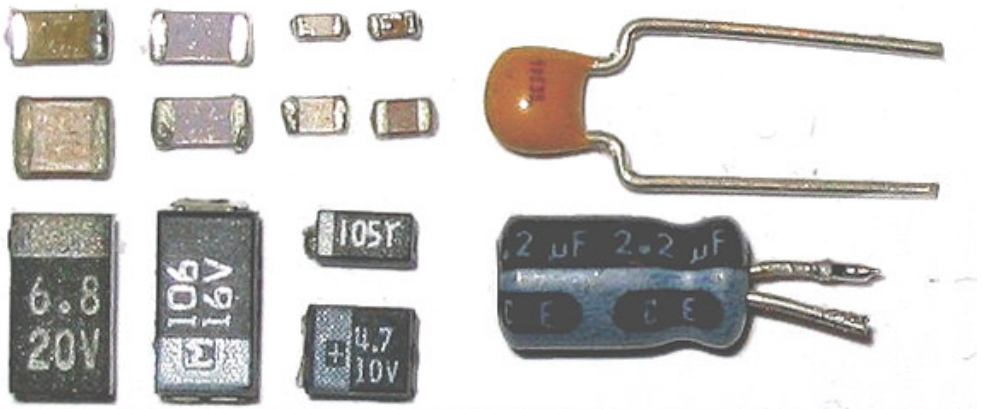
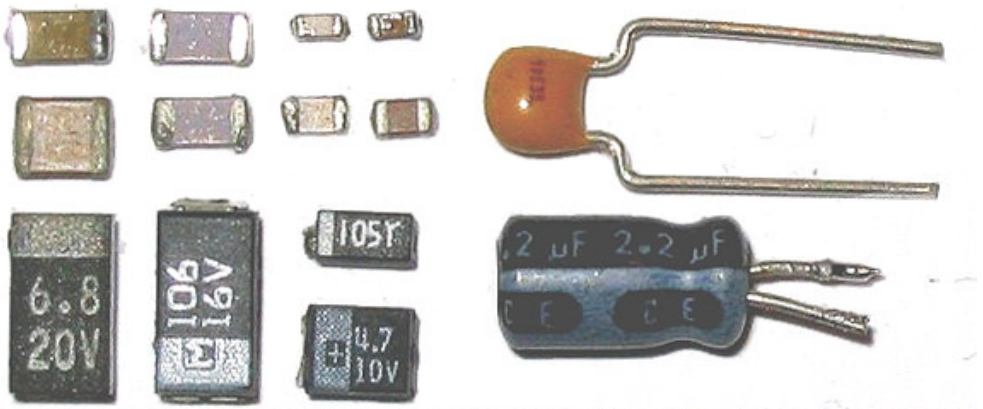
Università degli studi di Trieste

Grafici con SAGE

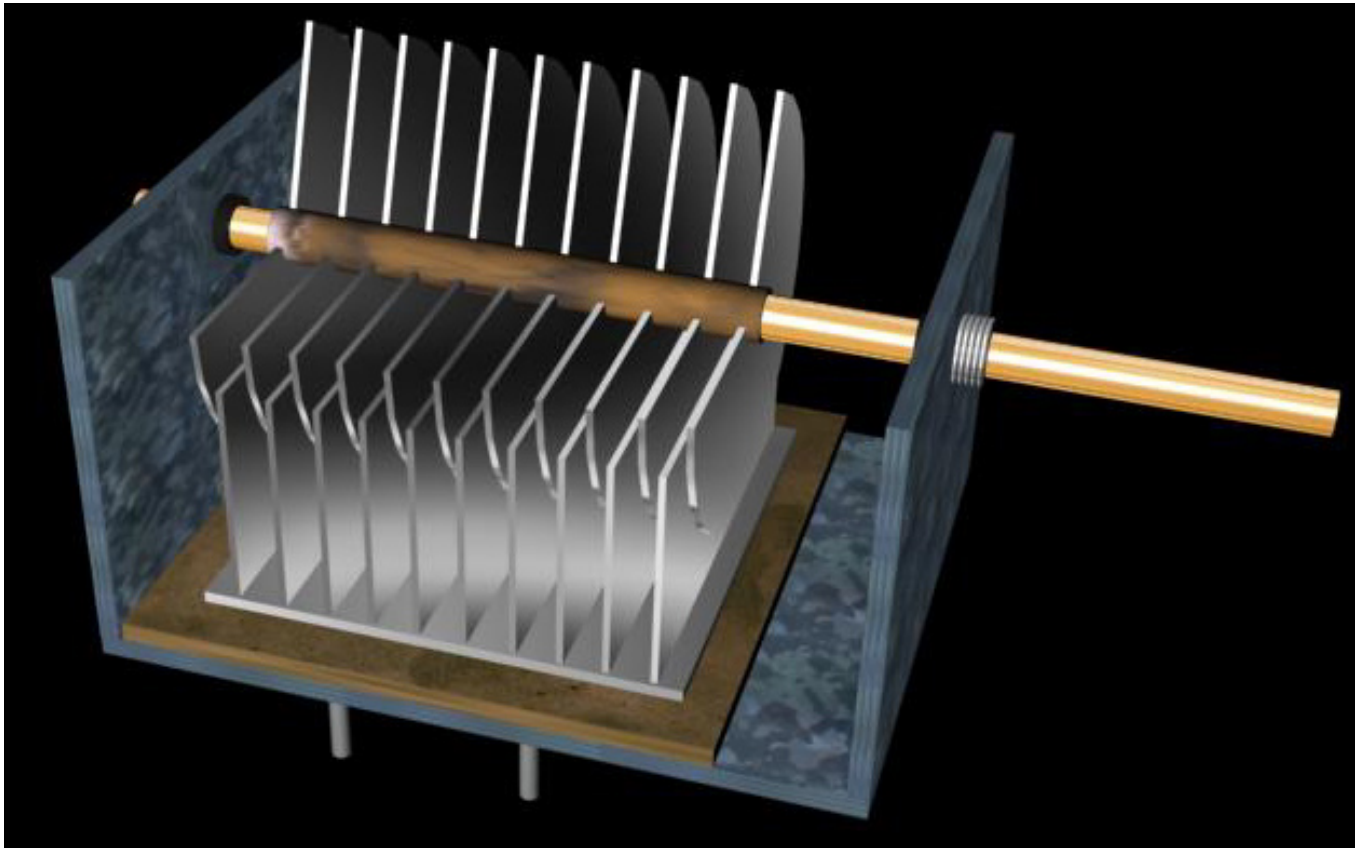
Stefano Piani

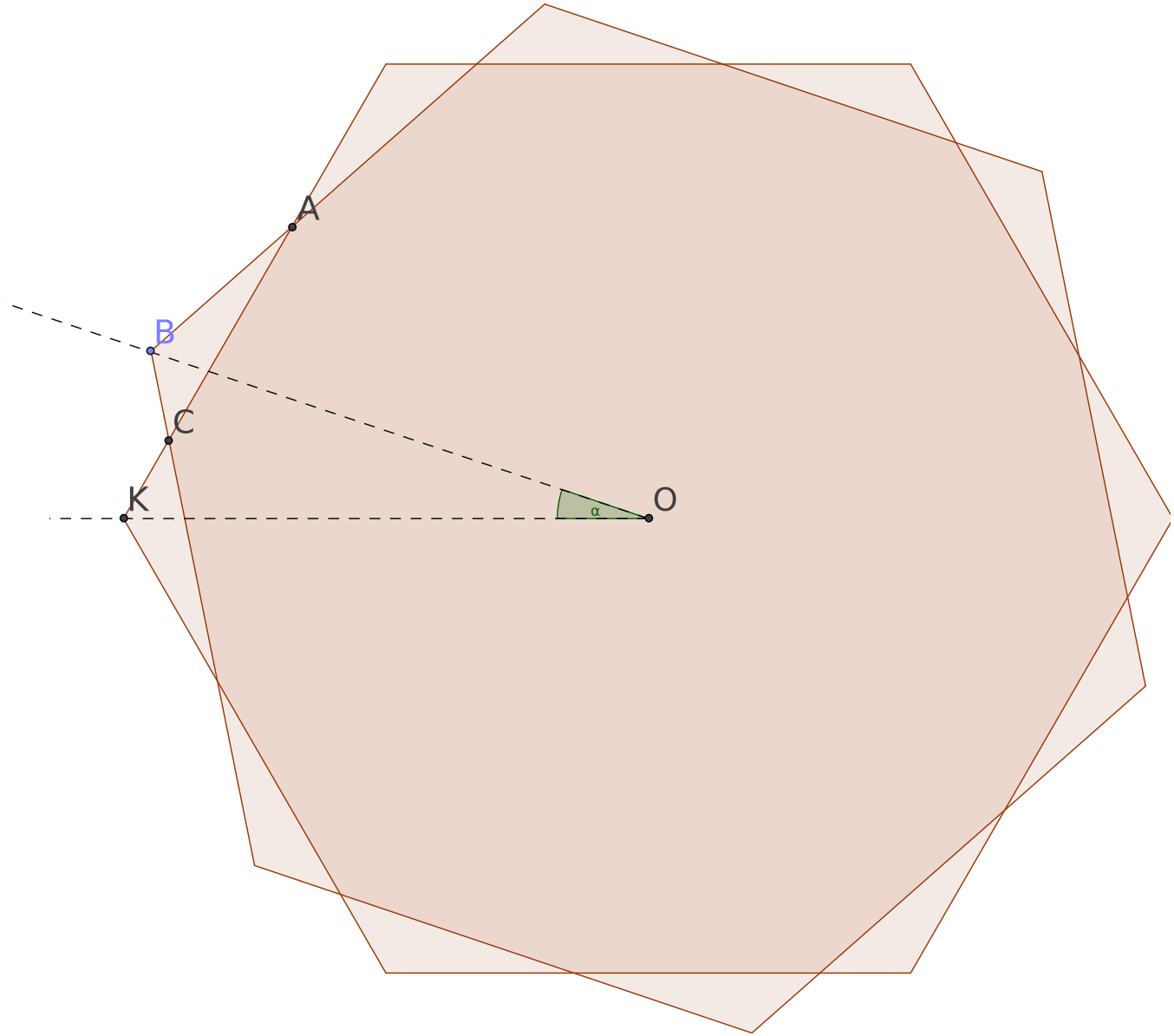
12 maggio 2014


6.8
20V
106
16V
105
4.7
10V
2 µF
2.2 µF



Un *condensatore* è un dispositivo utilizzato in elettronica per immagazzinare corrente e carica.




A
B
C
K
O
α

Il comando `plot()`

```
plot(f, opzione=valore)
```

Il comando `plot()`

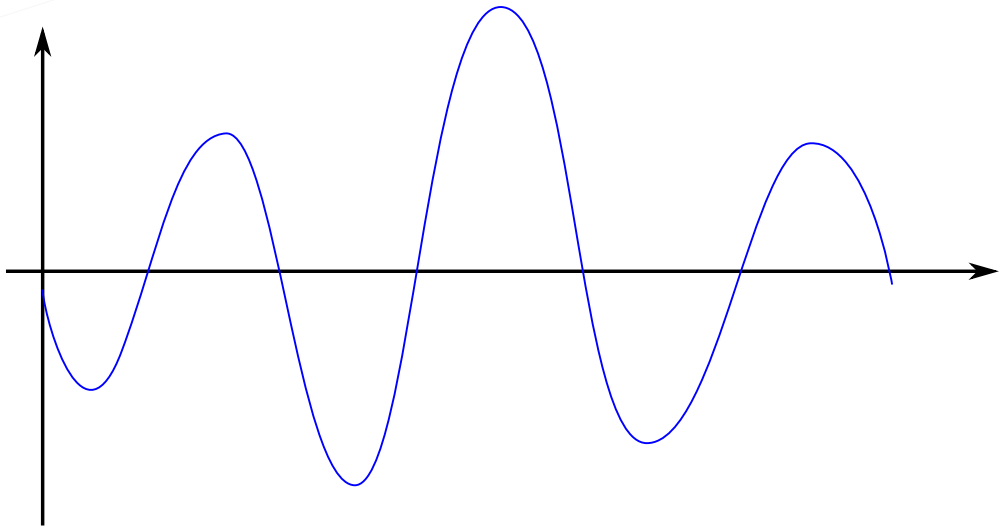
```
plot(f, opzione=valore)
```

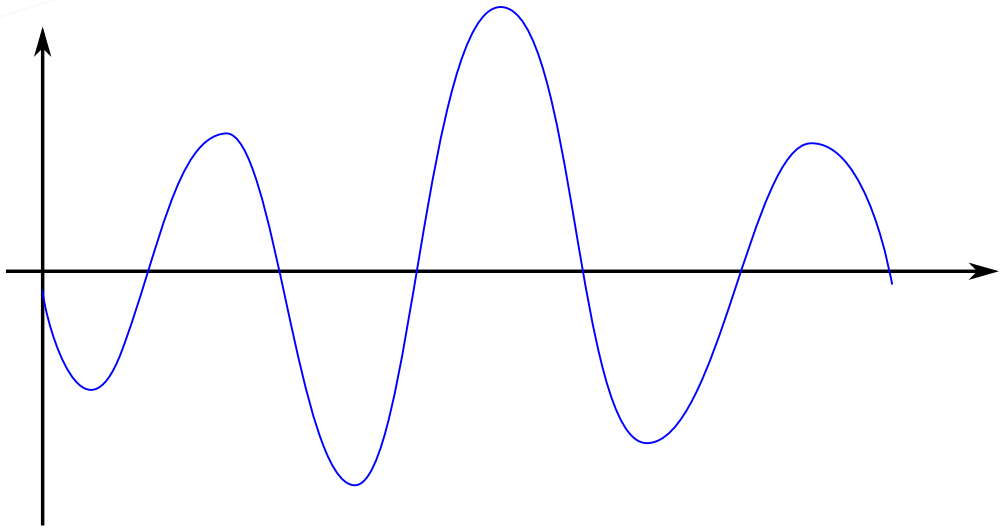
Le opzioni sono:

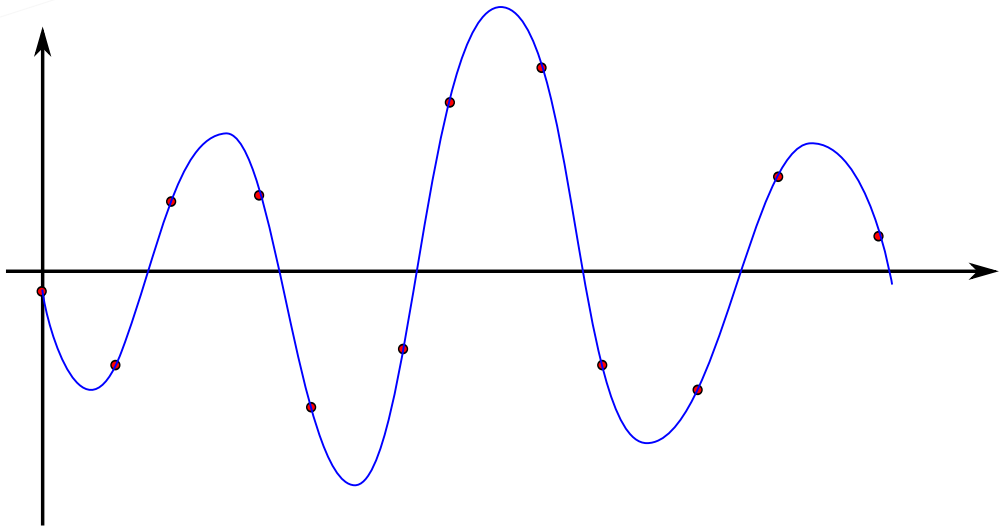
- xmin
- xmax
- color
- ymin
- ymax
- scale (da scegliersi tra “linear”, “semilogx”, “semilogy”, “loglog”)
- plot_points

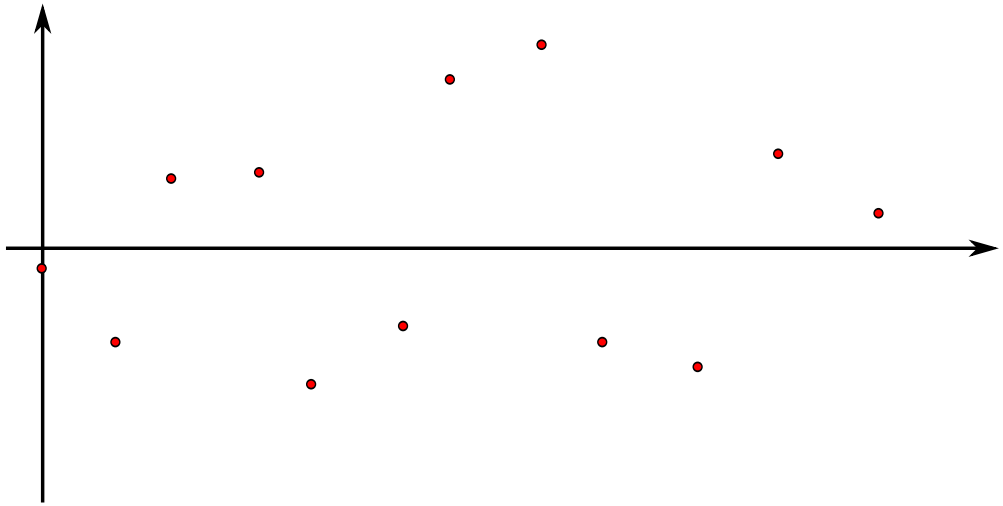
Le stesse opzioni (eccetto plot_points) valgono anche per i comandi:

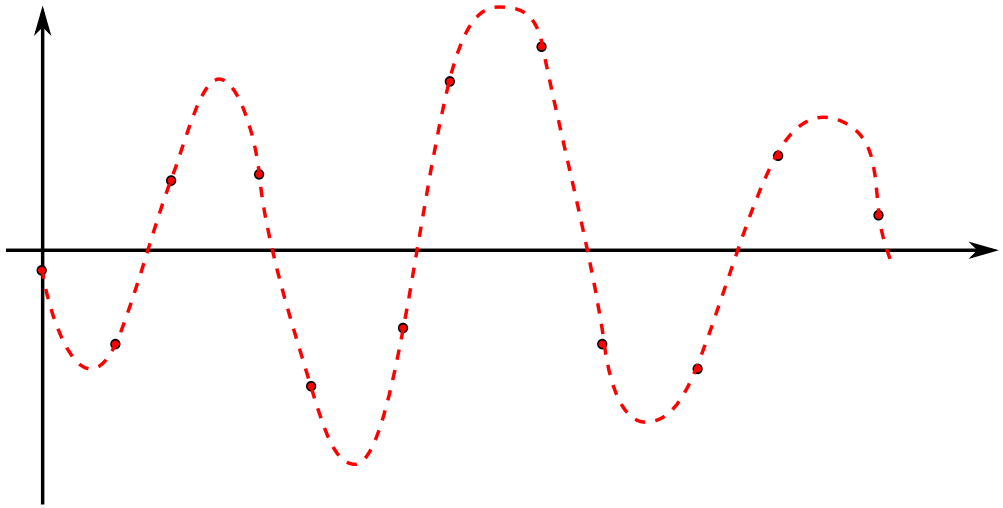
- `parametric_plot((f(t),g(t)), (t,0,2), opzione=valore)`
- `implicit_plot(f(x,y),(x,-3,3),(y,-3,3), opzione=valore)`
- `complex_plot(f(z),(-1, 2), (-3, 4), opzione=valore)`

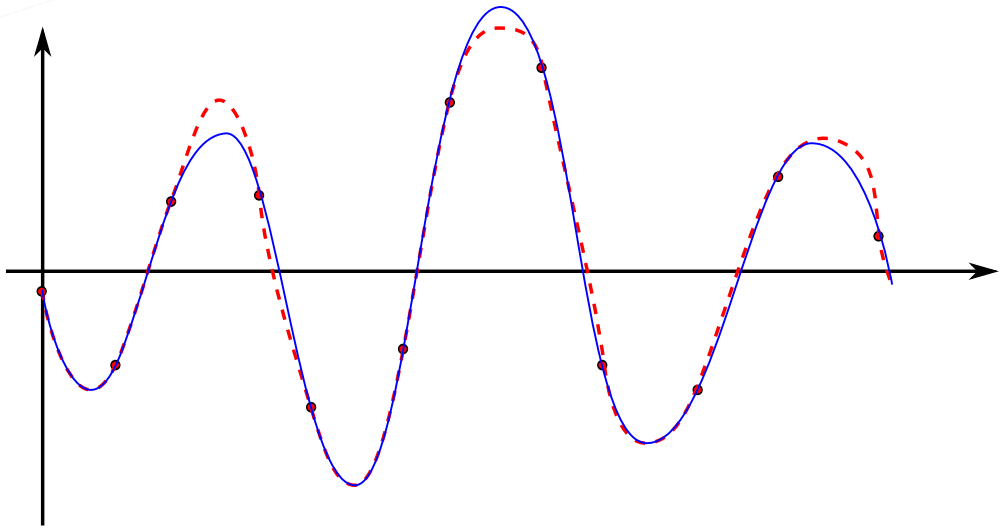












Problema

Data una sequenza di valori, trovare una funzione “liscia” che li congiunga tutti, cercando di ottenere una approssimazione dell'onda iniziale.

Problema

Data una sequenza di valori, trovare una funzione “liscia” che li congiunga tutti, cercando di ottenere una approssimazione dell'onda iniziale.

```
valori=[1,2,1,0,3,-1,0]
```

Comandi per disegno 2D

- point
- line

Comandi per disegno 2D

- point
- line
- arrow

Comandi per disegno 2D

- point
- line
- arrow
- circle
- disk
- ellipse
- arc

Comandi per disegno 2D

- point
- line
- arrow
- circle
- disk
- ellipse
- arc
- polygon
- text

- point

  `point((1,2))` oppure `point([(1,2),(3,4),...])`

  Opzioni:

  - `color='red'`
  - `size = 50`

- point

  `point((1,2))` oppure `point([(1,2),(3,4),...])`
  Opzioni:
  - `color='red'`
  - `size = 50`
- line

  `line([(1,2),(3,4),...])`
  Opzioni:
  - `color='red'`
  - `linestyle=`
    - `'-'` normale
    - `'-'` tratteggiata
    - `'-.'` tratto punto
    - `':'` a puntini
    - `'None'` o `' '` o `''` per visualizzare solo i punti

- circle

  `circle(centro,raggio)`

  Opzioni:
  - `facecolor='red'`
  - `edgecolor='blue'`
  - `filled=False`
  - `thickness = 1`

- circle

  `circle(centro,raggio)`

  Opzioni:
  - `facecolor='red'`
  - `edgecolor='blue'`
  - `filled=False`
  - `thickness = 1`
- polygon

  `polygon([(0,0), (1,1), (0,1)])`

  Opzioni:
  - `color='red'`
  - `fill = True`

Esercizio

Disegnate il grafico di $\sin(x)$ e $\cos(x)$ (sovrapposti) nell'intervallo $[-2\pi, 2\pi]$. Il grafico di $\sin(x)$ deve essere in rosso mentre il grafico di $\cos(x)$ deve essere in blu. Evidenziate i punti dove i due grafici si incrociano con dei punti gialli.

Quanto detto fin'ora si si riporta perfettamente anche al caso 3d!

Quanto detto fin'ora si si riporta perfettamente anche al caso 3d! Ad esempio il seguente codice

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),(y,0,4))
```

realizza un grafico 3d.

Quanto detto fin'ora si si riporta perfettamente anche al caso 3d! Ad esempio il seguente codice

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),(y,0,4))
```

realizza un grafico 3d.
Chi realizza queste immagini?

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

- jmol
- tachyon
- java3d
- canvas3d

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

- jmol
- tachyon
- java3d
- canvas3d

Si decide quale usare con l'opzione *viewer*

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),
                    (y,0,4), viewer='tachyon')
```

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

- jmol
- tachyon
- java3d
- canvas3d

Si decide quale usare con l'opzione *viewer*

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),
                    (y,0,4), viewer='tachyon')
```

Per usare *jmol* a volte e' necessario dare prima il comando

```
sage.plot.plot.EMBEDDED_MODE=False
```

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

- jmol
- tachyon
- java3d
- canvas3d

Si decide quale usare con l'opzione *viewer*

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),
                    (y,0,4), viewer='tachyon')
```

Per usare *jmol* a volte e' necessario dare prima il comando

```
sage.plot.plot.EMBEDDED_MODE=False
```

Canvas3d riporta solo i contorni e non è adatto ai grafici.

Sage usa 4 diverse tecnologie per realizzare i suoi grafici:

- jmol
- tachyon
- java3d
- canvas3d

Si decide quale usare con l'opzione *viewer*

```
y = var('y')
plot3d(sin(x)*log(y),(x,-4,4),
                    (y,0,4), viewer='tachyon')
```

Per usare *jmol* a volte e' necessario dare prima il comando

```
sage.plot.plot.EMBEDDED_MODE=False
```

Canvas3d riporta solo i contorni e non è adatto ai grafici.
Java3d non funziona MAI!

`plot`	$\rightarrow$	`plot3d`
`parametric_plot`	$\rightarrow$	`parametric_plot3d` `parametric_surface`
`implicit_plot`	$\rightarrow$	`implicit_plot3d`
`line`	$\rightarrow$	`line`
`circle`	$\rightarrow$	`sphere`
`rectange`	$\rightarrow$	`frame`
		Platonic Solids (`cube`)

Esempio

Scrivere una funzione “piano_tangente” che riceva in input una funzione in due variabili e un suo punto e restituisca un grafico contenente la funzione e il piano tangente a quella funzione passante per il punto.

Animazioni

In SAGE è possibile creare animazioni a partire da una lista di grafici. Per esempio

```
L = [plot(x), plot(-x), plot(x^2)]
ani = animate(L)
ani.show()
```

Animazioni

In SAGE è possibile creare animazioni a partire da una lista di grafici. Per esempio

```
L = [plot(x), plot(-x), plot(x^2)]
ani = animate(L)
ani.show()
```

Per tenere "fissi" gli assi è conveniente passare le seguenti opzioni ad animate

```
ani = animate(L, xmin=-1, xmax=-1, ymin=-1, ymax=1)
```

Animazioni

In SAGE è possibile creare animazioni a partire da una lista di grafici. Per esempio

```
L = [plot(x), plot(-x), plot(x^2)]
ani = animate(L)
ani.show()
```

Per tenere "fissi" gli assi è conveniente passare le seguenti opzioni ad animate

```
ani = animate(L, xmin=-1, xmax=-1, ymin=-1, ymax=1)
```

Inoltre è possibile cambiare la velocità di esecuzione con l'opzione "delay" nel seguente modo

```
ani.show(delay=100)
```

Il valore di "delay" è espresso in centesimi di secondo.

Esempio

Visualizzare una animazione che rappresenti il fascio di coniche

$$x^2 + ky^2 = 1$$

al variare di k tra -1 e 1.

Esempio

Visualizzare una animazione che rappresenti il fascio di coniche

$$x^2 + ky^2 = 1$$

al variare di k tra -1 e 1.

Per salvare il risultato ci sono due possibilità:

- `ani.ffmpeg(savefile='C:\grafico.avi')`
- `ani.gif(savefile='C:\grafico.gif')`

Esercizio

Scrivere una funzione che, dati in input le coordinate di 3 punti nel piano, disegni tre cerchi aventi centri in tali punti e tra loro a 2 a 2 tangenti (quando possibile).

Esercizio

Scrivere una funzione che dato un punto P, un raggio r e un numero naturale $n > 3$ disegni il poligono regolare di n lati inscritto nella circonferenza di centro P avente raggio r. Si aggiunga un'opzione che permetta di avere invece in output un'animazione per cui questo poligono ruota attorno al suo centro.

Esercizio

Creare una funzione "piano_tangente_implicito", simile alla funzione "piano_tangente", che prenda in input una superficie tridimensionale definita da una equazione implicita ed un punto e restituisca il grafico del piano tangente in quel punto.