lektion8

December 3, 2025

Inhalt

1 Lektion 8

1.1 Wiederholung: Lösen von Gleichungen (solveset)

[1]:
```
%matplotlib inline
from sympy import *
import matplotlib.pyplot as plt
import numpy as np
init_printing()
x = symbols('x')
%config InlineBackend.figure_format = 'svg'
```

[2]:
```
gl = Eq((x-1)**2, 4-x)
gl
```

[2]: $(x-1)^2 = 4-x$

[3]:
```
gl.lhs
```

[3]: $(x-1)^2$

[4]:
```
gl.rhs
```

[4]: $4 - x$

[5]:
```
Lsgn = solveset(gl, x)
Lsgn
```

[5]: $\left\{\frac{1}{2} - \frac{\sqrt{13}}{2}, \frac{1}{2} + \frac{\sqrt{13}}{2}\right\}$

[6]:
```
# Test durch einsetzen
for lsg in Lsgn:
    display(gl.subs(x,lsg))
```

True

True

1.1.1 Wurzeln von Polynomen

[7]:
```
p = Eq(x**5 - 20*x + 7, 0)
Lsg = solveset(p)
Lsg
```

[7]: $\{\mathrm{CRootOf}\,(x^5 - 20x + 7, 0), \mathrm{CRootOf}\,(x^5 - 20x + 7, 1), \mathrm{CRootOf}\,(x^5 - 20x + 7, 2), \mathrm{CRootOf}\,(x^5 - 20x + 7, 3), \mathrm{CR}$

[8]:
```
LsgR = solveset(p, domain = Reals)
LsgR
```

[8]: $\{\mathrm{CRootOf}\,(x^5 - 20x + 7, 0), \mathrm{CRootOf}\,(x^5 - 20x + 7, 1), \mathrm{CRootOf}\,(x^5 - 20x + 7, 2)\}$

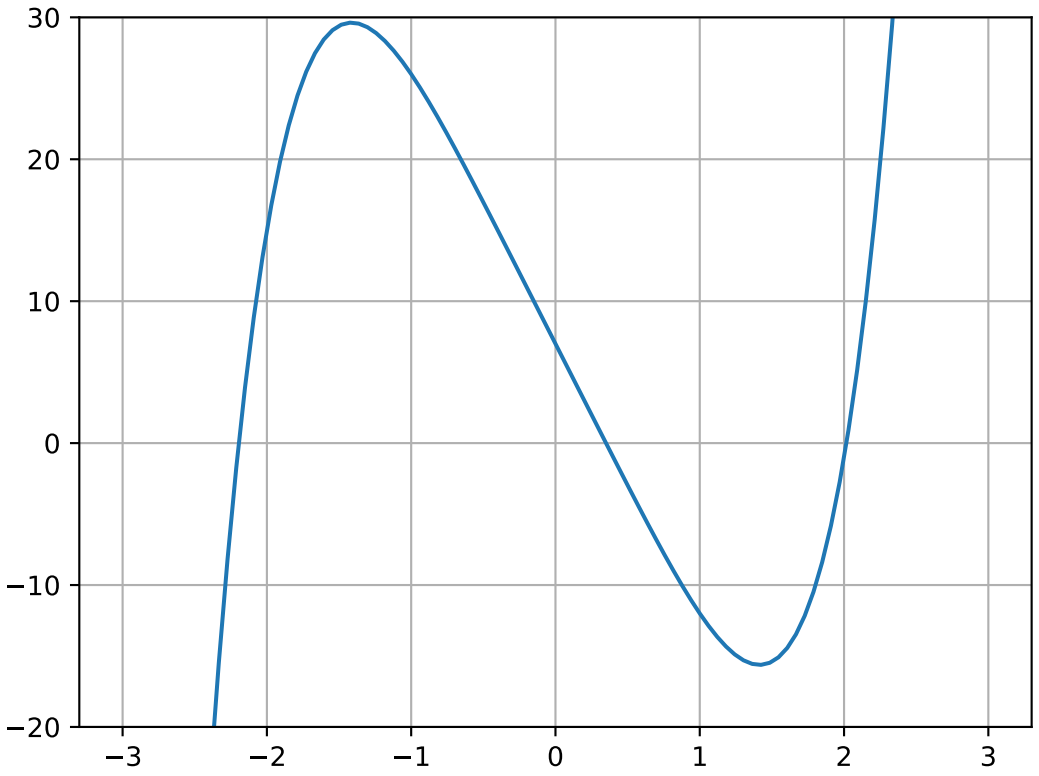
[9]:
```
[l.n() for l in LsgR] # CRootOf kann man numerisch auswerten
```

[9]: $[-2.19444446392152,\ 0.350263599776741,\ 2.01630911327803]$

[10]:
```
[l.n() for l in Lsg]
```

[10]: $[-2.19444446392152,\ 0.350263599776741,\ 2.01630911327803,\ -0.0860641245666237 - 2.12350978358086i,\ -0.08$

[11]:
```
fig, ax = plt.subplots()
xn = np.linspace(-3, 3, 100)
ax.plot(xn, lambdify(x, p.lhs)(xn))
ax.grid()
ax.set_ylim(-20, 30);
```

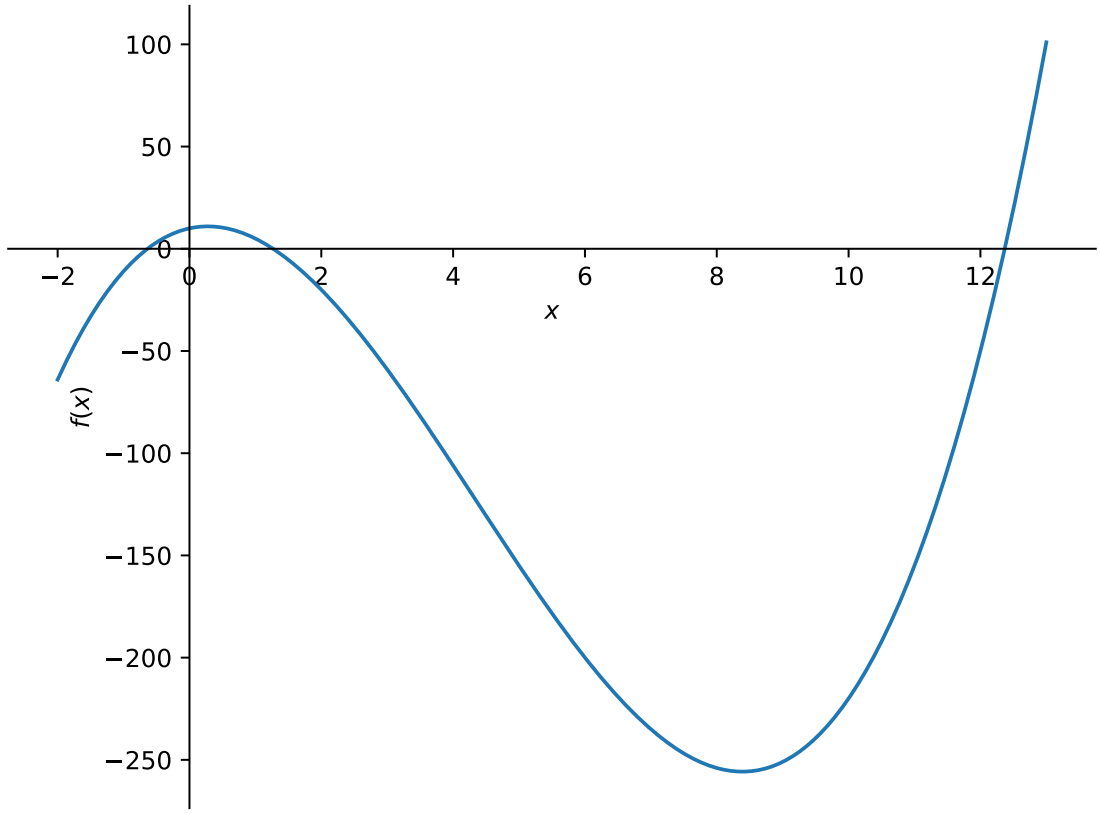


scheint zu passen :)

[12]:
```
p = x**3 - 13*x**2 + 7*x + 10
gl = Eq(p, 0)
lsgn = solveset(gl)
lsgn
```

[12]:
$$\left\{ \frac{13}{3} + \frac{4\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3}, -\frac{\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3} + \frac{148\,\mathrm{re}\left(\frac{1}{\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54}+\frac{\sqrt{227127}i}{18}}}\right)}{9} + \frac{\sqrt{111}\sin\left(\frac{\operatorname{atan}\left(\right.\right.}{3}\right.$$

[13]:
```
plot(p, (x, -2, 13));
```



[14]:
```
lsgnR = solveset(gl, domain=Reals)
lsgnR
```

[14]:
$$\mathbb{R}\cap\left\{\frac{13}{3}+\frac{4\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3},-\frac{\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3}+\frac{148\operatorname{re}\left(\frac{1}{\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54}+\frac{\sqrt{227127}i}{18}}}\right)}{9}+\frac{\sqrt{111}\sin\left(\frac{\text{at}}{}\right.\right.$$

das ist nicht hilfreich

[15]:
```
[trigsimp(l) for l in lsgn]
```

[15]:
$$\left[\frac{13}{3}+\frac{4\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3},\ -\frac{4\sqrt{37}\sin\left(\frac{\pi-2\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{6}\right)}{3}+\frac{13}{3},\ -\frac{4\sqrt{37}\sin\left(\frac{2\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)+\pi}{6}\right)}{3}+\frac{13}{3}\right]$$

[16]:
```
[simplify(p.subs(x, l)).n() for l in lsgn]
```

[16]:
$$[0,\ -5.0\cdot10^{-138}+5.0\cdot10^{-138}i,\ -5.0\cdot10^{-138}+5.0\cdot10^{-138}i]$$

[17]:
```
wurzeln = roots(p) # für Polynome kann man roots probieren
wurzeln
```

[17]:
$$\left\{\frac{13}{3}+\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54}+\frac{\sqrt{227127}i}{18}}+\frac{148}{9\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54}+\frac{\sqrt{227127}i}{18}}}:1,\ \frac{13}{3}+\frac{148}{9\left(-\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54}+\frac{\sqrt{2}}{}}}\right.$$

[18]:
```
[simplify(w) for w in wurzeln]
```

[18]:
$$\left[\frac{13}{3}+\frac{148\sqrt[3]{2}}{3\sqrt[3]{3305+3\sqrt{227127}i}}+\frac{2^{\frac{2}{3}}\sqrt[3]{3305+3\sqrt{227127}i}}{6},\ \frac{-1184\sqrt[3]{2}+\left(1-\sqrt{3}i\right)\left(52+2^{\frac{2}{3}}\left(-1+\sqrt{3}i\right)\sqrt[3]{3305+3\sqrt{}}\right.}{12\left(1-\sqrt{3}i\right)\sqrt[3]{3305+3\sqrt{227}}}\right.$$

Ein Trick, der manchmal weiterhilft

[19]:
```
[simplify(w.conjugate().conjugate() - w.conjugate()) for w in wurzeln]
```

[19]:
$[0,\ 0,\ 0]$

offenbar sind die Wurzeln reell

[20]:
```
[simplify((w.conjugate().conjugate() + w.conjugate()) / 2) for w in wurzeln] #␣
 ↪das kam schon bei solveset raus
```

[20]:
$$\left[\frac{13}{3}+\frac{4\sqrt{37}\cos\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}\right)}{3},\ -\frac{4\sqrt{37}\sin\left(\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}+\frac{\pi}{6}\right)}{3}+\frac{13}{3},\ -\frac{4\sqrt{37}\sin\left(-\frac{\operatorname{atan}\left(\frac{3\sqrt{227127}}{3305}\right)}{3}+\frac{\pi}{6}\right)}{3}+\frac{13}{3}\right]$$

[21]:
```
[p.subs(x, w).simplify() for w in wurzeln]
```

[21]:
$[0,\ 0,\ 0]$

1.2 Nichtpolynomielle Gleichungen

eine Gleichung mit abzählbar vielen Lösungen

[22]:
```
glt = Eq(sin(x), cos(x))
glt
```

[22]:
$$\sin(x)=\cos(x)$$

[23]:
```
Lsg = solveset(glt)
Lsg
```

[23]:
$$\left\{2n\pi+\frac{5\pi}{4}\,\middle|\,n\in\mathbb{Z}\right\}\cup\left\{2n\pi+\frac{\pi}{4}\,\middle|\,n\in\mathbb{Z}\right\}$$

[ ]:
```
for i, lsg in Lsg:
    display(lsg)
```

[24]:
```
for i, lsg in zip( range(7), Lsg): # die 'ersten' sieben Lösungen
    display(lsg)
    #print(glt.subs(x,lsg))
```

$$\frac{5\pi}{4}$$
$$\frac{\pi}{4}$$
$$\frac{13\pi}{4}$$
$$\frac{9\pi}{4}$$
$$-\frac{3\pi}{4}$$
$$-\frac{7\pi}{4}$$
$$\frac{21\pi}{4}$$

[25]:
```
print(Lsg)
```

```
Union(ImageSet(Lambda(_n, 2*_n*pi + 5*pi/4), Integers), ImageSet(Lambda(_n,
2*_n*pi + pi/4), Integers))
```

[26]:
```
Quadrate = ImageSet(Lambda(x, x**2), Naturals)
Quadrate
```

[26]:
$$\{x^2 \mid x \in \mathbb{N}\}$$

[27]:
```
9 in Quadrate
```

[27]:
```
True
```

[28]:
```
glt = Eq(sin(2*x), cos(x))
glt
```

[28]:
$$\sin(2x) = \cos(x)$$

[29]:
```
Lsg = solveset(glt, x)
Lsg
```

[29]:
$$\left\{2n\pi + \frac{\pi}{2} \mid n \in \mathbb{Z}\right\} \cup \left\{2n\pi + \frac{3\pi}{2} \mid n \in \mathbb{Z}\right\} \cup \left\{2n\pi + \frac{5\pi}{6} \mid n \in \mathbb{Z}\right\} \cup \left\{2n\pi + \frac{\pi}{6} \mid n \in \mathbb{Z}\right\}$$

[30]:
```
Lsg = solveset(glt, x, Interval(-5, 10))  #Lösung in einem Intervall
Lsg
```

[30]:
$$\left\{-\frac{3\pi}{2}, -\frac{7\pi}{6}, -\frac{\pi}{2}, \frac{\pi}{6}, \frac{\pi}{2}, \frac{5\pi}{6}, \frac{3\pi}{2}, \frac{13\pi}{6}, \frac{5\pi}{2}, \frac{17\pi}{6}\right\}$$

[31]:
```
glt = Eq(8*sin(x), x+1)
glt
```

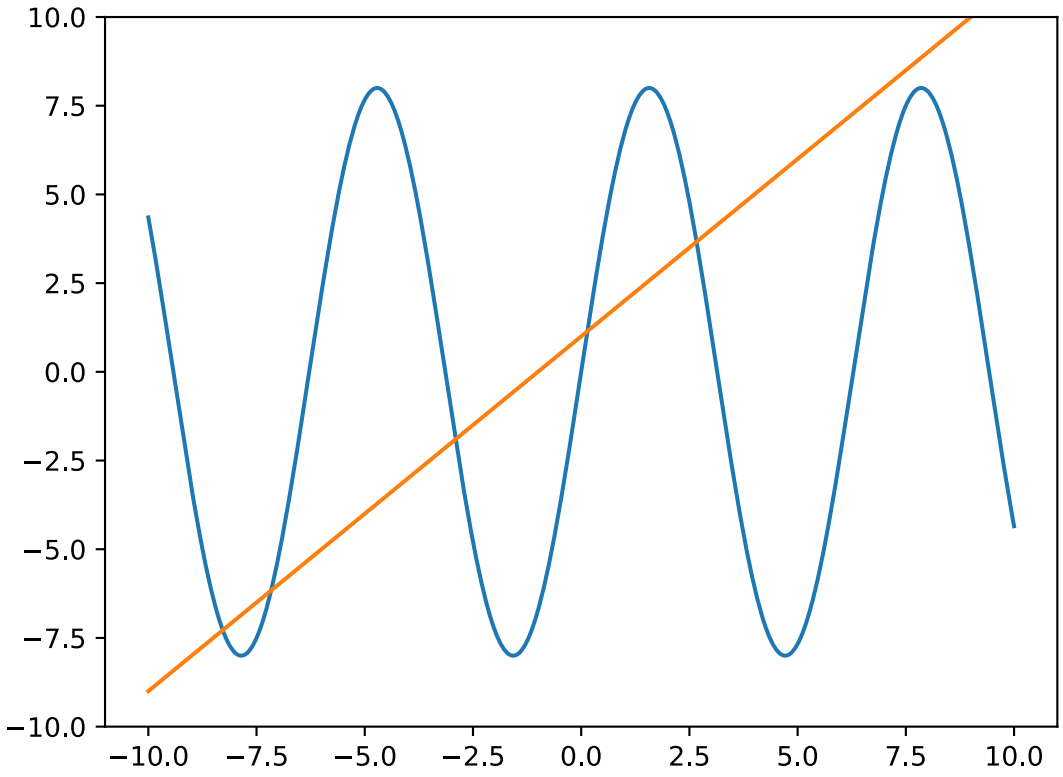
[31]:

$$8\sin(x) = x + 1$$

[32]:
```
Lsgt = solveset(glt)
Lsgt
```

[32]:
$$\{x \mid x \in \mathbb{C} \wedge -x + 8\sin(x) - 1 = 0\}$$

[33]:
```
gl = glt.lhs
gr = glt.rhs
xn = np.linspace(-10, 10, 500)
gln = lambdify(x, gl)
grn = lambdify(x, gr)
fig, ax = plt.subplots()
ax.plot(xn, gln(xn), xn, grn(xn))
ax.set_ylim((-10, 10));
```



1.2.1 noch ein paar Gleichungen zum Testen

[34]:
```
solveset(exp(x), x) # exp(x) = 0
```

[34]:
$$\emptyset$$

[35]:
```
glg = Eq(exp(x), E)
solveset(glg)
```

[35]: $\{2ni\pi + 1 \mid n \in \mathbb{Z}\}$

[36]:
```
solveset(Eq(exp(x**2), 1), x)  # exp(x**2) = 1
```

[36]: $\left\{x \mid x \in \mathbb{C} \wedge e^{x^2} - 1 = 0\right\}$

[37]:
```
solveset(Eq(exp(x**2), 1), x, domain=Reals)  # exp(x**2) = 1
```

[37]: $\{0\}$

[38]:
```
solveset(exp(x) + x, x)
```

[38]: $\{x \mid x \in \mathbb{C} \wedge x + e^{x} = 0\}$

[39]:
```
solveset(exp(x) + x, x, domain=Reals)
```

[39]: $\{x \mid x \in \mathbb{R} \wedge x + e^{x} = 0\}$

[40]:
```
solveset(x**2 + 1, x, domain=Reals)
```

[40]: $\emptyset$

[41]:
```
solveset(Eq(2**(2 - x), 3**(2 * x - 1)), x)
```

[41]: $\left\{\frac{\log(3) + 2\log(2)}{\log(2) + 2\log(3)}\right\}$

[42]:
```
solveset(log(x + 1) + log(x - 2), x)  # log(x+1)+log(x-2) = 0
```

[42]: $\left\{\frac{1}{2} - \frac{\sqrt{13}}{2}, \frac{1}{2} + \frac{\sqrt{13}}{2}\right\}$

Andere Wahlen für domain: Naturals, Naturals0, Integers, Complexes (Voreinstellung)

1.3 Lösen von Gleichungen (solve)

1.3.1 die paar Gleichungen zum Testen jetzt mit solve

sympy.solve gibt eine Liste mit Lösungen zurück

[43]:
```
p = Eq(x**5 - 20*x + 7, 0)
lsg = solve(p, x)
lsg
```

[43]: $[\mathrm{CRootOf}(x^5 - 20x + 7, 0),\ \mathrm{CRootOf}(x^5 - 20x + 7, 1),\ \mathrm{CRootOf}(x^5 - 20x + 7, 2),\ \mathrm{CRootOf}(x^5 - 20x + 7, 3),\ C$

[44]:
```
p = Eq(x**3 - 13*x**2 + 7*x + 10, 0)
lsgn = solve(p) # das sind die Lösungen wie bei roots
lsgn
```

[44]:
$$\left[\frac{13}{3} + \left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54} + \frac{\sqrt{227127}i}{18}} + \frac{148}{9\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54} + \frac{\sqrt{227127}i}{18}}}, \frac{13}{3} + \frac{148}{9\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{3305}{54} + \frac{\sqrt{227127}i}{18}}}\right.$$

[45]:
```
glt = Eq(sin(x), cos(x))
glt
```

[45]:
$$\sin(x) = \cos(x)$$

[46]:
```
solve(glt, x)
```

[46]:
$$\left[\frac{\pi}{4}\right]$$

[47]:
```
glt = Eq(sin(2*x), cos(x))
solve(glt, x)
```

[47]:
$$\left[-\frac{\pi}{2}, \frac{\pi}{6}, \frac{\pi}{2}, \frac{5\pi}{6}\right]$$

hier fehlen jede Menge Lösungen; hier ist solve schlechter als solveset

[48]:
```
solve(Eq(8*sin(x), x+1), x)
```

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
Cell In[48], line 1
----> 1 solve(Eq(8*sin(x), x+1), x)

File /local/home/schaedle/miniconda3/envs/compla24/lib/python3.12/site-packages
 ↪sympy/solvers/solvers.py:1168, in solve(f, *symbols, **flags)
   1166         solution = _solve_undetermined(f[0], symbols, flags)
   1167     if not solution:
-> 1168         solution = _solve(f[0], *symbols, **flags)
   1169 else:
   1170     linear, solution = _solve_system(f, symbols, **flags)

File /local/home/schaedle/miniconda3/envs/compla24/lib/python3.12/site-packages
 ↪sympy/solvers/solvers.py:1722, in _solve(f, *symbols, **flags)
   1719 # ----------- end of fallback ----------------------------
   1721 if result is False:
-> 1722     raise NotImplementedError('\n'.join([msg, not_impl_msg % f]))
   1724 result = _remove_duplicate_solutions(result)
   1726 if flags.get('simplify', True):
```

```
NotImplementedError: multiple generators [x, sin(x)]
No algorithms are implemented to solve equation (-x - 1) + 8*sin(x)
```

solve wirft hier einen NotImplemented error; das ist ärgerlich

[49]:
```
solve(Eq(exp(x), 0), x)
```

[49]:
[]

[50]:
```
glg = Eq(exp(x), E)
solve(glg, dict=True)
```

[50]:
$[\{x : 1\}]$

[51]:
```
solve(Eq(exp(x**2), 1), x)
```

[51]:
[0]

Die Lösung von $e^{x^2} = 1$ ist $=0$ und die findet solve. Hier ist solve besser als solveset

[52]:
```
solve(exp(x) + x, x)
```

[52]:
$[-W(1)]$

Hier ist solve besser als solveset

[53]:
```
xp = symbols('x', positive=True)
xr = symbols('x', real=True)
solve(xr**2 + 1, xr)
```

[53]:
[]

[54]:
```
solve(Eq(2**(2-x), 3**(2*x-1)), x)
```

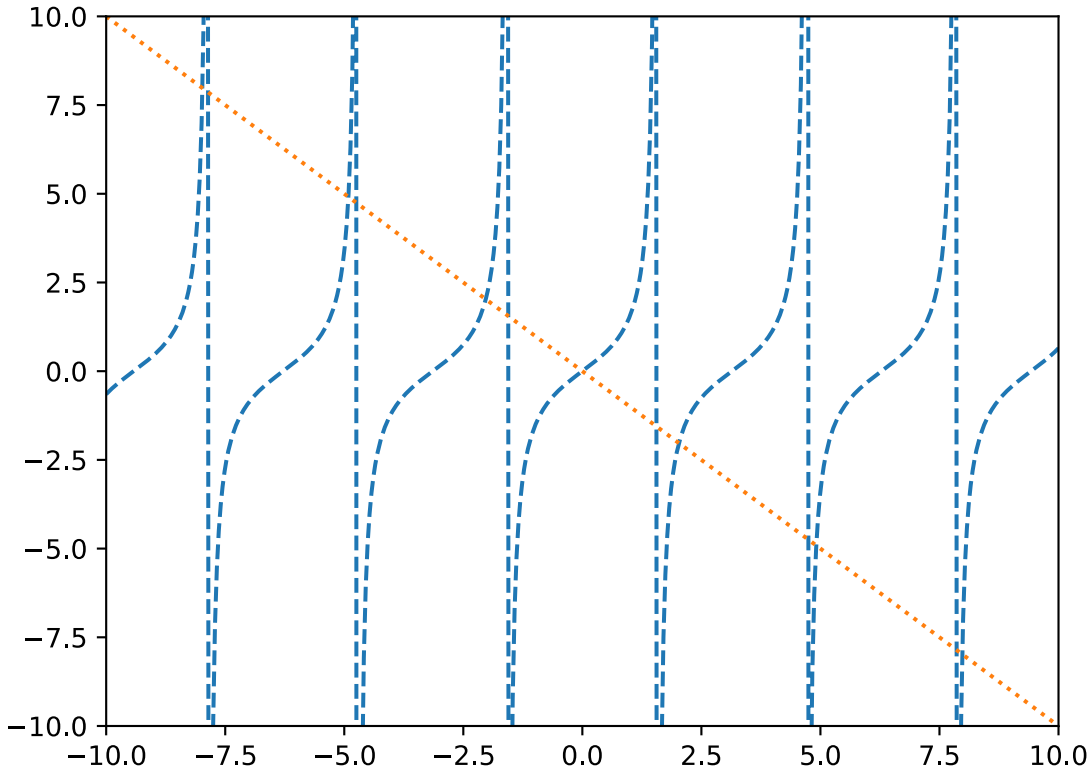
[54]:
$$\left[\frac{\log(12)}{\log(18)}\right]$$

1.4 Numerische Lösung von Gleichungen

[55]:
```
def glp(lb, ub, gl, nn=500):
    ''' Zeichne Gleichung gl in (lb, ub)'''
    fig = plt.figure()
    ax = fig.gca()
    xn = np.linspace(lb, ub, nn)
    ax.plot(xn, lambdify(x, gl.lhs)(xn), '--')
    ax.plot(xn, lambdify(x, gl.rhs)(xn), ':')
    return fig, ax
```

[56]:
```
gl = Eq(tan(x), -x)
fig, ax = glp(-10, 10, gl)
```

```
ax.axis([-10, 10, -10, 10]);
```



[57]:
```
solveset(gl)
```

[57]: $\{x \mid x \in \mathbb{C} \wedge x + \tan(x) = 0\}$

[58]:
```
nsolve(gl, x, 4, dict=True)
```

[58]: $[\{x : 2.02875783811043\}]$

[59]:
```
nsolve(gl, x, (np.pi/2, 3*np.pi/2), solver='bisect')
```

[59]: 2.02875783811043

[60]:
```
nsolve(gl, x, (3*np.pi/2, 5*np.pi/2), solver='bisect')
```

[60]: 4.91318043943488

[61]:
```
[((2*i-1)/2+.001, (2*i+1)/2-.001) for i in range(-1,2)]
```

[61]: $[(-1.499,\ -0.501),\ (-0.499,\ 0.499),\ (0.501,\ 1.499)]$

[62]:
```
ps = np.array([
    nsolve(gl, x,
           (np.pi * (2 * i - 1) / 2 + .001, np.pi * (2 * i + 1) / 2 - .001),
           solver='bisect') for i in range(-3, 4)
])
ps
```

[62]:
```
array([-7.97866571241324, -4.91318043943488, -2.02875783811043, 0,
       2.02875783811043, 4.91318043943488, 7.97866571241324], dtype=object)
```

[63]:
```
ax.plot(ps,-ps,'ro'); # das geht nur in notebook oder qt Modus
```

1.5 Gleichungen mit Parameter

[64]:
```
a, y = symbols('a y')
```

Wir betrachten die Gleichung

$$x^2 = a$$

[65]:
```
solve(Eq(x**2, a), x)
```

[65]:
$$[-\sqrt{a},\ \sqrt{a}]$$

[66]:
```
solveset(Eq(x**2, a), x)
```

[66]:
$$\{-\sqrt{a}, \sqrt{a}\}$$

1.5.1 Lambert W

[67]:
```
solve(Eq(exp(-a * x), 3 * x**a), x)
```

[67]:
$$\left[W\left(e^{-\frac{\log(3)}{a}}\right)\right]$$

[68]:
```
solveset(Eq(exp(-a * x), 3 * x**a), x)
```

[68]:
$$\left\{x \;\middle|\; x \in \mathbb{C} \wedge x^a e^{ax} - \frac{1}{3} = 0\right\}$$

Die Lambert W-Funktion ist die Umkehrfunktion von

$$x \mapsto x\mathrm{e}^x$$

[69]:
```
sol = solve(Eq(x * exp(x), y), x)
sol
```

[69]:
$$[W\left(y\right)]$$

[70]:
```
sol[0].subs(y, 1).n()
```

[70]:
0.567143290409784

[71]:
```
soln = lambdify(y, sol[0])
soln
```

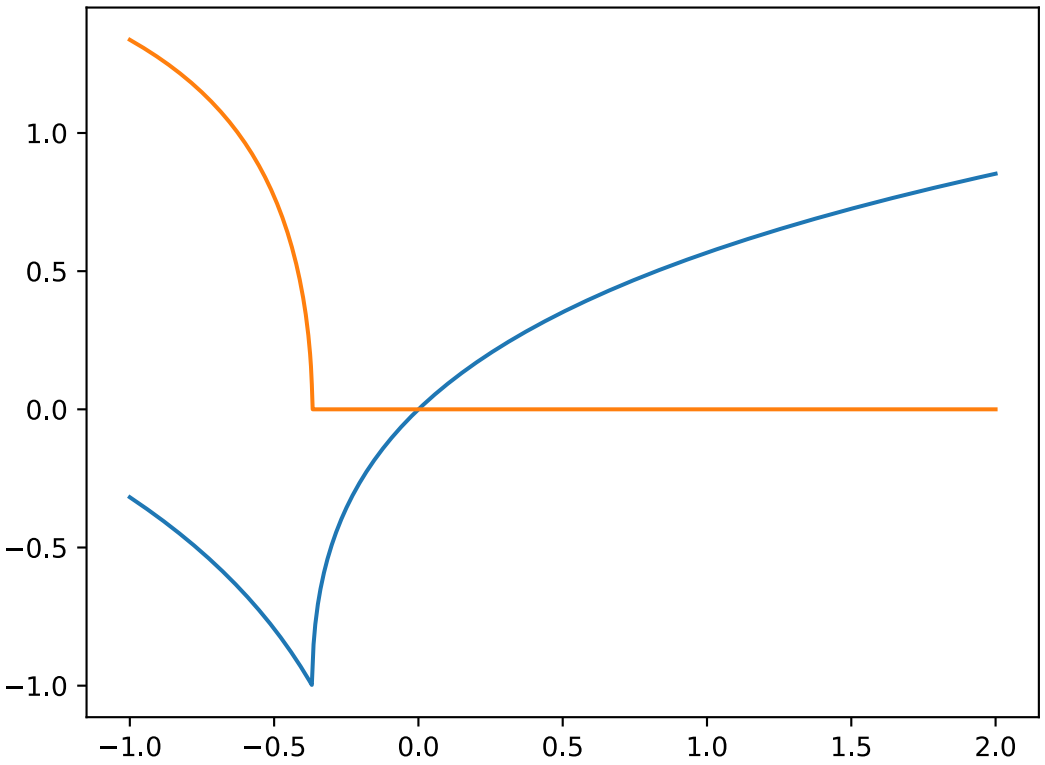
[71]:
```
<function _lambdifygenerated(y)>
```

[72]:
```
soln(1)
```

[72]:
```
np.complex128(0.5671432904097838+0j)
```

Allgemeine Information: https://en.wikipedia.org/wiki/Lambert_W_function

[73]:
```
xn = np.linspace(-1, 2, 1000)
yn = soln(xn)
fig, ax = plt.subplots()
ax.plot(xn, yn.real, xn, yn.imag);
```



1.5.2 Polynomielle Gleichungen

[74]:
```
#from sympy import *
x, y = symbols('x y')
gl = Eq(x**2 + y**2, 1)
sol1 = solve(gl)
sol1
```

[74]: $\left[\left\{x : -\sqrt{1-y^2}\right\}, \left\{x : \sqrt{1-y^2}\right\}\right]$

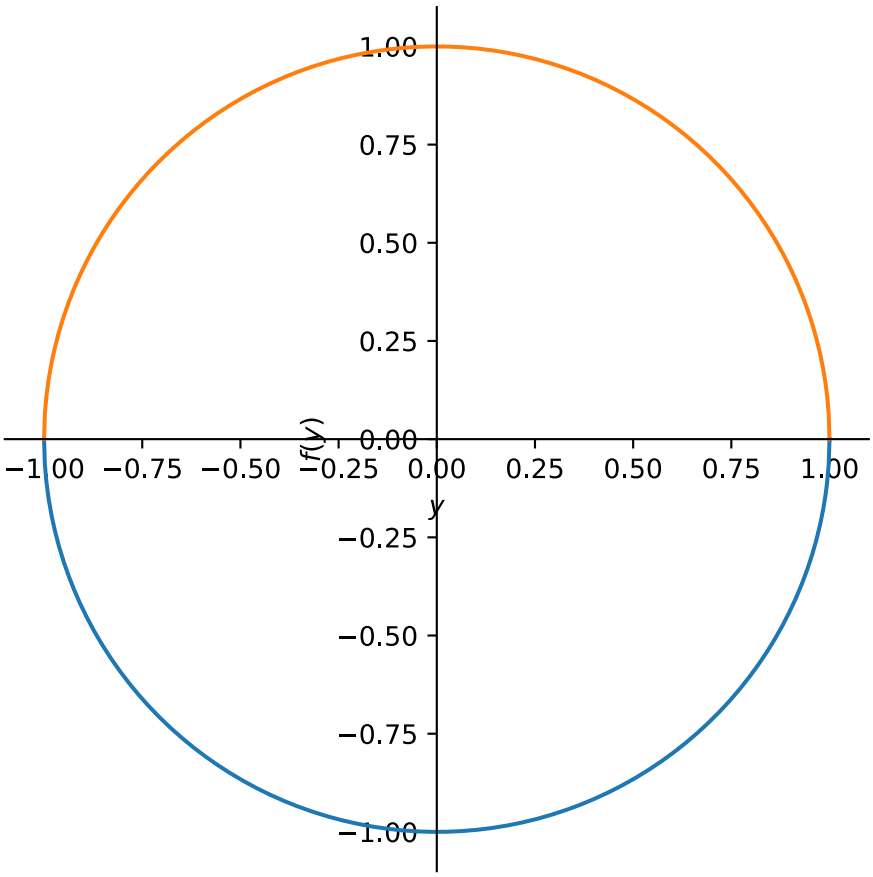
[75]:
```
sol2 = solveset(gl, x)
sol2
```

[75]: $\left\{-\sqrt{-(y-1)(y+1)}, \sqrt{-(y-1)(y+1)}\right\}$

[76]:
```
sol1
sol1[0][x]
```

[76]: $-\sqrt{1-y^2}$

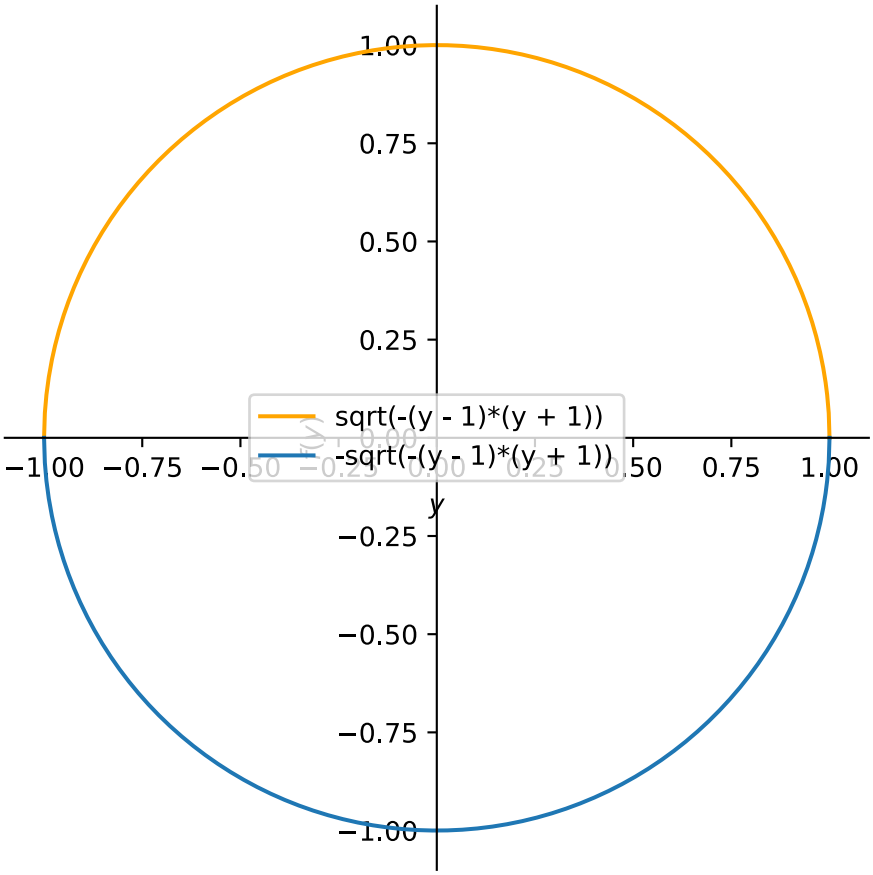
[77]:
```
pl = plot(sol1[0][x], sol1[1][x], (y, -1, 1), aspect_ratio=(1, 1))
```



[78]:
```
sol2
```

[78]: $\left\{-\sqrt{-(y-1)(y+1)}, \sqrt{-(y-1)(y+1)}\right\}$

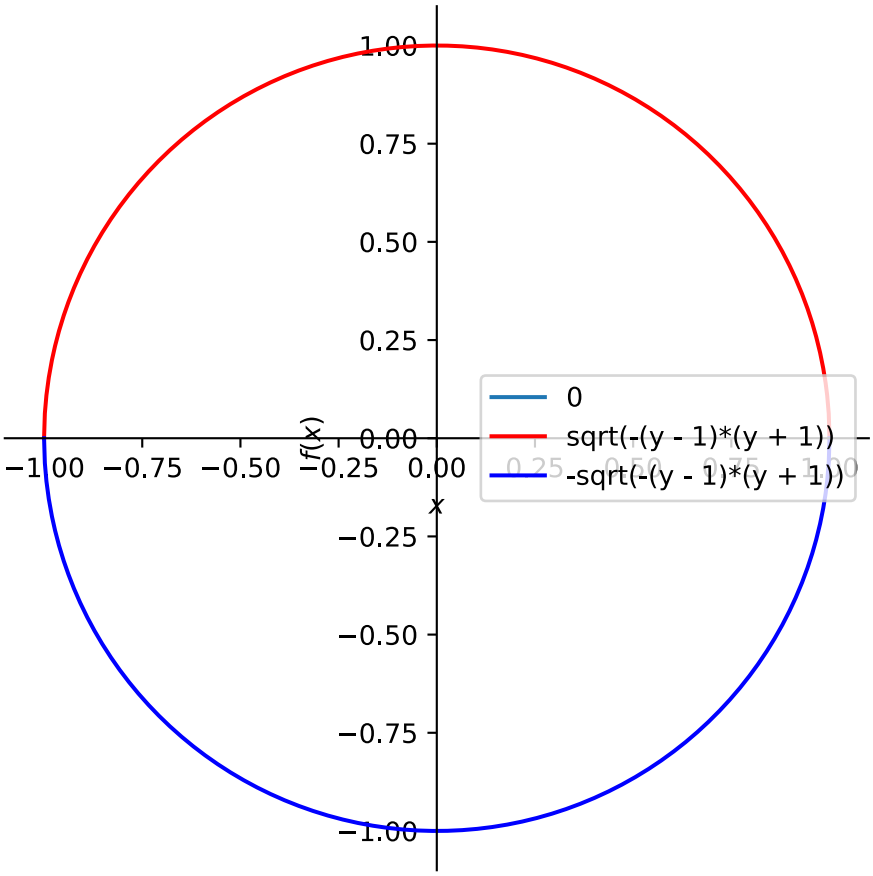
[79]:
```
pl1 = plot(list(sol2)[0], (y, -1, 1), aspect_ratio=(1, 1), show=False)
pl2 = plot(list(sol2)[1], (y, -1, 1), aspect_ratio=(1, 1), show=False)
pl1.extend(pl2)
pl1[0].line_color = 'orange'
pl1.legend = True
pl1.show()
```



[80]:
```
pl = plot(0, (x, 0, 0), show=False, aspect_ratio=(1, 1))
for sol in sol2:
    pl.extend(plot(sol, (y, -1, 1), show=False))

pl[1].line_color = 'red'
pl[2].line_color = 'blue'
pl.legend = True
```

```
pl.show()
```



[81]:
```
gl = Eq(x * y, 0)
sol1 = solve(gl)
sol1
```

[81]: $[\{x : 0\},\ \{y : 0\}]$

[82]:
```
gl = Eq((x**2 + y**2)**2 + 3*x**2*y - y**3, 0)
sols = solve(gl)
gl
```

[82]: $3x^2y - y^3 + \left(x^2 + y^2\right)^2 = 0$

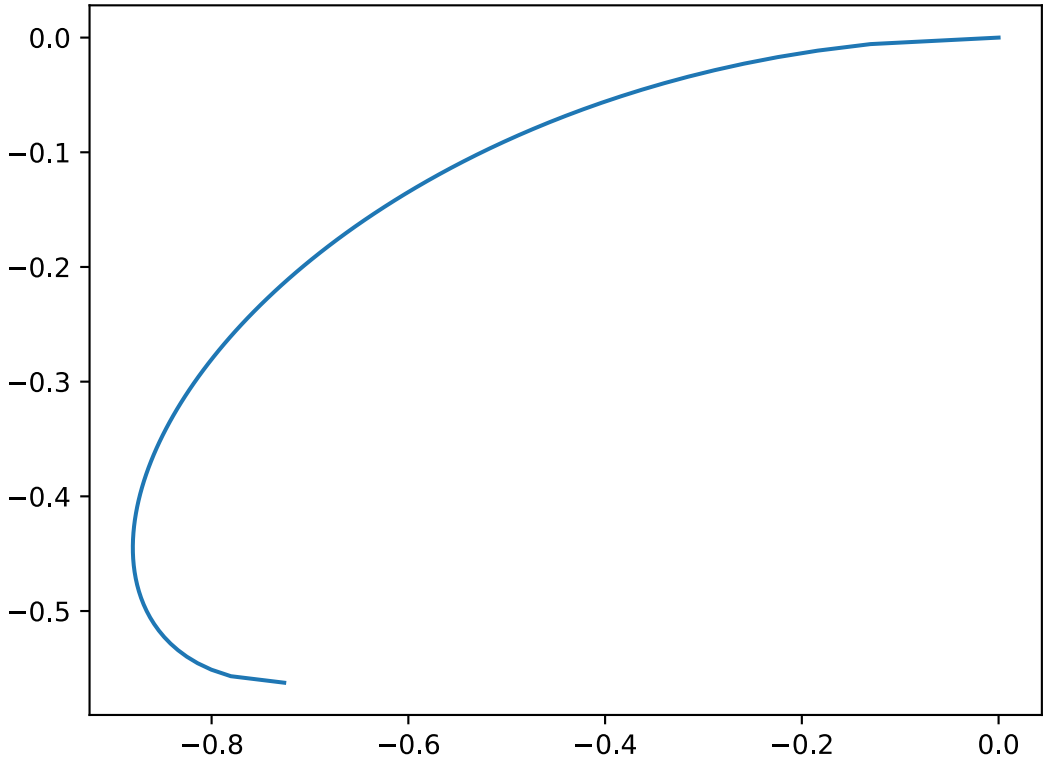
[83]:
```
sols
```

[83]: $\left[\left\{x : -\sqrt{-y^2 - \frac{y\sqrt{16y+9}}{2} - \frac{3y}{2}}\right\},\ \left\{x : \sqrt{-y^2 - \frac{y\sqrt{16y+9}}{2} - \frac{3y}{2}}\right\},\ \left\{x : -\sqrt{-y^2 + \frac{y\sqrt{16y+9}}{2} - \frac{3y}{2}}\right\},\ \{$

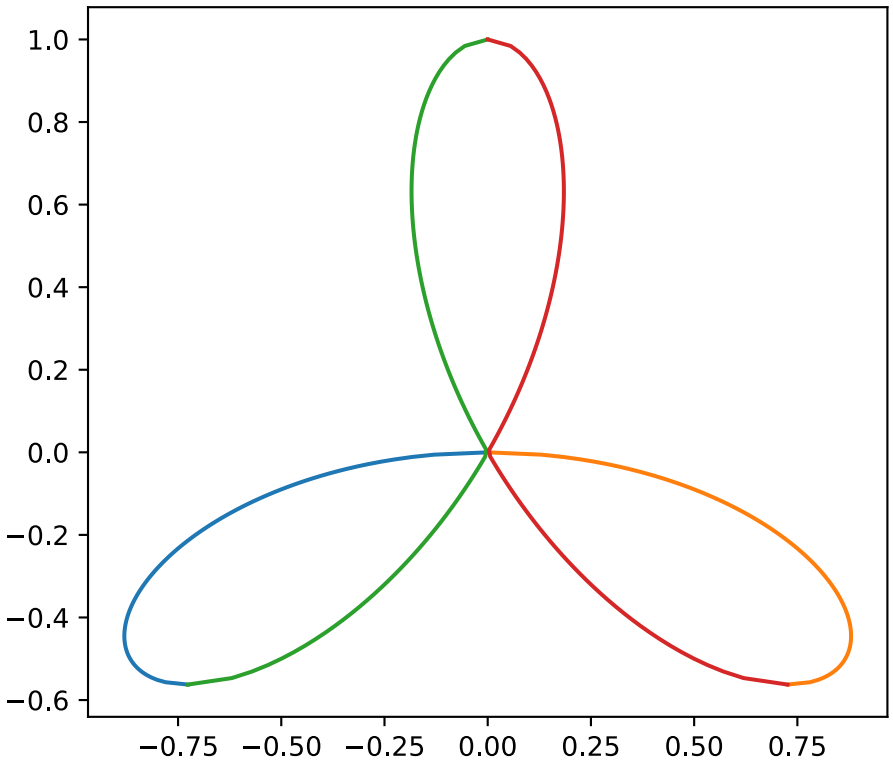
[84]:
```
import numpy as np
import matplotlib.pyplot as plt

yn = np.linspace(-9 / 16, 0, 100)
fig = plt.figure()
ax = fig.gca()
ax.plot(lambdify(y, sols[0][x])(yn), yn)
```

[84]:
```
[<matplotlib.lines.Line2D at 0x7f0bace8c5f0>]
```



[85]:
```
fig = plt.figure()
ax = fig.gca()
for i, sol in enumerate(sols):
    if i in [2, 3]:
        yn = np.linspace(-9 / 16, 1, 100)
    else:
        yn = np.linspace(-9 / 16, 0, 100)
    ax.plot(lambdify(y, sol[x])(yn), yn)
ax.set_aspect('equal')
```



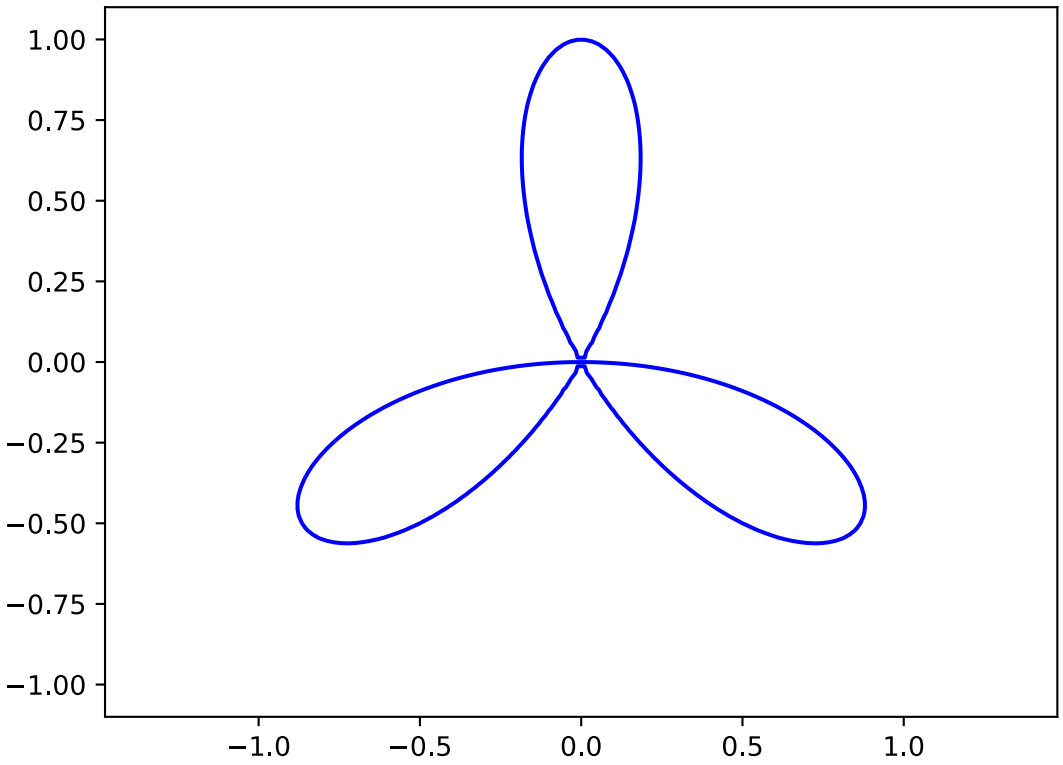
einfacher

[86]:
```
gl
```

[86]: $3x^2y - y^3 + \left(x^2 + y^2\right)^2 = 0$

[87]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-1.1, 1.1, 100)
X, Y = np.meshgrid(xn, xn)
ax.contour(X, Y, lambdify((x, y), gl.lhs)(X, Y), [0], colors='blue')
ax.axis('equal')
```

[87]: $(-1.1,\ 1.1,\ -1.1,\ 1.1)$



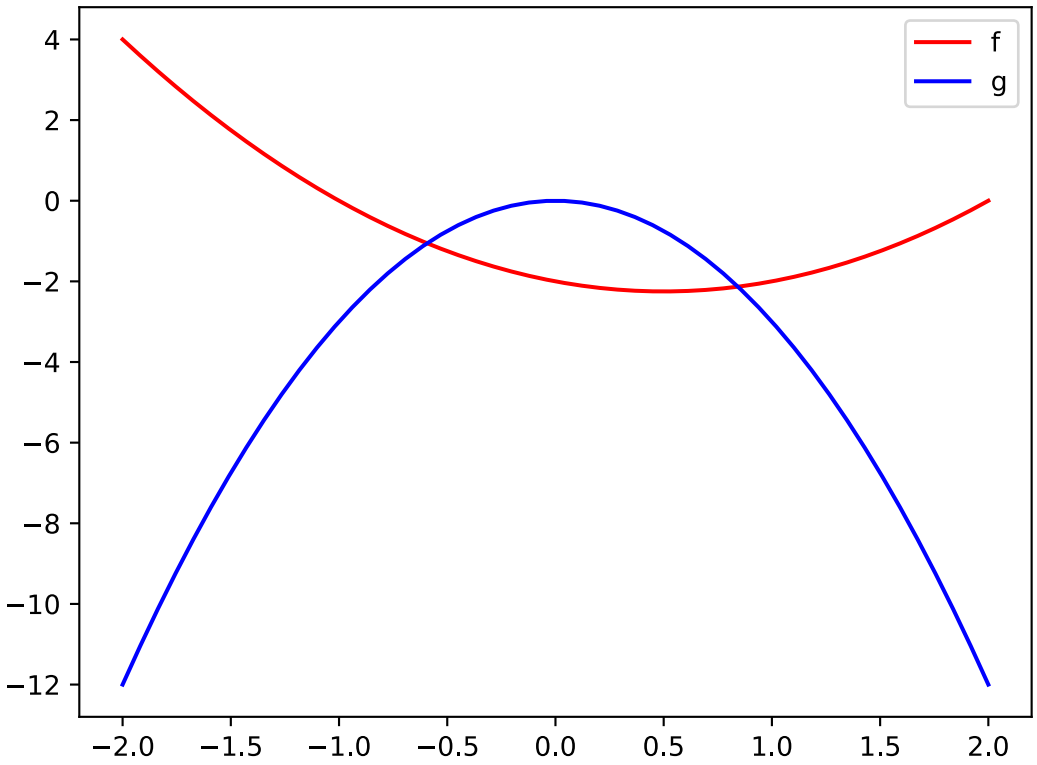
1.6 Ungleichungen

[88]:
```
f = x**2 - x - 2
g = -3 * x**2
sol = solve(f < g)
sol
```

[88]:
$$x < \frac{1}{8} + \frac{\sqrt{33}}{8} \wedge \frac{1}{8} - \frac{\sqrt{33}}{8} < x$$

[89]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-2, 2)
ax.plot(xn, lambdify(x, f)(xn), 'r', label='f')
ax.plot(xn, lambdify(x, g)(xn), 'b', label='g')
ax.legend()
```

[89]:
```
<matplotlib.legend.Legend at 0x7f0badaa8e00>
```



[90]:
```
sol.subs(x, 0)
```

[90]: True

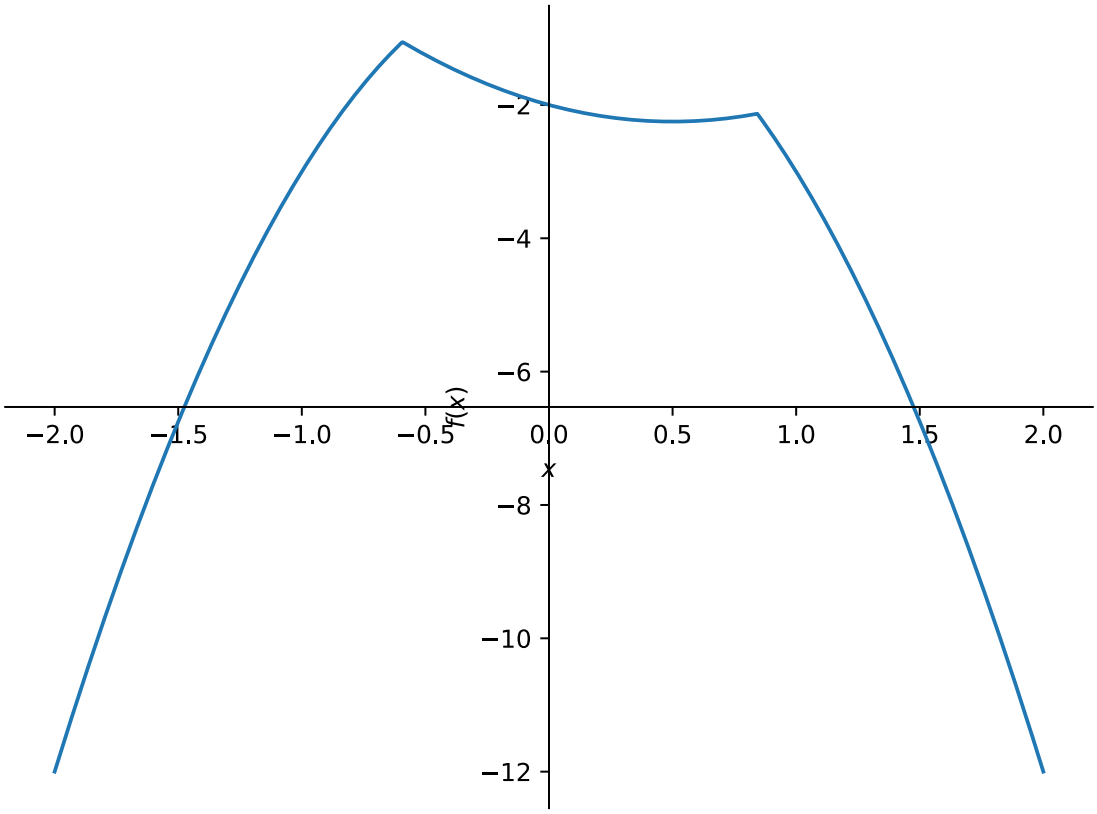
[91]:
```
solI = sol.as_set()  # Lösungsintervall
solI
```

[91]: $\left(\frac{1}{8} - \frac{\sqrt{33}}{8}, \frac{1}{8} + \frac{\sqrt{33}}{8}\right)$

[92]:
```
type(solI)
```

[92]: `sympy.sets.sets.Interval`

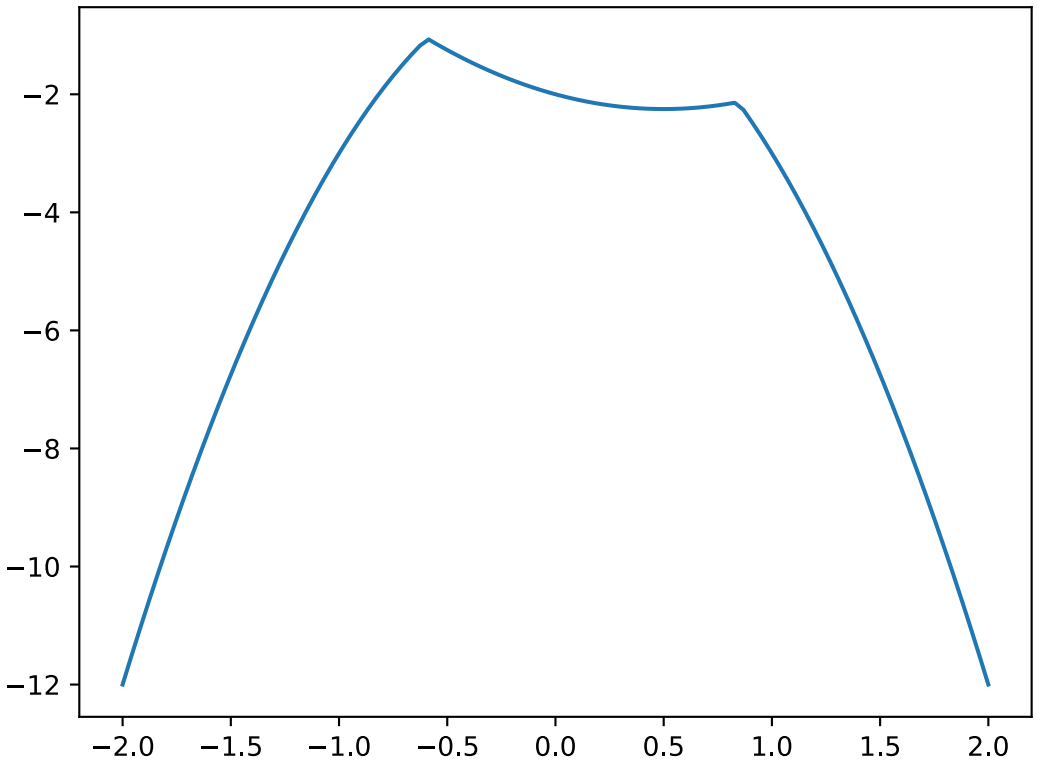
[93]:
```
p1 = Piecewise((f, sol), (g, True))
plot(p1, (x, -2, 2))
```



[93]:
```
<sympy.plotting.backends.matplotlibbackend.matplotlib.MatplotlibBackend at
0x7f0bb6498d10>
```

[94]:
```
pn = lambdify(x, p1)
xn = np.linspace(-2, 2, 100)
plt.figure()
plt.plot(xn, pn(xn))
```

[94]:
```
[<matplotlib.lines.Line2D at 0x7f0bacc53c80>]
```



[95]:
```
sols = solveset(f > g)
sols
```

[95]: $\{x \mid x \in \mathbb{C} \wedge x^2 - x - 2 > -3x^2\}$

[96]:
```
sols = solveset(f > g, domain=Reals)
sols  # Lösungsmenge, Vereinigung von zwei Intervallen
```

[96]: $\left(-\infty, \frac{1}{8} - \frac{\sqrt{33}}{8}\right) \cup \left(\frac{1}{8} + \frac{\sqrt{33}}{8}, \infty\right)$

[97]:
```
f = sin(x)
g = cos(x)
sols = solveset(f > g, x, domain=Reals)
sols  # da fehlt was
```

[97]: $\left(\frac{\pi}{4}, \frac{5\pi}{4}\right)$

[98]:
```
sol = solve(f > g, x)
sol  # und hier auch
```

[98]: $\frac{\pi}{4} < x \wedge x < \frac{5\pi}{4}$

[ ]: