Einführung in die Numerische Mathematik

Wintersemester 2012/2013
Prof. Dr. S. Beuchler
Markus Burkow

universitätbonn

Übungsblatt 9. Abgabe am **Dienstag vor der Vorlesung (bis 10:15 Uhr)**.

Aufgabe 1. (Vektoriteration)

Gegeben sei eine diagonalisierbare Matrix $A \in \mathbb{R}^{n\times n}$ mit den Eigenwerten $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}$. Mit Hilfe der Vektoriteration sei der betragsmäßig größte Eigenwert λ_1 und ein zugehöriger Eigenvektor v_1 bestimmt worden.

a) Finde einen Vektor $v \in \mathbb{R}^n$, so dass $Q_v e_1 = \alpha v_1$ für die zugehörige Householder-Matrix Q_v und ein geeignetes $\alpha \in \mathbb{R}$ gilt.

b) Zeige, dass A durch

$$Q_v^T A Q_v = \begin{pmatrix} \lambda_1 & b^T \\ 0 & A_1 \end{pmatrix}$$

auf Blockstruktur transformiert werden kann, wobei $b \in \mathbb{R}^{n-1}$ und $A_1 \in \mathbb{R}^{(n-1)\times(n-1)}$ sind.

c) Zeige, dass die Matrix A_1 die Eigenwerte $\lambda_2, \ldots, \lambda_n$ besitzt.

Damit kann die Vektoriteration auf die kleinere Matrix A_1 angewendet werden, um den nächsten Eigenwert und einen zugehörigen Eigenvektor zu berechnen. Dieses als *Householder-Deflation* bekannte Verfahren kann iteriert werden, um alle Eigenwerte und alle Eigenvektoren zu bestimmen.

(5 Punkte)

Aufgabe 2. (QR–Algorithmus)

Zeigen Sie die Bemerkung aus der Vorlesung:
Die durch den QR–Algorithmus erzeugten Matrizen $A^{(k)}$ haben Hessenberg–Form, falls $A^{(1)}$ Hessenberg–Form hat.

(5 Punkte)

Aufgabe 3. (QR–Algorithmus mit doppelten Shift)

Die Anwendung des QR–Algorithmus auf reelle Matrizen mit komplexen Eigenwertpaaren führt unter Verwendung von reellen Shifts zu keiner Konvergenz. Um dies zu vermeiden, führt man einen sogenannten Doppelshift τ_k und $\tau_{k+1} = \overline{\tau}_k$ ein.
Zeigen Sie:
$A^{(k+2)}$ ist reell.

(5 Punkte)

Aufgabe 4. (QR–Verfahren)

Es sei $A^{(k)}$ die k-te Iterierte des QR-Verfahrens, wobei $A \in \mathbb{C}^{n\times n}$ diagonalisierbar ist mit $|\lambda_1| > \ldots > |\lambda_n|$, $q = \frac{|\lambda_2|}{|\lambda_1|}$, $\widetilde{q} = \frac{|\lambda_n|}{|\lambda_{n-1}|}$ und Eigenvektoren x_i mit $\|x_i\| = 1$. Es seien $\widetilde{Q}_k = Q_1 \cdot \ldots \cdot Q_{k-1} = [q_1^{(k)}, \ldots, q_n^{(k)}]$ und $\widetilde{R}_k = R_{k-1} \cdot \ldots \cdot R_1 = [r_{ij}]_{i,j=1}^k$ die aus der Vorlesung bekannten Matrizen. Damit gelten folgende Aussagen:

a) Sei $e_1 = \sum_i \alpha_i x_i$ mit $\alpha_1 \neq 0$. Damit gilt:

$$A^{(k)} e_1 = r_{11}^{(k)} q_1^{(k)}$$

b) Es sei $q_1 = x_1 + r$. Zeigen Sie, dass $\|r\| = \mathcal{O}(q^k)$ und dass damit folgende Abschätzung gilt:

$$\|q_1 - x_1\| \leq C \cdot q^k$$

c)

$$\lim_{k\to\infty} \|A^{(k)} e_1 - \lambda_1 e_1\|_2 = \mathcal{O}(q^k)$$

d) Sei A invertierbar, damit folgt:

$$\widetilde{Q}_k^* = \widetilde{R}_k A^{-(k+1)}$$

e)

$$\lim_{k\to\infty} \|e_n^* A^{(k)} - \lambda_n e_n^*\|_2 = \mathcal{O}(\widetilde{q}^k)$$

Zusammenfassend gilt:

$$A^{(k)} \approx \left(\begin{array}{c|ccc} \lambda_1 & & & \\ 0 & & * & \\ \vdots & & & \\ \hline 0 & \ldots & 0 & \lambda_n \end{array}\right)$$

(5 Punkte)

Programmieraufgabe 1. Gegeben sei eine symmetrische Tridiagonalmatrix $A \in \mathbb{R}^{n\times n}$. Man implementiere den **QR** Algorithmus zur Bestimmung der Eigenwerte von A.
Die dabei auftretenden orthogonalen Transformationen sind sinnvollerweise mit *Givens-Rotationen* durchzuführen. Hierfür kann der auf Aufgabenblatt 4 programmierte Givens-Algorithmus benutzt werden.
Man teste das Programm an den folgenden Beispielen :

a) $A = \begin{pmatrix} 2 & -1 & 0 & \ldots & \ldots & 0 \\ -1 & 2 & -1 & & & \vdots \\ 0 & -1 & 2 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & -1 \\ 0 & \ldots & \ldots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N\times N}$ für $N = 16, 32, 128, 512$.

Plotten Sie die Eigenwerte und in einem weiteren Plot den Betragsfehler zu den analytisch gegebenen Eigenwerten in Abhängigkeit der Iterationen.

$$\lambda_j = 2\left(1 - \cos(\frac{\pi \cdot j}{N+1})\right)$$

b) $B = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & -2 & -1 \end{pmatrix} \in \mathbb{R}^{7\times 7}$

c)

$$\mathcal{T}_n = \begin{pmatrix} \alpha_0 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \alpha_1 & \beta_2 & 0 & \dots \\ 0 & \beta_2 & \alpha_2 & \beta_3 & \\ \vdots & & & \ddots & \beta_{n-1} \\ 0 & \dots & 0 & \beta_{n-1} & \alpha_{n-1} \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{1}$$

mit $\alpha_i = 0$ und $\beta_i = \frac{i}{\sqrt{4i^2-1}}$. (s. Skript 1.41 und Beispiel 1.9 (Legendre–Polynome)).

Interpretieren Sie das Verhalten des Algorithmus. Wie müsste der Algorithmus verändert werden um die Probleme zu beheben? Begründung?

Die Abgabe der Programmieraufgaben erfolgt in den CIP-Pools am 17.12. und 18.12.2012. Die Listen für die Anmeldung zu den Abgabe-Terminen hängen in der kommenden Woche aus.