Abgabe - Übungsblatt [3]

Angewandte Mathematik: Numerik

[Felix Lehmann] [Markus Menke]

26. November 2020

Aufgabe 1

A ist gleich der Vandermonde-Matrix

$$A = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}$$

$$v = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 4 \end{pmatrix}$$

So ergibt sich das folgende lineare Ausgleichsproblem:

$$\begin{pmatrix} c \\ b \\ a \end{pmatrix} = \arg\min_y ||v - Ay||_2$$

Nach Skript Satz 2.2 suchen wir nun nach einer Lösung der Gausschen Normalengleichung:

$A * Ax = A * v$

Setzen wir unsere Werte ein, erhalten wir folgende Gleichung:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \begin{pmatrix} c \\ b \\ a \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 4 \end{pmatrix}$$

Zuerst lösen wir die Matrixprodukte auf

$$\leftrightarrow \begin{pmatrix} 4 & 2 & 6 \\ 2 & 6 & 8 \\ 6 & 8 & 18 \end{pmatrix} \begin{pmatrix} c \\ b \\ a \end{pmatrix} = \begin{pmatrix} 7 \\ 9 \\ 19 \end{pmatrix}$$

Wir lösen das LGS mit dem Gauss-Algorithmus:

$$\left(\begin{array}{ccc|c} 4 & 2 & 6 & 7 \\ 2 & 6 & 8 & 9 \\ 6 & 8 & 18 & 19 \end{array}\right) \leftrightarrow \left(\begin{array}{ccc|c} 4 & 2 & 6 & 7 \\ 4 & 12 & 16 & 18 \\ 12 & 16 & 36 & 38 \end{array}\right) \leftrightarrow \left(\begin{array}{ccc|c} 4 & 2 & 6 & 7 \\ 0 & 10 & 10 & 11 \\ 0 & 10 & 18 & 17 \end{array}\right)$$

$$\leftrightarrow \left(\begin{array}{ccc|c} 4 & 2 & 6 & 7 \\ 0 & 10 & 10 & 11 \\ 0 & 0 & 8 & 16 \end{array}\right) \rightarrow a = 3/4, b = 7/20, c = 9/20$$

Aufgabe 2

Nope...

Aufgabe 3

a)

```
def ComputeTPSWeights(X, Y, Z):
    m = X.shape[0]
    A = np.zeros(shape=(m, m))
    for i in range(m):
        vi = np.array([X[i], Y[i]])
        for j in range(m):
            vj = np.array([X[j], Y[j]])
            r = np.linalg.norm(vi-vj)
            A[j, i] = (r**2) * np.log(max(1.0e-8,r))
    w = np.linalg.solve(A, Z)
    return w
```

b)

```
def EvaluateTPSSpline(XNew, YNew, X, Y, Weights):
    if (len(XNew.shape) > 1):
        XNew = XNew[0, :]
    if (len(YNew.shape) > 1):
        YNew = YNew[:, 0]
    m = XNew.shape[0]
    res = np.zeros(shape=(m, m))
    for ix in range(m):
        for iy in range(m):
            tmp = 0
            for j in range(len(Weights)):
                r = np.linalg.norm(np.array([XNew[ix]-X[j
                    ],YNew[iy]-Y[j]]))
                tmp += Weights[j] * ((r**2) * np.log(max
                    (1.0e-8,r)))
            res[iy, ix] = tmp
    return res
```