Einführung in die Numerische Mathematik

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

universitätbonn

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

Aufgabe 1. (GMRES-Verfahren)

Löse das System $Ax = b$,

$$A = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 3 & 2 & 1 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 2 \\ 1 \\ 3 \end{pmatrix},$$

mit Hilfe der GMRES–Verfahren zum Startwert $x_0 = \left(0, \frac{1}{2}, \frac{1}{2}, \frac{3}{2}\right)^T$ und $m = 4$.

(5 Punkte)

Aufgabe 2. (Vektoriteration)

Gegeben sei die Matrix:

$$A = \begin{pmatrix} -1 & -4 & 0 \\ 2 & 5 & 0 \\ 2 & 4 & 2 \end{pmatrix} \in \mathbb{R}^{n,n}$$

a) Führen Sie drei Iterationen mit der Vektoriteration aus. Als Startvektor $x_0 = (1, 0, 0)^T$.

b) Berechnen Sie die Eigenwerte und vergleichen Sie die Ergebnisse mit den Iterationen.

(5 Punkte)

Aufgabe 3. (Invarianzeigenschaften des Lanczos-Verfahrens)

Zeige die folgenden Eigenschaften des Verfahrens.

a) Das Lanczos-Verfahren erzeugt angewendet auf die Matrix $A - \sigma I$ für beliebiges $\sigma \in \mathbb{C}$ bei gleichem Startvektor x_0 stets dieselbe Matrix W_k.

b) Das Lanczos-Verfahren erzeugt angewendet auf die Matrix A mit Startvektor x_0 dieselbe Tridiagonalmatrix T_k wie für die Matrix $Q^H A Q$ mit Startvektor $Q^H x_0$, solange Q unitär ist.

(5 Punkte)

Aufgabe 4. (QR–Verfahren für symmetrische tridiagonale Matrizen)

Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch tridiagonal. Zur Berechnung der Eigenwerte mit dem QR–Verfahren wurde bisher die QR–Zerlegung mittels Givens–Rotationen berechnet. Eigenwerte und Eigenvektoren ließen sich aus den entstehenden Matrizen Q und R einfach ablesen. Falls lediglich die Eigenwerte von A von Interesse sind, kann ein effizienteres

Verfahren konstruiert werden.

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

1. Bestimme die Givens–Rotationsmatrix Q_0, welche die erste Spalte transformiert. Danach bilde das Produkt $B = Q_0^* A Q_0$
   Zeigen Sie:

$$B = \begin{pmatrix} \alpha_0' & \beta_1' & y & 0 & \dots & 0 \\ \beta_1' & \alpha_1' & \beta_2' & 0 & \dots & \\ y & \beta_2' & \alpha_2 & \beta_3 & \ddots & \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \beta_{n-1} \\ 0 & \dots & & 0 & \beta_{n-1} & \alpha_{n-1} \end{pmatrix}$$

2. Im Gegensatz zur normalen QR–Zerlegung mit Givens–Rotationen wird nun eine Rotationsmatrix U_1 bestimmt, welche den dritten und zweiten Eintrag in der ersten Spalte von B bearbeitet.
   Zeigen Sie:

$$U_1^* B U_1 = \begin{pmatrix} \alpha_0' & \beta_1'' & 0 & 0 & \dots & 0 \\ \beta_1'' & \alpha_1'' & \beta_2'' & y' & \ddots & \\ 0 & \beta_2'' & \alpha_2' & \beta_3' & 0 & \\ 0 & y' & \beta_3' & \alpha_3' & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & \beta_{n-1} \\ 0 & \dots & & 0 & \beta_{n-1} & \alpha_{n-1} \end{pmatrix}$$

3. Zeigen Sie, dass das sukzessive Anwenden dieser Vorgehensweise auf eine Tridiagonalmatrix $A^1 = U_{n-2}^* \cdot \ldots \cdot U_1^* B U_1 \cdot \ldots \cdot U_{n-2}$ führt.

4. Zeigen Sie, dass $\widetilde{U} = Q_0 \cdot U_1 \cdot \ldots \cdot U_{n-2}$ die korrekte Transformationsmatrix in einem Schritt des QR–Verfahrens angewendet auf A darstellt.

(5 Punkte)

Programmieraufgabe 1. (QR–Verfahren für tridiagonale Matrizen)

Ähnlich zur Programmieraufgabe auf Übungszettel 9 soll in folgender Programmieraufgabe der QR–Algorithmus implementiert werden. Hierbei soll speziell die tridiagonale Gestalt der Matrix ausgenutzt werden. Um eine speichereffiziente Darstellung und Berechnung der entstehenden Matrizen zu erhalten wird folgender Algorithmus vorgeschlagen:

Die Matrix A soll lediglich mit Hilfe von zwei Vektoren `a[n]` und `b[n-1]` gespeichert werden. Desweiteren soll die QR–Zerlegung und damit auch Auswirkung der Givens–Rotationsmatrix auf die betroffenen Zeilen im Multiplikationsschritt $A^{(k+1)} = Q_k^* A^{(k)} Q_k$ lokal ausgeführt werden. Es sollen explizit keine Matrix-Matrix-Produkte berechnet werden. Hierfür bietet sich folgende Herangehensweise an:

$$\mathcal{T}_n = \begin{pmatrix} a_1 & b_1 & 0 & \dots & 0 \\ b_1 & a_2 & b_2 & 0 & \dots \\ 0 & b_2 & a_3 & b_3 & \\ \vdots & & & \ddots & b_{n-1} \\ 0 & \dots & 0 & b_{n-1} & a_n \end{pmatrix} \in \mathbb{R}^{n \times n}$$

QR–Zerlegung $\Downarrow$ mit Givens–Rotationen

$$\mathcal{R} = \begin{pmatrix} g_1 & d_1 & e_1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & g_i & d_i & e_i & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & e_{n-2} \\ & & & & & \ddots & d_{n-1} \\ & & & & & & g_n \end{pmatrix} \quad + \quad c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \quad + \quad s = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}$$

Mit $c_i = \cos\phi_i$ und $s_i = \sin\phi_i$. Somit sind fünf Vektoren zu speichern um die QR–Zerlegung mit Givens–Rotationen durchzuführen. Mit diesen Vektoren ist somit das Produkt $A^{(k+1)} = R^{(k)}Q^{(k)}$ berechenbar ohne eine Matrix-Matrix-Multiplikation durchzuführen.

Zusammenfassend ergibt sich folgender Pseudocode für den QR–Algorithmus. Übersetzen Sie diesen in ein lauffähiges C/C++ Programm.

```
a i = A ii, i=1,...,n,
b i = A i,i+1 = A i+1,i, i=1,...,n-1
for k=1,...,itmax
    x 1 = a 1,
    y 1 = b 1
    for j=1,...,n-1
    % compute R, store c's & s's
        g j = x j^2 + b j^2
        c j = x j / g j,
        s j = b j / g j
        d j = c j * y j + s j * a j+1
        x j+1 = -s j * y j + c j * a j+1
        if j = n-1
            e j = s j * b j+1,
            y j+1 = c j * b j+1
        end
    end
    gn = xn
    a 1 = s 1 * d 1 + c 1 * g 1,
    b 1 = s 1 * g 2
    for j=2,...,n-1
    % compute R * s's & c's, store in A
        a j = s j * d j + c j * c j-1 * g j
        b j = s j * g j+1
    end
    a n = c n-1 * g n
    if (||b|| < ep) then break
end
```

In einem zweiten Programm soll der Algorithmus aus Aufgabe 4 implementiert werden, auch hierbei werden lediglich Hauptdiagonale und Nebendiagonale gespeichert. Damit sieht ein QR–Schritt in Pseudocode wie folgt aus:

```
x= a_1
```

```
y=b_1
for p=1...n-1
    if |x|< 10^-15
        w=-y, c=0, s=1
    else
        w=sqrt(x^2+y^2), c=x/w, s=-y/w
    end
    z = 2*c*b_p*s
    a_p = a_p-z
    a_p+1 = a_p+1 + z
    b_p = c*s+(c^2-s^2)*b_p
    x = b_p
    if p > 1 then b_p-1 = w end
    if p < n-1
        y=-s*b_p+1
        b_p+1 = c*b_p+1
    end
end
```

Plotten Sie Laufzeit beider Algorithmen anhand der Matrizen von Aufgabenblatt 9 Programmieraufgabe. Ziehen Sie zu Vergleichszwecken den Algorithmus der Programmieraufgabe des vorigen Aufgabenblattes hinzu.

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.