lektion13

January 23, 2020

Table of Contents

[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,t,n = symbols('x t n')
y = Function('y')
u = Function('u')
```

1 Lektion 13

2 Differentialgleichungen Teil 2

2.1 Wiederholung

2.1.1 Skalare lineare DGL erster Ordnung

[2]:
```
y = Function('y')
t, tau = symbols('t tau', real = True)
dgl = Eq(y(t).diff(t)-y(t), 0)
dgl
```

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

[3]:
```
awe = {y(0) : 1} # Anfangswert
```

[4]:
```
dsolve(dgl,y(t),ics=awe)
```

[4]:
$$y(t) = e^{t}$$

2.1.2 Skalare lineare DGL zweiter Ordnung

Anfangswertproblem

$$\ddot{y}(t) = -2\dot{y}(t) - y(t) + 3\cos(t) \quad \text{für } t \in (1, 10)$$

$$y(1) = 1, \;\; \dot{y}(1) = -1 \quad \text{Anfangswerte}$$

[5]:
```
dgl2 = Eq(y(t).diff(t,2), -2*y(t).diff(t) - y(t) + 3*cos(t))
dgl2
```

[5]:
$$\frac{d^2}{dt^2}y(t) = -y(t) + 3\cos(t) - 2\frac{d}{dt}y(t)$$

[6]:
```
awe = {y(1) : 1, y(t).diff(t).subs(t,1) : 1}
```

[7]:
```
asol = dsolve(dgl2, y(t), ics=awe)
asol
```

[7]:
$$y(t) = \left(\frac{et\left(-3\sqrt{2}\sin\left(\frac{\pi}{4}+1\right)+4\right)}{2} + \frac{e\left(-2+3\cos(1)\right)}{2}\right)e^{-t} + \frac{3\sin(t)}{2}$$

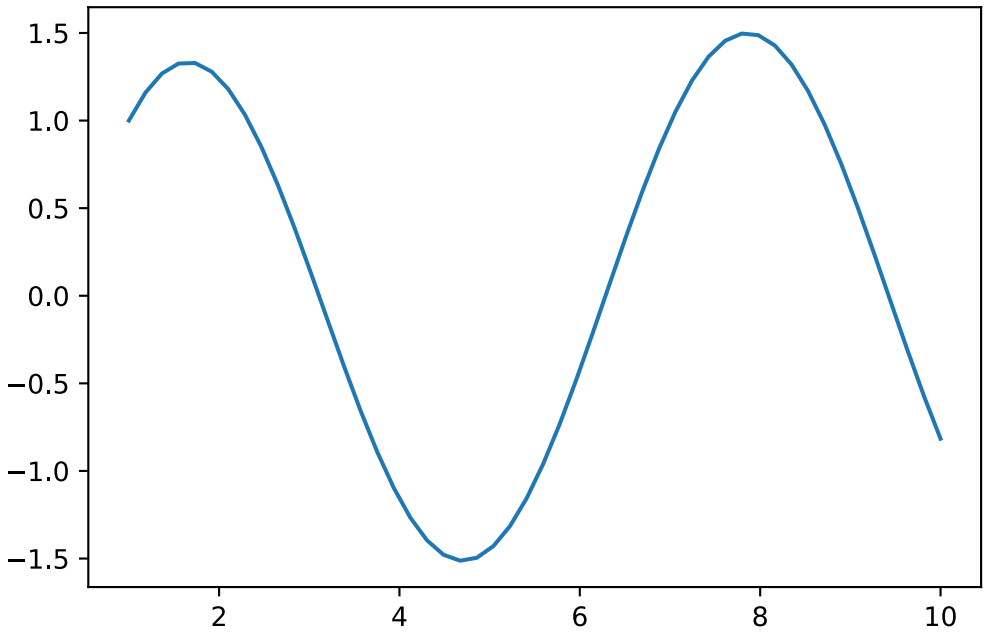
[8]:
```
asol.rhs
```

[8]:
$$\left(\frac{et\left(-3\sqrt{2}\sin\left(\frac{\pi}{4}+1\right)+4\right)}{2} + \frac{e\left(-2+3\cos(1)\right)}{2}\right)e^{-t} + \frac{3\sin(t)}{2}$$

[9]:
```
#asoln = lambdify(t,asol.rhs) # das funktioniert in neueren Sympyversionen
```

[10]:
```
asoln = lambdify(t, asol.rhs, modules=[{'e' : np.exp(1)}, 'numpy']) #␣
 ↪workaround für ältere Versionen
tn = np.linspace(1,10)

fig = plt.figure(1)
ax = fig.gca()
ax.plot(tn,asoln(tn))
```

[10]:
```
[<matplotlib.lines.Line2D at 0x7f0aa1c41d50>]
```



Randwertproblem

$$\ddot{y}(t) = -2\dot{y}(t) - y(t) + 3\cos(t) \quad \texttt{für } t \in (1, 10)$$

$$y(1) = 1, \;\; y(10) = -2 \quad \texttt{Randwerte}$$

[11]:
```
rwe = {y(1):1, y(10):-2}
dgl2
```

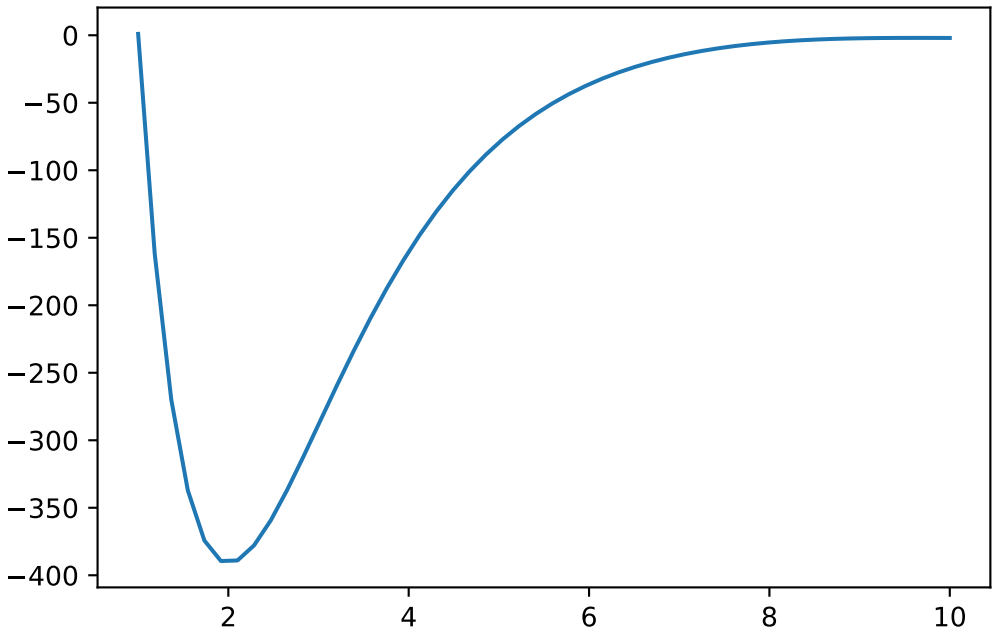
[11]:
$$\frac{d^2}{dt^2}y(t) = -y(t) + 3\cos(t) - 2\frac{d}{dt}y(t)$$

[12]:
```
rsol = dsolve(dgl2, y(t), ics=rwe)
rsol
```

[12]:
$$y(t) = \left(\frac{et\left(-4e^9 - 2 + 3\sin(1) - 3e^9\sin(10)\right)}{18} + \frac{e\left(3e^9\sin(10) - 30\sin(1) + 20 + 4e^9\right)}{18}\right)e^{-t} + \frac{3\sin(t)}{2}$$

[13]:
```
rsoln = lambdify(t,rsol.rhs, modules=[{'E':np.exp(1)}, 'numpy'])
tn = np.linspace(1,10)

fig = plt.figure(2)
ax = fig.gca()
ax.plot(tn,rsoln(tn))
```

[13]: [<matplotlib.lines.Line2D at 0x7f0aa1c29550>]



2.2 Pendelgleichung (physikalisches Pendel)

$$\ddot{\alpha}(t) = -\sin(\alpha(t))$$

äquivalent zu System 1. Ordnung

$$y_1(t) := \alpha(t), \quad y_2(t) := \frac{d}{dt}\alpha(t)$$

$$\begin{aligned} \dot{y}_1(t) &= y_2(t) \\ \dot{y}_2(t) &= -\sin(y_1(t)) \end{aligned}$$

y_1: Winkel

y_2: Winkelgeschwindigkeit

[14]:
```
dgl = Eq(y(t).diff(t,2),-sin(y(t)))
dgl
```

[14]: $$\frac{d^2}{dt^2}y(t) = -\sin\left(y(t)\right)$$

[15]:
```
dsolve(dgl)
```

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

      NotImplementedError                       Traceback (most recent call last)

      <ipython-input-15-f07466621aa9> in <module>
   ----> 1 dsolve(dgl)


      /local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/ode.
↪py in dsolve(eq, func, hint, simplify, ics, xi, eta, x0, n, **kwargs)
      644         hints = _desolve(eq, func=func,
      645             hint=hint, simplify=True, xi=xi, eta=eta, type='ode',␣
↪ics=ics,
   --> 646             x0=x0, n=n, **kwargs)
      647         eq = hints.pop('eq', eq)
      648         all_ = hints.pop('all', False)


      /local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/
↪deutils.py in _desolve(eq, func, hint, ics, simplify, **kwargs)
      242                 str(eq) + " is not a solvable differential equation in␣
↪" + str(func))
      243         else:
   --> 244             raise NotImplementedError(dummy + "solve" + ": Cannot␣
↪solve " + str(eq))
      245     if hint == 'default':
      246         return _desolve(eq, func, ics=ics, hint=hints['default'],␣
↪simplify=simplify,


      NotImplementedError: solve: Cannot solve sin(y(t)) + Derivative(y(t), (t,␣
↪2))
```

2.2.1 Phasenraum und Trajektorien

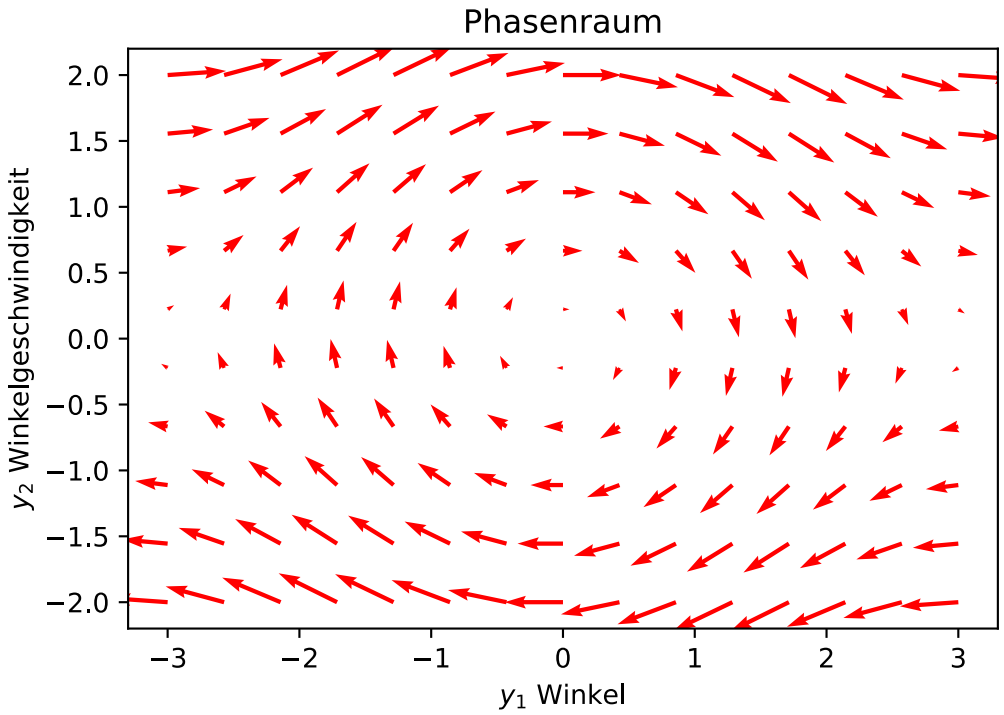
vgl. Ana II Kurzskript Kap. 19 http://www.math.uni-duesseldorf.de/~internet/ana2_19/vorlesung.p

[16]:
```
# Achtung die Reihenfolge von y und t ist hier im Vergleich zu obiger Ana II␣
↪Vorlesung vertauscht
def f(y, t):
    y1 = y[0]
    y2 = y[1]
    return y2,-np.sin(y1)
```

[17]:
```
y1 = np.linspace(-3.0, 3.0, 15)
y2 = np.linspace(-2.0, 2.0, 10)
Y1, Y2 = np.meshgrid(y1, y2)
t0 = 0
```

[18]:
```
U, V = f([Y1, Y2], t0)
```

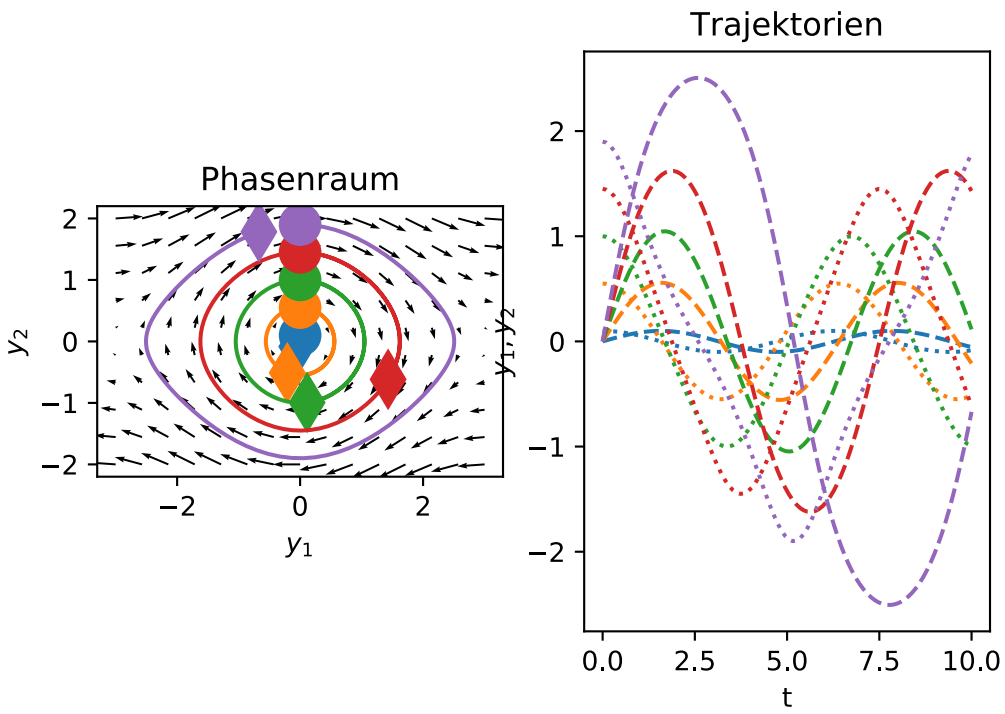
[19]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title('Phasenraum')
ax.quiver(Y1, Y2, U, V, angles='xy', color='r')
ax.set_aspect('equal');
ax.set_xlabel('$y_1$ Winkel');
ax.set_ylabel('$y_2$ Winkelgeschwindigkeit' );
```



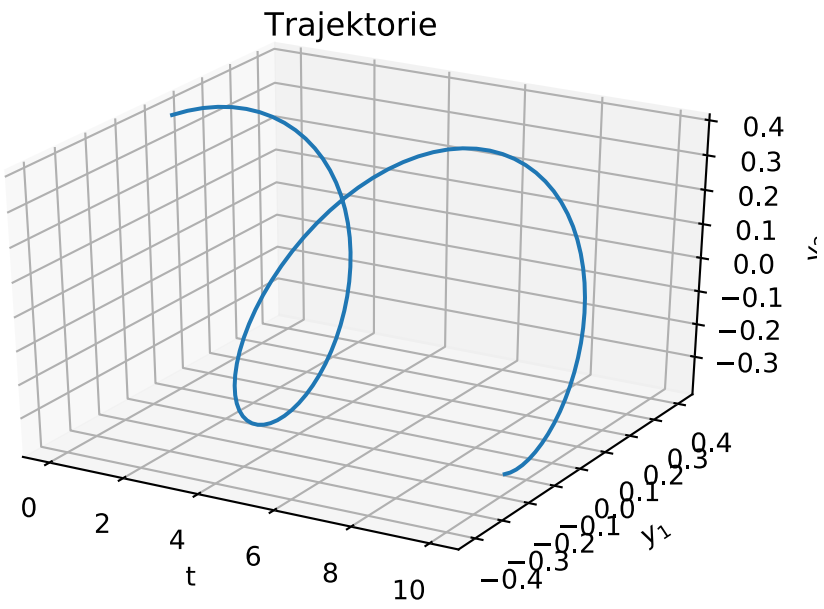
[20]:
```
from scipy.integrate import odeint  # numerische Integration (numerisches␣
↪Lösungsverfahren für DGL)
tn = np.linspace(0,1,10)
y0 = [0.0, 1] # Anfangswert
yn = odeint(f, y0, tn) #berechnet Näherungslösung y für jeden Wert in tn
yn
```

[20]:
```
array([[0.        , 1.        ],
       [0.11088278, 0.99383983],
       [0.22040221, 0.97550971],
       [0.32722793, 0.94544907],
       [0.43009301, 0.90435276],
       [0.52781962, 0.85312046],
       [0.61933847, 0.79279585],
       [0.70370131, 0.72450326],
       [0.78008653, 0.64938844],
       [0.84779864, 0.56856905]])
```

[21]:
```
fig = plt.figure(3)

ax1 = fig.add_subplot(121)
ax1.quiver(Y1, Y2, U, V, angles='xy', color='k')
ax1.set_aspect('equal');
ax1.set_xlabel('$y_1$');
ax1.set_ylabel('$y_2$');
ax1.set_title('Phasenraum')

ax2 = fig.add_subplot(122)
ax2.set_title('Trajektorien')

tn = np.linspace(0, 10, 100)
for y20 in np.linspace(0.1, 1.9, 5):
    y0 = [0.0, y20]

    yn = odeint(f, y0, tn)

    line, = ax1.plot(yn[:,0], yn[:,1], '-') # Phasenporträt (Lösung im␣
 ↪Phasenraum)
    ax1.plot(yn[0,0], yn[0,1], 'o', color=line.get_color(), markersize=15)  #␣
 ↪Startwert
    ax1.plot(yn[-1,0], yn[-1,1], 'd', color=line.get_color(), markersize=15) #␣
 ↪Wert zum Endzeitpunkt

    ax2.plot(tn, yn[:,0], '--', color=line.get_color()) # y_1
    ax2.plot(tn, yn[:,1], ':', color=line.get_color())  # y_2
    ax2.set_xlabel('t')
    ax2.set_ylabel('$y_1, y_2$')
```



[22]:
```
from mpl_toolkits.mplot3d import Axes3D

y0 = [0.0,0.4]
yn = odeint(f,y0,tn)

fig = plt.figure(4)
ax = fig.add_subplot(111,projection='3d')
ax.plot3D(tn,yn[:,0],yn[:,1])
ax.set_xlabel('t')
ax.set_ylabel('$y_1$')
ax.set_zlabel('$y_2$')
ax.set_title('Trajektorie')
plt.show()
```



[23]:
```
ax.view_init(0, 0)
plt.show()
```

[24]:
```
ax.view_init(0, 90)
plt.show()
```

[25]:
```
ax.view_init(90,-90)
plt.show()
```

falls der Winkel klein ist, ist

$$\sin(\alpha) \approx \alpha$$

und wir erhalten als Approximation

$$\dot{u}_1(t) = u_2(t) \tag{1}$$
$$\dot{u}_2(t) = -u_1(t)) \tag{2}$$

[26]:
```
dgl3 = Eq(u(t).diff(t,2), -u(t))
awe = {u(0):y0[0], u(t).diff(t).subs(t,0):y0[1]}
awe
sol3 = dsolve(dgl3, u(t), ics=awe).rhs
```

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

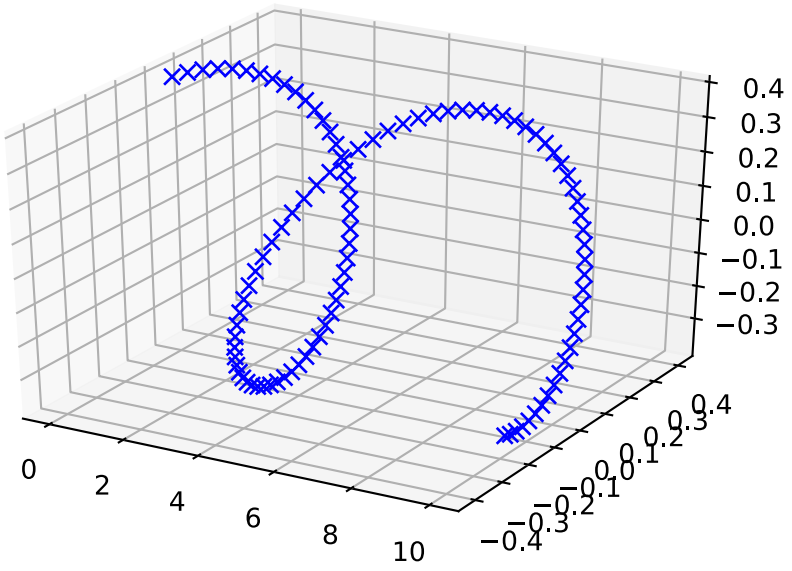
```
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-26-619b69d49e3b> in <module>
  2 awe = {u(0):y0[0], u(t).diff(t).subs(t,0):y0[1]}
  3 awe
----> 4 sol3 = dsolve(dgl3, u(t), ics=awe).rhs


/local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/ode.
↪py in dsolve(eq, func, hint, simplify, ics, xi, eta, x0, n, **kwargs)
677             # The key 'hint' stores the hint needed to be solved for.
678             hint = hints['hint']
--> 679             return _helper_simplify(eq, hint, hints, simplify, ics=ics)
680
681 def _helper_simplify(eq, hint, match, simplify=True, ics=None,␣
↪**kwargs):


/local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/ode.
↪py in _helper_simplify(eq, hint, match, simplify, ics, **kwargs)
722     if ics and not 'power_series' in hint:
723         if isinstance(rv, Expr):
--> 724             solved_constants = solve_ics([rv], [r['func']], cons(rv),␣
↪ics)
725             rv = rv.subs(solved_constants)
726         else:


/local/schaedle/miniconda/lib/python3.7/site-packages/sympy/solvers/ode.
↪py in solve_ics(sols, funcs, constants, ics)
830     # we use NotImplementedError in this case.
831     if not solved_constants:
--> 832         raise ValueError("Couldn't solve for initial conditions")
833
834     if solved_constants == True:


ValueError: Couldn't solve for initial conditions
```

[27]:
```
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot3D(tn,yn[:,0],yn[:,1],'bx')
ax.plot3D(tn,lambdify(t,sol3)(tn),lambdify(t,sol3.diff(t))(tn),'ro')
ax.set_xlabel('t')
ax.set_ylabel('$u_1$')
ax.set_zlabel('$u_2$')
ax.set_title('Trajektorie')
```

```
plt.show()
```

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

NameError                                 Traceback (most recent call last)

<ipython-input-27-2e4436840291> in <module>
      2 ax = fig.add_subplot(111,projection='3d')
      3 ax.plot3D(tn,yn[:,0],yn[:,1],'bx')
----> 4 ax.plot3D(tn,lambdify(t,sol3)(tn),lambdify(t,sol3.diff(t))(tn),'ro')
      5 ax.set_xlabel('t')
      6 ax.set_ylabel('$u_1$')


NameError: name 'sol3' is not defined
```



2.3 Bessel Differentialgleichung

[28]:
```
dgl = Eq(y(x).diff(x,2), -1/x * y(x).diff(x)+ 1/x**2*y(x) + 4*y(x))
dgl
```

[28]:
$$\frac{d^2}{dx^2}y(x) = 4y(x) - \frac{\frac{d}{dx}y(x)}{x} + \frac{y(x)}{x^2}$$

[29]:
```
# dsolve(dgl) # funktioniert nicht
```

[30]:
```
a = symbols('a:18')
N = len(a)
```

[31]:
```
ys = sum([a[j]*x**j for j in range(N)])
ys
```

[31]:
$$a_0 + a_1 x + a_{10} x^{10} + a_{11} x^{11} + a_{12} x^{12} + a_{13} x^{13} + a_{14} x^{14} + a_{15} x^{15} + a_{16} x^{16} + a_{17} x^{17} + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7 + a_8 x^8 + a_9 x^9$$

[32]:
```
gl = dgl.subs(y(x),ys).doit()
gl
```

[32]:
$$2\left(45a_{10}x^8 + 55a_{11}x^9 + 66a_{12}x^{10} + 78a_{13}x^{11} + 91a_{14}x^{12} + 105a_{15}x^{13} + 120a_{16}x^{14} + 136a_{17}x^{15} + a_2 + 3a_3 x + 6a_4 x^2 \right.$$
$$4a_0 + 4a_1 x + 4a_{10}x^{10} + 4a_{11}x^{11} + 4a_{12}x^{12} + 4a_{13}x^{13} + 4a_{14}x^{14} + 4a_{15}x^{15} + 4a_{16}x^{16} +$$
$$4a_{17}x^{17} + 4a_2 x^2 + 4a_3 x^3 + 4a_4 x^4 + 4a_5 x^5 + 4a_6 x^6 + 4a_7 x^7 + 4a_8 x^8 + 4a_9 x^9 -$$
$$a_1 + 10a_{10}x^9 + 11a_{11}x^{10} + 12a_{12}x^{11} + 13a_{13}x^{12} + 14a_{14}x^{13} + 15a_{15}x^{14} + 16a_{16}x^{15} + 17a_{17}x^{16} + 2a_2 x + 3a_3 x^2 + 4a$$
$$\frac{a_0 + a_1 x + a_{10}x^{10} + a_{11}x^{11} + a_{12}x^{12} + a_{13}x^{13} + a_{14}x^{14} + a_{15}x^{15} + a_{16}x^{16} + a_{17}x^{17}}{x} + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 +$$
$$x^2$$

[33]:
```
gl1 = (gl.lhs - gl.rhs).expand()
gl1
```

[33]:
$$-4a_0 - \frac{a_0}{x^2} - 4a_1 x - 4a_{10}x^{10} + 99a_{10}x^8 - 4a_{11}x^{11} + 120a_{11}x^9 - 4a_{12}x^{12} + 143a_{12}x^{10} - 4a_{13}x^{13} + 168a_{13}x^{11} - 4a_{14}x^{14} + 195a_{14}x^{12} - 4a_{15}x^{15} + 224a_{15}x^{13} - 4a_{16}x^{16} + 255a_{16}x^{14} - 4a_{17}x^{17} + 288a_{17}x^{15} - 4a_2 x^2 + 3a_2 - 4a_3 x^3 + 8a_3 x - 4a_4 x^4 + 15a_4 x^2 - 4a_5 x^5 + 24a_5 x^3 - 4a_6 x^6 + 35a_6 x^4 - 4a_7 x^7 + 48a_7 x^5 - 4a_8 x^8 + 63a_8 x^6 - 4a_9 x^9 + 80a_9 x^7$$

[34]:
```
##gl1.as_poly(t).all_coeffs()
# klappt nicht
```

[35]:
```
gl1.coeff(x**(-2))
```

[35]:
$$-a_0$$

[36]:
```
gl1.coeff(x**(-1))
```

[36]:
$$0$$

[37]:
```
gl1.coeff(x,-1)
```

[37]:
$$0$$

[38]:
```
gl1.coeff(x,0) # x Term
```

[38]:
$$-4a_0 + 3a_2$$

[39]:
```
gls = []
for j in range(N+1):
    glg = Eq(gl1.coeff(x,j-2),0)
    #if glg != True:
    gls.append(glg)
gls
```

[39]: $[-a_0 = 0,\ \texttt{True},\ -4a_0 + 3a_2 = 0,\ -4a_1 + 8a_3 = 0,\ -4a_2 + 15a_4 = 0,\ -4a_3 + 24a_5 = 0,\ -4a_4 + 35a_6 = 0,\ -4a$

[40]:
```
solve(gls[:-1])
```

[40]: $\left\{a_0 : 0,\ a_1 : 2880a_9,\ a_{10} : 0,\ a_{11} : \frac{a_9}{30},\ a_{12} : 0,\ a_{13} : \frac{a_9}{1260},\ a_{14} : 0,\ a_{15} : \frac{a_9}{70560},\ a_{16} : 0,\ a_{17} : \frac{a_9}{5080320},\ a_2 : 0,\right.$

[41]:
```
var = list(a).copy()
del var[1]
var
```

[41]: $[a_0,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7,\ a_8,\ a_9,\ a_{10},\ a_{11},\ a_{12},\ a_{13},\ a_{14},\ a_{15},\ a_{16},\ a_{17}]$

[42]:
```
Lsg = solve(gls[:-1],var)
Lsg
```

[42]: $\left\{a_0 : 0,\ a_{10} : 0,\ a_{11} : \frac{a_1}{86400},\ a_{12} : 0,\ a_{13} : \frac{a_1}{3628800},\ a_{14} : 0,\ a_{15} : \frac{a_1}{203212800},\ a_{16} : 0,\ a_{17} : \frac{a_1}{14631321600},\ a_2 :\right.$

[43]:
```
Lsg[a[1]] = a[1]
Lsg
```

[43]: $\left\{a_0 : 0,\ a_1 : a_1,\ a_{10} : 0,\ a_{11} : \frac{a_1}{86400},\ a_{12} : 0,\ a_{13} : \frac{a_1}{3628800},\ a_{14} : 0,\ a_{15} : \frac{a_1}{203212800},\ a_{16} : 0,\ a_{17} : \frac{a_1}{146313216}\right.$

[44]:
```
q = [Lsg[a[2*j+1]]/Lsg[a[2*j+3]] for j in range(int(N/2)-2)]
q
```

[44]: $[2,\ 6,\ 12,\ 20,\ 30,\ 42,\ 56]$

[45]:
```
q = [Lsg[a[2*j+1]]/Lsg[a[2*j+3]]/(j+2) for j in range(int(N/2)-2)]
q
```

[45]: $[1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7]$

Also ist

$$\frac{a_{2j+1}}{a_{2j+3}} = (j+1)(j+2)$$

Test

[46]:
```
q = [Lsg[a[2*j+1]]/Lsg[a[2*j+3]] - (j+1)*(j+2) for j in range(int(N/2)-2)]
q
```

[46]: $[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$

und damit

$$a_{2n+1} = \prod_{j=0}^{n-1} \frac{1}{(j+1)(j+2)} a_1 = \frac{a_1}{(n)!(n+1)!}$$

[47]:
```
for j in range(int(N/2)-2):
    display((Lsg[a[2*j+1]], a[1]/factorial(j)/factorial(j+1)))
```

$(a_1,\ a_1)$

$\left(\frac{a_1}{2},\ \frac{a_1}{2}\right)$

$\left(\frac{a_1}{12},\ \frac{a_1}{12}\right)$

$\left(\frac{a_1}{144},\ \frac{a_1}{144}\right)$

$\left(\frac{a_1}{2880},\ \frac{a_1}{2880}\right)$

$\left(\frac{a_1}{86400},\ \frac{a_1}{86400}\right)$

$\left(\frac{a_1}{3628800},\ \frac{a_1}{3628800}\right)$

[48]:
```
yR = Sum(x**(2*n+1)/factorial(n)/factorial(n+1),(n,0,oo))
yR
```

[48]: $\sum_{n=0}^{\infty} \frac{x^{2n+1}}{n!\,(n+1)!}$

[49]:
```
u = yR.doit()
u
```

[49]: $I_1\left(2x\right)$

[50]:
```
srepr(u)
```

[50]: "besseli(Integer(1), Mul(Integer(2), Symbol('x')))"

[51]:
```
?besseli
```

[52]:
```
tmp  = dgl.subs(y(x),u).doit()
tmp
```

[52]: $3I_1\left(2x\right) + I_3\left(2x\right) = 4I_1\left(2x\right) - \frac{I_0\left(2x\right) + I_2\left(2x\right)}{x} + \frac{I_1\left(2x\right)}{x^2}$

[53]:
```
simplify(tmp.lhs - tmp.rhs)
```

[53]: 0

2.4 Bernoulli DGL

[54]:
```
dgl = Eq(y(x).diff(x),sqrt(x*y(x)))
dgl
```

[54]:
$$\frac{d}{dx}y(x) = \sqrt{xy(x)}$$

[57]:
```
sol = dsolve(dgl)
sol
```

[57]:
$$y(x) = \frac{C_1^2}{4} - \frac{C_1\sqrt{x^3}}{3} + \frac{x^3}{9}$$

[59]:
```
C1 = sol.atoms(Symbol).difference(dgl.atoms(Symbol)).pop()
```

[61]:
```
sol.rhs.subs(C1,1)
```

[61]:
$$\frac{x^3}{9} - \frac{\sqrt{x^3}}{3} + \frac{1}{4}$$

[66]:
```
y = sol.rhs
y
```

[66]:
$$\frac{C_1^2}{4} - \frac{C_1\sqrt{x^3}}{3} + \frac{x^3}{9}$$

[67]:
```
gl1 = Eq(y.subs(x,1),Rational(1,50)) #Startwert
gl1
```

[67]:
$$\frac{C_1^2}{4} - \frac{C_1}{3} + \frac{1}{9} = \frac{1}{50}$$

[68]:
```
K1 = solve(gl1)
K1
```

[68]:
$$\left[\frac{2}{3} - \frac{\sqrt{2}}{5}, \quad \frac{\sqrt{2}}{5} + \frac{2}{3}\right]$$