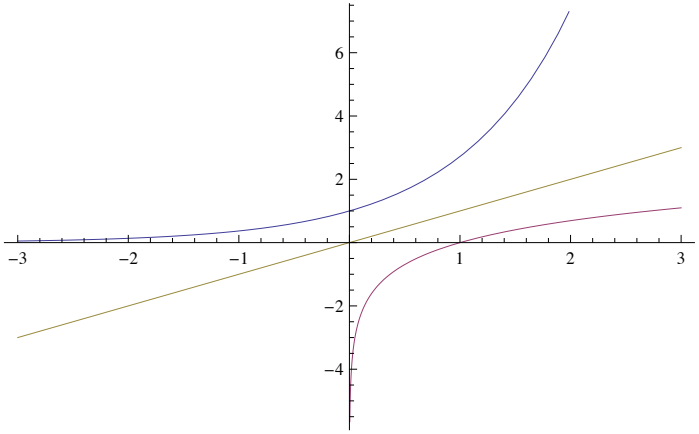
```
? Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, ...}, {x, x_{min}, x_{max}}] plots several functions f_i. ≫

```
Plot[{x Sin[1 / x], Abs[x], -Abs[x]}, {x, -1 / 2, 1 / 2}]
```



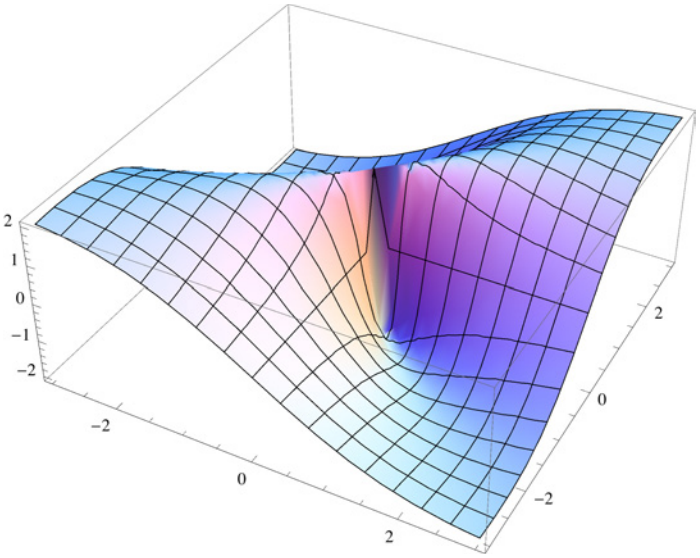
```
Plot[{Exp[x], Log[x], x}, {x, -3, 3}]
```



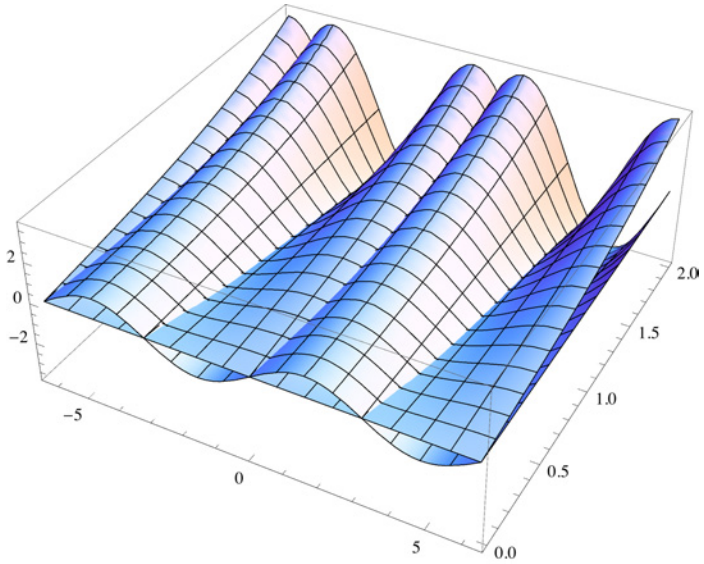

```
? Plot3D
```

Plot3D[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a three-dimensional plot of f as a function of x and y.
Plot3D[{f_1, f_2, ...}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several functions. ≫

```
Plot3D[10 x y / (2 x^2 + 3 y^2), {x, -3, 3}, {y, -3, 3}]
```



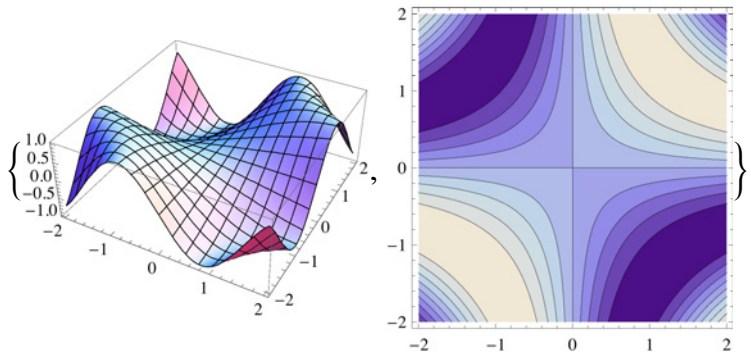
```
Plot3D[{Re[Sin[x + I y]], Im[Sin[x + I y]]}, {x, -2 Pi, 2 Pi}, {y, 0, 2}]
```




```
? ContourPlot
```

ContourPlot[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a contour plot of f as a function of x and y.
ContourPlot[$f == g$, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots contour lines for which $f = g$.
ContourPlot[{$f_1 == g_1$, $f_2 == g_2$, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several contour lines. ≫

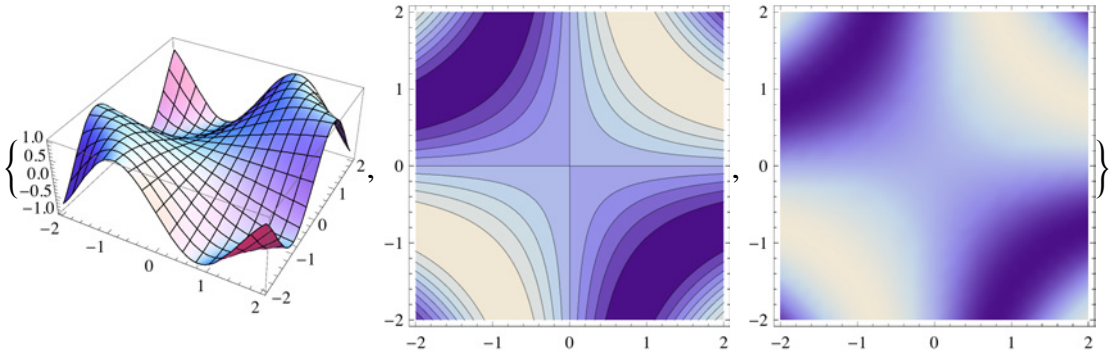
```
{Plot3D[Sin[x y], {x, -2, 2}, {y, -2, 2}], ContourPlot[Sin[x y], {x, -2, 2}, {y, -2, 2}]}
```



```
? DensityPlot
```

DensityPlot[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] makes a density plot of f as a function of x and y. ≫

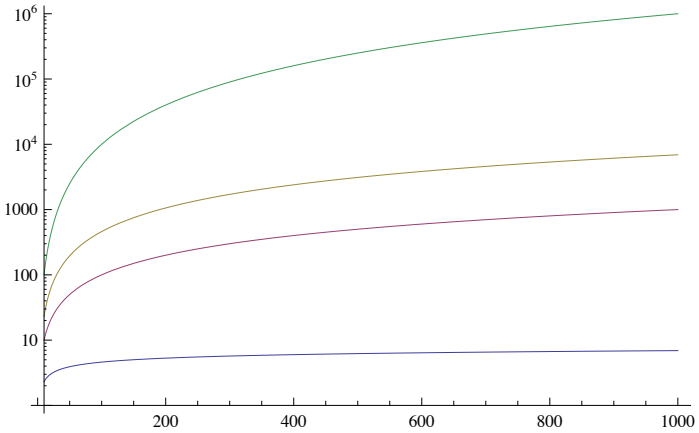
```
{Plot3D[Sin[x y], {x, -2, 2}, {y, -2, 2}],
 ContourPlot[Sin[x y], {x, -2, 2}, {y, -2, 2}], DensityPlot[Sin[x y], {x, -2, 2}, {y, -2, 2}]}
```



```
? LogPlot
```

LogPlot[f, {x, x_{min}, x_{max}}] generates a log plot of f as a function of x from x_{min} to x_{max}.
LogPlot[{f_1, f_2, ...}, {x, x_{min}, x_{max}}] generates log plots of several functions f_i. ≫

```
LogPlot[Tooltip@{Log[n], n, n Log[n], n^2}, {n, 10, 10^3}]
```



Function visualization: uso delle opzioni

La parte più significativa nelle funzionalità di visualizzazione dei dati in *Mathematica* è composta dall'uso delle opzioni per perso alizzare il risultato grafico. La grafica, infatti, risulta una delle componenti con il maggior numero di possibili diversi output, s dal punto di vista del calcolo come ad esempio nella scelta delle precisione per il calcolo dei punti da rappresentare, sia dal pun di vista dell'*estetica* con le centinaia di possibili varianti che ciascun grafico può assumere, ad esempio dal colore, spessore e tra teggio delle linee al colore del riempimento o dello sfondo, alle dimensioni del riquadro e così via. Vediamo ad esempio quan opzioni ha la **Plot**.

```
Options[Plot]
```

$$\left\{\text{AlignmentPoint} \to \text{Center}, \text{AspectRatio} \to \frac{1}{\phi}, \text{Axes} \to \text{True}, \text{AxesLabel} \to \text{None}, \text{AxesOrigin} \to \text{Automatic}, \text{AxesStyle} \to \{\},\right.$$
$$\text{Background} \to \text{None}, \text{BaselinePosition} \to \text{Automatic}, \text{BaseStyle} \to \{\}, \text{ClippingStyle} \to \text{None}, \text{ColorFunction} \to \text{Automatic},$$
$$\text{ColorFunctionScaling} \to \text{True}, \text{ColorOutput} \to \text{Automatic}, \text{ContentSelectable} \to \text{Automatic}, \text{CoordinatesToolOptions} \to \text{Automatic},$$
$$\text{DisplayFunction} :\to \$\text{DisplayFunction}, \text{Epilog} \to \{\}, \text{Evaluated} \to \text{Automatic}, \text{EvaluationMonitor} \to \text{None}, \text{Exclusions} \to \text{Automatic},$$
$$\text{ExclusionsStyle} \to \text{None}, \text{Filling} \to \text{None}, \text{FillingStyle} \to \text{Automatic}, \text{FormatType} :\to \text{TraditionalForm}, \text{Frame} \to \text{False},$$
$$\text{FrameLabel} \to \text{None}, \text{FrameStyle} \to \{\}, \text{FrameTicks} \to \text{Automatic}, \text{FrameTicksStyle} \to \{\}, \text{GridLines} \to \text{None},$$
$$\text{GridLinesStyle} \to \{\}, \text{ImageMargins} \to 0., \text{ImagePadding} \to \text{All}, \text{ImageSize} \to \text{Automatic}, \text{ImageSizeRaw} \to \text{Automatic},$$
$$\text{LabelStyle} \to \{\}, \text{MaxRecursion} \to \text{Automatic}, \text{Mesh} \to \text{None}, \text{MeshFunctions} \to \{\#1 \&\}, \text{MeshShading} \to \text{None},$$
$$\text{MeshStyle} \to \text{Automatic}, \text{Method} \to \text{Automatic}, \text{PerformanceGoal} :\to \$\text{PerformanceGoal}, \text{PlotLabel} \to \text{None}, \text{PlotLegends} \to \text{None},$$
$$\text{PlotPoints} \to \text{Automatic}, \text{PlotRange} \to \{\text{Full}, \text{Automatic}\}, \text{PlotRangeClipping} \to \text{True}, \text{PlotRangePadding} \to \text{Automatic},$$
$$\text{PlotRegion} \to \text{Automatic}, \text{PlotStyle} \to \text{Automatic}, \text{PreserveImageOptions} \to \text{Automatic}, \text{Prolog} \to \{\}, \text{RegionFunction} \to (\text{True} \&),$$
$$\left.\text{RotateLabel} \to \text{True}, \text{TargetUnits} \to \text{Automatic}, \text{Ticks} \to \text{Automatic}, \text{TicksStyle} \to \{\}, \text{WorkingPrecision} \to \text{MachinePrecision}\right\}$$

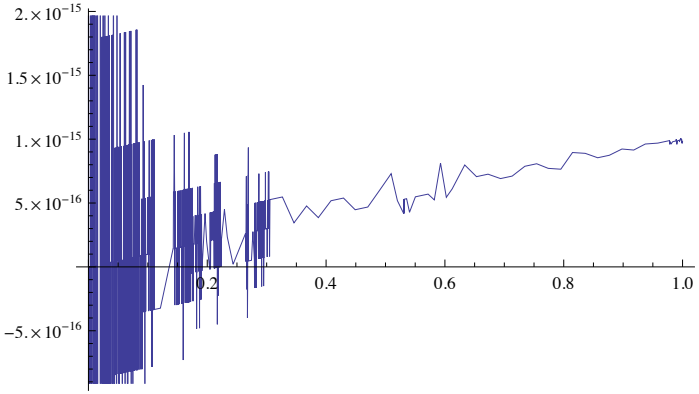
```
Length[%]
```

59

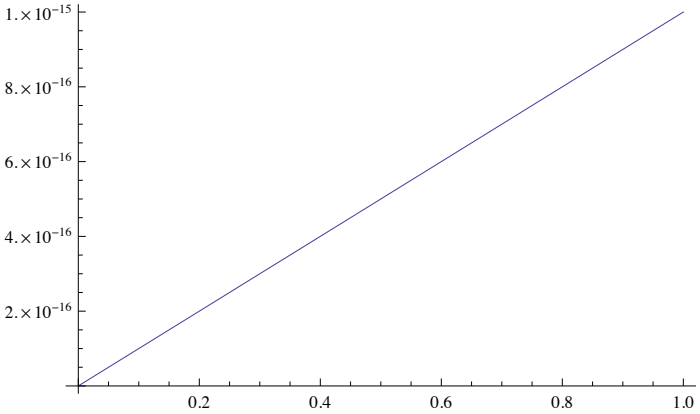
Vediamone qualcuna

La precisione del calcolo (WorkingPrecision)

```
Plot[z/10^15 + Im[WeierstrassZeta[z, {3, 4}]], {z, 0, 1}]
```



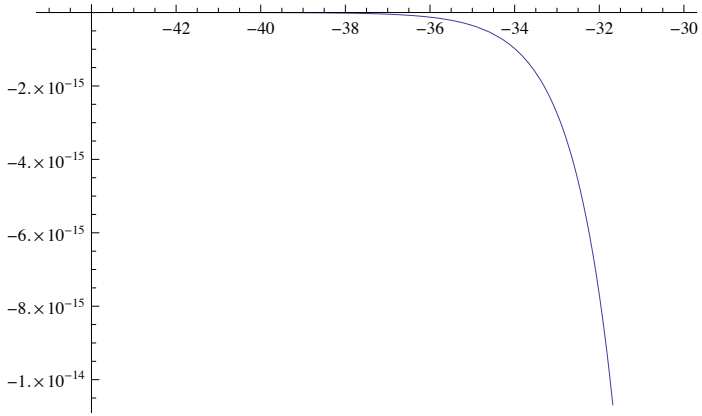

```
Plot[z/10^15 + Im[WeierstrassZeta[z, {3, 4}]], {z, 0, 1}, WorkingPrecision → 32]
```



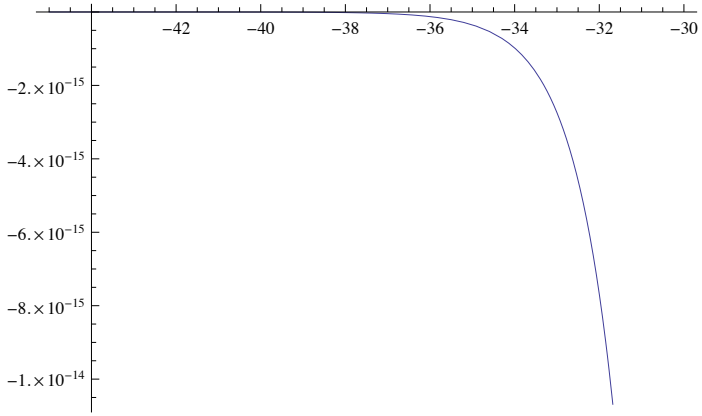

```
f = ∫_-∞^t 20 Exp[x]/x dx
```

ConditionalExpression[20 Ei(t), Im(t) ≠ 0 ⋁ Re(t) < 0]

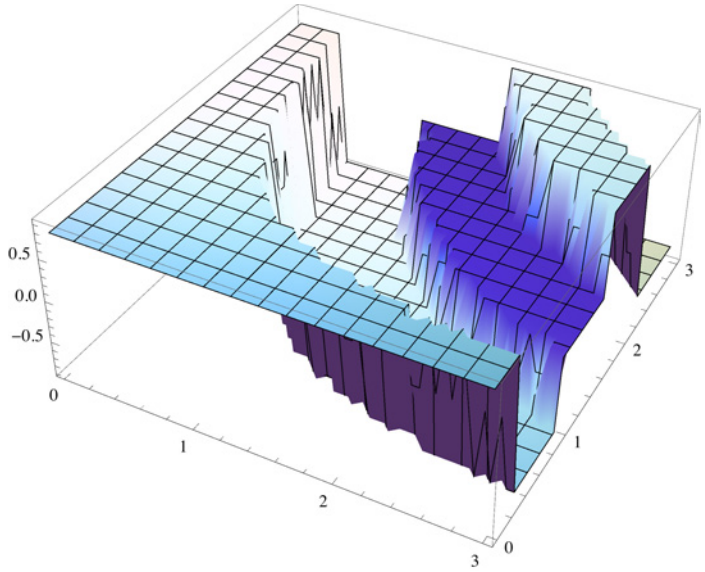
```
Plot[Sin[f], {t, -45, -30}]
```



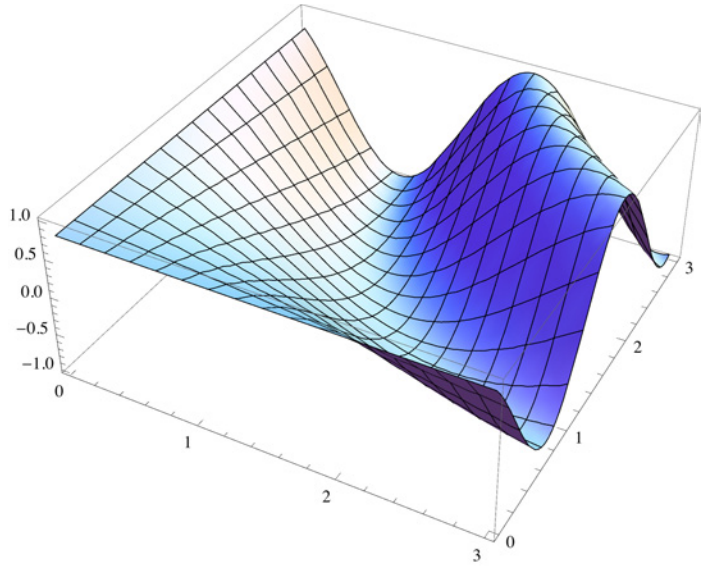

```
Plot[Sin[f], {t, -45, -30}, WorkingPrecision → 32]
```




```
Plot3D[Sin[x y + 10 ^ 16], {x, 0, 3}, {y, 0, 3}]
```

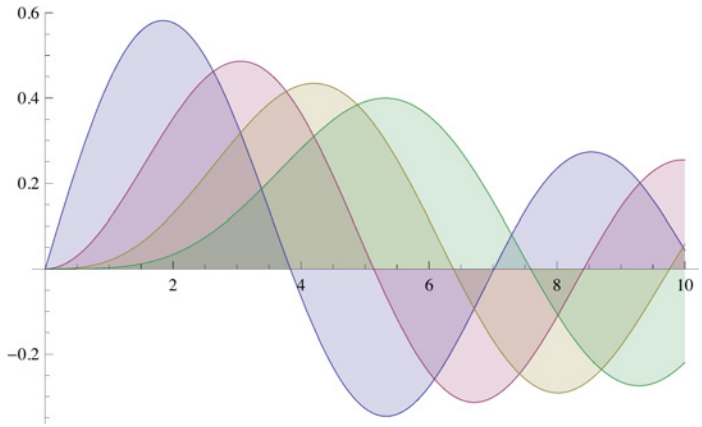



```
Plot3D[Sin[x y + 10 ^ 16], {x, 0, 3}, {y, 0, 3}, WorkingPrecision → 20]
```

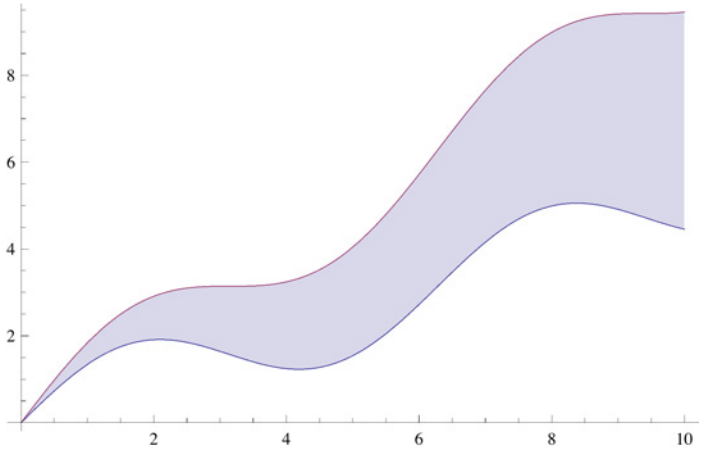


Il riempimento dell'area sottostante la curva o tra più curve (Filling)

```
Plot[Evaluate[Table[BesselJ[n, x], {n, 4}]], {x, 0, 10}, Filling → Axis]
```

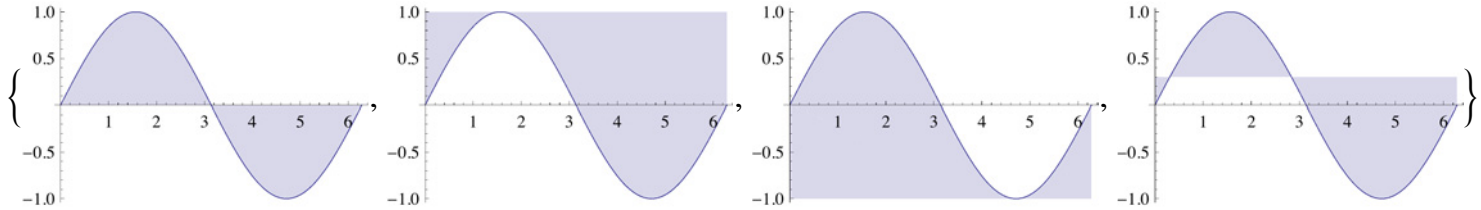



```
Plot[{Sin[x] + x / 2, Sin[x] + x}, {x, 0, 10}, Filling → {1 → {2}}]
```



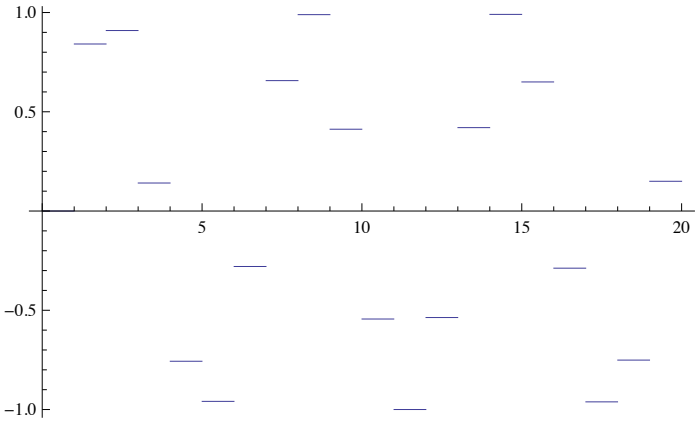
Varie possibilità di personalizzare il filling

```
Table[Plot[Sin[x], {x, 0, 2 Pi}, Filling → f], {f, {Axis, Top, Bottom, 0.3}}]
```

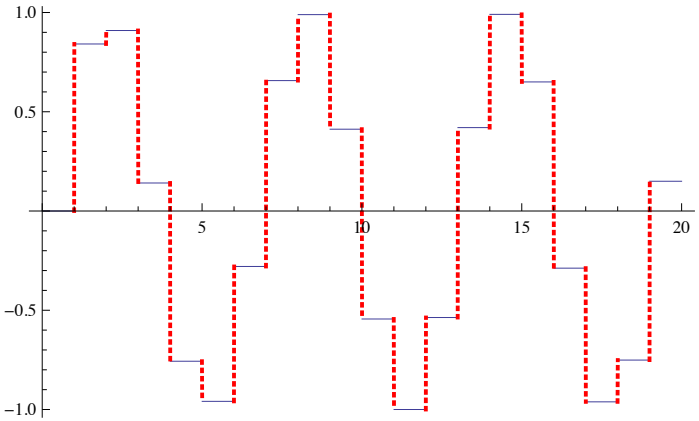


Evidenziare le singolarità (Exclusions / ExclusionsStyle)

```
Plot[Sin[Floor[x ]], {x, 0, 20}]
```

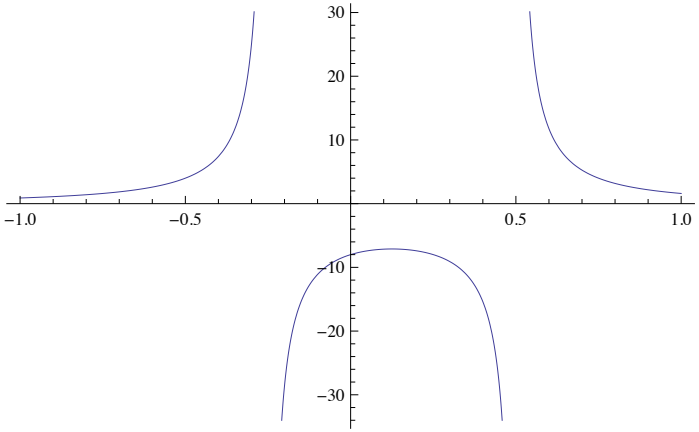



```
Plot[Sin[Floor[x]], {x, 0, 20}, ExclusionsStyle → Directive[{Dashing[{.0025, .01}], Red, Thick}]]
```



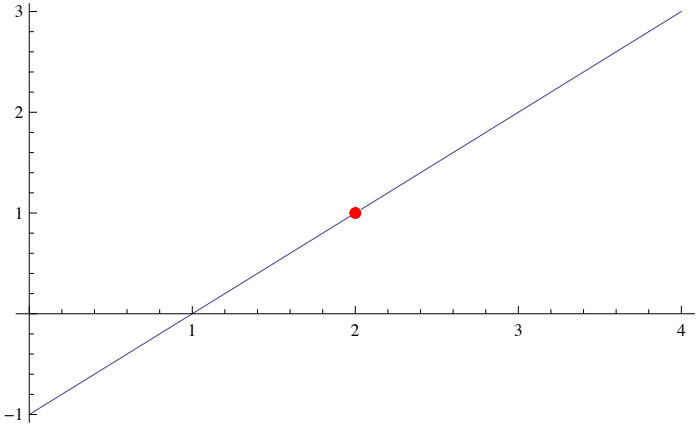
È possibile specificare i punti da escludere.

```
Plot[1/((x + 1/4) (x - 1/2)), {x, -1, 1}, Exclusions → {x == -1/4, x == 1/2}]
```

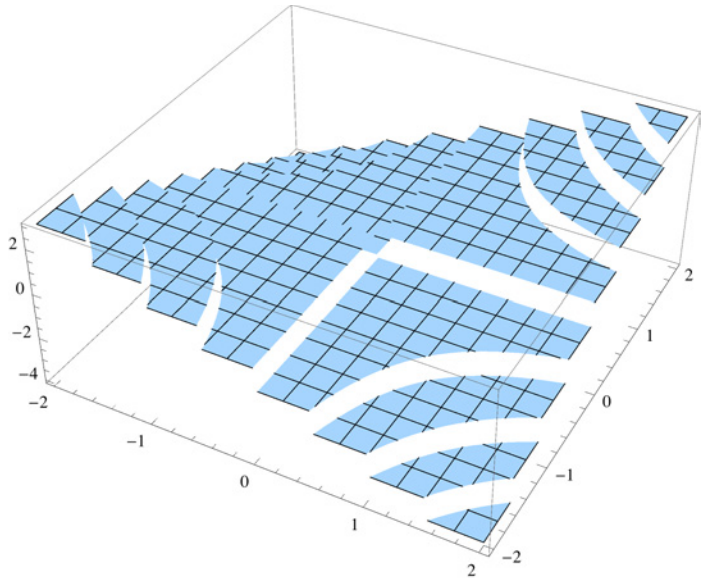


Anche le singolarità eliminabili possono essere indicate utilizzando la stessa tecnica.

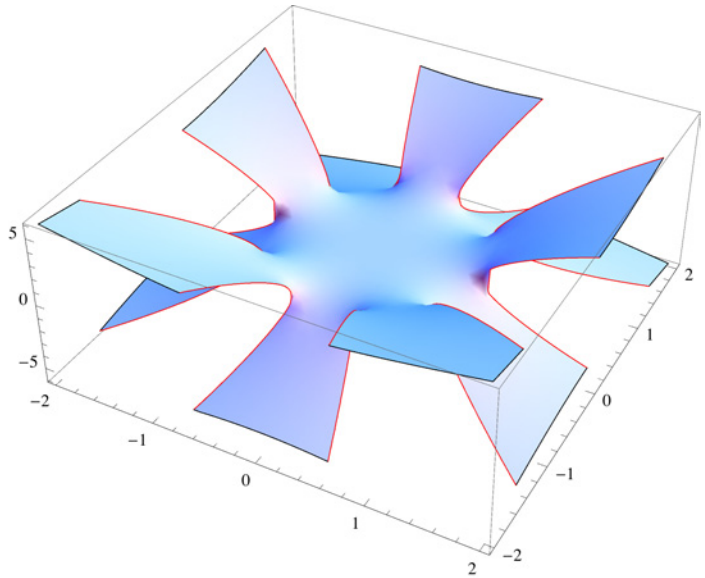
```
Plot[(x^2 - 3 x + 2)/(x - 2), {x, 0, 4}, Exclusions → {x == 2},
 ExclusionsStyle → {None, {AbsolutePointSize[6], Red}}]
```



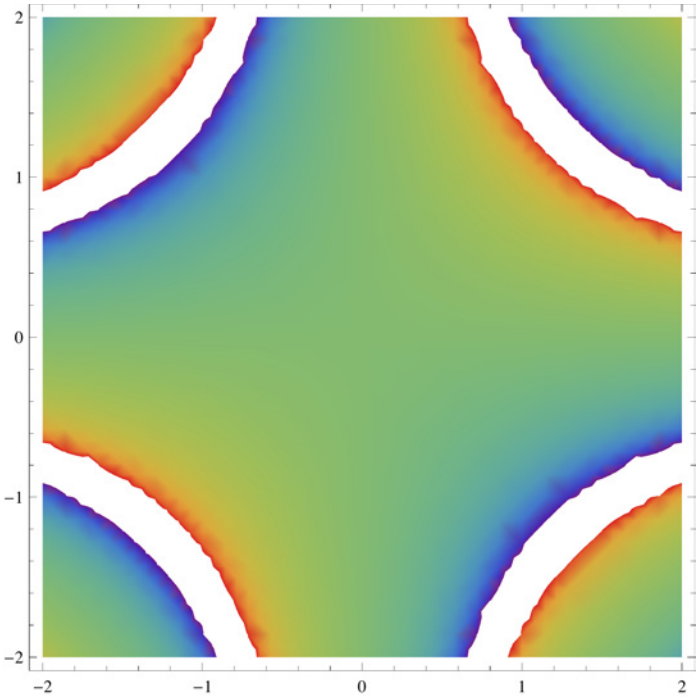

```
Plot3D[Floor[x y], {x, -2, 2}, {y, -2, 2}]
```




```
Plot3D[Im[ArcSin[(x + I y)^5]], {x, -2, 2}, {y, -2, 2}, ExclusionsStyle → {None, Red}, Mesh → None]
```



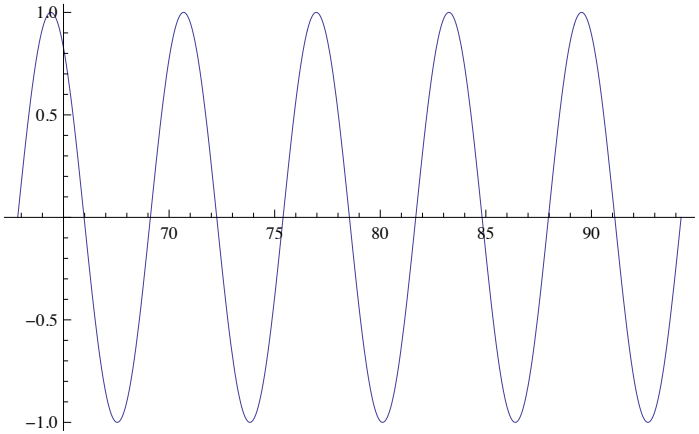

```
DensityPlot[Tan[x y], {x, -2, 2}, {y, -2, 2}, ColorFunction → "Rainbow", Exclusions → {Cos[x y] == 0}]
```



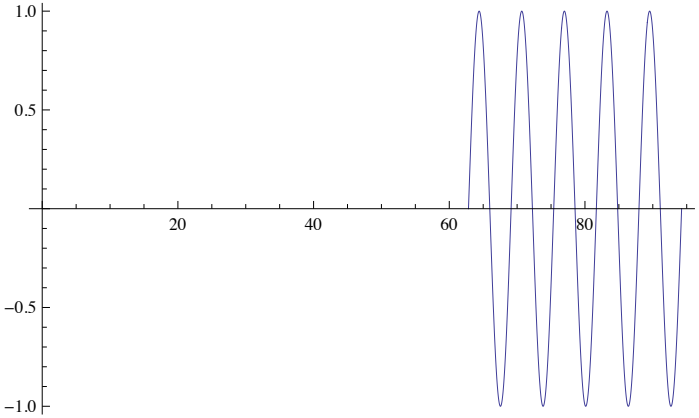
Origine degli assi (AxesOrigin)

Se il punto {0, 0} non è presente nel range del plot automaticamente viene scelto un altro punto come origine degli assi. S vogliamo evidenziare comunque il punto {0, 0} possiamo usare **AxesOrigin**.

```
Plot[Sin[x], {x, 20 π, 30 π}]
```



```
Plot[Sin[x], {x, 20 π, 30 π}, AxesOrigin -> {0, 0}]
```

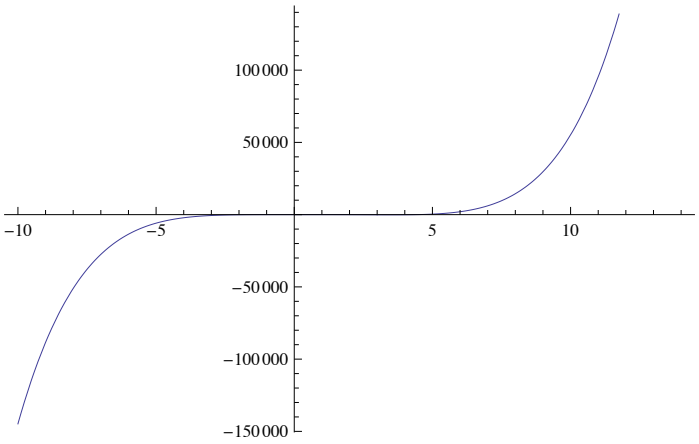


I grafico sembra non completo, ma di fatto è corretto perchè il **PlotRange** utilizza il range sulle *x* e dunque non disegna (o megl non calcola) il grafico fuori da questo intervallo.

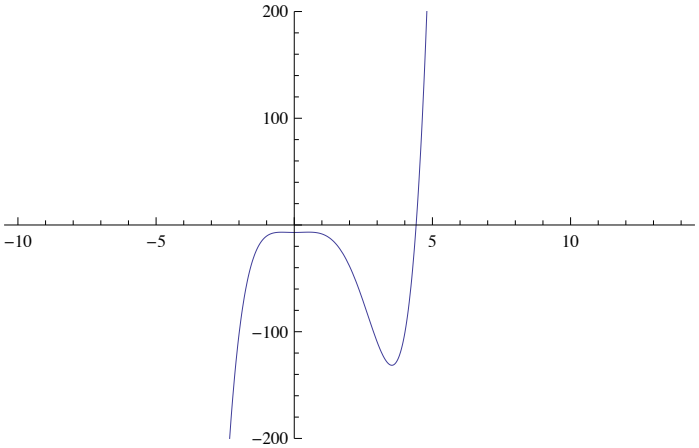
L'intervallo giusto per la variabile dipendente (PlotRange)

In alcuni casi potrebbe essere necessario specificare un diverso range per la funzione in quanto la scelta compiuta automatic mente potrebbe non essere ottimale.

```
Plot[x^5 - 4.5 x^4 + 2.1 x^2 - 7, {x, -10, 14}]
```



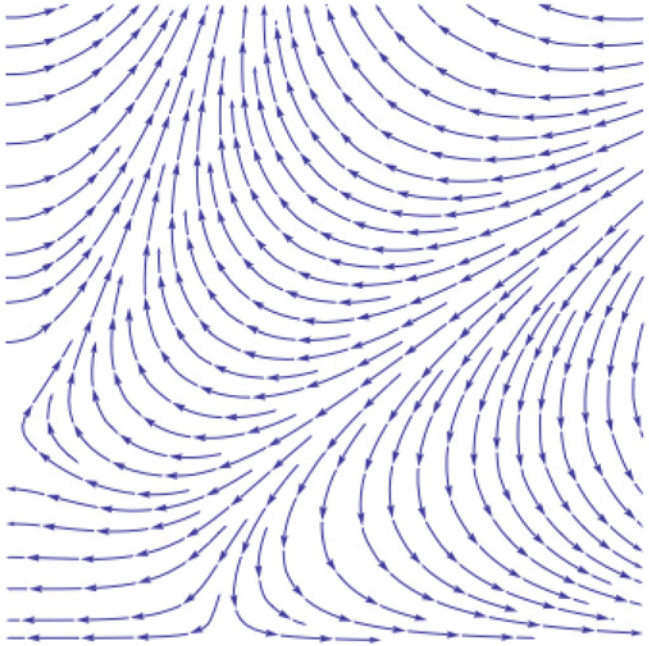

```
Plot[x^5 - 4.5 x^4 + 2.1 x^2 - 7, {x, -10, 14}, PlotRange → {-200, 200}]
```



Sovrapporre un'immagine 2d ad una superfice 3D (Texture)

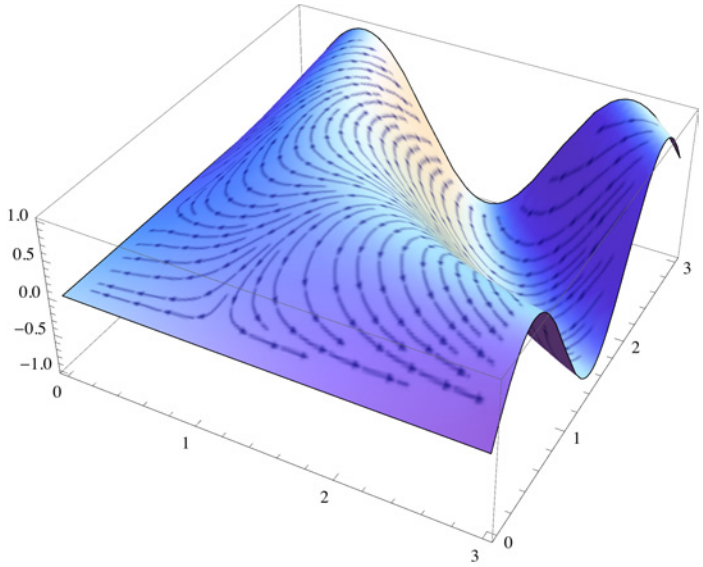
La funzionalità di *texture* (aggiunta nella versione 8 di *Mathematica*) consente di sovrapporre un'immagine di "tessitura" ad u qualsiasi grafico.

```
arrows = Rasterize[StreamPlot[Evaluate[{Re[(x + i y)^4 - 1], -Im[(x + i y)^4 - 1]}], {x, 0, 3},
    {y, 0, 3}, VectorScale → {Automatic, Automatic, Log[#5 + 1] &}, Frame → False]]
```

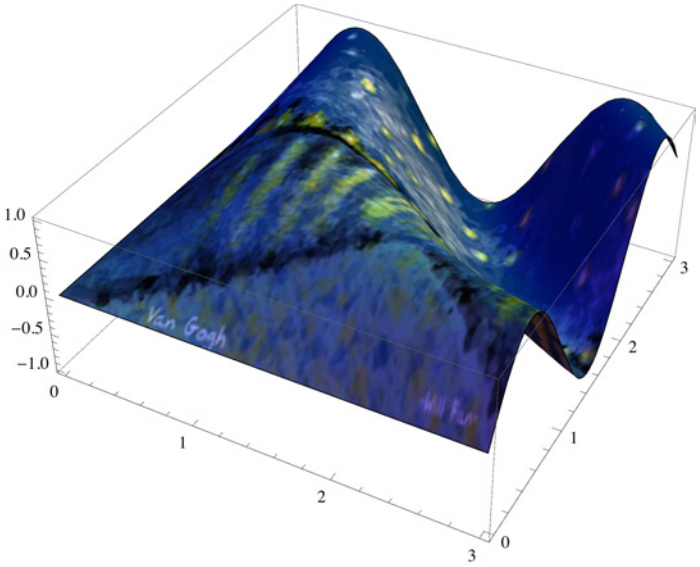


Ora usiamo il grafico di sopra (rasterizzato proprio per renderlo più snello in termini di codice e dunque più *leggero*)

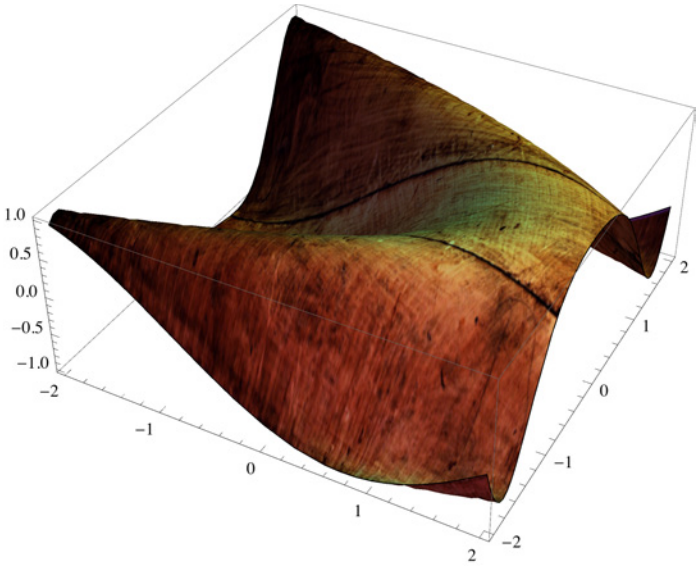
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}, Mesh → None, PlotStyle → Texture[arrows]]
```



```
van = Rasterize[
   Import["http://static.blogo.it/06blog/van-gogh-al-vittoriano-di-roma/VanGoghaRoma4.jpg"]];

Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}, Mesh → None, PlotStyle → Texture[van]]
```



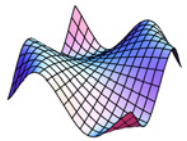

```
legno =
  Rasterize[Import["http://www.talo.it/mywp/wp-content/gallery/textures/texture-wood.jpg"]];

Plot3D[Sin[x + y^2], {x, -2, 2}, {y, -2, 2}, Mesh → None, PlotStyle → Texture[legno]]
```



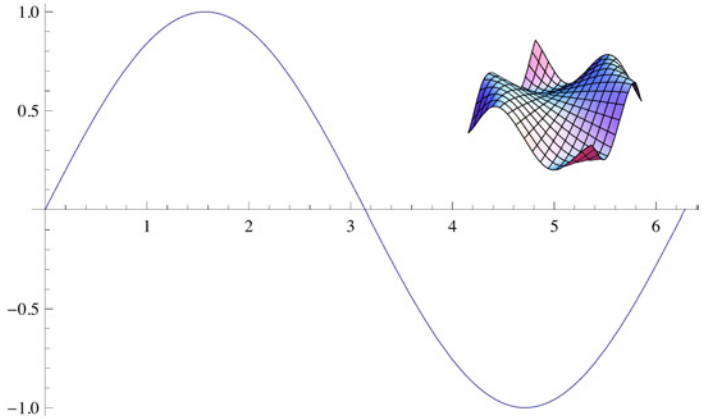
Aggiungere elementi non direttamente collegati al calcolo del grafico (Epilog / Inset)

Inset è una direttiva (non un'opzione) che permette di aggiungere un oggetto all'interno di un grafico, ad esempio trami l'opzione **Epilog**. **Inset** è molto più potente della direttiva **Text** che permette di inserire un testo. Infatti con **Inset** si possor inserire anche oggetti quali panel, slider, ecc.

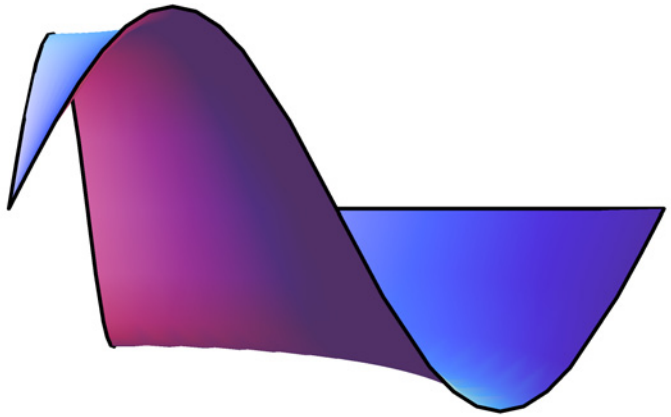
```
gr = Plot3D[Sin[x y], {x, -2, 2}, {y, -2, 2}, Axes → False, ImageSize → 100, Boxed → False]
```



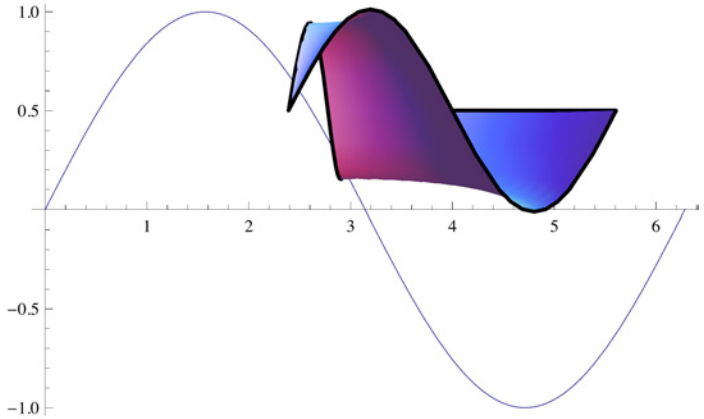
```
Plot[Sin[x], {x, 0, 2 π}, Epilog → Inset[gr, {5, .5}]]
```



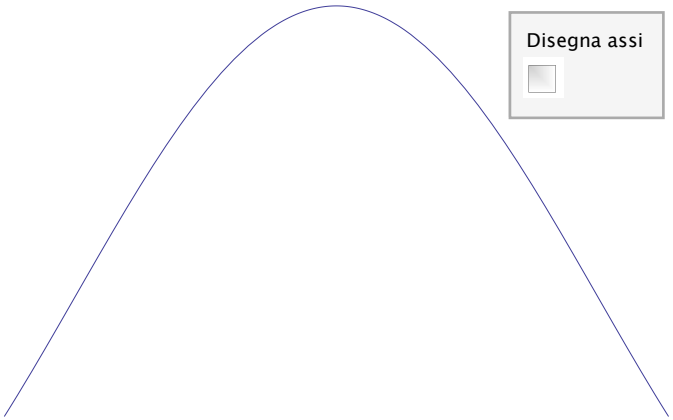

```
gr = RevolutionPlot3D[Sin[x], {x, 0, 2 Pi}, {θ, 0, Pi / 2}, Axes → False,
   Boxed → False, ViewPoint → Front, Mesh → None, BoundaryStyle → Thick]
```



```
Plot[Sin[x], {x, 0, 2 π}, Epilog → Inset[gr, {4, .5}]]
```




```
Dynamic[Plot[Cos[x], {x, -2, 2}, Axes → assi, ImagePadding → 0,
  Epilog → Inset[Panel[Column[{"Disegna assi", Checkbox[Dynamic[assi]]}]], {1.5, .8}]]]
```



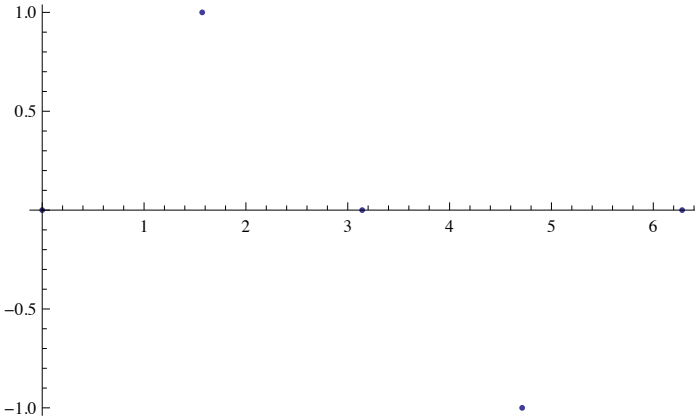
Data visualization: ListPlot, ListPlot3D, ecc.: visualizzare i dati

Quando dobbiamo visualizzare dati non noti in forma analitica abbiamo un'altra serie di funzioni dedicate alla grafica di funzio
note per punti (Data Visualization). Anche in questo caso abbiamo moltissime opzioni per personalizzare i risultati.

```
punti = {{0, 0}, {π/2, 1}, {π, 0}, {3 π/2, -1}, {2 π, 0}}
```

$$\begin{pmatrix} 0 & 0 \\ \frac{\pi}{2} & 1 \\ \pi & 0 \\ \frac{3\pi}{2} & -1 \\ 2\pi & 0 \end{pmatrix}$$

```
ListPlot[punti]
```



I punti sono poco visibili, con l'aiuto delle opzioni possiamo migliorare la visualizzazione

```
Options[ListPlot]
```

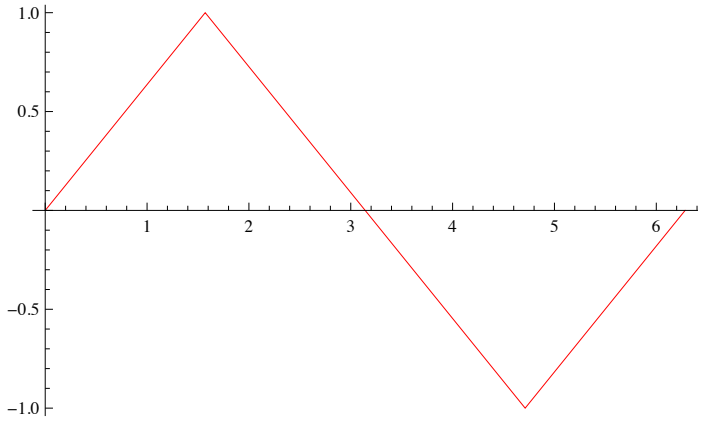
$$\left\{\text{AlignmentPoint} \to \text{Center}, \text{AspectRatio} \to \frac{1}{\phi}, \text{Axes} \to \text{True}, \text{AxesLabel} \to \text{None}, \text{AxesOrigin} \to \text{Automatic}, \text{AxesStyle} \to \{\},\right.$$

Background → None, BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None, ColorFunction → Automatic,
ColorFunctionScaling → True, ColorOutput → Automatic, ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
DataRange → Automatic, DisplayFunction :→ $DisplayFunction, Epilog → {}, Filling → None, FillingStyle → Automatic,
FormatType :→ TraditionalForm, Frame → False, FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic,
FrameTicksStyle → {}, GridLines → None, GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All, ImageSize → Automatic,
ImageSizeRaw → Automatic, InterpolationOrder → None, Joined → False, LabelStyle → {}, MaxPlotPoints → ∞,
Mesh → None, MeshFunctions → {#1 &}, MeshShading → None, MeshStyle → Automatic, Method → Automatic,
PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None, PlotMarkers → None, PlotRange → Automatic,
PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,

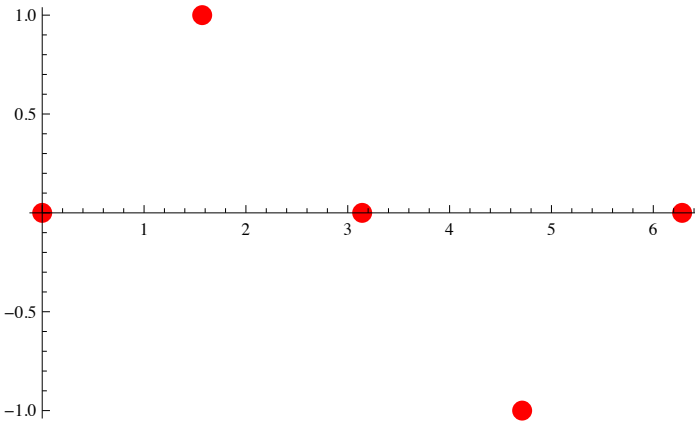
$$\left.\text{PreserveImageOptions} \to \text{Automatic}, \text{Prolog} \to \{\}, \text{RotateLabel} \to \text{True}, \text{TargetUnits} \to \text{Automatic}, \text{Ticks} \to \text{Automatic}, \text{TicksStyle} \to \{\}\right\}$$

PlotStyle

```
ListPlot[punti, PlotStyle -> RGBColor[1, 0, 0], Joined -> True]
```

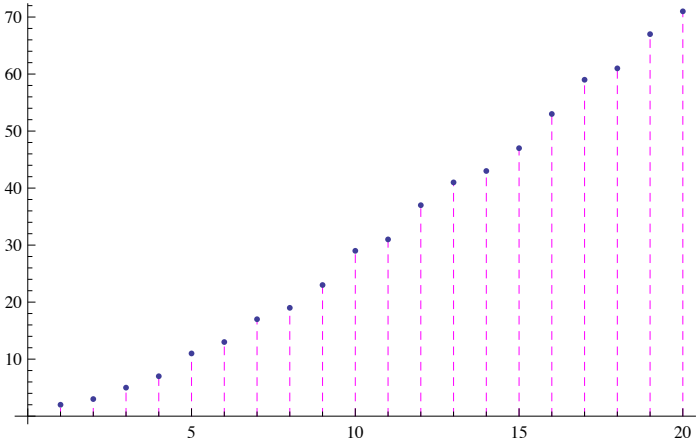


```
ListPlot[punti, PlotStyle -> {RGBColor[1, 0, 0], PointSize[0.03]}]
```



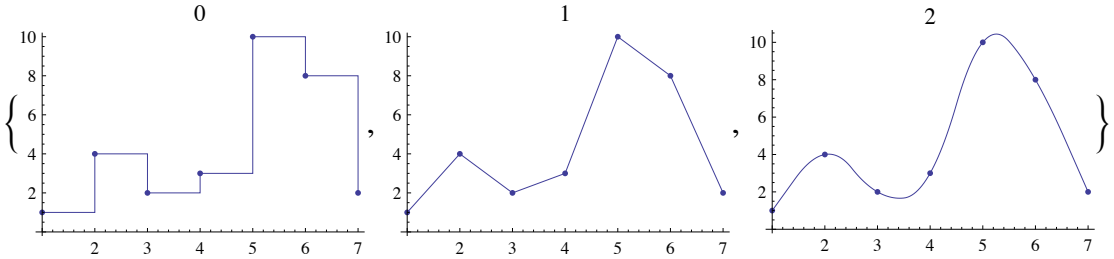
Filling

```
ListPlot[Table[Prime[n], {n, 20}], Filling → Axis, FillingStyle → Directive[Magenta, Dashed]]
```



InterpolationOrder

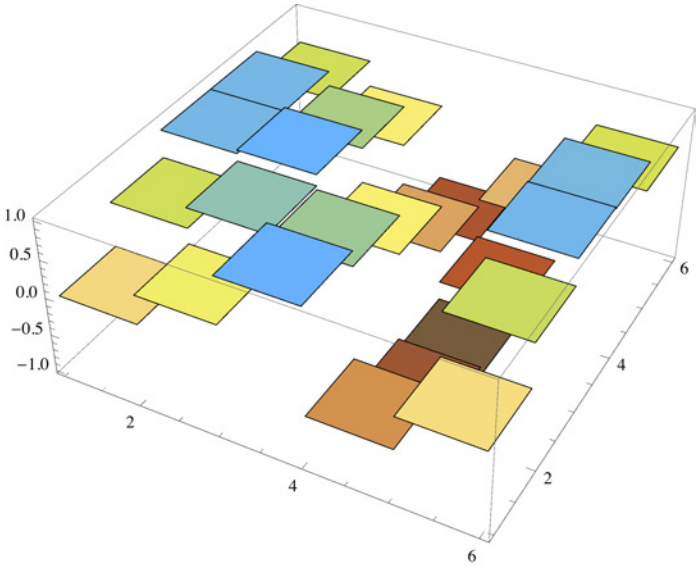
```
Table[ListPlot[{{1, 4, 2, 3, 10, 8, 2}}, Joined → True,
   InterpolationOrder → i, Mesh → Full, PlotLabel → i], {i, 0, 2}]
```



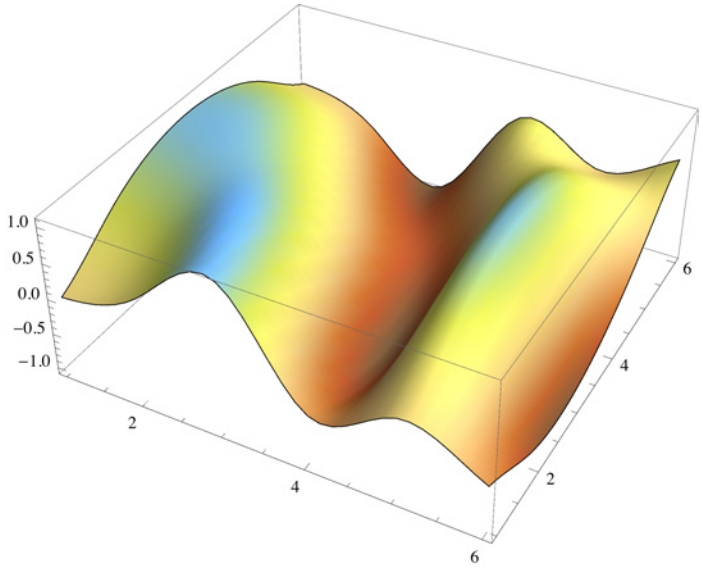
InterpolationOrder per ListPlot3D

```
data = Table[Sin[j ^ 2 + i], {i, 0, Pi, Pi / 5}, {j, 0, Pi, Pi / 5}];
```

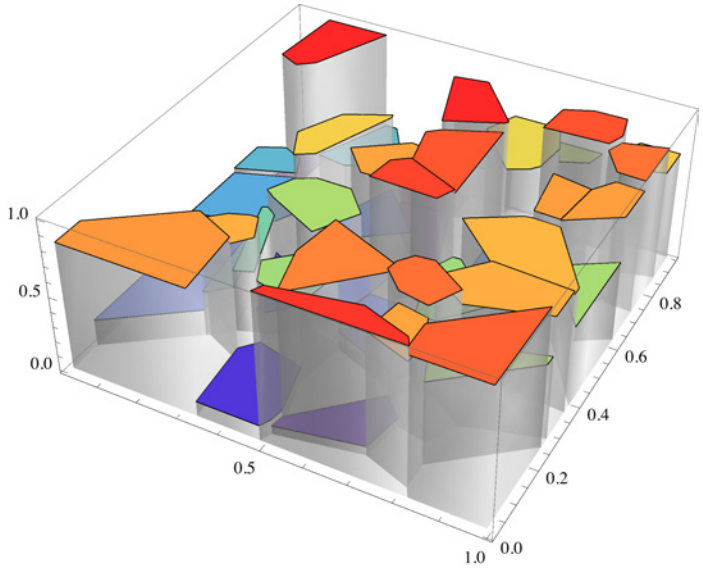
```
ListPlot3D[data, Mesh → None, InterpolationOrder → 0, ColorFunction → "SouthwestColors"]
```



```
ListPlot3D[data, Mesh → None, InterpolationOrder → 3, ColorFunction → "SouthwestColors"]
```




```
ListPlot3D[RandomReal[{}, {50, 3}], InterpolationOrder → 0,
 Filling → Bottom, Mesh → None, ColorFunction → "Rainbow"]
```



Drawing tool: come modificare i grafici manualmente

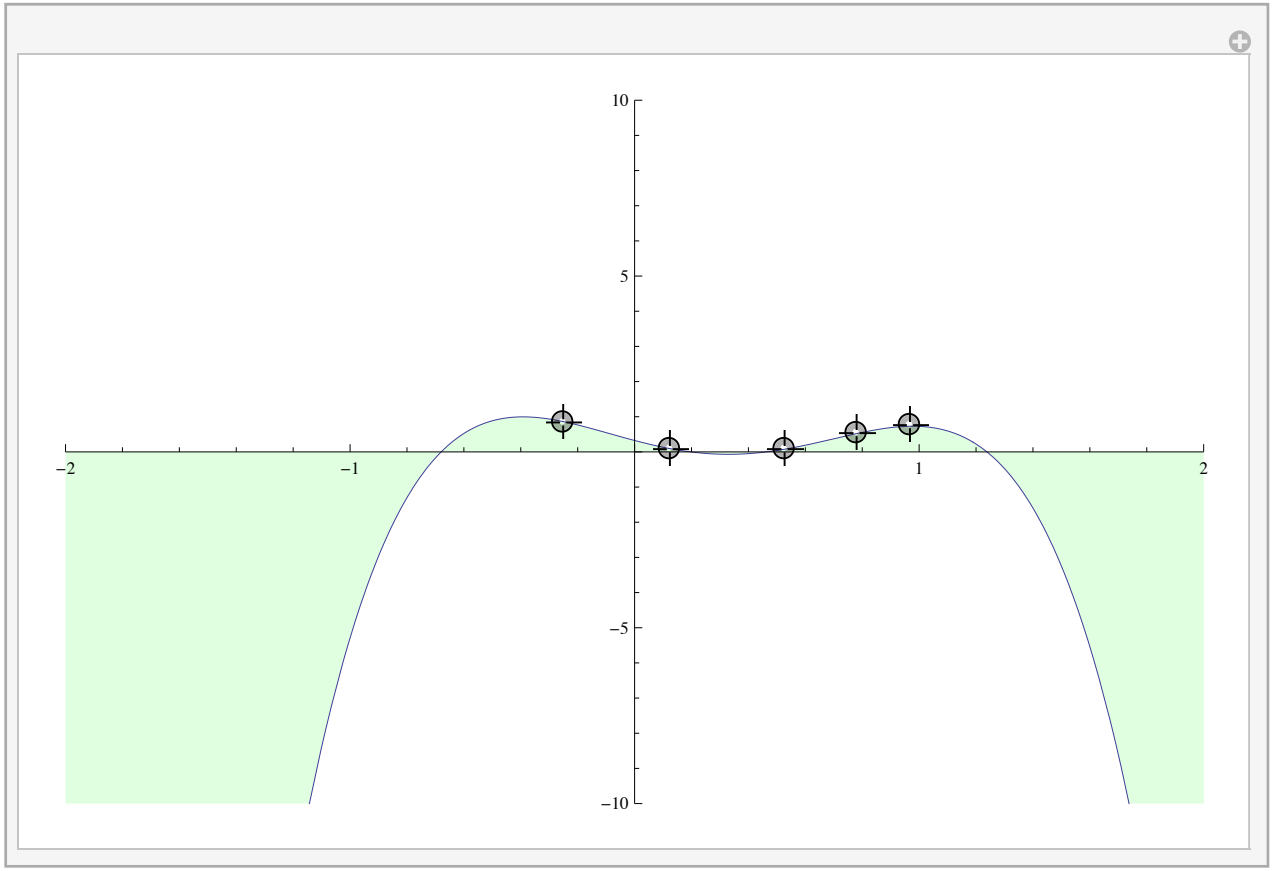
I grafici si possono personalizzare anche dopo averli realizzati, tramite la palette Drawing Tools (menu Graphics)

Qualche caso particolare di visualizzazione

I grafici si prestano bene alla realizzazione di applicazioni interattive semplici da usare ma molto efficaci nel trasmettere concetti sintesi sui dati.

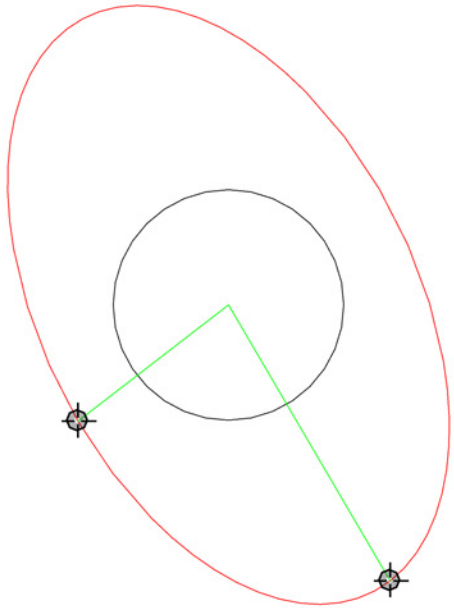
Esempio di interpolazione:

```
Manipulate[
 Plot[InterpolatingPolynomial[pts, x], {x, -2, 2}, PlotRange → {{-2, 2}, {-10, 10}},
  Filling → Axis, FillingStyle → LightGreen, ImageSize → 600],
 {{pts, {{-0.25, 0.86}, {0.53, 0.08}, {0.97, 0.73}, {0.12, 0.10}, {0.78, 0.52}}},
  {-2, -10}, {2, 10}, Locator, LocatorAutoCreate → True}]
```



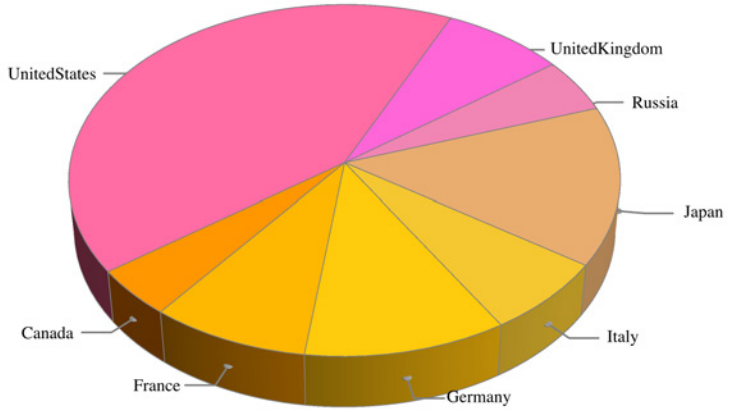
Trasformare una circonferenza utilizzando la matrice definita dalle coordinate di due punti nel piano

```
DynamicModule[{v1 = {2, 0}, v2 = {-1, 1}},
 Dynamic@Graphics[{Circle[], Red, GeometricTransformation[Circle[], Transpose[{v1, v2}]],
    Green, Line[{{0, 0}, v1}], Line[{{0, 0}, v2}],
    Locator[Dynamic[v1]], Locator[Dynamic[v2]]}, PlotRange → 3]]
```



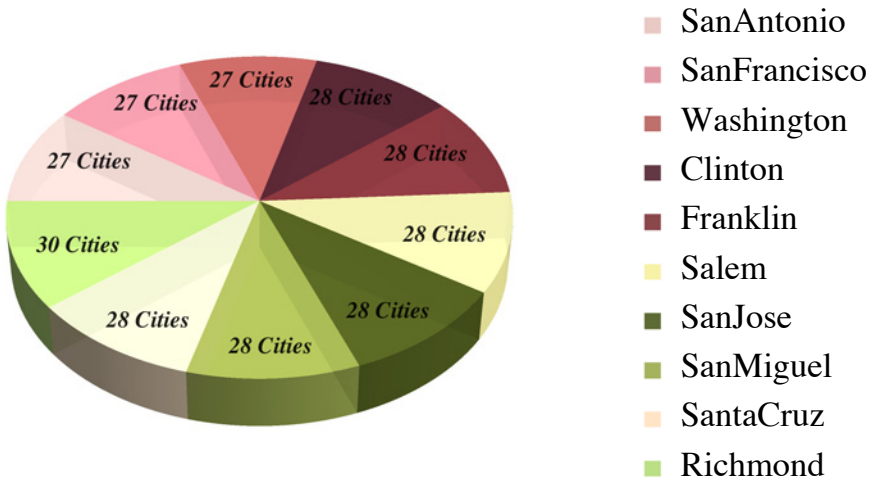
Grafici a torta

```
PieChart3D[CountryData[#, "GDP"] & /@ CountryData["G8"], SectorOrigin → 1.2 Pi,
 LabelingFunction → (Placed[NumberForm[#, 2, NumberPadding → {"$", ""}], Tooltip] &),
 ChartStyle -> "FruitPunchColors",
 ChartLabels → Placed[CountryData["G8"], "RadialCallout"], PlotRange → All]
```



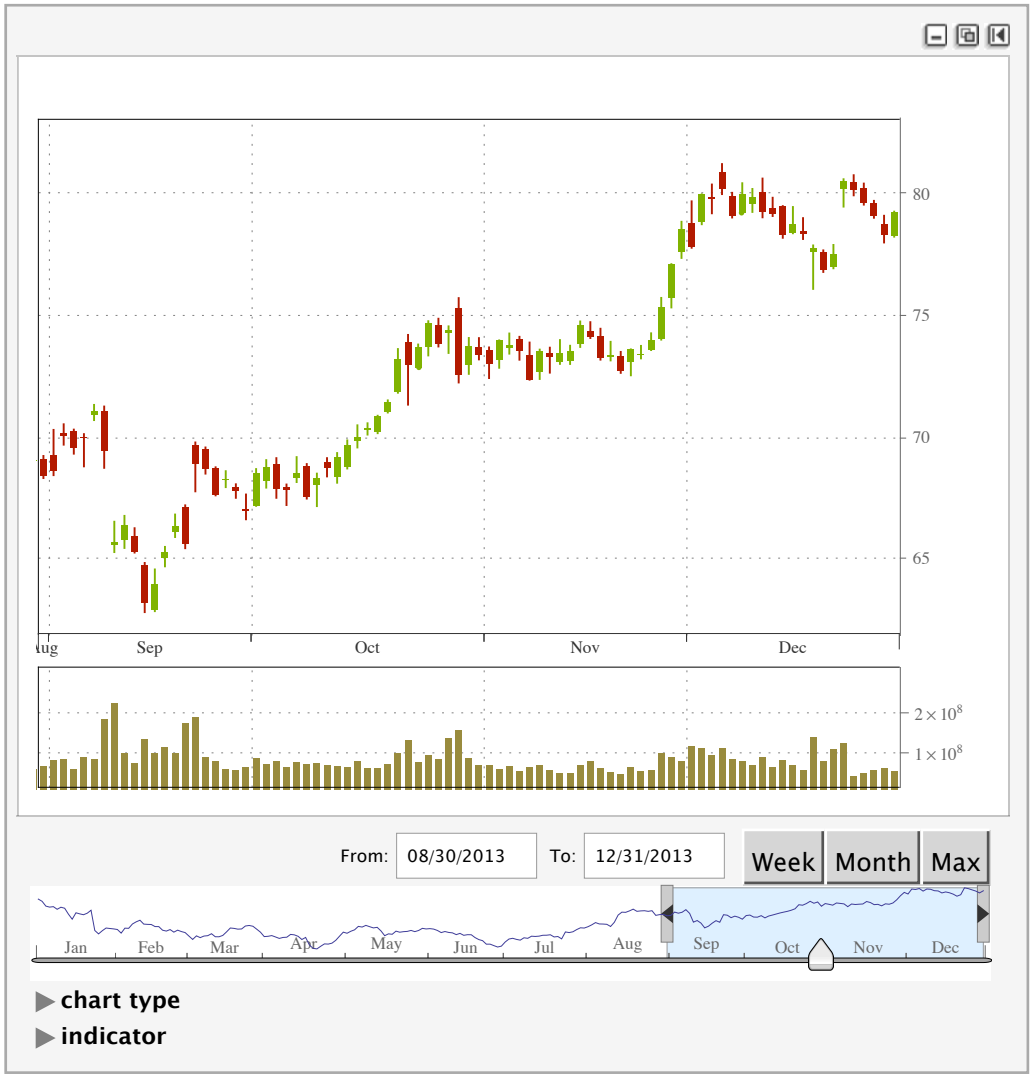

```
city = Take[SortBy[Tally[First /@ CityData[]], Last], -10];
```

```
PieChart3D[city[[All, 2]], LabelingFunction →
  (Placed[ToString[#] ~~ " Cities", "RadialOuter", Style[#, 12, Bold, Italic] &] &),
 ChartStyle → 62, ChartLegends → city[[All, 1]],
 ChartBaseStyle → {Opacity[0.9], EdgeForm[None]}, ImageSize -> Medium]
```



Qualcosa di ancora più complesso (InteractiveTradingChart e Monitor)

```
InteractiveTradingChart[{"AAPL", {{2013, 1, 1}, {2013, 12, 31}}}]
```




```
values = {};
Monitor[NIntegrate[Sqrt[x (1 - x)], {x, 0, 1},
  EvaluationMonitor :→ (Pause[0.025]; AppendTo[values, x];)], Quiet@ListPlot[values]]
```

0.392699

Conclusioni

Come molte altre caratteristiche, la grafica trae vantaggio dall'insieme ampio ed unico di funzionalità integrate nel linguagg *Mathematica*. Avere la possibilità di rappresentare grafici dinamici, che si alimentano da fonti dati esterne o che si possor manipolare agevolmente tramite cursori e controller è certamente un modo nuovo di vedere la visualizzazione dei dati in ambi tecnico scientifico e per molti aspetti è reso possibile solo grazie alle numerose funzionalità uniche di *Mathematica*.