Institute of Mathematics and Image Computing, Lübeck

MA4030: Optimierung, Sommersemester 2025

Übungsblatt 2

Abgabe von: *Fynn-Ole Claussen, 770712 und Johann Fischer, 779072 und Mika Kohlhammer, 779098*

A2\code\P4.py

```
## Hier werden sÃ¤mtliche in dieser Datei benÃ¶tigten Pakete und Module geladen
import numpy as np
import matplotlib.pyplot as plt


## TODO: Aufgabenteil 4a. Dimensionsumformungen eines Vektors
print('### 4a ###')
# BEGIN SOLUTION
v = np.array(range(12))
v=v.reshape(3, 4)
print('v(3,4) =\n',v,'\n')
v=v.reshape(4, 3)
print('v(4,3) =\n',v,'\n')
v=np.ndarray.flatten(v)
print('v(12,) =\n',v,'\n')
# END SOLUTION


## TODO: Aufgabenteil 4b. Logische Indizierung bei Vektoren
print('\n### 4b ###')
# BEGIN SOLUTION
v1 = np.array([],dtype=int) #Erstelle Arrays zum Speichern
v2 = np.array([],dtype=int)
for i in range(len(v)):
    if v[i]>3:                      #Falls das i-te Element in v >3 wird dies in v1 gespeichert.
        v1=np.append(v1,v[i])
print('v_i aus v mit v_i>3 : ',v1,'\n')
for i in range(len(v)):
    if v[i]<7:                      # Analog zu vorher mit 'i-tes Element <7'
        v2=np.append(v2,v[i])
print('v_i aus v mit v_i<7 : ',v2,'\n')
# END SOLUTION


## TODO: Aufgabenteil 4c. Einheitsmatrix erzeugen
print('\n### 4c ###')
# BEGIN SOLUTION
C1 = np.eye(100)
C2 = np.diag(np.ones(100))
print('C1=C2 : ', np.array_equal(C1, C2))
# END SOLUTION


## TODO: Aufgabenteil 4d. Hauptminor der Einheitsmatrix visualisieren
# BEGIN SOLUTION
```

```
C3 = C1[0:9,0:9]
plt.spy(C3)
plt.show()
# END SOLUTION


## TODO: Aufgabenteil 4e. exp und log von ndarrays
print('\n### 4e ###')
A = np.array([[1, 0, 3, 4], [3, 1, 1, 0], [0, -1, 2, 3], [1, 0, 0, -1]])
B = np.array([[8, 10, -13, 6], [5, 5.5, -1, -4], [4, 6.5, -11, 7], [0, 0, 2, -2]])
# BEGIN SOLUTION
print('exp(A) =\n',np.exp(A),'\n')
print('log(A) =\n',np.log(A),'\n')
print('Die Funktionen wenden Komponentenweise die Exponential-Funktion bzw. den Logarithmus auf die Matrix
      'Dabei tritt das Problem auf, dass log(0) bzw allgemien log(x) fÃ¼r x<=0 nicht definiert ist.\n'
      'Hier wird "-inf" bzw. "nan" (not a number) ausgegeben')
# END SOLUTION


## TODO: Aufgabenteil 4f. Zeile und Spalte ausgeben
print('\n### 4f ###')
# BEGIN SOLUTION
print('Erste Spalte von B: \n',B[:,0])
print('Erste Zeile von B: \n',B[0,:])
# END SOLUTION


## TODO: Aufgabenteil 4g. LGS lÃ¶sen
print('\n### 4g ###')
# BEGIN SOLUTION
print('Da det(A) =',np.linalg.det(A),'ungleich 0 ist, ist A invertierbar.\n')

X = np.zeros((4,4))
max_residuen = []
for i in range(4):
    b_i = B[:,i]
    x_i = np.linalg.inv(A)@b_i
    X[:,i] = x_i
    residuen = A@x_i-b_i
    max_residuen.append(float(np.max(np.abs(residuen))))
print('i) \nX = \n',X,'\nDas Ergebnis der Maxinumsnorm der Resdiuen der Spalten : ',max_residuen,'\n')

X2 = np.linalg.solve(A,B)
#X3 = np.linalg.inv(A)@B #berechnet Numerisches Ergebnis wie in Aufgabenteil i)
print('ii) \nX = \n',X2)
Residuum = np.linalg.norm(A@X2-B,np.inf)
print('Das BetragsmÃ¤ÃŸig grÃ¶ÃŸte Element der Residualmatrix ist: ',Residuum,'\n')
# END SOLUTION

#TERMINAL OUTPUT:
```

4a

v(3,4) =

[[0 1 2 3]

[4 5 6 7]

[8 9 10 11]]

v(4,3) =

[[0 1 2]

[3 4 5]

[6 7 8]

[9 10 11]]

v(12,) =

[0 1 2 3 4 5 6 7 8 9 10 11]

4b

v_i aus v mit v_i>3 : [4 5 6 7 8 9 10 11]

v_i aus v mit v_i<7 : [0 1 2 3 4 5 6]

4c

C1=C2 : True

4e

exp(A) =

[[2.71828183 1. 20.08553692 54.59815003]

[20.08553692 2.71828183 2.71828183 1.]

[1. 0.36787944 7.3890561 20.08553692]

[2.71828183 1. 1. 0.36787944]]

log(A) =

[[0. -inf 1.09861229 1.38629436]

[1.09861229 0. 0. -inf]

[-inf nan 0.69314718 1.09861229]

[0. -inf -inf nan]]

Die Funktionen wenden Komponentenweise die Exponential-Funktion bzw. den Logarithmus auf die Matrix an.

Dabei tritt das Problem auf, dass log(0) bzw allgemien log(x) für x<=0 nicht definiert ist.

Hier wird "-inf" bzw. "nan" (not a number) ausgegeben

4f

Erste Spalte von B:

[8. 5. 4. 0.]

Erste Zeile von B:

[8. 10. -13. 6.]

4g

Da det(A) = 2.9999999999999996 ungleich 0 ist, ist A invertierbar.

i)

X =

[[1.00000000e+00 2.00000000e+00 -1.00000000e+00 -1.00000000e+00]

[1.00000000e+00 -5.00000000e-01 2.00000000e+00 -2.00000000e+00]

[1.00000000e+00 -1.77635684e-15 2.66453526e-15 1.00000000e+00]

[1.00000000e+00 2.00000000e+00 -3.00000000e+00 1.00000000e+00]]

Das Ergebnis der Maxinumsnorm der Resdiuen der Spalten : [5.329070518200751e-15, 5.329070518200751e-15, 7.105427357601002e-15, 2.6645352591003757e-15]

ii)

X =

[[1. 2. -1. -1.]

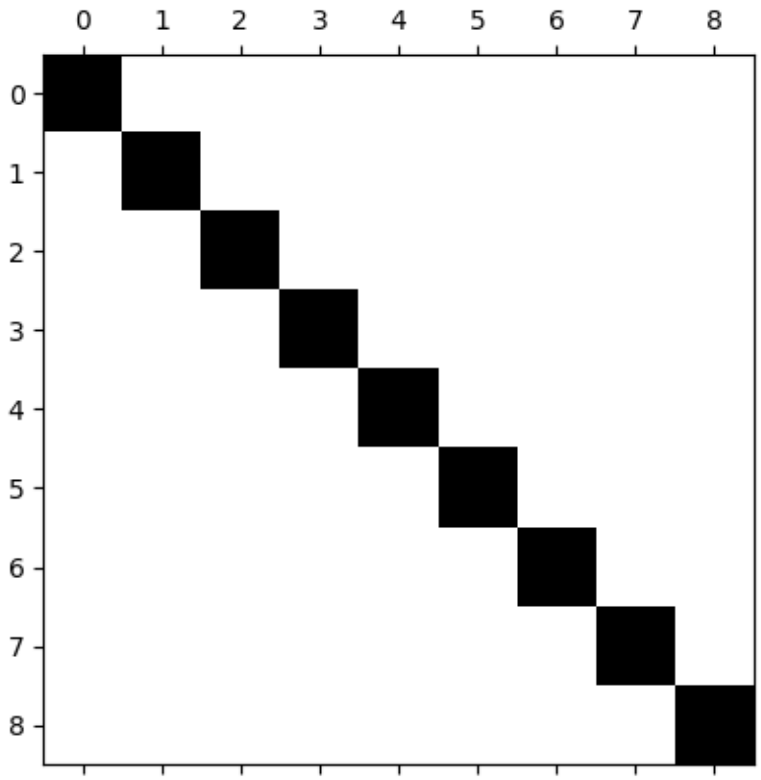
[1. -0.5 2. -2.]

[1. 0. 0. 1.]

[1. 2. -3. 1.]]

Das Betragsmäßig größte Element der Residualmatrix ist: 0.0

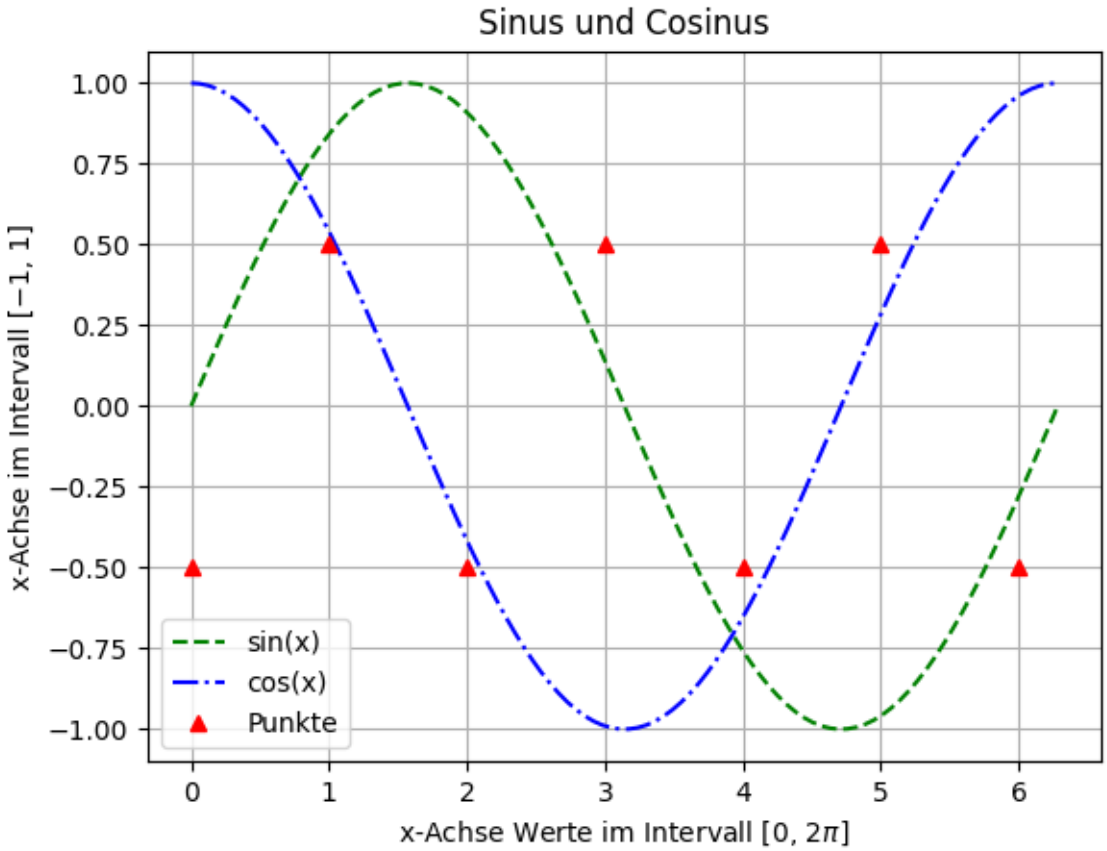
```
#PLOTS:
```

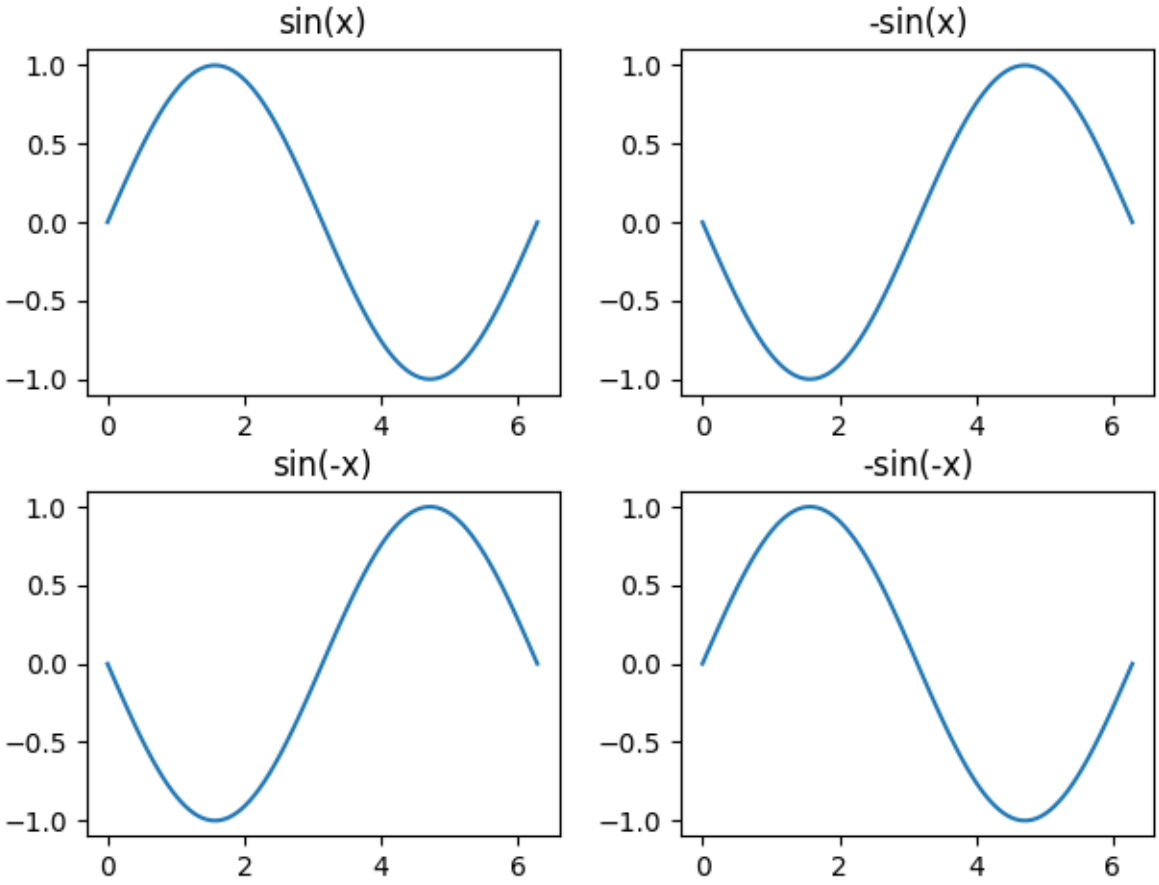


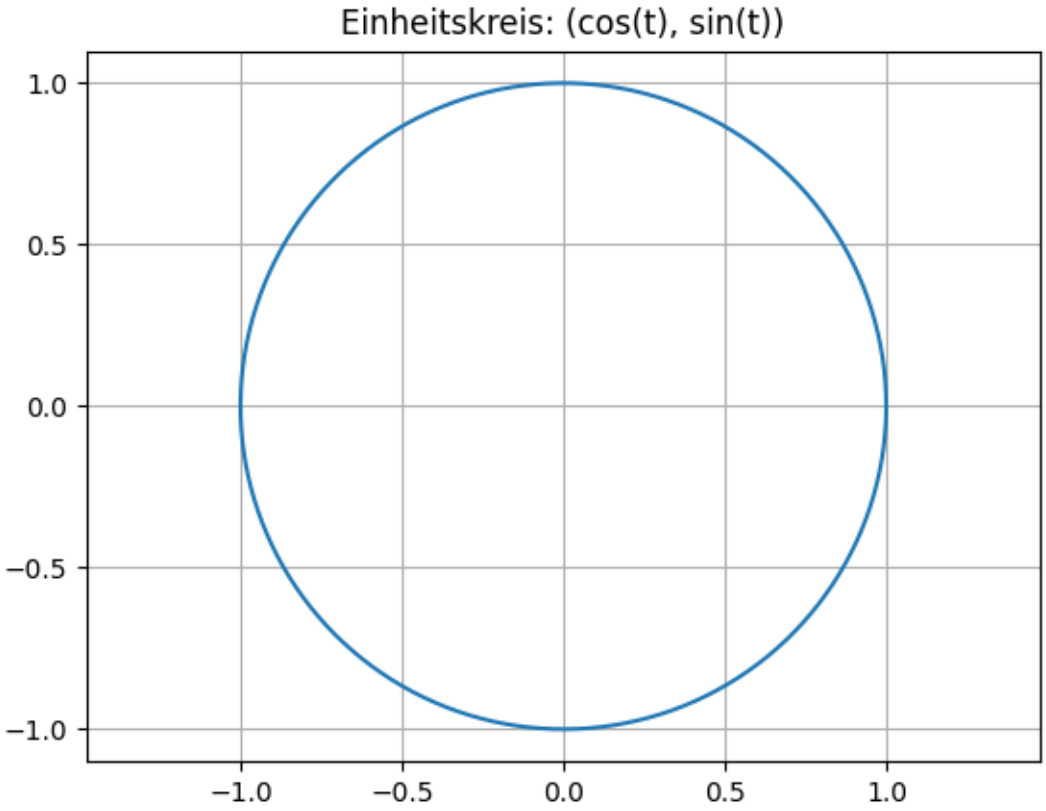
A2\code\P5.py

```
import numpy as np
import matplotlib.pyplot as plt


## TODO: Aufgabenteil 5a. Sinus- und Kosinus-Plots erstellen
# Plot 1
x = np.linspace(0, 2*np.pi,100)  # Gitterpunkte
plt.plot(x, np.sin(x), 'g--', label='sin(x)')  # Sinus

# ErgÃ¤nzen Sie die weiteren Bestandteile des Plots
# BEGIN SOLUTION
plt.plot(x,np.cos(x),'b-.',label='cos(x)') #Cosinus
# Zusaetzliche Punkte:
punkte_x = [i for i in range(7)]
punkte_y = [(-1)**(i+1) * 0.5 for i in range(7)]
plt.plot(punkte_x,punkte_y,'r^',label='Punkte')
#Kosmetische Anpassungen:
plt.xlabel('x-Achse Werte im Intervall [0, 2'r'$\pi$'']')
plt.ylabel('x-Achse im Intervall [â^'1, 1]')
plt.title('Sinus und Cosinus')
plt.legend()
plt.grid(True)
# END SOLUTION
plt.show()

# Plot 2
fig, ax = plt.subplots(nrows=2, ncols=2)
fig.tight_layout(pad=1.5)

ax[0, 0].plot(x, np.sin(x))
ax[0, 0].set_title('sin(x)')

# ErgÃ¤nzen Sie die weiteren Bestandteile des Plots
# BEGIN SOLUTION
ax[0, 1].plot(x, np.sin(x)*(-1))
ax[0, 1].set_title('-sin(x)')

ax[1, 0].plot(x, np.sin(-x))
ax[1, 0].set_title('sin(-x)')

ax[1, 1].plot(x, -np.sin(-x))
ax[1, 1].set_title('-sin(-x)')
# END SOLUTION
plt.show()


## TODO: Aufgabenteil 5b. Einheitskreis plotten
# BEGIN SOLUTION
plt.close('all')
t = np.linspace(0, 2*np.pi, 1000)   # Diskretierungspunkte
plt.plot(np.cos(t),np.sin(t))   # Plotten des Einheitskreises
plt.title('Einheitskreis: (cos(t), sin(t))')
#Anpassen der Achsen
```

```
plt.grid(True)
plt.xticks(np.arange(-1, 1.1, 0.5))
plt.yticks(np.arange(-1, 1.1, 0.5))
plt.axis('equal')
plt.show()
# END SOLUTION

#TERMINAL OUTPUT:

#PLOTS:
```




sin(x)
-sin(x)
sin(-x)
-sin(-x)
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6


Einheitskreis: (cos(t), sin(t))
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0