lektion12

January 23, 2020

Table of Contents

1 Lektion 12

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

1.1 Daten exportieren und importieren

[2]:
```
H  = Matrix(5, 5, [Rational(1, j+i+1) for i in range(5) for j in range(5)])
H
```

[2]:
$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

[3]:
```
str(H)
```

[3]:
```
'Matrix([[1, 1/2, 1/3, 1/4, 1/5], [1/2, 1/3, 1/4, 1/5, 1/6], [1/3, 1/4, 1/5,
1/6, 1/7], [1/4, 1/5, 1/6, 1/7, 1/8], [1/5, 1/6, 1/7, 1/8, 1/9]])'
```

1.1.1 Pythoncode

[4]:
```
srepr(H) # erzeugt ausführbaren Pythoncode
```

[4]:
```
'MutableDenseMatrix([[Integer(1), Rational(1, 2), Rational(1, 3), Rational(1,
4), Rational(1, 5)], [Rational(1, 2), Rational(1, 3), Rational(1, 4),
Rational(1, 5), Rational(1, 6)], [Rational(1, 3), Rational(1, 4), Rational(1,
5), Rational(1, 6), Rational(1, 7)], [Rational(1, 4), Rational(1, 5),
Rational(1, 6), Rational(1, 7), Rational(1, 8)], [Rational(1, 5), Rational(1,
6), Rational(1, 7), Rational(1, 8), Rational(1, 9)]])'
```

1.1.2 Datei öffnen und in Dateien schreiben

[5]:
```
with open('output.txt','w') as f:  # w: write
    f.write(srepr(H) + '\n')
```

[ ]:
```

```

[6]:
```
%less output.txt
```

Latex in Dateien schreiben

[7]:
```
x,y = symbols('x y')
f = log(x)*sqrt(x**2/3+y**2)
df = f.diff(x, y).simplify()
df
f = Function('f')
lhs = f(x, y).diff(x, y)
eq = Eq(lhs, df)
```

```
eq
```

[7]:
$$\frac{\partial^2}{\partial y \partial x} f(x,y) = \frac{\sqrt{3} y \left(-x^2 \log\left(x\right) + x^2 + 3y^2\right)}{x \left(x^2 + 3y^2\right)^{\frac{3}{2}}}$$

[8]:
```
with open('math.tex', 'w') as f:
    f.write(latex(eq) + '\n')
```

[9]:
```
%less math.tex
```

[10]:
```
z = symbols('z:5')
V = Matrix(5, 5, [z[j]**i for j in range(5) for i in range(5)])
V
```

[10]:
$$\begin{bmatrix} 1 & z_0 & z_0^2 & z_0^3 & z_0^4 \\ 1 & z_1 & z_1^2 & z_1^3 & z_1^4 \\ 1 & z_2 & z_2^2 & z_2^3 & z_2^4 \\ 1 & z_3 & z_3^2 & z_3^3 & z_3^4 \\ 1 & z_4 & z_4^2 & z_4^3 & z_4^4 \end{bmatrix}$$

1.1.3 weiteren Text zu Dateien hinzufügen

[11]:
```
with open('output.txt', 'a') as f:  # 'a' append / anhängen
    f.write(srepr(2*H) + '\n')
    f.write(srepr(V))
```

[12]:
```
with open('math.tex', 'a') as f:
    f.write(latex(V) + '\n')
```

1.1.4 Inhalt von Dateien auslesen

[13]:
```
with open('output.txt') as f:  # read (default)
    HH = S(f.readline())
    H2 = S(f.readline())
    VV = S(f.readline())
```

[14]:
```
HH, H2, VV
```

[14]:
$$\left(\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}, \begin{bmatrix} 2 & 1 & \frac{2}{3} & \frac{1}{2} & \frac{2}{5} \\ 1 & \frac{2}{3} & \frac{1}{2} & \frac{2}{5} & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{2} & \frac{2}{5} & \frac{1}{3} & \frac{2}{7} \\ \frac{1}{2} & \frac{2}{5} & \frac{1}{3} & \frac{2}{7} & \frac{1}{4} \\ \frac{2}{5} & \frac{1}{3} & \frac{2}{7} & \frac{1}{4} & \frac{2}{9} \end{bmatrix}, \begin{bmatrix} 1 & z_0 & z_0^2 & z_0^3 & z_0^4 \\ 1 & z_1 & z_1^2 & z_1^3 & z_1^4 \\ 1 & z_2 & z_2^2 & z_2^3 & z_2^4 \\ 1 & z_3 & z_3^2 & z_3^3 & z_3^4 \\ 1 & z_4 & z_4^2 & z_4^3 & z_4^4 \end{bmatrix} \right)$$

[15]:
```
%less output.txt
```

[16]:
```
# '%' IPython Magie (shell Befehl 'less')
# Ändere output.txt in Editor
```

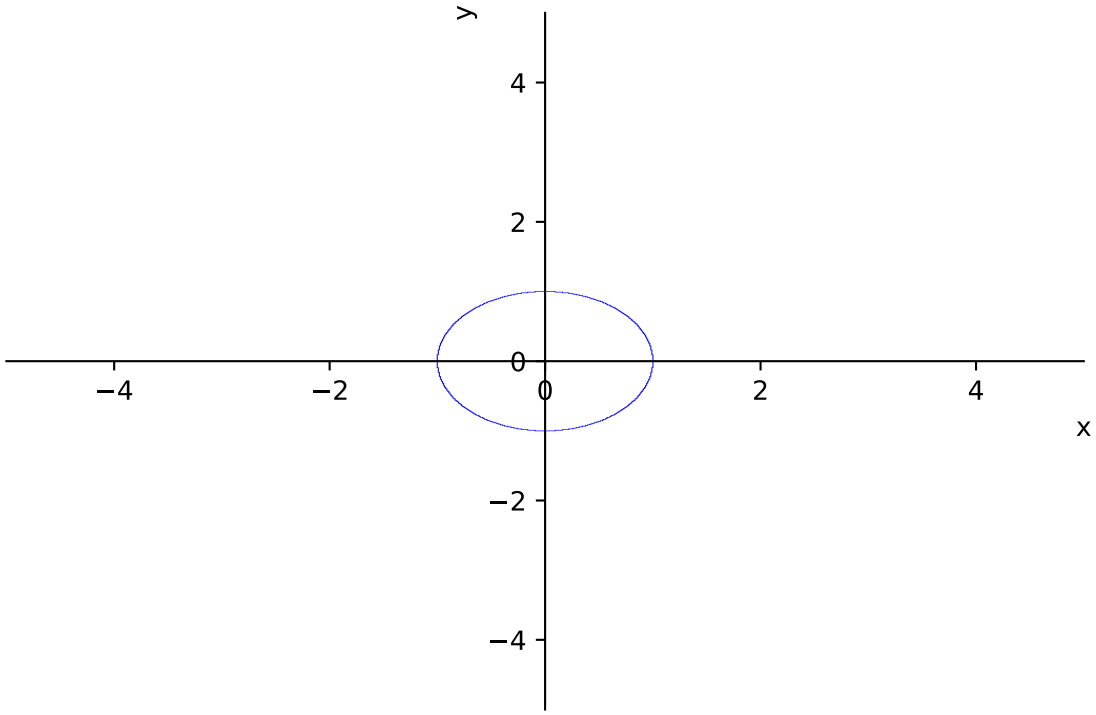
[17]:
```
with open('output.txt') as f:  # read (default)
    for zeile in f:
        print(zeile)
    else:
        print('Ende erreicht')
```

```
MutableDenseMatrix([[Integer(1), Rational(1, 2), Rational(1, 3), Rational(1, 4),
Rational(1, 5)], [Rational(1, 2), Rational(1, 3), Rational(1, 4), Rational(1,
5), Rational(1, 6)], [Rational(1, 3), Rational(1, 4), Rational(1, 5),
Rational(1, 6), Rational(1, 7)], [Rational(1, 4), Rational(1, 5), Rational(1,
6), Rational(1, 7), Rational(1, 8)], [Rational(1, 5), Rational(1, 6),
Rational(1, 7), Rational(1, 8), Rational(1, 9)]])

MutableDenseMatrix([[Integer(2), Integer(1), Rational(2, 3), Rational(1, 2),
Rational(2, 5)], [Integer(1), Rational(2, 3), Rational(1, 2), Rational(2, 5),
Rational(1, 3)], [Rational(2, 3), Rational(1, 2), Rational(2, 5), Rational(1,
3), Rational(2, 7)], [Rational(1, 2), Rational(2, 5), Rational(1, 3),
Rational(2, 7), Rational(1, 4)], [Rational(2, 5), Rational(1, 3), Rational(2,
7), Rational(1, 4), Rational(2, 9)]])

MutableDenseMatrix([[Integer(1), Symbol('z0'), Pow(Symbol('z0'), Integer(2)),
Pow(Symbol('z0'), Integer(3)), Pow(Symbol('z0'), Integer(4))], [Integer(1),
Symbol('z1'), Pow(Symbol('z1'), Integer(2)), Pow(Symbol('z1'), Integer(3)),
Pow(Symbol('z1'), Integer(4))], [Integer(1), Symbol('z2'), Pow(Symbol('z2'),
Integer(2)), Pow(Symbol('z2'), Integer(3)), Pow(Symbol('z2'), Integer(4))],
[Integer(1), Symbol('z3'), Pow(Symbol('z3'), Integer(2)), Pow(Symbol('z3'),
Integer(3)), Pow(Symbol('z3'), Integer(4))], [Integer(1), Symbol('z4'),
Pow(Symbol('z4'), Integer(2)), Pow(Symbol('z4'), Integer(3)), Pow(Symbol('z4'),
Integer(4))]])
Ende erreicht
```

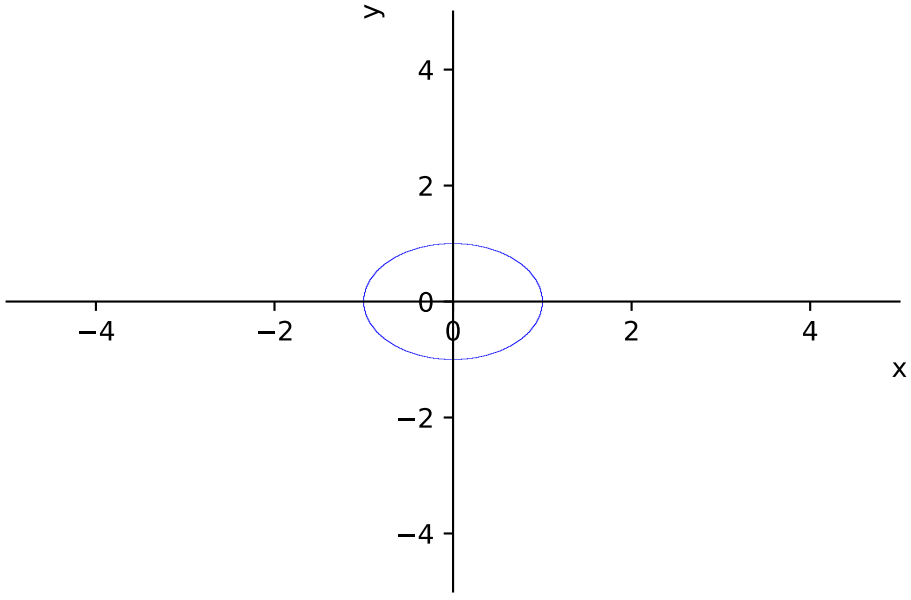
[18]:
```
%less output.txt
```

1.2 Bilder exportieren / speichern

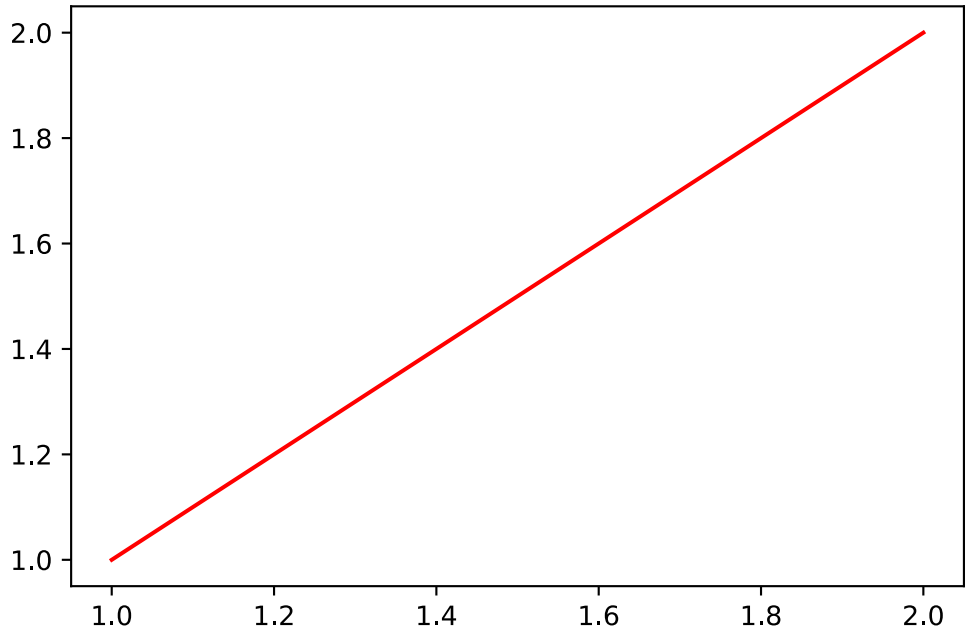
[19]:
```
p = plot_implicit(x**2+y**2-1)
```



[20]:
```
p.save('kreis.png');
```



[21]:
```
fig = plt.figure()
plt.plot([1,2], [1,2], 'r');
```



[22]:
```
fig.savefig('rotelinie.png', format = 'png')
fig.savefig('rotelinie.pdf', format = 'pdf')
fig.savefig('rotelinie.eps', format = 'eps')
fig.savefig('rotelinie.svg', format = 'svg')
```

[23]:
```
fig.canvas.get_supported_filetypes()
```

[23]:
```
{'ps': 'Postscript',
 'eps': 'Encapsulated Postscript',
 'pdf': 'Portable Document Format',
 'pgf': 'PGF code for LaTeX',
 'png': 'Portable Network Graphics',
 'raw': 'Raw RGBA bitmap',
 'rgba': 'Raw RGBA bitmap',
 'svg': 'Scalable Vector Graphics',
 'svgz': 'Scalable Vector Graphics',
 'jpg': 'Joint Photographic Experts Group',
 'jpeg': 'Joint Photographic Experts Group',
 'tif': 'Tagged Image File Format',
 'tiff': 'Tagged Image File Format'}
```

1.3 Differentialgleichungen

vgl. Analysis II

1.3.1 erstes Beispiel

Gesucht ist eine differenzierbare Funktion $y : \mathbb{R} \to \mathbb{R}$ mit

$$\dot{y}(t) = y(t).$$

Hierbei ist $\dot{y}(t) = \frac{d}{dt}y(t)$ die Ableitung von y

[24]:
```
y = Function('y')
t, tau = symbols('t tau', real = True)
dgl = Eq(y(t).diff(t)-y(t), 0)
dgl
```

[24]:
$$-y(t) + \frac{d}{dt}y(t) = 0$$

[25]:
```
sol = dsolve(dgl, y(t))
sol
```

[25]:
$$y(t) = C_1 e^t$$

[26]:
```
C1 = sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop()
C1
```

[26]:
$$C_1$$

[27]:
```
solve(sol, C1).pop()
```

[27]:
$$y(t)e^{-t}$$

[28]:
```
C1 = solve(sol, C1).pop().subs(t, 0)
C1
```

[28]:
$$y(0)$$

[29]:
```
checkodesol(dgl, sol)
```

[29]:
```
(True, 0)
```

1.3.2 inhomogene lineare DGL

$$\dot{u}(t) = u(t) + \sin(t)$$

[30]:
```
u = Function('u')
t = symbols('t', real = True)
dgl = Eq(u(t).diff(t)-u(t)-sin(t))
sol = dsolve(dgl,u(t))
sol
```

```
/local/schaedle/miniconda/lib/python3.7/site-
packages/sympy/core/relational.py:470: SymPyDeprecationWarning:

Eq(expr) with rhs default to 0 has been deprecated since SymPy 1.5.
Use Eq(expr, 0) instead. See
https://github.com/sympy/sympy/issues/16587 for more info.

  deprecated_since_version="1.5"
```

[30]:
$$u(t) = \left(C_1 - \frac{e^{-t}\sin(t)}{2} - \frac{e^{-t}\cos(t)}{2}\right)e^{t}$$

[31]:
```
C1 = sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop()
C1 = solve(sol,C1).pop().subs(t, 0)
C1
```

[31]:
$$u(0) + \frac{1}{2}$$

[32]:
```
w = (sol.subs(sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop(), C1)).rhs
w
```

[32]:
$$\left(u(0) + \frac{1}{2} - \frac{e^{-t}\sin(t)}{2} - \frac{e^{-t}\cos(t)}{2}\right)e^{t}$$

1.3.3 Variation der Konstanten Formel

Die Lösung von

$$\dot{u}(t) = au(t) + g(t, u(t)), \quad u(0) = u_0$$

ist

$$u(t) = \mathrm{e}^{at}u_0 + \int_0^t \mathrm{e}^{a(t-\tau)}g(\tau, u(\tau))d\tau$$

[33]:
```
v = u(0)*exp(t) + integrate(exp(t-tau)*(sin(tau)),(tau,0,t))
v
```

[33]:
$$u(0)e^{t} + \frac{e^{t}}{2} - \frac{\sin(t)}{2} - \frac{\cos(t)}{2}$$

[34]:
```
simplify(v-w)
```

[34]:
$$0$$

1.3.4 Lösung mit einem Reihenansatz

$$\dot{y}(t) = y(t), \quad y(0) = 1$$

mit einem Reihenansatz

[35]:
```
y0, t, C = symbols('y0 t C')
a = symbols('a:8')
y = Function('y')
```

[36]:
```
dgl = Eq(y(t).diff(t),y(t))
dgl
```

[36]:
$$\frac{d}{dt}y(t) = y(t)$$

[37]:
```
ys = sum([a[i]*t**i for i in range(8)])
ys = ys.subs(a[0],1)  # y(0) = 1
ys
```

[37]:
$$a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 + a_7 t^7 + 1$$

[38]:
```
gl = dgl.subs(y(t),ys).doit()
gl
```

[38]:
$$a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 + 6a_6 t^5 + 7a_7 t^6 = a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 + a_7 t^7 + 1$$

[39]:
```
gl.coeff(t)
```

[39]:
$$0$$

[40]:
```
gl1 = (gl.lhs - gl.rhs).expand()
gl1
```

[40]:
$$-a_1 t + a_1 - a_2 t^2 + 2a_2 t - a_3 t^3 + 3a_3 t^2 - a_4 t^4 + 4a_4 t^3 - a_5 t^5 + 5a_5 t^4 - a_6 t^6 + 6a_6 t^5 - a_7 t^7 + 7a_7 t^6 - 1$$

[41]:
```
gls = gl1.as_poly(t).all_coeffs()
gls[1:]
```

[41]:
$$[-a_6 + 7a_7,\ -a_5 + 6a_6,\ -a_4 + 5a_5,\ -a_3 + 4a_4,\ -a_2 + 3a_3,\ -a_1 + 2a_2,\ a_1 - 1]$$

[42]:
```
ac = solve(gls[1:])
ac
```

[42]:
$$\left\{a_1 : 1,\ a_2 : \frac{1}{2},\ a_3 : \frac{1}{6},\ a_4 : \frac{1}{24},\ a_5 : \frac{1}{120},\ a_6 : \frac{1}{720},\ a_7 : \frac{1}{5040}\right\}$$

[43]:
```
ac[a[0]] = 1  # wir raten das Bildungsgesetz der a's
acc = [ac[j] for j in a]
acc
```

[43]:

$$\left[1,\ 1,\ \frac{1}{2},\ \frac{1}{6},\ \frac{1}{24},\ \frac{1}{120},\ \frac{1}{720},\ \frac{1}{5040}\right]$$

[44]:
```
[ acc[j]/acc[j+1] for j in range(7)]
```

[44]:
$$[1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7]$$

[45]:
```
[ acc[j] - 1/factorial(j) for j in range(8)]
```

[45]:
$$[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$$

[46]:
```
n = symbols('n')
yr = Sum(t**n/factorial(n) ,(n,0,oo))
yr
```

[46]:
$$\sum_{n=0}^{\infty} \frac{t^n}{n!}$$

[47]:
```
yr.doit()
```

[47]:
$$e^t$$

1.3.5 Logistische Gleichung

einfaches Modell zur Beschreibung von Wachstum in einem Habitat mit beschränkten Resourcen

$$\dot{y}(t) = (1 - y(t))y(t)$$

[48]:
```
y = Function('y')
t = symbols('t', real = True)
dgl = Eq(y(t).diff(t)-(1-y(t))*y(t))
sol = dsolve(dgl,y(t))
sol
```

```
/local/schaedle/miniconda/lib/python3.7/site-
packages/sympy/core/relational.py:470: SymPyDeprecationWarning:

Eq(expr) with rhs default to 0 has been deprecated since SymPy 1.5.
Use Eq(expr, 0) instead. See
https://github.com/sympy/sympy/issues/16587 for more info.

  deprecated_since_version="1.5"
```

[48]:
$$y(t) = \frac{1}{C_1 e^{-t} + 1}$$

[49]:
```
K1 = sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop()
K1
```

[49]: C_1

[50]:
```
C1 = solve(sol,K1)
C1
```

[50]: $\left[-e^t + \frac{e^t}{y(t)}\right]$

[51]:
```
C1 = solve(sol,K1).pop().subs(t, 0)
C1
```

[51]: $-1 + \frac{1}{y(0)}$

[52]:
```
sol.subs(sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop(),C1)
```

[52]: $y(t) = \frac{1}{\left(-1 + \frac{1}{y(0)}\right) e^{-t} + 1}$

1.3.6 Zweite Ordnung DGL

$$\ddot{y}(t) = -2\dot{y}(t) + y(t) - 100\cos(t)$$

[53]:
```
dgl2 = Eq(y(t).diff(t,2) + 2*y(t).diff(t) -y(t)+100*cos(t),0)
dgl2
```

[53]: $-y(t) + 100\cos(t) + 2\frac{d}{dt}y(t) + \frac{d^2}{dt^2}y(t) = 0$

[54]:
```
sol = dsolve(dgl2,y(t))
sol
```

[54]: $y(t) = C_1 e^{t\left(-1+\sqrt{2}\right)} + C_2 e^{t\left(-\sqrt{2}-1\right)} - 25\sin(t) + 25\cos(t)$

Anfangswerte

$$y(0) = 10, \quad \dot{y}(0) = 0$$

[55]:
```
[Eq(sol.rhs.subs(t, 0), 10), Eq(sol.rhs.diff(t).subs(t, 0), 0)]
```

[55]: $\left[C_1 + C_2 + 25 = 10, \quad C_1\left(-1+\sqrt{2}\right) + C_2\left(-\sqrt{2}-1\right) - 25 = 0\right]$

[56]:
```
CS = [c_ for c_ in sol.atoms(Symbol).difference(dgl2.atoms(Symbol))]
CS
```

[56]:

$[C_1,\ C_2]$

[57]:
```
Constants = solve((sol.rhs.subs(t,0)-10, sol.rhs.diff(t).subs(t,0)),CS)
Constants
```

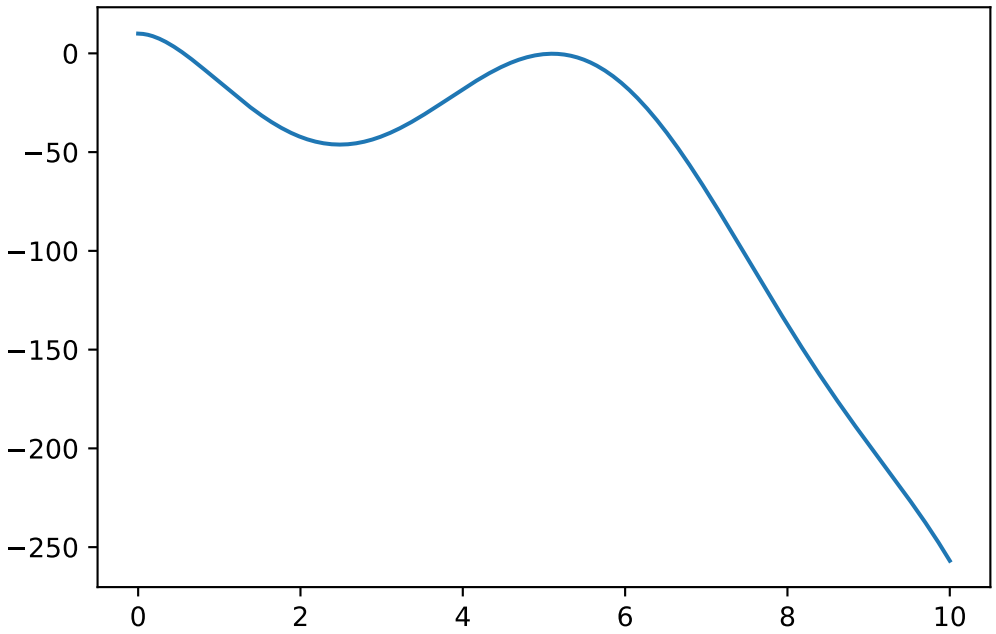
[57]:
$$\left\{C_1 : -\frac{15}{2} + \frac{5\sqrt{2}}{2},\ C_2 : -\frac{15}{2} - \frac{5\sqrt{2}}{2}\right\}$$

[58]:
```
sol = sol.subs(Constants)
sol
```

[58]:
$$y(t) = \left(-\frac{15}{2} + \frac{5\sqrt{2}}{2}\right) e^{t\left(-1+\sqrt{2}\right)} + \left(-\frac{15}{2} - \frac{5\sqrt{2}}{2}\right) e^{t\left(-\sqrt{2}-1\right)} - 25\sin(t) + 25\cos(t)$$

[59]:
```
tn = np.linspace(0,10,500)
yn = lambdify(t,sol.rhs)
fig = plt.figure()
plt.plot(tn,yn(tn))
```

[59]:
```
[<matplotlib.lines.Line2D at 0x7f9a7977f5d0>]
```



1.3.7 Systeme von DGLen

Matrixexponentialfunktion

[60]:
```
t = symbols('t',real=True)
```

[61]:
```
A = Matrix(3,3,[2,1,0,0,2,1,0,0,2])
A
```

[61]:
$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

[62]:
```
(t*A).exp()
```

[62]:
$$\begin{bmatrix} e^{2t} & te^{2t} & \frac{t^2e^{2t}}{2} \\ 0 & e^{2t} & te^{2t} \\ 0 & 0 & e^{2t} \end{bmatrix}$$

[63]:
```
u0 = Matrix(3,1,[1,2,3])
u0
```

[63]:
$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

[64]:
```
u = (t*A).exp()*u0
u
```

[64]:
$$\begin{bmatrix} \frac{3t^2e^{2t}}{2} + 2te^{2t} + e^{2t} \\ 3te^{2t} + 2e^{2t} \\ 3e^{2t} \end{bmatrix}$$

[65]:
```
diff(u,t) - A*u
```

[65]:
$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Gekoppeltes Pendel (Kleinwinkelnäherung)

$$\ddot{y}(t) = w(t) - y(t) + \cos(2t) \tag{1}$$
$$\ddot{w}(t) = y(t) - w(t) \tag{2}$$

Äquivalent zum System erster Ordnung mit

$$x_0 = y, x_1 = \dot{y}, x_2 = w, x_3 = \dot{w}$$

ergibt

$$\dot{x}_0(t) = x_1(t) \tag{3}$$
$$\dot{x}_1(t) = x_2(t) - x_0(t) + \cos(2t) \tag{4}$$
$$\dot{x}_2(t) = x_3(t) \tag{5}$$
$$\dot{x}_3(t) = x_0(t) - x_2(t) \tag{6}$$

[66]:
```
y = Function('y')
w = Function('w')
t, tau = symbols('t tau',real=True)
```

[67]:
```
dgl = (Eq( y(t).diff(t,2), w(t)-y(t)+cos(t)), \
Eq( w(t).diff(t,2), y(t)-w(t)))
dgl
```

[67]:
$$\left(\frac{d^2}{dt^2}y(t) = w(t) - y(t) + \cos\left(t\right),\ \frac{d^2}{dt^2}w(t) = -w(t) + y(t)\right)$$

[68]:
```
#dsolve(dgl,(y(t),w(t)))
```

[69]:
```
A = Matrix(4, 4, [0, 1, 0, 0,\
                  -1, 0, 1, 0,\
                  0, 0, 0, 1,\
                  1, 0, -1, 0])
A
```

[69]:
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 \end{bmatrix}$$

[70]:
```
MexpA = lambda t: simplify((t*A).exp())
```

Wir nutzen die Variation der Konstantenformel

[71]:
```
u0 = Matrix(4, 1, [1, 0, 1, 0])
f = lambda t:  Matrix(4, 1, [0, cos(2*t), 0, 0])
u0, f(tau)
```

[71]:
$$\left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix},\ \begin{bmatrix} 0 \\ \cos\left(2\tau\right) \\ 0 \\ 0 \end{bmatrix}\right)$$

[72]:
```
integrand = simplify(MexpA(t-tau)*f(tau))
integrand
```

[72]:
$$\begin{bmatrix} \frac{\left(2t-2\tau+\sqrt{2}\sin\left(\sqrt{2}(t-\tau)\right)\right)\cos\left(2\tau\right)}{4} \\ \frac{\left(\cos\left(\sqrt{2}(t-\tau)\right)+1\right)\cos\left(2\tau\right)}{2} \\ \frac{\left(2t-2\tau-\sqrt{2}\sin\left(\sqrt{2}(t-\tau)\right)\right)\cos\left(2\tau\right)}{4} \\ \frac{\left(1-\cos\left(\sqrt{2}(t-\tau)\right)\right)\cos\left(2\tau\right)}{2} \end{bmatrix}$$

[73]:
```
u = MexpA(t)*u0 + integrate(integrand, (tau, 0, t))
```

[74]:
```
u
```

[74]:
$$\begin{bmatrix} \frac{\sin^2(t)\sin^2\left(\sqrt{2}t\right)}{2} + \frac{\sin^2(t)\cos^2\left(\sqrt{2}t\right)}{2} - \frac{\sin^2\left(\sqrt{2}t\right)}{4} - \frac{\cos(2t)}{8} - \frac{\cos^2\left(\sqrt{2}t\right)}{4} + \frac{\cos\left(\sqrt{2}t\right)}{4} + \frac{9}{8} \\ \sin(t)\sin^2\left(\sqrt{2}t\right)\cos(t) + \sin(t)\cos(t)\cos^2\left(\sqrt{2}t\right) + \frac{\sin(2t)}{4} - \frac{\sqrt{2}\sin\left(\sqrt{2}t\right)}{4} \\ -\frac{\sin^2(t)\sin^2\left(\sqrt{2}t\right)}{2} - \frac{\sin^2(t)\cos^2\left(\sqrt{2}t\right)}{2} + \frac{\sin^2\left(\sqrt{2}t\right)}{4} - \frac{\cos(2t)}{8} + \frac{\cos^2\left(\sqrt{2}t\right)}{4} - \frac{\cos\left(\sqrt{2}t\right)}{4} + \frac{9}{8} \\ -\sin(t)\sin^2\left(\sqrt{2}t\right)\cos(t) - \sin(t)\cos(t)\cos^2\left(\sqrt{2}t\right) + \frac{\sin(2t)}{4} + \frac{\sqrt{2}\sin\left(\sqrt{2}t\right)}{4} \end{bmatrix}$$

[75]:
```
simplify(u.diff(t) - A*u - f(t)) , u.subs(t,0)-u0 # Test
```

[75]:
$$\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)$$