



ulm university universität
uulm

**Fakultät für
Mathematik und
Wirtschafts-
wissenschaften**

Institut für Numerische
Mathematik

High Performance implementation of QR factorization

Bachelorarbeit an der Universität Ulm

Vorgelegt von:

Florian Krötz
florian.kroetz@uni-ulm.de

Gutachter:

Dr. Michael Lehn
Dr. Un Leserlich

Betreuer:

Dr. Michael Lehn

2018

Fassung 17. April 2018

© 2018 Florian Krötz

Satz: PDF- \LaTeX 2 _{ε}

Inhaltsverzeichnis

1	Einleitung	1
2	Intel MKL	2
3	QR factorisation	3
3.1	QR-Zerlegung	3
	Definition	3
	QR Anwendung oder so was	3
3.2	Householder Spiegelung	3
3.2.1	Householder Vector	3
3.2.2	Apply vector	3
3.3	LAPACK QR	4
3.4	NUM1 Urban QR	4
3.5	Unterschiede der Algorithmen	6
3.6	QR Blocked	6
3.6.1	Calc Factor T larft	6
3.6.2	Apply H larfb	7
3.6.3	Iterativer Algorithmus	7
3.6.4	Rekursiver Algorithmus	7
4	Implementierung und Benchmarks	8
4.1	MKL Wrapper	8
4.2	Benchmarks	8
A	Quelltexte	9
	Literaturverzeichnis	10

1 Einleitung

Für was brauch ich die QR?

Warum muss die schnell sein?

Was soll die Arbeit?

2 Intel MKL

Kapitel über die Wichtigkeit der Intel MKL.

3 QR factorisation

3.1 QR-Zerlegung

Definition

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

$$A = QR \quad (3.1)$$

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

QR Anwendung oder so was

-LGS -Ausgleichsprobleme -QR-Verfahren

3.2 Householder Spiegelung

3.2.1 Householder Vector

3.2.2 Apply vector

$$H = I - \frac{vv'}{\tau} \quad (3.2)$$

$$HA_2 = A_2 - \frac{vv'}{\tau} A_2 \quad (3.3)$$

$$= A_2 - \frac{v}{\tau} * (v' * A_2) \quad (3.4)$$

3.3 LAPACK QR

Der von LAPACK benutzte Algorithmus [2]

$$H = I - \tau \omega \omega^T \quad (3.5)$$

$$\tau = \frac{\alpha - \beta}{\beta} \quad (3.6)$$

$$\alpha = A(i, i) \quad (3.7)$$

$$\beta = \text{sign}(\alpha) \left| \sqrt{\alpha^2 + \|x\|^2} \right| \quad (3.8)$$

$$x = A(i+1:m, i) \quad (3.9)$$

$$\omega = A(i+1:m, i) * \frac{1}{\alpha - \beta} \quad (3.10)$$

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
```

3.4 NUM1 Urban QR

Algorithmus aus Numerik 1

Mathe

$$H = I - 2 \frac{\omega \omega^T}{\omega^T \omega} \quad (3.11)$$

$$\omega_1 = \frac{x - \alpha e_1}{x_1 - \alpha} \quad (3.12)$$

$$\alpha^2 = \|x\|^2 \quad (3.13)$$

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 = 1: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)

```


3.5 Unterschiede der Algorithmen

LAPCK hat das Tau

Vor und Nachteile oder so was

3.6 QR Blocked

Geblockte Alorighmus

$$H = I - VTV' \quad (3.14)$$

$$H' = I - VT'V' \quad (3.15)$$

$$H'A_2 = A_2 - VT'V'A_2 \quad (3.16)$$

Betrachte A geblockt

$$A = \left(\begin{array}{c|c} A_{0,0} & A_{0,bs} \\ \hline A_{bs,0} & A_{bs,bs} \end{array} \right) \quad (3.17)$$

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

$$\left(\begin{array}{c} A_{0,0} \\ A_{bs,0} \end{array} \right) \leftarrow \left(\begin{array}{c} Q_{0,0} \backslash R_{0,0} \\ Q_{bs,0} \end{array} \right) \quad (3.18)$$

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.

$$\left(\begin{array}{c} A_{0,bs} \\ A_{bs,bs} \end{array} \right) \leftarrow H^T \left(\begin{array}{c} A_{0,bs} \\ A_{bs,bs} \end{array} \right) \quad (3.19)$$

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

3.6.1 Calc Factor T larft

[1]

3.6.2 Apply H larfb

3.6.3 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
```

3.6.4 Rekursiver Algorithmus

4 Implementierung und Benchmarks

Irgend was über die HPC Bibliothek

4.1 MKL Wrapper

4.2 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