



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 19. April 2018

© 2018 Florian Krötz

Satz: PDF- \LaTeX 2 _{ϵ}

Inhaltsverzeichnis

1	Einleitung	1
1.1	Intel MKL	1
1.1.1	QR Anwendung oder so was	1
2	QR factorisation	2
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	5
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	7
2.6.4	Rekursiver Algorithmus	8
3	Implementierung und Benchmarks	9
3.1	MKL Wrapper	9
3.2	Benchmarks	9
A	Quelltexte	10
	Literaturverzeichnis	11

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

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

2.2 Householder-Transformation

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

$$H = I - \tau vv^T \quad (2.2)$$

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

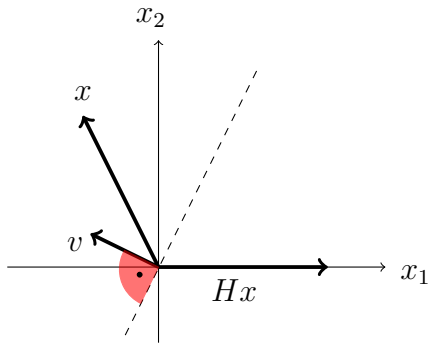


Abbildung 2.1: Spiegelung an der zu v orthogonalen Ebene

2.2.1 Householder Vector

2.2.2 Apply vector

$$H = I - \frac{vv'}{\tau} \quad (2.3)$$

$$HA_2 = A_2 - \frac{vv'}{\tau} A_2 \quad (2.4)$$

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

2.3 LAPACK QR

Der von LAPACK benutzte Algorithmus [2]

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

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

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

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

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

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

Algorithmus

```

1 householderVektor(Vektor v, alpha, tau)
2   beta = sign(sqrt(alpha ^2 + norm(x)^2), alpha)
3   tau = (alpha - beta) / beta
4   scal(1/(alpha - beta), v)

```

```

1 tau=zeros(min(m,n))
2 for i = 0 : min(m,n)
3   householderVektor(A(i+1:m,i), A(i,i), tau(i))
4   if (i < n && tau != 0)
5     AII = A(i,i)
6     A(i,i)= 1
7     A = A - tau *w(w'*A) // MV und rank1
8     A(i,i) = AII

```

2.4 NUM1 Urban QR

Algorithmus aus Numerik 1

Mathe

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

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

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

Algorithmus

```

1 householderVektor(Vektor x, omega, beta)
2   n = length(x)
3   if n > 1
4     sigma = x(2:end)'*x(2:end);
5     if sigma==0
6       beta = 0;
7     else
8       mu = sqrt(x(1)^2+sigma);
9       if x(1)<=0

```

```

10     tmp = x(1) - mu;
11     else
12         tmp = -sigma / (x(1) + mu);
13     end
14     beta = 2*tmp^2/(sigma + tmp^2);
15     x(2:end) = x(2:end)/tmp;
16 end
17 v = [1;x(2:end)];
18 else
19     beta = 0;
20     v = 1;
21 end

```

```

1 for i = i:n
2     housevector(A(i:m, i), w, beta)
3     A(i:m,i:n) = (I(m-i+1) - beta * w * w')*A(i:m,i:n)
4     if i < m
5         A(i + 1 : m, i) = w(2:m-i+1)

```

2.5 Unterschiede der Algorithmen

LAPCK hat das Tau

Vor und Nachteile oder so was

2.6 QR Blocked

Geblockte Alorighmus

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

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

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

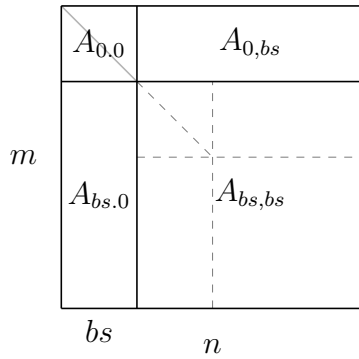


Abbildung 2.2: Partitionierung vom A

Betrachte A geblockt

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

Berechne QR Zerlegung für Blöcke $A_{0,0}$ und $A_{bs,0}$

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

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 $A_{bs,bs}$ an.

$$\left(\begin{array}{c} A_{0,bs} \\ A_{bs,bs} \end{array} \right) \leftarrow H^T \left(\begin{array}{c} A_{0,bs} \\ A_{bs,bs} \end{array} \right) \quad (2.20)$$

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

2.6.1 Calc Factor T larft

[1]

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

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

2.6.2 Apply H larfb

Die Funktion larfb berechnet.

$$H^T A = A - VT^T V^T A \quad (2.21)$$

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

```

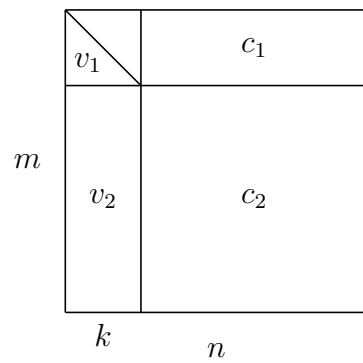


Abbildung 2.3: Partitionierung vom A

end if
end for

2.6.4 Rekursiver Algorithmus

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] 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