Computerphysik WS 2017/2018

Prof. Dr. Petra Imhof, FU Berlin

Übungsblatt 3: Lineare Gleichungssysteme und Matrizen (20 Punkte)

7. November 2017

Abgabe bis **16.11.2017** 23:55 Uhr

Hinweis: Benutzen Sie numpy wie in den einzelnen Aufgaben angegeben.

Aufgabe 3.1: Gauss-Eliminierung (8 Punkte)

Betrachten Sie das folgende Gleichungssystem in Matrixschreibweise $\mathbf{Ax} = \mathbf{b}$

(a) (4 Punkte) Implementieren sie eine Funktion, welche die Matrix $\mathbf{A}$ (als $N \times N$ array) und den Vektor $\mathbf{b}$ (als $N \times 1$ array) als Eingabeparameter bekommt und das lineare Gleichungssystem mittles Gauss-elimination löst. Diese Funktion soll für die Berechnung der oberen Dreiecksmatrix (Eliminationsschritt) sowie die eigentliche Lösung durch Rückwärts - oder Vorwärtseinsetzen jeweils eine separate Funktion benutzen.

(b) (2 Punkt) Erweitern Sie Ihre Implementierung um die Möglichkeit der Pivotisierung.

(c) (2 Punkte) Nutzen Die die in (a) and (b) implementierten Funktionen un lösen Sie damit die folgende Gleichungssysteme einmal mit und einmal ohne Pivotisierung

Hinweis: Zum Testen Ihrer Ergebnisse können Sie zum Vergleich `np.linalg.solve()` *verwenden.*

$$\mathbf{A_1} = \begin{pmatrix} 1. & -1. & -10. & -9. & 8. & 5. \\ 1. & 4. & 5. & -1. & -4. & 3. \\ 3. & 0. & -1. & 5. & -4. & 5. \\ -8. & 7. & -7. & 8. & 2. & 7. \\ 0. & 5. & -9. & -7. & 2. & 1. \\ -2. & -6. & 10. & -8. & -5. & 3. \end{pmatrix} \text{ und } \mathbf{b_1} = \begin{pmatrix} 3. \\ -2. \\ -4. \\ 4. \\ 2. \\ 1. \end{pmatrix}.$$

$$\mathbf{A_2} = \begin{pmatrix} 1. & -3. & 2. & 2. \\ 2. & -6. & 1. & 4. \\ -1. & 2. & 3. & 4. \\ 0. & -1. & 1. & 1. \end{pmatrix} \text{ und } \mathbf{b_2} = \begin{pmatrix} -1. \\ 1. \\ 12. \\ 0. \end{pmatrix}.$$

Aufgabe 3.2: LU-Zerlegung (5 Punkte)

Gegeben sei die Matrix **A**.

$$\mathbf{A} = \begin{pmatrix} x & 1 & 9 & 7 & 3 \\ 6 & x & 4 & 5 & 9 \\ 8 & 9 & x & 0 & 5 \\ 6 & 6 & 5 & x & 4 \\ 2 & 4 & 2 & 2 & x \end{pmatrix}.$$

(a) (3 Punkte) Definieren Sie eine Funktion, welche eine beliebige, quadratische Matrix mittels LU-Zerlegung (mit optionaler Pivotisierung) in eine linksuntere (L) und rechtsobere (R) Dreiecksmatrix zerlegt, so dass gilt P*A=L*R mit P der Permutationsmatrix. Als Eingabeparameter soll die Funktion einen $(n \times n)$ `numpy.array` erhalten und $(n \times n)$ `numpy.arrays` mit R, L und P zurück geben.

(b) (2 Punkte) Berechnen Sie L und R für $\mathbf{x} = \mathbf{5}$ oder $\mathbf{x} = \mathbf{0}$ mit und ohne Pivotisierung. Besprechen Sie ihre Ergebnisse.

Aufgabe 3.3: Numpy 'Fingerübung' (7 Punkte)

In dieser Aufgabe sollen Sie den Umgang mit dem Python Paket `numpy` lernen. Aus diesem Grund sind Sie dazu aufgefordert *ausnahmsweise* die komplette numpy Bibliothek für diese Aufgabe zu nutzen.

(a) Array & Umformungen (3 Punkte)

Hinweis: Alle Unteraufgaben sind in einer Zeile zu lösen. Aufgabe (a)v kann ebenfalls in einer Zeile gelöst, werden, zwei sind aber auch erlaubt.

i) Erstellen Sie ein numpy Array D mit 100 Einträgen, wobei jedes folgende Element der Nachfolger des aktuellen Eintrags ist (e.g. [1, 2, 3, ...]).

ii) Berechnen Sie die den Abstand eines Elements zum folgenden Nachbarn für D (e.g. [1, 2, 3] → [2 - 1, 3 - 2] → [1, 1]). *Hinweis: Indexing*

iii) Geben Sie das 11. bis 21. Element von D aus.

iv) Formen Sie D zu einer 10×10 Matrix um → D_{Matrix}.

v) Erstellen Sie ein Array E mit 100 Einträgen, wobei jedes folgende Element der Vorgänger des aktuellen Eintrags ist (e.g. [100, 99, 98, ...]) und Formen Sie E zu einer 10×10 Matrix E_{Matrix} um.

(b) Informationen aus Arrays (2 Punkte)

Hinweis: Drei Zeilen Code sind pro Unteraufgabe zugelassen.

i) Berechnen Sie die Varianz, den Mittelwert und die Standardabweichung für jeweils eine Zeile in D_{Matrix}.

ii) Berechnen Sie die Varianz, den Mittelwert und die Standardabweichung für jeweils eine Zeile in E_{Matrix}.

(c) Lineare Algebra mit Arrays (2 Punkte)

Hinweis: Alle Unteraufgaben sind in einer Zeile zu lösen. Aufgabe (c)ii kann in zwei Zeilen gelöst werden.

i) Multiplizieren Sie $E_{Matrix} \cdot D_{Matrix}$ miteinander. (Matrizen Multiplikation)

ii) Wandeln Sie D in eine Matrix der Dimension 1×100 (D_{Vec})und E in eine Matrix der Dimension 1×100 (E_{Vec}) um.

iii) Multiplizieren Sie $D_{Vec} \cdot E_{Vec}^T$ miteinander. (Matrizen Multiplikation)

iv) Berechnen Sie die euklidische Distanz zwischen D_{Vec} und E_{Vec}.

Hinweis: Numpy Doc: `https://docs.scipy.org/doc/numpy-1.13.0/reference/index.html`