lektion4

January 29, 2018

Table of Contents

1 Lektion 4

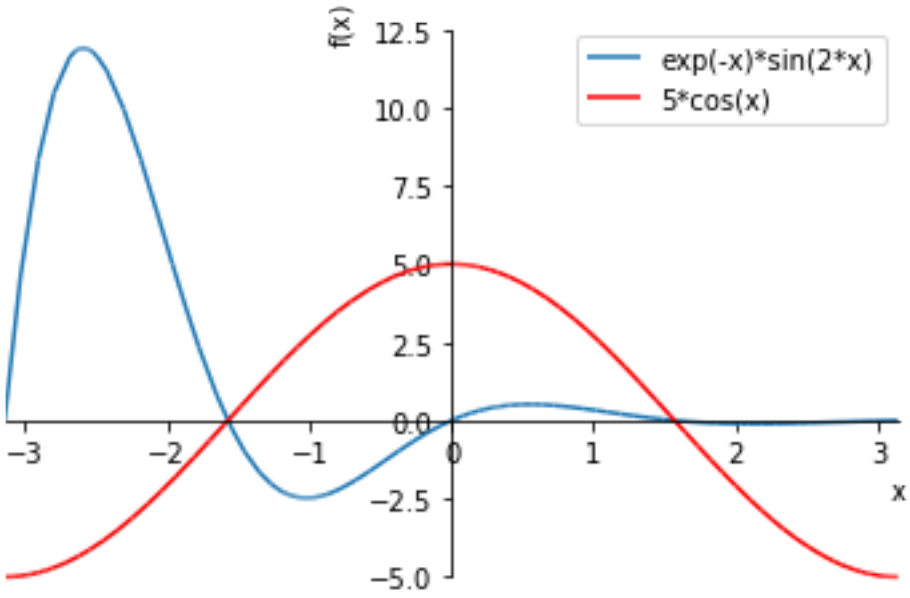
1.1 Graphik

```
In [1]: import matplotlib.pyplot as plt
        import sympy as sp
        import numpy as np

In [2]: #%matplotlib notebook
        %matplotlib inline
```

1.2 Sympy Graphik

1.2.1 Einfache Graphen von Funktionen

```
In [3]: x = sp.symbols('x')
        f = sp.exp(-x)*sp.sin(2*x)
        p1 = sp.plot(f,(x,-sp.pi,sp.pi),show=False)
        p2 = sp.plot(5*sp.cos(x),(x,-sp.pi,sp.pi),show=False)
        p1.extend(p2)
        p1.legend=True
        p1[1].line_color='red'
        p1.show()
        dir(p1)
```

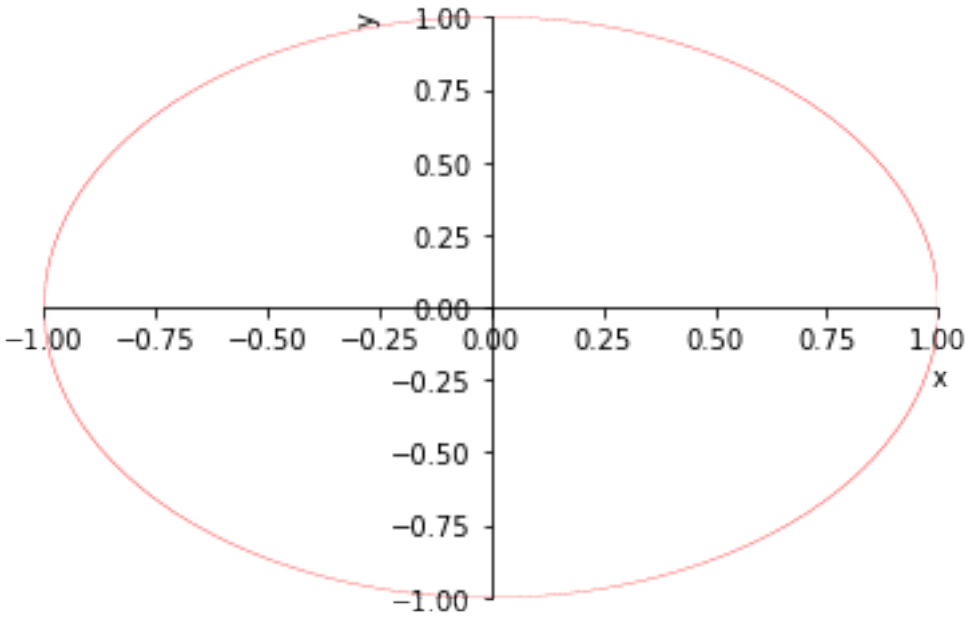



```
Out[3]: ['__class__',
         '__delattr__',
         '__delitem__',
         '__dict__',
         '__dir__',
         '__doc__',
         '__eq__',
         '__format__',
         '__ge__',
         '__getattribute__',
         '__getitem__',
         '__gt__',
         '__hash__',
         '__init__',
         '__init_subclass__',
         '__le__',
         '__lt__',
         '__module__',
         '__ne__',
         '__new__',
         '__reduce__',
         '__reduce_ex__',
         '__repr__',
         '__setattr__',
         '__setitem__',
         '__sizeof__',
```

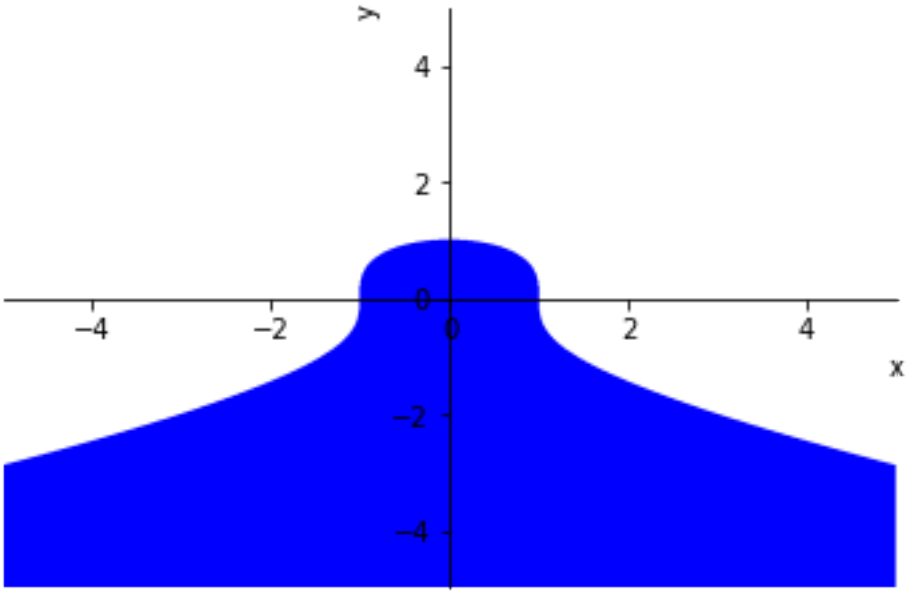
```
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_backend',
 '_series',
 'append',
 'aspect_ratio',
 'autoscale',
 'axis',
 'axis_center',
 'backend',
 'extend',
 'legend',
 'margin',
 'save',
 'show',
 'title',
 'xlabel',
 'xlim',
 'xscale',
 'ylabel',
 'ylim',
 'yscale']
```

1.2.2 Implizit gegebene Kurven

```
In [4]: x,y,z =sp.symbols('x y z')
        p3 = sp.plotting.plot_implicit(sp.Eq(x**2+y**2,1),(x,-1,1),(y,-1,1),line_color='red')
        ax = p3._backend.ax
        ax.set_aspect('equal')
```



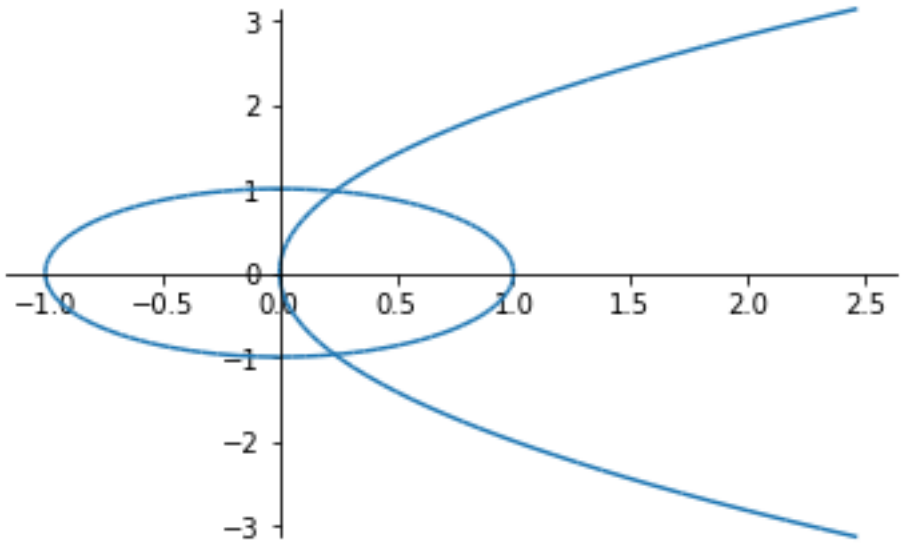
In [5]:
```
sp.plotting.plot_implicit(sp.Le(x**2+y**3,1))
```



Out[5]: <sympy.plotting.plot.Plot at 0x7fe052bb1b38>

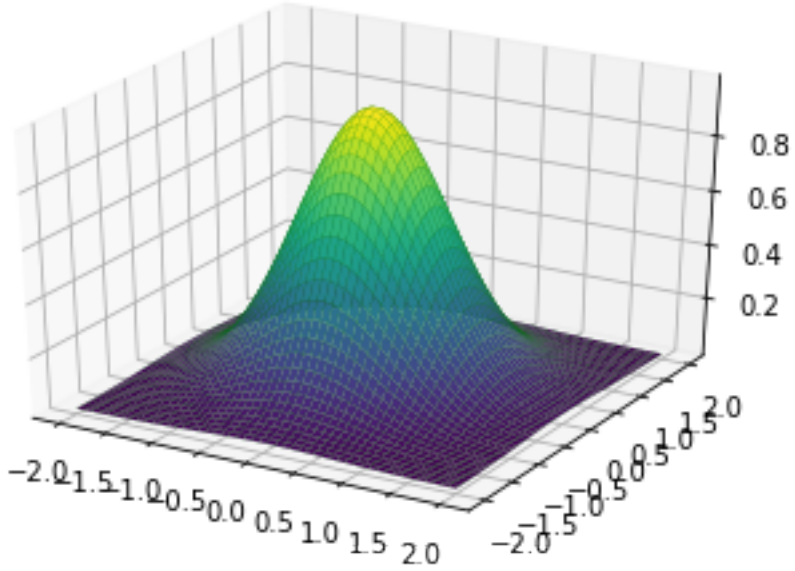
In [6]:
```
t = sp.symbols('t')
p1 = sp.plotting.plot_parametric((sp.sin(t),sp.cos(t)),((t/2)**2,t),(t,-sp.pi,sp.pi))
```

```
ax = p1._backend.ax
ax.set_aspect('equal')
```



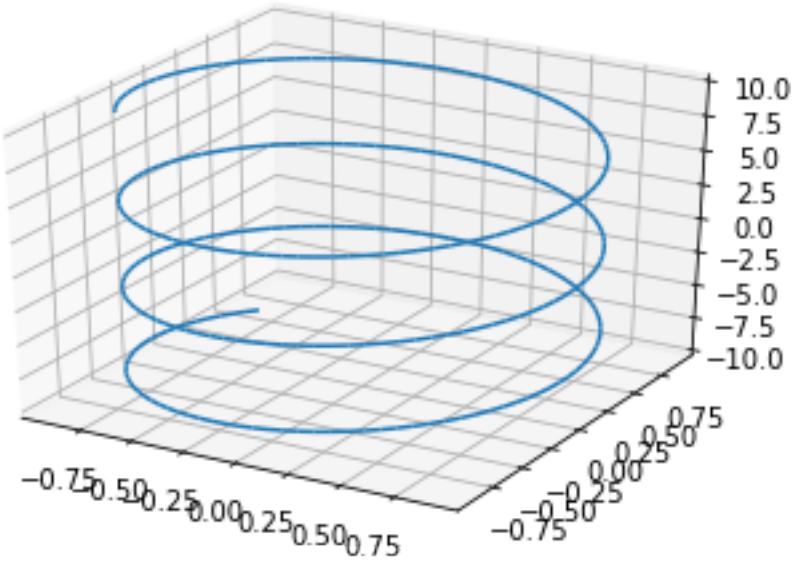
1.3 Sympy Graphik 3D

```
In [7]: sp.plotting.plot3d(sp.exp(-x**2-y**2),(x,-2,2),(y,-2,2),surface_color='green');
```



1.3.1 Parametrische Kurven

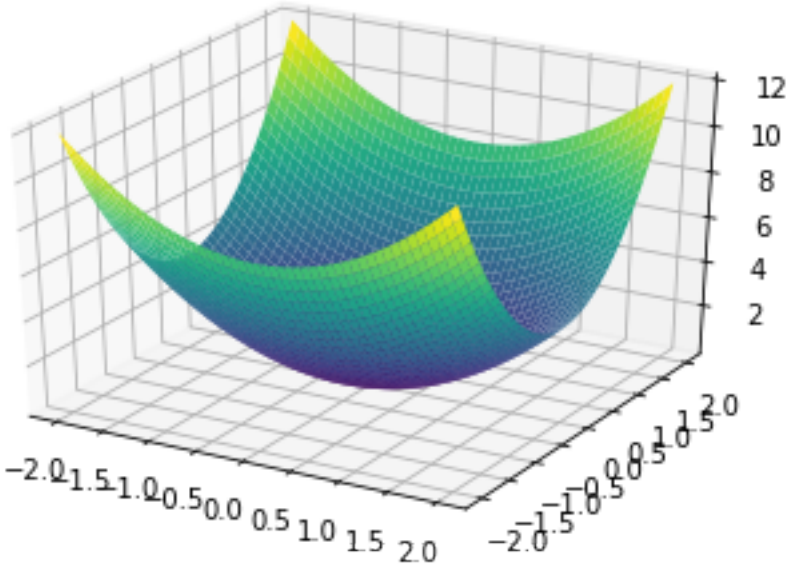
In [8]:
```
sp.plotting.plot3d_parametric_line(sp.cos(x),sp.sin(x),x,(x,-10,10))
```



Out[8]: <sympy.plotting.plot.Plot at 0x7fe0509e2cc0>

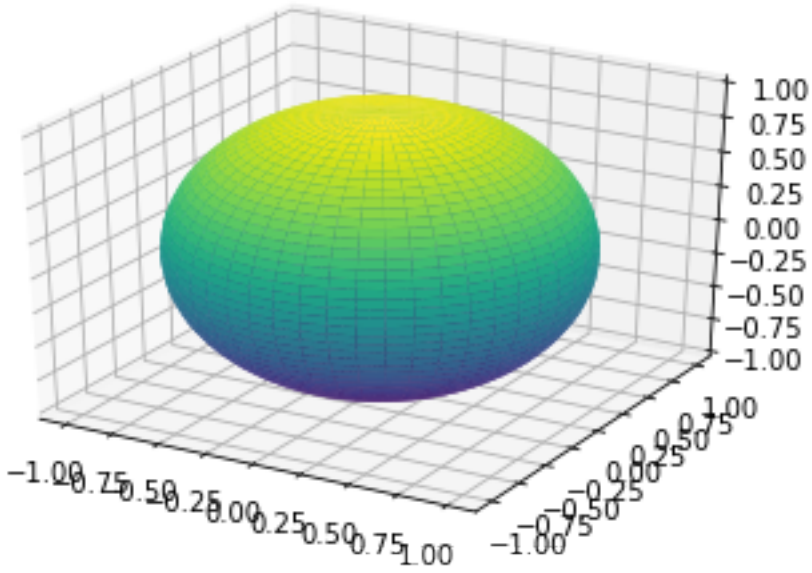
1.3.2 Parametrische Flaechen

In [9]:
```
sp.plotting.plot3d_parametric_surface(x,y,x**2+2*y**2,(x,-2,2),(y,-2,2))
```



```
Out[9]: <sympy.plotting.plot.Plot at 0x7fe0505e70b8>
```

```
In [10]: azimuth, elevation, radius = sp.symbols('azimuth elevation radius') # Kugel
         radius = 1
         p4 = sp.plotting.plot3d_parametric_surface(radius*sp.cos(elevation)*sp.cos(azimuth), \
                                                    radius*sp.cos(elevation)*sp.sin(azimuth), \
                                                    radius*sp.sin(elevation),(elevation,-sp.pi/2,sp.p
         ax = p4._backend.ax
         ax.set_aspect('equal')
```



1.4 Matplotlib Graphik

1.4.1 Einfache Kurven

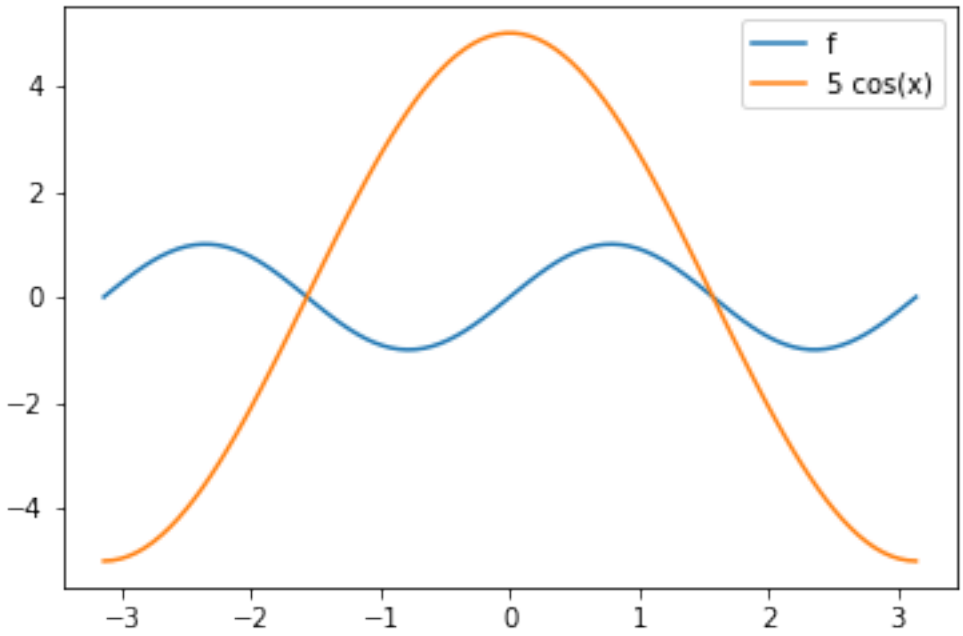
```
In [11]: f = sp.sin(2*x)
         fn = sp.lambdify(x,f,'numpy')
```

```
In [12]: gn = lambda x: 5*np.cos(x)
```

```
In [13]: xn = np.linspace(-np.pi,np.pi,100)
```

```
In [14]: plt.figure()
         plt.plot(xn,fn(xn),label='f')
         plt.plot(xn,gn(xn),label='5 cos(x)')
         plt.legend()
```

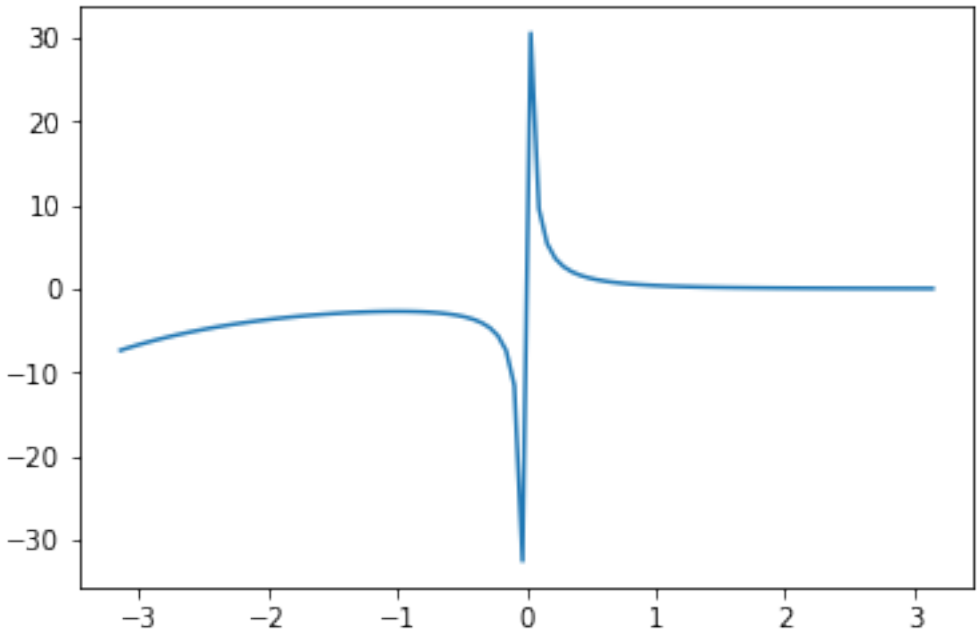
Out[14]: <matplotlib.legend.Legend at 0x7fe0505d1da0>



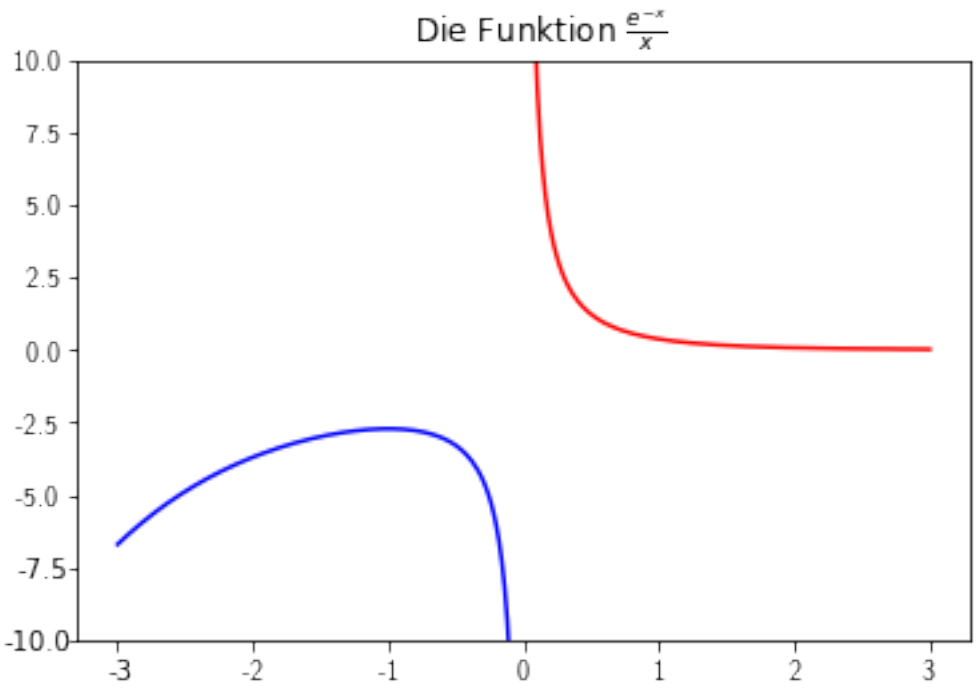
In [15]:
```
x = sp.Symbol('x')
h = sp.exp(-x)/x
hn = sp.lambdify(x,h)
```

In [16]:
```
plt.figure()
plt.plot(xn,hn(xn))
```

Out[16]: [<matplotlib.lines.Line2D at 0x7fe0504b2748>]

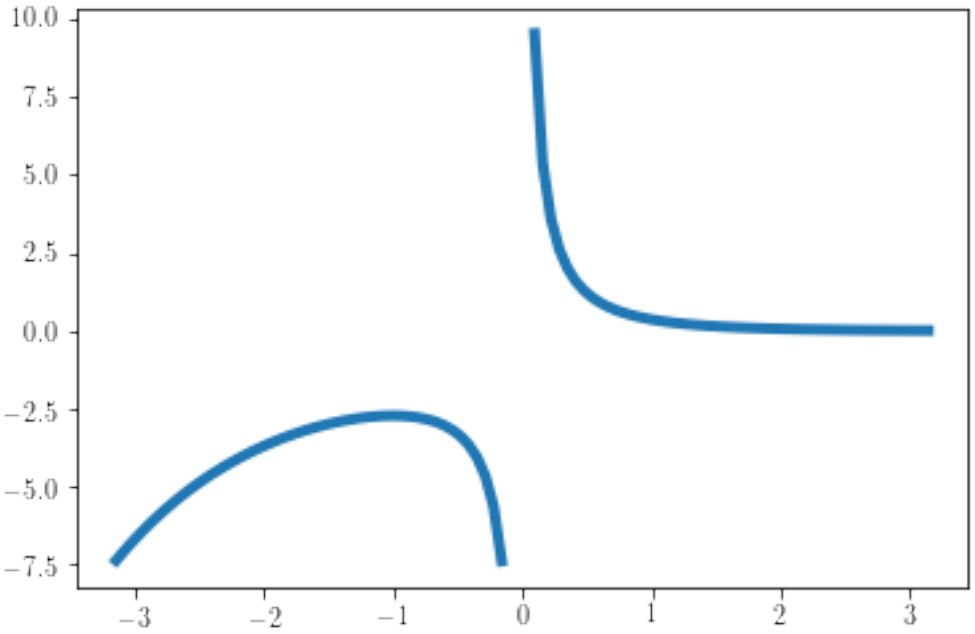



```
In [17]: xm = np.linspace(-3,-0.0001,150)
         xp = -xm
         plt.figure()
         plt.plot(xm,hn(xm),'b')
         plt.plot(xp,hn(xp),'r')
         plt.axis(ymin=-10,ymax=10)
         plt.rc('text',usetex=True)
         txt = sp.latex(h)
         plt.title(r'Die Funktion ${0:s}$'.format(txt))
         #plt.title(str(h))

Out[17]: Text(0.5,1,'Die Funktion $\\frac{e^{- x}}{x}$')
```

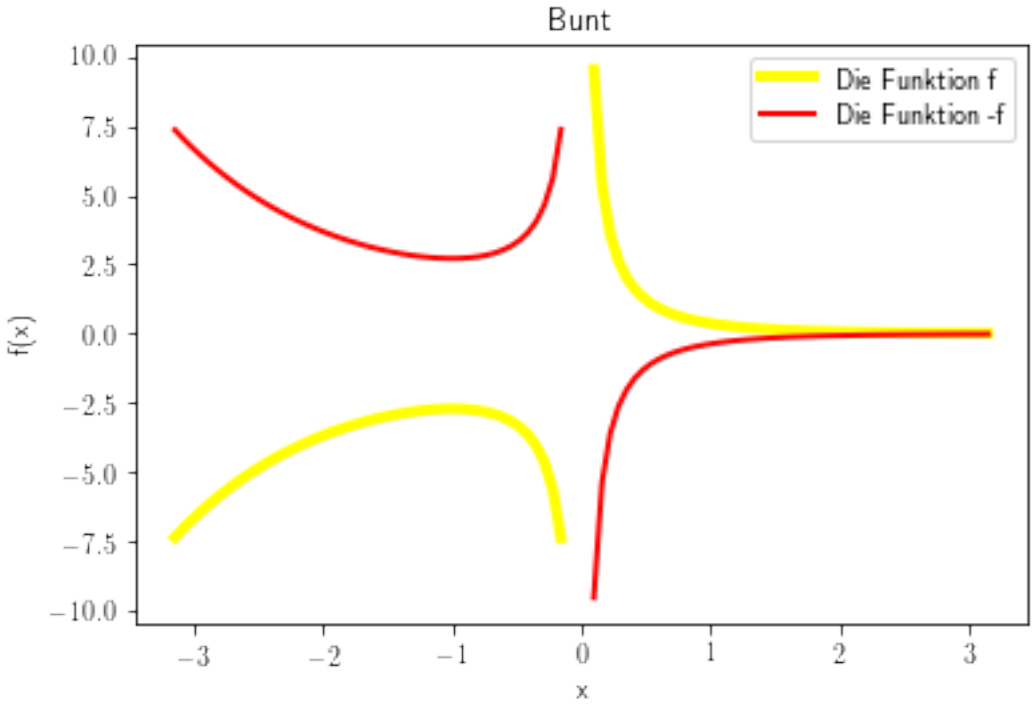


In [18]:
```
plt.figure()
yn = hn(xn)
yn[abs(yn)>10] = np.nan
plt.plot(xn,yn,linewidth=4)
```

Out[18]: [<matplotlib.lines.Line2D at 0x7fe0508fdf60>]

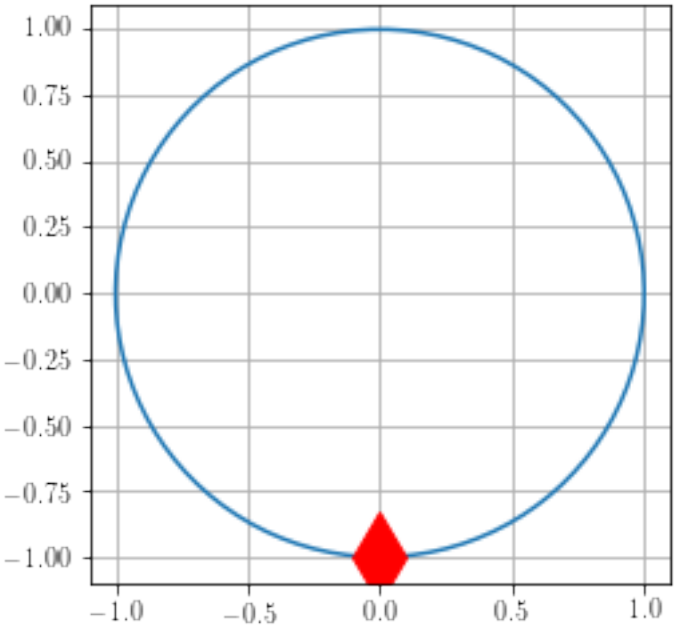


```
In [19]: fig = plt.figure()
         plt.plot(xn,yn,color='yellow',linewidth=4,label='Die Funktion f')
         plt.plot(xn,-yn,color='red',linewidth=2,label='Die Funktion -f')
         plt.ylabel('f(x)')
         plt.xlabel('x')
         plt.title('Bunt')
         plt.legend()
         plt.show()
```



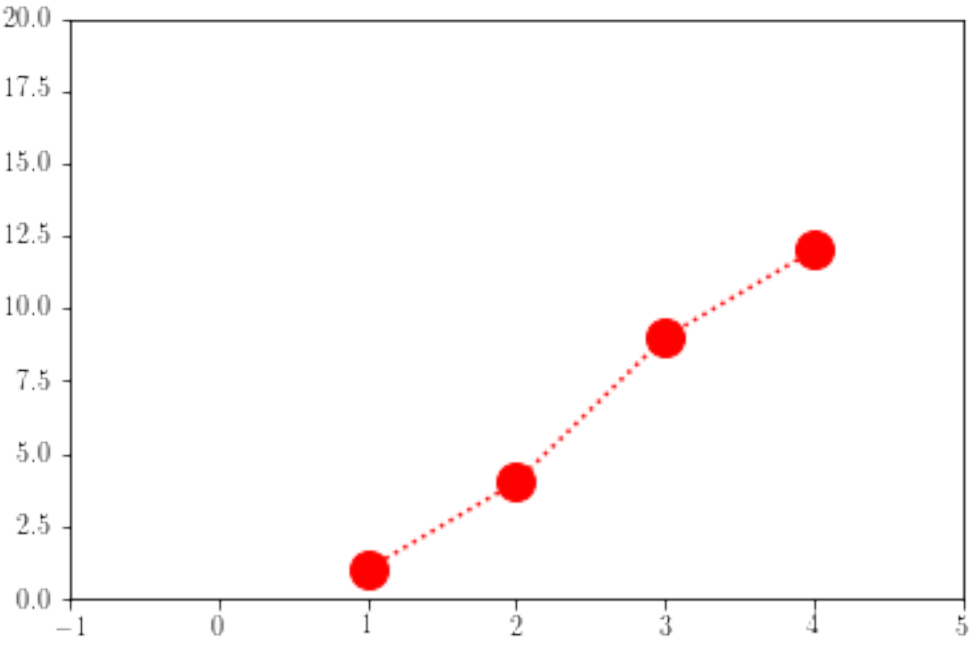
1.5 Matplotlib parametrische Kurven (plot)

```
In [20]: fig = plt.figure()
         ax = fig.gca()
         p1 = ax.plot(np.sin(xn),np.cos(xn))
         p2 = ax.plot(np.sin(xn[0]),np.cos(xn[0]),'rd',MarkerSize=24)

         ax.set_aspect('equal')

         plt.grid()
         plt.show()
```



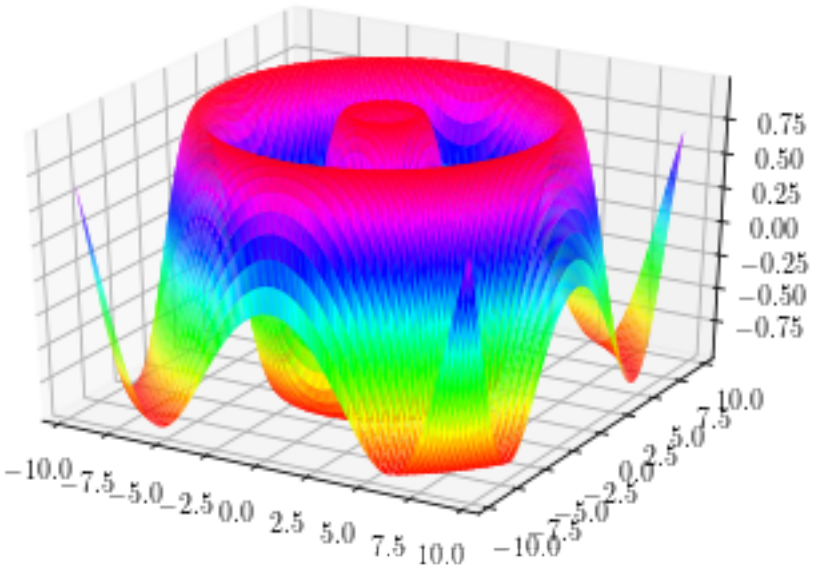

```
In [21]: plt.figure()
         plt.plot([1, 2, 3, 4],[1, 4, 9, 12],'ro:',MarkerSize=14)
         # Farbe: rgbcm
         # Marker: xo+*
         plt.axis([-1,5,0,20])
```

```
Out[21]: [-1, 5, 0, 20]
```

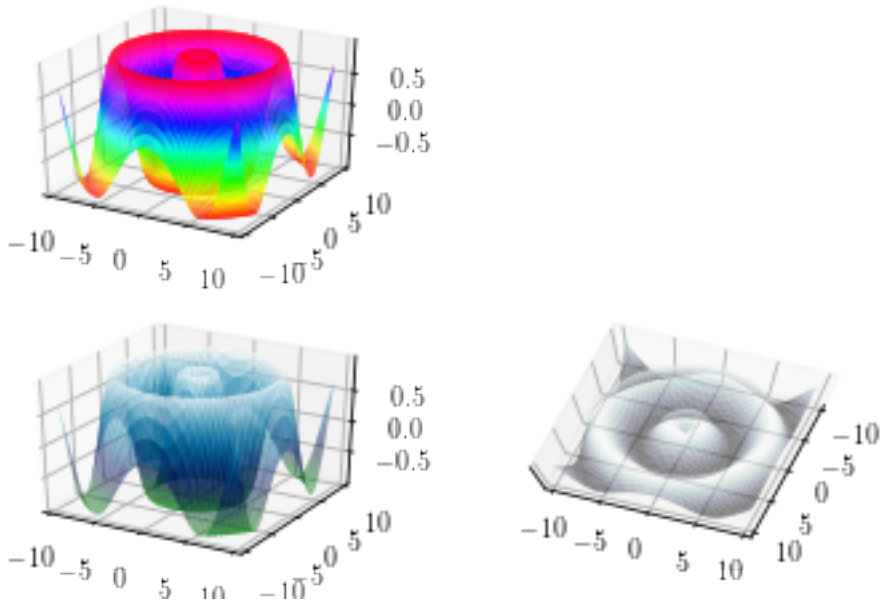


1.6 Matplotlib 3D Graphik (surface wireframe)

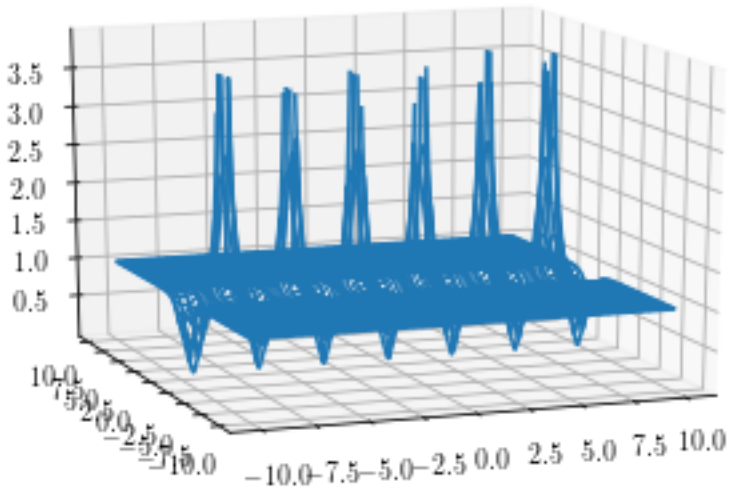
```
In [22]: from mpl_toolkits.mplot3d import Axes3D

In [23]: x = sp.Symbol('x')
         y = sp.Symbol('y')
         f = sp.sin(sp.sqrt(x**2+y**2))
         f

Out[23]: sin(sqrt(x**2 + y**2))

In [24]: fn = sp.lambdify((x,y), f)   # probehalber 'numpy' weglassen
         f.subs(x,1.).subs(y,2.), fn(1,2)

Out[24]: (0.786749131547214, 0.786749131547214)

In [25]: xn = np.linspace(-3*np.pi, 3*np.pi,100)
         yn = np.linspace(-3*np.pi, 3*np.pi,100)
         X, Y = np.meshgrid(xn, yn)
         X.shape

         a = np.linspace(-1,1,3)
         b = np.linspace(-1,1,5)
         a
         AA, BB = np.meshgrid(a,b)
         display(AA,BB)

array([[-1.,  0.,  1.],
       [-1.,  0.,  1.],
       [-1.,  0.,  1.],
       [-1.,  0.,  1.],
       [-1.,  0.,  1.]])


array([[-1. , -1. , -1. ],
       [-0.5, -0.5, -0.5],
       [ 0. ,  0. ,  0. ],
       [ 0.5,  0.5,  0.5],
       [ 1. ,  1. ,  1. ]])


In [26]: fig = plt.figure()
         ax = fig.add_subplot(111, projection='3d')
         Z = fn(X, Y)
         s1 = ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                              cmap=plt.cm.hsv, linewidth=0);
```




```
In [27]: fig = plt.figure()
         ax1 = fig.add_subplot(221,projection='3d')
         ax1.plot_surface(X, Y, Z, rstride=1, cstride=1,
                          cmap=plt.cm.hsv, linewidth=1,
                          alpha=1);
         ax2 = fig.add_subplot(223,projection='3d')
         ax2.plot_surface(X, Y, Z,
                          cmap=plt.cm.ocean, linewidth=1,
                          alpha=0.5);
         ax4 = fig.add_subplot(224,projection='3d')
         ax4.plot_surface(X, Y, Z,
                          cmap=plt.cm.bone, linewidth=1,
                          alpha=0.5);
         ax4.view_init(80,20)
         ax4.set_zticks([])

Out[27]: []
```




```
In [28]: f = abs(sp.tan(x+sp.I*y))
         fn = sp.lambdify((x,y),f,'numpy')

In [29]: xn = np.linspace(-10, 10, 400)
         yn = xn
         X, Y = np.meshgrid(xn, yn)
         Z=fn(X,Y)
         Z[Z>4]=np.nan

In [ ]: fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        ax.plot_wireframe(X, Y, Z)
        ax.view_init(15,-110)
```




```
In [ ]: fig = plt.figure()
        from matplotlib.colors import Normalize
        Normierung = Normalize(0,2)
        ax = fig.add_subplot(111, projection='3d')
        ax.plot_surface(X, Y, Z,cmap=plt.cm.hsv,linewidth=0,rstride=1,cstride=1,norm=Normierung)
        ax.view_init(15,-110)
```