Prof. Dr. Karsten Urban
Dipl.-Math. oec. Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WiSe 2017/2018
Blatt 05
16.11.2017

Angewandte Numerik 2

Besprechung in den Tutorien in der Woche vom 20.11.2017 bis 24.11.2017

Für dieses Übungsblatt gibt es 10 Theorie- und 30 Matlab-Punkte, sowie 14 Theorie- und 15 Matlab-Zusatzpunkte. Punkte, die mit einem * gekennzeichnet sind, sind Zusatzpunkte.
Die 60-Prozent-Grenzen liegen aktuell (inklusive Blatt 05) bei 62,4 Theoriepunkten und 51,6 Matlabpunkten.

Aufgabe 18 *(cg-Verfahren)* (8T Punkte)

Betrachten Sie die quadratische Funktion $f(x) = \frac{1}{2}x^T Ax - b^T x$, $x \in \mathbb{R}^2$, mit

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix} \qquad \text{und} \qquad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Bestimmen Sie das Minimum von f unter Verwendung des cg-Verfahrens. Verwenden Sie als Startwert $x^{(0)} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$ und rechnen Sie mit Brüchen. Erklären Sie in jedem Schritt, welche anschauliche Bedeutung dieser Schritt hat. Sie müssen nicht den Algorithmus 2.4.1 aus dem Skript anwenden, sondern können sich an der Herleitung des cg-Verfahrens in der Vorlesung orientieren.
Wie viele Iterationsschritte haben Sie benötigt, um das exakte Minimum zu berechnen? Woran liegt das?

Aufgabe 19 *(Programmieraufgabe: cg-Verfahren)* (6M+3M+2M+3M* Punkte)

a) Schreiben Sie eine Matlabfunktion `xk = cgVerfahren(A, b, x0, maxIt, tol)`, die mit dem cg-Verfahren das lineare Gleichungssystems $Ax = b$ iterativ löst. `x0` soll dabei der Startwert und `maxIt` eine obere Schranke für die Anzahl der durchgeführten Iterationen sein. `tol` soll die Genauigkeit der Lösung steuern. Ihre Matlabfunktion soll im Vektor `xk` alle Iterationswerte $x^{(k)}$ zurückgeben.

b) Schreiben Sie ein Matlabskript `testCgVerfahren`, das Ihre Matlabfunktion `cgVerfahren` am Beispiel aus Aufgabe 14 vom letzten Übungsblatt 5 testet. Zeichnen Sie den Verlauf der Iteration in ein Schaubild. Zeichnen Sie in das gleiche Schaubild auch den Verlauf der Iteration des Gradienten-Verfahrens. Erläutern Sie das Schaubild.

c) Das cg-Verfahren ist gerade so konstruiert, dass die einzelnen Richtungen $\{d^{(1)}, \ldots, d^{(n)}\}$ paarweise A-orthogonal, also bezüglich des Energieskalarprodukts orthogonal, sind.

   Verifizieren Sie die paarweise A-Orthogonalität der einzelnen Richtungen $\{d^{(1)}, \ldots, d^{(n)}\}$ numerisch. Passen Sie dazu Ihre Matlabfunktion `xk = cgVerfahren(A, b, x0, maxIt, tol)` und/oder Ihr Matlabskript `testCgVerfahren` geeignet an.

d) Erweitern Sie Ihr Matlabskript `testCgVerfahren`. Zeichnen Sie in ein neues Schaubild den Verlauf der Iteration des Gradienten-Verfahrens und den Verlauf der Iteration des cg-Verfahrens für das Beispiel aus Aufgabe 18. Erklären Sie auch dieses Schaubild.

Aufgabe 20 *(Programmieraufgabe: Iterationsverlauf des cg-Verfahrens)* (4M*+2T* Punkte)

a) Erweitern Sie Ihr Matlabskript aus Aufgabe 15 vom letzten Übungsblatt: Zeichnen Sie in das Schaubild auch den Iterationsverlauf des cg-Verfahrens ein. Zur Berechnung der Iterierten können Sie Ihre Matlabfunktion `xk = cgVerfahren(A, b, x0, maxIt, tol)` aus der obigen Aufgabe 19 verwenden.

b) Wie verläuft die Iteration des cg-Verfahrens hinsichtlich der Höhenlinien?

Aufgabe 21 *(Programmieraufgabe: Konvergenzraten des Gradienten- und des cg-Verfahrens)* (10M+2T+2M*+3M*+2T* Punkte)

a) Schreiben Sie ein Matlabskript `konvergenzRaten`, mit dem Sie die Konvergenzraten des Gradienten-Verfahrens und des cg-Verfahrens vergleichen können.

Die Konvergenzrate eines Verfahrens im k-ten Schritt sei dabei wie in Aufgabe 16 definiert als $c^{(k)} = \frac{\|x^{(k)}-x^*\|}{\|x^{(k-1)}-x^*\|}$. x^* ist die exakte Lösung, die Sie mit dem Matlaboperator \ berechnen können. Als Norm können Sie die Euklidische Norm oder die Energie-Norm verwenden.

Wählen Sie als Testbeispiel eine Tridiagonalmatrix `A` der Dimension `n`, die auf der Diagonalen jeweils den Wert 2 und auf den beiden Nebendiagonalen jeweils den Wert -1 hat. Mit den Matlabbefehlen `e = ones(n,1); A = spdiags([-e 2*e -e], -1:1, n,n)` können Sie eine solche Matrix erzeugen. Wählen Sie als rechte Seite `b = ones(n,1)` und als Startwert `x0 = zeros(n,1)`. Testen Sie für die Dimension `n = 100`, aber auch mit anderen Werten für `n`.

Zeichnen Sie in ein Schaubild für jeden Iterationsschritt die Konvergenzrate des Gradienten-Verfahrens und in ein weiteres Schaubild die Konvergenzrate des cg-Verfahrens ein.

b) Erläutern Sie die Schaubilder. Erhalten Sie für unterschiedliche Werte von `n` qualitativ unterschiedliche Schaubilder?

c) Verifizieren Sie auch an diesem Beispiel numerisch die paarweise A-Orthogonalität der einzelnen Richtungen $\{d^{(1)}, \ldots, d^{(n)}\}$ des cg-Verfahrens.

d) Vergleichen Sie die Konvergenzraten der beiden Verfahren auch am Beispiel aus Aufgabe 16.

e) Erläutern Sie auch diese Schaubilder. Was passiert, wenn Sie a klein wählen, was, wenn a groß ist? Woran liegt das?

Aufgabe 22 *(Programmieraufgabe: Vorkonditioniertes cg-Verfahren)* (6M+3M+3M*+2T* Punkte)

a) Schreiben Sie eine Matlabfunktion `xk = pcgVerfahren(vorKond, A, b, x0, maxIt, tol)`, die mit dem pcg-Verfahren das lineare Gleichungssystem $Ax = b$ iterativ löst. `vorKond` ist dabei ein *function handle*, über das die Anwendung des Vorkonditionierers auf einen Residuums-Vektor $r^{(k)}$ realisiert wird. Überlegen Sie sich, welche Parameter Sie dieser Funktion übergeben müssen und welchen Wert sie zurückliefert. `x0` ist der Startwert für das pcg-Verfahren und `maxIt` eine obere Schranke für die Anzahl der durchgeführten Iterationen. `tol` soll die Genauigkeit der Lösung steuern. Ihre Matlabfunktion soll im Vektor `xk` alle Iterationswerte $x^{(k)}$ zurückgeben.

b) Schreiben Sie ein Matlabskript `konvergenzRatenA22`, mit dem Sie die Konvergenzraten des cg-Verfahrens und des vorkonditionierten cg-Verfahrens vergleichen können. Sie dürfen auch Ihr Matlabskript `konvergenzRaten` aus der vorigen Aufgabe anpassen.

Die Konvergenzrate eines Verfahrens im k-ten Schritt ist wie auf dem letzten Übungsblatt definiert als $c^{(k)} = \frac{\|x^{(k)}-x^*\|}{\|x^{(k-1)}-x^*\|}$. x^* ist dabei die exakte Lösung. Als Norm können Sie die Euklidische Norm oder die Energie -Norm verwenden.

Verwenden Sie zum Vergleich der beiden Verfahren die auf der Homepage in der Datei `A.txt` bereitgestellte Matrix. Diese können Sie mit dem Matlabbefehl `A = load('A.txt')` laden. Wählen Sie für die rechte Seite $b = A \cdot (1, ..., 1)^T$, für den Startwert $x^{(0)} = (0, ..., 0)^T$ und für die Toleranz `tol` den Wert 10^{-8}.

Die Vorkonditionierung soll mittels Diagonalskalierung erreicht werden.

c) Zeichnen Sie in ein Schaubild für jeden Iterationsschritt die Konvergenzrate des cg-Verfahrens und in ein weiteres Schaubild die Konvergenzrate des vorkonditionierten cg-Verfahrens ein. Dazu dürfen Sie Ihre Matlabfunktion `cgVerfahren(A, b, x0, maxIt, tol)` aus Aufgabe 19 verwenden.

d) Was beobachten Sie? Berücksichitigen Sie bei Ihren Erläuterungen auch die Laufzeiten der beiden Verfahren (Matlab-Befehle `tic` und `toc`).

Aufgabe 23 *(Gewöhnliche Differentialgleichungen: Richtungsfeld)* (5T*+3T* Punkte)

Gegeben sei die Differentialgleichung

$$y' = ay - by^3 \tag{1}$$

mit reellen Konstanten $a, b > 0$.

a) Skizzieren Sie das Richtungsfeld für $a = 4, b = 1$ in $-3 \leq t \leq 3, -3 \leq y \leq 3$.

b) Es sei y eine Lösung von (1) mit $y_0 = y(0)$. Argumentieren Sie alleine mit Hilfe des Richtungsfelds, wie sich $y(t)$ für $t \to \infty$ in Abhängigkeit von y_0 verhält.

Hinweise:
Die Programmieraufgaben sind in Matlab zu erstellen. Der Source Code muss strukturiert und dokumentiert sein. Senden Sie **spätestens 24 Stunden vor Ihrem Tutorium** alle Matlab-Files und alle Ergebnisse in einer E-mail mit dem Betreff **Loesung-Blatt05** an **angewandte.numerik@uni-ulm.de**.