Interdisziplinäres Zentrum
für Wissenschaftliches Rechnen (IWR)
Ruprecht–Karls–Universität Heidelberg

Dr. Peter Zaspel

Übungsblatt 9

21.06.2016 – 28.06.2016

Einführung in die Numerik SS 2016

Aufgabe 1. QR-Zerlegung von rechteckigen Matrizen (2+2+0.5+1.5 Punkte)

Sei $A \in \mathbb{R}^{m \times n}$ mit $m \geq n$, $\text{Rang}(A) = n$. Sei $A = QR$ die QR-Zerlegung von A mit $R = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$ Zeigen Sie, dass dann gilt:

a. $A^T A \in \mathbb{R}^{n \times n}$ ist symmetrisch positiv definit.

b. $\text{cond}_2(A^T A) = \text{cond}_2(A)^2$.

c. Welche Nachteile sehen Sie darin, die Normalengleichung $A^T A$ über eine LR-Zerlegung zu lösen?

d. $\text{cond}_2(A) = \text{cond}_2(R) = \text{cond}_2(\tilde{R}) \geq \frac{\max\limits_i |r_{ii}|}{\min\limits_k |r_{kk}|}$.

Bemerkung: Für eine rechteckige Matrix A ist die Kondition allgemein definiert als

$$\text{cond}(A) := \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|y\|=1} \|Ay\|} \ .$$

Aufgabe 2. Curve-Fitting (2+2+1 Punkte)

Gegeben seien die folgenden Wertepaare:

z_i	0	0.15	0.31	0.5	0.6	0.75
y_i	1.0	1.004	1.031	1.117	1.223	1.422

Gesucht ist ein reelles Polynom $P(z)$ ersten und zweiten Grades, das den Fehler

$$\chi^2 = \sum_{i=1}^{6} |P(z_i) - y_i|^2$$

minimiert.

Zur Erinnerung: Ein reelles Polynom der Ordnung k hat die Gestalt: $P_k(z) = x_0 + x_1 z + ... + x_k z^k$ mit Koeffizienten $x_0, ..., x_k \in \mathbb{R}$.

a. Formulieren Sie das Problem um in die Gestalt: Finden Sie ein $x = (x_0, x_1, \ldots, x_k) \in \mathbb{R}^{k+1}$ so dass $\|Ax - b\|_2^2$ minimal ist, d.h. gebe jeweils (für $k = 1$ und $k = 2$) die zugehörige Matrix A sowie den Vektor b an.

b. Stellen Sie die Normalengleichung für $k = 1$ und $k = 2$ auf.

c. Ist die Lösung der Normalengleichung eindeutig?

Sie brauchen die Normalengleichung aus b) hier nicht lösen.

Aufgabe 3. QR-Zerlegung von A (5 Punkte)
Berechnen Sie die QR-Zerlegung der Matrix

$$A = \begin{bmatrix} 0 & 0 & 2 \\ -2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$$

per Hand mit Hilfe des Householder-Verfahrens.

Hinweis: In der Vorlesung hatten wir die Signum-Funktion sign(·) verwendet, um das Vorzeichen bei der Definition der Householder-Transformationen festzulegen. Dies geschah aus Stabilitätsgründen. In dieser Aufgabe werden Sie formal sign(0) auswerten müssen. Hier verwenden Sie bitte sign(0) := ±1, d.h. Ihnen steht frei, welches Vorzeichen Sie wählen. Die (ggf. klassische) Definition sign(0) = 0 führt hier zu keiner korrekten Lösung.

Aufgabe 4. Eigenwerte von Transformationsmatrizen (2+2 Punkte)

a. Berechnen Sie die Eigenwerte der Householder-Transformation $H = I - 2\frac{vv^\top}{v^\top v} \in \mathbb{R}^{n \times n}$.

b. Berechnen Sie die Eigenwerte der Givens-Transformation $G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \in \mathbb{R}^{2\times 2}$ mit $c^2 + s^2 = 1$.

Aufgabe 5. Praktische Umsetzung der QR-Zerlegung mittels Householder-Transformationen (5+5+0 Punkte)
In dieser Aufgabe soll das QR-Verfahren mittels Householder-Transformationen umgesetzt werden. Als Test-Matrix kommt dabei die Matrix aus Aufgabe 3 zum Einsatz. Im Moodle-Kurs finden Sie das Programmgerüst `householder.cpp`.

a. Implementieren Sie die Funktion `void apply_householder(double** A, double* v, int offset, int n)`, die die Householder-Matrix $H = I - 2vv^\top$ mit $\|v\| = 1$ auf die Systemmatrix A anwendet. Bei der Implementierung sollten Sie (optional) folgendermaßen vorgehen:

- `offset` kann verwendet werden um die Untermatrix zu spezifizieren bzgl. der die Householdertransformation konstruiert wird, d.h. `offset = 1` würde für die zweite Householdertransformation stehen, die die unteren Elemente der zweiten Spalte in A eliminiert.
- Bei Verwendung von `offset` kan es sich als sinnvoll erweisen, das Array `v` von voller Länge `n` zu spezifizieren, aber die Werte von $\vec{v}$ nur in die letzten `n-offset` Elemente einzutragen.
- Für Aufgabenteil b kann es sich als sinnvoll erweisen, die Householder-Trafo bzgl. eines Versatzes $offset$ auf alle unteren $n - offset$ Zeilen anzuwenden, statt nur auf den unteren rechten $(n - offset) \times (n - offset)$-Block.

b. Schreiben Sie die Funktion `void compute_qr(double** A, double** Q, int n)`, die für die $n \times n$-Matrix A die QR-Zerlegung mittels der Funktion `apply_householder` umsetzt. Nach der Ausführung von `compute_qr` soll in A die rechte obere Dreiecksmatrix stehen. Hier noch ein (optionaler) Tipp:

- Sie können die Matrix Q bauen, indem Sie sukzessive `apply_householder` auf eine Einheitsmatrix anwenden (Das Hauptprogramm übergibt die Matrix Q bereits als Einheitsmatrix.), und das erzeugte Produkt transponieren.

c. Das Hauptprogramm führt `compute_qr` für die Systemmatrix aus Aufgabe 3 aus und testet die Korrektheit Ihrer Implementierung durch aufmultiplizieren der Matrizen Q und R.

Abgabe: 28.06.2016, 14:00-14:15 Uhr. (Mappen in der Vorlesung)