4. Projekt zur Veranstaltung "Numerische Mathematik II" QR-Verfahren und Cosinustransformation

Prof. Dr. Rainer Fischer

SS 2020

1. **QR-Verfahren:** (ca. 17 Punkte)

   (a) Implementieren Sie das einfache QR-Verfahren in Python als Funktion *qr1* (ohne Transformation auf Hessenberg-Form) und berechnen Sie damit alle Eigenwerte einer Matrix. Sie dürfen in jedem Schritt den Befehl *qr* zur QR-Zerlegung benutzen.

   (b) Implementieren Sie nun das QR-Verfahren mit Shift und Deflation als Funktion *qr2*. Machen Sie Ihr Verfahren außerdem robust, so dass es auch mit komplexen Eigenwerten zurecht kommt, d. h. für die entstehende $2x2$ Matrix entlang der Diagonalen die Eigenwerte mit Hilfe des charakteristischen Polynoms bestimmt. Zur Berechnung der benötigten QR-Zerlegungen dürfen Sie wieder den Befehl *qr* benutzen.

   (c) Verbessern Sie Ihr Verfahren noch weiter zu einer Funktion *qr3*, indem Sie die gegebene Matrix vorab durch Householder-Spiegelungen möglichst effizient auf obere Hessenberg-Form bringen.

   (d) Testen Sie Ihre drei Verfahren mit verschiedenen Matrizen und beschreiben Sie die Unterschiede im Konvergenzverhalten und der Anzahl an Iterationen. Benutzen Sie u. a. dazu u.a. die folgenden Matrizen: N-Matrizen, bei denen alle Elemente in der ersten und letzten Spalte sowie auf der Hauptdiagonalen 1 sind und alle übrigen Elemente 0. Testen Sie außerdem mit

$$A = \begin{pmatrix} 5 & 4 & 2 \\ 0 & 1 & -1 \\ -1 & -1 & 3 \end{pmatrix} \quad , \quad B = \begin{pmatrix} -1 & 1 & 3 \\ 3 & 3 & -2 \\ -5 & 2 & 7 \end{pmatrix}$$

$$C = \begin{pmatrix} -5 & -10 & -10 & 5 \\ 4 & 16 & 11 & -8 \\ 12 & 13 & 8 & -4 \\ 22 & 48 & 28 & -19 \end{pmatrix} \quad , \quad D = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

2. **Diskrete Cosinus-Transformation:** (ca. 16 Punkte)

   (a) Bestimmen Sie in Python mit Hilfe der diskreten 2D Cosinus-Transformation die Koeffizienten der interpolierenden Funktion $P_n(s,t)$

   - zu den Datenpunkten $(1,1,1)$, $(1,2,1)$, $(2,1,1)$, $(2,2,1)$ und $(i,j,0)$ für alle anderen $0 \leq i,j \leq 3$.
   - zu den Datenpunkten $(1,1,-1)$, $(1,2,-1)$, $(3,1,-1)$, $(3,2,-1)$ und $(i,j,3)$ für alle anderen $0 \leq i,j \leq 3$.

   Plotten Sie jeweils Funktion und Datenpunkte und schreiben Sie jeweils das interpolierende Polynom von Hand hin.

   (b) Verwenden Sie grayscale JPEG-Bilder Ihrer Wahl, die Sie in Python laden. In Moodle stehen dazu beispielsweise ein 8×8 Pixel großes und ein größeres Bild zur Verfügung. Außerdem steht Ihnen ein Programmgerüst zur Verfügung, das Sie im Folgenden ergänzen sollen.

   i. Transformieren Sie in der Funktion *quant1* den 8×8 Block mit der 2D DCT in den Frequenzbereich.

   ii. Führen Sie die lineare Quantisierung mit $p = 1$, $p = 2$ oder $p = 4$ und der Matrix aus dem Lesetext durch und rekonstruieren Sie daraus den Block durch die inverse 2D DCT.

   iii. Verwenden Sie nun als Quantisierungsmatrix die bei jpeg häufig verwendete Matrix

$$Q = p \cdot \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

   und $p = 1$, $p = 2$ oder $p = 4$.

   iv. Zerlegen Sie im Hauptprogramm das Bild in 8×8 Blöcke und führen Sie mit der Funktion *quant1* Transformation, Quantisierung und Rekonstruktion des gesamten Bildes durch. Plotten Sie jeweils Originalbild und rekonstruiertes Bild im Vergleich.

Hinweise:

- Spätester Abgabetermin für dieses Projekt ist Montag, der **13. Juli 2020** um 23.55 Uhr.
- Laden Sie dazu in einer Zip-Datei Ihre lauffähigen und kommentierten py-Files sowie eine separate Datei (Word, PDF, etc.) mit Erklärungen, Bewertungen, Plots und evtl. eingescannten Handrechnungen in Moodle hoch.
- Die Abgabe ist in 2er-Gruppen möglich. Bitte schreiben Sie in der hochgeladenen Datei **beide** Namen in den Dateinamen, damit eine eindeutige Zuordnung möglich ist.