| Aufgabe | A1 | A2 | A3 | $\sum$ |
|---|---|---|---|---|
| Punkte | | | | |

**Aufgabe 1.** a) Beh.:

$$\|f\|_\infty = \max_{x\in[0,1]} |f(x)|, \quad f \in C^0([0,1],\mathbb{R}).$$

ist eine Norm auf $C^0([0,1],\mathbb{R})$.

*Beweis.* Seien $f, g \in C^0([0,1],\mathbb{R})$.

(N1) Es ist $\|f\|_\infty = \max_{x\in[0,1]} |f(x)| \geq 0$. Außerdem ist

$$\|f\|_\infty = 0 \implies \max_{x\in[0,1]} \underbrace{|f(x)|}_{\geq 0} = 0 \implies f(x) = 0 \quad \forall x \in [0,1] \implies f = 0 \in C^0([0,1],\mathbb{R}).$$

(N2) Sei $\alpha \in \mathbb{R}$. Dann folgt

$$\|\alpha f\|_\infty = \max_{x\in[0,1]} |\alpha f(x)| = \max_{x\in[0,1]} \underbrace{|\alpha|}_{\geq 0} \underbrace{|f(x)|}_{\geq 0} = |\alpha| \max_{x\in[0,1]} |f(x)| = |\alpha| \|f\|_\infty.$$

(N3) Es ist

$$\|f+g\|_\infty = \max_{x\in[0,1]} |f(x)+g(x)| \leq \max_{x\in[0,1]} \left( \underbrace{|f(x)|}_{\geq 0} + \underbrace{|g(x)|}_{\geq 0} \right) = \max_{x\in[0,1]} |f(x)| + \max_{x\in[0,1]} |g(x)| = \|f\|_\infty + \|g\|_\infty.$$

□

b) Beh.:

$$\|f\|_1 = \max_{x\in[0,1]} |f(x)|, \quad f \in C^0([0,1],\mathbb{R}).$$

ist eine Norm auf $C^0([0,1],\mathbb{R})$.

*Beweis.* Seien $f, g \in C^0([0,1],\mathbb{R})$.

(N1) Es ist $\|f\|_1 = \int_0^1 \underbrace{|f(x)|}_{\geq 0} \,\mathrm{d}x \geq 0$. Außerdem ist wegen der Monotonie des R.-Integrals:

$$\|f\|_1 = 0 \implies \int_0^1 \underbrace{|f(x)|}_{\geq 0} \,\mathrm{d}x = 0 \implies f(x) = 0 \quad \forall x \in [0,1] \implies f = 0 \in C^0([0,1],\mathbb{R}).$$

(N2) Sei $\alpha \in \mathbb{R}$. Dann folgt mit der Linearität des R.-Integrals

$$\|\alpha f\|_1 = \int_0^1 |\alpha f(x)| \,\mathrm{d}x = \int_0^1 |\alpha||f(x)| \,\mathrm{d}x = |\alpha| \int_0^1 |f(x)| \,\mathrm{d}x = |\alpha| \|f\|_1.$$

(N3) Es ist

$$\|f+g\|_1 = \int_0^1 |f(x)+g(x)| \,\mathrm{d}x \leq \int_0^1 |f(x)| \,\mathrm{d}x + \int_0^1 |g(x)| \,\mathrm{d}x = \|f\|_1 + \|g\|_1.$$

□

c) Für die gegebene Funktionenfolge gilt für $k \in \mathbb{N}$:

$$\|u_k\|_\infty = \max_{x\in[0,1]} |u_k(x)| = \max_{x\in[x_{k+1},x_k]} \sin\left(\frac{x_k - x}{x_k - x_{k+1}}\pi\right) = 1.$$

Für die 1-Norm folgt

$$\begin{aligned}
\|u_k\|_1 &= \int_0^1 u_k(x)\,\mathrm{d}x \\
&= \int_{x_{k+1}}^{x_k} \sin\left(\frac{x_k - x}{x_k - x_{k+1}}\pi\right)\,\mathrm{d}x \\
&= \left[\frac{x_k - x_{k+1}}{\pi}\cos\left(\frac{x_k - x}{x_k - x_{k+1}}\pi\right)\right]\Big|_{x_{k+1}}^{x_k} \\
&= 2\frac{x_k - x_{k+1}}{\pi}
\end{aligned}$$

Mit $x_k = \frac{1}{k}$ folgt

$$\begin{aligned}
\|u_k\|_1 &= \frac{2}{\pi}\left(\frac{1}{k} - \frac{1}{k+1}\right) \\
&= \frac{2}{\pi}\left(\frac{1}{k^2+k}\right) \\
&\xrightarrow{k\to\infty} 0.
\end{aligned}$$

Beh.: $\|\cdot\|_1$ und $\|\cdot\|_\infty$ sind nicht äquivalent.

*Beweis.* Ang.: $\|\cdot\|_1$ und $\|\cdot\|_\infty$ seien äquivalent, dann existiert ein $m \in \mathbb{R}$, s.d. $\forall f \in C^0([0,1],\mathbb{R})$ gilt

$$m\|f\|_\infty \le \|f\|_1 \quad \text{also insbes.} \quad m\|u_k\|_\infty \le \|u_k\|_1 \quad \forall k \in \mathbb{N}.$$

Wegen $\|u_k\|_\infty = 1\ \forall k \in \mathbb{N}$, folgt also

$$m \le \|u_k\|_1 \quad \forall k \in \mathbb{N} \quad \lightning \text{ zu } \|u_k\| \xrightarrow{k\to\infty} 0.$$

Also sind $\|\cdot\|_1$ und $\|\cdot\|_\infty$ nicht äquivalent. □

**Aufgabe 2.** (i) Beh.: $\|\cdot\|_F$ ist eine Norm auf $\mathbb{K}^{n\times n}$.

*Beweis.* Sei $A \in \mathbb{K}^{n\times n}$ beliebig.

(N1) Es ist $\|A\|_F = \left(\sum_{i,j=1}^n \underbrace{|a_{ij}|^2}_{\ge 0}\right)^{\frac{1}{2}} \ge 0$. Außerdem gilt

$$\|A\|_F = 0 \implies \left(\sum_{i,j=1}^n \underbrace{|a_{ij}|^2}_{\ge 0}\right)^{\frac{1}{2}} \implies a_{ij} = 0 \quad \forall i,j = 1,\dots n \implies A = 0.$$

(N2) Sei $\alpha \in \mathbb{K}$ beliebig. Dann ist

$$\|\alpha A\|_F = \left(\sum_{i,j=1}^n |\alpha a_{ij}|^2\right)^{\frac{1}{2}} = \left(|\alpha|^2 \sum_{i,j=1}^n |a_{ij}|^2\right)^{\frac{1}{2}} = |\alpha|\|A\|_F.$$

(N3) Sei $B \in \mathbb{K}^{n\times n}$. Durch Identifikation von Matrizen aus $\mathbb{K}^{n\times n}$ mit der Frobeniusnorm und Vektoren aus $\mathbb{K}^{n\cdot n}$ mit der Euklidschen Norm, gilt die Cauchy-Schwarz-Ungleichung. Dann

folgt

$$\begin{aligned}
\|A+B\|_F^2 &= \sum_{i,j=1}^{n} |a_{ij}+b_{ij}|^2 \\
&= \sum_{i,j=1}^{n} |a_{ij}^2 + 2a_{ij}b_{ij} + b_{ij}^2| \\
&\leq \sum_{i,j=1}^{n} |a_{ij}|^2 + 2\sum_{i,j=1}^{n} |a_{ij}b_{ij}| + \sum_{i,j=1}^{n} |b_{ij}|^2 \\
&\overset{\text{C.S.U.}}{\leq} \|A\|_F^2 + 2\|A\|_F\|B\|_F + \|B\|_F^2 \\
&= \left(\|A\|_F + \|B\|_F\right)^2.
\end{aligned}$$

Damit folgt die Behauptung.

□

(ii) Beh.: $\forall A \in \mathbb{K}^{n\times n}$, $x \in \mathbb{K}^n$ gilt

$$\|Ax\|_2 \leq \|A\|_F\|x\|_2.$$

*Beweis.* Seien $A \in \mathbb{K}^{n\times n}$ und $x \in \mathbb{K}^n$ beliebig. $A_i$ bezeichne die $i$-te Zeile der Matrix $A$. Dann gilt

$$\begin{aligned}
\|Ax\|_2^2 &= \sum_{i=1}^{n} \left( \sum_{j=1}^{n} |a_{ij}x_j| \right)^2 \\
&= \sum_{i=1}^{n} (A_i, x)_2^2 \\
&\overset{\text{C.S.U.}}{\leq} \sum_{i=1}^{n} \|A_i\|_2^2\|x\|_2^2 \\
&= \|x\|_2^2 \sum_{i=1}^{n} \|A_i\|_2^2 \\
&= \|x\|_2^2 \sum_{i=1}^{n}\sum_{j=1}^{n} |a_i|^2 \\
&= \|x\|_2^2\|A\|_F^2.
\end{aligned}$$

Damit folgt die Behauptung. □

(iii) Beh.: $\forall A, B \in \mathbb{K}^{n\times n}$ gilt

$$\|AB\|_F \leq \|A\|_F\|B\|_F.$$

*Beweis.* Seien $A \in \mathbb{K}^{n\times n}$ und $x \in \mathbb{K}^n$ beliebig. $A_i$ bezeichne die $i$-te Zeile der Matrix $A$, $B_j$ die $j$-te Spalte (!) der Matrix $B$. Dann gilt

$$\begin{aligned}
\|AB\|_F^2 &= \sum_{i,j=1}^{n} \left| \sum_{k=1}^{n} a_{ik}b_{kj} \right|^2 \\
&= \sum_{i,j=1}^{n} (A_i, B_j)_2^2 \\
&\overset{\text{C.S.U.}}{\leq} \sum_{i,j=1}^{n} \|A_i\|_2^2\|B_j\|_2^2 \\
&= \sum_{i=1}^{n} \|A_i\|_2^2 \sum_{j=1}^{n} \|B_j\|_2^2 \\
&= \|A\|_F\|B\|_F.
\end{aligned}$$

□

**Aufgabe 3.** a) Auszug aus *rohrleitungsnetzwerk.cc*

```
// Funktion zum Aufstellen der Matrix
template<class NumberType>
void flussMatrix( hdnum::DenseMatrix<NumberType> &A ) {
    int M( A.rowsize() );
    int N( A.colsize() );
    if(M!=N)
        HDNUM_ERROR("Matrix muss quadratisch sein!");

    // Numerierung wie auf Zettel 3 nur mit 0 beginnend
    // also v_0, ... ,v_(N^2-1)
    // der Referenzknoten v_r hat Druck 0

    // berechnung der kantenlaenge
    int n = floor(sqrt(N+1));

    for(int i = 0; i < N; i++) {
        int edges = 0;
        if ((i+1)%n != 0) { // nicht linker rand
            edges++;
            if (i-1 >= 0) A(i, i-1) = -1; // falls nicht der referenzknoten
        }
        if ((i+2)%n != 0) { // nicht rechter rand
            edges++;
            A(i, i+1) = -1;
        }
        if (i+n < N) { // nicht unterer rand
            edges++;
            A(i, i+n) = -1;
        }
        if (i-n >= -1) { // nicht oberer rand
            edges++;
            if (i-n >= 0) A(i, i-n) = -1; // falls nicht der referenzknoten
        }
        A(i, i) = edges;
    }
}
```

Funktion zum Aufstellen der Matrix

b) In `DenseMatrix` sind schon die Zeilen- und Spaltensummennorm definiert. Auszug aus *rohrleitungsnetzwerk.cc*

```
// Funktion zur Berechnung der Frobenius-Norm einer Matrix
template<class NumberType>
NumberType frobeniusNorm(const hdnum::DenseMatrix<NumberType> &A) {
    // Error checking
    int M(A.rowsize());
    int N(A.colsize());
    if(M!=N)
        HDNUM_ERROR("Matrix muss quadratisch sein!");

    NumberType result=0.0;

    // iteriere ueber alle zeilen und spalten, quadriere die elemente und
    summiere
    for (int i=0; i < N; i++) {
        for (int j=0; j < N; j++) {
            result += pow(A(i,j), 2);
        }
    }

    // ziehe wurzel aus summe
    return sqrt(result);
}
```

Frobeniusnorm

c) Auszug aus *rohrleitungsnetzwerk.cc*

```
// Funktion zur Berechnung des betragsgroessten Eigenwertes mit Potenzmethode
template<class NumberType>
NumberType maxEigenwert(const hdnum::DenseMatrix<NumberType> &A) {
    // Error checking
    int M(A.rowsize());
```

```
6       int N(A.colsize());
7       if(M!=N)
8           HDNUM_ERROR("Matrix muss quadratisch sein!");
9
10      // start vektor
11      hdnum::Vector<NumberType> r(N);
12      r[0] = 0;
13      // work copy
14      hdnum::Vector<NumberType> r_tmp(N);
15      // finde vektor mit Ar != 0
16      for (int i = 0;;i++) {
17          A.mv(r_tmp, r); // r_tmp = Ar
18          if (r_tmp.two_norm() > 0) {
19              // Ar != 0
20              break;
21          } else if (i >= 100) {
22              // breche ab, wenn kein passender startvektor gefunden werden kann
23              HDNUM_ERROR("Kann keinen Vektor mit Ax != 0 finden. Ist A = 0?");
24          }
25          // Ar = 0, modifiziere start vektor
26          r[i % N] += 1;
27      }
28      // fuehre iterationsschritt maximal 10000 mal aus
29      for (int k=0; k<10000; k++) {
30          A.mv(r_tmp, r); // r_tmp = Ar
31          r_tmp /= r_tmp.two_norm(); // normiere r_tmp
32          if ((r-r_tmp).two_norm() == 0)
33              // breche ab ab, wenn maximale genauigkeit erreicht wurde
34              break;
35          r = r_tmp;
36      }
37      // berechne eigenwert mit rayleigh quotient
38      A.mv(r_tmp, r); // r_tmp = Ar
39      return (r * r_tmp)/r.two_norm_2();
40  }
```

Eigenwertberechnung