lektion7

December 3, 2025

Table of Contents

1 Lektion 7

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

1.1 Wiederholung

1.1.1 Sympy Ausdrücke (expressions) vs. Funktionen

[2]:
```
x = symbols('x')
f_ausdruck = x**2-sin(pi*x)
f_ausdruck
```

[2]:
$$x^2 - \sin(\pi x)$$

[3]:
```
y = symbols('y')
f_symfunk = Lambda(y, y**2-sin(pi*y))
```

```
f_symfunk
```

[3]: $(y \mapsto y^2 - \sin(\pi y))$

[4]:
```
f_ausdruck2 = f_symfunk(x)
f_ausdruck2
```

[4]: $x^2 - \sin(\pi x)$

Auswerten/Einsetzen

[5]:
```
f_symfunk(2)
```

[5]: 4

[6]:
```
#f_ausdruck(2)
```

[7]:
```
f_ausdruck.subs(x, 2)
```

[7]: 4

[8]:
```
f_symfunk.subs(2, 33)
```

[8]: $(y \mapsto y^{33} - \sin(\pi y))$

[9]:
```
f_pyfun = lambda zahl : zahl**2 - sin(pi*zahl)
f_pyfun
```

[9]:
```
<function __main__.<lambda>(zahl)>
```

[10]:
```
f_pyfun(2)
```

[10]: 4

[11]:
```
def f_pyfun2(argument):
    return argument**2 - sin(pi*argument)
```

[12]:
```
neuername = f_pyfun2
```

[13]:
```
f_pyfun2
```

[13]:
```
<function __main__.f_pyfun2(argument)>
```

[14]:
```
f_pyfun2.__name__
```

[14]:
```
'f_pyfun2'
```

[15]:
```
neuername(2)
```

[15]: 4

[16]:
```
f_pyfun2(2)
```

[16]: 4

[17]:
```
f_pyfun2 == neuername
```

[17]:
```
True
```

[18]:
```
f_ausdruck.diff(x)
```

[18]: $2x - \pi \cos(\pi x)$

[19]:
```
f_symfunk.diff(x)
```

[19]: 0

[20]:
```
f_pyfun(x).diff(x)
```

[20]: $2x - \pi \cos(\pi x)$

[21]:
```
f_pyfun2(y).diff(y)
```

[21]: $2y - \pi \cos(\pi y)$

universelle Funktionen (ufunc) numpy Funktionen

[22]:
```
f_ufunc = lambdify(x, f_symfunk(x))
f_ufunc
```

[22]:
```
<function _lambdifygenerated(x)>
```

[23]:
```
f_ufunc(2)
```

[23]: 4.0

[24]:
```
z = symbols('z')
f_ufunc2 = lambdify(z, f_symfunk(z))
```

[25]:
```
f_ufunc2(2)
```

[25]: 4.0

[26]:
```
f_ufunc(np.array([1, .5, 3])), f_ufunc2(np.array([1, .5, 3]))
```

[26]:
```
(array([ 1.  , -0.75,  9.  ]), array([ 1.  , -0.75,  9.  ]))
```

[27]:
```
np.random.rand(3,4)
```

[27]:
```
array([[0.95545576, 0.53324759, 0.68713506, 0.57441411],
       [0.57212433, 0.92327042, 0.36822776, 0.67261957],
```

```
       [0.16059674, 0.29230321, 0.15071369, 0.73875597]])
```

[28]:
```
f_ufunc(np.random.rand(3,4))
```

[28]:
```
array([[-0.5834788 ,  0.43698773,  0.18110552, -0.2069252 ],
       [-0.20742986, -0.77668106, -0.14450532, -0.71192996],
       [-0.62256167,  0.05214432, -0.69772648, -0.64443285]])
```

1.2 Polynome I

[29]:
```
p = x**2 - 1
q = 2*x + 2
display(p)
display(q)
```

$x^2 - 1$

$2x + 2$

[30]:
```
d, r = div(p, q)
display(d)
display(r)
```

$\frac{x}{2} - \frac{1}{2}$

0

[31]:
```
s = x + y**2
t = 1
degree(s, x), degree(s, y) , degree(t, x)
```

[31]: $(1,\ 2,\ 0)$

1.3 Polynome II

[32]:
```
p = Poly(x**25 -1)
p
```

[32]: $\mathrm{Poly}\left(x^{25} - 1, x, domain = \mathbb{Z}\right)$

[33]:
```
q = Poly([2, 0, -2], x, domain=ZZ) # Polynom aus der Liste der Koeffizienten
q
```

[33]: $\mathrm{Poly}\left(2x^2 - 2, x, domain = \mathbb{Z}\right)$

[34]:
```
qq = Poly([2, 0, -2], x, domain=QQ)
qq
```

[34]: $\mathrm{Poly}\left(2x^2 - 2, x, domain = \mathbb{Q}\right)$

[35]:
```
Poly({2: 2, 0: -2}, x, domain=ZZ)
```

[35]: $\text{Poly}\left(2x^2 - 2, x, domain = \mathbb{Z}\right)$

[36]:
```
s.as_poly()
```

[36]: $\text{Poly}\left(x + y^2, x, y, domain = \mathbb{Z}\right)$

[37]:
```
degree(s.as_poly(), y) # per Default wird der Grad bzgl. x ausgegeben
```

[37]: 2

1.3.1 Polynomdivision

[38]:
```
p = Poly(2*x**4 - 5*x**3 + 3*x**2 + x - 1, domain=ZZ)
q = Poly(2*x-2)
```

[39]:
```
p, q
```

[39]: $\left(\text{Poly}\left(2x^4 - 5x^3 + 3x^2 + x - 1, x, domain = \mathbb{Z}\right),\ \text{Poly}\left(2x - 2, x, domain = \mathbb{Z}\right)\right)$

[40]:
```
div(p, q, domain=ZZ)
```

[40]: $\left(\text{Poly}\left(x^3, x, domain = \mathbb{Z}\right),\ \text{Poly}\left(-3x^3 + 3x^2 + x - 1, x, domain = \mathbb{Z}\right)\right)$

[41]:
```
quo(p, q, domain = ZZ)
```

[41]: $\text{Poly}\left(x^3 - 2x^2, x, domain = \mathbb{Z}\right)$

[42]:
```
rem(p, q, domain = ZZ)
```

[42]: $\text{Poly}\left(x^3 - x^2 + x - 1, x, domain = \mathbb{Z}\right)$

[43]:
```
q * quo(p, q, domain= ZZ) + rem(p, q, domain=ZZ) - p
```

[43]: $\text{Poly}\left(0, x, domain = \mathbb{Z}\right)$

[44]:
```
p = Poly(p.as_expr(), x,  domain=QQ)
p
```

[44]: $\text{Poly}\left(2x^4 - 5x^3 + 3x^2 + x - 1, x, domain = \mathbb{Q}\right)$

[45]:
```
q = poly(q.as_expr(), x, domain = QQ)
q
```

[45]: $\text{Poly}\left(2x - 2, x, domain = \mathbb{Q}\right)$

[46]:
```
s,t = div(p, q, domain = QQ)
s, t
```

[46]:
$$\left(\text{Poly}\left(x^3 - \frac{3}{2}x^2 + \frac{1}{2}, x, domain = \mathbb{Q}\right),\ \text{Poly}\left(0, x, domain = \mathbb{Q}\right)\right)$$

[47]:
```
s*q- p
```

[47]:
$$\text{Poly}\left(0, x, domain = \mathbb{Q}\right)$$

1.3.2 Polynomgrad

[48]:
```
p = poly(2*x**3 + 3*x*y)
degree(p)
```

[48]:
$$3$$

[49]:
```
degree(p, x)
```

[49]:
$$3$$

[50]:
```
degree(p, y)
```

[50]:
$$1$$

[51]:
```
r = poly(0, x, domain=QQ)
r
```

[51]:
$$\text{Poly}\left(0, x, domain = \mathbb{Q}\right)$$

[52]:
```
degree(r) # Achtung
```

[52]:
$$-\infty$$

1.3.3 Koeffizienten von Polynomen

[53]:
```
q = 3*x**2 + x - 1
```

[54]:
```
q.coeff(x, 2)
```

[54]:
$$3$$

[55]:
```
p = prod([x-j for j in range(5)])
p.expand()
```

[55]:
$$x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$$

[56]:
```
p.expand().coeff(x, 4)
```

[56]:
$$-10$$

Koeffizienten eines Polynoms p als Liste c, sodass

$$p(x) = \sum_{i=0}^{n} c[i]x^i$$

[57]:
```
p.expand()
```

[57]: $x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$

[58]:
```
n = 5
c = [p.expand().coeff(x, j) for j in range(degree(p, x) + 1)]
c
```

[58]: $[0,\ 24,\ -50,\ 35,\ -10,\ 1]$

Koeffizienten eines Polynoms p als Dictionary d,sodass

$$p(x) = \sum_{k \in d} d[k] x^k$$

$k \in d$ sind die keys des Dictionarys

[59]:
```
p.expand()
```

[59]: $x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$

[60]:
```
d = {
    i: p.expand().coeff(x, i)
    for i in range(degree(p, x) + 1) if p.expand().coeff(x, i)
}
d
```

[60]: $\{1 : 24,\ 2 : -50,\ 3 : 35,\ 4 : -10,\ 5 : 1\}$

[61]:
```
p = Poly(p)
p
```

[61]: $\text{Poly}\left(x^5 - 10x^4 + 35x^3 - 50x^2 + 24x, x, domain = \mathbb{Z}\right)$

[62]:
```
c.reverse() # dreht die Liste um
pc = Poly(c, x)
pc
```

[62]: $\text{Poly}\left(x^5 - 10x^4 + 35x^3 - 50x^2 + 24x, x, domain = \mathbb{Z}\right)$

[63]:
```
pd = Poly(d,x)
pd
```

[63]: $\text{Poly}\left(x^5 - 10x^4 + 35x^3 - 50x^2 + 24x, x, domain = \mathbb{Z}\right)$

[64]:
```
#p.coeff(x, 4)
```

[65]:
```
p.as_expr().coeff(x, 4)
```

[65]: -10

[66]:
```
p.coeff_monomial(x**4)
```

[66]: -10

[67]:
```
p.all_coeffs()
```

[67]: $[1,\ -10,\ 35,\ -50,\ 24,\ 0]$

[68]:
```
p.coeffs() # das ist nicht so hilfreich, da die 0 Koeffizienten fehlen
```

[68]: $[1,\ -10,\ 35,\ -50,\ 24]$

1.3.4 Wurzeln/Nullstellen von Polynomen

[69]:
```
a, b, c, d = symbols('a b c d')
```

[70]:
```
p = x**4 - x**2
roots(p)  # Wurzel als Dictionary key: Wurzel value: Vielfachheit
```

[70]: $\{-1 : 1,\ 0 : 2,\ 1 : 1\}$

[71]:
```
roots(a * x**2 + b * x + c, x)
```

[71]: $$\left\{-\frac{b}{2a} - \frac{\sqrt{-4ac+b^2}}{2a} : 1,\ -\frac{b}{2a} + \frac{\sqrt{-4ac+b^2}}{2a} : 1\right\}$$

[72]:
```
p = Poly(a * x**3 + b * x**2 + c * x + d, x)
roots(p, x)
```

[72]: $$\left\{-\frac{-\frac{3c}{a}+\frac{b^2}{a^2}}{3\sqrt[3]{\frac{\sqrt{-4\left(-\frac{3c}{a}+\frac{b^2}{a^2}\right)^3+\left(\frac{27d}{a}-\frac{9bc}{a^2}+\frac{2b^3}{a^3}\right)^2}}{2}+\frac{27d}{2a}-\frac{9bc}{2a^2}+\frac{b^3}{a^3}}} - \frac{\sqrt[3]{\frac{\sqrt{-4\left(-\frac{3c}{a}+\frac{b^2}{a^2}\right)^3+\left(\frac{27d}{a}-\frac{9bc}{a^2}+\frac{2b^3}{a^3}\right)^2}}{2}+\frac{27d}{2a}-\frac{9bc}{2a^2}+\frac{b^3}{a^3}}}{3} - \frac{b}{3a} :$$

[73]:
```
p = Poly([1, -5, 4, 4 ,3, 9], x)
p
```

[73]: $\mathrm{Poly}\left(x^5 - 5x^4 + 4x^3 + 4x^2 + 3x + 9, x, domain = \mathbb{Z}\right)$

[74]:
```
roots(p, x) # Glück gehabt
```

[74]: $\{-1 : 1,\ 3 : 2,\ -i : 1,\ i : 1\}$

[75]:
```
real_roots(p, x)
```

[75]: $[-1,\ 3,\ 3]$

[76]:
```
p = Poly([2048, 0, -6656, 0, 8320, -224, -4768, 336, 1120, -126, -56, 7, 0], x)
p
```

[76]:
$$\mathrm{Poly}\left(2048x^{12} - 6656x^{10} + 8320x^{8} - 224x^{7} - 4768x^{6} + 336x^{5} + 1120x^{4} - 126x^{3} - 56x^{2} + 7x, x, domain = \mathbb{Z}\right)$$

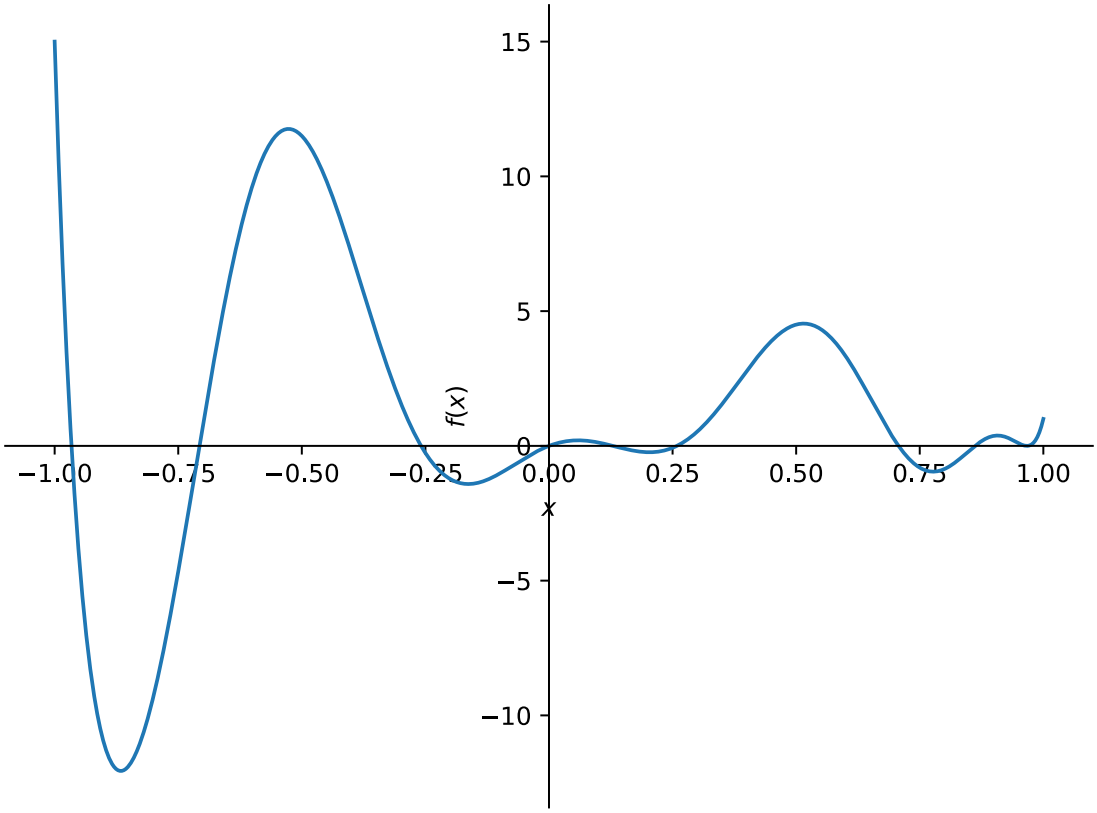
[77]:
```
roots(p) # das sind nicht alle
```

[77]:
$$\left\{0 : 1,\ -\frac{\sqrt{2}}{2} : 1,\ \frac{\sqrt{2}}{2} : 1,\ -\sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}} : 1,\ \sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}} : 1,\ -\sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}} : 1,\ \sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}} : 1\right\}$$

[78]:
```
len(roots(p))
```

[78]:
7

[79]:
```
plot(p, (x, -1, 1))
```



[79]:
```
<sympy.plotting.backends.matplotlibbackend.matplotlib.MatplotlibBackend at
0x7ff137d740e0>
```

[80]:
```
ar = p.all_roots()
ar
```

[80]:
$$\left[\text{CRootOf}\left(16x^4 - 16x^2 + 1, 0\right),\ -\frac{\sqrt{2}}{2},\ \text{CRootOf}\left(16x^4 - 16x^2 + 1, 1\right),\ 0,\ \text{CRootOf}\left(64x^5 - 112x^3 + 56x - 7, 0\right),\ \text{C}\right.$$

[113]:
```
nr = nroots(p)  # wir können sie numerisch ausrechnen
for n, r in enumerate(nr):
    print(f'{n+1}.\tWurzel: {r}')
```

```
1.      Wurzel: -0.965925826289068
2.      Wurzel: -0.707106781186548
3.      Wurzel: -0.258819045102521
4.      Wurzel: 0
5.      Wurzel: 0.129280137861210
6.      Wurzel: 0.258819045102521
7.      Wurzel: 0.707106781186548
8.      Wurzel: 0.861683187713787
9.      Wurzel: 0.965925826289068
10.     Wurzel: 0.967347488153037
11.     Wurzel: -0.979155406864017 - 0.2371318772878*I
12.     Wurzel: -0.979155406864017 + 0.2371318772878*I
```

1.4 Lösen von Gleichungen (solveset)

$$(x-1)^2 = 4 - x$$

[81]:
```
gl = Eq((x-1)**2, 4-x)
gl
```

[81]:
$$(x-1)^2 = 4 - x$$

[82]:
```
gl.lhs
```

[82]:
$$(x-1)^2$$

[83]:
```
gl.rhs
```

[83]:
$$4 - x$$

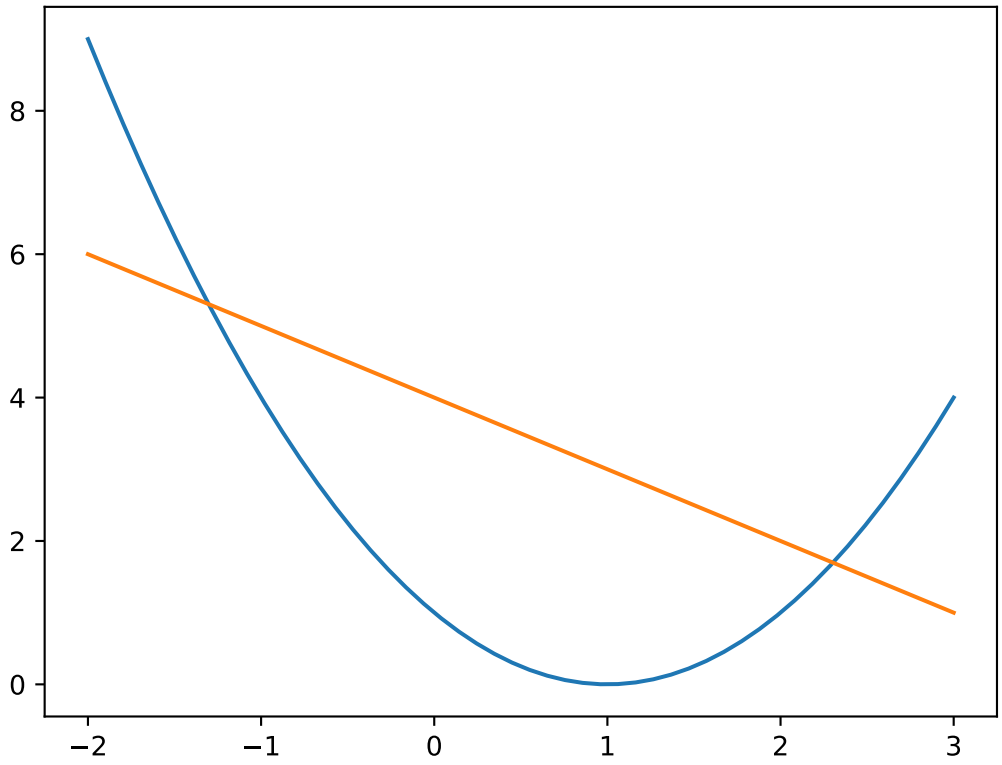
[84]:
```
Lsgn = solveset(gl, x)
type(Lsgn)
```

[84]:
```
sympy.sets.sets.FiniteSet
```

[85]:
```
Lsgn
```

[85]:
$$\left\{\frac{1}{2} - \frac{\sqrt{13}}{2}, \frac{1}{2} + \frac{\sqrt{13}}{2}\right\}$$

[86]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-2, 3)
ax.plot(xn, lambdify(x, gl.lhs)(xn))
ax.plot(xn, lambdify(x, gl.rhs)(xn))
plt.show()
```



[87]:
```
# Test durch einsetzen
for lsg in Lsgn:
    display(gl.subs(x, lsg))
```

True

True

[88]:
```
solveset((x-1)**2 - 4+x, x)  # loest (x-1)**2-4+x = 0
```

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

[89]:
```
ar = solveset(p.as_expr(), x)
ar
```

[89]:

$$\left\{0, -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \mathrm{CRootOf}\left(16x^4 - 16x^2 + 1, 0\right), \mathrm{CRootOf}\left(16x^4 - 16x^2 + 1, 1\right), \mathrm{CRootOf}\left(16x^4 - 16x^2 + 1, 2\right), \mathrm{CRoot}\right.$$

[90]:
```
ars = FiniteSet() # Menge aller Wurzeln
for ar_ in ar:
    if isinstance(ar_, CRootOf):
        if ar_.args[1] == 0:
            ars |= solveset(ar_.args[0], x)
            # in place union äquivalent zu "ars = Union(ars,solveset(...))"
    else:
        ars |= {ar_}
ars
```

[90]:
$$\left\{0, -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}}, \sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}}, -\sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}}, \sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}}, \mathrm{CRootOf}\left(64x^5 - 112x^3 + 56x - 7, 0\right), \mathrm{CRootOf}\left(6\right.\right.$$

[91]:
```
arr = []
for ar_ in ar:
    if isinstance(ar_, ComplexRootOf):
        if ar_.args[1] == 0:
            rr = solveset(ar_.args[0])
            for r_ in rr:
                arr.append(r_)
    else:
        arr.append(ar_)
arr
```

[91]:
$$\left[0, -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}}, \sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}}, -\sqrt{\frac{1}{2} - \frac{\sqrt{3}}{4}}, -\sqrt{\frac{\sqrt{3}}{4} + \frac{1}{2}}, \mathrm{CRootOf}\left(64x^5 - 112x^3 + 56x - 7, 0\right), \mathrm{CRoo}\right.$$