ulm university universität
uulm

Fakultät für Mathematik und Wirtschafts-wissenschaften

Institut für Numerische Mathematik

Cache-optimierte QR-Zerlegung

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Florian Krötz
florian.kroetz@uni-ulm.de

Gutachter:
Dr. Michael Lehn
Dr. Andreas Borchert

Betreuer:
Dr. Michael Lehn

2018

Fassung 4. Mai 2018

Satz: PDF-LaTeX 2ε

Inhaltsverzeichnis

1 Einleitung

Für was brauch ich die QR?

Warum muss die schnell sein?

Warum das ganze?

1.1 Cache

Wie funktioniert der Und warum Cache-Optimierung

1.2 Intel MKL

Kapitel über die Wichtigkeit der Intel MKL.

1.2.1 QR Anwendung oder so was

-LGS -Ausgleichsprobleme -QR-Verfahren

2 BLAS

Die Abkürzung BLAS steht für Basic Linear Algebra Subprograms.

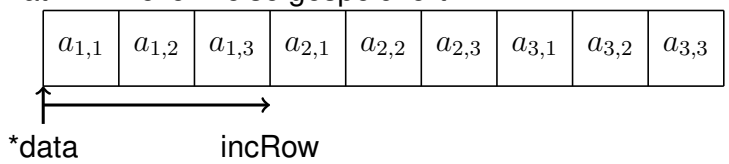
2.1 Datenstruktur für Matrizen

Um vollbesetzte Matrizen zu speichern benötigt man eine Speicherfläche für alle Einträge der Matrix, Informationen wie die Einräge im Speicher organisiert sind und Informationen über die Größe der Matrix.

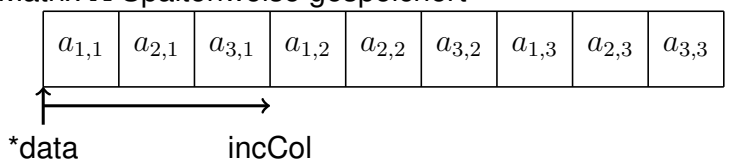
Es ist sinnvoll vollbesetzte Matrizen entweder Zeilen- oder Spaltenweise abzuspeichern. Das bedeutet entweder die Zeilen oder die Spalten der Matrix liegen hintereinander im Speicher.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Matrix A Zeilenweise gespeichert



Matrix A Spaltenweise gespeichert



Eine Datenstruktur benötigt ein Zeiger auf eine Speicherfläche, Informationen ob die Matrix Zeilen- oder Spaltenweise gespeichert ist und die Dimension der Matrix.

So eine Datenstruktur könnte in C so aussehen.

```
struct Matrix {
  double * data;
  std::ptrdiff_t incRow, incCol;
  std::size_t numRows, numCols;
}
```

Für Intel MKL Routinen müssen die Matrizen zeilenweise gespeichert sein.

2.2 Einige BLAS-Routinen

Im Folgenden werden einige BLAS-Routinen beschrieben, die bei der QR-Zerlegung benutzt werden. BLAS-Routinen werden bei MKL und LAPACK nach folgendem Schema benannt. Der erste Buchstabe im Namen gibt an für welchen Datentype die Funktion implementiert wurde. Der Rest zeigt an was die Funktion tut.
Beispiel: „dgemm“, das d zeigt an die Funktion ist für Doubles und „gemm“ steht für das Matrix-Matrix Produkt.

2.2.1 Matrix-Matrix Produkt (gemm)

Die „gemm“ Funktion berechnet das Matrix-Matrix Produkt. Der Funktion werden die Matrizen A, B und C und die Skalare α und β übergeben. Außerdem werden 2 Flags übergeben ob die Matrizen A und B transponiert werden sollen.
Die Funktion berechnet

$$C \leftarrow \beta C + \alpha AB \tag{2.1}$$

Falls $\beta = 0$ wird die Matrix C zuerst mit Nullen initialisiert. Falls C Einträge hat die NaN (Not a Number) sind, werden diese somit mit 0 überschrieben.

2.2.2 Matrix-Vector Produkt (gemv)

Die Funktion „gemv“ berechnet das Matrix-Vector Produkt. Der Funktion wird die Matrix A die Vektoren x und y und die Skalare α und β übergeben. Außerdem wird

ein Flag übergeben das anzeigt ob die Matrixa Transponiert werden soll.
Die Funktion berechnet

$$y \leftarrow \beta y + \alpha A x \tag{2.2}$$

Falls $\beta = 0$ wird der Vektor y zuerst mit Nullen initialisiert. Falls y Einträge hat die NaN (Not a Number) sind, werden diese somit mit 0 überschrieben.

2.2.3 Rank1 update (ger)

Die Funktion „ger“ berechnet ein dyadische Produkt aus den Vektoren x und y, skaliert die daraus resultierende Matrix mit α und addiert das Ergebnis auf A.// Der Funktion wird die Matrix A die Vektoren x und y und das Skalar α übergeben.
Die Funktion berechnet

$$A \leftarrow A + \alpha x y^T \tag{2.3}$$

2.2.4 Matrix-Matrix Produkt (trmm)

Die Funktion „trmm“ berechnet das Matrix-Matrix Produkt einer Dreiecksmatrix mit einer voll besetzten Matrix. Der Funktion wird die Dreiecksmatrix A, die Matrix B und das Skalar α übergeben. Außerdem werden Flags mit übergeben die anzeigen ob A eine obere oder unter Dreiecksmatrix ist, ob A eine strikte oder unipotente Dreiecksmatrix ist, ob A von links oder rechts auf B multipliziert werden soll und ob A transponiert werden soll.
Die Funktion berechnet

$$B \leftarrow \alpha \cdot op(A) \cdot B \qquad \text{or} \qquad B \leftarrow \alpha \cdot B \cdot op(A) \tag{2.4}$$

2.2.5 Matrix-Vector Produkt (trmv)

Die Funktion berechnet das Matrix-Vector Produkt für Dreiecksmatrizen. Die Funktion berechnet

$$x \leftarrow \alpha A x \tag{2.5}$$

3 QR factorisation

3.1 QR-Zerlegung

3.1.1 Definition

Eine Matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$ besitzt eine eindeutige QR-Zerlegung.

$$A = QR \tag{3.1}$$

mit einer orthogonalen Matrix $Q \in \mathbb{R}^{m \times m}$ und einer oberen Dreiecksmatrix $R \in \mathbb{R}^{n \times n}$ [2]

Eine QR Zerlegung kann mit einer Householder-Transformation berechnet werden.

Beispiel

Lösung eines Minimierungsproblem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2 \tag{3.2}$$

mit Matrix $A \in \mathbb{R}^{m \times n}$ mit $rang(A) = n < m$ für die eine QR Zerlegung existiert. R besitzt die Gestalt

$$R = \left(\begin{array}{ccc} * & * & * \\ & * & * \\ & & * \\ \hline & 0 & \end{array} \right) = \left(\begin{array}{c} \hat{R} \\ \hline 0 \end{array} \right)$$

$\hat{R}$ stellt eine obere Dreiecksmatrix dar. Damit kann man das Minimierungs Problem

wie folgt modifizieren mit $A = QR$

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2 = \min_{x \in \mathbb{R}^n} \|Q^T(Ax - b)\|^2 = \min_{x \in \mathbb{R}^n} \|Rx - Q^T b\|^2 \tag{3.3}$$

Also löst

$$Rx = Q^T b \tag{3.4}$$

das Minimierungsproblem (3.2). Da R eine Dreiecksmatrix ist, lässt sich (3.4) leicht mit Rückwärtseinsetzen lösen.

3.2 Householder-Transformation

Sei $v \in \mathbb{R}^n$ und $\tau \in \mathbb{R}$ dann wir die $n \times n$ Matrix

$$H = I - 2\frac{vv^T}{v^T v} \tag{3.5}$$

als Householder-Transformation und der Vektor v als Householder-Vektor bezeichnet. Eine Householder-Transformation $H = I - 2\frac{vv^T}{v^T v}$ ist orthogonal und symmetrisch. [2]
Die Householder-Transformation spiegelt den Vektor x auf die Achse x_1. Dazu multipliziert man H von links auf x.

$$Hx = \alpha e_1 \tag{3.6}$$

mit $\alpha \in \mathbb{R}$ und e_1 erster kanonischer Einheitsvektor. Der Householder-Vektor steht senkrecht auf der Achse an der x gespiegelt wird.
Die Abbildung 3.1 veranschaulicht die Spiegelung der Vektors x and der gestrichelt eingezeichneten Ebene auf die x_1 Achse. [Abbildung 3.1]

Eine Householder-Transformation kann die eine Matrix A wie folgt transformieren.

$$H_1 A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \quad , \quad H_2 H_1 A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

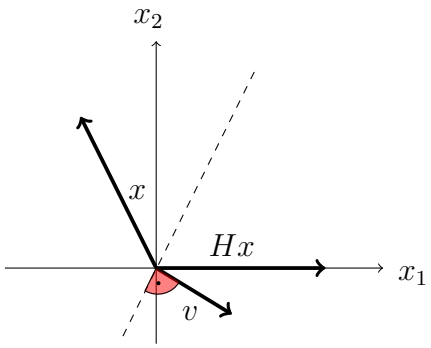


Abbildung 3.1: Beispiel Householder-Transformation mit $x = (-1, 2)^T$

So erhält man folgende Faktorisierung

$$R = H_{n-1}H_{n-2} \cdot ... \cdot H_1 A \Leftrightarrow A = (H_1 \cdot ... \cdot H_n)R \Rightarrow Q = H_1 \cdot ... \cdot H_n$$

Q ist also das Produkt aller Householder-Transformationen.

3.2.1 Householder Vector

Wie muss der Vekor v aussehen damit (3.6) gilt.
„Mit $Hx = x - 2\frac{vv^T}{v^Tv}x = x - \lambda v \stackrel{!}{=} \alpha e_1$ folgt $v \in \text{span}\{x - \alpha e_1\}$. “ [2]Warum?
Setze nun $v = x - \alpha e_1$ in $Hx = \alpha e_1$ ein

$$\begin{aligned} Hx =& x - \frac{2}{v^Tv}v(v^Tx) = x - 2\frac{v^Tx}{v^Tv}v \\ =& x - \frac{(x - \alpha e_1)^Tx}{\|x - \alpha e_1\|^2}(x - \alpha e_1) = \underbrace{\left(1 - \frac{2(x - \alpha e_1)^Tx}{\|x - \alpha e_1\|^2}\right)}_{\stackrel{!}{=}0} x + \alpha e_1 \frac{2(x - \alpha e_1)^Tx}{\|x - \alpha e_1\|^2} \stackrel{!}{=} \alpha e_1 \end{aligned}$$

Damit der Faktor vor dem x verschwindet muss gelten

$$1 = \frac{2(x - \alpha e_1)^Tx}{\|x - \alpha e_1\|^2} \Leftrightarrow (x - \alpha e_1)^T(x - \alpha e_1) = 2x^Tx - 2\alpha x_1 \Leftrightarrow \alpha = \pm\sqrt{x^Tx}$$

Wie ist das Vorzeichen von $\alpha = \pm\sqrt{x^Tx}$ zu wählen um $v = x - \alpha e_1$ zu berechnen?

Wählt man das Vorzeichen positive kann Auslöschung auftreten falls x annähernd ein positives Vielfaches von e_1 ist.

LAPACK [3] vermeidet die Auslöschung indem das Vorzeichen entgegengesetzt gewählt wird. Das bedeutet x wird immer auf die gegenüberliegende Seite gespiegelt.

Im Numerik 1 Skript [2] wird das Vorzeichen immer positiv gewählt $\alpha = |\sqrt{x^T x}| = \|x\|_2$. Eine mögliche Auslöschung im Fall $x_1 > 0$ wird hier durch die Umformung

$$v_1 = x_1 - \|x\|_2 = \frac{x_1^2 - \|x\|_2^2}{x_1 + \|x\|_2} = \frac{-(x_2^2 + ... + x_n^2)}{x_1 - \|x\|_2} \tag{3.7}$$

vermieden.

Der Vorteil bei der von LAPACK verwendeten Methode ist, das hier nur die Norm berechnet werden muss wohingegen bei dem anderen Algorithmus das Skalarprodukt $x^T x$ berechnet werden muss was bei Vektoren mit vielen Einträgen und großen werten schneller zu einem Überlauf führen kann. Es muss jedoch ein Algorithmus algorithmus gewählt werden der die Norm berechnet $\|x\| = \sqrt{x^T x}$ ohne das Skalarprodukt explizit auszurechnen.

Algorithmus 1 Housholder-Vector(LAPACK DLARFG)

$x \in \mathbb{R}^n$

LAPACK QR

Der von LAPACK benutzte Algorithmus [3]

$$\begin{aligned} x &= A(i+1:m, i) \\ x_1 &= A(i,i) \\ \alpha &= -1 * \text{sign}(x_1) \left| \sqrt{x_1^2 + \|x\|^2} \right| \\ \tau &= \frac{x_1 - \alpha}{\alpha} \\ v &= A(i+1:m, i) * \frac{1}{x_1 - \alpha} \end{aligned}$$

3.2.2 Householder-Transformation anwenden

Ein aufwändiges Matrix-Matrix Produkt kann bei der Anwendung der Housholder-Matrix $H = I - \tau v v^T$ auf die Matrix A umgangen werden, indem man geschickt

Klammert.

$$HA = (I - \tau v v^T)A = A - \tau v v^T A = A - \tau v * (v^T * A)$$

Statt eines Matrix-Matrix Produkts, muss man nun nur ein Matrix-Vektor Produkt und ein dyadisches Produkt berechnen. Das Matrix-Vektor Produkt und das dyadisches Produkt haben nur einen Aufwand von $O(n^2)$.

Das führt auf den Algorithmus 2.

Algorithmus 2 Ungeblockte Housholder-Transformation

$A \in \mathbb{R}^{m \times n}$
for i = 0 : n **do**
  [v, τ] = housevector($A(i : m, i)$)
  $w \leftarrow v^T * A$ (dgemv)
  $A \leftarrow \tau * v * w + A$ (dger)
  if i > m **then**
    $A(i + 1 : m, j) \leftarrow v(2 : m - i + 1)$
  end if
end for

Der Algorithmus 2 überschreibt die Matrix A mit R. Da R eine obere Dreiecksmatrix ist, werden unter der Diagonalen die Housholder-Vektoren gespeichert. A hat also die Form

$$A = \begin{pmatrix} R & R & R \\ v_1 & R & R \\ v_1 & v_2 & R \\ v_1 & v_2 & v_3 \end{pmatrix}$$

3.3 QR Blocked

Ein geblockter Algorithmus ist sinnvoll damit der Cache bei großen Matrizen optimal ausgenutzt wird.

Betrachte die Matrix $A \in \mathbb{R}^{m \times n}$ geblockt, mit einer geeigneten Blockgröße bs.

$$A = \left(\begin{array}{c|c} A_{0,0} & A_{0,\text{bs}} \\ \hline A_{\text{bs},0} & A_{\text{bs,bs}} \end{array}\right) \tag{3.8}$$

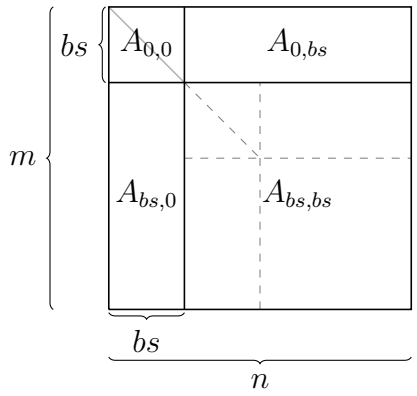


Abbildung 3.2: Partitionierung vom A

Die Abbildung 3.2 zeigt schematisch die Partitionierung von A.

Nun wird QR Zerlegung für den Block $\left(\frac{A_{0,0}}{A_{\text{bs},0}}\right)$ mit Algorithmus 2 berechnet.

$$\left(\frac{A_{0,0}}{A_{\text{bs},0}}\right) \leftarrow \left(\frac{Q_{0,0} \backslash R_{0,0}}{Q_{\text{bs},0}}\right) \tag{3.9}$$

Das Bedeutet im block $A_{0,0}$ steht nun auf und über der Diagonalen $R_{0,0}$, unterhalb der Diagonalen und im block $A_{\text{bs},0}$ stehen die Householder-Vektoren.

Damit

Berechne $H_0 ... H_{bs}$ aus $Q_{0,0}$ und $Q_{bs,0}$ mit $H = I - V * T * V^T$.
Wende H^T auf $A_{0,\text{bs}}$ und $A_{0,\text{bs}}$ an.

$$\left(\frac{A_{0,\text{bs}}}{A_{bs,\text{bs}}}\right) \leftarrow H^T \left(\frac{A_{0,\text{bs}}}{A_{bs,\text{bs}}}\right) \tag{3.10}$$

Betrachte nun den Block $A_{bs,\text{bs}}$ wie in (3.8), in der Abbildung 3.2 gestrichelt Dargestellt.

[Abbildung 3.2]

3.3.1 Calc Factor T larft

Die Funkton bekommt eine Dreiecksmatrix $V \in \mathbb{R}^{m \times k}$ einen Vektor $\tau \in \mathbb{R}^k$ und eine Matrix $T \in \mathbb{R}^{k \times k}$ übergeben. Die Funktion berechnet eine Dreiecksmatrix T so dass

$$H_1 H_2 ... H_k = I - VTV^T \qquad \text{mit} \qquad H_i = I - \tau_i v_i v_i^T$$

Warum und wie das Funktoniert wird hier beschreiben [1].

3.3.2 Apply H larfb

Die Funktion larfb bekommt eine Dreiecksmatrix $V \in \mathbb{R}^{m \times k}$, eine Dreiecksmatrix $T \in \mathbb{R}^{k \times k}$ und eine Matrix $C \in \mathbb{R}^{m \times n}$ übergeben. Die Funktion wendet eine Block Reflector Matrix $H = C - VTV^T$ von rechts auf die Matrix C an. Mit einem weiteren Übergabeparameter kann angeben werden ob die Block Reflector Matrix noch transponiert werden soll. Die Funktion berechnet also

$$C \leftarrow HC = C - VTV^T C \quad \text{oder} \quad C \leftarrow H^T C = C - VT^T V^T C$$

Die Abbildung 3.3 zeigt die Partitionierung der Matrix A für die Funktion larfb.
Falls $m > k$ werden die Matrizen V und C aufgeteilt in $V = \left(\frac{V_1}{V_2}\right)$ und $C = \left(\frac{C_1}{C_2}\right)$.
Dabei wird V genau so gewählt, dass $V_1 \in \mathbb{R}^{k \times k}$ der Dreiecksteil der Matrix und quadratisch ist und $V_2 \in \mathbb{R}^{m-k \times k}$ der Rest der Matrix. Die Matrix C wird in $C_1 \in \mathbb{R}^{k \times n}$ und $C_2 \in \mathbb{R}^{m-k \times n}$ aufgeteilt.
Die Aufteilung ist Notwendig da die BLAS-Funktion trmm (matrix-matrix product where one input matrix is triangular) nur für Quadratische Dreiecksmatrizen implementiert ist.

Im Fall $m = k$ ist die Aufteilung nicht Notwendig da V quadratisch ist.

$$(C_1^T * V_1 * T * V_1^T)^T$$
$$V_1 * T^T * V_1^T * C_1$$

Dies führt zu dem[Algorithmus 3]

Algorithmus 3 Block reflector anwenden

$W \leftarrow C_1^T$ (copy)
$W \leftarrow W * V_1$ (trmm)
if m > k **then**
  $W \leftarrow W + C_2^T * V_2$ (gemm)
end if
$W \leftarrow W * T^T$ or $W * T$ (trmm)
if m > k **then**
  $C_2 \leftarrow C_2 - V_2 * W^T$ (gemm)
end if
$W \leftarrow W * V_1^T$ (trmm)
$C_1 \leftarrow C_1 - W^T$

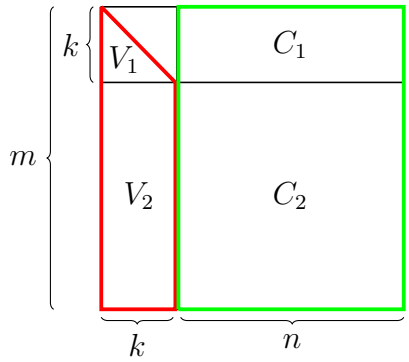


Abbildung 3.3: Partitionierung vom A für larfb

[Abbildung 3.3]

3.3.3 Iterativer Algorithmus

[Algorithmus 4]

3.3.4 Rekursiver Algorithmus

e

Algorithmus 4 Iterativer Algorithmus

for i = 0 : n **do**
  QR = A;
  if i + ib > n **then**
    Calc T: $H = I - VTV^T$
    Apply H: $A = H^T A$
  end if
end for

4 Implementierung und Benchmarks

Irgend was über die HPC Bibliothek

4.1 Fehler Schätzer

$$err = \frac{\|A - QR\|_i}{\|A\|_i \cdot \min(m, n) \cdot \varepsilon} \tag{4.1}$$

mit $\|\cdot\|_i$ passender Norm und ε die kleinste darstellbare Zahl.

Die QR-Zerlegung ist gut genug falls $err < 1$

4.2 MKL Wraper

4.3 Benchmarks

A Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

```
1 #include<stdio.h>
2 int main(int argc, char ** argv) {
3   printf("Hallo HPC \n");
4   return 0;
5 }
```

B Block Reflector

Versuch einer Herleitung

Vorwärts
n=2;

$$\begin{aligned}
H_1H_2x = (I - \tau_1 v_1 v_1^T)(I - \tau_2 v_2 v_2^T)x \\
=&(I - \tau_1 v_1 v_1^T - \tau_2 v_2 v_2^T + \tau_1 v_1 v_1^T \tau_2 v_2 v_2^T)x \\
=&x - \tau_1 v_1 v_1^T x - \tau_2 v_2 v_2^T x + \tau_1 \tau_2 v_1 (v_1^T v_2) v_2^T x \\
=&x - \tau_1 v_1 v_1^T x - \tau_2 v_2 v_2^T x + \tau_1 \tau_2 (v_1^T v_2) v_1 v_2^T x
\end{aligned}$$

n=3;

$$\begin{aligned}
H_1H_2H_3x = (I - \tau_1 v_1 v_1^T)(I - \tau_2 v_2 v_2^T)(I - \tau_3 v_3 v_3^T)x \\
=&(I - \tau_1 v_1 v_1^T - \tau_2 v_2 v_2^T + \tau_1 v_1 v_1^T \tau_2 v_2 v_2^T)(I - \tau_3 v_3 v_3^T)x \\
=&(I - \tau_1 v_1 v_1^T - \tau_2 v_2 v_2^T - \tau_3 v_3 v_3^T \\
&+ \tau_1 v_1 v_1^T \tau_2 v_2 v_2^T \\
&+ \tau_1 v_1 v_1^T \tau_3 v_3 v_3^T \\
&+ \tau_2 v_2 v_2^T \tau_3 v_3 v_3^T \\
&- \tau_1 v_1 v_1^T \tau_2 v_2 v_2^T \tau_3 v_3 v_3^T)x \\
=&x - \tau_1 v_1 v_1^T x - \tau_2 v_2 v_2^T x - \tau_3 v_3 v_3^T x \\
&+ \tau_1 \tau_2 (v_1^T v_2) v_1 v_2^T x \\
&+ \tau_1 \tau_3 (v_1^T v_3) v_1 v_3^T x \\
&+ \tau_2 \tau_3 (v_2^T v_3) v_2 v_3^T x \\
&- \tau_1 \tau_2 \tau_3 (v_1^T v_2 v_2^T v_3) v_1 v_3^T x
\end{aligned}$$

Rückwärts

n=2

$$
\begin{aligned}
H_{1,2}x = (I - VTV^T)x &= x - VTV^T x \\
&= x - (v_1, v_2) \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix} x \\
&= x - (v_1, v_2) \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} v_1^T x \\ v_2^T x \end{pmatrix} \\
&= x - (v_1, v_2) \begin{pmatrix} av_1^T x + bv_2^T x \\ cv_2^T x \end{pmatrix} \\
&= x - v_1(av_1^T x + bv_2^T x) - v_2(cv_2^T x) \\
&= x - av_1 v_1^T x - bv_1 v_2^T x - cv_2 v_2^T x
\end{aligned}
$$

n=3

$$
\begin{aligned}
H_{1,2,3}x = (I - VTV^T)x &= x - VTV^T x \\
&= x - (v_1, v_2, v_3) \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ v_3^T \end{pmatrix} x \\
&= x - (v_1, v_2, v_3) \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} v_1^T x \\ v_2^T x \\ v_3^T \end{pmatrix} \\
&= x - (v_1, v_2, v_3) \begin{pmatrix} av_1^T x + bv_2^T x + cv_3^T \\ dv_2^T x + ev_3^T \\ fv_3^T \end{pmatrix} \\
=& x - v_1(av_1^T x + bv_2^T x + cv_3^T x) \\
& - v_2(dv_2^T x + ev_3^T x) \\
& - v_3(fv_3^T) \\
=& x - av_1 v_1^T x - bv_1 v_2^T x - cv_1 v_3^T x \\
& - dv_2 v_2^T x - ev_2 v_3^T \\
& - fv_3 v_3^T
\end{aligned}
$$

Koeffizienten Vergleich

n=2

$$
\begin{aligned}
a &= \tau_1 \\
b &= -\tau_1 \tau_2 (v_1^T v_2) \\
c &= \tau_2 \\
T &= \begin{pmatrix} \tau_1 & -\tau_1 \tau_2 (v_1^T v_2) \\ 0 & \tau_2 \end{pmatrix}
\end{aligned}
$$

n=3

$$\begin{aligned}=&x - av_1v_1^Tx - bv_1v_2^Tx - cv_1v_3^Tx\\ &- dv_2v_2^Tx - ev_2v_3^T\\ &- fv_3v_3^T\end{aligned}$$

vs

$$\begin{aligned}=&x - \tau_1v_1v_1^Tx - \tau_2v_2v_2^Tx - \tau_3v_3v_3^Tx\\ &+ \tau_1\tau_2(v_1^Tv_2)v_1v_2^Tx\\ &+ \tau_1\tau_3(v_1^Tv_3)v_1v_3^Tx\\ &+ \tau_2\tau_3(v_2^Tv_3)v_2v_3^Tx\\ &- \tau_1\tau_2\tau_3(v_1^Tv_2v_2^Tv_3)v_1v_3^Tx\end{aligned}$$

ist

$$\begin{aligned}a =&\tau_1\\ b =&-\tau_1\tau_2(v_1^Tv_2)\\ c =&-\tau_1\tau_2\tau_3(v_1^Tv_2v_2^Tv_3) + \tau_1\tau_3(v_1^Tv_3)\\ d =&\tau_2\\ e =&-\tau_2\tau_3(v_2^Tv_3)\\ f =&\tau_3\end{aligned}$$

$$T = \begin{pmatrix} a & b & c\\ 0 & d & e\\ 0 & 0 & f\end{pmatrix} = \begin{pmatrix} \tau_1 & -\tau_1\tau_2(v_1^Tv_2) & -\tau_1\tau_2\tau_3(v_1^Tv_2v_2^Tv_3) + \tau_1\tau_3(v_1^Tv_3)\\ 0 & \tau_2 & -\tau_2\tau_3(v_2^Tv_3)\\ 0 & 0 & \tau_3\end{pmatrix}$$

Produkt orthogonaler Matrizen ist orthogonal. Sei $A^{-1} = A^T, B^{-1} = B^T$

$$(AB)^{-1} = B^{-1}A^{-1} = B^TA^T = (AB)^T$$

$\Rightarrow H = I - VTV^T$ ist orthogonal
ich bin doof Q ist ja auch orthogonal

Literaturverzeichnis

[1] JOFFRAIN, Thierry ; LOW, Tze M. ; QUINTANA-ORTÍ, Enrique S. ; GEIJN, Robert van d. ; ZEE, Field G. V.: Accumulating Householder Transformations, Revisited. In: *ACM Trans. Math. Softw.* 32 (2006), Juni, Nr. 2, 169–179. http://dx.doi.org/10.1145/1141885.1141886. – DOI 10.1145/1141885.1141886. – ISSN 0098–3500

[2] STEFAN A. FUNKEN, Karsten U.: *Einführung in die Numerische Lineare Algebra*. Ulm, Germany, 2016

[3] TENNESSEE, Univ. of California B. o. ; LTD.., NAG: *LAPACK unblocked QR*. http://www.netlib.org/lapack/explore-3.1.1-html/dgeqr2.f.html, 2006. – [Online; zugegriffen 31-01-2018]

Name: Florian Krötz Matrikelnummer: 884948

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Florian Krötz