In mathematischen Fragen darf man sich auch über den kleinsten Fehler nicht hinwegsetzen.
Isaac Newton, (1643-1727) englischer Mathematiker, Physiker und Astronom

UNIVERSITÄT DES SAARLANDES
FR 6.1 Mathematik
Prof. Dr. S. Rjasanow
L. Huwig, M.Sc.

8. Übung zur Vorlesung
Praktische Mathematik
im Sommersemester 2016

Abgabe: Donnerstag, den 16.06.2015 vor der Vorlesung.

Aufgabe 8.1. (4 Punkte)

Lösen Sie das lineare Gleichungssystem $Ax = b$ mit

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \quad , \quad b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

mit dem Verfahren der konjugierten Gradienten. Wählen Sie als Startvektor $x_0 = (0, 0, 0)^T$. Wieviele Iterationen sind (höchstens) dazu nötig?

Aufgabe 8.2. ((je 0.5) + (2 + 2) = 5.5 Punkte)

(a) Konvergiert die Potenzmethode, wenn die Eigenwerte der Matrix $A \in \mathbb{R}^{4\times 4}$

(i) $3, -1, -2$ und -5,
(ii) $4, 4, -1$ und -3,
(iii) $5, 3, 2$ und -5

sind? Begründen Sie Ihre Antwort.

(b) Seien $A \in \mathbb{R}^{2\times 2}$, $y_0 \in \mathbb{R}^2$,

$$A := \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix}, \quad y_0 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

gegeben.

(i) Berechnen Sie eine Nährung an den betragsgrößten Eigenwert $\lambda_{\max}$ von A mit der Potenzmethode. Führen sie 5 Schritte aus.

(ii) Alternativ werde folgende Iteration betrachtet

$$\begin{aligned} Aw_n &= w_{n-1} \\ w_0 &:= y_0 \quad \text{wie oben} \end{aligned}$$

Kann man auf diese Weise ebenfalls (einfache) Eigenwerte diagonalisierbarer Matrizen bestimmen?
Falls ja, führen Sie ebenfalls 5 Schritte durch und vergleichen Sie die Ergebnisse.

Normieren Sie die Iterationsvektoren jeweils mit der $\|\cdot\|_2$-Norm.
Hinweis: Verwenden Sie als Nährung für die gesuchten Eigenwerte den Rayleigh-Quotienten

$$R[x_{k+1}] = \frac{< x_k, x_{k+1} >}{< x_k, x_k >} \overset{k\to\infty}{\longrightarrow} \lambda_{\max}$$

Aufgabe 8.3. (Programmieraufgabe, Methode der konjugierten Gradienten, 2 + 2 + 2 + 2 = 8 Punkte) In dieser Aufgabe soll die Methode der konjugierten Gradienten implementiert werden. Insbesondere soll dieses Verfahren auch für dünnbesetzte Matrizen realisiert werden. Hierzu wurde das `Basic_LinAlg` Paket um einige Funktionen zum Lesen, Schreiben sowie zur Matrix-Vektor-Multiplikation von solchen Matrizen erweitert. Desweiteren wurden zwei Funktionen zur Berechnung des Skalarproduktes von zwei Vektoren und zur Skalierung von Vektoren hinzugefügt.

(a) Implementieren Sie die Funktion `cg_sparse()` im zur Verfügung gestellten Grundgerüst. Diese Funktion soll das cg-Verfahren für dünnbesetzte Matrizen realisieren. Neben den bisher bekannten Parametern wird dieser Funktion noch eine Abbruchgenauigkeit ε übergeben. Brechen Sie das Verfahren ab, sobald entweder die maximale Iterationszahl oder die Abbruchgenauigkeit für das Residuum

$$\frac{\|Ax_{m+1} - b\|}{\|Ax_0 - b\|} = \frac{\|r_{m+1}\|}{\|r_0\|} < \varepsilon$$

erreicht wurde. Die Funktion soll die Anzahl der durchgeführten Iterationsschritte zurück geben.

(b) Schreiben Sie ein Programm (`main_Teil1.c`), das die Lösung des linearen Gleichungssystems $Ax = b$, mit den Daten von der Internetseite, mit Hilfe des cg-Verfahrens approximiert. Bei `A_sparse_ascii.dat` handelt es sich um die selbe Matrix wie auf dem 7. Übungsblatt, allerdings im sparse Format. Als Startvektor wählen Sie $x^{(0)} = b$. Starten Sie den Algorithmus mit $K = 5, 10, 15, \ldots, 100$ Iterationsschritten und geben Sie die Anzahl der Iterationen und den relativen Fehler

$$\frac{\|x_K - x\|}{\|x\|}$$

auf der Konsole aus. (*Hinweis: Wählen Sie hierbei $\varepsilon = 0$.*)

Vergleichen Sie Ihre Ergebnisse mit denen zum Jacobi- und Relaxationsverfahren. Was stellen Sie fest?

(c) Ein Nachteil dieser Implementierung ist, dass das Verfahren nun lediglich für dünnbesetzte Matrizen anwendbar ist. Was aber wenn die Matrix voll abgespeichert ist oder im symmetrischen Format vorliegt? Implementieren Sie aus diesem Grund die Funktion `cg()`. Diese Funktion unterscheidet sich in zwei Argumenten: Die Matrix A wird als `void*` Zeiger übergeben und es existiert ein zusätzliches Argument, ein Funktionszeiger `mult` auf eine Funktion, die eine Matrix-Vektor-Multiplikation durchführt. Auf diese Weise übergibt der Benutzer eine Funktion zur Matrix-Vektor-Multiplikation und die Implementierung ist unabhängig vom verwendeten Format der Matrixspeicherung. (*Hinweis: Sie müssen lediglich zwei Zeilen aus Teil (a) verändern.*)

(d) Führen Sie die selben Tests aus wie in Teil (b) und validieren Sie Ihre Ergebnisse (`main_Teil2.c`). Verwenden Sie hierbei die voll abgespeicherte Matrix vom 7. Übungsblatt und schreiben Sie eine kleine Funktion (sogenannte Wrapper Funktion), die die Matrix-Vektor-Multiplikation ausführt und der `cg()` Methode übergeben werden kann. Mit Hilfe des Kommandozeilenbefehls `time` können Sie die Ausführungsdauer Ihres Programmes messen (z.B. `time ./a.out`). Was stellen Sie fest, wenn Sie die Geschwindigkeit Ihrer beiden Implementierungen vergleichen? Erklären Sie dieses Verhalten!