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 19. April 2018

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

Inhaltsverzeichnis

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.

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.

$$A = QR \tag{2.1}$$

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

2.2 Householdertransformation

2.2.1 Householder Vector

2.2.2 Apply vector

$$\begin{aligned} H =& I - \frac{vv'}{\tau} && (2.2)\\ HA_2 =& A_2 - \frac{vv'}{\tau} A_2 && (2.3)\\ =& A_2 - \frac{v}{\tau} * (v' * A_2) && (2.4) \end{aligned}$$

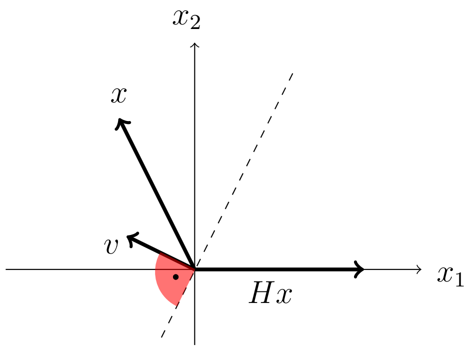

Abbildung 2.1: Householder Trans

2.3 LAPACK QR

Der von LAPACK benutzte Algorithmus [2]

$$H = I - \tau\omega\omega^T \tag{2.5}$$

$$\tau = \frac{\alpha - \beta}{\beta} \tag{2.6}$$

$$\alpha = A(i,i) \tag{2.7}$$

$$\beta = \mathsf{sign}(\alpha)\left|\sqrt{\alpha^2 + \|x\|^2}\right| \tag{2.8}$$

$$x = A(i+1:m,i) \tag{2.9}$$

$$\omega = A(i+1:m,i) * \frac{1}{\alpha - \beta} \tag{2.10}$$

Algorithmus

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

```
tau=zeros(min(m,n))
for i = 0 : min(m,n)
  householderVektor(A(i+1:m,i), A(i,i), tau(i))
  if (i < n && tau != 0)
    AII = A(i,i)
    A(i,i)= 1
    A = A - tau *w(w'*A) // MV und rank1
```

```
    A(i,i) = AII
```

2.4 NUM1 Urban QR

Algorithmus aus Numerik 1

Mathe

$$H = I - 2\frac{\omega\omega^T}{\omega^T\omega} \tag{2.11}$$

$$\omega_1 = \frac{x - \alpha e_1}{x_1 - \alpha} \tag{2.12}$$

$$\alpha^2 = \|x\|^2 \tag{2.13}$$

Algorithmus

```
householderVektor(Vektor x, omega, beta)
  n = length(x)
  if n> 1
    sigma = x(2:end)'*x(2:end);
    if sigma==0
      beta = 0;
    else
      mu = sqrt(x(1)^2+sigma);
      if x(1)<=0
        tmp = x(1) - mu;
      else
        tmp = -sigma / (x(1) + mu);
      end
      beta = 2*tmp^2/(sigma + tmp^2);
      x(2:end) = x(2:end)/tmp;
    end
    v = [1;x(2:end)];
  else
    beta = 0;
    v = 1;
  end
```

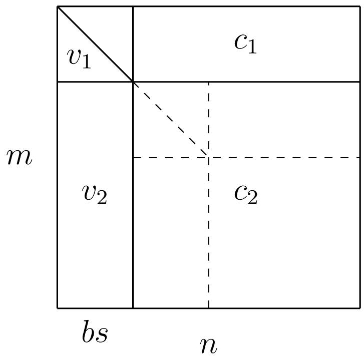

Abbildung 2.2: Partitionierung vom A

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

$$H = I - VTV' \tag{2.14}$$

$$H' = I - VT'V' \tag{2.15}$$

$$H'A_2 = A_2 - VT'V'A_2 \tag{2.16}$$

Betrachte A geblockt

$$A = \left(\begin{array}{c|c} A_{0,0} & A_{0,\text{bs}} \\ \hline A_{\text{bs},0} & A_{\text{bs},\text{bs}} \end{array}\right) \tag{2.17}$$

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

$$\left(\begin{array}{c} A_{0,0} \\ \hline A_{\text{bs},0} \end{array}\right) \leftarrow \left(\begin{array}{c} Q_{0,0} \backslash R_{0,0} \\ \hline Q_{\text{bs},0} \end{array}\right) \tag{2.18}$$

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.

$$\left(\begin{array}{c} A_{0,\text{bs}} \\ \hline A_{0,\text{bs}} \end{array}\right) \leftarrow H^T \left(\begin{array}{c} A_{0,\text{bs}} \\ \hline A_{0,\text{bs}} \end{array}\right) \tag{2.19}$$

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

2.6.1 Calc Factor T larft

[1]

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

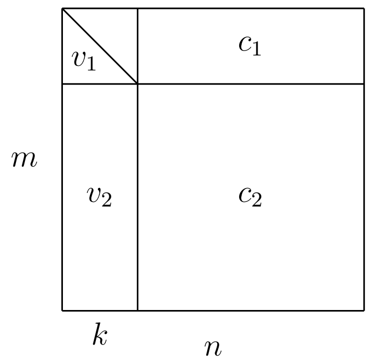

Abbildung 2.3: Partitionierung vom A

$$
\begin{aligned}
H_{1,2}x &= (I - VTV^T)x = x - VTV^T x \\
&= x - (v_1, v_2) \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix} x \\
&= x - (v_1, v_2) \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} v_1^T x \\ v_2^T x \end{pmatrix} \\
&= x - (v_1, v_2) \begin{pmatrix} av_1^T x + bv_2^T x \\ cv_2^T x \end{pmatrix} \\
&= x - v_1(av_1^T x + bv_2^T x) - v_2(cv_2^T x) \\
&= x - av_1v_1^T x - bv_1v_2^T x - cv_2v_2^T x
\end{aligned}
$$

2.6.2 Apply H larfb

Die Funktion larfb berechnet.

$$H^T A = A - VT^T V^T A \tag{2.20}$$

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

3 Implementierung und Benchmarks

Irgend was über die HPC Bibliothek

3.1 MKL Wraper

3.2 Benchmarks

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;
}
```

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