Angewandte Mathematik
Modellierung und Numerik
Daniel Tenbrinck, Tim Roith, Lea Föcke

Einführung in die Numerik

Programmieraufgaben Blatt 03, Abgabe: Mittwoch, 07.12.2022, 12:00
Wintersemester 2022/2023

AUFGABE 1: Householder-Verfahren; 5 + 5 Punkte

In dieser Aufgabe implementieren wir die QR-Zerlegung einer Matrix $A \in \mathbb{R}^{m \times n}$ mit Hilfe von Householder-Reflexionen. Wir testen den Algorithmus an folgendem mathematischen Problem. Gesucht wird ein Polynom $P\colon \mathbb{R} \to \mathbb{R}$ vom Grad $n-1, n \in \mathbb{N}$, das möglichst gut $m \in \mathbb{N}$ gegebene Datenpaare $(x_i, y_i), i = 1, \ldots, m$ ausgleichen soll für $m \geq n$. Das Polynom ist hierbei gegeben durch

$$P(x) := p_1 x^{n-1} + \ldots + p_{n-1} x + p_n$$

mit reellen Koeffizienten $p_i \in \mathbb{R}, i = 1, \ldots, n$, die wir zu einem Vektor $p \in \mathbb{R}^n$ zusammenfassen.
Wir betrachten für dieses Problem die sogenannte **Vandermonde-Matrix** $V \in \mathbb{R}^{m,n}$ und die rechte Seite $b \in \mathbb{R}^m$, die wie folgt gegeben sind:

$$V := \begin{pmatrix} x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \ldots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_m^{n-1} & x_m^{n-2} & \ldots & x_m & 1 \end{pmatrix}, \qquad b := \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

Wir erhalten das optimale Polynom P indem wir nun das lineare Ausgleichsproblem $Vp = b$ lösen.

(i) Schreiben Sie eine Funktion `qr(A, mode='full', alg='Householder)` welche Algorithmus 3.21 aus der Vorlesung realisiert.

- Überprüfen Sie, dass $m \geq n$ gilt.
- Achten Sie hierbei darauf niemals die Householder-Matrizen explizit zu berechnen!
- Das keyword `mode` soll die Ausgabe der Funktion wie folgt definieren,
  - `mode='full'`: Hier werden die Matrizen `R` $\in \mathbb{R}^{m \times n}$ und `Q` $\in \mathbb{R}^{m \times m}$ ausgegeben.
  - `mode='reduced'`: Hier wird die reduzierte Zerlegung `R` $\in \mathbb{R}^{n \times n}$ und `Q` $\in \mathbb{R}^{m \times n}$ ausgegeben.
  - `mode='R'`: Hier wird **nur** `R` $\in \mathbb{R}^{n \times n}$ ausgegeben. Achten Sie darauf, dass `Q` hier **nicht** berechnet wird.

- Das keyword `alg` soll bestimmen welcher Algorithmus benutzt wird. In der aktuellen Aufgabe sollen Sie nur die Householder-Variante implementieren.

(ii) Erzeugen Sie Daten $(x_i, y_i) \in \mathbb{R}^2, i = 1 \ldots, m$ indem Sie das Intervall $[-3, 3]$ durch Punkte $-3 = x_1 < \ldots < x_m = 3$ diskretisieren und

$$y_i = \sin(3x_i) + x_i + \xi_i$$

setzten wobei ξ_i eine normalverteilte Zufallsgröße ist mit $\xi_i \sim \mathcal{N}(0, 0.05)$. Fitten Sie ein Polynom vom Grad $n-1$ zu diesen Daten, indem das lineare Ausgleichsproblem $Vp = b$ mit der Funktion `qr` lösen. Visualisieren Sie die Lösung für verschiedene $n \in \mathbb{N}$.
Hinweis: Benutzen Sie für die Zufallsgröße die Funktion `numpy.random.normal`.

— AUFGABE 2: Givens-Rotationen; 15 + 5 -Punkte —

Wir implementieren in dieser Aufgabe Givens-Rotationen um eine QR-Zerlegung zu erhalten. Dazu machen wir uns zunächst Folgendes klar. Es sei $a \in \mathbb{R}^2$ ein zweidimensionaler Vektor. Eine Rotation um den Nullpunkt mit Winkel θ kann mit der sogenannten **Givens-Matrix**

$$Q_\theta := \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} =: \begin{pmatrix} c & -s \\ s & c \end{pmatrix}$$

realisiert werden, siehe Abb. 1a.
Wir wollen nun a auf den Einheitsvektor e_1 drehen. Dazu berechnen wir

$$\begin{aligned} Q_\theta a &= \|a\| \, e_1, \\ \Leftrightarrow \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} &= \begin{pmatrix} \|a_1\| \\ 0 \end{pmatrix}, \\ \Leftrightarrow \begin{cases} ca_1 - sa_2 = \|a\|, \\ sa_1 + ca_2 = 0. \end{cases} \end{aligned}$$

Zwei mögliche Lösungen sind gegeben durch $c = a_1/\|a\|$ und $s = -a_2/\|a\|$, siehe Abb. 1b. Man beachte, dass wir den Winkel θ **nicht** explizit berechnen müssen!
Um nun einen Vektor $a \in \mathbb{R}^m$ auf $e_1^{(m)}$ zu rotieren, führen wir sukzessive zweidimensionale Rotationen durch. Dazu definieren wir die folgende Givens-Matrix $Q_{\theta,i,j} \in \mathbb{R}^{m,m}$

$$Q_{\theta,i,j} := \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

welche die Einträge i und j von a auf 1 und 0 respektive rotiert. Wenden wir nun diese Givens-Matrix auf die Matrix A an um die Einträge i und j der ersten Spalte auf 0 und 1 zu rotieren, so müssen wir $Q_{\theta,i,j}$ **nicht** explizit berechnen, denn es gilt

$$Q_{\theta,i,j}\begin{pmatrix} A_{1,1} & \dots & A_{1,n} \\ \vdots & & \vdots \\ A_{i,1} & \dots & A_{i,n} \\ \vdots & & \vdots \\ A_{j,1} & \dots & A_{j,n} \\ \vdots & & \vdots \\ A_{m,1} & \dots & A_{m,n} \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ \vdots & \vdots & & \vdots \\ 1 & cA_{i,2} - sA_{j,2} & \dots & cA_{i,n} - sA_{j,n} \\ \vdots & & & \vdots \\ 0 & sA_{i,2} + cA_{j,2} & \dots & sA_{i,n} + cA_{j,n} \\ \vdots & & & \vdots \\ A_{m,1} & & \dots & A_{m,n} \end{pmatrix}.$$

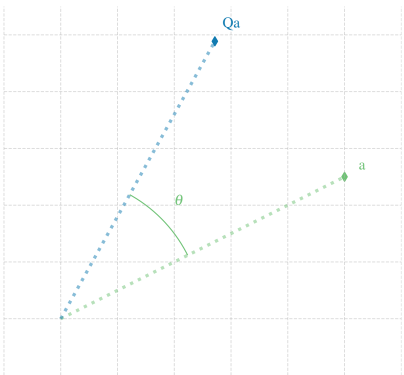
Es werden also nur die i-te und j-te Zeile verändert. Die genau Reihenfolge in welcher wir die Zeilen rotieren ist relevant, die Transformation der ersten Spalte könnte z.B. so aussehen,

$$Q_{\theta_1,1,2} \dots Q_{\theta_m,m-1,m} A$$

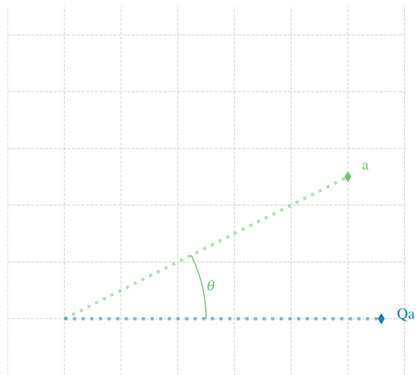
wobei wir unten anfangen und alle Paare nach oben hin entsprechend zum Einheitsvektor rotieren.

(i) Fügen Sie zur Funktion `rq` aus Aufgabe 1 die Option `alg='Givens'` hinzu, indem Sie die QR-Zerlegung mit Givens-Rotationen wie oben beschrieben implementieren.

(ii) Testen Sie Ihre Implementierung anhand geeigneter Beispiele. Überlegen Sie sich insbesondere ein Beispiel, bei dem der Givens-Algorithmus Vorteile gegenüber der Householder-Variante hat.



(a) Visualisierung einer Givens-Rotation um den Winkel θ.



(b) Visualisierung einer Givens-Rotation auf e_1.