Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Mario Ohlberger
M.Sc.-Math. Julia Brunken

21. September 2016

Übung zum Kompaktkurs

Einführung in die Programmierung zur Numerik mit Python

Sommersemesterferien 2016 — Hausaufgabe

Abgabe Ihres Programms als Mailanhang mit vollständigen Namen an julia.brunken@uni-muenster.de bis spätestens Mittwoch, 28. September 2016, 17:00 Uhr

Aufgabe 1 (Gauß-Algorithmus)

Sei $A \in \mathbb{R}^{n\times n}$ eine reelle Matrix und $b \in \mathbb{R}^n$ ein Vektor. Gesucht ist der Lösungsvektor $x \in \mathbb{R}^n$, für den gilt:

$$Ax = b$$

Aus der linearen Algebra kennen Sie den Gauß-Algorithmus mit Spalten-Pivot-Suche als ein Lösungsverfahren für ein solches lineares Gleichungssystem. Beim Pivoting wird im k-ten Schritt das betragsmäßig größte Element in der k-ten Spalte auf oder unterhalb der Diagonalen gesucht und die zugehörige Zeile anschließend mit der k-ten Zeile vertauscht.
Schreiben Sie ein `python` Programm namens $aufgabe1.py$ zur Realisierung des Gauß-Algorithmus. Verwenden Sie dabei `numpy.array` als grundlegende Datenstruktur für Vektoren als auch Matrizen. Gehen Sie in folgenden Schritten vor:

(a) Implementieren Sie eine Funktion, die das Pivoting durchführt, d.h. zu gegebenem k das Pivot-Element (und dessen Position) bestimmt.

(b) Schreiben Sie eine Funktion, die in einer gegebenen Matrix zwei Zeilen vertauscht.

(c) Implementieren Sie, aufbauend auf diesen zwei Funktionen, den Gauß-Algorithmus als Funktion, die zu gegebenem A, b das Gleichungssystem löst. Überprüfen Sie insbesondere, ob A und b vom richtigen Datentyp sind und ihre Größen richtig gewählt sind.

(d) Im Falle einer nicht invertierbaren Matrix soll Ihr Programm einen entsprechenden `ValueError` erzeugen. Überlegen Sie sich hierzu, woran man im Algorithmus diesen Fall erkennt.

Testen Sie das Verfahren an folgendem Gleichungssystem:

$$\begin{pmatrix} 2 & -1 & 5 & 7 \\ -1 & 1 & 1 & 4 \\ 1 & 2 & -3 & -2 \\ 3 & -1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 43 \\ 20 \\ -12 \\ 4 \end{pmatrix}$$

Aufgabe 2 (Finite Differenzen für eine elliptische Differentialgleichung)

Für eine hinreichend oft differenzierbare Funktion u lässt sich die zweite Ableitung in einem Punkt x durch folgenden Differenzenquotienten approximieren:

$$u''(x) \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}.$$

Betrachten Sie nun das folgende Randwertproblem: Gesucht ist u mit

$$u''(x) = x^2 \qquad \text{mit} \quad u(0) = u(1) = 0 \tag{1}$$

für eine Funktion $u : [0,1] \to \mathbb{R}$. Für die Diskretisierung dieses kontinuierlichen Problems setzt man $u_i \approx u(i \cdot h)$ für $h := \frac{1}{N+1}$, $N \in \mathbb{N}$. Mithilfe von obigem Differenzenquotienten erhält man nun ein lineares Gleichungssystem mit gesuchtem Lösungsvektor $(u_1, u_2, \ldots, u_N)^T$ sowie $u_0 = u_{N+1} = 0$.
Schreiben Sie ein `python` Programm namens *aufgabe2.py* zum numerischen Lösen des Randwertproblems. Zunächst implementieren Sie eine Funktion `diskrete_loesung` in Abhängigkeit von N, die die diskrete Lösung zurückgibt. Gehen Sie dabei in folgenden Schritten vor:

(a) Zur Bestimmung des gesuchten Lösungsvektors ergibt sich nach obigem Ansatz folgendes lineares Gleichungssystem:

$$(N+1)^2 \cdot \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & 0 \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ 0 & & & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} (\frac{1}{N+1})^2 \\ (\frac{2}{N+1})^2 \\ \vdots \\ (\frac{N-1}{N+1})^2 \\ (\frac{N}{N+1})^2 \end{pmatrix}$$

Implementieren Sie eine Funktion `matrix_vektor`, die als Argument die Zahl $N \in \mathbb{N}$ erhält und die gegebene Matrix als auch die gegebene rechte Seite als `numpy.array` der entsprechenden Größe zurückgibt.

(b) Verwenden Sie zum Lösen des Gleichungssystems aus (a) den Gauß-Algorithmus aus Aufgabe 1. Vereinfachen Sie gegebenfalls den Algorithmus im Hinblick auf die Gestalt der Matrix, indem Sie nur noch Haupt- und Nebendiagonalelemente der Matrix im Algorithmus berücksichtigen. Rufen Sie diesen Algorithmus mit den richtigen Argumenten in Ihrer Funktion `diskrete_loesung` auf, um die Lösung des Systems aus (a) zu erhalten.
Hinweis: Zur besseren Übersicht bietet es sich an, den modifizierten Gauß-Algorithmus in einer seperaten .py Datei zu speichern und dann zu importieren.

(c) Um die ganze diskrete Lösung zu erhalten, erweitern Sie anschließend den Lösungsvektor aus (b) um die Einträge u_0 und u_{N+1}.

Die exakte Lösung des Randwertproblems ist $u(x) = \frac{1}{12}x^4 - \frac{1}{12}x$. Nun sollen die exakte und die approximierte Lösung verglichen und visualisiert werden.

(d) Implementieren Sie die exakte Lösung als `python` Funktion.

(e) Implementieren Sie folgendes, simples Fehlermaß: Der durchschnittliche Fehler e zwischen der exakten Lösung u und der approximierten Lösung $\overline{u}$ sei definiert als

$$e(u, \overline{u}) = \frac{1}{N+2} \sum_{i=0}^{N+1} |u(i \cdot h) - u_i|.$$

Schreiben Sie eine entsprechende Funktion `fehler`, die den Vektor der exakten Werte und den Vektor der approximierten Werte erhält und den Fehler zurückgibt.

(f) Importieren Sie die `matplotlib` und implementieren Sie eine Funktion `visualisieren` zur Visualisierung der exakten und diskreten Lösung in einem Diagramm mit einer Legende im Intervall $[0, 1]$. Beachten Sie hierfür, dass Sie zur vernünftigen Darstellung der exakten Lösung deutlich mehr Definitionsstellen (x-Werte) als für die Darstellung der diskreten Lösung benötigen.

(g) **(Zusatzaufgabe)** Fassen Sie alle bislang implementierten Funktionen in einer Klasse `Diskretisierung` zusammen, die mit N initialisiert wird und im Konstruktor bereits die nötige Matrix und die nötigen Vektoren anlegt und den Lösungsvektor als Klassenvariable bestimmt. Ihre Klasse enthält Klassenmethoden zur Fehlerberechnung und Visualisierung, die von außen aufgerufen werden.

Testen Sie Ihr Programm für $N = 4, 8, 16, 32, 64$, dabei geben Sie für jedes N den Wert des Fehlermaßes aus (e) aus und nutzen Ihre Funktion aus (f) für die Visualisierung. Geben Sie abschließend einen Text aus, in dem Sie die Ergebnisse kurz bewerten.