Übung 1 *LU-Zerlegung konkret*

Gegeben sei das Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} 0 & -4 & 10 & \frac{15}{2} \\ -2 & 6 & 3 & 10 \\ 2 & -6 & 7 & -\frac{11}{2} \\ -2 & 10 & -12 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 52 \\ 59 \\ -11 \\ -18 \end{pmatrix}.$$

- Berechnen Sie mit angegebenem Rechenweg die LU-Zerlegung von A mit Spaltenpivotisierung. Um den Tutoren die Arbeit etwas zu erleichtern wählen Sie bitte diese Pivotisierung:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Bestimmen Sie nun die Determinante von A sowie die Matrix A^{-1} und lösen Sie $Ax = b$.
- Berechnen Sie die Konditionszahl $\text{cond}_\infty(A)$.

(5 Punkte)

Übung 2 *Eine spezielle Tridiagonalmatrix*
Gegeben sei $T \in \mathbb{R}^{n\times n}$ von folgender Form:

$$T = \begin{pmatrix} a & b & & & & \\ c & a & b & & & \\ & c & a & b & & \\ & & \ddots & \ddots & \ddots & \\ & & & c & a & b \\ & & & & c & a \end{pmatrix}.$$

mit $bc > 0$.

(a) Zeigen Sie: T besitzt für $k = 1, ..., n$ die Eigenwerte

$$\lambda_k = a + 2\, b\, \nu \cos(\frac{k\pi}{n+1})$$

mit den Eigenvektoren

$$v_k = \left(\nu \sin(\frac{k\pi}{n+1}), \nu^2 \sin(2\frac{k\pi}{n+1}), \cdots, \nu^n \sin(n\frac{k\pi}{n+1})\right)^T.$$

wobei $\nu = \sqrt{\frac{c}{b}}$ ist.

Nützliche Hilfe: Für $l, x \in \mathbb{R}$ gilt die nützliche Formel:

$$2\cos(x)\sin(lx) = \sin((l+1)x) + \sin((l-1)x)$$

(b) Berechnen Sie für $a = 2$ und $b = c = -1$ die Kondition $\text{cond}_2(T)$. Geben Sie ihr Verhalten für $n \longrightarrow \infty$ in Landaunotation an.

(3+2 Punkte)

Übung 3 *Zur Gauß Elimination*

a) Nach k Schritten der Gauß-Elimination hat die Matrix $A^{(k)}$ folgende Blockgestalt:

$$A^{(k)} = \begin{bmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & B^{(k)} \end{bmatrix} \tag{1}$$

wobei $R_{11}^{(k)} \in \mathbb{R}^{k\times k}$, $R_{12}^{(k)} \in \mathbb{R}^{k\times(n-k)}$ und $B^{(k)} \in \mathbb{R}^{(n-k)\times(n-k)}$.
Zeigen Sie: Für die Blockzerlegung

$$B^{(k)} = \begin{bmatrix} \alpha^{(k)} & (w^{(k)})^T \\ \sigma^{(k)} & C^{(k)} \end{bmatrix} \tag{2}$$

mit $C^{(k)} \in \mathbb{R}^{(n-k-1)\times(n-k-1)}$ und $\sigma^{(k)}, w^{(k)} \in \mathbb{R}^{n-k-1}$, $\alpha^{(k)} \neq 0$ gilt die Rekursionsformel

$$B^{(k+1)} = C^{(k)} - \frac{1}{\alpha^{(k)}}\sigma^{(k)}(w^{(k)})^T. \tag{3}$$

b) Begründen Sie, dass auch folgender Algorithmus die LU Zerlegung einer Matrix $A \in \mathbb{R}^{n\times n}$ durchführt. Es wird angenommen, dass keine Zeilenvertauschungen notwendig sind. Außerdem wird das Ergebnis der LU-Zerlegung wieder direkt in der Matrix A gespeichert. Dabei gehören die Einträge unterhalb der Diagonalen zu L und die restlichen zu U. Ein formaler Beweis ist nicht verlangt.

for $(i = 2...n)$ **do**
  for $(j = 2...i)$ **do**
    $a_{i,j-1} = a_{i,j-1} \,/\, a_{j-1,j-1}$
    for $(k = 1..j-1)$ **do**
      $a_{i,j} = a_{i,j} - a_{i,k}\, a_{k,j}$
    end for
  end for
  for $(j = i+1...n)$ **do**
    for $(k = 1..i-1)$ **do**
      $a_{i,j} = a_{i,j} - a_{i,k} a_{k,j}$
    end for
  end for
end for

(2+3 Punkte)

Übung 4 *Dünnbesetzte Matrizen (Praktische Übung)*

In der Vorlesung haben wir gesehen dass viele Diskretisierungsmatrizen eine dünnbesetzte Struktur, oft in Bandform, aufweisen. Einerseits benötigen solche Matrizen prinzipiell deutlich weniger Speicherplatz, andererseits können bspsw. Matrix-Vektor-Produkte sehr effizient implementiert werden. Geeignete iterative Lösungsverfahren können unter Ausnutzung dieser Eigenschaften weit größere Gleichungssysteme lösen als das bei direkten Lösern möglich wäre. Zum Beispiel wären moderne Simulationsverfahren in der Flugzeugentwicklung ohne diesen Ansatz kaum machbar.

In der Praxis setzt das natürlich eine spezielle Implementierung einer Matrix-Datenstruktur voraus. Im Gegensatz zur Klasse `DenseMatrix` die wir bisher genutzt haben darf diese nur Matrixeinträge ungleich Null speichern, sollte aber weitgehend beliebige Strukturen der Besetzung sowie gängige Operationen unterstützen.

- Implementieren Sie eine entsprechende Klasse `SparseMatrix` nach dem vorgegebenen Programmgerüst. Hier wird jeder Matrixeintrag als ein struct namens `MatrixEntry` repräsentiert, indem Zeile und Spalte sowie Wert des Eintrags enthalten sind. Null-Einträge sollen nicht gespeichert werden, wodurch die Dünnbesetztheit ausgenutzt werden kann.

  Insbesondere soll es möglich sein, über die Methode `AddEntry` Einträge zur Matrix hinzuzufügen. Die Methode `mv` soll ein Matrix-Vektor-Produkt analog zur gleichnamigen Methode der `DenseMatrix`-Klasse liefern, aber nur auf den gespeicherten Nicht-Null-Einträgen arbeiten.

- Testen Sie Ihre Implementierung indem Sie die gleiche Matrix mit Bandstruktur sowohl als `DenseMatrix` als auch als `SparseMatrix` aufsetzen, mit einem beliebigen Vektor multiplizieren und das Ergebnis auf Gleichheit überprüfen. Sie können dazu die zuvor implementierte Matrix aus dem Rohrleitungsnetz benutzen.

- Sobald Sie sich der korrekten Funktion vergewissert haben können Sie die Skalierbarkeit Ihrer Implementierung gegen `DenseMatrix` testen. Führen Sie wie zuvor Matrix-Vektor-Produkte auf einer Bandmatrix aus, variieren Sie aber nun deren Größe $N \times N$, wobei $N = 2^n$ mit $n \in \{4, ..., 14\}$.

  Nutzen Sie die `Timer`-Klasse, um dabei die Rechenzeit der `mv`-Operation beider Implementierungen zu bestimmen. Plotten Sie diese Zeiten in einem log-log-Plot gegen die Matrixgröße. Plotten Sie zum Vergleich zusätzlich eine beliebige lineare und eine quadratische Funktion. Entspricht die Komplexität der Operationen Ihren Erwartungen?

(2+1+2 Punkte)