D-ITET
Prof. Dr. J. Waldvogel

Numerische Methoden

SS 2003

# Serie 6

**1.** Ein Schritt des QR–Algorithmus zur Berechnung aller Eigenwerte von $A$ lautet in MATLAB $[\mathtt{Q},\mathtt{R}] = \mathtt{qr}(\mathtt{A}); \mathtt{A} = \mathtt{R} * \mathtt{Q};$.

**a)** Iterieren Sie dies mit der Matrix

$$A = \begin{pmatrix} 10 & 9 & 8 & 9 \\ 2 & 8 & 4 & 7 \\ 6 & 5 & 6 & 2 \\ 5 & 0 & 8 & 4 \end{pmatrix} \quad \in \mathbb{R}^{4\times 4}$$

bis zur Konvergenz gegen eine Matrix der Form

$$R = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} \quad (\times \neq 0).$$

Relle einfache und verschiedene Eigenwerte erscheinen in betragsmässig absteigender Reihenfolge auf der Diagonalen von $R$. In diesem Beispiel konvergiert der Diagonalblock $(r_{kl})$, $2 \leq k, l \leq 3$ nicht. Nach einigen Iterationen kann er jedoch zur Berechnung der komplexen Eigenwerte verwendet werden (ein Schul–Eigenwert–Problem).

**b)** Vergleichen Sie mit $\mathtt{eig}(\mathtt{A})$.

**2.** (Steilster Abstieg) Wir untersuchen das Konvergenzverhalten des *Verfahrens des (reinen) steilsten Abstiegs* (steepest descent) am homogenen linearen Gleichungssystem $Ax = 0$ mit $x \in \mathbb{R}^n$, $A = A^T$ und $A$ positiv definit, d.h.

$$F(x) := \frac{1}{2} x^T A x \; > \; 0 \quad \forall x \neq 0.$$

Mit dem beliebigen Startvektor $x_0 \in \mathbb{R}^n$ lautet die Iterationsvorschritt für $k = 0, 1, \ldots$

$$r_k = A x_k, \quad x_{k+1} = x_k - \varrho r_k.$$

**a)** Zeigen Sie, dass $F(x_k - \varrho r_k)$ minimal wird für

$$\varrho = \frac{r_k^T r_k}{r_k^T A r_k}.$$

**Bitte wenden!**

**b)** Seien
(i) $A = \mathrm{diag}(1, c^2)$ und $x_0 = (c, 1)^T$
(ii) $A = \mathrm{diag}(1, c, c^2)$ und $x_0 = (c^2, c, 1)^T$.

Führen Sie für (i) zwei Iterationsschritte von Hand durch und bestimmen Sie das Konvergenzverhalten für grosse $c$. Bestimmen Sie für (ii) das Konvergenzverhalten experimentell mit MATLAB. Wählen Sie dazu in (ii) $c = 10$ und verwenden Sie jeweils $\varrho$ aus Aufgabenteil a). Der Aufgabenteil (ii) ist fakultativ.

**3.** (**PA**) Sei $A_n$ die symmetrische Tridiagonalmatrix

$$A_n = \begin{pmatrix} 1 & 1 & & & & \\ 1 & 2 & 2 & & & \\ & 2 & 3 & 3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & n-2 & n-1 & n-1 \\ & & & & n-1 & n \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Berechnen Sie im Fall $n = 300$ die Euklidische Norm $\|A\|_2 = \max_i |\lambda_i|$ durch Vektoriteration *ohne* explizite Erzeugung von $A$ ($\lambda_i$ bezeichnen die Eigenwerte von $A$).

**a)** Schreiben Sie eine MATLAB–Funktion `y = sparsemult(x, n)` für die Abbildung

$$x \in \mathbb{R}^n \mapsto y := A_n x.$$

Diese Funktion kann zum Beispiel den Vektor `d = 1 : n` und Teile davon für eine effiziente Berechnung des Matrix–Vektor–Produktes verwenden. Achten Sie darauf, dass Sie keine Operationen mit Nullelementen von $A$ ausführen, und versuchen Sie, *ohne* `for`–Schleifen auszukommen: Machen Sie Gebrauch von der $.*$–Operation in MATLAB.

**b)** Vergleichen Sie die Zeiten, die MATLAB benötigt, um das Produkt $Ax$ zu berechnen (für ein beliebiges $x \in \mathbb{R}^{300}$). Berechnen Sie hierzu das Produkt $Ax$ mit Ihrer Funktion aus a) und vergleichen Sie mit der Berechnung des Matrix–Vektor–Produktes `A * x` von MATLAB, indem Sie die Matrix `A`

(i) "normal" definieren (also vollbesetzt) und
(ii) als `sparse` deklarieren.
Hinweis: Mit `tic; op; toc;` kann die Anzahl Sekunden zur Ausführung der operation `op` gemessen werden. Messen Sie auch die Zeiten, die zur Erzeugung von `A` benötigt werden (ausser natürlich bei Ihrer Funktion aus 3a))!

**c)** Bestimmen Sie nun mit Vektoriteration unter Verwendung Ihrer Funktion aus a) $\|A\|_2$. Normieren Sie die Näherung zum Eigenvektor in jeder Iteration auf Länge 1.