

# 01

November 8, 2022

Dieses Notebook kann als Vorlage für die Plots von Blatt 1 benutzt werden.

```
[ ]: # was hier passiert, ist nicht so wichtig, sondern soll nur eine drehbare ↵  
      ↪ 3D-Ansicht ermöglichen  
      %matplotlib widget  
      import matplotlib.pyplot as plt  
      import numpy as np
```

```
[ ]: ## es ist oft eine gute idee, teile eines programms, die man immer wieder ↵  
      ↪ benutzt, in funktionen auszulagern
```

```
from typing import Callable
```

```
def curve3D( t : list[float], x_t : Callable[[float], float], y_t : ↵  
            ↪ Callable[[float], float], z_t : Callable[[float], float] ):  
    """A 3D trajectory  
  
    Arguments:  
    t (array): parameter for the trajectory  
  
    Returns:  
    A list of three elements, the (x,y,z)-points of the trajectory  
    """  
    return [[x_t(e) for e in t], [y_t(e) for e in t], [z_t(e) for e in t]]
```

```
def curve2D( t : list[float], x_t : Callable[[float], float], y_t : ↵  
            ↪ Callable[[float], float] ):  
    """A 2D trajectory  
  
    Arguments:  
    t (array): parameter for the trajectory  
    x_t (Function): function along x  
    y_t (Function): function along y  
  
    Returns:  
    A list of two elements, the (x,y)-points of the trajectory
```

```

"""
return [[x_t(e) for e in t], [y_t(e) for e in t]]

```

```

[ ]: ## "linspace" erzeugt ein array mit 100 einträgen von 0 bis 10 in jeweils
      ↪ gleichem abstand
parameter_range = np.linspace(0, 10, 100)

```

```

[ ]: ## curve2D gibt eine liste mit zwei elementen zurück, die wir separat
      ↪ "auspacken" können

from math import sin, cos, pi

xdata, ydata = curve2D(parameter_range, lambda x : cos(x), lambda y : cos( y +
      ↪ pi / 2 ) )
fig = plt.figure('Kurven 2D')
ax = plt.axes()
plt.subplot(2,2,1)
plt.plot( xdata, ydata, 'o' )
plt.title('(i)')
ax.annotate( 'start', (xdata[0], ydata[0]))
ax.annotate( 'end', (xdata[-1], ydata[-1]))
ax.set_xlabel('X')
ax.set_ylabel('Y')

xdata, ydata = curve2D(parameter_range, lambda x : cos(x), lambda y : cos( y +
      ↪ pi / 6 ) )
ax = plt.axes()
plt.subplot(2,2,2)
plt.plot( xdata, ydata, 'o' )
plt.title('(ii)')
ax.annotate( 'start', (xdata[0], ydata[0]))
ax.annotate( 'end', (xdata[-1], ydata[-1]))
ax.set_xlabel('X')
ax.set_ylabel('Y')

xdata, ydata = curve2D(parameter_range, lambda x : cos(x), lambda y : cos( 2 *
      ↪ y + pi / 2 ) )
ax = plt.axes()
plt.subplot(2,2,3)
plt.plot( xdata, ydata, 'o' )
plt.title('(iii)')
ax.annotate( 'start', (xdata[0], ydata[0]))
ax.annotate( 'end', (xdata[-1], ydata[-1]))
ax.set_xlabel('X')
ax.set_ylabel('Y')

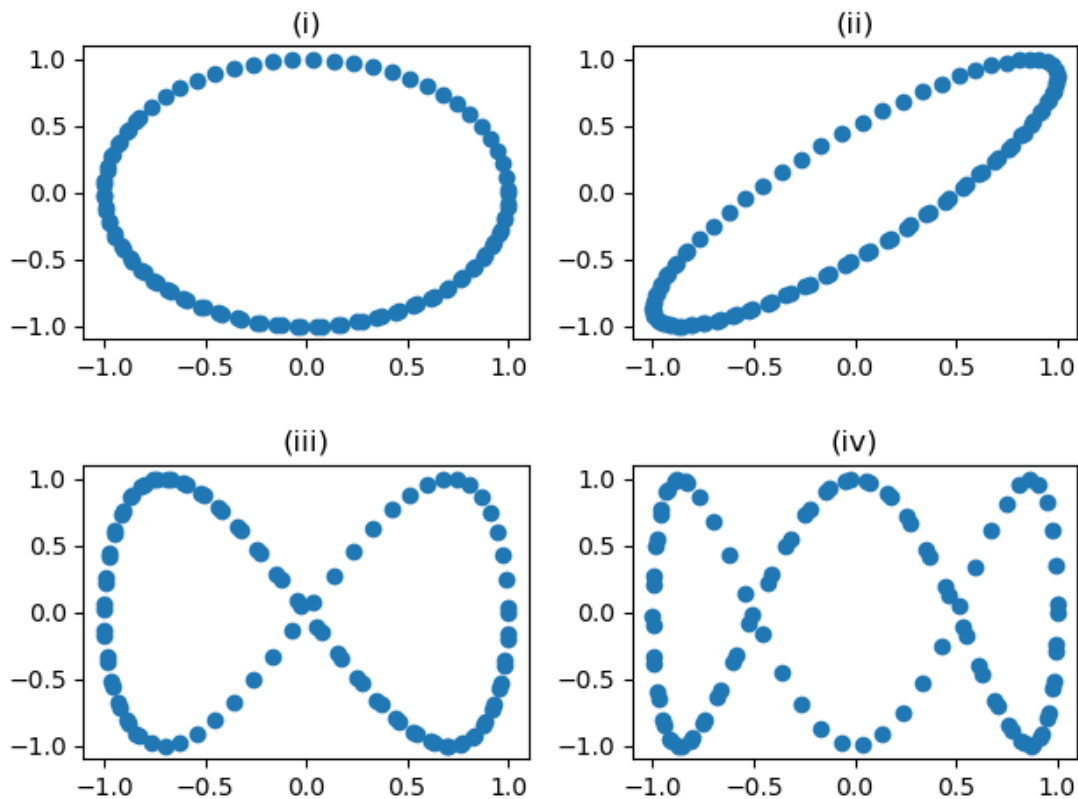
```

```

xdata, ydata = curve2D(parameter_range, lambda x : cos(x), lambda y : cos( 3 *
    ↪ y + pi / 2 ) )
ax = plt.axes()
plt.subplot(2,2,4)
plt.plot( xdata, ydata, 'o' )
plt.title('(iv)')
ax.annotate( 'start', (xdata[0], ydata[0]))
ax.annotate( 'end', (xdata[-1], ydata[-1]))
ax.set_xlabel('X')
ax.set_ylabel('Y')

fig.tight_layout(h_pad=2)
plt.show()

```



```

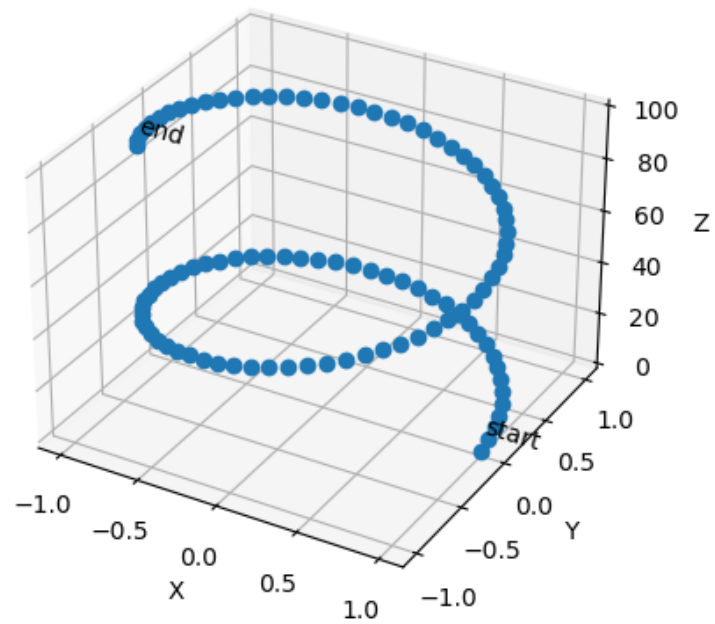
[ ]: ## curve3D gibt eine liste mit drei elementen zurück, die wir separat
    ↪ "auspacken" können
xdata, ydata, zdata = curve3D(parameter_range, lambda x: cos(x), lambda y :
    ↪ sin(y), lambda z : 10* z)
fig = plt.figure('Kurve 3D')

```

```

ax = plt.axes(projection='3d')
ax.plot( xdata, ydata, zdata, 'o' )
ax.text( xdata[0], ydata[0], zdata[0], 'start', 'x' )
ax.text( xdata[-1], ydata[-1], zdata[-1], 'end', 'x' )
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()

```



[ ]: