Wissenschaftliches Rechnen I

Wintersemester 2015/2016
Prof. Dr. Marc Alexander Schweitzer
Sa Wu

universitätbonn

Übungsblatt 0.

Aufwärmblatt, keine Abgabe, Besprechung in den Übungen 28.10, 30.10

- Die Website der Vorlesung findet sich unter `http://schweitzer.ins.uni-bonn.de/teaching/lectures/ws15_v3e1/`
- Dieses Blatt dient der Wiederholung und der Vorbereitung. Insbesondere soll auch ein (Wieder)Einstieg in Python/NumPy/matplotlib erleichtert werden.
- Bedingung für die Zulassung zur Klausur ist die erfolgreiche Teilnahme am Übungsbetrieb. Hierfür sollen *50%* der Punkte von normalen Aufgaben sowie *50%* der Punkte aus den Programmieraufgaben erreicht werden.
- Sie können die Aufgaben in Gruppen von bis zu 3 Studenten abgeben. Damit brauchen Sie pro 3 Studenten nur eine ordentliche, abgabetaugliche Reinschrift anfertigen. Für den Lernerfolg und das Bestehen der Klausur wird die Bearbeitung aller Aufgaben allen Hörern der Vorlesung nahe gelegt.
- Die Übungen finden im Seminarraum 5.002 im 5. Stock im HRZ in der Wegelerstraße 6 statt. Die vorläufigen Termine
  - Mittwoch, 0800-1000
  - Freitag, 1200-1400

  können auch in der ersten Vorlesung eventuell nach Bedarf geändert werden.
- Aufgrund der Veranstaltung “Panorama of Mathematics” (`http://www.hcm.uni-bonn.de/events/eventpages/2015/panorama`) entfällt die Vorlesung am 22.10.
- Upon sufficient demand, the lecture may be held in English.

Programming exercise 1. (Python/NumPy/matplotlib Wiederholung)

a) Lernen Sie ihre Kollegen kennen und finden Sie sich zu Abgabegruppen zusammen.

b) Besorgen Sie sich eine Installation von Python (`http://www.python.org`, `http://ipython.org`), NumPy (`http://numpy.org`) und matplotlib (`http://matplotlib.org`). Die Pakete sind auch in Rechnern der CIP Pools in der Wegelerstraße 6 und Endenicher Allee 60 und des Praktikumsraumes 6.020 in der Wegelerstraße 6 installiert. Sie sind als Teil des (sehr viel größeren) Gesamtpakets Anaconda (`https://www.continuum.io/why-anaconda`) erhältlich.

c) Alternativ ist ein VirtualBox-Image, bekannt aus der Vorlesung Einführung in die Numerik, mit den erforderlichen Paketen unter `http://ins.uni-bonn.de/teaching/vorlesungen/Numerik_SoSe15/software/software.html` erhältlich.

d) Machen Sie sich mit Python/NumPy/matplotlib vertraut, indem sie zu den Stützstellen $x_0 = -1, x_1 = 0, x_2 = 1$ die Lagrangepolynome

$$L_i = \prod_{j=0, j\neq i}^{2} \frac{x - x_j}{x_i - x_j}$$

für $i = 0, 1, 2$ plotten.

e) Überlegen Sie sich eine Abbildung $L_i \mapsto$ "'Mitglied Ihrer Abgabegruppe"' und bauen Sie eine dazu passende Legende in eine Grafik mit Plots aller L_i mit ein.

f) Erstellen Sie ein Bild von separaten Plots der L_i, nebeneinander und mit Bildunterschriften versehen.

Programming exercise 2. (Einfaches Anfangswertproblem)
Betrachten Sie das Anfangswertproblem

$$\epsilon y'(x) + y(x) = 0 \qquad\qquad y(0) = 1$$

aus der Vorlesung.

a) Wie lautet die exakte Lösung y_ϵ dieses Anfangswertproblems für $\epsilon \neq 0$?

b) Plotten Sie für $\epsilon = 2^{-k}, k = 1, \ldots, 10$ die exakte Lösung y_ϵ auf $[0, 1]$. Verwenden Sie dabei ein sinnvolles Plotkommando, welches in der Darstellung ϵ erkennen lässt. Eine Liste dieser Kommandos ist unter `http://matplotlib.org/api/pyplot_summary.html` einsehbar.

c) Implementieren Sie eine Python Methode, welche zu gegebenem ϵ und Maschenweite $h = \frac{1}{N}$ und $x_i = ih, i = 0, \ldots, N$ Approximationen $y_{\epsilon,i} \approx y_\epsilon(x_i)$ mit explizitem Euler Verfahren berechnet.

d) Erstellen Sie für $\epsilon = 2^{-1}$ und $\epsilon = 2^{-10}$ sinnvolle Konvergenzplots für $N = 2^1, \ldots, 2^{10}$ mit dem Fehler

$$e_{\epsilon,N} = \max_{i=0,\ldots,N} |y_\epsilon(x_i) - y_{\epsilon,i}| \, .$$

Exercise 1. (Vorkonditioniertes CG-Verfahren)

Es seien $A, C \in C^{n\times n}$ hermitesch, positiv definit und $b \in C^n$.
Betrachten Sie die Schritte des CG-Verfahrens zum Skalarprodukt $\langle x, y\rangle_C = \langle x, Cy\rangle$ vom Gleichungssystem

$$(AC)\, y = b\,, \qquad\qquad x = Cy\,.$$

Formen Sie die dabei entstehenden Ausdrücke so um, dass eine Berechnungsvorschrift für Approximationen von x entsteht und nur noch das euklidische Skalarprodukt vorkommt. Verwenden Sie dabei

$$e_k := Cd_k\,, \qquad\qquad s_k := Cr_k\,,$$

wobei r_k die Residuen und d_k Suchrichtungen sind, um berechnete Ausdrücke effizient wieder zu verwenden.
Das dabei entstehende Verfahren ist als Preconditioned Conjugate Gradient (PCG) bekannt. Dabei gibt es verschiedene Ansätze zur Wahl von C. Alternativ ist auch eine Konstruktion basierend auf $(CA)\, x = Cb$ möglich.

Programming exercise 3. (PCG für dünnbesetzte Matrizen)

Bei vielen in der Praxis auftretenden Linearen Gleichungssystem sind die Matrizen dünnbesetzt. Es sind also in der Matrix $A \in \mathbb{R}^{n\times n}$ in der Regel statt $O(n^2)$ nur noch $O(n)$ Matrixeinträge von 0 verschieden. Diese Struktur kann oft ausgenutzt werden, um die Komplexität der Algorithmen für solche Matrizen wesentlich zu reduzieren. In dieser Aufgabe soll mit sogenannten CSR-Matrizen, einer Speicherform von dünnbesetzte Matrizen, das PCG-Verfahren implementiert werden.

a) Machen Sie sich mit dem Modul `scipy.sparse` vertraut.
`http://docs.scipy.org/doc/scipy/reference/sparse.html`

b) Machen Sie sich mit der Konstruktion von CSR-Matrizen vertraut. Konstruieren Sie damit die 10×10 Matrix

$$A = \begin{pmatrix}
4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 16 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 64 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 256 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1024 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4096 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 16384 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 65536 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 262144 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1048576
\end{pmatrix}$$

Überprüfen Sie die Anzahl der Nichtnulleinträge der erzeugten CSR-Matrix.

c) Implementieren Sie eine Python Methode zur effizienten Bestimmung der invertierten Diagonalen $D(A)^{-1}$ einer dünnbesetzten CSR-Matrix aus `scipy`.

d) Implementieren Sie damit eine Python Methode zur iterativen Lösung von $Ax = b$ mit dem PCG-Verfahren mit Vorkonditionierer $C = D(A)^{-1}$. Das Verfahren soll dabei bei Erreichen einer maximalen Iterationszahl `maxIt` oder eines Fehlers $\|Ax - b\| <$ `eps` abbrechen.

e) Bestimmen Sie damit die Lösung von $Ax = b$ mit A aus b) und $b = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T$ mit Startvektor $x_0 = 0$, `maxIt`=100 und `eps`=10^{-13}.