lektion8

January 23, 2020

Table of Contents

1 Lektion 8

1.1 Gleichungen mit Parameter

[1]:
```
from sympy import *
init_printing()
import matplotlib.pyplot as plt
import numpy as np
#%matplotlib notebook
```

```
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
x, y, z, a, b, c, d = symbols('x y z a b c d')
```

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

[2]: $\left[-\sqrt{a},\ \sqrt{a}\right]$

1.1.1 Lambert W

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

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

[4]:
```
solveset(Eq(exp(-a*x), 3*x**a), x)
```

[4]: $\left\{x \mid x \in \mathbb{C} \wedge -3x^{a} + e^{-ax} = 0\right\}$

[5]:
```
sol = solve(Eq(x*exp(x), y), x)
sol
```

[5]: $[W(y)]$

Die Lambert W-Funktion ist die Umkehrfunktion von x*exp(x)

[6]:
```
sol[0].subs(y, 1).n()
```

[6]: 0.567143290409784

[7]:
```
soln = lambdify(y, sol[0])
soln
```

[7]: <function _lambdifygenerated(y)>

[8]:
```
soln(1)
```

```
                                                                            ␣
↪---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-8-58123f41105c> in <module>
----> 1 soln(1)


<lambdifygenerated-1> in _lambdifygenerated(y)
  1 def _lambdifygenerated(y):
```

```
----> 2     return (LambertW(y))

    NameError: name 'LambertW' is not defined
```

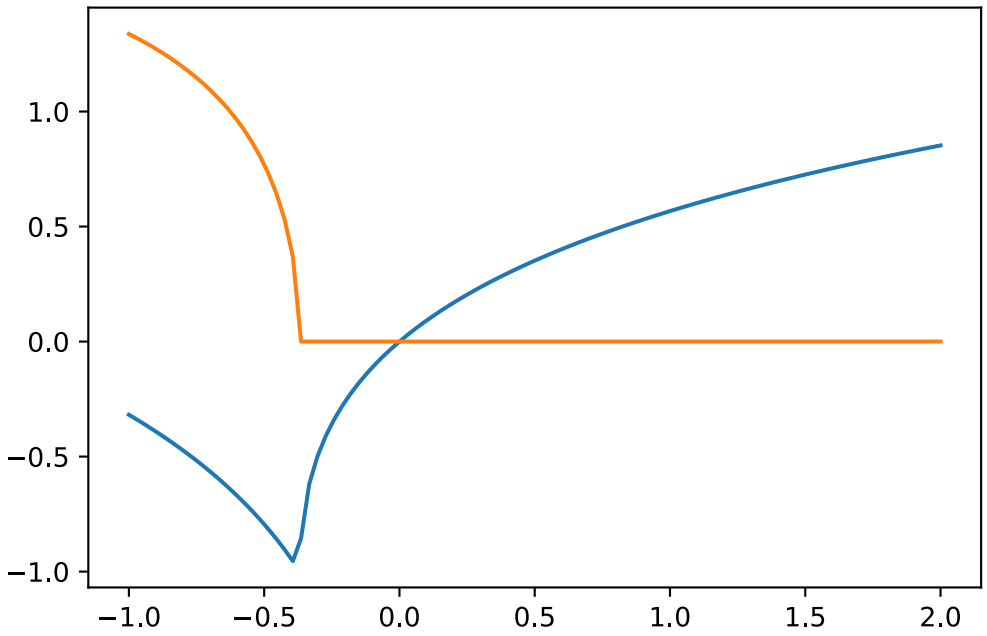
Scipy: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.lambertw.html#scipy.

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

[9]:
```
from scipy.special.lambertw import lambertw
soln = lambdify(y, sol[0], modules={'LambertW' : lambertw})
soln(1)
```

[9]:
```
(0.5671432904097838+0j)
```

[10]:
```
xn = np.linspace(-1, 2, 100)
yn = soln(xn)
plt.plot(xn, yn.real, xn, yn.imag);
```



1.1.2 Polynomielle Gleichungen

[11]:
```
gl = Eq(x**2+y**2, 1)
sol1 = solve(gl)
sol1
```

[11]:

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

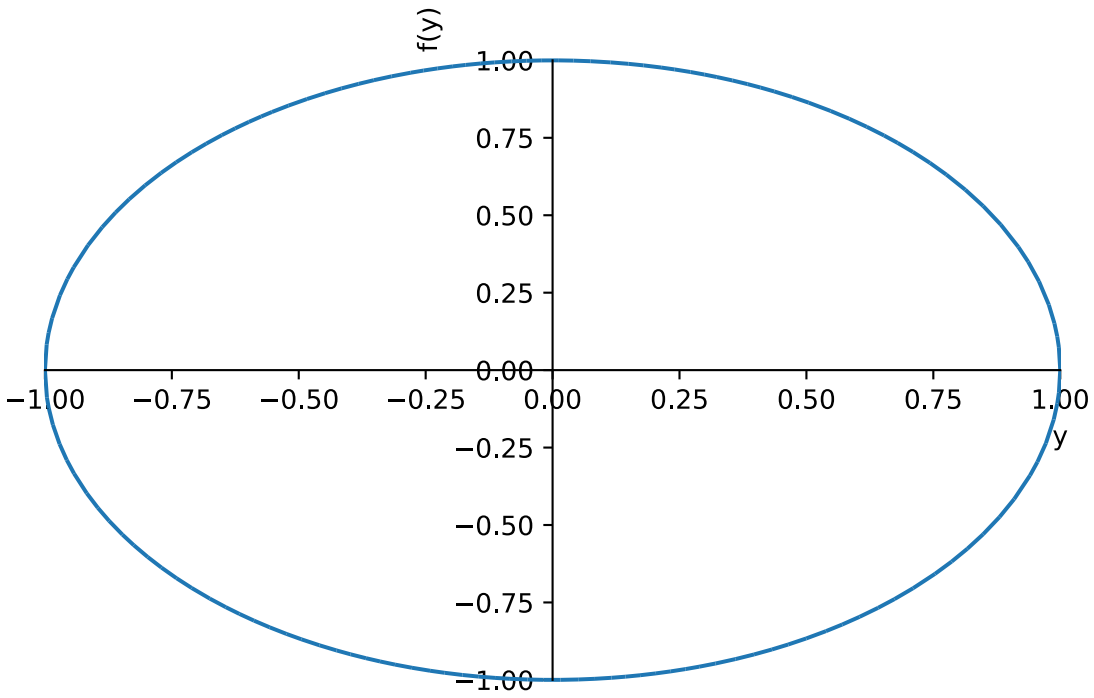
[12]:
```
sol2 = solveset(gl, x)
sol2
```

[12]:
$$\left\{-\sqrt{-(y-1)(y+1)}, \sqrt{-(y-1)(y+1)}\right\}$$

[13]:
```
sol1
```

[13]:
$$\left[\left\{x : -\sqrt{1-y^2}\right\},\ \left\{x : \sqrt{1-y^2}\right\}\right]$$

[14]:
```
pl = plot(sol1[0][x], sol1[1][x], (y, -1, 1))
pl._backend.ax.set_aspect('equal')
```




```
␣
↪---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-14-26c139bc0d12> in <module>
  1 pl = plot(sol1[0][x], sol1[1][x], (y, -1, 1))
----> 2 pl._backend.ax.set_aspect('equal')
```

```
AttributeError: 'list' object has no attribute 'set_aspect'
```

[ ]:
```
pl1 = plot(sol1[0][x], (y, -1, 1), show = False)
pl2 = plot(sol1[1][x], (y, -1, 1), show = False)
pl1.extend(pl2)
pl1[1].line_color='red'
pl1.legend = True
pl1.show()
pl1._backend.ax.set_aspect('equal')
```

[ ]:
```
sol2
```

[ ]:
```
pl1 = plot(sol2[0], (y, -1, 1), show = False)
pl2 = plot(sol2[1], (y, -1, 1), show = False)
pl1.extend(pl2)
pl1[1].line_color='red'
pl1.legend = True
pl1.show()
pl1._backend.ax.set_aspect('equal')
```

[ ]:
```
pl = plot(0,(y, -1, 1), show = False)
for sol in sol2:
    pl.extend(plot(sol, (y, -1, 1), show = False))
pl[1].line_color='red'
pl.legend = True
pl.show()
pl._backend.ax.set_aspect('equal')
```

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

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

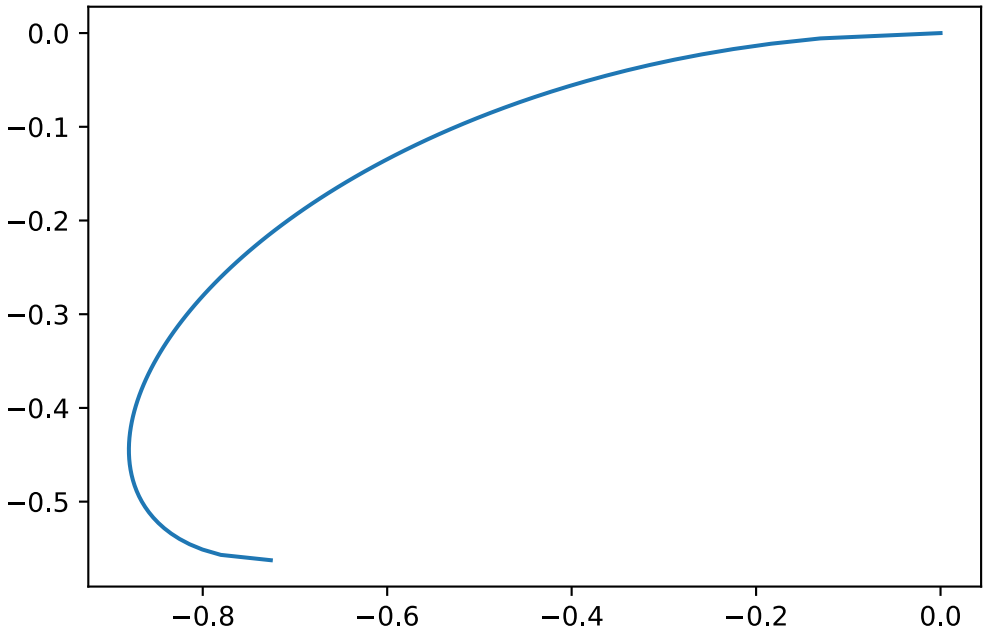
[15]:
```
gl = Eq((x**2 + y**2)**2 + 3*x**2*y - y**3, 0)
sols = solve(gl)
sols
gl
```

[15]:
$$3x^2y - y^3 + \left(x^2 + y^2\right)^2 = 0$$

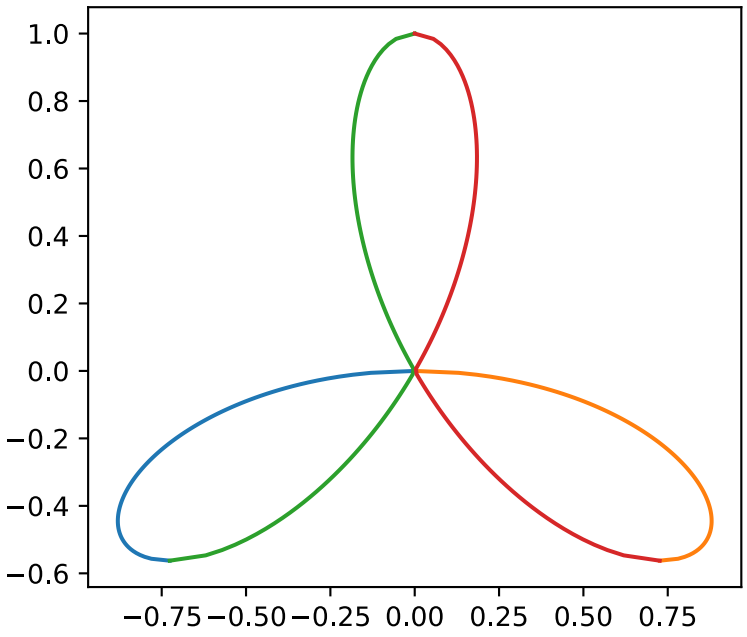
[16]:
```
yn = np.linspace(-9/16, 0, 100)
fig = plt.figure()
```

```
ax = fig.gca()
ax.plot(lambdify(y, sols[0][x])(yn), yn)
```

[16]: [<matplotlib.lines.Line2D at 0x7f8432480490>]



[17]:
```
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')
```

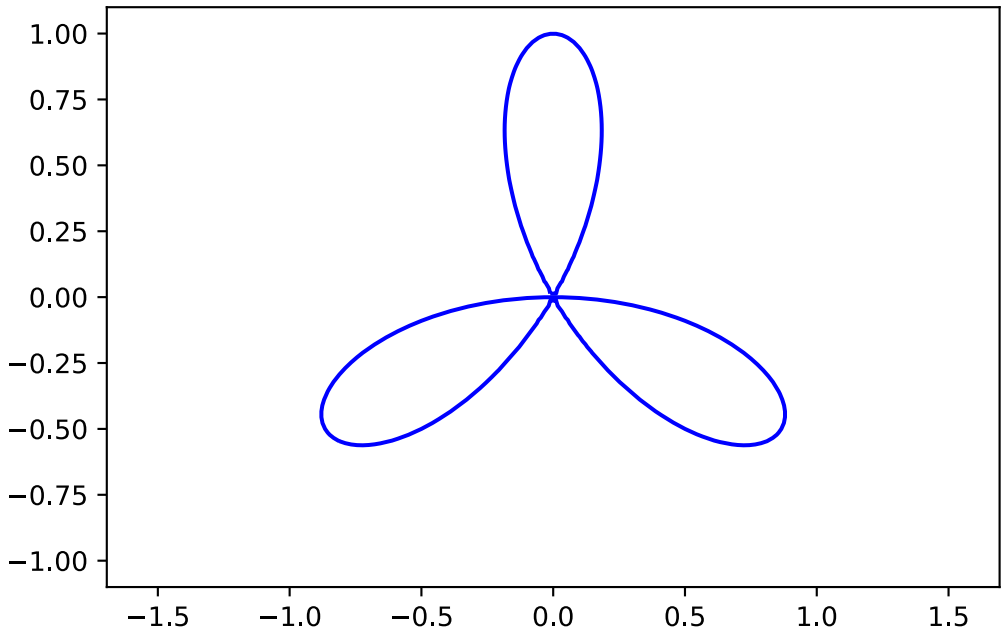


einfacher

[18]:
```
gl
```

[18]: $3x^2y - y^3 + \left(x^2 + y^2\right)^2 = 0$

[19]:
```
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');
```

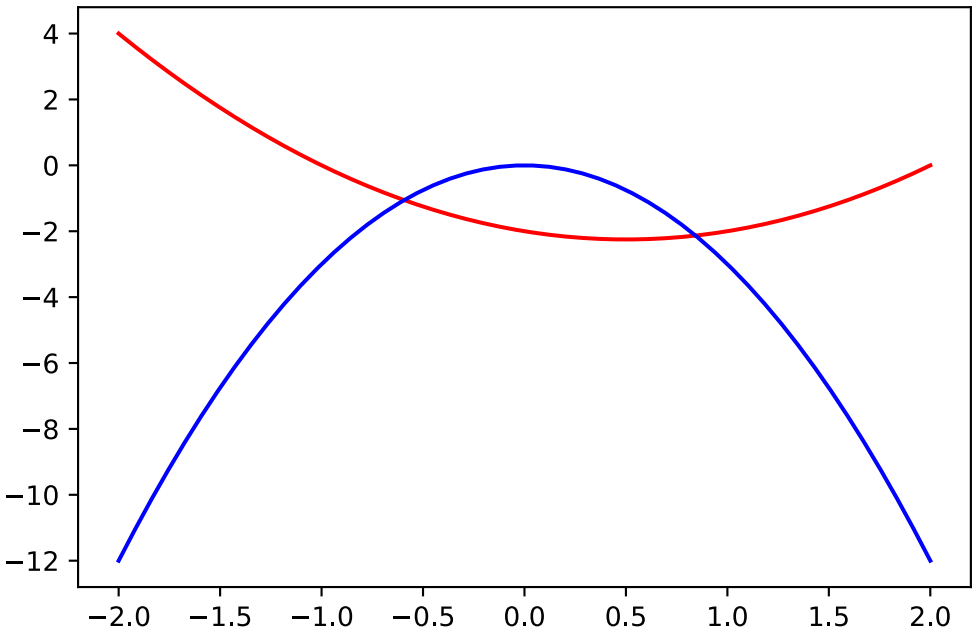


1.2 Ungleichungen

[20]:
```
f = x**2 - x -2
g = -3*x**2
sol = solve(f<g)
sol
```

[20]:
$$x < \frac{1}{8} + \frac{\sqrt{33}}{8} \wedge \frac{1}{8} - \frac{\sqrt{33}}{8} < x$$

[21]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-2,2)
ax.plot(xn, lambdify(x, f)(xn), 'r')
ax.plot(xn, lambdify(x, g)(xn), 'b');
```



[22]:
```
sol.subs(x, 1)
```

[22]:
```
False
```

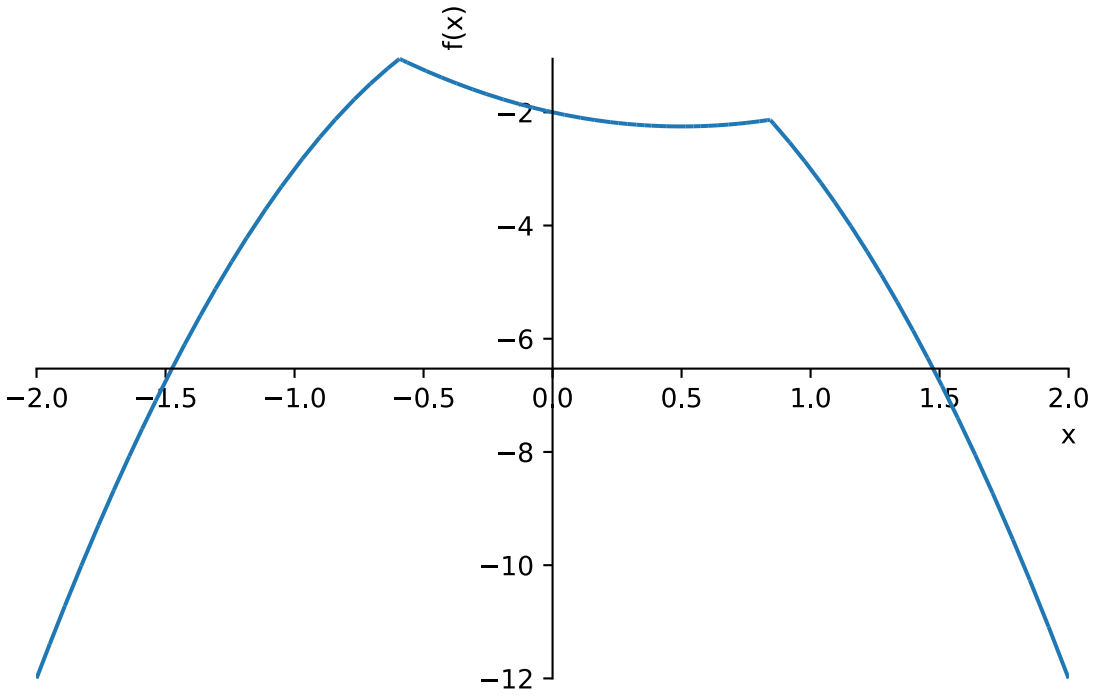
[23]:
```
solI = sol.as_set()  # Lösungsintervall
solI
```

[23]:
$$\left(\frac{1}{8} - \frac{\sqrt{33}}{8}, \frac{1}{8} + \frac{\sqrt{33}}{8}\right)$$

[24]:
```
type(solI)
```

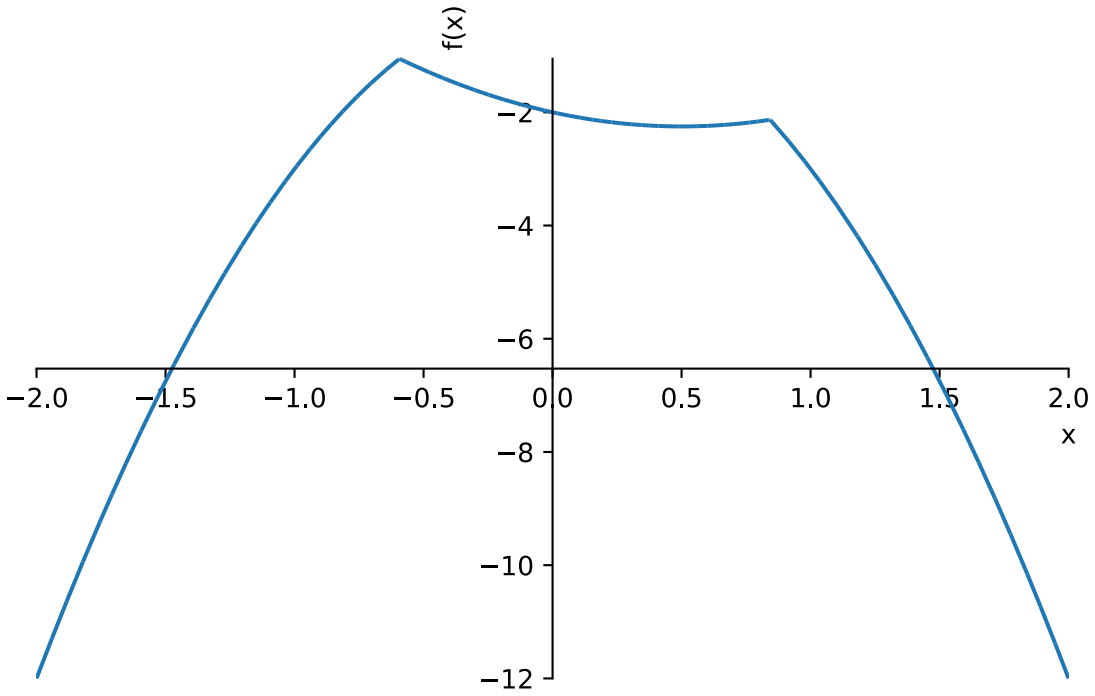
[24]:
```
sympy.sets.sets.Interval
```

[25]:
```
p1 = Piecewise((f, sol), (g, True))
plot(p1, (x, -2, 2))
```



[25]: <sympy.plotting.plot.Plot at 0x7f8432390610>

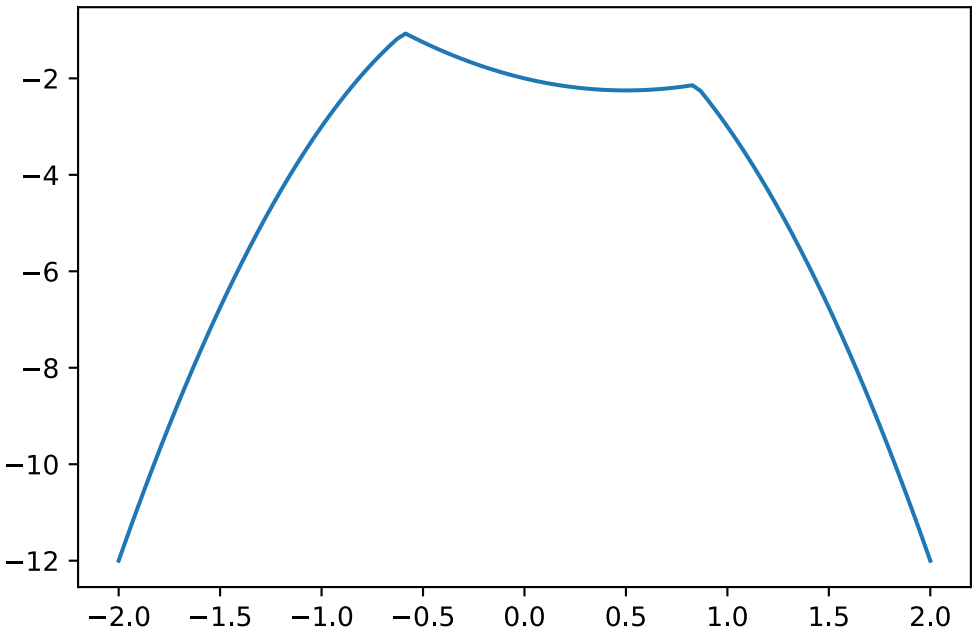
[26]:
```
p2 = Piecewise((f, sol.as_set().contains(x)), (g, True))
plot(p2, (x, -2, 2))
```



[26]: <sympy.plotting.plot.Plot at 0x7f843221f710>

[27]:
```
pn = lambdify(x, p1)
xn = np.linspace(-2, 2, 100)
plt.figure()
plt.plot(xn, pn(xn))
```

[27]: [<matplotlib.lines.Line2D at 0x7f843211b210>]



[28]:
```
sols = solveset(f>g)
```

```
␣
↪---------------------------------------------------------------------------

NotImplementedError                       Traceback (most recent call last)

<ipython-input-28-3c2288973943> in <module>
----> 1 sols = solveset(f>g)


/local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/
↪solveset.py in solveset(f, symbol, domain)
   2009     f = piecewise_fold(f)
   2010
-> 2011     return _solveset(f, symbol, domain, _check=True)
   2012
   2013


/local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/
↪solveset.py in _solveset(f, symbol, domain, _check)
    931                 Inequalities in the complex domain are
    932                 not supported. Try the real domain by
--> 933                 setting domain=S.Reals'''))
```

```
934         try:
935             result = solve_univariate_inequality(


    NotImplementedError:
Inequalities in the complex domain are not supported. Try the real
domain by setting domain=S.Reals
```

[32]:
```
sols = solveset(f>g, domain = S.Reals)
sols  # Lösungsmenge, Vereinigung von zwei Intervallen
```

[32]:
$$\left(-\infty, \frac{1}{8} - \frac{\sqrt{33}}{8}\right) \cup \left(\frac{1}{8} + \frac{\sqrt{33}}{8}, \infty\right)$$

[33]:
```
f = sin(x)
g = cos(x)
sols = solveset(f>g, x, domain = S.Reals)
sols  # da fehlt was
```

[33]:
$$\left(\frac{\pi}{4}, \frac{5\pi}{4}\right)$$

[34]:
```
sol = solve(sin(x)>cos(x), x)
sol
```

[34]:
$$\frac{\pi}{4} < x \wedge x < \frac{5\pi}{4}$$

1.3 Gleichungssysteme

[35]:
```
x = symbols('x:2')
x
```

[35]:
$$(x_0,\ x_1)$$

1.3.1 Lineare Gleichungssysteme

[36]:
```
glL = [Eq(x[0]+x[1], a), Eq(2*x[0]-b*x[1], 3)]
glL
```

[36]:
$$[x_0 + x_1 = a,\ -bx_1 + 2x_0 = 3]$$

[37]:
```
glT = (Eq(x[0]+x[1], a), Eq(2*x[0]-b*x[1], 3))
glS = {Eq(x[0]+x[1], a), Eq(2*x[0]-b*x[1], 3)}
```

[38]:
```
sol = solve(glT, (x[0], x[1]))
sol
```

[38]:
$$\left\{x_0 : \frac{ab+3}{b+2},\ x_1 : \frac{2a-3}{b+2}\right\}$$

[39]:
```
sol = solve(glS, x)
sol
```

[39]:
$$\left\{x_0 : \frac{ab+3}{b+2},\ x_1 : \frac{2a-3}{b+2}\right\}$$

[40]:
```
linsolve(glT, x), linsolve(glL, x) #, linsolve(glS, x)
```

[40]:
$$\left(\left\{\left(\frac{ab+3}{b+2},\ \frac{2a-3}{b+2}\right)\right\},\ \left\{\left(\frac{ab+3}{b+2},\ \frac{2a-3}{b+2}\right)\right\}\right)$$

1.3.2 Nichtlineare Gleichungssysteme

[41]:
```
gn = (Eq(x[0]**2+x[1]**2-1,0),Eq(x[0]-x[1],0))
gn
```

[41]:
$$\left(x_0^2 + x_1^2 - 1 = 0,\ x_0 - x_1 = 0\right)$$

[42]:
```
lsg = solve(gn,x)
lsg
```

[42]:
$$\left[\left(-\frac{\sqrt{2}}{2},\ -\frac{\sqrt{2}}{2}\right),\ \left(\frac{\sqrt{2}}{2},\ \frac{\sqrt{2}}{2}\right)\right]$$

[43]:
```
lsgdict = [{x[j]: l[j]  for j in range(2)} for l in lsg]
lsgdict
```

[43]:
$$\left[\left\{x_0 : -\frac{\sqrt{2}}{2},\ x_1 : -\frac{\sqrt{2}}{2}\right\},\ \left\{x_0 : \frac{\sqrt{2}}{2},\ x_1 : \frac{\sqrt{2}}{2}\right\}\right]$$

[44]:
```
[gl.subs(l) for l in lsgdict for gl in gn ]
```

[44]:
[True, True, True, True]

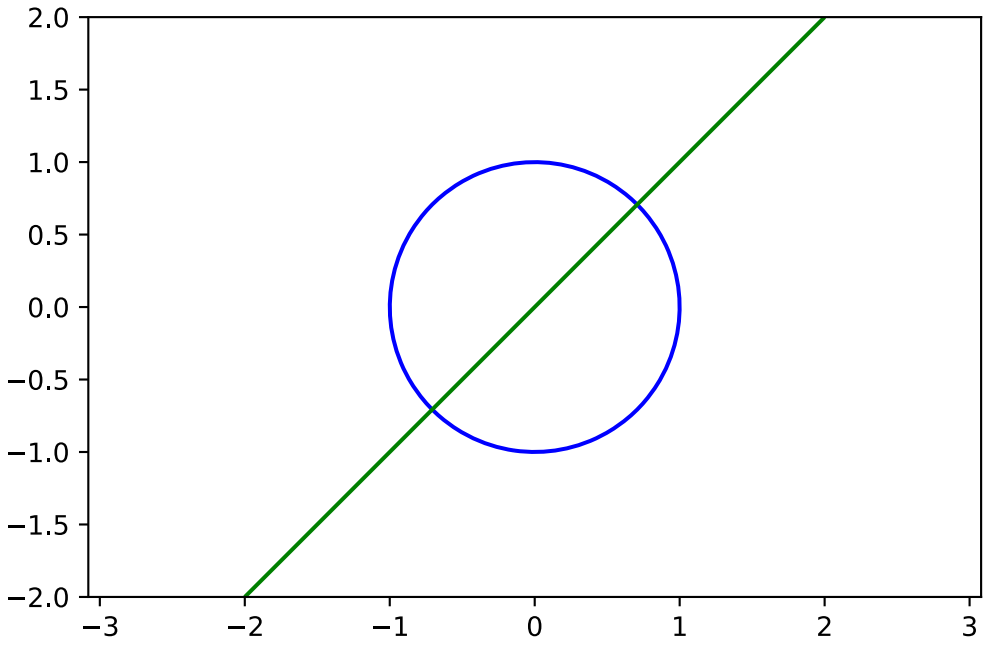
[45]:
```
gn[0].lhs
```

[45]:
$$x_0^2 + x_1^2 - 1$$

[46]:
```
gn[1]
```

[46]:
$$x_0 - x_1 = 0$$

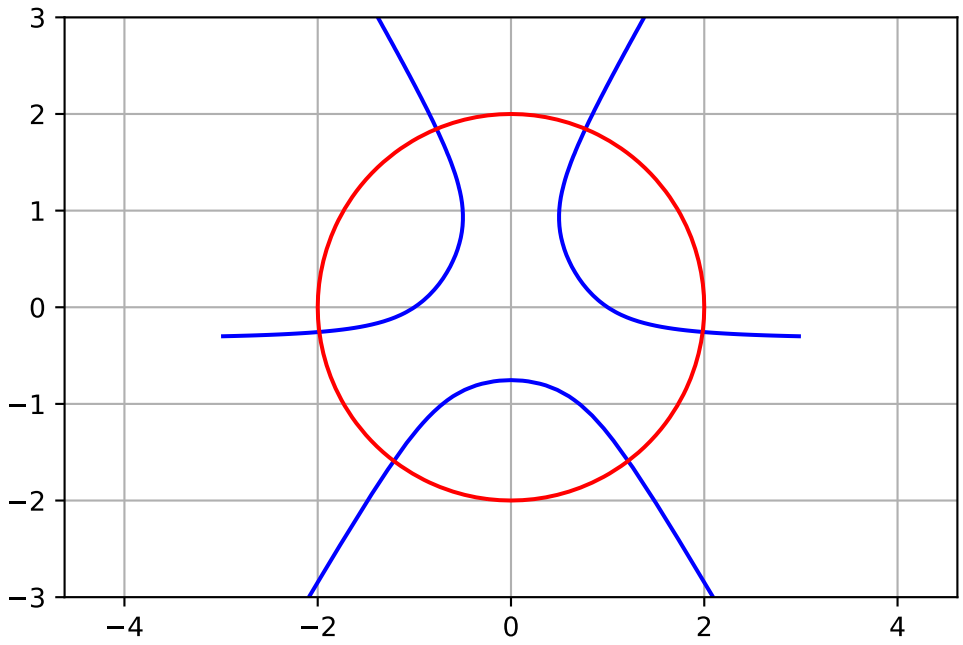
[47]:
```
xn = np.linspace(-2, 2, 100)
X0, X1 = np.meshgrid(xn, xn)
fig = plt.figure()
ax = fig.gca()
ax.contour(X0, X1, lambdify(x, gn[0].lhs)(X0, X1), [0], colors='blue')
ax.contour(X0, X1, lambdify(x, gn[1].lhs)(X0, X1), [0], colors='green')
ax.axis('equal');
```



[48]:
```
f = x[0]**2+x[1]**2+3*x[0]**2*x[1]-x[1]**3
g = x[0]**2+x[1]**2
f, g
```

[48]: $\left(3x_0^2x_1 + x_0^2 - x_1^3 + x_1^2,\ \ x_0^2 + x_1^2\right)$

[49]:
```
xn = np.linspace(-3, 3, 101)
X0, X1 = np.meshgrid(xn, xn)
fig = plt.figure()
ax = fig.gca()
ax.contour(X0, X1, lambdify(x, f)(X0, X1), [1], colors='blue')
ax.contour(X0, X1, lambdify(x, g)(X0, X1), [4], colors='red')
ax.axis('equal');
ax.grid()
```



[50]:
```
sol1 = nonlinsolve((f, g-4), x)
sol1
```

[50]:

$$\left\{\left(-2^{\frac{2}{3}}\sqrt{\frac{2-2\sqrt{3}i-4\cdot 2^{\frac{2}{3}}\sqrt[3]{1+\sqrt{3}i}+\sqrt[3]{2}\left(1+\sqrt{3}i\right)^{\frac{2}{3}}+\sqrt[3]{2}\sqrt{3}i\left(1+\sqrt{3}i\right)^{\frac{2}{3}}}{\left(1+\sqrt{3}i\right)^{\frac{10}{3}}}},\ \frac{1}{\left(-\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{1}{2}+\frac{\sqrt{3}i}{2}}}+\left(-\frac{1}{2}+\right.\right.\right.$$

[51]:
```
for l in sol1:
    print('x_0 : ', l[0].n(), 'x_1 : ', l[1].n())
```

```
x_0 :  0.684040286651337 + 0.e-24*I x_1 :  1.87938524157182 + 0.e-20*I
x_0 :  -0.684040286651337 - 1.18091519273054e-26*I x_1 :  1.87938524157182 +
0.e-20*I
x_0 :  1.96961550602442 + 0.e-21*I x_1 :  -0.347296355333861 - 0.e-22*I
x_0 :  -1.96961550602442 x_1 :  -0.347296355333861 - 0.e-22*I
x_0 :  1.28557521937308 + 3.24269556573663e-24*I x_1 :  -1.53208888623796 +
0.e-22*I
x_0 :  -1.28557521937308 x_1 :  -1.53208888623796 + 0.e-22*I
```

[52]:
```
sol2 = solve((f-1, g-4))
sol2
```

[52]:

$$\left[\left\{x_0 : -\frac{\sqrt{2}\sqrt{\frac{23-8\left(3+\sqrt{55}i\right)^{\frac{2}{3}}-3\sqrt{55}i+12\sqrt[3]{3+\sqrt{55}i}+4\sqrt{55}i\sqrt[3]{3+\sqrt{55}i}}{\left(3+\sqrt{55}i\right)^{\frac{4}{3}}}}}{2},\ x_1 : \frac{1}{\sqrt[3]{\frac{3}{8}+\frac{\sqrt{55}i}{8}}}+\sqrt[3]{\frac{3}{8}+\frac{\sqrt{55}i}{8}}\right\},\ \left\{x_0 : \frac{\sqrt{2}\sqrt{\frac{23-}{}}}{}\right.\right.$$

[54]:
```
for l in sol2:
    x0 = l[x[0]]
    x1 = l[x[1]]
    print('x_0 : {0: 12.9f} + {1: 12.9f} i \t'.format(re(x0).n(), im(x0).n()), \
          'x_1 : {0: 12.9f} + {1: 12.9f} i   '.format(re(x1).n(), im(x1).n()))
```

```
                                                                            ␣
↪---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)

<ipython-input-54-bc4f7e48bde3> in <module>
  3     x1 = l[x[1]]
  4     print('x_0 : {0: 12.9f} + {1: 12.9f} i \t'.format(re(x0).n(),␣
↪im(x0).n()), \
----> 5           'x_1 : {0: 12.9f} + {1: 12.9f} i   '.format(re(x1).n(),␣
↪im(x1).n()))

TypeError: unsupported format string passed to Zero.__format__
```

[55]:
```
im(sol2[0][x[1]]).n()
```

[55]:
$$0$$

[56]:
```
for l in sol2:
    x0 = l[x[0]]
    x1 = l[x[1]]
    print('x_0 : {0: 12.9f} + {1: 12.9e} i \t'.format(float(re(x0).n()),␣
↪float(im(x0).n())), \
          'x_1 : {0: 12.9f} + {1: 12.9e} i   '.format(float(re(x1).n()),␣
↪float(im(x1).n())))
```

```
x_0 : -0.770477482 + -2.817295628e-165 i         x_1 :  1.845633888 +
0.000000000e+00 i
x_0 :  0.770477482 +  2.817295628e-165 i         x_1 :  1.845633888 +
0.000000000e+00 i
x_0 : -1.213127093 + -1.497356852e-22 i          x_1 : -1.590070016 +
3.693191447e-127 i
x_0 :  1.213127093 +  2.994713704e-22 i          x_1 : -1.590070016 +
```

```
3.693191447e-127 i
x_0 :  1.983604574 + -2.994713704e-22 i          x_1 : -0.255563872 +
-1.846595724e-127 i
x_0 : -1.983604574 +  5.989427409e-22 i          x_1 : -0.255563872 +
-1.846595724e-127 i
```

1.4 Python rechnet komplex

1.4.1 Imaginäre Einheit

[57]:
```
x, y = symbols('x y')
z = x + I*y
z
```

[57]: $x + iy$

[58]:
```
z**4
```

[58]: $(x + iy)^4$

1.4.2 Real- und Imaginärteil

[59]:
```
abs(z)
```

[59]: $|x + iy|$

[60]:
```
re(z**4).expand()
```

[60]: $(\operatorname{re}(x))^4 - 4(\operatorname{re}(x))^3 \operatorname{im}(y) - 6(\operatorname{re}(x))^2(\operatorname{re}(y))^2 - 12(\operatorname{re}(x))^2 \operatorname{re}(y) \operatorname{im}(x) - 6(\operatorname{re}(x))^2(\operatorname{im}(x))^2 + 6(\operatorname{re}(x))^2(\operatorname{im}(y))^2 + 12\operatorname{re}(x)(\operatorname{re}(y))^2 \operatorname{im}(y) + 24\operatorname{re}(x)\operatorname{re}(y)\operatorname{im}(x)\operatorname{im}(y) + 12\operatorname{re}(x)(\operatorname{im}(x))^2 \operatorname{im}(y) - 4\operatorname{re}(x)(\operatorname{im}(y))^3 + (\operatorname{re}(y))^4 + 4(\operatorname{re}(y))^3 \operatorname{im}(x) + 6(\operatorname{re}(y))^2(\operatorname{im}(x))^2 - 6(\operatorname{re}(y))^2(\operatorname{im}(y))^2 + 4\operatorname{re}(y)(\operatorname{im}(x))^3 - 12\operatorname{re}(y)\operatorname{im}(x)(\operatorname{im}(y))^2 + (\operatorname{im}(x))^4 - 6(\operatorname{im}(x))^2(\operatorname{im}(y))^2 + (\operatorname{im}(y))^4$

?

[61]:
```
re(z)
```

[61]: $\operatorname{re}(x) - \operatorname{im}(y)$

[62]:
```
im(z)
```

[62]: $\operatorname{re}(y) + \operatorname{im}(x)$

[63]:
```
x, y = symbols('x y', real=True)
z = x + I*y
```

[64]:
```
re(z**4).expand()
```

[64]: $x^4 - 6x^2y^2 + y^4$

[65]:
```
abs(z)
```

[65]: $\sqrt{x^2 + y^2}$

1.5 Vereinfachungen (simplify) vgl. Lektion 1

[66]:
```
t = sin(x)**2 + cos(x)**2
t
```

[66]: $\sin^2(x) + \cos^2(x)$

[67]:
```
simplify(t)
```

[67]: 1

[68]:
```
r = (x**3 + x**2 +x +1) / (x**2 +2* x +1)
r
```

[68]: $\frac{x^3 + x^2 + x + 1}{x^2 + 2x + 1}$

[69]:
```
simplify(r)
```

[69]: $\frac{x^2 + 1}{x + 1}$

[70]:
```
ex = exp((x-1)**2+log(c*exp(y**2)-exp(4*x))-(x+1)**2)
ex
```

[70]: $\left(ce^{y^2} - e^{4x}\right) e^{(x-1)^2-(x+1)^2}$

[71]:
```
simplify(ex)
```

[71]: $ce^{-4x+y^2} - 1$

[72]:
```
p = x**2 + 2*x + 1
```

[73]:
```
simplify(p)
```

[73]: $x^2 + 2x + 1$

1.5.1 Faktorisieren (factor) und ausmultiplizieren (expand)

[74]:
```
pf = factor(p)
pf
```

[74]: $(x+1)^2$

[75]:
```
q = expand(pf)
q
```

[75]: x^2+2x+1

[76]:
```
r = (x**2-y**2)/(x+y)**2
r
```

[76]: $\frac{x^2-y^2}{(x+y)^2}$

[77]:
```
factor(r)
```

[77]: $\frac{x-y}{x+y}$

[78]:
```
r1 = expand(r) + x**2/(y**2-1)
r1
```

[78]: $\frac{x^2}{x^2+2xy+y^2}+\frac{x^2}{y^2-1}-\frac{y^2}{x^2+2xy+y^2}$

[79]:
```
together(r1)
```

[79]: $\frac{x^2\left(y^2-1\right)+x^2\left(x^2+2xy+y^2\right)-y^2\left(y^2-1\right)}{\left(y^2-1\right)\left(x^2+2xy+y^2\right)}$

1.5.2 "cancel" bringt rationale Ausdrücke in gekürzte Standardform

[80]:
```
cancel(r)
```

[80]: $\frac{x-y}{x+y}$

[81]:
```
g = (x*exp(-x)-(x-1)*exp(x))**3
g
```

[81]: $\left(xe^{-x}-(x-1)\,e^{x}\right)^3$

[82]:
```
g = expand(g)
g
```

[82]:

$$-x^3e^{3x} + 3x^3e^x - 3x^3e^{-x} + x^3e^{-3x} + 3x^2e^{3x} - 6x^2e^x + 3x^2e^{-x} - 3xe^{3x} + 3xe^x + e^{3x}$$

[83]:
```
factor(g)
```

[83]:
$$-\left(xe^{2x} - x - e^{2x}\right)^3 e^{-3x}$$

[84]:
```
simplify(g)
```

[84]:
$$\left(x^3 + 3x^2\left(1 - x\right)e^{2x} + \left(-x^3e^{2x} + 3x^3 + 3x^2e^{2x} - 6x^2 - 3xe^{2x} + 3x + e^{2x}\right)e^{4x}\right)e^{-3x}$$

1.5.3 Zusammenfassen (collect)

[85]:
```
collect(g, x)
```

[85]:
$$x^3\left(-e^{3x} + 3e^x - 3e^{-x} + e^{-3x}\right) + x^2\left(3e^{3x} - 6e^x + 3e^{-x}\right) + x\left(-3e^{3x} + 3e^x\right) + e^{3x}$$

[86]:
```
collect(g, exp(x))
```

[86]:
$$x^3e^{-3x} + \left(-3x^3 + 3x^2\right)e^{-x} + \left(3x^3 - 6x^2 + 3x\right)e^x + \left(-x^3 + 3x^2 - 3x + 1\right)e^{3x}$$

[87]:
```
collect(g, exp(x), exact=True)
```

[87]:
$$-x^3e^{3x} - 3x^3e^{-x} + x^3e^{-3x} + 3x^2e^{3x} + 3x^2e^{-x} - 3xe^{3x} + \left(3x^3 - 6x^2 + 3x\right)e^x + e^{3x}$$

[88]:
```
collect(g, x**2, exact=True)
```

[88]:
$$-x^3e^{3x} + 3x^3e^x - 3x^3e^{-x} + x^3e^{-3x} + x^2\left(3e^{3x} - 6e^x + 3e^{-x}\right) - 3xe^{3x} + 3xe^x + e^{3x}$$

[89]:
```
k = 1+x/(x -2/(x-4/(8-x))) # Kettenbruch
k
```

[89]:
$$\frac{x}{x - \frac{2}{x - \frac{4}{8-x}}} + 1$$

[90]:
```
simplify(k)
```

[90]:
$$\frac{x}{x - \frac{2}{x + \frac{4}{x-8}}} + 1$$

[91]:
```
cancel(k)
```

[91]:
$$\frac{2x^3 - 16x^2 + 6x + 16}{x^3 - 8x^2 + 2x + 16}$$

1.5.4 Partialbruchzerlegung

[92]:
```
r = (x*y+2*x+1)/(x**2-3*x+1)
r
```

[92]:
$$\frac{xy + 2x + 1}{x^2 - 3x + 1}$$

[93]:
```
r = apart(r, x, full=True) #Partialbruchzerlegung
r
```

[93]:
$$\operatorname{RootSum}\left(w^2 - 3w + 1, \left(a \mapsto \frac{a\left(\frac{3y}{5} + \frac{8}{5}\right) - \frac{2y}{5} - \frac{7}{5}}{-a + x}\right)\right)$$

[94]:
```
rr = r.doit()
rr
```

[94]:
$$\frac{-\frac{2y}{5} + \left(\frac{3}{2} - \frac{\sqrt{5}}{2}\right)\left(\frac{3y}{5} + \frac{8}{5}\right) - \frac{7}{5}}{x - \frac{3}{2} + \frac{\sqrt{5}}{2}} + \frac{-\frac{2y}{5} + \left(\frac{\sqrt{5}}{2} + \frac{3}{2}\right)\left(\frac{3y}{5} + \frac{8}{5}\right) - \frac{7}{5}}{x - \frac{3}{2} - \frac{\sqrt{5}}{2}}$$

[95]:
```
cancel(rr)
```

[95]:
$$\frac{xy + 2x + 1}{x^2 - 3x + 1}$$

1.5.5 trigsimp und powsimp

[96]:
```
f = sin(x)**4 - 2*sin(x)**2*cos(x)**2 + cos(x)**4
f
```

[96]:
$$\sin^4(x) - 2\sin^2(x)\cos^2(x) + \cos^4(x)$$

[97]:
```
simplify(f), trigsimp(f)
```

[97]:
$$\left(\frac{\cos(4x)}{2} + \frac{1}{2},\ \frac{\cos(4x)}{2} + \frac{1}{2}\right)$$

[98]:
```
simplify(sinh(x)**2+cosh(x)**2), trigsimp(sinh(x)**2+cosh(x)**2)
```

[98]:
$$(\cosh(2x),\ \cosh(2x))$$

[99]:
```
expand(cos(x+y))
```

[99]:
$$\cos(x + y)$$

[100]:
```
expand_trig(cos(x+y))
```

[100]:
$$-\sin(x)\sin(y) + \cos(x)\cos(y)$$

[101]:
```
expand_trig(sinh(x+y))
```

[101]: $\sinh(x)\cosh(y) + \sinh(y)\cosh(x)$

[102]:
```
f = expand_trig(tan(4*x))
f
```

[102]: $$\frac{-4\tan^3(x) + 4\tan(x)}{\tan^4(x) - 6\tan^2(x) + 1}$$

[103]:
```
trigsimp(f)
```

[103]: $\tan(4x)$

[104]:
```
simplify(x**a*x)
```

[104]: x^{a+1}

[105]:
```
powsimp(x**a*x**b)
```

[105]: x^{a+b}

[106]:
```
f = x**a*x*sin(x)/cos(x)
trigsimp(f)
```

[106]: $xx^a\tan(x)$

[107]:
```
powsimp(f)
```

[107]: $$\frac{x^{a+1}\sin(x)}{\cos(x)}$$

[108]:
```
simplify(f)
```

[108]: $x^{a+1}\tan(x)$

1.6 Umformungen (rewrite)

[109]:
```
sin(2*x).rewrite(cot)
```

[109]: $$\frac{2\cot(x)}{\cot^2(x) + 1}$$

[110]:
```
sin(2*x).rewrite(cos)
```

[110]: $\cos\left(2x - \frac{\pi}{2}\right)$

[111]:
```
sin(2*x).rewrite(exp)
```

[111]:

$$-\frac{i\left(e^{2ix}-e^{-2ix}\right)}{2}$$

[112]:
```
cot(x+1).rewrite(tan)
```

[112]:
$$\frac{1}{\tan\left(x+1\right)}$$

[113]:
```
tan(x).rewrite(sin)
```

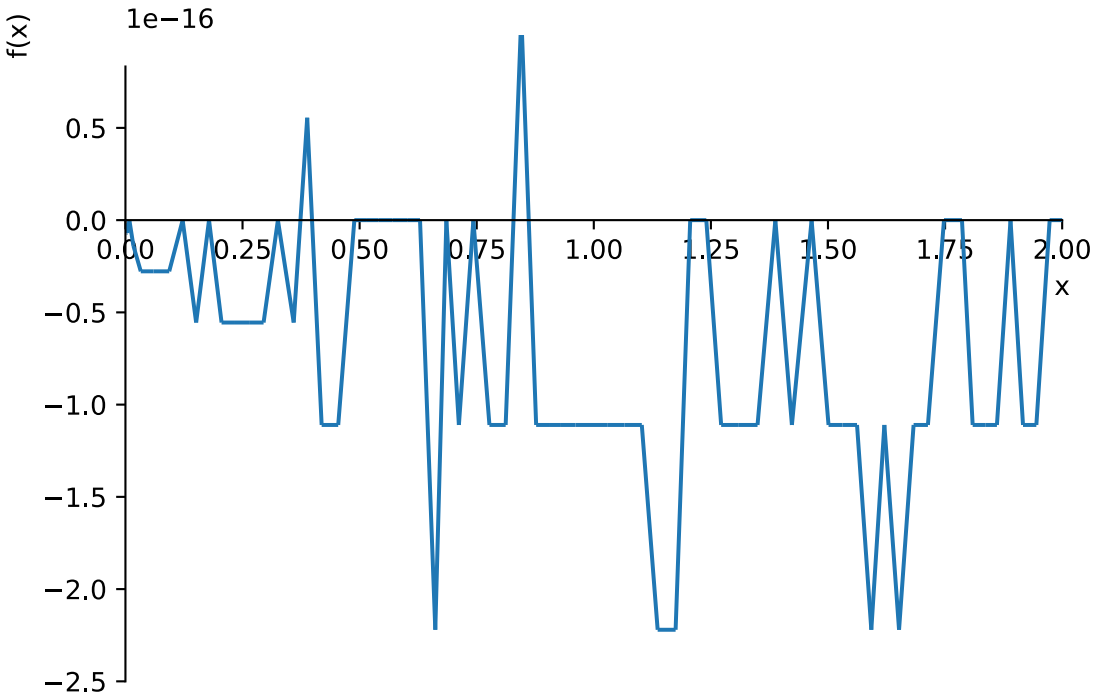
[113]:
$$\frac{2\sin^2\left(x\right)}{\sin\left(2x\right)}$$

[114]:
```
x = symbols('x', positive=True)
```

[115]:
```
besselj(Rational(1,2), x).rewrite(sin) # Schade
```

[115]:
$$J_{\frac{1}{2}}\left(x\right)$$

[116]:
```
plot(besselj(Rational(1, 2), x) - sin(x)*sqrt(2/pi/x), (x, 0, 2));
```



[117]:
```
simplify(besselj(Rational(1, 2), x) - sin(x)*sqrt(2/pi/x))
```

[117]:
$$0$$

[118]:
```
gamma(4)  # Verallgemeinerung von Fakultät  gamma(n+1) = n! für natürliche Zahlen
```

[118]:
$$6$$

[119]:
```
gamma(5)
```

[119]:
$$24$$

[120]:
```
f = gamma(x)*gamma(x+Rational(1,2)) - 2**(1-2*x)*sqrt(pi)*gamma(2*x)
f
```

[120]:
$$-2^{1-2x}\sqrt{\pi}\Gamma\left(2x\right) + \Gamma\left(x\right)\Gamma\left(x + \frac{1}{2}\right)$$

[121]:
```
simplify(f)
```

[121]:
$$0$$