Technische Universität Berlin
Fakultät II; Institut für Mathematik
M. Karow

Wintersemester 23

Numerische Mathematik I für Ingenieurwissenschaften

2. Übungsblatt zur Vorlesung

Aufgabe 1

Ü) Zeige, dass die Hilbertmatrix der Größe n

$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & & & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & & & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & & \cdots & \frac{1}{2n-1} \end{bmatrix} = \left[\frac{1}{j+k-1}\right]_{j,k\leq n}$$

positiv definit ist.

Bemerkung: Die Hilbertmatrix der Größe n kann mit dem MATLAB Befehl `A=hilb(n)` erzeugt werden. Hilbertmatrizen sind schlecht konditioniert, siehe Programmieraufgabe.

Aufgabe 2

Ü) Zu einer Matrix $A = [a_{ik}] \in \mathbb{R}^{n\times n}$ definiert man die Zeilensummen und die Spaltensummen folgendermaßen:

$$Z_i(A) = \sum_{k=1}^{n} |a_{ik}|, \qquad S_k(A) = \sum_{i=1}^{n} |a_{ik}|.$$

In der Vorlesung wurde erwähnt und an einem Beispiel demonstriert, dass $\|A\|_\infty = \max_{i=1}^n Z_i(A)$. Die von der ∞-Norm induzierte Matrixnorm ist also die Zeilensummennorm. Demonstriere am Beispiel von 2×2-Matrizen, dass

$$\|A\|_1 = \max_{k=1}^n S_k(A).$$

Die von der 1-Norm induzierte Matrixnorm ist also die Spaltensummennorm.

Aufgabe 3 — 4 Punkte

Berechne die Normen $\|A\|_p$, $\|A^{-1}\|_p$ und die Konditionszahlen $\mathrm{cond}_p(A)$ für $p \in \{1, 2, \infty\}$ und folgende Matrizen.

Üa) $A = \begin{bmatrix} 4 & -13 \\ 8 & -1 \end{bmatrix}$, Üb) $A = \begin{bmatrix} 0 & -2 & 0 \\ 5 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}$, Üc) $A = I$ (Einheitsmatrix).

$Ha)$ $A = \begin{bmatrix} 22 & 3 \\ -18 & 13 \end{bmatrix}$, $Hb)$ $A = \begin{bmatrix} 0 & -4 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 0 & 4 & 0 \end{bmatrix}$.

Tipps: Die Formel für die Inverse einer 2×2 Matrix ist:

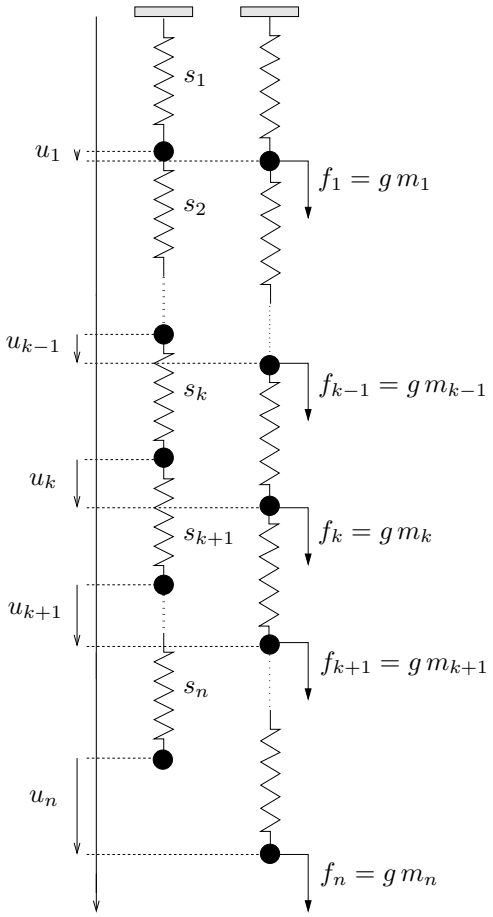
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \Rightarrow \qquad A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

Die Inverse einer blockdiagonalen Matrix bekommt man, indem man die Diagonalblöcke invertiert:

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \qquad \Rightarrow \qquad A^{-1} = \begin{bmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{bmatrix}.$$

Aufgabe 4

Ü) Ein wenig Mechanik. Gegeben sei die folgende hängende Kette aus (masselosen) Federn mit Steifigkeiten $s_k > 0$, an denen die Massen $m_k > 0$ befestigt sind. Links sieht man die Kette im entspannten Zustand, d.h. ohne Wirkung der Gravitation. Wenn aber die Gravitation wirkt, werden die Massen m_k mit der Kraft $f_k = g\, m_k$ nach unten gezogen (g =Fallbeschleunigung), die Federn dehnen sich, und die Massen werden um die Strecken u_k nach unten verschoben. Diese Situation sieht man rechts.



Zeige: Zwischen den Kräften f_k und den Verschiebungen u_k besteht die Beziehung

$$\underbrace{\begin{bmatrix} s_1+s_2 & -s_2 & 0 & & & & \\ -s_2 & s_2+s_3 & -s_3 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -s_k & s_k+s_{k+1} & -s_{k+1} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -s_{n-1} & s_{n-1}+s_n & -s_n \\ & & & & 0 & -s_n & s_n \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{k-1} \\ u_k \\ u_{k+1} \\ \vdots \\ u_{n-2} \\ u_{n-1} \\ u_n \end{bmatrix}}_{u} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ \\ f_k \\ \\ \vdots \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}.$$

Bemerkung: Die Steifigkeitsmatrix S ist positiv definit. Physikalische Begründung: $\frac{1}{2}u^\top Su > 0$ ist die elastische Energie, die in den verformten Federn gespeichert ist.

Programmieraufgabe (darf auch mit Python gelöst werden)

(a) Sei `A=hilb(n)` die Hilbertmatrix aus Aufgabe 1. Sei $b \in \mathbb{R}^n$ die Summe der ersten und der letzten Spalte von A, in MATLAB-Notation:

`b=A(:,1)+A(:,n)`.

Die Lösung des Gleichungssystems $Ax = b$ ist

$$x = [\,1\ \ 0\ \ \ldots\ \ 0\ \ 1\,]^\top \in \mathbb{R}^n.$$

Wenn n groß ist, dann hat A eine große Konditionszahl. Aus diesem Grund kann das Gleichungssystem $Ax = b$ numerisch nicht mehr exakt gelöst werden.

Erstelle mit MATLAB eine Tabelle (d.h. eine Matrix) T, die folgende Daten enthält.

- In der ersten Spalte von T stehen die Zahlen $n = 2, 3, \ldots, 13$.
- In der zweiten Spalte von T stehen die Konditionszahlen der Hilbertmatrizen der Dimensionen $n = 2, 3, \ldots, 13$ bezüglich der ∞-Norm. Zur Berechnung der Konditionszahlen verwende die MATLAB-Funktion `cond`. Erklärungen dazu, wie man diese Funktion benutzt, bekommt man indem man im Command Window `doc cond` eingibt. Aus Formatierungsgründen sollen die Konditionszahlen mit 10^{-16} multipliziert werden, bevor sie in T eingetragen werden.
- In der dritten Spalte von T stehen die relativen Fehler $\frac{\|x-x_{bs}\|_\infty}{\|x\|_\infty}$, wobei x_{bs} die numerische Lösung von $Ax = b,\ A \in \mathbb{R}^{n\times n}$, ist, die man mit dem Backslash bekommt: $x_{bs} = A\backslash b$. Zur Berechnung der ∞-Norm kann die MATLAB-Funktion `norm` benutzt werden (siehe `doc norm`).
- In der vierten Spalte von T steht der relative Fehler $\frac{\|x-x_{lr}\|_\infty}{\|x\|_\infty}$, wobei x_{lr} die numerische Lösung von $Ax = b$ ist, die man bekommt, wenn man zunächst mit $[L, R, P] = lu(A)$ eine LR-Zerlegung durchführt und anschließend mit Vorwärts- und Rückwärtseinsetzen löst (hierbei darf die Backslash-Anweisung benutzt werden, Multiplikation von b mit der Permutationsmatrix P nicht vergessen).
- In der fünften Spalte von T steht der relative Fehler $\frac{\|x-x_{chol}\|_\infty}{\|x\|_\infty}$, wobei x_{chol} die numerische Lösung von $Ax = b$ ist, die man bekommt, wenn man zunächst mit $R = chol(A)$ eine Cholesky-Zerlegung durchführt und anschließend mit Vorwärts- und Rückwärtseinsetzen löst (hierbei darf die Backslash-Anweisung benutzt werden. Merke R ist *obere* Dreiecksmatrix).
- In der sechsten Spalte steht der Probewert $\|A\,x_{chol} - b\|_\infty$.

Welcher Algorithmus schneidet am schlechtesten ab? Was geschieht, wenn man die Tabelle bis $n = 14$ oder $n = 15$ erstellen will? Schreiben Sie die Antworten als Kommentare ans Ende Ihres Programms. Kommentarzeilen werden in MATLAB mit dem %-Zeichen eingeleitet. Sie werden feststellen, dass die Antworten vom benutzten Computer abhängen. Die rechnen nämlich, je nach Modell, an den Grenzen der Rechengenauigkeit etwas verschieden.

(b) In den Anwendungen kommen häufig Tridiagonalmatrizen vor (siehe z.B. die Steifigkeitsmatrix der Federkette). Für solche Matrizen lässt sich der Aufwand bei der Berechnung der Cholesky-Zerlegung erheblich reduzieren: Jede symmetrische positiv definite Tridiagonalmatrix A hat eine Cholesky-Zerlegung der Form:

$$\underbrace{\begin{bmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & b_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & b_{n-1} \\ & & & & b_{n-1} & a_n \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} c_1 & & & & & \\ d_1 & c_2 & & & & \\ & d_2 & c_3 & & & \\ & & d_3 & \ddots & & \\ & & & \ddots & & \\ & & & & d_{n-1} & c_n \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} c_1 & d_1 & & & & \\ & c_2 & d_2 & & & \\ & & c_3 & d_3 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & d_{n-1} \\ & & & & & c_n \end{bmatrix}}_{L^T}$$

Schreibe eine Matlab-Funktion `tridiagloes(a,b,f)`, welche das Gleichungssystem `Au=f` löst, ohne dass in den Zwischenrechnungen eine Matrix gebildet wird. Die Eingabedaten `a,b,f` sind Vektoren. Die Funktion berechnet zunächst die Koeffizienten c_j, d_j: erst c_1, dann d_1, dann c_2, usw.. Anschließend wird die Gleichung $Au = f$ durch Vorwärts- und Rückwärtseinsetzen mit der Matrix L bzw. L^T gelöst. Die

Matrix L wird dabei aber nicht wirklich gebildet (weil sie viele für die Rechnung überflüssige Nullen enthält).
Löse mit dieser Funktion das Gleichungssystem für die Federkette aus Aufgabe 4 für den Fall $s_1 = s_2 = \ldots = s_n = 1$, $f_1 = f_2 = \ldots, = f_n = 1$ und für verschiedene Werte von n. Wie hängen die Einträge des Verschiebungsvektors $[u_1 \;\; u_2 \;\; \ldots \;\; u_n]^T$ von n ab?
Tipp: Die Lösung für $n = 3$ ist $u = [3\ 5\ 6]$.

(c) Gegeben sei die Matrix

$$\texttt{ente} = \begin{bmatrix} 0 & 1 & 1 & 1.5 & 2.5 & 3 & 3 & 4 & 2 & 2 & 1 & -3.7 & -5 & -4 & -3 & 0 \\ 1 & 0 & 3.5 & 4 & 4 & 3.5 & 3 & 3 & 2 & -1 & -2 & -2 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Die Spalten dieser Matrix sind die Eckpunkte eines Polygonzuges, der wie eine Ente aussieht. Durch die Anweisung `plot(ente(1,:),ente(2,:))` wird der Umriss der Ente gezeichnet. Sei $A \in \mathbb{R}^{2\times 2}$. Durch die Multiplikation

$$\texttt{A} * \texttt{ente}$$

erhält man eine neue Matrix, deren Spalten die Eckpunkte der mit A multiplizierten (transformierten) Ente sind. Nach demselben Prinzip kann man eine Kreislinie transformieren. Zunächst erzeugt man sich eine Matrix `kreis`, deren Spalten Punkte auf dem Kreis sind (hierfür Polarkoordinaten benutzten: $x = r\,\cos(t),\ y = r\,\sin(t),\ t \in [0, 2\pi]$). Anschließend mit

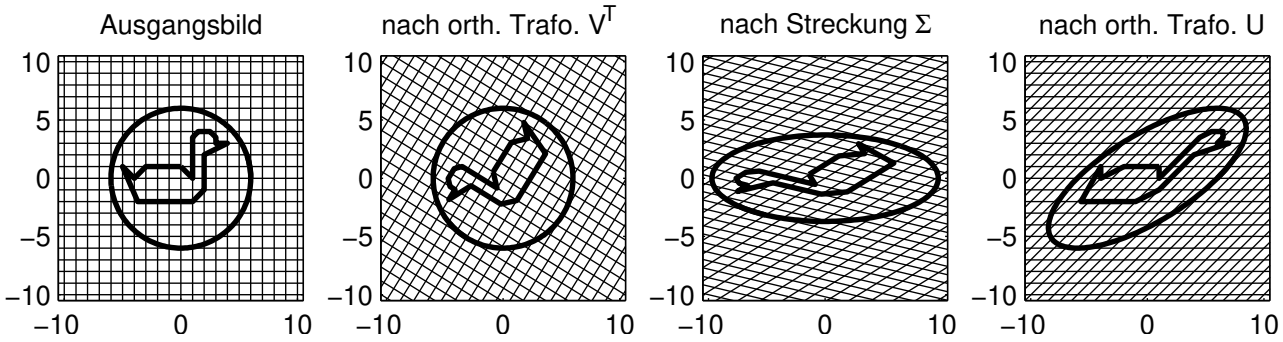
$$\texttt{A} * \texttt{kreis}$$

die Punkte auf dem mit A transformierten Kreis berechnen. Schreibe ein Programm, das

- von einer gegebenen Matrix A eine Singulärwertzerlegung $A = U\Sigma V^T$ berechnet. (Die Anweisung dazu ist `svd(A)`; eine Erklärung bekommt man mit `doc svd`),
- in einer Reihe die folgenden Bilder zeigt:

  1) die untransformierte Ente, und einen Kreis,
  2) die mit V^T transformierte Ente und den mit V^T transformierten Kreis,
  3) die mit $\Sigma\, V^T$ transformierte Ente und den mit $\Sigma\, V^T$ transformierten Kreis,
  4) die mit A transformierte Ente und den mit A transformierten Kreis.

Beispiel (das Hintergrundgitter ist aber nicht Bestandteil der Aufgabe):



Alle 4 Bilder sollen in einem Diagramm enthalten sein (`subplot` benutzen, analog zur Plot-Aufgabe auf Übungsblatt 1). Es muss nicht genau dieses Beispiel reproduziert werden. Sie dürfen gerne andere interessante Bildfolgen abgeben. Im gezeigten Beispiel sind die beiden Bilder in der Mitte im Vergleich zu den äußeren Bildern gespiegelt. Das ist rechnerabhängig. Die Singulärwertzerlegung ist nämlich nicht ganz eindeutig.