lektion10

January 29, 2018

Table of Contents


```
In [12]: import matplotlib.pyplot as plt
         import numpy as np
         from sympy import *
         init_printing()
         x,y,z = symbols('x y z')
         #%matplotlib notebook
         %matplotlib inline
```

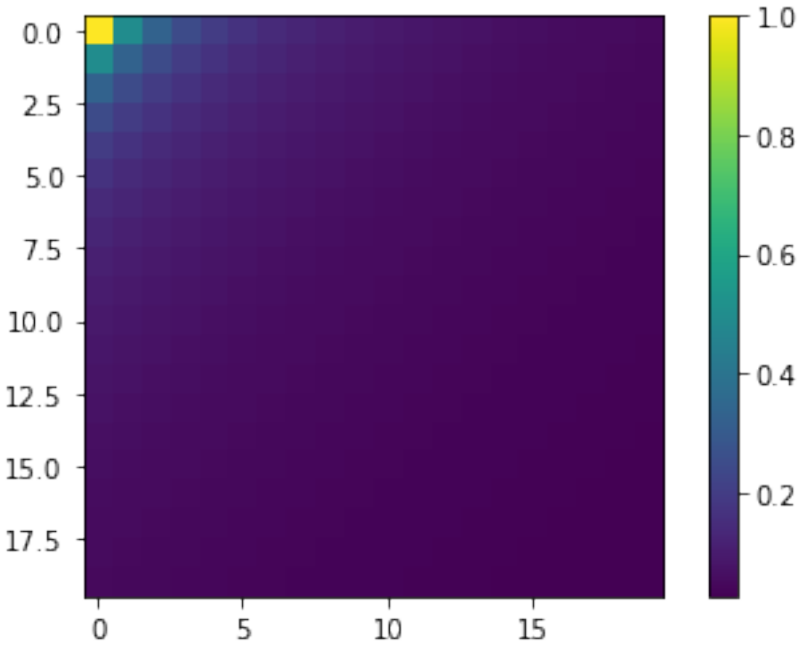
1 Lektion 10

1.1 Lineare Algebra II

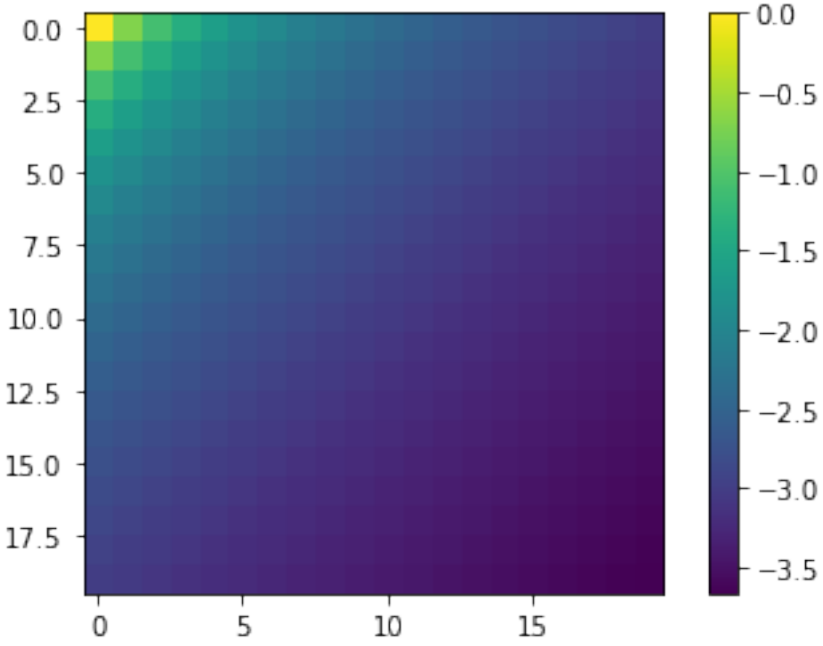
1.1.1 Matrixplots

```
In [13]: H = Matrix(20,20,lambda i,j : 1/(i+j+1))
```

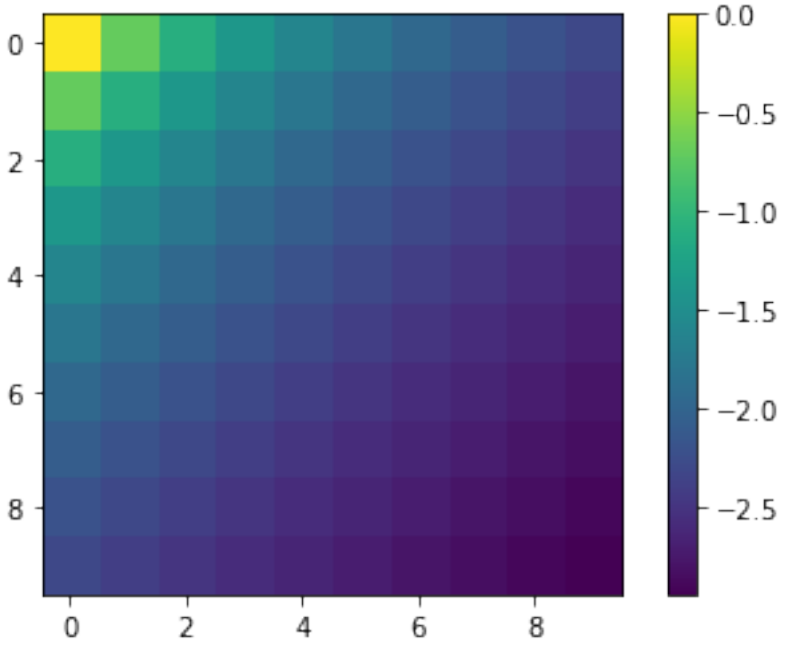
```
In [14]: HH = matrix2numpy(H)
         plt.figure()
         plt.imshow(HH.astype(float))
         plt.colorbar();
```



In [15]:
```
plt.figure()
plt.imshow(np.log(HH.astype(float)))
plt.colorbar();
```



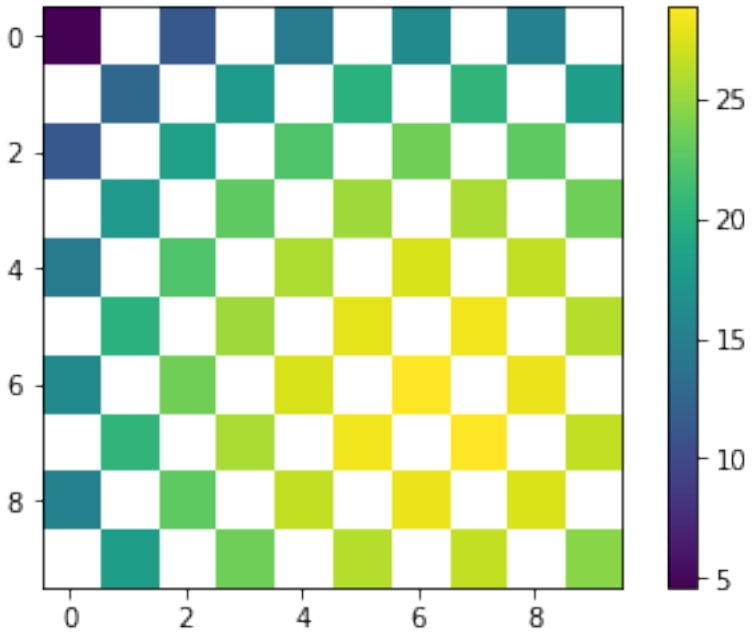
In [16]:
```
H = Matrix(10,10,lambda i,j : 1/(i+j+1))
HH = matrix2numpy(H)
plt.figure()
plt.imshow(np.log(HH.astype(float)))
plt.colorbar();
```



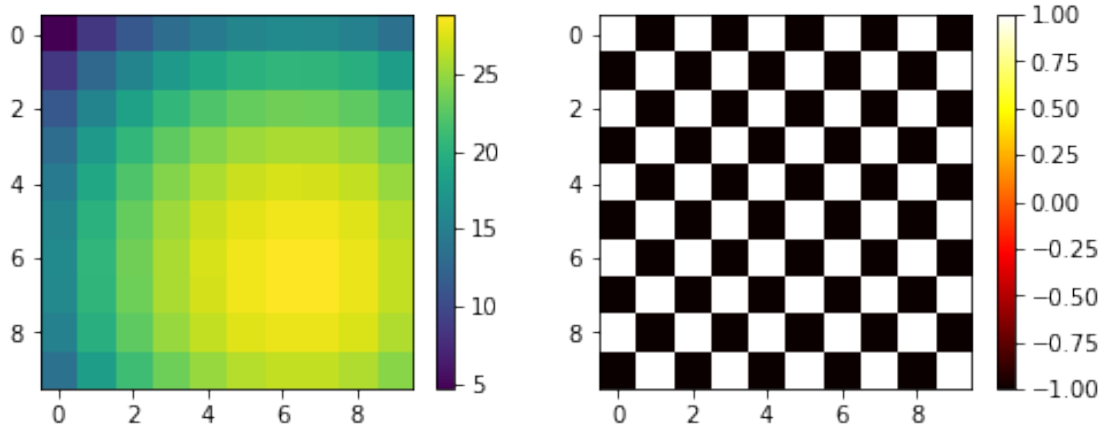
In [17]:
```
Hinv = H**(-1)
HHinv = matrix2numpy(Hinv)
plt.figure()
plt.imshow(np.log(HHinv.astype(float)))
plt.colorbar()
```

```
/local/schaedle/miniconda3/envs/py36/lib/python3.6/site-packages/ipykernel_launcher.py:4: Runtim
  after removing the cwd from sys.path.
```

Out[17]: <matplotlib.colorbar.Colorbar at 0x7f8add903860>



In [18]:
```
fig = plt.figure(figsize=(8,3))
fig.add_subplot(121)
plt.imshow(np.log(np.abs(HHinv.astype(float))))
plt.colorbar()
fig.add_subplot(122)
plt.imshow(np.sign(HHinv.astype(float)),cmap='hot')
plt.colorbar();
```



1.1.2 Eigenwerte, Eigenvektoren, Jordannormalform

In [19]:
```
A = Matrix(3,3,[-4,-2,-3,5,3,3,5,2,4])
A
```

Out[19]:

$$\begin{bmatrix} -4 & -2 & -3 \\ 5 & 3 & 3 \\ 5 & 2 & 4 \end{bmatrix}$$

In [20]:
```
A.eigenvals()
```

Out[20]:

$$\{1 : 3\}$$

In [21]:
```
A.eigenvects()
```

Out[21]:

$$\left[\left(1,\ 3,\ \left[\begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{3}{5} \\ 0 \\ 1 \end{bmatrix}\right]\right)\right]$$

In [22]:
```
A.diagonalize()
```

```
---------------------------------------------------------------------------

MatrixError                               Traceback (most recent call last)

<ipython-input-22-842d3ad24c22> in <module>()
----> 1 A.diagonalize()


/local/schaedle/miniconda3/envs/py36/lib/python3.6/site-packages/sympy/matrices/matrices
   1073
   1074         if not self.is_diagonalizable(reals_only=reals_only, clear_cache=False):
-> 1075             raise MatrixError("Matrix is not diagonalizable")
   1076
   1077         eigenvecs = self._cache_eigenvects


MatrixError: Matrix is not diagonalizable
```

In [23]:
```
T,J = A.jordan_form()
T,J
```

Out[23]:

$$\left(\begin{bmatrix} -5 & 1 & -\frac{2}{5} \\ 5 & 0 & 1 \\ 5 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

In [24]:
```
T*J*T.inv() == A
```

Out[24]:
```
True
```

In [25]:
```
v = T[:,2]
v, A*v
```

Out[25]:

$$\left(\begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}\right)$$

1.1.3 Berechung des Rangs

In [26]:
```
M = Matrix(3,3,range(1,10))
M
```

Out[26]:

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

In [27]:
```
M.rank()
```

Out[27]:

$$2$$

In [28]:
```
M = Matrix(3,2,[2*x+2,2*y-2,2*x+2,-2*y+2,y-1,x+1])
M
```

Out[28]:

$$\begin{bmatrix} 2x+2 & 2y-2 \\ 2x+2 & -2y+2 \\ y-1 & x+1 \end{bmatrix}$$

In [29]:
```
M.rank()
```

Out[29]:

$$2$$

In [30]:
```
M.row_op(1,lambda r,j: r-M[0,j])
M1 = M.copy()
M1
```

Out[30]:

$$\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & -4y+4 \\ y-1 & x+1 \end{bmatrix}$$

In [31]:
```
M1.row_op(0,lambda r,j: r*(y-1))    # falls y != 1
M2 = M1.copy()
M2
```

Out[31]:

$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & -4y+4 \\ y-1 & x+1 \end{bmatrix}$$

In [32]:
```
M2.row_op(2,lambda r,j: r*(2*x+2))  # falls x != -1
M3 = M2.copy()
M3
```

Out[32]:

$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & -4y+4 \\ (2x+2)(y-1) & (x+1)(2x+2) \end{bmatrix}$$

In [33]:
```
M3.row_op(2,lambda r,j: r-M3[0,j])
M4 = M3.copy()
M4
```

Out[33]:

$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & -4y+4 \\ 0 & (x+1)(2x+2)-(y-1)(2y-2) \end{bmatrix}$$

In [34]:
```
M4.expand()
```

Out[34]:

$$\begin{bmatrix} 2xy-2x+2y-2 & 2y^2-4y+2 \\ 0 & -4y+4 \\ 0 & 2x^2+4x-2y^2+4y \end{bmatrix}$$

In [35]:
```
factor(M4) # Rang 2 falls x != -1 und y!=1
```

Out[35]:

$$\begin{bmatrix} 2(x+1)(y-1) & 2(y-1)^2 \\ 0 & -4(y-1) \\ 0 & 2(x+y)(x-y+2) \end{bmatrix}$$

In [36]:
```
Mx = M.copy()
Mx = Mx.subs(x,-1)
M, Mx  # Rang 2 falls x = -1 und y!=1
```

Out[36]:

$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & -4y+4 \\ y-1 & x+1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 2y-2 \\ 0 & -4y+4 \\ y-1 & 0 \end{bmatrix} \right)$$

In [37]:
```
My = M.copy()
My = My.subs(y,1)
M, My   # Rang 2 falls x != -1 und y=1
```

Out[37]:

$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & -4y+4 \\ y-1 & x+1 \end{bmatrix}, \quad \begin{bmatrix} 2x+2 & 0 \\ 0 & 0 \\ 0 & x+1 \end{bmatrix} \right)$$

In [38]:
```
Mxy = M.copy()
M, Mxy.subs({x:-1,y:1})
```

Out[38]:

$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & -4y+4 \\ y-1 & x+1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right)$$

1.1.4 Normen

In [39]:
```
v = Matrix(1,3,[1,2,3])
v
```

Out[39]:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

In [40]:
```
v.norm()
```

Out[40]:

$$\sqrt{14}$$

In [41]:
```
v.norm(1)
```

Out[41]:

$$6$$

In [42]: `v.norm(oo)`

Out[42]:

$$3$$

In [43]: `A, A.norm() # Frobenius Norm`

Out[43]:

$$\left(\begin{bmatrix} -4 & -2 & -3 \\ 5 & 3 & 3 \\ 5 & 2 & 4 \end{bmatrix}, \quad 3\sqrt{13} \right)$$

In [44]: `sqrt(trace(A*A.T))`

Out[44]:

$$3\sqrt{13}$$

In [45]: `A.norm(2)`

Out[45]:

$$\sqrt{\sqrt{3363} + 58}$$

In [46]: `(A*A.T).eigenvals()`

Out[46]:

$$\left\{ 1 : 1, \quad -\sqrt{3363} + 58 : 1, \quad \sqrt{3363} + 58 : 1 \right\}$$

1.1.5 Kreuzprodukt

In [47]:
```
w = Matrix(1,3,[1,-2,1])
v,w
```

Out[47]:

$$\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \right)$$

In [48]:
```
z = v.cross(w)
z
```

Out[48]:

$$\begin{bmatrix} 8 & 2 & -4 \end{bmatrix}$$

In [49]:
```
w.cross(v)
```

Out[49]:

$$\begin{bmatrix} -8 & -2 & 4 \end{bmatrix}$$

1.2 Reihenentwicklung (Taylor)

In [50]:
```
tr = tan(x).series(x,0,6)
tr.removeO()
```

Out[50]:

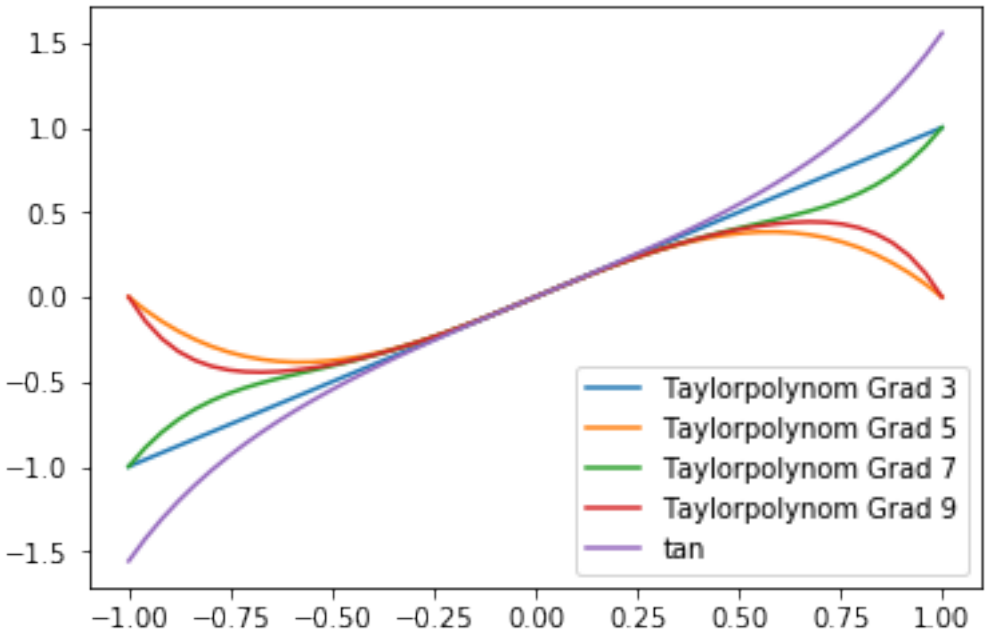
$$\frac{2x^5}{15} + \frac{x^3}{3} + x$$

In [51]:
```
ts = {}
for k in range(3,10,2):
    ts[k] = (x/(1+x**2)).series(x,0,k).removeO()

for tp in ts:
    print(tp)
```

```
3
5
7
9
```

In [52]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-1,1)
for tp in ts:
    ax.plot(xn,lambdify(x,ts[tp])(xn),label='Taylorpolynom Grad {0}'.format(tp))

ax.plot(xn,np.tan(xn),label='tan')
plt.legend()
plt.legend(loc=4)
fig.show();
```

```
/local/schaedle/miniconda3/envs/py36/lib/python3.6/site-packages/matplotlib/figure.py:418: UserW
  "matplotlib is currently using a non-GUI backend, "
```



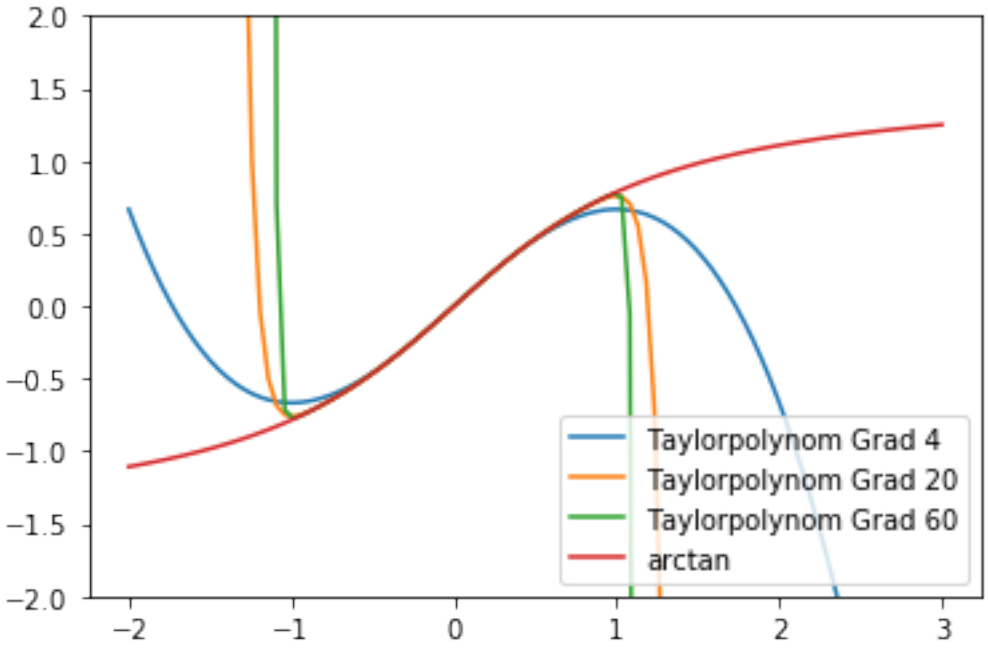
In [53]:
```
ar={}
for n in [4,20,60]:
    ar[n]= series(atan(x),x,0,n).removeO()
ar
```

Out[53]:

$$\left\{4 : -\frac{x^3}{3} + x, \quad 20 : -\frac{x^{19}}{19} + \frac{x^{17}}{17} - \frac{x^{15}}{15} + \frac{x^{13}}{13} - \frac{x^{11}}{11} + \frac{x^9}{9} - \frac{x^7}{7} + \frac{x^5}{5} - \frac{x^3}{3} + x, \quad 60 : -\frac{x^{59}}{59} + \frac{x^{57}}{57} - \frac{x^{55}}{55} + \frac{x^{53}}{53} - \right.$$

In [54]:
```
xn = np.linspace(-2,3,100)
fig2 = plt.figure()
ax = fig2.gca()
for tp in ar:
    ax.plot(xn,lambdify(x,ar[tp])(xn), \
            label='Taylorpolynom Grad {0}'.format(tp))

ax.plot(xn,np.arctan(xn),label='arctan')

ax.set_ylim(-2,2)
plt.legend()
plt.legend(loc=4)
fig2.show()
```

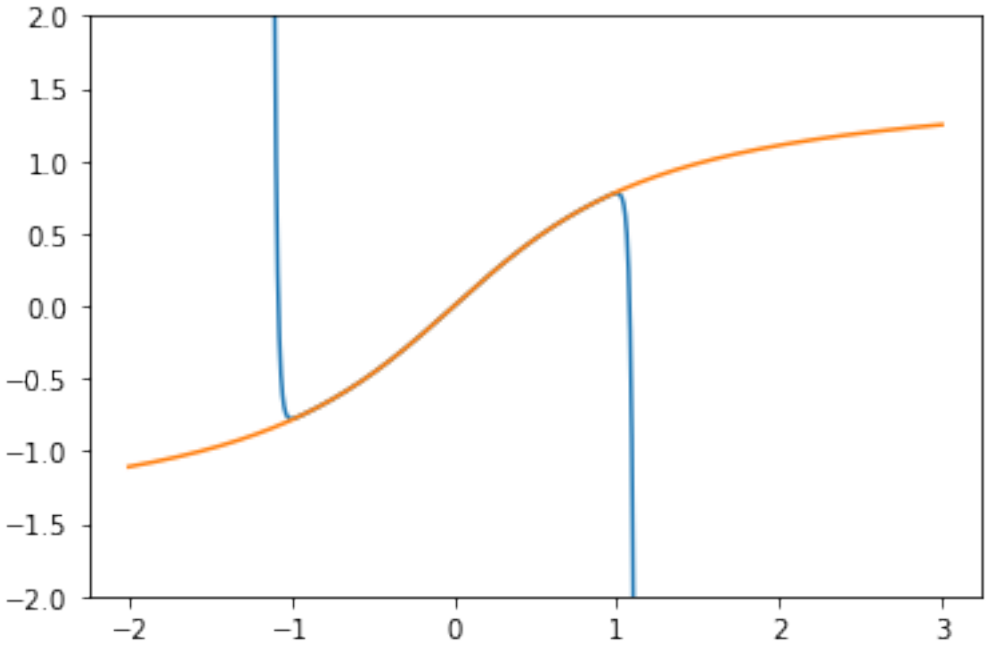
```
/local/schaedle/miniconda3/envs/py36/lib/python3.6/site-packages/matplotlib/figure.py:418: UserW
  "matplotlib is currently using a non-GUI backend, "
```



In [55]:
```
fig = plt.figure()
ax = fig.gca()
xn = np.linspace(-2,3,1000)
ax.plot(xn,lambdify(x,ar[60])(xn))
ax.plot(xn,np.arctan(xn))
ax.set_ylim(-2,2)
```

Out[55]:

$$(-2,\quad 2)$$




```
In [56]: series(atan(x),x,oo,12)
```

Out[56]:

$$\frac{1}{11x^{11}} - \frac{1}{9x^9} + \frac{1}{7x^7} - \frac{1}{5x^5} + \frac{1}{3x^3} - \frac{1}{x} + \frac{\pi}{2} + \mathcal{O}\left(\frac{1}{x^{12}}; x \to \infty\right)$$

```
In [57]: ai = series(atan(x),x,oo,12).removeO()

In [58]: xn2 = np.linspace(1/2,3,500)
         ax.plot(xn2,lambdify(x,ai)(xn2))

Out[58]: [<matplotlib.lines.Line2D at 0x7f8add5db5f8>]

In [59]: a1 = series(atan(x),x,1,12).removeO()

In [60]: xn1 = np.linspace(-2,3)
         ax.plot(xn1,lambdify(x,a1)(xn1))

Out[60]: [<matplotlib.lines.Line2D at 0x7f8add7132b0>]

In [61]: series(ln(x)**2/(1+x)**2,x,+oo)
```

Out[61]:

$$-\frac{4}{x^5}\log^2\left(\frac{1}{x}\right) + \frac{3}{x^4}\log^2\left(\frac{1}{x}\right) - \frac{2}{x^3}\log^2\left(\frac{1}{x}\right) + \frac{1}{x^2}\log^2\left(\frac{1}{x}\right) + \mathcal{O}\left(\frac{1}{x^6}\log^2\left(\frac{1}{x}\right); x \to \infty\right)$$

In [62]:
```
f = 1-cos(x**2)
g = x*(x-sin(x))
f.series(x,0,10), g.series(x,0,10)
```

Out[62]:

$$\left(\frac{x^4}{2} - \frac{x^8}{24} + \mathcal{O}\left(x^{10}\right), \quad \frac{x^4}{6} - \frac{x^6}{120} + \frac{x^8}{5040} + \mathcal{O}\left(x^{10}\right)\right)$$

In [63]:
```
p = f.series(x,0,10).removeO()
q = g.series(x,0,10).removeO()
p,q
```

Out[63]:

$$\left(-\frac{x^8}{24} + \frac{x^4}{2}, \quad \frac{x^8}{5040} - \frac{x^6}{120} + \frac{x^4}{6}\right)$$

In [64]:
```
b = cancel(p/q)
b
```

Out[64]:

$$-\frac{210x^4 - 2520}{x^4 - 42x^2 + 840}$$

In [65]:
```
b.subs(x,0)
```

Out[65]:

$$3$$

In [66]:
```
limit(f/g,x,0)
```

Out[66]:

$$3$$