lektion10

December 12, 2019

Table of Contents

1 Lektion 10

Lineare Algebra Teil 2

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
from sympy import *
init_printing()
x,y,z = symbols('x y z')
%matplotlib inline
```

1.1 Eigenwerte, Eigenvektoren, Jordannormalform

[2]:
```
A = Matrix(3, 3, lambda i, j : abs(i-j))
A[-1,0] = 0
A
```

[2]:
$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

[4]:
```
A.eigenvals()
```

[4]:
$$\left\{ \left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right) \sqrt[3]{\frac{\sqrt{57}}{9} + 1} + \frac{2}{3\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right) \sqrt[3]{\frac{\sqrt{57}}{9} + 1}} : 1, \quad \frac{2}{3\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right) \sqrt[3]{\frac{\sqrt{57}}{9} + 1}} + \left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right) \sqrt[3]{\frac{\sqrt{57}}{9} + 1} \right.$$

[5]:
```
A.eigenvals(simplify=True)
```

[5]:
$$\left\{ \frac{\sqrt[3]{3}\left(2\sqrt[3]{3} + \left(\sqrt{57}+9\right)^{\frac{2}{3}}\right)}{3\sqrt[3]{\sqrt{57}+9}} : 1, \quad -\frac{\sqrt[3]{3}\left(8\sqrt[3]{3} + \left(1-\sqrt{3}i\right)^2\left(\sqrt{57}+9\right)^{\frac{2}{3}}\right)}{6\left(1-\sqrt{3}i\right)\sqrt[3]{\sqrt{57}+9}} : 1, \quad -\frac{\sqrt[3]{3}\left(8\sqrt[3]{3} + \left(1+\sqrt{3}i\right)^2\left(\sqrt{57}+9\right)\right.}{6\left(1+\sqrt{3}i\right)\sqrt[3]{\sqrt{57}+9}} \right.$$

[6]:
```
A[-1,0] = 1 # etwas übersichtlicher
A.eigenvals()
```

[6]:
$$\left\{ -1 : 1, \quad \frac{1}{2} - \frac{\sqrt{13}}{2} : 1, \quad \frac{1}{2} + \frac{\sqrt{13}}{2} : 1 \right\}$$

[7]:
```
elvs = A.eigenvects()
elvs
```

[7]:
$$\left[\left(-1, \ 1, \ \left[\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}\right]\right), \ \left(\frac{1}{2} - \frac{\sqrt{13}}{2}, \ 1, \ \left[\begin{bmatrix} -\frac{\frac{\sqrt{13}}{2} + \frac{1}{2} + 2\left(-\frac{1}{2} + \frac{\sqrt{13}}{2}\right)^2}{\left(-1 + \left(-\frac{1}{2} + \frac{\sqrt{13}}{2}\right)^2\right)\left(-\frac{1}{2} + \frac{\sqrt{13}}{2}\right)} \\ -\frac{-\frac{5}{2} + \frac{\sqrt{13}}{2}}{-1 + \left(-\frac{1}{2} + \frac{\sqrt{13}}{2}\right)^2} \\ 1 \end{bmatrix}\right]\right), \ \left(\frac{1}{2} + \frac{\sqrt{13}}{2}, \ 1, \ \left[\left[-\frac{\frac{1}{2} + \frac{\sqrt{13}}{2}}{\left(-1 + \left(-\frac{\sqrt{13}}{2}\right.\right.\right.\right.\right.$$

[11]:
```
elvs[0][2][0]
```

[11]:
$$\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

[13]:
```
n = len(elvs)
m = len(elvs[0][2][0])
L = zeros(n, n)
```

```
T  = zeros(m, n)
for k, elv in enumerate(elvs):
    L[k, k] = elv[0]
    T[:, k] = elv[2][0]
T, L
```

[13]:
$$\left(\begin{bmatrix} -1 & -\frac{-\frac{\sqrt{13}}{2}+\frac{1}{2}+2\left(-\frac{1}{2}+\frac{\sqrt{13}}{2}\right)^2}{\left(-1+\left(-\frac{1}{2}+\frac{\sqrt{13}}{2}\right)^2\right)\left(-\frac{1}{2}+\frac{\sqrt{13}}{2}\right)} & -\frac{\frac{1}{2}+\frac{\sqrt{13}}{2}+2\left(-\frac{\sqrt{13}}{2}-\frac{1}{2}\right)^2}{\left(-1+\left(-\frac{\sqrt{13}}{2}-\frac{1}{2}\right)^2\right)\left(-\frac{\sqrt{13}}{2}-\frac{1}{2}\right)} \\ 1 & -\frac{-\frac{5}{2}+\frac{\sqrt{13}}{2}}{-1+\left(-\frac{1}{2}+\frac{\sqrt{13}}{2}\right)^2} & -\frac{-\frac{5}{2}-\frac{\sqrt{13}}{2}}{-1+\left(-\frac{\sqrt{13}}{2}-\frac{1}{2}\right)^2} \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & \frac{1}{2}-\frac{\sqrt{13}}{2} & 0 \\ 0 & 0 & \frac{1}{2}+\frac{\sqrt{13}}{2} \end{bmatrix}\right)$$

[15]:
```
L, simplify(T)
```

[15]:
$$\left(\begin{bmatrix} -1 & 0 & 0 \\ 0 & \frac{1}{2}-\frac{\sqrt{13}}{2} & 0 \\ 0 & 0 & \frac{1}{2}+\frac{\sqrt{13}}{2} \end{bmatrix}, \begin{bmatrix} -1 & -\frac{\sqrt{13}}{2}-\frac{1}{2} & -\frac{1}{2}+\frac{\sqrt{13}}{2} \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}\right)$$

[17]:
```
simplify(A@T - T@L)
```

[17]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

[19]:
```
# das geht auch kürzer
A.diagonalize()
```

[19]:
$$\left(\begin{bmatrix} -1 & -\frac{\sqrt{13}}{2}-\frac{1}{2} & -\frac{1}{2}+\frac{\sqrt{13}}{2} \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & \frac{1}{2}-\frac{\sqrt{13}}{2} & 0 \\ 0 & 0 & \frac{1}{2}+\frac{\sqrt{13}}{2} \end{bmatrix}\right)$$

[20]:
```
A = Matrix(3,3,[-4,-2,-3,5,3,3,5,2,4])
A
```

[20]:
$$\begin{bmatrix} -4 & -2 & -3 \\ 5 & 3 & 3 \\ 5 & 2 & 4 \end{bmatrix}$$

[21]:
```
A.eigenvals()
```

[21]:
$$\{1 : 3\}$$

[22]:
```
A.eigenvects()
```

[22]:
$$\left[\left(1,\ 3,\ \left[\begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{3}{5} \\ 0 \\ 1 \end{bmatrix}\right]\right)\right]$$

[23]:
```
A.left_eigenvects() # linke Eigenvektoren
```

[23]:
$$\left[\left(1,\ 3,\ \left[\begin{bmatrix}1 & 1 & 0\end{bmatrix},\ \begin{bmatrix}1 & 0 & 1\end{bmatrix}\right]\right)\right]$$

[24]:
```
A.diagonalize()
```

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

MatrixError                                Traceback (most recent call last)

<ipython-input-24-842d3ad24c22> in <module>
----> 1 A.diagonalize()

~/miniconda3/envs/CompAna/lib/python3.7/site-packages/sympy/matrices/
↪matrices.py in diagonalize(self, reals_only, sort, normalize)
   1146
   1147         if not self.is_diagonalizable(reals_only=reals_only):
-> 1148             raise MatrixError("Matrix is not diagonalizable")
   1149
   1150         eigenvecs = self.eigenvects(simplify=True)

MatrixError: Matrix is not diagonalizable
```

[25]:
```
elvs = A.eigenvects()
n = 3
L = zeros(n, n)
T  = zeros(n, n)
k = 0
for elv in elvs:
    for ev in elv[2]:
        L[k, k] = elv[0]
        T[:,k] = ev
        k = k+1
T = T[:,:k]
L = L[:k,:k]
T, L
```

[25]:
$$\left(\begin{bmatrix}-\frac{2}{5} & -\frac{3}{5}\\ 1 & 0\\ 0 & 1\end{bmatrix},\ \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)$$

[26]:
```
A@T - T@L
```

[26]:
$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

[27]:
```
T, J = A.jordan_form()
T, J
```

[27]:
$$\left(\begin{bmatrix} -5 & 1 & -\frac{2}{5} \\ 5 & 0 & 1 \\ 5 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

[28]:
```
T*J*T.inv() == A
```

[28]:
```
True
```

[29]:
```
v = T[:,2]
v, A*v
```

[29]:
$$\left(\begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} -\frac{2}{5} \\ 1 \\ 0 \end{bmatrix}\right)$$

1.1.1 Berechung des Rangs

[30]:
```
M = Matrix(3, 3, range(1, 10))
M
```

[30]:
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

[31]:
```
M.rank()
```

[31]:
$$2$$

[33]:
```
M = Matrix(3, 2, [2*x+2, 2*y-2, 2*x+2, -2*y+2, y-1, x+1])
M
```

[33]:
$$\begin{bmatrix} 2x+2 & 2y-2 \\ 2x+2 & 2-2y \\ y-1 & x+1 \end{bmatrix}$$

[34]:
```
M.rank() # ?
```

[34]:
$$2$$

[35]:
```
M.row_op(1, lambda r, j: r-M[0, j])
M1 = M.copy()
```

```
M1
```

[35]:
$$\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & 4-4y \\ y-1 & x+1 \end{bmatrix}$$

[36]:
```
M1.row_op(0, lambda r, j: r*(y-1))    # falls y != 1
M2 = M1.copy()
M2
```

[36]:
$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & 4-4y \\ y-1 & x+1 \end{bmatrix}$$

[37]:
```
M2.row_op(2,lambda r,j: r*(2*x+2))  # falls x != -1
M3 = M2.copy()
M3
```

[37]:
$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & 4-4y \\ (2x+2)(y-1) & (x+1)(2x+2) \end{bmatrix}$$

[38]:
```
M3.row_op(2,lambda r,j: r-M3[0,j])
M4 = M3.copy()
M4
```

[38]:
$$\begin{bmatrix} (2x+2)(y-1) & (y-1)(2y-2) \\ 0 & 4-4y \\ 0 & (x+1)(2x+2)-(y-1)(2y-2) \end{bmatrix}$$

[39]:
```
M4.expand()
```

[39]:
$$\begin{bmatrix} 2xy-2x+2y-2 & 2y^2-4y+2 \\ 0 & 4-4y \\ 0 & 2x^2+4x-2y^2+4y \end{bmatrix}$$

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

[40]:
```
Mx = M.copy()
Mx = Mx.subs(x,-1)
M, Mx  # Rang 2 falls x = -1 und y!=1
```

[40]:
$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & 4-4y \\ y-1 & x+1 \end{bmatrix}, \begin{bmatrix} 0 & 2y-2 \\ 0 & 4-4y \\ y-1 & 0 \end{bmatrix} \right)$$

[41]:
```
My = M.copy()
My = My.subs(y,1)
M, My   # Rang 2 falls x != -1 und y=1
```

[41]:
$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & 4-4y \\ y-1 & x+1 \end{bmatrix}, \begin{bmatrix} 2x+2 & 0 \\ 0 & 0 \\ 0 & x+1 \end{bmatrix}\right)$$

[42]:
```
Mxy = M.copy()
M, Mxy.subs({x:-1,y:1}) # Rang 0 falls x = -1 und y = 1
```

[42]:
$$\left(\begin{bmatrix} 2x+2 & 2y-2 \\ 0 & 4-4y \\ y-1 & x+1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\right)$$

1.2 Normen

1.2.1 Vektornormen

[43]:
```
v = Matrix(1,3,[1,2,3])
v
```

[43]:
$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

[44]:
```
v.norm() # euklidische Norm
```

[44]:
$$\sqrt{14}$$

[45]:
```
v.norm(1)
```

[45]:
$$6$$

[46]:
```
v.norm(oo)
```

[46]:
$$3$$

1.2.2 Matrixnormen

[48]:
```
A, A.norm() # Frobenius Norm  ||A|| = sqrt(sum (|a[i,j]|**2))
```

[48]:
$$\left(\begin{bmatrix} -4 & -2 & -3 \\ 5 & 3 & 3 \\ 5 & 2 & 4 \end{bmatrix}, 3\sqrt{13}\right)$$

[50]:
```
sqrt(trace(A*A.T))
```

[50]:
$$3\sqrt{13}$$

Definition Mit $||A||_2$ wird die 2-Norm von A bezeichnet

$$||A||_2 = (\text{größter Eigenwert von } A^T A)^{1/2}$$

[51]:
```
A.norm(2)
```

[51]:
$$\sqrt{\sqrt{3363} + 58}$$

[52]:
```
(A*A.T).eigenvals()
```

[52]:
$$\left\{1 : 1,\ \ 58 - \sqrt{3363} : 1,\ \ \sqrt{3363} + 58 : 1\right\}$$

[53]:
```
A
```

[53]:
$$\begin{bmatrix} -4 & -2 & -3 \\ 5 & 3 & 3 \\ 5 & 2 & 4 \end{bmatrix}$$

[54]:
```
A.norm(1)
```

[54]:
$$14$$

[55]:
```
A.norm(oo)
```

[55]:
$$11$$

1.3 Weitere Matrixzerlegungen

1.3.1 LU Zerlegung

vgl. CompLA VL8

Satz: Jede invertierbare $n \times n$ Matrix A besitzt eine Zerlegung der Form

$$PA = LU,$$

mit einer Permutationsmatrix P, einer unteren Dreiecksmatrix L und einer oberen Dreiecksmatrix U.

Eine Zerlegung der Form $A = LR$ existiert, falls die Matrix A ohne Zeilentausch in obere Dreiecksgestallt überführt werden kann, d.h. falls während der Rechnung keine Division durch Null auftritt.

[56]:
```
A = Matrix([[1, -1, 2], [0, 0, -1], [0, 2, -1]])
A
```

[56]:
$$\begin{bmatrix} 1 & -1 & 2 \\ 0 & 0 & -1 \\ 0 & 2 & -1 \end{bmatrix}$$

[58]:
```
L, U, perm = A.LUdecomposition() # Zeile 2 und 3 muessen vertauscht werden
L, U, perm
```

[58]:

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

[59]:
```
P = eye(A.shape[0]).permuteBkwd(perm)
P, L, U
```

[59]:
$$\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & -1 & 2\\ 0 & 2 & -1\\ 0 & 0 & -1\end{bmatrix}\right)$$

[61]:
```
P@A == L@U
```

[61]:
```
True
```

[62]:
```
def element(i, j):
    return 1/(i+j+1)
# Hilbertmatrix
H = Matrix(3, 3, element) # Hilbertmatrix
H
```

[62]:
$$\begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3}\\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4}\\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\end{bmatrix}$$

[63]:
```
L, U, perm = H.LUdecomposition()
L, U, perm
```

[63]:
$$\left(\begin{bmatrix}1 & 0 & 0\\ \frac{1}{2} & 1 & 0\\ \frac{1}{3} & 1 & 1\end{bmatrix}, \begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3}\\ 0 & \frac{1}{12} & \frac{1}{12}\\ 0 & 0 & \frac{1}{180}\end{bmatrix}, []\right)$$

[64]:
```
L,U,perm = H.LUdecomposition()
P = eye(H.shape[0]).permuteBkwd(perm)
P@H - L@U
```

[64]:
$$\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

[65]:
```
# die Inverse von H ist bekannt. Die Eintraege der inversen sind:
def element(i, j, n):
    return (-1)**(i+j)*(i+j+1)*binomial(n+i, n-j-1)*binomial(n+j,␣
 ↪n-i-1)*binomial(i+j, i)**2

Hi = Matrix(3, 3, lambda i, j : element(i, j, 3))
Hi
```

[65]:

$$\begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}$$

[66]:
```
H*Hi
```

[66]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1.3.2 LDL- und Choleskyzerlegung

vgl. CompLA VL8

Die LU-Zerlegung einer hermiteschen, positiv definiten Matrix $A \in \mathbb{C}^{n\times n}$ ist ohne Zeilentausch berechenebar.

Satz: Jede hermitesche, positiv definite Matrix $A \in \mathbb{C}^{n\times n}$ besitzt eine Zerlegung der Form

$$A = LDL^T,$$

wobei L eine untere Dreiecksmatrix mit Diagonaleinträgen $l_{i,i} = 1,\ i = 0,\dots,n-1$ und D eine Diagonalmatrix mit positiven Einträgen ist.

Mit

$$D^{1/2} := \begin{pmatrix} \sqrt{d_0} & & & \\ & \sqrt{d_1} & & \\ & & \ddots & \\ & & & \sqrt{d_{n-1}} \end{pmatrix}$$

erhält man eine Zerlegung der Form

$$A = LDL^T = \left(LD^{1/2}\right)\left(D^{1/2}L^T\right) = \tilde{L}\tilde{L}^T$$

mit einer unteren Dreiecksmatrix $\tilde{L}$. Diese Zerlegung nennt man **Cholesky Zerlegung** .

[67]:
```
HC = H.copy()
HC[0,-1] = (1+I)/3
HC[-1,0] = (1-I)/3
HC[-1,-1] = 1
HC
```

[67]:
$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3}+\frac{i}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3}-\frac{i}{3} & \frac{1}{4} & 1 \end{bmatrix}$$

[68]:
```
HC.is_hermitian
```

[68]:
```
True
```

[69]:
```
[simplify(re(l))>0 for l in HC.eigenvals()]
```

[69]:
[True, True, True]

[70]:
```
HC.LDLdecomposition()
```

[70]:
$$\left(\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} - \frac{i}{3} & 1 + 2i & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{12} & 0 \\ 0 & 0 & \frac{13}{36} \end{bmatrix}\right)$$

[71]:
```
HC.cholesky()
```

[71]:
$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{6} & 0 \\ \frac{1}{3} - \frac{i}{3} & 2\sqrt{3}\left(\frac{1}{12} + \frac{i}{6}\right) & \frac{\sqrt{13}}{6} \end{bmatrix}$$

[72]:
```
H.is_symmetric()
[simplify(re(l))>0 for l in H.eigenvals()]
```

[72]:
[True, True, True]

[73]:
```
[simplify(im(l)) for l in H.eigenvals()]
```

[73]:
[0, 0, 0]

[74]:
```
H.LDLdecomposition()
```

[74]:
$$\left(\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{12} & 0 \\ 0 & 0 & \frac{1}{180} \end{bmatrix}\right)$$

[75]:
```
H.cholesky()
```

[75]:
$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{6} & 0 \\ \frac{1}{3} & \frac{\sqrt{3}}{6} & \frac{\sqrt{5}}{30} \end{bmatrix}$$

1.3.3 QR Zerlegung

vgl. CompLA VL 9

Die QR Zerlegung einer $n \times m$ Matrix $ A $ ist eine Zerlegung

$$A = QR,$$

mit einer Matrix Q mit orthonormalen Spalten und einer oberen Dreiecksmatrix R.

Satz: Jede $n \times m$ Matrix A mit $n \geq m$ und vollem Rang besitzt eine QR Zerlegung.

[76]:
```
Q, R = A.QRdecomposition()
Q, R
```

[76]:
$$\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}, \begin{bmatrix}1 & -1 & 2\\ 0 & 2 & -1\\ 0 & 0 & 1\end{bmatrix}\right)$$

[79]:
```
Q.T@Q, Q@R-A
```

[79]:
$$\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

[80]:
```
# das ist ein Bug in sympy. Sympy macht das nur für reelle Matrizen richtig
AC = Matrix(3, 3, [I, 1 , 1, 1, 4, -1, 1, -1, 4])
Q, R =  AC.QRdecomposition()
simplify(Q), simplify(R)
```

[80]:
$$\left(\begin{bmatrix}\frac{\sqrt{3}i}{3} & \frac{1}{3}-\frac{i}{4} & \frac{\sqrt{1615}(326-207i)}{19380}\\ \frac{\sqrt{3}}{3} & \frac{3}{4}-\frac{i}{12} & -\frac{\sqrt{1615}(21+23i)}{19380}\\ \frac{\sqrt{3}}{3} & -\frac{1}{2}-\frac{i}{12} & \frac{\sqrt{1615}(264-113i)}{19380}\end{bmatrix}, \begin{bmatrix}\sqrt{3} & \frac{\sqrt{3}(3+i)}{3} & \frac{\sqrt{3}(3+i)}{3}\\ 0 & 4 & -\frac{29}{12}-\frac{i}{2}\\ 0 & 0 & \frac{\sqrt{1615}}{12}\end{bmatrix}\right)$$

[81]:
```
simplify(Q.H@Q) , Q@R - AC
```

[81]:
$$\left(\begin{bmatrix}1 & -\frac{\sqrt{3}i}{6} & \frac{\sqrt{4845}(6-77i)}{9690}\\ \frac{\sqrt{3}i}{6} & 1 & \frac{2\sqrt{1615}(1+3i)}{1615}\\ \frac{\sqrt{4845}(6+77i)}{9690} & \frac{2\sqrt{1615}(1-3i)}{1615} & 1\end{bmatrix}, \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

[82]:
```
B = Matrix(4, 3, [3, 1 , 1, 1, 4, -1, 1, -1, 4, 1, 1, 1])
B
```

[82]:
$$\begin{bmatrix}3 & 1 & 1\\ 1 & 4 & -1\\ 1 & -1 & 4\\ 1 & 1 & 1\end{bmatrix}$$

[83]:
```
Q, R = B.QRdecomposition()
Q, R
```

[83]:
$$\left(\begin{bmatrix}\frac{\sqrt{3}}{2} & -\frac{3\sqrt{537}}{358} & -\frac{45\sqrt{10382}}{10382}\\ \frac{\sqrt{3}}{6} & \frac{41\sqrt{537}}{1074} & \frac{13\sqrt{10382}}{5191}\\ \frac{\sqrt{3}}{6} & -\frac{19\sqrt{537}}{1074} & \frac{42\sqrt{10382}}{5191}\\ \frac{\sqrt{3}}{6} & \frac{5\sqrt{537}}{1074} & \frac{25\sqrt{10382}}{10382}\end{bmatrix}, \begin{bmatrix}2\sqrt{3} & \frac{7\sqrt{3}}{6} & \frac{7\sqrt{3}}{6}\\ 0 & \frac{\sqrt{537}}{6} & -\frac{121\sqrt{537}}{1074}\\ 0 & 0 & \frac{5\sqrt{10382}}{179}\end{bmatrix}\right)$$

[84]:
```
Q.T@Q, Q@R-B
```

[84]:

$$\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\right)$$

[85]:
```
BT = B.T.copy()
BT
```

[85]:
$$\begin{bmatrix}3 & 1 & 1 & 1\\ 1 & 4 & -1 & 1\\ 1 & -1 & 4 & 1\end{bmatrix}$$

[86]:
```
Q, R = BT.QRdecomposition()
Q, R
```

[86]:
$$\left(\begin{bmatrix}\frac{3\sqrt{11}}{11} & -\frac{7\sqrt{22}}{198} & -\frac{5\sqrt{2}}{18}\\ \frac{\sqrt{11}}{11} & \frac{19\sqrt{22}}{99} & \frac{2\sqrt{2}}{9}\\ \frac{\sqrt{11}}{11} & -\frac{17\sqrt{22}}{198} & \frac{11\sqrt{2}}{18}\end{bmatrix}, \begin{bmatrix}\sqrt{11} & \frac{6\sqrt{11}}{11} & \frac{6\sqrt{11}}{11} & \frac{5\sqrt{11}}{11}\\ 0 & \frac{9\sqrt{22}}{11} & -\frac{113\sqrt{22}}{198} & \frac{7\sqrt{22}}{99}\\ 0 & 0 & \frac{35\sqrt{2}}{18} & \frac{5\sqrt{2}}{9}\end{bmatrix}\right)$$

[87]:
```
Q.T@Q, Q@R-BT
```

[87]:
$$\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\right)$$

1.4 sympy Matrizen in numpy Matrizen umwandeln

[88]:
```
Hnp1 = matrix2numpy(H)  # das ist ein numpy array mit sympy Eintraegen␣
↪(Rationals)
Hnp1
```

[88]:
```
array([[1, 1/2, 1/3],
       [1/2, 1/3, 1/4],
       [1/3, 1/4, 1/5]], dtype=object)
```

[89]:
```
Hnp1[1,1], type(Hnp1[1,1])
```

[89]:
```
(1/3, sympy.core.numbers.Rational)
```

[90]:
```
Hnp1.astype('float')
```

[90]:
```
array([[1.        , 0.5       , 0.33333333],
       [0.5       , 0.33333333, 0.25      ],
       [0.33333333, 0.25      , 0.2       ]])
```

[91]:
```
Hnp1[1,1]
```

[91]:

$\frac{1}{3}$

[ ]:
```
Hnp = Hnp1.astype('float')
Hnp
```

[92]:
```
Hnp2 = matrix2numpy(H, dtype='float')
Hnp2
```

[92]:
```
array([[1.        , 0.5       , 0.33333333],
       [0.5       , 0.33333333, 0.25      ],
       [0.33333333, 0.25      , 0.2       ]])
```

[93]:
```
type(Hnp2[1,1])
```

[93]:
```
numpy.float64
```

[94]:
```
Hnp3 = np.array(H) # das gleiche wie Hnp1
Hnp3
```

[94]:
```
array([[1, 1/2, 1/3],
       [1/2, 1/3, 1/4],
       [1/3, 1/4, 1/5]], dtype=object)
```

[95]:
```
type(Hnp3[1,1])
```

[95]:
```
sympy.core.numbers.Rational
```

[96]:
```
Hnp4 = np.array(H, dtype='float') # das gleiche wie Hnp2
Hnp4
```

[96]:
```
array([[1.        , 0.5       , 0.33333333],
       [0.5       , 0.33333333, 0.25      ],
       [0.33333333, 0.25      , 0.2       ]])
```

[97]:
```
type(Hnp4[1,1])
```

[97]:
```
numpy.float64
```

1.5 Matrixfunktionen

[98]:
```
sin(H)
```

[98]:
$$\sin\left(\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}\right)$$

[99]:
```
H.applyfunc(sin) # sinus angewandt auf jeden Eintrag der Matrix
```

[99]:
$$\begin{bmatrix} \sin(1) & \sin\left(\frac{1}{2}\right) & \sin\left(\frac{1}{3}\right) \\ \sin\left(\frac{1}{2}\right) & \sin\left(\frac{1}{3}\right) & \sin\left(\frac{1}{4}\right) \\ \sin\left(\frac{1}{3}\right) & \sin\left(\frac{1}{4}\right) & \sin\left(\frac{1}{5}\right) \end{bmatrix}$$

1.5.1 Matrixwurzel

[101]:
```
B = Matrix(3, 3, [4, 1 , 1, 1, 4, -1, 1, -1, 4])
```

[102]:
```
sqrtB = B**(S(1)/2)
sqrtB
```

[102]:
$$\begin{bmatrix} \frac{\sqrt{2}}{3} + \frac{2\sqrt{5}}{3} & -\frac{\sqrt{2}}{3} + \frac{\sqrt{5}}{3} & -\frac{\sqrt{2}}{3} + \frac{\sqrt{5}}{3} \\ -\frac{\sqrt{2}}{3} + \frac{\sqrt{5}}{3} & \frac{\sqrt{2}}{3} + \frac{2\sqrt{5}}{3} & -\frac{\sqrt{5}}{3} + \frac{\sqrt{2}}{3} \\ -\frac{\sqrt{2}}{3} + \frac{\sqrt{5}}{3} & -\frac{\sqrt{5}}{3} + \frac{\sqrt{2}}{3} & \frac{\sqrt{2}}{3} + \frac{2\sqrt{5}}{3} \end{bmatrix}$$

[104]:
```
erg = sqrtB @ sqrtB  - B
simplify(erg)
```

[104]:
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

[ ]:
```
simplify(erg)
```

1.5.2 Matrixexponentialfunktion

Die Exponentialfunktion $z \mapsto e^z$ ist als Reihe definiert

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!}.$$

Setzt man hier formal für z eine Matrix ein, so erhält man die Matrixexponentilafunktion (vgl. Ana II).

[105]:
```
exp(B)
```

[105]:
$$\begin{bmatrix} \frac{e^2}{3} + \frac{2e^5}{3} & -\frac{e^2}{3} + \frac{e^5}{3} & -\frac{e^2}{3} + \frac{e^5}{3} \\ -\frac{e^2}{3} + \frac{e^5}{3} & \frac{e^2}{3} + \frac{2e^5}{3} & -\frac{e^5}{3} + \frac{e^2}{3} \\ -\frac{e^2}{3} + \frac{e^5}{3} & -\frac{e^5}{3} + \frac{e^2}{3} & \frac{e^2}{3} + \frac{2e^5}{3} \end{bmatrix}$$

[106]:
```
B.applyfunc(exp)
```

[106]:
$$\begin{bmatrix} e^4 & e & e \\ e & e^4 & e^{-1} \\ e & e^{-1} & e^4 \end{bmatrix}$$

1.5.3 Matrix mit Zufallszahlen

[108]:
```
M = randMatrix(2, 3) # zufällige ganze Zahle zwischen 0 .. 99
M
```

[108]:
$$\begin{bmatrix} 76 & 94 & 10 \\ 81 & 39 & 9 \end{bmatrix}$$

1.6 Kopieren von Matrizen

[110]:
```
BB = B
```

[111]:
```
display(B), display(BB);
```

$$\begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 4 \end{bmatrix}$$

[113]:
```
BB[0, 0]=9
```

[114]:
```
display(B), display(BB);
```

$$\begin{bmatrix} 9 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 4 \end{bmatrix}$$

[115]:
```
BBB = B.copy()
```

[116]:
```
BBB[2, 2] = 99
```

[117]:
```
display(B), display(BBB);
```

$$\begin{bmatrix} 9 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 1 & 1 \\ 1 & 4 & -1 \\ 1 & -1 & 99 \end{bmatrix}$$

1.7 Kreuzprodukt

[118]:
```
w = Matrix(1, 3, [1, -2, 1])
v, w
```

[118]:
$$\left(\begin{bmatrix}1 & 2 & 3\end{bmatrix}, \begin{bmatrix}1 & -2 & 1\end{bmatrix}\right)$$

[119]:
```
z = v.cross(w)
z
```

[119]:
$$\begin{bmatrix}8 & 2 & -4\end{bmatrix}$$

[120]:
```
w.cross(v)
```

[120]:
$$\begin{bmatrix}-8 & -2 & 4\end{bmatrix}$$