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 30. April 2018

Satz: PDF-LaTeX 2ε

Inhaltsverzeichnis

1 Einleitung

Für was brauch ich die QR?

Warum muss die schnell sein?

Was soll der Scheiß?

1.1 Cache

Wie funktioniert der Und warum Cache-Optimierung

1.2 Intel MKL

Kapitel über die Wichtigkeit der Intel MKL.

1.2.1 QR Anwendung oder so was

-LGS -Ausgleichsprobleme -QR-Verfahren

2 BLAS

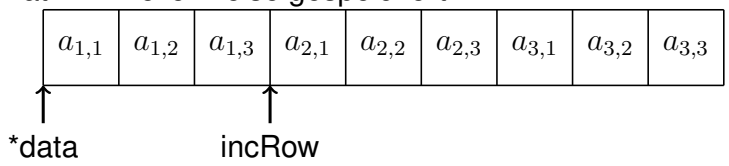
2.1 Datenstruktur für Matrizen

Um vollbesetzte Matrizen zu speichern benötigt man eine Speicherfläche für alle Einträge der Matrix, Informationen wie die Einräge im Speicher organisiert sind und Informationen über die Größe der Matrix.

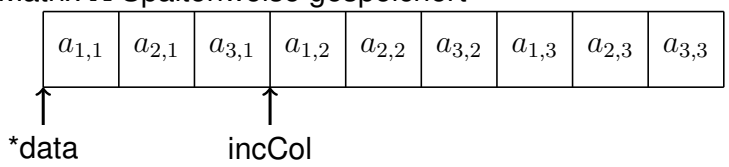
Es ist sinnvoll vollbesetzte Matrizen entweder Zeilen-der Matrix liegen hintereinander im Speicher oder Spaltenweise abzuspeichern. Das bedeutet die Zeilen oder Spalten der Matrix liegen hintereinander im Speicher.

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

Matrix A Zeilenweise gespeichert



Matrix A Spaltenweise gespeichert



Eine Datenstruktur benötigt ein Zeiger auf eine Speicherfläche, Informationen ob die Matrix Zeilen- oder Spaltenweise gespeichert ist und die Dimension der Matrix.

So eine Datenstruktur könnte in C so aussehen.

```
1 struct Matrix {
```

```
2   double * data;
3   std::ptrdiff_t incRow, incCol;
4   std::size_t numRows, numCols;
5 }
```

Für Intel MKL und LAPACK Routinen müssen die Matrizen zeilenweise gespeichert.

2.2 Einige Blasroutinen

2.2.1 Matrix-Matrix Produkt gemm

$$C \leftarrow \beta * C + \alpha * A * B \tag{2.1}$$

2.2.2 Matrix-Vector Produkt gemv

$$y \leftarrow \alpha * A * x + \beta * y \tag{2.2}$$

2.2.3 Rank1 update ger

$$A \leftarrow A + \alpha * x * y^T \tag{2.3}$$

2.2.4 Matrix-Matrix Produkt trmm

$$B \leftarrow \alpha * op(A) * B \qquad \text{or} \qquad B \leftarrow \alpha * B * op(A) \tag{2.4}$$

2.2.5 Matrix-Vector Produkt trmv

$$x \leftarrow \alpha * A * x \qquad \text{or} \qquad x \leftarrow \alpha * A^T * x \tag{2.5}$$

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.

$$A = QR \tag{3.1}$$

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.

Motivation

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

$$R = \left(\begin{array}{ccc} * & * & * \\ & * & * \\ & & * \\ \hline & 0 & \end{array} \right) = \left(\begin{array}{c} \hat{R} \\ \hline 0 \end{array} \right)$$

$\hat{R}$ 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 $v \in \mathbb{R}^n$ und $\tau \in \mathbb{R}$ dann wir 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 $\alpha \in \mathbb{R}$ und e_1 erster kanonischer Einheitsvektor. Der Householder-Vektor steht senkrecht auf der Achse an der x gespiegelt wird.
Die Abbildung 3.1 veranschaulicht die Spiegelung der Vektors x and der gestrichelt eingezeichneten Ebene auf die x_1 Achse.

3.2.1 Householder Vector

Wie muss der Vekor v aussehen damit (3.6) gilt.

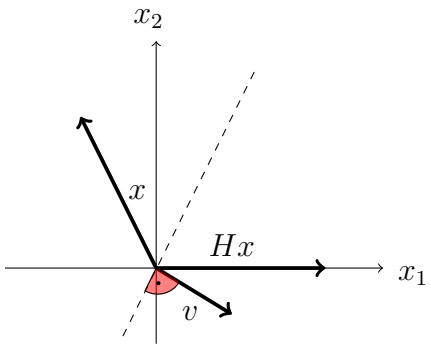


Abbildung 3.1: Beispiel Householder-Transformation mit $x = (-1, 2)^T$

Algorithmus 1 Housholder-Vector

$A \in \mathbb{R}^{m \times n}$
for i = 0 : n **do**
  [v, τ] = housevector($A(i : m, i)$)
  $w \leftarrow v^T * A$ (dgemv)
  $A \leftarrow \tau * v * w + A$ (dger)
  if i > m **then**
    $A(i + 1 : m, j) \leftarrow v(2 : m - i + 1)$
  end if
end for

3.2.2 Apply vector

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

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

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

Das führt auf den Algorithmus 2.

Der Algorithmus 2 überschreibt die Matrix A mit R. Da R eine obere Dreiecksmatrix ist, werden unter der Diagonalen die Housholder-Vektoren gespeichert. A hat also

Algorithmus 2 Ungeblockte Housholder-Transformation

$A \in \mathbb{R}^{m \times n}$
for i = 0 : n **do**
  [v, τ] = housevector($A(i : m, i)$)
  $w \leftarrow v^T * A$ (dgemv)
  $A \leftarrow \tau * v * w + A$ (dger)
  if i > m **then**
    $A(i + 1 : m, j) \leftarrow v(2 : m - i + 1)$
  end if
end for

die Form

$$A = \begin{pmatrix} R & R & R \\ v_1 & R & R \\ v_1 & v_2 & R \\ v_1 & v_2 & v_3 \end{pmatrix}$$

3.2.3 LAPACK QR

Der von LAPACK benutzte Algorithmus [3]

$$\tau = \frac{\alpha - \beta}{\beta} \tag{3.7}$$

$$\alpha = A(i, i) \tag{3.8}$$

$$\beta = \text{sign}(\alpha) \left| \sqrt{\alpha^2 + \|x\|^2} \right| \tag{3.9}$$

$$x = A(i + 1 : m, i) \tag{3.10}$$

$$v = A(i + 1 : m, i) * \frac{1}{\alpha - \beta} \tag{3.11}$$

3.2.4 NUM1 Urban QR

Algorithmus aus Numerik 1

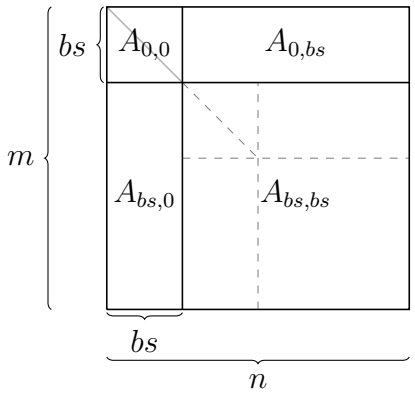


Abbildung 3.2: Partitionierung vom A

$$v_1 = \frac{x - \alpha e_1}{x_1 - \alpha} \tag{3.12}$$

$$\alpha^2 = \|x\|^2 \tag{3.13}$$

3.3 QR Blocked

Ein geblockter Algorithmus ist sinnvoll um bei großen Matrizen den Cache optimal auszunutzen. Geblockte Alorighmus

$$H = I - VTV^T$$
$$H^T = I - VT^TV^T$$
$$H^T A_{bs,bs} = A_{bs,bs} - VT^TV^T A_{bs,bs}$$

Betrachte A geblockt, mit einer geeigneten Blockgröße bs.

$$A = \left(\begin{array}{c|c} A_{0,0} & A_{0,\text{bs}} \\ \hline A_{\text{bs},0} & A_{\text{bs,bs}} \end{array}\right) \tag{3.14}$$

Die Abbildung 3.2 zeigt schematisch die Partitionierung von A.

Berechne nun QR Zerlegung für den Block $\left(\frac{A_{0,0}}{A_{\text{bs},0}}\right)$

$$\left(\frac{A_{0,0}}{A_{\text{bs},0}}\right) \leftarrow \left(\frac{Q_{0,0}\backslash R_{0,0}}{Q_{\text{bs},0}}\right) \tag{3.15}$$

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.

$$\left(\frac{A_{0,\text{bs}}}{A_{0,\text{bs}}}\right) \leftarrow H^T \left(\frac{A_{0,\text{bs}}}{A_{0,\text{bs}}}\right) \tag{3.16}$$

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

3.3.1 Calc Factor 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].

Versuch einer Herleitung

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

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

3.3.2 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 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 $V = \left(\frac{V_1}{V_2}\right)$ und $C = \left(\frac{C_1}{C_2}\right)$.
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.

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

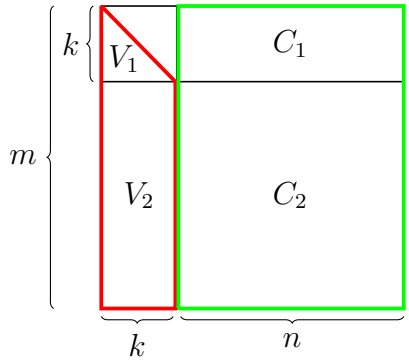


Abbildung 3.3: Partitionierung vom A für larfb

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)
$C_1 \leftarrow C_1 - W^T$

3.3.3 Iterativer Algorithmus

3.3.4 Rekursiver Algorithmus

e

Algorithmus 4 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
```

4 Implementierung und Benchmarks

Irgend was über die HPC Bibliothek

4.1 MKL Wraper

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] 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