Prof. Dr. Stefan Funken
Dipl.-Math Markus Bantle
M.Sc. Andreas Bantle
Institut für Numerische Mathematik
Universität Ulm

Numerik 3
WS 12/13

Übungsblatt 11

(Besprechung Mo. 21.1. 2013)

Aufgabe 35 *(QZ Algorithmus)*

Ziel dieser Aufgabe ist es, den QZ Algorithmus für verallgemeinerte Eigenwertprobleme

$$Ax = \lambda Bx, \quad A, B \in \mathbb{R}^{n\times n}$$

zu realisieren. Im Folgenden gehen dir davon aus, dass A Hessenberg Form und B obere Dreiecksgestalt hat. Alle orthogonalen Transformationen werden als Householder-Spiegelungen realisiert.

(a) Die Funktion `[v,beta] = HouseholderVector(x)` (Download von der Homepage) bestimmt für einen Vektor `x` den Householdervektor `v` und den Skalierungsfaktor `beta`, so dass `x` durch die zu `v` und `beta` gehörige Householder-Spiegelung auf αe_1 abgebildet wird. Schreiben Sie eine Funktion `[v,beta] = HouseholderVectorRow(x)` die einen Vektor `v` und einen Skalierungsfakor `beta` berechnet, so dass die zugehörige Householder Spiegelung den Vektor `x` ($\in \mathbb{R}^n$) auf αe_n spiegelt.

(b) Überlegen Sie sich, wie Sie die in Teil (a) gegebenen Funktionen verwenden können, um folgende Householder-Transformationen zu realisieren:

$$x = \begin{pmatrix} * \\ * \\ * \end{pmatrix} \longrightarrow \begin{pmatrix} * \\ 0 \\ 0 \end{pmatrix} \qquad x = \begin{pmatrix} * \\ * \\ * \end{pmatrix} \longrightarrow \begin{pmatrix} 0 \\ 0 \\ * \end{pmatrix}$$

$$x = \begin{pmatrix} * & * & * \end{pmatrix} \longrightarrow \begin{pmatrix} * & 0 & 0 \end{pmatrix} \qquad x = \begin{pmatrix} * & * & * \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & * \end{pmatrix}$$

Der nächste Schritt ist die Realisierung eines QZ-Schritts. Zunächst müssen wir die Shift-Parameter a und b bestimmen. Dazu brauchen wir den unteren 2×2-Block der Matrix $M = AB^{-1}$.

(c) Leiten Sie her, wie die Einträge des Blockes $M(n-1:n, n-1:n)$ aussehen. Hinweis: Überlegen Sie sich, wieso Sie nur die Matrix-Blöcke $A(n-1:n, n-2:n)$ und $B^{-1}(n-2:n, n-1:n)$ betrachten müssen. Bestimmen sie die benötigten Einträge von B^{-1} durch geschicktes Rückwärtseinsetzen.

(d) Schreiben Sie eine Funktion `function [s,t] = eigM2by2(A,B)`, die $s = \text{trace}(\text{M(n-1:n, n-1:n)})$ und $t = \det(M(n-1:n, n-1:n))$ zurück gibt.

Nun müssen wir den Vektor $v = (M - bI)(M - aI)e_1 = (v_1, v_2, v_3, 0, \ldots, 0)^T$ in Abhängigkeit von s und t berechnen.

(e) Leiten Sie eine Strategie für die Berechnung der Einträge von v_1, v_2 und v_3 in Abhängigkeit von s und t her. Hinweis: Es gilt: $s = a + b$ und $t = a \cdot b$. Berechnen Sie zuerst $w := Me_1 = (w_1, w_2, 0, \ldots, 0)^T$ und $u = B^{-1}w$. Die ersten Spalten von B^{-1} können durch geschicktes Vorwärtseinsetzen bestimmt werden.

(f) Schreiben Sie eine Funktion `[v1,v2,v3] = Mtildee1(A,B,s,t)`, die die Einträge von v berechnet und zurück gibt.

(g) Schreiben Sie eine Funktion `function [A,B] = QZstep(A,B)`, in der ein QZ-Schritt durchgeführt wird.

Falls die Matrix B auf der Diagonalen eine Null enthält, können wir das ausnutzen um in $A(n-1,n)$ eine Null zu erzeugen.

(h) Schreiben Sie eine Funktion `function [A,B] = pushDown(A,B, nz)`, in der mittels geschickter Householder Spiegelungen die Null auf der Diagonalen von B an der Stelle $B(nz,nz)$ nach unten geschoben wird und so bei $A(n-1,n)$ eine Null erzeugt wird.

Schließlich können wir wie beim QR Verfahren einen Deflations-Algorithmus implementieren um A auf quasi-obere Dreiecksgestalt zu bringen. Ist dies geschehen, so können wir mit Hilfe der 1×1 bzw. 2×2 Blöcken auf der Diagonalen von A und der Diagonaleinträge von B die Eigenwerte λ_i bestimmen.

(i) Verstehen und vervollständigen Sie die Funktion `[A,B,ew] = QZAlgorithm(A,B,tol)`.

(j) Testen Sie die Funktion, in dem Sie die Ergebnisse mit denen der Matlab-Funktion `qz` bzw. `eig` (Angewandt auf $B\backslash A$) vergleichen