ulm university universität uulm

Fakultät für Mathematik und Wirtschaftswissenschaften

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 26. Mai 2018

Satz: PDF-$\LaTeX\,2_\varepsilon$

Inhaltsverzeichnis

1 Einleitung

Wozu dient die QR-Zerlegung?

Warum muss die QR-Zerlegung schnell sein?

1.1 Cache

1.2 Intel MKL

Kapitel über die Wichtigkeit der Intel MKL.

1.2.1 Anwendung der QR-Zerlegung

-LGS lösen
-Ausgleichsprobleme lösen
-QR-Verfahren Eigenwerte b erechnen.

2 BLAS

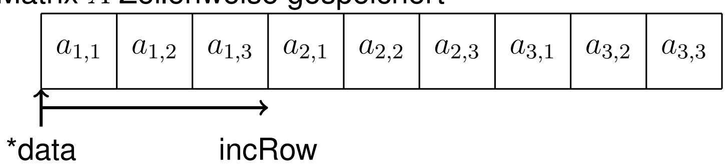
Die Abkürzung BLAS steht für Basic Linear Algebra Subprograms.

2.1 Datenstruktur für Matrizen

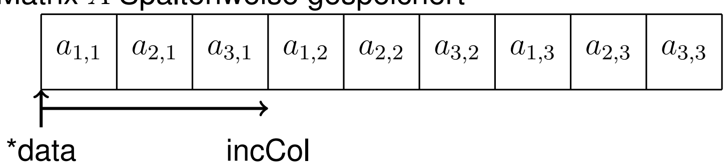
Vollbesetzte Matrizen werden bei BLAS entweder zeilen- oder spaltenweise abgespeichert. Das bedeutet, dass entweder die Zeilen- oder die Spalten der Matrix hintereinander im Speicher stehen.

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

Matrix A Zeilenweise gespeichert



Matrix A Spaltenweise gespeichert



Eine Datenstruktur benötigt folgende Elemente:

- einen Zeiger auf eine Speicherfläche
- Informationen ob die Matrix zeilen- oder spaltenweise gespeichert ist

- die Dimension der Matrix.

Eine derartige 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 meist nach folgendem Schema benannt. Der erste Buchstabe im Namen gibt an für welchen Datentype die Funktion implementiert wurde. Der Rest beschreibt die Funktion der Funktion.
Beispiel: „dgemm“, das d zeigt an die Funktion ist für Doubles und „gemm“ steht für „generel Matrix Matrix“, die Funktion berechnet also das Matrix-Matrix Produkt für Matrizen deren Einträge Doubles sind.

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.

Blas tecnicla forum netlib

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-Zerlegung

3.1 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.

3.1.1 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

$$R = \begin{pmatrix} * & * & * \\ & * & * \\ & & * \\ \hline & 0 & \end{pmatrix} = \begin{pmatrix} \hat{R} \\ \hline 0 \end{pmatrix}$$

$\hat{R}$ 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 $v \in \mathbb{R}^n$ ein Vektor dann wird 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 dem Skalar $\alpha \in \mathbb{R}$ und e_1 als ersten kanonischen Einheitsvektor. Der Householder-Vektor steht senkrecht auf der Ebene an welcher x gespiegelt wird.
Die Abbildung 3.1 veranschaulicht die Spiegelung des Vektors x an der gestrichelt eingezeichneten Ebene auf die Achse x_1. [Abbildung 3.1]

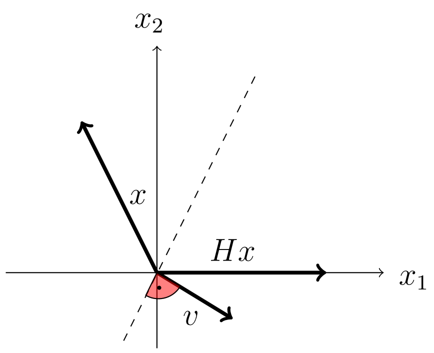


Abbildung 3.1: Beispiel Householder-Transformation mit $x = (-1, 2)^T$

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

$$H_1A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \quad , \quad H_2H_1A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

So erhält man folgende Faktorisierung

$$R = H_{n-1}H_{n-2} \cdot ... \cdot H_1A \Leftrightarrow A = (H_1 \cdot ... \cdot H_n)R \Rightarrow Q = H_1 \cdot ... \cdot H_n$$

Q ist also das Produkt aller Householder-Transformationen.

3.2.1 Householder Vector

Damit (3.6) gilt, muss der Vektor folgendermaßen berechnet werden.
Mit $Hx = x - 2\frac{vv^T}{v^Tv}x = x - \lambda v \overset{!}{=} \alpha e_1$ folgt $v \in \text{span}\{x - \alpha e_1\}$. [2]
Die Definition des Vektors $v = t(x - \alpha e_1)$ wird in $Hx = \alpha e_1$ eingesetzt

$$\begin{aligned} Hx =& x - \frac{2}{v^Tv}v(v^Tx) = x - 2\frac{v^Tx}{v^Tv}v \\ =& x - 2\frac{t(x-\alpha e_1)^Tx}{t(x-\alpha e_1)^Tt(x-\alpha e_1)}t(x-\alpha e_1) = x - 2\frac{(x-\alpha e_1)^Tx}{(x-\alpha e_1)^T(x-\alpha e_1)}(x-\alpha e_1) \\ =& x - \frac{(x-\alpha e_1)^Tx}{\|x-\alpha e_1\|_2^2}(x-\alpha e_1) = \underbrace{\left(1 - \frac{2(x-\alpha e_1)^Tx}{\|x-\alpha e_1\|_2^2}\right)}_{\overset{!}{=}0} x + \alpha e_1 \underbrace{\frac{2(x-\alpha e_1)^Tx}{\|x-\alpha e_1\|_2^2}}_{\overset{!}{=}1} \overset{!}{=} \alpha e_1 \end{aligned}$$

Damit das Letzte $=$ gilt muss

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

Das Vorzeichen von $\alpha = \pm\sqrt{x^Tx}$ kann man frei wählen, um $v = x - \alpha e_1$ zu berechnen.

Wählt man das Vorzeichen positiv 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 Skript von Numerik 1 [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 folgende Umformung vermieden.

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

Um den Vektor v später auf der frei werdenden Diagonalen von A speichern zu können wird er auf $v_1 = 1$ normiert. Dies geschieht mit

$$v = \frac{x - \alpha e_1}{x_1 - \alpha}$$

Mit der Normierung kann man den Faktor $\tau = \dfrac{2}{v^T v}$ berechnen.

$$\tau = \frac{2}{v^T v} = \frac{2(x_1 - \alpha)^2}{(x - \alpha e_1)^T (x - \alpha e_1)} = \frac{2(x_1 - \alpha)^2}{\|x\|_2^2 - 2\alpha x^T e_1 + \alpha^2} = \frac{2(x_1 - \alpha)^2}{2\alpha(\alpha - x_1)} = \frac{x_1 - \alpha}{\alpha}$$

Mit dem Faktor $\tau = \dfrac{2}{v^T v}$ kann man die Householder-Transformation schreiben als

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

Das macht man, weil die Berechnung des Skalarprodukts relativ aufwändig ist. Da man das Skalarprodukt zur Berechnung des Householder-Vektors benötigt, kann man damit direkt den Faktor τ berechnen.

Algorithmus 1 Housholder-Vector(LAPACK DLARFG)

Input: $x \in \mathbb{R}^n$
$\alpha = -1 * \text{sign}(x_1)\|x\|_2$
$\tau = \frac{x_1 - \alpha}{\alpha}$
$v = \frac{x - \alpha e_1}{x_1 - \alpha}$
Output: Householder-Vektor v , τ

3.2.2 Householder-Transformation anwenden

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

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

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

Das führt zum Algorithmus 2.

Algorithmus 2 Ungeblockte Housholder-Transformation

```
Input: A ∈ ℝ^{m×n}
for i = 0 : n do
    (v_i, τ_i) ← housevector(A(i : m, i))
    w ← v^T * A(i : m, i : n) (dgemv)
    A(i : m, i : n) ← τ * v * w + A (dger)
    if i > m then
        A(i + 1 : m, i) ← v(2 : m − i + 1)
    end if
end for
Output: A QR zerlegt, Vektor τ ∈ ℝ^n
```

Matizen sind 0 indizeiert notiert

Der Algorithmus 2 überschreibt die Matrix A mit R. Da R eine obere Dreiecksmatrix ist, werden unter der Diagonale die Housholder-Vektoren gespeichert. Da die Householder-Vektoren auf $v_1 = 1$ normiert wurden, muss das erste Element des Vektors nicht mit gespeichert werden. Die Householder-Vektoren können dadurch unterhalb der Diagonalen gespeichert werden. Die Matrix A hat also die Form

$$A = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ v_1 & r_{2,2} & r_{2,3} \\ v_1 & v_2 & r_{3,3} \\ v_1 & v_2 & v_3 \end{pmatrix}$$

3.3 Geblockte QR-Zerlegung

Ein geblockter Algorithmus ist sinnvoll, um bei großen Matrizen den Cache optimal zu nutzen.

Die Idee beim geblockten Algorithmus ist die Matrix in Blöcke aufzuteilen, die geblockte QR-Zerlegung für die Blöcke zu berechnen un die dabei entstandenen Householder-Transformationen auf den Rest der Matrix anzuwenden.

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

$$A = \left(\begin{array}{c|c} A_{0,0} & A_{0,\text{bs}} \\ \hline A_{\text{bs},0} & A_{\text{bs},\text{bs}} \end{array}\right) \tag{3.7}$$

Die Abbildung 3.2 zeigt schematisch die Partitionierung von A.

Die Blockgröße bs wird so gewählt das die Geschwindigkeit der ungeblockten QR-Zerlegung für den Block $\left(\begin{array}{c} A_{0,0} \\ \hline A_{\text{bs},0} \end{array}\right)$ optimal ist.

Für diesen Block wird nun die QR-Zerlegung mit dem ungeblockten Algorithmus 2 berechnet.

$$\left(\begin{array}{c} A_{0,0} \\ \hline A_{\text{bs},0} \end{array}\right) \leftarrow \left(\begin{array}{c} Q_{0,0} \backslash R_{0,0} \\ \hline Q_{\text{bs},0} \end{array}\right) \tag{3.8}$$

Im Block $A_{0,0}$ steht auf und über der Diagonalen $R_{0,0}$. Unterhalb der Diagonalen und im Block $A_{\text{bs},0}$ stehen die Householder-Vektoren.

Nun muss man die bei der ungeblocketn QR-Zerlegung verwendeten Housholder-Transformationen auf die Restliche Matrix $\left(\begin{array}{c} A_{\text{bs},\text{bs}} \\ \hline A_{\text{bs},\text{bs}} \end{array}\right)$ anwenden.

Man kann das Produkt mehrerer Householder-Transformationen schreiben als

[1]

Die Anwendung der Matrix $I - V * T * V^T$ auf $\left(\begin{array}{c} A_{\text{bs},\text{bs}} \\ \hline A_{\text{bs},\text{bs}} \end{array}\right)$ erfolgt in 2 Schritten.

Zuerst wird von der Funktion "larft"die Matrix T berechnet. Dann wird $I - V * T * V^T$ von der Funktion "larfbäuf $\left(\begin{array}{c} A_{\text{bs},\text{bs}} \\ \hline A_{\text{bs},\text{bs}} \end{array}\right)$ angewendet.

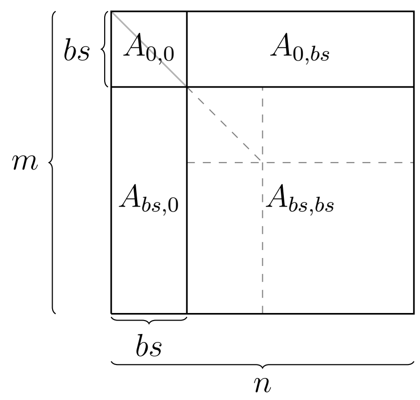


Abbildung 3.2: Partitionierung vom A

$$\left(\frac{A_{0,\text{bs}}}{A_{bs,\text{bs}}}\right) \leftarrow H^T \left(\frac{A_{0,\text{bs}}}{A_{bs,\text{bs}}}\right) \tag{3.9}$$

Der Block $A_{bs,\text{bs}}$ wird erneut aufgeteilt. Das ist in Abbildung 3.2 gestrichelt dargestellt.

Fahre solange fort bis $A_{bs,\text{bs}}$ gleich der Blockgröße ist.

[Abbildung 3.2]

3.3.1 Berechnung der Matrix 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 Anwenden von $I - VTV^T$ 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

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

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 $V = \begin{pmatrix} V_1 \\ \hline V_2 \end{pmatrix}$ und $C = \begin{pmatrix} C_1 \\ \hline C_2 \end{pmatrix}$.
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.

$$\begin{aligned} &(C_1^T * V_1 * T * V_1^T)^T \\ &V_1 * T^T * V_1^T * C_1 \end{aligned}$$

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]

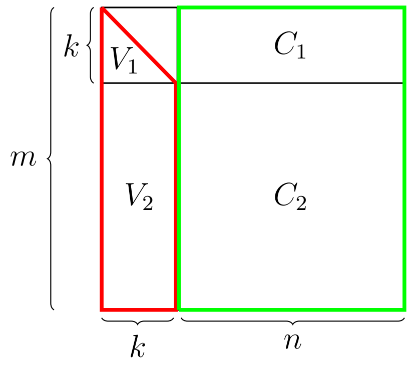


Abbildung 3.3: Partitionierung vom A für larfb

3.3.3 Iterativer Algorithmus

[Algorithmus 4]

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

Kurze Beschreibung der HPC-Bibliothek

Die verwendete Bibliothek wurde in der Vorlesung HPC1 entwickelt.

objektorientiert

eventuell Beispiel

4.1 Fehlerschätzer

Es wurde der Fehlerschätzer von ATLAS verwendet.

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

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

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

Als Norm wurde die Unendlichnorm $\|\cdot\|_\infty$ gewählt. Die Unendlich-Norm entspricht der Zeilensummennorm, die für eine Matrix $A \in \mathbb{R}^{m\times n}$ gegeben ist durch

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{n} |a_{ij}|$$

ϵ ist auf dem Test-System $2.220446 \cdot 10^{-16}$

4.2 MKL-Wrapper

4.3 Benchmarks

peak performance einzeichnen

A Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

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

B Block Reflector

Das Produkt aus Householder-Transformationen $H_1 \cdot ... \cdot H_n$ lässt sich schreiben als

$$H_1 \cdot ... \cdot H_n = I - VTV^T$$

mit einer unteren Dreiecksmatrix $V \in \mathbb{R}^{m \times n}$ die die Housholder-Vektoren enthält und eine oberen Dreiecksmatrix $T \in \mathbb{R}^{n \times n}$ [1]
Beweis:
n=2 Vorwärts

$$\begin{aligned} H_1 H_2 x = &(I - \tau_1 v_1 v_1^T)(I - \tau_2 v_2 v_2^T)x \\ =&(I - \tau_1 v_1 v_1^T - \tau_2 v_2 v_2^T + \tau_1 v_1 v_1^T \tau_2 v_2 v_2^T)x \\ =&x - \tau_1 v_1 v_1^T x - \tau_2 v_2 v_2^T x + \tau_1 \tau_2 v_1 (v_1^T v_2) v_2^T x \\ =&x - \tau_1 v_1 v_1^T x - \tau_2 v_2 v_2^T x + \tau_1 \tau_2 (v_1^T v_2) v_1 v_2^T x \end{aligned}$$

Rückwärts

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

Koeffizienten Vergleich

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

n=3
Vorwärts

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

Rückwärts

$$
\begin{aligned}
H_{1,2,3}x = (I - VTV^T)x = x - VTV^T x \\
= x - (v_1, v_2, v_3) \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ v_3^T \end{pmatrix} x \\
= x - (v_1, v_2, v_3) \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} \begin{pmatrix} v_1^T x \\ v_2^T x \\ v_3^T \end{pmatrix} \\
= x - (v_1, v_2, v_3) \begin{pmatrix} a v_1^T x + b v_2^T x + c v_3^T \\ d v_2^T x + e v_3^T \\ f v_3^T \end{pmatrix} \\
=&x - v_1(a v_1^T x + b v_2^T x + c v_3^T x) \\
&- v_2(d v_2^T x + e v_3^T x) \\
&- v_3(f v_3^T) \\
=&x - a v_1 v_1^T x - b v_1 v_2^T x - c v_1 v_3^T x \\
&- d v_2 v_2^T x - e v_2 v_3^T \\
&- f v_3 v_3^T
\end{aligned}
$$

Koeffizienten Vergleich

$$
\begin{aligned}
a =& \tau_1 \\
b =& -\tau_1\tau_2(v_1^T v_2) \\
c =& -\tau_1\tau_2\tau_3(v_1^T v_2 v_2^T v_3) + \tau_1\tau_3(v_1^T v_3) \\
d =& \tau_2 \\
e =& -\tau_2\tau_3(v_2^T v_3) \\
f =& \tau_3 \\
T =& \begin{pmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{pmatrix} = \begin{pmatrix} \tau_1 & -\tau_1\tau_2(v_1^T v_2) & -\tau_1\tau_2\tau_3(v_1^T v_2 v_2^T v_3) + \tau_1\tau_3(v_1^T v_3) \\ 0 & \tau_2 & -\tau_2\tau_3(v_2^T v_3) \\ 0 & 0 & \tau_3 \end{pmatrix}
\end{aligned}
$$

Mit Induktion kann man zeigen... siehe paper Im Paper wird gezeit wie man das verallgemeinern kann. Und weiter?

B.0.1 Orthogonal

Eine quadratische Matrix $Q \in \mathbb{R}^{n \times n}$ ist orthogonal, dann gilt

$$QQ^T = Q^TQ = I$$

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

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

Die Househlder-Transformation $H = I - 2\frac{vv^T}{v^Tv}$ ist symmetrisch und orthogonal das heißt $H^{-1} = H^T$
Da vv^T symmetrisch ist ($(vv^T)^T = vv^T$), folgt

$$H^T = \left(I - 2\frac{vv^T}{v^Tv}\right)^T = I - 2\frac{vv^T}{v^Tv} = H$$

Orthogonalität

$$HH^T = \left(I - 2\frac{vv^T}{v^Tv}\right)\left(I - 2\frac{vv^T}{v^Tv}\right) = I - 2\frac{vv^T}{v^Tv} - 2\frac{vv^T}{v^Tv} + \underbrace{4\frac{vv^Tvv^T}{(v^Tv)^2}}_{=4\frac{(v^Tv)vv^T}{(v^Tv)^2}=4\frac{vv^T}{v^Tv}} = I$$

$\Rightarrow H = I - VTV^T$ und Q sind 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