Programmierpraktikum für Fortgeschrittene

Vergleich zweier Implementationen des Householder-Verfahrens

von
Michael Conrad

WS 97/98
BUGH Wuppertal

Inhaltsverzeichnis

1 QR-Zerlegung nach Householder **2**

1.1 Householder-Matrizen 2

1.2 Das Householder-Verfahren 4

1.2.1 1. Schritt 4

1.2.2 k. Schritt 4

2 Die Implementierung **5**

3 Ein Vergleich von `H_QRxsc` und `H_QRst` **8**

4 Das Programm **12**

1 QR-Zerlegung nach Householder

Jede reguläre $(n \times n)$-Matrix A läßt sich in ein Produkt der Form $A = Q \cdot R$ mit einer orthogonalen Matrix Q und einer oberen Dreiecksmatrix R zerlegen. Bei der Methode von Householder wird dabei in jedem Berechnungsschritt des gleichnamigen Algorithmus eine Spiegelung des $\mathbb{R}^n$ konstruiert. Diese Aufgabe übernehmen die sogenannten Householder-Matrizen, die in Abschnitt 1.1 mit ihren Eigenschaften eingeführt werden. Der daraus resultierende Algorithmus wird in Abschnitt 1.2 vorgestellt, seine Implementierung wird in Abschnitt 2 erläutert. Im vorliegenden Programm ist der Algorithmus zweimal implementiert: zum einen wird er mit den in PASCAL-XSC zur Verfügung gestellten Möglichkeiten des exakten Rechnens ausgeführt. Dies leistet die Prozedur `H_QRxsc`. Zum anderen wird die gleiche Zerlegung in `H_QRst` mit der Arithmetik des Standard-PASCAL noch einmal vollzogen und anschließend werden die verschiedenen resultierenden Zerlegungen `Qxsc`, `Rxsc` bzw. `Qst` und `Rst` miteinander verglichen.

1.1 Householder-Matrizen

Im folgenden sei stets mit $n \in \mathbb{N}$ die Dimension des zu Grunde liegenden Raumes $\mathbb{R}^n$ bezeichnet und $M_n(\mathbb{R})$ seien die quadratischen $(n \times n)$-Matrizen mit reellen Koeffizienten. Ist $A \in M_n(\mathbb{R})$ ein Matrix, so bezeichnet a^j für $j = 1, \dots, n$ den j-ten Spaltenvektor und a_{ij} die Komponenten. Weiter sei $I \in M_n(\mathbb{R})$ die Einheitsmatrix.

Definition 1. *Eine Matrix $H \in M_n(\mathbb{R})$ heißt Householder-Matrix, wenn es einen Vektor $h = (h_1, \dots, h_n) \in \mathbb{R}^n$ der euklidischen Länge $\|h\| = 1$ gibt, so daß H von der Form $H = I - 2hh^T$ ist.*

Wir schreiben auch H_h, falls der erzeugende Vektor h explizit angegeben werden soll.
Offenbar sind Householder-Matrizen von der Gestalt

$$H = \begin{pmatrix} 1-2h_1^2 & -2h_1h_2 & \cdots & -2h_1h_n \\ -2h_2h_1 & 1-2h_2^2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ -2h_nh_1 & \cdots & \cdots & 1-2h_n^2 \end{pmatrix}$$

und daher symmetrisch. Weiter gilt

$$H^2 = (I - 2hh^T)^2 = I - 4hh^T + 4hh^Thh^T = I$$

wegen $h^Th = 1$, also sind Householder-Matrizen orthogonal. Bezeichnet man mit

$$h^\perp := \{x \in \mathbb{R}^n : \langle x, h\rangle = 0\}$$

den zu Span(h) orthogonalen $(n-1)$-dimensionalen Unterraum des $\mathbb{R}^n$, so beschreibt die Transformation $x \mapsto Hx$ die Spiegelung des $\mathbb{R}^n$ an $h^\perp$. Zerlegt

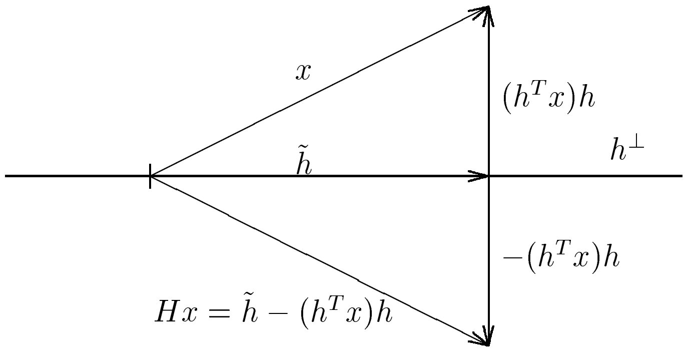

Abbildung 1: Die Householder-Transformation

man nämlich x in seine Komponenten in Richtung von h und den dazu orthogonalen Anteil, also $x = (h^T x)h + (x - (h^T x)h)$, so folgt

$$\begin{aligned} Hx &= (I - 2hh^T)x \\ &= (h^T x)h + (x - (h^T x)h) - 2hh^T(h^T x)h \\ &= -(h^T x)h + (x - (h^T x)h). \end{aligned}$$

Gilt für Vektoren $a, b \in \mathbb{R}^n$ die Beziehung $Ha = b$, so folgt $\|a\| = \|b\|$ und $a - b = 2(h^T a)h$, d.h. man erhält $h = \pm\frac{a-b}{\|a-b\|}$, falls $a \neq b$ ($\Leftrightarrow$ $a \notin h^\perp$). Man kann also den die Householder-Matrix definierenden Einheitsvektor h bis auf das Vorzeichen rekonstruieren. Sind umgekehrt $a, b \in \mathbb{R}^n$ mit $\|a\| = \|b\|$ und $a \neq b$, so gilt mit $h := \pm(a - b)/\|a - b\|$

$$H_h a = b \quad \text{und} \quad H_h b = a.$$

Ist $(q^1, \ldots, q^n)$ eine Orthonormalbasis des $\mathbb{R}^n$, so ist wegen der Orthogonalität von H auch $(Hq^1, \ldots, Hq^n) =: (q'^1, \ldots, q'^n)$ eine Orthonormalbasis des $\mathbb{R}^n$. Für jedes $x \in \mathbb{R}^n$ und $i \in \{1, \ldots, n\}$ erhält man dann:

$$\begin{aligned} x^T q'^i &= x^T H q^i \\ &= (H^T x)^T q^i \\ &= (Hx)^T q^i \\ &= (x - 2(h^T x)h)q^i \\ &= x^T q^i - 2(h^T x)h^T q^i \\ &= x^T q^i - 2(h^T q^i)\sum_{j=1}^{n}(x^T q^i)(h^T q^j) \\ &= [1 - 2(h^T q^i)^2](x^T q^i) - 2(h^T q^i)\sum_{j=1, j\neq i}^{n}(x^T q^j)(h^T q^j) \end{aligned}$$

Ist $h = \frac{u}{\|u\|}$, so berechnet man die Koeffizienten von x bezüglich $(q'^1, \ldots, q'^n)$ aus den Koeffizienten von u bezüglich $(q^1, \ldots, q^n)$ mit $N := \frac{1}{2}\|u\|$ also wie folgt:

$$x^T q'^i = \left[1 - \frac{1}{N}(u^T q^i)^2\right](x^T q^i) - \frac{1}{N}(u^T q^i)\sum_{j=1, j\neq i}^{n}(x^T q^j)(u^T q^j)$$

Es sind nun die erfordelichen Hilfsmittel zur Verfügung gestellt, um den resultierenden Algorithmus einzuführen.

1.2 Das Householder-Verfahren

Gegeben: Eine invertierbare Matrix $A \in M_n(\mathbb{R})$ mit den Spaltenvektoren $a^1, \ldots, a^n \in \mathbb{R}^n$.

Gesucht: QR-Zerlegung von A, d.h. orthonormale $q^1, \ldots, q^n \in \mathbb{R}^n$ und $r_{ij} \in \mathbb{R}$, $i = 1, \ldots, j, j = 1, \ldots, n$ mit

$$a^j = \sum_{i=1}^{n} r_{ij} q^i, \quad j = 1, \ldots, n.$$

1.2.1 1. Schritt

Bezeichnet e^i den i-ten Einheitsvektor des $\mathbb{R}^n$, so ist durch $q^{i(0)} := e^i$ eine Orthonormalbasis $(q^{1(0)}, \ldots, q^{n(0)})$ des $\mathbb{R}^n$ gegeben. Da die Spaltenvektoren $a^1, \ldots, a^n$ linear unabhängig sind, ist $\delta := a^1 \neq 0$. Für wenigstens ein $\sigma \in \{-1, 1\}$ ist daher

$$\delta \neq \sigma \|\delta\| q^{1(0)}.$$

Beide Vektoren haben jedoch die gleiche euklidische Länge $\|\delta\|$. Setzt man nun

$$u := \delta - \sigma \|\delta\| q^{1(0)}, \qquad h := \frac{u}{\|u\|}, \tag{1}$$

so wird $\sigma \|\delta\| q^{1(0)}$ durch H_h auf δ gespiegelt und umgekehrt. Ferner ist mit $q^{i(1)} := H_h q^{i(0)}, i = 1, \ldots, n$ durch $(q^{1(1)}, \ldots, q^{n(1)})$ eine Orthonormalbasis des $\mathbb{R}^n$ gegeben und aus

$$\begin{aligned} H_h(\sigma \|\delta\| q^{1(0)}) &= \sigma \|\delta\| (H_h q^{1(0)}) \\ &= \sigma \|\delta\| q^{1(1)} \\ &= \delta \\ &= a^1 \end{aligned}$$

folgt

$$a^1 = \sigma \|\delta\| q^{1(1)} =: r_{11} \cdot q^{1(1)}.$$

1.2.2 k. Schritt

Es sei $2 \leq k \leq n$. Zu Beginn des k-ten Schrittes liegt eine Orthonormalbasis $(q^{1(k-1)}, \ldots, q^{n(k-1)})$ des $\mathbb{R}^n$ vor und es gilt

$$a^j = \sum_{i=1}^{j} r_{ij} q^{i(k-1)}, \quad j = 1, \ldots, (k-1)$$

sowie

$$a^j = \sum_{i=1}^{n} \alpha_{ij} q^{i(k-1)}, \quad j = k, \ldots, n$$

mit $\alpha_{ij} = a^{jT} q^{i(k-1)}$. Damit ist durch

$$a^k = \sum_{i=1}^{k-1} \alpha_{ik} q^{i(k-1)} + \underbrace{\sum_{i=k}^{n} \alpha_{ik} q^{i(k-1)}}_{=:\delta}$$

ein Vektor $\delta \neq 0$ gegeben, da sonst die linear unabhängigen Vektoren $a^1, \ldots, a^k$ in Span$(q^{1(k-1)}, \ldots, q^{k-1(k-1)})$ enthalten wären.
Analog zum ersten Schritt gilt für wenigstens ein $\sigma \in \{-1, 1\}$ nun $\delta \neq \sigma \|\delta\| q^{k(k-1)}$.
Wieder setzt man

$$u := \delta - \sigma \|\delta\| q^{k(k-1)}, \qquad h := \frac{u}{\|u\|}, \tag{2}$$

und $\sigma \|\delta\| q^{k(k-1)}$ wird durch H_h auf δ gespiegelt und umgekehrt. Ferner ist mit $q^{i(k)} := H_h q^{i(k-1)}$ durch $(q^{1(k)}, \ldots, q^{n(k)})$ eine Orthonormalbasis des $\mathbb{R}^n$ gegeben. Mit dieser Festlegung wird das, was in den vorangegangenen $(k-1)$ Schritten erreicht wurde, nicht zerstört, insbesondere gilt $Hq^{i(k-1)} = q^{i(k-1)}$ und aus

$$H_h(\sigma \|\delta\| q^{k(k-1)}) = \sigma \|\delta\| q^{k(k)} = \delta$$

folgt

$$\begin{aligned} a^k &= \sum_{i=1}^{k-1} \underbrace{\alpha_{ik}}_{=:r_{ik}} \underbrace{q^{i(k-1)}}_{=q^{i(k)}} + \underbrace{\sigma \|\delta\| q^{k(k)}}_{=:r_{kk}} \\ &= \sum_{i=1}^{k} r_{ik} q^{i(k)}, \end{aligned}$$

d.h. man ist einen Schritt näher am Ziel. Insbesondere ist nach Beendigung des n-ten Schrittes durch $(q^{1(n)}, \ldots, q^{n(n)})$ eine Orthonormalbasis des $\mathbb{R}^n$ gegeben und es gilt $A = Q \cdot R$.

2 Die Implementierung

Zu Beginn des k-ten Schrittes $(k \geq 1)$ liegen folgende Größen vor:

1. Eine Orthonormalbasis $(q^{1(k-1)}, \ldots, q^{n(k-1)})$ des $\mathbb{R}^n$, die durch die Komponenten $q_{l,i} \in \mathbb{R}$, $l, i = 1, \ldots, n$, gegeben ist und für die

$$q^{i(k-1)} = \sum_{l=1}^{n} q_{l,i} e^l = e^{lT} q^{i(k-1)}$$

gilt. Die Initialisierung vor dem 1. Schritt erfolgt dabei durch $q_{l,i} = \delta_{li}$, d.h. $(q^{1(0)}, \ldots, q^{n(0)}) = (e^1, \ldots, e^n)$.

2. Die Elemente r_{ij} mit $i = 1, \ldots, j, j = 1, \ldots, k-1$ der Dreiecksmatrix R. Offenbar gilt für $j = 1, \ldots, k-1$

$$a^j = \sum_{i=1}^{j} r_{ij} q^{i(k-1)},$$

dies entfällt im ersten Schritt.

3. α_{ij} für $i = 1, \ldots, n, j = k, \ldots, n$ mit $\alpha_{ij} = a^{jT} q^{i(k-1)}$ und

$$a^j = \sum_{i=1}^{n} \alpha_{ik} q^{i(k-1)} = \sum_{i=1}^{k-1} \alpha_{ik} q^{i(k-1)} + \underbrace{\sum_{i=k}^{n} \alpha_{ik} q^{i(k-1)}}_{=:\delta \neq 0}$$

Zu Beginn des 1. Schrittes ist $\alpha_{ij} = a_{ij}$ für $i, j = 1, \ldots, n$.

Man beachte, daß bei der Implementierung die neu zu berechnenden α_{ij} in `A[i,j]` gespeichert werden.

Nach dem Householder-Verfahren gemäß Abschnitt 1.2 sind nun folgende Rechenschritte zu leisten, wobei die in der Implementierung vorgenommenen Bezeichungen stets durch `typewriter`-Ausdrücke gekennzeichnet sind:
Man berechne $\delta^T \delta = \sum_{i=k}^{n} a_{ik}^2$, im Programm entspricht dies der Variablen `dtd`. Ist `dtd=0`, so bricht der Algorithmus ab, denn die Vektoren $(a^1, \ldots, a^n)$ sind (numerisch) linear abhängig. Andernfalls wird wie folgt vorgegangen:

- `d`$:= \|\delta\| = \sqrt{\delta^T \delta}$
- Man wählt $\sigma \in \{-1, 1\}$ derart, daß

  $$u := \delta - \underbrace{\sigma \|\delta\|}_{=:\texttt{msd}} \cdot q^{k(k-1)} \neq 0.$$

  Dabei erweist es sich als fehlertheoretisch günstig

  $$\sigma := \begin{cases} -1 & \text{, falls } a_{kk} > 0 \\ 1 & \text{, falls } a_{kk} \leq 0 \end{cases}$$

  zu wählen. Diese Wahl entspricht der Festlegung von `msd` durch

  $$\textbf{if } \alpha_{kk} > 0 \, \textbf{then} \, \texttt{msd} := \|\delta\| \, \textbf{else} \, \texttt{msd} := -\|\delta\|.$$

- Es wird dann die k-te Spalte von R gemäß $r_{ik} := \alpha_{ik}$ für $i = 1, \ldots, k-1$ und $r_{kk} := -\texttt{msd}$ belegt.
- Für die Komponenten $u_i := u^T q^{i(k-1)}, i = k, \ldots, n$ gilt

  $$u = \delta + \texttt{msd} q^{k(k-1)} = (\alpha_{kk} + \texttt{msd}) q^{k(k-1)} + \sum_{i=k+1}^{n} \alpha_{ik} q^{i(k-1)}.$$

  Man kann also $u_k = \alpha_{kk} + \texttt{msd}$ und $u_i = \alpha_{ik}$ für $i = k+1, \ldots, n$ setzen und macht dies auch.

Man hat dann mit $q^{i(k)} := H_h q^{i(k-1)}$ für alle $i = 1, \ldots, n$ folgende Beziehungen:

$$q^{i(k)} = q^{i(k-1)}, \quad \text{falls } i \leq k-1, a^j = \sum_{i=1}^{j} r_{ij} q^{i(k)}, \quad j = 1, \ldots, k$$

und für $l = k+1, \ldots, n$ ist

$$a^l = \sum_{i=1}^{n} \underbrace{\alpha_{il}^{neu}}_{=a^{lT} \cdot q^{i(k)}} \cdot q^{i(k)} = \sum_{i=1}^{k-1} \underbrace{\alpha_{il}^{neu}}_{=\alpha_{il}} \cdot \underbrace{q^{i(k)}}_{q^{i(k-1)}} + \sum_{i=k}^{n} \alpha_{il}^{neu} \cdot q^{i(k)}.$$

Ferner ist

$$q^{i(k)} = \sum_{l=1}^{n} q_{l,i}^{neu} e^l = e^{lT} q^{i(k)}$$

und wegen $q^{i(k)} = q^{i(k-1)}$ für $i = 1, \ldots, k-1$ folgt $q_{l,i}^{neu} = q_{l,i}$ für $i = 1, \ldots, k-1$ und $j = 1, \ldots, n$. Es genügt also bei der Implementierung folgende Gleichungen zu lösen:

$$\alpha_{ij}^{neu} := a^{lT} \quad , \quad l = k+1, \ldots, n; i = k, \ldots, n \tag{3}$$

$$q_{l,i}^{neu} := e^{lT} q^{i(k)} \quad , \quad l = 1, \ldots, n; i = k, \ldots, n. \tag{4}$$

Um (3) und (4) zu lösen, sind also zu $x \in \mathbb{R}^n$ die Skalarprodukte $x^T q^{i(k)}$ für $i = k, \ldots, n$ aus den Komponenten $x_j := x^T q^{j(k-1)}, j = 1, \ldots, n$ zu berechnen. Wie bei der Erläuterung des Verfahrens bereits festgestellt, gilt mit $N := \frac{\|u\|^2}{2}$ für $i = k, \ldots, n$ folgende Gleichheit:

$$x^T q^{i(k)} = \underbrace{x^T q^{i(k-1)}}_{=x_i} - \underbrace{\frac{1}{N} \underbrace{(u^T q^{i(k-1)})}_{=u_i}}_{=:v_i} \cdot \sum_{j=k}^{n} \underbrace{x^T q^{j(k-1)}}_{=x_j} \cdot (u^T q^{j(k-1)})$$

Daraus resultiert folgende Möglichkeit zur Berechnung der $x^T q^{i(k)}, i = k, \ldots, n$, die auch der Implementierung entspricht:

- Es gilt

$$\begin{aligned}
\texttt{grN} &:= \frac{\|u\|^2}{2} \\
&= \left(\sum_{i=k}^{n} u_i^2 \right) /2 \\
&= \left((\alpha_{kk} + \texttt{msd})^2 + \sum_{i=k+1}^{n} \alpha_{ik}^2 \right) /2 \\
&= \left(\alpha_{kk}^2 + 2\texttt{msd}\alpha_{kk} + \underbrace{\texttt{msd}^2}_{=\texttt{dtd}} + \sum_{i=k+1}^{n} \alpha_{ik}^2 \right) /2 \\
&= \frac{2\delta^T \delta + 2\texttt{msd}\alpha_{kk}}{2} \\
&= \delta^T \delta + \alpha_{kk}\texttt{msd},
\end{aligned}$$

wobei $\delta^T\delta = \sum_{i=k}^{n} \alpha_{ik}^2$ benutzt wurde. Man beachte, daß das Ergebnis unabhängig von x und i ist!

- Man berechne $v_i := u_i/\mathtt{grN}$ für $i = k, \ldots, n$.
- Man berechne $p_x := \sum_{j=k}^{n} x_j u_j$.
- Man berechne $x^T q^{i(k)} = x_i - v_i p_x$ für $i = k, \ldots, n$.

Nach diesem Verfahren werden (3) und (4) in der Prozedur `H_QRxsc` bzw. in `H_QRst` berechnet, wobei eben in `H_QRxsc` von der speziellen PASCAL-XSC Arithmetik Gebrauch gemacht wird, was nach den oben angegebenen Schritten offensichtlich möglich ist.

3 Ein vergleichender Test der Prozeduren `H_QRxsc` und `H_QRst`

Beim Vergleich der beiden Prozeduren `H_QRxsc` und `H_QRst` gilt es auf zwei Dinge zu achten: zum einen bietet es sich an, bei der Eingabe der Matrix A dafür zu sorgen, daß die Spaltenvektoren zwar linear unabhängig sind, sich ihre Differenz aber nur wenig unterscheidet, daß also $\|a^i - a^j\|$ sehr klein ist. In diesen Fällen zeigen Orthogonalisierungsverfahren nämlich bekanntermaßen Schwächen. Zum anderen ist es natürlich zweckmäßig, dafür zu sorgen, daß die angegebenen Koeffizienten a_{ij} der Matrix darstellbare Maschinenzahlen sind, da sonst bereits beim Einlesen Rundungsfehler auftreten.
Während der Testphase gibt das Programm außer den beiden berechneten QR-Zerlegungen folgende Ergebnisse in eine Datei aus:

1. Es wird geprüft, ob die Spaltenvektoren der Matrizen Q_{xsc} und Q_{st} wirklich ein Orthonormalsystem bilden. Dazu werden die Skalarprodukte $q^{iT} \cdot q^j$ für $i = 1, \ldots, n, j = i, \ldots, n$ berechnet.
2. Es wird die Näherung $A' = QR$ berechnet und geprüft, wie diese von der ursprünglichen Matrix A abweicht, in dem die betragsmäßig größte Komponente der Differenzmatrix $A - A'$ ausgegeben wird.

Die in die Datei ausgebenen Zerlegungen und Testergebnisse sind im Anhang per Ausdruck aufgeführt.

Zunächst werden die unten angegebenen Matrizen $A \in M_5(\mathbb{R})$ und $B \in M_9(\mathbb{R})$ getestet. Diese sind so gewählt, daß die Spaltenvektoren „nahe“ $(1, \ldots, 1)$ liegen und jeweils $\|a^j\| \approx \sqrt{5}, \|b^j\| \approx \sqrt{9}$ gilt.

Test A

Es sei

$$A := \begin{pmatrix} 1+\tfrac{1}{16} & 1-\tfrac{1}{16} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} \\ 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{16} & 1-\tfrac{1}{16} \\ 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} \\ 1-\tfrac{1}{128} & 1+\tfrac{1}{16} & 1-\tfrac{1}{16} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} \\ 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} \end{pmatrix}.$$

Test B

$$B := \begin{pmatrix} 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} \\ 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} \\ 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} \\ 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} \\ 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} \\ 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} & 1-\tfrac{1}{64} \\ 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} & 1+\tfrac{1}{64} \\ 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1+\tfrac{1}{32} & 1-\tfrac{1}{32} \\ 1+\tfrac{1}{64} & 1-\tfrac{1}{64} & 1+\tfrac{1}{128} & 1-\tfrac{1}{128} & 1+\tfrac{1}{256} & 1-\tfrac{1}{256} & 1+\tfrac{1}{512} & 1-\tfrac{1}{512} & 1-\tfrac{1}{1024} \end{pmatrix}.$$

Als Ergebnis für die Matrix A läßt sich festhalten, daß die Zerlegung $Q_{xsc}R_{xsc}$ zwar eine bessere Näherung darstellt, da die beschriebene Differenz fast halb so groß ist wie bei $Q_{st}R_{st}$, doch berücksichtigt man, daß bei der Orthogonalität der Vektoren aus Q_{xsc} gegenüber Q_{st} keine sensationellen Fortschritte erzielt worden sind. so rechtfertigt das Ergebnis noch nicht den höheren Aufwand. Ebenfalls enttäuschend ist das Ergebnis bei der höher dimensionierten Matrix B. Die Verbesserung im Differenzwert von $Q_{xsc}R_{xsc}$ gegenüber $Q_{st}R_{st}$ ist vernachlässigbar und im Punkte Orthogonalität gilt sogar manchmal $\|q_{st}^{iT} q_{st}^{j}\| \leq \|q_{xsc}^{iT} q_{xsc}^{j}\|, i \neq j$.

Es liegt nahe, zu vermuten, daß die Komponenten der angegebenen Matrizen betragsmäßig zu groß sind, so daß der Vorteil des exakten Rechnens der PASCAL-XSC Arithmetik nicht zum Tragen kommt. Die Ergebnisse der Testmatrix C unterstützen diese Annahme:

Test C

Für $C' = (c'_{ij}) \in M_9(\mathbb{R})$ mit $c'_{ij} = 100000$ für alle i, j und $C'' \in M_9(\mathbb{R})$ mit

$$C'' = \begin{pmatrix} +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} \\ -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} \\ +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} \\ -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} \\ +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} \\ -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} & -\frac{1}{64} \\ +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} & +\frac{1}{64} \\ -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & +\frac{1}{32} & -\frac{1}{32} \\ +\frac{1}{64} & -\frac{1}{64} & +\frac{1}{128} & -\frac{1}{128} & +\frac{1}{256} & -\frac{1}{256} & +\frac{1}{512} & -\frac{1}{512} & -\frac{1}{1024} \end{pmatrix}.$$

sei

$$C := C' + C''.$$

Wie die beigelegten Ergebnisse belegen, spielt der Vorteil von Pascal-XSC hier keine Rolle. Dies liegt daran, daß der Vorkommaanteil der Komponenten jeweils zo groß ist, daß es bei beiden Rundungsarten zu Informationsverlust kommt. Daher werden in den nächsten beiden Beispielen Matrizen gewählt, deren Einträge nicht gerade winzig, aber doch recht klein sind und keine ganzzahligen Anteile besitzen.

Test D

Es sei $D \in M_9(\mathbb{R})$ mit

$$D := \begin{pmatrix} 2^{-1} & 2^{-2} & \cdots & 2^{-9} \\ 2^{-2} & 2^{-3} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 2^{-9} & \vdots & \vdots & 2^{-17} \end{pmatrix}$$

Test E

Um wenigstens ansatzweise den Überblick zu behalten, bezeichnen wir nun abkürzend

$$x := \frac{1}{4096} = 2^{-12} \text{ und } y := \frac{1}{2048} = 2^{-11}$$

und verteilen diese auf einer Matrix $E \in M_{11}(\mathbb{R})$ folgendermaßen:

$$E := \begin{pmatrix} y & x & y & x & y & x & y & x & y & x & y \\ x & y & x & y & x & y & x & y & x & y & x \\ y & y & x & y & x & y & x & y & x & y & x \\ x & x & y & x & y & x & y & x & y & x & y \\ y & y & y & x & y & x & y & x & y & x & y \\ x & x & x & y & x & y & x & y & x & y & x \\ y & y & y & y & x & y & x & y & x & y & x \\ x & x & x & x & y & x & y & x & y & x & y \\ y & y & y & y & y & x & y & x & y & x & y \\ x & x & x & x & x & y & x & y & x & y & x \\ y & y & y & y & y & y & x & y & x & y & x \end{pmatrix}$$

Während in beiden Fällen der Standard-PASCAL Algorithmus keine QR-Zerlegung mehr berechnet, zerlegt der PASCAL-XSC Algorithmus die jeweiligen Matrizen und die Abweichungen liegen ähnlich wie bei den vorherigen Beispielen bei einer Größenordnung von 10^{-16} bzw. $2 \cdot 10^{-19}$.

Als Fazit bleibt festzuhalten, daß die Implementierung des Householder-Verfahrens mit PASCAL-XSC Arithmetik dann wirkliche Vorteile bringt, wenn die Komponenten der Ausgangsmatrix sehr klein sind. Dann nämlich werden auch Zerlegungen berechnet, wenn der Standard-Algorithmus bereits signalisiert, daß die Ausgangsmatrix singulär sei.

4 Das Programm

```
program househ(input, output, ein, aus) ;

{ * Programmierpraktikum Hauptstudium    * }
{ *                                      * }
{ * Michael Conrad,                      * }
{ * WS 97 / 98                           * }
{ * Bergische Universitaet GH Wuppertal  * }

use mv_ari;

var n          : integer ;
    ein, aus   : text ;

{*******************************************************}

procedure H_QRxsc(A : rmatrix;
                  var Qxsc, Rxsc : rmatrix;
                  var errxsc : boolean);

var u, v, pQ, pA       : rvector[1..ub(A,1)];
    dtd, d, msd, grN   : real;
    n, k, l, j, i      : integer;

begin

  {* Initialisierungen *}
  n := ub(A,1) ;
  errxsc := false ;
  Qxsc := id(n) ; {Qxsc=(nxn) Einheitsmatrix}
  k := 1;

  while k<=n do begin

    dtd := #*(for i:=k to n sum(A[i,k]*A[i,k]) );
    if dtd=0.0 then begin
        errxsc := true; {Matrix (num.) singulaer, Abbruch}
        k := n+1
      end
    else begin {dtd>0.0}

      d := sqrt(dtd);

      {* Ausloeschung verhindernde Festlegung des Vorzeichens *}
      if A[k,k] > 0.0 then
        msd := d
```

```
        else msd := -d;

        {* im k-ten Schritt festzulegende Elemente von Rxsc *}
        for i:=1 to k-1 do
          Rxsc[i,k] := A[i,k];
        Rxsc[k,k] := -msd ;

        {* Komponenten u[k],..,u[n] von u bez. des aktuellen ONS *}
        u[k] := #*(A[k,k]+msd) ;
        for i:=k+1 to n do
          u[i] := A[i,k];

        {* Berechnung von grN und v[i]:=u[i]/grN, i=k,..,n *}
        grN := #*(dtd+msd*A[k,k]);
        for i:=k to n do v[i] := u[i]/grN;

        {*** Berechnung der neuen Koeffizienten von A ***}
        {* 1.) Berechnung von pA[k+1],..,pA[n] *}
        for l:=k+1 to n do
          pA[l] := #*(for j:=k to n sum(A[j,l]*u[j]) );
        {* 2.) Berechnung von A[i,l], l=k+1,..,n, i=k,..,n *}
        for l:= k+1 to n do
          for i:=k to n do
            A[i,l] := #*(A[i,l]-v[i]*pA[l]) ;

        {*** Berechnung der neuen Komponenten von Qxsc ***}
        {* 1.) Berechnung von pQ[1],..,pQ[n] *}
        for l:=1 to n do
          pQ[l] := #*(for j:=k to n sum(Qxsc[l,j]*u[j]) );
        {* 2.) Berechnung von Qxsc[l,i], l=1,..,n, i=k,..,n *}
        for l:=1 to n do
          for i:=k to n do
            Qxsc[l,i] := #*(Qxsc[l,i]-v[i]*pQ[l]) ;

        k := k+1 ;

    end; {else}
  end; {while}
end; {H_QRxsc}



{******************************************************}

procedure H_QRst(A : rmatrix;
                 var Qst, Rst : rmatrix;
                 var errst : boolean);
```

```
var u, v, pQ, pA       : rvector[1..ub(A,1)];
    dtd, d, msd, grN   : real;
    n, k, l, j, i      : integer;

begin

  {* Initialisierungen *}
  n := ub(A,1) ;
  errst := false ;
  Qst := id(n) ; {Qst=(nxn) Einheitsmatrix}
  k := 1;

  while k<=n do begin

    dtd := 0.0;
    for i:=k to n do dtd := dtd + A[i,k]*A[i,k];
    if dtd=0.0 then begin
        errst := true; {Matrix (num.) singulaer, Abbruch}
        k := n+1
      end
    else begin {dtd>0.0}

      d := sqrt(dtd);

      {* Ausloeschung verhindernde Festlegung des Vorzeichens *}
      if A[k,k] > 0.0 then
        msd := d
      else msd := -d;

      {* im k-ten Schritt festzulegende Elemente von Rst *}
      for i:=1 to k-1 do
        Rst[i,k] := A[i,k];
      Rst[k,k] := -msd ;

      {* Komponenten u[k],.., u[n] von u bez. des aktuellen ONS *}
      u[k] := A[k,k]+msd ;
      for i:=k+1 to n do
        u[i] := A[i,k];

      {* Berechnung von grN und v[i]:=u[i]/grN, i=k,.., n *}
      grN := dtd+msd*A[k,k];
      for i:=k to n do v[i] := u[i]/grN;

      {*** Berechnung der neuen Koeffizienten von A ***}
      {* 1.) Berechnung von pA[k+1],.., pA[n] *}
      for l:=k+1 to n do begin
```

```
        pA[l] := 0.0;
        for j:=k to n do pA[l] := pA[l] + A[j,l]*u[j];
      end;
      {* 2.) Berechnung von A[i,l], l=k+1,..,n, i=k,..,n *}
      for l:= k+1 to n do
        for i:=k to n do
          A[i,l] := A[i,l]-v[i]*pA[l] ;

      {*** Berechnung der neuen Komponenten von Qst ***}
      {* 1.) Berechnung von pQ[1],..,pQ[n] *}
      for l:=1 to n do begin
        pQ[l] := 0.0;
        for j:=k to n do pQ[l] := pQ[l] + Qst[l,j]*u[j];
      end;
      {* 2.) Berechnung von Qst[l,i], l=1,..,n, i=k,..,n *}
      for l:=1 to n do
        for i:=k to n do
          Qst[l,i] := Qst[l,i]-v[i]*pQ[l] ;

      k := k+1 ;

    end; {else}
  end; {while}
end; {H_QRst}


{*****************************************************}

procedure H_QRtest(A, Qxsc, Rxsc ,Qst , Rst : rmatrix ;
                   errxsc , errst : boolean;
                   var aus : text);

var QR : rmatrix[1..ub(A,1),1..ub(A,1)];
    n  : integer ;

{* Ausgabe der Ergebnisse in Datei, dann          *}
{* Vergleich der in den vorangegangen Prozeduren *}
{* erzielten Ergebnisse durch                     *}
{* 1.) Pruefung der Orthogonalitaet von Q         *}
{* 2.) Vergleich A mit A'=QR                      *}


  procedure ausgabe(Q, R : rmatrix; n, k : integer; var aus : text) ;

    var i, j : integer ;
  begin
    writeln(aus) ;
```

```
  if k=1 then writeln(aus,'Ergebnis der XSC-Zerlegung')
  else writeln(aus,'Ergebnis der Standard-Zerlegung');
  writeln(aus,'Berechnete Orthonormalbasis') ;
  writeln(aus) ;
  for j := 1 to n do
    begin
      for i := 1 to n do
        writeln(aus,'Q[',i,',',j,'] = ',Q[i,j]) ;
      writeln(aus);
    end;
  writeln(aus) ;
  writeln(aus,'Berechnete Dreicksmatrix');
  writeln(aus) ;
  for j := 1 to n do
    begin
      for i := 1 to j do
        writeln(aus,'R[',i,',',j,'] = ',R[i,j]) ;
      writeln(aus);
    end;
  writeln(aus) ;
end; {ausgabe}


procedure orthotest(Q : rmatrix; n : integer) ;

  var x, y     : rvector[1 .. n] ;
      skal     : real ;
      i, j, k  : integer ;
begin
  for i:=1 to n do begin
    {x = i-ter spaltenvektor von Q }
    for k:=1 to n do x[k] := Q[k,i] ;
    for j:=i to n do begin
      {y = j-ter Spaltenvektor von Q, j=i,..,n}
      for k:=1 to n do y[k] := Q[k,j] ;
      {Skalarprodukt der Vektoren}
      skal := x*y ;
      writeln(aus,'q[',i,']*q[',j,'] = ',skal);
    end;
    writeln(aus) ;
  end;
end; {orthotest}


procedure differenz(A, QR : rmatrix; n : integer) ;

  var D      : rmatrix[1..n,1..n] ;
```

```
      max   : real ;
      i, j  : integer ;
  begin
    D := A - QR ;
    max := 0.0 ;
    for i := 1 to n do
      for j := 1 to n do {Komponentenweiser Vergleich der Differenzmatrix}
        if abs(D[i,j]) > max then max := abs(D[i,j]) ;
    writeln(aus,'Die groesste Abweichung an einer Stelle der Differenzmatrix')
    writeln(aus,'betraegt : ',max);
  end; {differenz}


begin

  n := ub(A,1) ;
  rewrite(aus) ;

  if errxsc then
    begin
      writeln(aus,'Unter Verwendung der Prozedur mit XSC-Arithmetik') ;
      writeln(aus,'ist die eingegebene Matrix A (numerisch) singulaer.') ;
      writeln(aus,'Es wurde daher keine QR-Zerlegung berechnet.');
      writeln(aus) ;
    end
  else begin {Matrix (num.) reg. mit H_QRxsc}
    ausgabe(Qxsc,Rxsc,n,1,aus) ;
    writeln(aus) ;
    writeln(aus,'Ergebnis mit Pascal-XSC Arithmetik') ;
    writeln(aus) ;
    orthotest(Qxsc,n) ;
    QR := Qxsc * Rxsc ;
    differenz(A,QR,n) ;
  end; {else}

  if errst then
    begin
      writeln(aus,'Unter Verwendung der Standard-Arithmetik') ;
      writeln(aus,'ist die eingegebene Matrix A (numerisch) singulaer.') ;
      writeln(aus,'Es wurde daher keine QR-Zerlegung berechnet.');
      writeln(aus) ;
    end
  else begin {Matrix (num.) reg mit H_QRst}
    ausgabe(Qst,Rst,n,2,aus) ;
    writeln(aus) ;
    writeln(aus,'Ergebnis mit Standard-Pascal Arithmetik') ;
    writeln(aus) ;
```

```
    orthotest(Qst,n) ;
    QR := Qst * Rst ;
    differenz(A,QR,n) ;
  end; {else}

end; {H_QRtest}



{****************************************************}
{***                  Hauptprogramm              ***}
{****************************************************}

procedure main(var aus: text; n: integer);

var i,j          : integer ;
    A,
    Qst, Qxsc,
    Rst, Rxsc    : rmatrix[1..n,1..n] ;
    errst,
    errxsc       : boolean ;

begin

  {*** Zeilenweises Einlesen der Matrix A ***}
  for i := 1 to n do
    for j := 1 to n do read(ein,A[i,j]);

  {*** Berechung der QR-Zerlegung unter Benutzung der
       PASCAL-XSC Arithmetik ***}
  H_QRxsc(A, Qxsc, Rxsc, errxsc);

  {*** Berechung der QR-Zerlegung mit Standard-Pascal-Arithmetik ***}
  H_QRst(A, Qst, Rst, errst);

  {*** vergleichender Test der vorliegenden Ergebnisse ***}
  H_QRtest(A, Qxsc, Rxsc, Qst, Rst, errxsc, errst, aus)

end; {main}

{****************************************************}

begin

  reset(ein);
  read(ein,n); {Daten in ein:
                  n : Dimension der Matrix A
```

```
                    A[1,1], .. , A[n,n] : Matrixkomponenten
              }
  main(aus,n);

end.
```