



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 29. April 2018

© 2018 Florian Krötz

Satz: PDF- $\text{\LaTeX}$  2 <sub>$\epsilon$</sub>

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Intel MKL . . . . .	1
1.1.1	QR Anwendung oder so was . . . . .	1
<b>2</b>	<b>QR factorisation</b>	<b>2</b>
2.1	QR-Zerlegung . . . . .	2
	Definition . . . . .	2
2.2	Householder-Transformation . . . . .	2
2.2.1	Householder Vector . . . . .	3
2.2.2	Apply vector . . . . .	3
2.3	LAPACK QR . . . . .	3
2.4	NUM1 Urban QR . . . . .	4
2.5	Unterschiede der Algorithmen . . . . .	4
2.6	QR Blocked . . . . .	5
2.6.1	Calc Factor T larft . . . . .	6
2.6.2	Apply H larfb . . . . .	7
2.6.3	Iterativer Algorithmus . . . . .	8
2.6.4	Rekursiver Algorithmus . . . . .	8
<b>3</b>	<b>Implementierung und Benchmarks</b>	<b>9</b>
3.1	MKL Wrapper . . . . .	9
3.2	Benchmarks . . . . .	9
<b>A</b>	<b>Quelltexte</b>	<b>10</b>
	<b>Literaturverzeichnis</b>	<b>11</b>

# **1 Einleitung**

Für was brauch ich die QR?

Warum muss die schnell sein?

Was soll die Arbeit?

## **1.1 Intel MKL**

Kapitel über die wichtigkeit der Intel MKL.

### **1.1.1 QR Anwendung oder so was**

-LGS -Ausgleichsprobleme -QR-Verfahren

## 2 QR factorisation

### 2.1 QR-Zerlegung

#### Definition

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

$$A = QR \quad (2.1)$$

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

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

### 2.2 Householder-Transformation

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

$$H = I - 2 \frac{vv^T}{v^T v} \quad (2.2)$$

als Householder-Transformation und der Vektor  $v$  als Householder-Vektor bezeichnet.

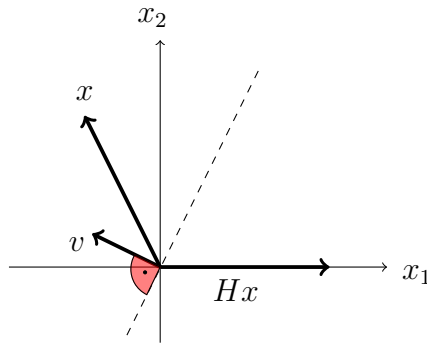


Abbildung 2.1: Spiegelung an der zu  $v$  orthogonalen Ebene

### 2.2.1 Householder Vector

### 2.2.2 Apply vector

$$H = I - \tau v v' \quad (2.3)$$

$$H A_2 = A_2 - \tau v v' A_2 \quad (2.4)$$

$$= A_2 - \tau v * (v' * A_2) \quad (2.5)$$

## 2.3 LAPACK QR

Der von LAPACK benutzte Algorithmus [3]

$$H = I - \tau \omega \omega^T \quad (2.6)$$

$$\tau = \frac{\alpha - \beta}{\beta} \quad (2.7)$$

$$\alpha = A(i, i) \quad (2.8)$$

$$\beta = \text{sign}(\alpha) \left| \sqrt{\alpha^2 + \|x\|^2} \right| \quad (2.9)$$

$$x = A(i+1 : m, i) \quad (2.10)$$

$$\omega = A(i+1 : m, i) * \frac{1}{\alpha - \beta} \quad (2.11)$$

Algorithmus

## 2.4 NUM1 Urban QR

Algorithmus aus Numerik 1

Mathe

$$H = I - 2 \frac{\omega \omega^T}{\omega^T \omega} \quad (2.12)$$

$$\omega_1 = \frac{x - \alpha e_1}{x_1 - \alpha} \quad (2.13)$$

$$\alpha^2 = \|x\|^2 \quad (2.14)$$

## 2.5 Unterschiede der Algorithmen

Die beiden Algorithmen unterscheiden sich lediglich durch eine Kleinigkeit in der Transformations Matrix.

$$H_{\text{Urban}} = I - \frac{2}{v^T v} v v^T \quad H_{\text{LAPACK}} = I - 2\tau v v^T$$

mit  $\tau = \frac{2}{v^T v}$ . Das  $\tau_i$  für jeden Householder-Vektor wird in einem Vektor  $\tau \in \mathbb{R}^n$  gespeichert.

$$H_i = I - \tau_i v_i v_i^T, \quad i \in 1, \dots, n$$

Durch die Verwendung des  $\tau$  muss das Skalarprodukt  $v^T v$  nur einmalig berechnet werden.

Hypothese:

Das Einsparungspotential ist nicht sehr groß. Das Skalarprodukt hat einen Aufwand von  $O(n)$ . Außerdem muss man den  $v$  nicht extra in den Cache laden, da er zur Berechnung des dyadischen Produkts  $v v^T$  schon im Cache liegt.

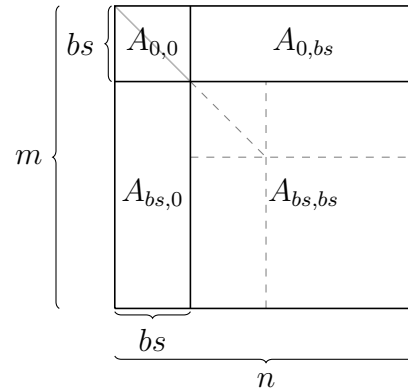


Abbildung 2.2: Partitionierung von A

## 2.6 QR Blocked

Geblockte Alorighmus

$$H = I - VTV' \quad (2.15)$$

$$H' = I - VT'V' \quad (2.16)$$

$$H'A_2 = A_2 - VT'V'A_2 \quad (2.17)$$

Betrachte A geblockt, mit einer geeigneten Blockgröße  $bs$ .

$$A = \left( \begin{array}{c|c} A_{0,0} & A_{0,bs} \\ \hline A_{bs,0} & A_{bs,bs} \end{array} \right) \quad (2.18)$$

Die Abbildung 2.2 zeigt schematisch die Partitionierung von A.

Berechne nun QR Zerlegung für den Block  $\left( \begin{array}{c} A_{0,0} \\ \hline A_{bs,0} \end{array} \right)$

$$\left( \begin{array}{c} A_{0,0} \\ \hline A_{bs,0} \end{array} \right) \leftarrow \left( \begin{array}{c} Q_{0,0} \setminus R_{0,0} \\ \hline Q_{bs,0} \end{array} \right) \quad (2.19)$$

Berechne  $H(0) \dots H(bs)$  aus  $Q_{0,0}$  und  $Q_{bs,0}$  mit  $H = I - V * T * V^T$ .



Wende  $H^T$  auf  $A_{0,bs}$  und  $A_{0,bs}$  an.

$$\begin{pmatrix} A_{0,bs} \\ A_{0,bs} \end{pmatrix} \leftarrow H^T \begin{pmatrix} A_{0,bs} \\ A_{0,bs} \end{pmatrix} \quad (2.20)$$

Fahre mit  $A_{0,bs}$  fort.

### 2.6.1 Calc Factor T larft

Die Funktion 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

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

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

Versuch einer Herleitung

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

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

### 2.6.2 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 angegeben 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 2.3 zeigt die Partitionierung der Matrix A für die Funktion larfb.

Falls  $m > k$  werden die Matrizen  $V$  und  $C$  aufgeteilt in  $V = \begin{pmatrix} V_1 \\ V_2 \end{pmatrix}$  und  $C = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}$ .

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:

```

W ← C1T (copy)
W ← W * V1 (trmm)
if m > k then
    W ← W + C2T * V2 (gemm)
end if
W ← W * TT or W * T (trmm)
if m > k then
    C2 ← C2 - V2 * WT (gemm)
end if
W ← W * V1T (trmm)
C1 ← C1 - WT

```

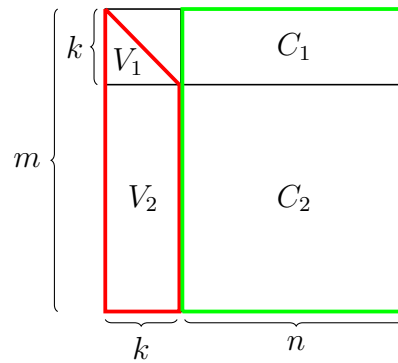


Abbildung 2.3: Partitionierung vom A für larfb

### 2.6.3 Iterativer Algorithmus

```

for i = 0 : n do
    QR = A;
    if i + ib > n then
        Calc T:  $H = I - VTV'$ 
        Apply H:  $A = H'A$ 
    end if
end for

```

### 2.6.4 Rekursiver Algorithmus

e

## **3 Implementierung und Benchmarks**

Irgend was über die HPC Bibliothek

### **3.1 MKL Wrapper**

### **3.2 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 }
```

# 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