lektion9

December 11, 2025

Inhalt

1 Lektion 9

1.1 Gleichungssysteme

Zu

$$F : \mathbb{R}^n \to \mathbb{R}^m \quad \underbrace{\begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix}}_{=x} \mapsto \underbrace{\begin{bmatrix} F_0(x_0, \dots x_{n-1}) \\ \vdots \\ F_{m-1}(x_0, \dots, x_{n-1}) \end{bmatrix}}_{=F(x)}$$

suche x, so, dass $F(x) = 0$.

[1]:
```
#%matplotlib notebook
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = [4, 3]
from sympy import *
a, b = symbols('a b')
x = symbols('x:2')
init_printing()
x # x ist ein Tupel mit zwei zwei Symbolen
```

[1]:
$$(x_0,\ x_1)$$

1.1.1 Lineare Systeme

Das machen wir nächste Woche ausführlicher, sobald wir Matrizen eingeführt haben.

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

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

[3]:
```
sol = linsolve(lgs, x)
sol
```

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

[4]:
```
sol = solve(lgs, x)
sol
```

[4]:
$$\left\{x_0 : \frac{ab+3}{b+2},\ x_1 : \frac{2a-3}{b+2}\right\}$$

1.1.2 Nichtlineare Systeme

[5]:
```
nls = [Eq(x[0]**2+x[1]**2, 1), Eq(x[0], x[1])]
nls
```

[5]:
$$[x_0^2 + x_1^2 = 1,\ x_0 = x_1]$$

[6]:
```
lsg = nonlinsolve(nls, x)
lsg
```

[6]:
$$\left\{\left(-\frac{\sqrt{2}}{2},\ -\frac{\sqrt{2}}{2}\right),\left(\frac{\sqrt{2}}{2},\ \frac{\sqrt{2}}{2}\right)\right\}$$

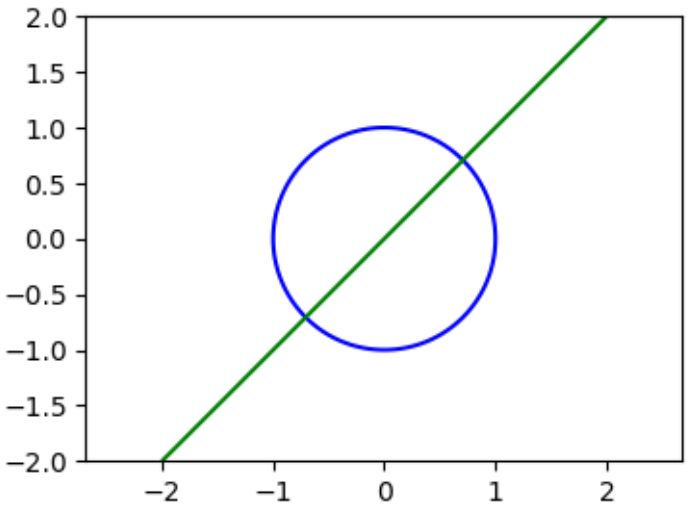
[7]:
```
lsg = solve(nls, x, )
lsg
```

[7]:
$$\left[\left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right), \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)\right]$$

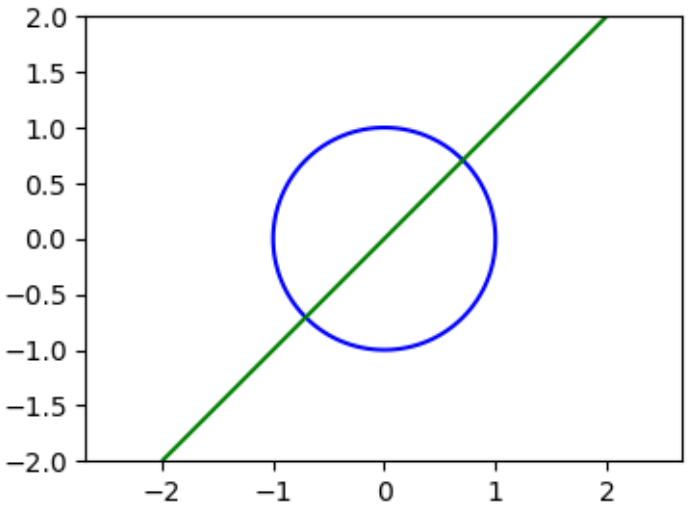
[8]:
```
nls
```

[8]:
$$[x_0^2 + x_1^2 = 1,\ x_0 = x_1]$$

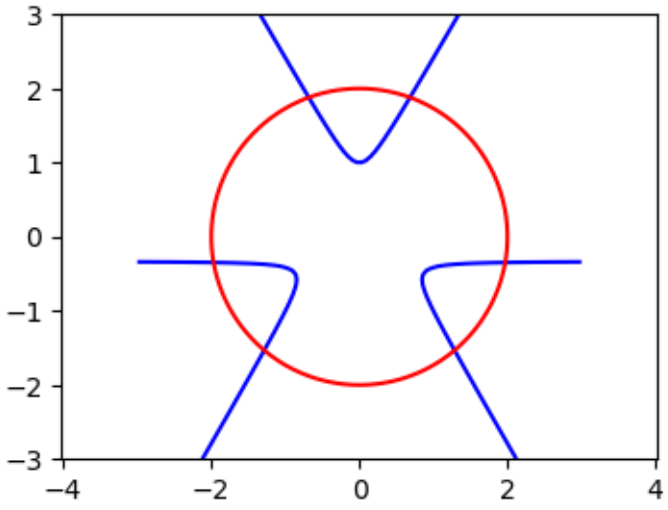
[9]:
```
xn = np.linspace(-2 , 2, 100)
X = np.meshgrid(xn, xn) # X ist Tupel aus X0, X1
fig, ax = plt.subplots()
ax.contour(*X, lambdify(x, nls[0].lhs-nls[0].rhs)(*X), [0], colors='blue')
ax.contour(*X, lambdify(x, nls[1].lhs-nls[1].rhs)(*X), [0], colors='green')
ax.axis('equal');
```



[10]:
```
# alternativ
xn = np.linspace(-2 , 2, 100)
X, Y = np.meshgrid(xn, xn)
fig, ax = plt.subplots()
ax.contour(X, Y, lambdify((x[0], x[1]), nls[0].lhs-nls[0].rhs)(X, Y), [0],␣
 ↪colors='blue')
ax.contour(X, Y, lambdify((x[0], x[1]), nls[1].lhs-nls[1].rhs)(X, Y), [0],␣
 ↪colors='green')
ax.axis('equal');
```



[11]:
```
x, y = symbols('x y')  # jetzt ist x ein einziges Symbol
f = x**2 + y**2 + 3 * x**2 * y - y**3
g = x**2 + y**2
xn = np.linspace(-3, 3, 100)
X, Y = np.meshgrid(xn, xn)
fig, ax = plt.subplots()
ax.contour(X, Y, lambdify((x, y), f)(X, Y), [0], colors='blue')
ax.contour(X, Y, lambdify((x, y), g)(X, Y), [4], colors='red') # g(x,y) = 4␣
 ↪Hoehenlinie
ax.axis('equal');
```



[12]:
```
sol = solve([f, g-4]  )
lsg = sol[0]
lsg
```

[12]:
$$\left\{x : -\sqrt[6]{2}\sqrt{\frac{-\sqrt[3]{2}-\frac{\sqrt{3}i\sqrt[3]{1+\sqrt{3}i}}{2}-\frac{\sqrt[3]{1+\sqrt{3}i}}{2}+\left(2+2\sqrt{3}i\right)^{\frac{2}{3}}}{\left(1+\sqrt{3}i\right)^{\frac{2}{3}}}},\ y : \frac{1}{\sqrt[3]{\frac{1}{2}+\frac{\sqrt{3}i}{2}}}+\sqrt[3]{\frac{1}{2}+\frac{\sqrt{3}i}{2}}\right\}$$

[13]:
```
for i, l in enumerate(sol):
    lxn = complex(l[x].n()) # complex wandelt sympy Summe (a+i*b) in python␣
 ↪complex um
    lyn = complex(l[y].n())
    print(f'{i}. Lsg \t x: {lxn.real:9.6f}{lxn.imag:+9.6f}i\t y: {lyn.real:9.
 ↪6f}{lyn.imag:+9.6f}i')
```

```
0. Lsg   x: -0.684040-0.000000i  y:  1.879385+0.000000i
1. Lsg   x:  0.684040+0.000000i  y:  1.879385+0.000000i
2. Lsg   x: -1.285575+0.000000i  y: -1.532089+0.000000i
3. Lsg   x:  1.285575-0.000000i  y: -1.532089+0.000000i
4. Lsg   x: -1.969616+0.000000i  y: -0.347296-0.000000i
5. Lsg   x:  1.969616-0.000000i  y: -0.347296-0.000000i
```

Beispiel (Fibonacci Zahlen) Die Fibonacci-Zahlen definiert durch die Rekursion

$$f_{n+2} = f_{n+1} + f_n \text{ mit } f_0 = 0,\ f_1 = 1$$

können auch durch $f_n = ax^n + by^n$ berechnet werden.

Wir bestimmen die a, b, x, y .

[14]:
```
a, b, x, y = symbols("a b x y")
unb = [a, b, x, y]
```

[15]:
```
def fib(n):
    return a*x**n + b*y**n
```

[16]:
```
# Menge der nichtlinearen Gleichungen
nls = FiniteSet()
for n_ in range(1, 5):
    nls |= {Eq(fibonacci(n_), fib(n_))}

# alternativ mit einer Liste
nll = []
for n_ in range(1, 5):
    nll += [Eq(fibonacci(n_), fib(n_))]

display(nls)
display(nll) # solve und nonlinsolve kommen mit beiden zurecht
```

$$\{1 = ax + by, 1 = ax^2 + by^2, 2 = ax^3 + by^3, 3 = ax^4 + by^4\}$$

$$[1 = ax + by,\ 1 = ax^2 + by^2,\ 2 = ax^3 + by^3,\ 3 = ax^4 + by^4]$$

[17]:
```
ls_solve = solve(nll) # solve liefert eine Liste mit Dictionaries für die␣
 ↪Unbekannten
ls_solve
```

[17]:
$$\left[\left\{a : -\frac{\sqrt{5}}{5},\ b : \frac{\sqrt{5}}{5},\ x : \frac{1}{2} - \frac{\sqrt{5}}{2},\ y : \frac{1}{2} + \frac{\sqrt{5}}{2}\right\},\ \left\{a : \frac{\sqrt{5}}{5},\ b : -\frac{\sqrt{5}}{5},\ x : \frac{1}{2} + \frac{\sqrt{5}}{2},\ y : \frac{1}{2} - \frac{\sqrt{5}}{2}\right\}\right]$$

[18]:
```
ls_nls = nonlinsolve(
    nls, unb)  # nonlinsolve liefert eine Menge mit (geordneten) Tupeln.
#Die Ordnung ist die in der Liste der Variablen
ls_nls
```

[18]:
$$\left\{\left(-\frac{\sqrt{5}}{5},\ \frac{\sqrt{5}}{5},\ \frac{1}{2} - \frac{\sqrt{5}}{2},\ \frac{1}{2} + \frac{\sqrt{5}}{2}\right), \left(\frac{\sqrt{5}}{5},\ -\frac{\sqrt{5}}{5},\ \frac{1}{2} + \frac{\sqrt{5}}{2},\ \frac{1}{2} - \frac{\sqrt{5}}{2}\right)\right\}$$

[19]:
```
lsg = {var_ : erg_ for erg_, var_  in zip(list(ls_nls)[0],unb)}
lsg # das ist der erste Eintrag von ls_solve
```

[19]:
$$\left\{a : -\frac{\sqrt{5}}{5},\ b : \frac{\sqrt{5}}{5},\ x : \frac{1}{2} - \frac{\sqrt{5}}{2},\ y : \frac{1}{2} + \frac{\sqrt{5}}{2}\right\}$$

[20]:
```
n = symbols('n', integer=True)
fib(n).subs(lsg)
```

[20]:
$$-\frac{\sqrt{5}\left(\frac{1}{2} - \frac{\sqrt{5}}{2}\right)^n}{5} + \frac{\sqrt{5}\left(\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^n}{5}$$

[21]:
```
for n_ in range(5):
    print(fibonacci(n_), (fib(n_).subs(lsg)) )
```

```
0 0
1 -sqrt(5)*(1/2 - sqrt(5)/2)/5 + sqrt(5)*(1/2 + sqrt(5)/2)/5
1 -sqrt(5)*(1/2 - sqrt(5)/2)**2/5 + sqrt(5)*(1/2 + sqrt(5)/2)**2/5
2 -sqrt(5)*(1/2 - sqrt(5)/2)**3/5 + sqrt(5)*(1/2 + sqrt(5)/2)**3/5
3 -sqrt(5)*(1/2 - sqrt(5)/2)**4/5 + sqrt(5)*(1/2 + sqrt(5)/2)**4/5
```

[22]:
```
for n_ in range(10):
    if simplify(fibonacci(n_) - fib(n_).subs(lsg)):  # 0 ist False
        raise ValueError(f"ungleich für n={n_}")
```

Kein Fehler, passt.

Wir prüfen jetzt die Rekursionsgleichung $f_{n+2} = f_{n+1} + f_n$

[23]:
```
Eq(fib(n + 2), fib(n + 1) + fib(n)).subs(lsg)
```

[23]:
$$-\frac{\sqrt{5}\left(\frac{1}{2}-\frac{\sqrt{5}}{2}\right)^{n+2}}{5}+\frac{\sqrt{5}\left(\frac{1}{2}+\frac{\sqrt{5}}{2}\right)^{n+2}}{5}=-\frac{\sqrt{5}\left(\frac{1}{2}-\frac{\sqrt{5}}{2}\right)^{n}}{5}-\frac{\sqrt{5}\left(\frac{1}{2}-\frac{\sqrt{5}}{2}\right)^{n+1}}{5}+\frac{\sqrt{5}\left(\frac{1}{2}+\frac{\sqrt{5}}{2}\right)^{n}}{5}+\frac{\sqrt{5}\left(\frac{1}{2}+\frac{\sqrt{5}}{2}\right)^{n+1}}{5}$$

[24]:
```
(fib(n + 2) - fib(n + 1) - fib(n)).subs(lsg).expand()
```

[24]:
$$0$$

1.2 Python rechnet komplex

[25]:
```
I**2
```

[25]:
$$-1$$

[26]:
```
exp(I * pi / 2)
```

[26]:
$$i$$

[27]:
```
((2 + I)**3).expand()
```

[27]:
$$2+11i$$

[28]:
```
a = x + I * y
a
```

[28]:
$$x+iy$$

[29]:
```
(a**2).expand()
```

[29]:
$$x^2+2ixy-y^2$$

[30]:
```
re(a**2).expand()
```

[30]:
$$\left(\operatorname{re}\left(x\right)\right)^2-2\operatorname{re}\left(x\right)\operatorname{im}\left(y\right)-\left(\operatorname{re}\left(y\right)\right)^2-2\operatorname{re}\left(y\right)\operatorname{im}\left(x\right)-\left(\operatorname{im}\left(x\right)\right)^2+\left(\operatorname{im}\left(y\right)\right)^2$$

[31]:
```
re(a)
```

[31]:
$$\operatorname{re}\left(x\right)-\operatorname{im}\left(y\right)$$

[32]:
```
im(a)
```

[32]:
$$\operatorname{re}\left(y\right)+\operatorname{im}\left(x\right)$$

[33]:
```
x, y = symbols('x y', real=True)
z = x + I*y
```

[34]:
```
re(z**2).expand()
```

[34]: $x^2 - y^2$

[35]:
```
abs(z)
```

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

1.3 Vereinfachen und zusammenfassen

vgl. Lektion 2

[36]:
```
x, y, z, a, b, c = symbols('x y z a b c')
```

1.3.1 faktorisieren (factor)

[37]:
```
q = x**2 + 2 * x + 1
```

[38]:
```
h = factor(q)
h
```

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

[39]:
```
q.factor()
```

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

1.3.2 ausmultiplizieren (expand)

[40]:
```
h
```

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

[41]:
```
h.expand()
```

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

[42]:
```
expand(h)
```

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

1.3.3 rationale Ausdrücke in gekürzte Standardform bringen

[43]:
```
f = (x**2 - y**2)/(x+y)**2
f
```

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

[44]:
```
cancel(f)
```

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

[45]:
```
ratsimp(f)
```

[45]: $-\frac{2y}{x+y}+1$

1.3.4 collect und coeff

[46]:
```
g = 0
for j in range(4):
    g += (x + j * y + (j * x - y) * exp(y))**j
g
```

[46]: $x+y+(x-y)\,e^{y}+\left(x+2y+(2x-y)\,e^{y}\right)^{2}+\left(x+3y+(3x-y)\,e^{y}\right)^{3}+1$

[47]:
```
f = expand(g)  # expand als Funktion # f = g.expand() expand als Methode
f
```

[47]: $27x^3e^{3y}+27x^3e^{2y}+9x^3e^{y}+x^3-27x^2ye^{3y}+63x^2ye^{2y}+51x^2ye^{y}+9x^2y+4x^2e^{2y}+4x^2e^{y}+x^2+9xy^2e^{3y}-51xy^2e^{2y}+63xy^2e^{y}+27xy^2-4xye^{2y}+6xye^{y}+4xy+xe^{y}+x-y^3e^{3y}+9y^3e^{2y}-27y^3e^{y}+27y^3+y^2e^{2y}-4y^2e^{y}+4y^2-ye^{y}+y+1$

[48]:
```
collect(f, x) # collect als Funktion
```

[48]: $x^3\left(27e^{3y}+27e^{2y}+9e^{y}+1\right)+x^2\left(-27ye^{3y}+63ye^{2y}+51ye^{y}+9y+4e^{2y}+4e^{y}+1\right)+x\left(9y^2e^{3y}-51y^2e^{2y}+63y^2e^{y}+27y^2-4ye^{2y}+6ye^{y}+4y+e^{y}+1\right)-y^3e^{3y}+9y^3e^{2y}-27y^3e^{y}+27y^3+y^2e^{2y}-4y^2e^{y}+4y^2-ye^{y}+y+1$

[49]:
```
f.collect(exp(y)) # collect als Methode
```

[49]: $x^3+9x^2y+x^2+27xy^2+4xy+x+27y^3+4y^2+y+\left(27x^3-27x^2y+9xy^2-y^3\right)e^{3y}+\left(27x^3+63x^2y+4x^2-51xy^2-4xy+9y^3+y^2\right)e^{2y}+\left(9x^3+51x^2y+4x^2+63xy^2+6xy+x-27y^3-4y^2-y\right)e^{y}+1$

[50]:
```
collect(f, exp(y), exact=True)
```

[50]: $27x^3e^{3y}+27x^3e^{2y}+x^3-27x^2ye^{3y}+63x^2ye^{2y}+9x^2y+4x^2e^{2y}+x^2+9xy^2e^{3y}-51xy^2e^{2y}+27xy^2-4xye^{2y}+4xy+x-y^3e^{3y}+9y^3e^{2y}+27y^3+y^2e^{2y}+4y^2+y+\left(9x^3+51x^2y+4x^2+63xy^2+6xy+x-27y^3-4y^2-y\right)e^{y}+1$

[51]:
```
f.collect(x**2, exact=True)
```

[51]: $27x^3e^{3y}+27x^3e^{2y}+9x^3e^{y}+x^3+x^2\left(-27ye^{3y}+63ye^{2y}+51ye^{y}+9y+4e^{2y}+4e^{y}+1\right)+9xy^2e^{3y}-51xy^2e^{2y}+63xy^2e^{y}+27xy^2-4xye^{2y}+6xye^{y}+4xy+xe^{y}+x-y^3e^{3y}+9y^3e^{2y}-27y^3e^{y}+27y^3+y^2e^{2y}-4y^2e^{y}+4y^2-ye^{y}+y+1$

[52]:
```
collect(f, x).coeff(x**2)
```

[52]: $-27ye^{3y} + 63ye^{2y} + 51ye^{y} + 9y + 4e^{2y} + 4e^{y} + 1$

[53]:
```
collect(f, exp(y)).coeff(exp(y))
```

[53]: $9x^3 + 51x^2y + 4x^2 + 63xy^2 + 6xy + x - 27y^3 - 4y^2 - y$

Achtung

[54]:
```
g = y + (x + 2 * z) * exp(x)
g.expand()
```

[54]: $xe^{x} + y + 2ze^{x}$

[55]:
```
g.expand().coeff(x, 0)
```

[55]: y

```
ausdruck.coef(x, 0)
```

gibt die von x unabhängingen Terme zurück. Der Term $2 * z * \exp(x)$ wird als von x abhängig interpretiert.

[56]:
```
expx = symbols('expx')
g.replace(exp(x), expx).expand().coeff(x, 0).replace(expx, exp(x))
```

[56]: $y + 2ze^{x}$

[57]:
```
g.replace(exp(x), expx).expand().coeff(x, 0)
```

[57]: $2expxz + y$

1.3.5 Kettenbruch

[58]:
```
k = 1+x/(x -2/(x-4/(8-x))) #  Kettenbruch (continued fraction)
k
```

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

[59]:
```
cancel(k)
```

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

[60]:
```
simplify(k)
```

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

[61]:
```
ratsimp(k)
```

[61]:
$$\frac{2x - 16}{x^3 - 8x^2 + 2x + 16} + 2$$

1.3.6 Partialbruchzerlegung

[62]:
```
h = (4*x**3+10*x**2+12)/(x**4+3*x**3+3*x**2+x)
h
```

[62]:
$$\frac{4x^3 + 10x^2 + 12}{x^4 + 3x^3 + 3x^2 + x}$$

[63]:
```
h = apart(h)  # Partialbruchzerlegung (parital fraction decomposition)
h
```

[63]:
$$-\frac{8}{x+1} - \frac{10}{(x+1)^2} - \frac{18}{(x+1)^3} + \frac{12}{x}$$

[64]:
```
together(h)
```

[64]:
$$\frac{2\left(-4x\left(x+1\right)^2 - 5x\left(x+1\right) - 9x + 6\left(x+1\right)^3\right)}{x\left(x+1\right)^3}$$

[65]:
```
ratsimp(h)
```

[65]:
$$\frac{4x^3 + 10x^2 + 12}{x^4 + 3x^3 + 3x^2 + x}$$

1.3.7 Vereinfachung unter Annahmen (assumptions)

[66]:
```
f = log(y / x) - log(y) + log(x)
f
```

[66]:
$$\log\left(x\right) - \log\left(y\right) + \log\left(\frac{y}{x}\right)$$

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

[67]:
$$\log\left(x\right) - \log\left(y\right) + \log\left(\frac{y}{x}\right)$$

[68]:
```
x, y = symbols('x y', positive=True)
f = log(y / x) - log(y) + log(x)
simplify(f)
```

[68]:
$$0$$

1.3.8 trigsimp und powsimp

[69]:
```
x, y = symbols('x y')
x.assumptions0, y.assumptions0
```

[69]:
```
({'commutative': True}, {'commutative': True})
```

[70]:
```
f = sin(x)**4 - 2 * sin(x)**2 * cos(x)**2 + cos(x)**4
f
```

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

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

[71]: $\frac{\cos(4x)}{2} + \frac{1}{2}$

[72]:
```
trigsimp(f)
```

[72]: $\frac{\cos(4x)}{2} + \frac{1}{2}$

[73]:
```
expand(cos(x + y)) # funktioniert nicht
```

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

[74]:
```
expand_trig(cos(x + y))
```

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

[75]:
```
expand_trig(sinh(x + y))
```

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

[76]:
```
a = symbols('a')
```

[77]:
```
powsimp(x**a * x**b)
```

[77]: x^{a+b}

[78]:
```
trigsimp(x**a * x * sin(x) / cos(x))
```

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

[79]:
```
powsimp(x**a * x * sin(x) / cos(x))
```

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

[80]:
```
simplify(x**a * x * sin(x) / cos(x))
```

[80]:

$x^{a+1} \tan(x)$

[81]:
```
powsimp(x**a * y**a)  # Uebungen
```

[81]: $x^a y^a$

[82]:
```
powsimp((x**a)**b)
```

[82]: $(x^a)^b$

1.4 Umformungen (rewrite)

[83]:
```
sin(2 * x).rewrite(tan)
```

[83]: $\frac{2\tan(x)}{\tan^2(x)+1}$

[84]:
```
sin(2 * x).rewrite(cos)
```

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

[85]:
```
sin(2 * x).rewrite(exp)
```

[85]: $-\frac{i\left(e^{2ix} - e^{-2ix}\right)}{2}$

[86]:
```
tan(x).rewrite(sin)
```

[86]: $\frac{2\sin^2(x)}{\sin(2x)}$

1.5 Reihenentwicklung (Taylor)

Für eine glatte Funktion f ist

$$Tf(x; x_0) := \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

die Taylorreihe. Sie muss nicht konvergieren!

[87]:
```
# die ersten acht Terme um den Entwicklungspunkt 0
series(exp(x), x, 0, 8)
```

[87]: $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + O\left(x^8\right)$

Falls die Funktion hinreichend oft differenzierbar ist, ist der Taylorrest, das was sympy als $O(x^n)$ angibt klein.

Eine Funktion f ist in $\mathcal{O}(x^n, x \to 0)$, falls

$$\limsup_{x \to 0} \left| \frac{f(x)}{x^n} \right| < \infty.$$

In sympy fehlt das $x \to 0$.

[88]:
```
T = 1 + x + O(x**2, (x, 0))
T
```

[88]:
$$1 + x + O\left(x^2\right)$$

[89]:
```
(T/x).expand()
```

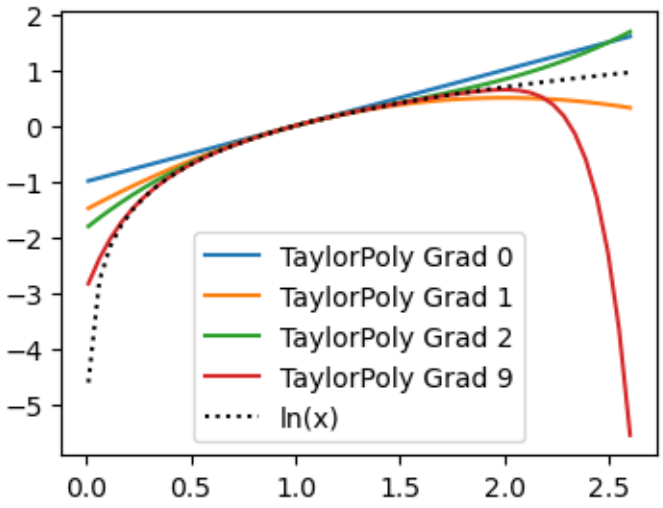
[89]:
$$\frac{1}{x} + 1 + O\left(x\right)$$

[90]:
```
# Taylorentwicklung von ln um x_0 = 1
T2s = {} # dictionary
for k in [1, 2, 3, 10]:
    T2s[k] = ln(x).series(x, 1, k+1).removeO()
T2s
```

[90]:
$$\left\{1 : x - 1,\ 2 : x - \frac{(x-1)^2}{2} - 1,\ 3 : x + \frac{(x-1)^3}{3} - \frac{(x-1)^2}{2} - 1,\ 10 : x - \frac{(x-1)^{10}}{10} + \frac{(x-1)^9}{9} - \frac{(x-1)^8}{8} + \frac{(x-\right.$$

[91]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(0.01, 2.6)
for k in T2s:
    ax.plot(xn, lambdify(x, T2s[k])(xn), label=f'TaylorPoly Grad {k-1}')

ax.plot(xn, np.log(xn), 'k:', label='ln(x)')
plt.legend(loc=8);
```



Die Taylorentwicklung kann zu einer Laurentententwicklung verallgemeinert werden. Dann kann man

auch ∞ als Entwicklungspunkt nehmen, oder Funktionen mit Singularitäten entwickeln (mehr dazu in Funktiontheorie).

[92]:
```
series(atan(x), x, oo, 5)
```

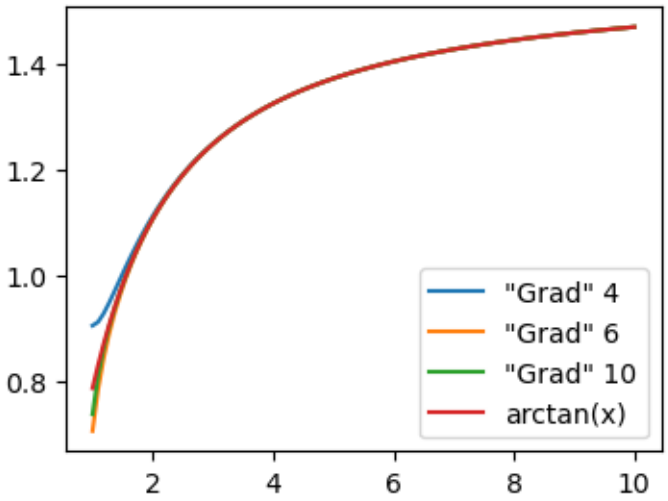
[92]: $$\frac{1}{3x^3} - \frac{1}{x} + \frac{\pi}{2} + O\left(\frac{1}{x^5}; x \to \infty\right)$$

[93]:
```
L = {}
for n in [4, 6, 10]:
    L[n] = series(atan(x), x, oo, n + 1).removeO()
L
```

[93]: $$\left\{4 : \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3},\ 6 : \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5},\ 10 : \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \frac{1}{9x^9}\right\}$$

[94]:
```
xn = np.linspace(1, 10, 100)
fig, ax = plt.subplots()
for n in L:
    ax.plot(xn, lambdify(x, L[n])(xn), label=f'"Grad" {n}')

ax.plot(xn, np.arctan(xn), label='arctan(x)')
plt.legend(loc=4);
```



Beispiel Rationale Approximation Wir suchen eine rationale Approximation an die Wurzelfunktion, so, dass jeweils der Anfang der Taylorreihe übereinstimmt.

[95]:
```
a = symbols('a:4')
X = symbols('X')
a
```

[95]:

$(a_0,\ a_1,\ a_2,\ a_3)$

[96]:
```
r = (a[0] + a[2] * X) / (1 + a[1] * X + a[3] * X**2)
tr = r.series(X, 0, 4).removeO()
tr
```

[96]:
$$X^3\left(a_0\left(-a_1^3+2a_1a_3\right)+a_2\left(a_1^2-a_3\right)\right)+X^2\left(a_0\left(a_1^2-a_3\right)-a_1a_2\right)+X\left(-a_0a_1+a_2\right)+a_0$$

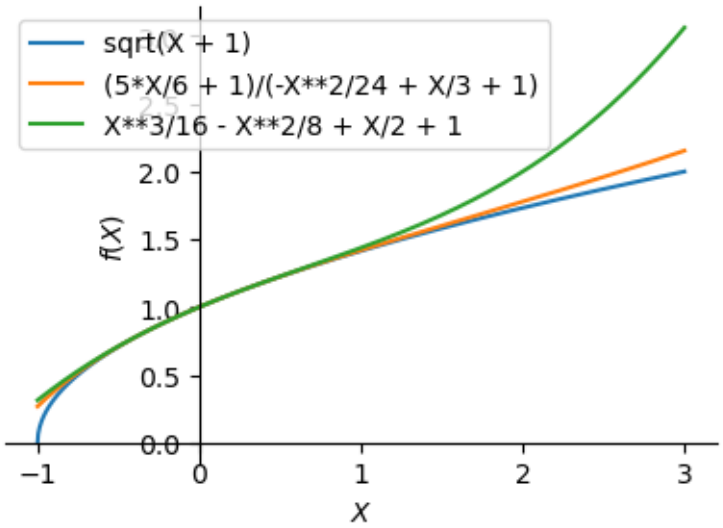
[97]:
```
f = (1 + X)**Rational(1, 2)
tf = f.series(X, 0, 4).removeO()
tf
```

[97]:
$$\frac{X^3}{16}-\frac{X^2}{8}+\frac{X}{2}+1$$

[98]:
```
# Gesucht sind a0, a1, a2 und a3 so, dass tf und tr übereinstimmen
sol = solve(Eq(tr, tf), a)
sol
```

[98]:
$$\left\{a_0:1,\ a_1:\frac{1}{3},\ a_2:\frac{5}{6},\ a_3:-\frac{1}{24}\right\}$$

[99]:
```
plot(f, r.subs(sol), tf, (X, -1, 3), legend=True);
```



1.6 Was ist eine `series`?

Wenn f eine Taylorreihe in dem Punkt hat, dann der Anfang der Taylorreihe.

[100]:
```
cos(x).series(x, 1, 9)
```

[100]:

$$\cos(1) - (x-1)\sin(1) - \frac{(x-1)^2\cos(1)}{2} + \frac{(x-1)^3\sin(1)}{6} + \frac{(x-1)^4\cos(1)}{24} - \frac{(x-1)^5\sin(1)}{120} - \frac{(x-1)^6\cos(1)}{720} + \frac{(x-1)^7\sin(1)}{5040} + \frac{(x-1)^8\cos(1)}{40320} + O\left((x-1)^9; x \to 1\right)$$

Wenn f eine Laurentreihe mit endlichem Nebenteil in ∞ hat, dann der Anfang des Haupteils. (mehr dazu in Funktionenentheorie)

Eine unendliche Reihe der Form

$$\sum_{n=-\infty}^{\infty} a_n(z-z_0)^n = \sum_{n=0}^{\infty} a_n(z-z_0)^n + \sum_{n=1}^{\infty} a_{-n}(z-z_0)^{-n}$$

heißt Laurentreihe mit Entwicklungspunkt z_0, und die Reihen

$$\sum_{n=0}^{\infty} a_n(z-z_0)^n \text{ bzw. } \sum_{n=1}^{\infty} a_{-n}(z-z_0)^{-n}$$

heißen der Nebenteil bzw. der Hauptteil der Laurentreihe. Eine Laurentreihe konvergiert im Punkt $z \in \mathbb{C}$, wenn sowohl Haupt- als auch Nebenteil konvergieren.

[101]:
```
g = (x**2 + 2 * x + 1)**2 / (x**2 + 2 * x - 1)
g.series(x, oo)
```

[101]: $$-\frac{48}{x^5} + \frac{20}{x^4} - \frac{8}{x^3} + \frac{4}{x^2} + 3 + 2x + x^2 + O\left(\frac{1}{x^6}; x \to \infty\right)$$

Sonst ist ‘series’ kein mathematisch sinnvoller Begriff

[102]:
```
h = exp(x) * g
h
```

[102]: $$\frac{(x^2+2x+1)^2 e^x}{x^2+2x-1}$$

[103]:
```
exp(x).series(x, oo)
```

[103]: e^x

[104]:
```
h.series(x, oo)
```

[104]: $$-\frac{48e^x}{x^5} + \frac{20e^x}{x^4} - \frac{8e^x}{x^3} + \frac{4e^x}{x^2} + 3e^x + 2xe^x + x^2e^x + O\left(\frac{e^x}{x^6}; x \to \infty\right)$$

[105]:
```
j = exp(x) * exp(1 / x)
j
```

[105]: $e^{\frac{1}{x}}e^x$

[106]:
```
j.series(x, oo)
```

[106]: $$\frac{e^x}{120x^5} + \frac{e^x}{24x^4} + \frac{e^x}{6x^3} + \frac{e^x}{2x^2} + \frac{e^x}{x} + e^x + O\left(\frac{e^x}{x^6}; x \to \infty\right)$$

[107]:
```
j1 = exp(x + 1 / x)
j1
```

[107]: $e^{x+\frac{1}{x}}$

[108]:
```
j1.series(x, oo)
```

[108]: $e^{x+\frac{1}{x}}$

[109]:
```
j*g
```

[109]: $$\frac{\left(x^2+2x+1\right)^2 e^{\frac{1}{x}} e^x}{x^2+2x-1}$$

[110]:
```
(j*g).series(x, oo)
```

[110]: $$-\frac{17531e^x}{560x^5}+\frac{10183e^x}{720x^4}-\frac{409e^x}{120x^3}+\frac{47e^x}{8x^2}+\frac{25e^x}{6x}+\frac{11e^x}{2}+3xe^x+x^2e^x+O\left(\frac{e^x}{x^6};x\to\infty\right)$$

[111]:
```
(j1*g).series(x, oo)
```

[111]: $$-\frac{48e^{x+\frac{1}{x}}}{x^5}+\frac{20e^{x+\frac{1}{x}}}{x^4}-\frac{8e^{x+\frac{1}{x}}}{x^3}+\frac{4e^{x+\frac{1}{x}}}{x^2}+3e^{x+\frac{1}{x}}+2xe^{x+\frac{1}{x}}+x^2e^{x+\frac{1}{x}}+O\left(\frac{e^x}{x^6};x\to\infty\right)$$