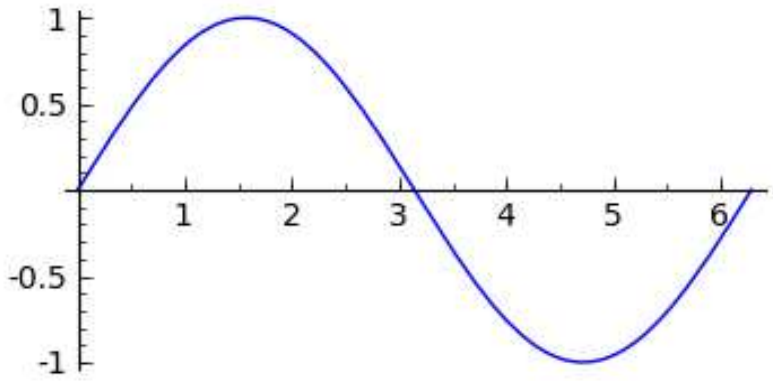
2-5-Crtanje_grafova-S

Verzija 1.1, 2010-10-28, K. Kumerički

Crtanje grafova

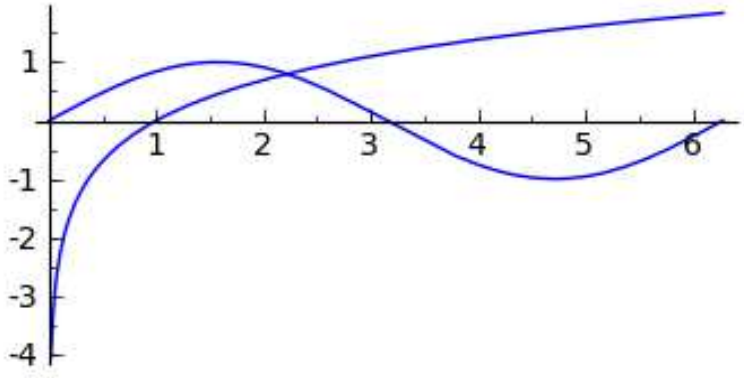
Glavna naredba za crtanje 2D grafova je plot()

```
plot(sin(x), (x,0,2*pi), figsize=[4,2])
```



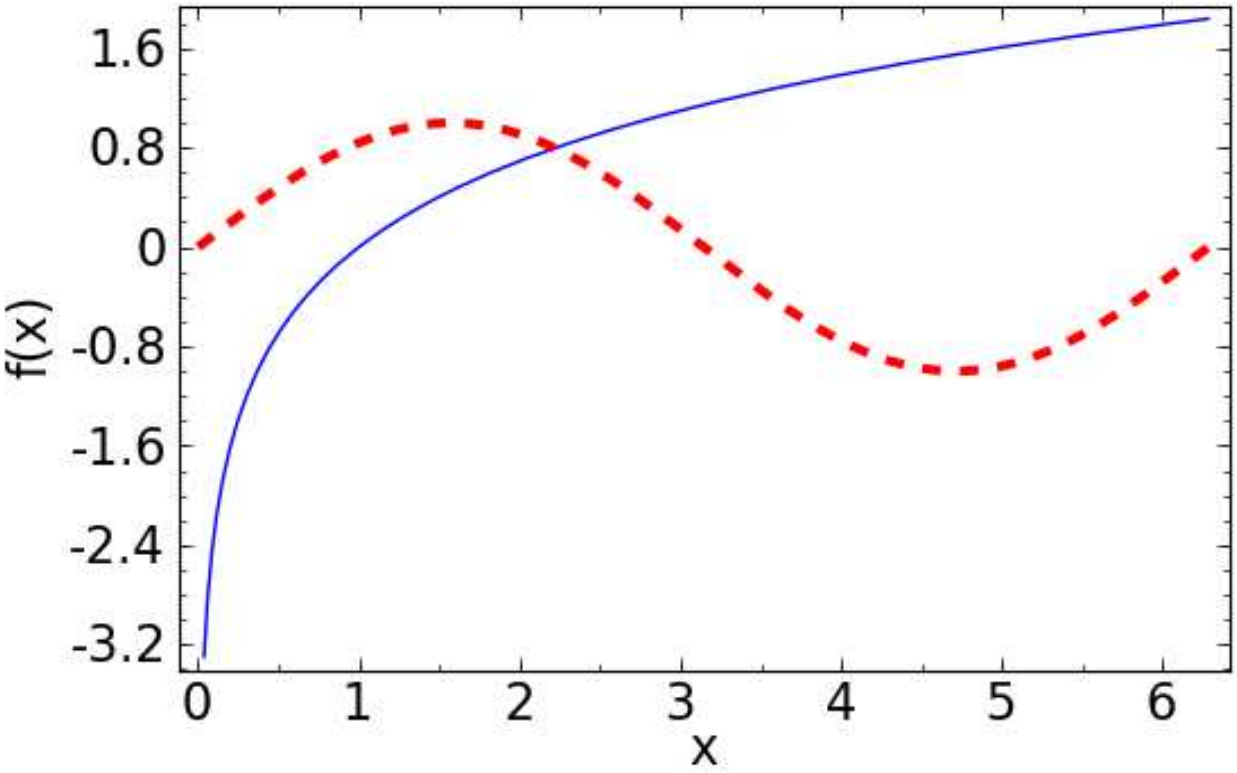
Za kombiniranje plotova može se kao prvi argument staviti lista funkcija ...

```
plot([sin(x), log(x)], (x,0,2*pi), figsize=[4,2])
```



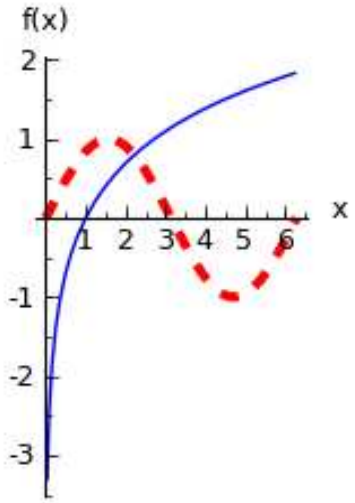
... ali je bolje naprosto zbrajati grafove pojedinih funkcija, što nam onda daje kontrolu nad svojstvima pojedinih linija (boja, debljina, ...).

```
P = plot(log(x), (x,0,2*pi))
P += plot(sin(x), (x,0,2*pi), color='red', linestyle='--',
thickness=3)
P.show(axes_labels=['x', 'f(x)'], frame=True, axes=False,
fontsize=16)
```



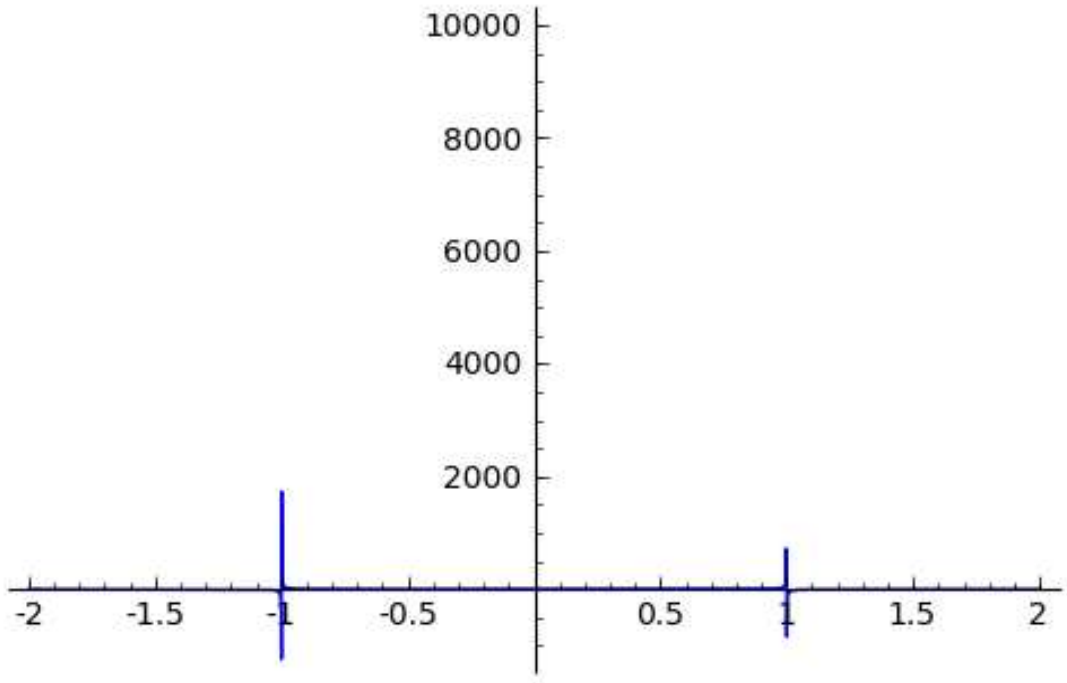
Premda to nije obavezno, gore je bilo logično svojstva pojedinih linija stavljati kao argumente funkcija plot(), a svojstva grafa kao cjeline u završni show(). Inače, korištenjem metode show() moguće je jednostavno ponovno iscrtavanje grafa uz promjenu nekih opcija:

```
P.show(frame=False, axes=True, aspect_ratio=2, fontsize=9)
```



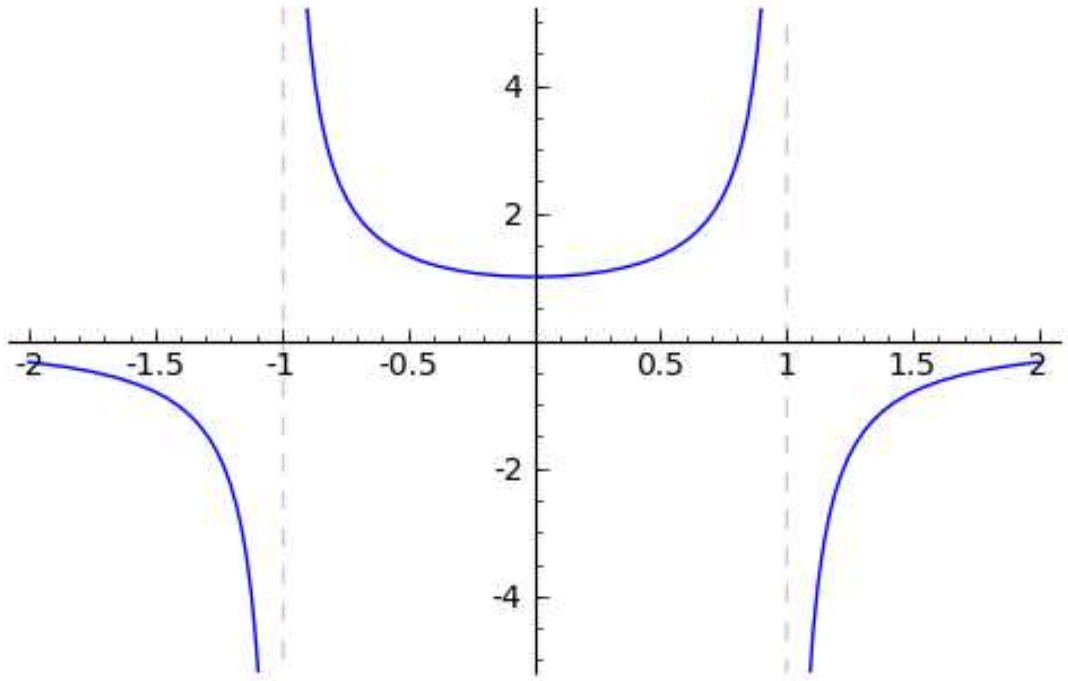
plot() sam određuje raspon vrijednosti ordinate pogodan za crtanje zadanih funkcija. Međutim, ukoliko funkcija ima singularitet u području crtanja, to ispada loše:

```
plot(1/(1-x^2), (x,-2,2))
```



Tada moramo eksplicitno ograničiti ordinatu opcijama ymin and ymax. (A može se i lijepo označiti položaj polova opcijom detect_poles.)

```
plot(1/(1-x^2), (x,-2,2), detect_poles='show', ymin=-5, ymax=5)
```



Grafove često želimo upotrijebiti i izvan Sage radnog lista, npr. u nekom članku ili prezentaciji. Eksportiranje grafova i drugih objekata postiže se uporabom funkcije save(). Format grafičke datoteke određen je ekstenzijom. Dopuštene ekstenzije su `.png`, `.ps`, `.eps`, `.svg`, and `.sobj`.

```
P.save('/tmp/graf.png')
```

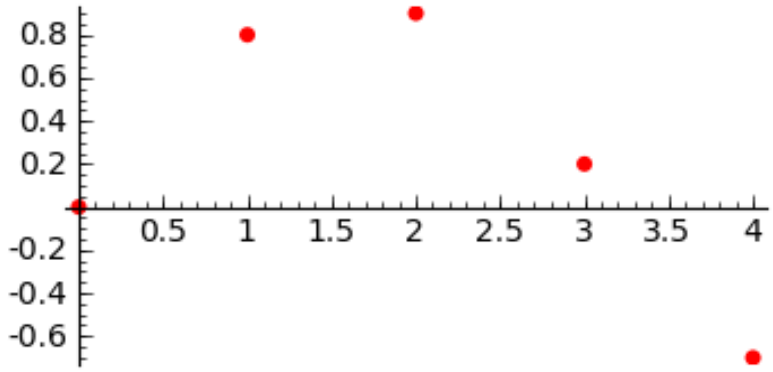
 Zadatak 2-5.1: Pronađite ovu datoteku na disku i prikažite je pomoću nekog programa za pregledavanje slika. (Na Windowsima je potrebno pristupiti virtualnoj Linux mašini putem ssh protokola (winscp, putty, ...) spajanjem na adresu na koju se spaja i Windows WWW browser, uz user=sage, pass=sage. No, u učionici F-107 to nije moguće pa naprosto spremite sliku klikom desnim gumbom miša na nju, pa "Save Image As ...".)

 Zadatak 2-5.2: Nacrtajte graf funkcije log(x) između x=0.5 i x=1.5 bez ikakvih oznaka, okvira i osi, dakle samo liniju.

 Zadatak 2-5.3: Koristeći funkciju parametric_plot() nacrtajte kružnicu. (na ekranu treba izgledati baš kao kružnica, a ne kao elipsa).

Za crtanje podataka organiziranih u listu parova $[[x_1, y_1], [x_2, y_2], \ldots]$ rabimo list_plot():

```
data = [[0,0], [1,0.8], [2, 0.9], [3, 0.2],[4, -0.7]]
list_plot(data, pointsize=20, color='red', figsize=[4,2])
```



Ovo je dovoljno za osnovne stvari. Funkcija plot() i ostale gore navedene su zapravo samo sučelje za tzv. matplotlib grafičku biblioteku. Za precizniju kontrolu nad crtanjem potrebno je ili izravno pozivati funkcije ove biblioteke ili, što ćemo mi raditi, koristiti modul *pylab*, koji ujedinjuje matplotlib i još neke važne pakete.

Pylab (tj. matplotlib) u načelu ne crta funkcije već samo skupove točaka (slično kao list_plot() gore), pa se crtanje linija dobiva iscrtavanjem i spajanjem gustih skupova točaka. Za kreiranje ovih točaka možemo koristiti pylab funkcije linspace() i logspace() koje ćemo i kasnije često koristiti. (Ove funkcije ne stvaraju obične liste već tzv. *numpy ndarray* (numpy=numerical python, ndarray=n-dimensional array) polja koja su interno efikasno implementirana u C programskom jeziku.)

```
import pylab   # učitavanje modula
```

```
t0 = [1, 2, 3]; print type(t0); print t0   # obična lista
t1 =pylab.linspace(1, 100, 5); print type(t1); print t1   #
numpy polje 1...100 s 5 linearno ekvidistantnih točaka
```

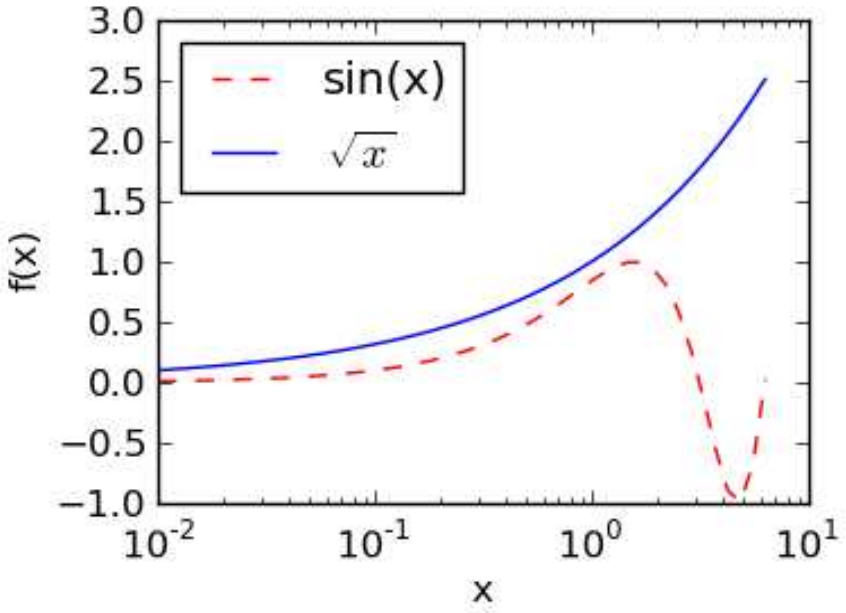
```
t2 = pylab.logspace(0, 2, 5); print type(t2); t2       # numpy
polje 10^0...10^2 s 5 logaritamski ekvidistantnih točaka
```

Ova polja imaju i zgodno svojstvo da se funkcije automatski distribuiraju po njima: $f([x_1, \ldots]) = [f(x_1), \ldots]$:

```
sqrt(t2)
```

Koristeći ovo, nacrtat ćemo graf dviju funkcija s logaritamskom apscisom. Graf treba inicijalizirati pylabovom funkcijom figure(), zatim se definiraju krivulje, osi, legenda itd. dok na kraju iscrtavanje dobijemo pozivom funkcije savefig().

```
pylab.figure(figsize=[4,3])
xs = pylab.logspace(-2, 0.8)                                  # 50
točaka je default
pylab.semilogx(xs, sin(xs), 'r--', label='sin(x)')       # za
boju i oblik linije postoje i skraćeni string argumenti
pylab.semilogx(xs, sqrt(xs), 'b', label='$\sqrt{x}$')
# labele se mogu formatirati i LaTeX-om
pylab.xlabel('x')
pylab.ylabel('f(x)')
pylab.legend(loc='upper left')
pylab.subplots_adjust(left=0.18, bottom=0.18)  # kontrolira
margine da se labele vide
pylab.savefig('fig')                                      # ime
'fig' je nebitno, ali mora postojati
```



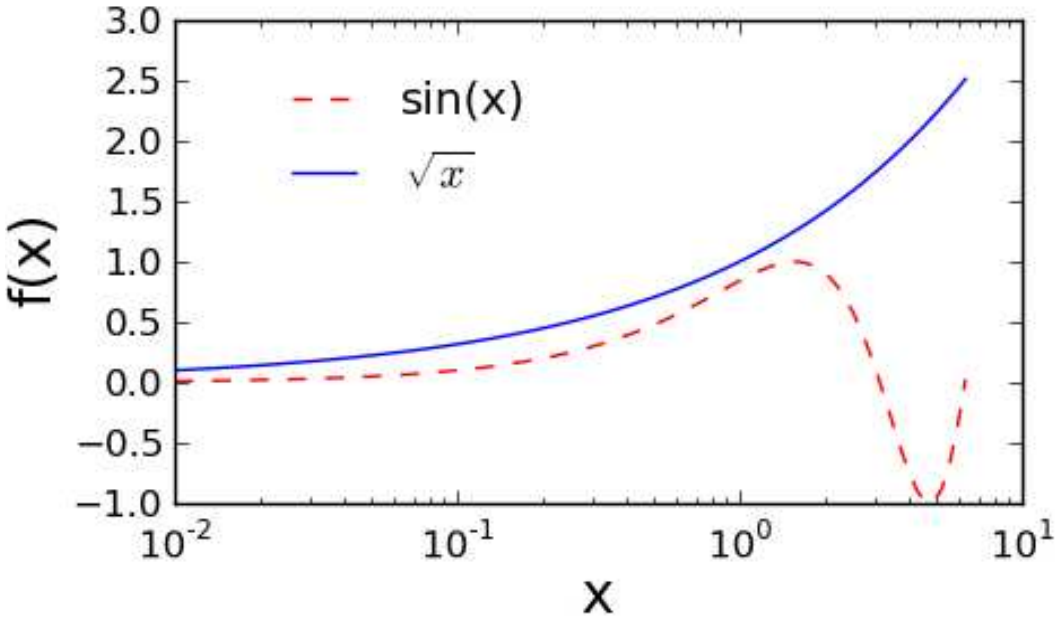
(*) Kod kopiranja primjera iz originalne pylab/matplotlib dokumentacije važno je znati da se iscrtavanje matplotlib grafova u Sage-u ne izvodi funkcijom show() već savefig() pa prema tome treba prilagoditi te primjere. Usput rečeno, savefig() sprema datoteku sa slikom u $HOME/.sage/sage_notebook.sagenb/home/kkumer/<kk>/cells/<nn>/imeslike.png, gdje je <kk> broj radnog lista vidljiv u URL-u ("WWW" adresi vidljivoj u WWW pregledniku), a <nn> je broj ćelije koji je vidljiv samo u tekstualnoj varijanti radnog lista (tipke "Text" ili "Edit" u notebooku), ali može se saznati i pomoću Unix shell komande 'find . -name "fig.png"' iz direktorija ...<kk>/cells.

(*) Radi informacije, pogledajmo kako se ova slika dobiva izravnim pozivima funkcija matplotlib i numpy biblioteka. Za razliku od svega gore, ovo dolje radi i kao samostojeća python skripta.

```
import matplotlib.pyplot as plt
import numpy as np
```

```
fig = plt.figure(figsize=[5,3])
ax = fig.add_subplot(1,1,1)             # moguce je dodati i
vise panela (subplot)  na jednu sliku
xs= np.logspace(-2.0, 0.8, 100)
ax.plot(xs, np.sin(xs), color='red', linestyle='--', label='sin
```

```
(x)')
ax.plot(xs, np.sqrt(xs), color='blue', label='$\sqrt{x}$')
ax.set_xscale('log')
ax.set_xlabel('x', fontsize=18)
ax.set_ylabel('f(x)', fontsize=18)
ax.legend(loc=(0.1,0.6)).draw_frame(0)     # loc = polozaj
legende u jedinicama dimenzije slike
fig.subplots_adjust(left=0.15, bottom=0.18)
fig.savefig('test')
```



Zadatak 2-5.4: Nacrtajte (pylabom) 100 slučajno raspoređenih točaka (hint: *pylab.rand()*, te *plot(..., linestyle='None', marker='o')*) s x i y koordinatama u intervalu (-1,1), na dijagramu s x i y osima u intervalu (-2, 2). (hint: *pylab.xlim()* i *pylab.ylim()*).

Za crtanje 3D grafova stoje na raspolaganju dvije softverske biblioteke:

- **JMOL** (default) traži da rade Java appleti u browseru (na Linuxu samo Sun Java funkcionira, što trenutno ne postoji za AMD64 arhitekturu (ovo možda nije točno) - vjerojatno ne u F-26).

- **Tachyon** - raytracing paket, radi bez Jave, nije moguć interaktivni rad s crtežom (okretanje mišem)

```
y = var('y')
```

```
P2 = plot3d(x^4+y^4, (x, -2, 2), (y, -2, 2)); P2.show()
```

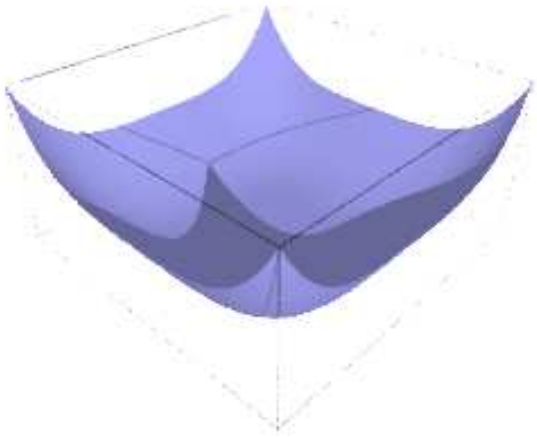
Plug-in content

Get Image

Klik mišem na sliku omogućuje zumiranje kotačićem miša. Desni klik otvara izbornik. Za "pravi 3D doživljaj" stavite dvobojne naočale pa onda desni klik -> Style -> Stereographic ->...

Ukoliko nemate mogućnost prikazivanja Java appleta, koristite 'tachyon':

```
P2.show(viewer='tachyon', figsize=[3,3])
```



```
L = plot3d(lambda x,y: 0, (-5,5), (-5,5), color="lightblue",
opacity=0.8)
P = plot3d(lambda x,y: 4 - x^3 - y^2, (-2,2), (-2,2),
color='green')
Q = plot3d(lambda x,y: x^3 + y^2 - 4, (-2,2), (-2,2),
color='orange')
(L + P + Q).show(viewer='tachyon', figsize=[3,3])
```