Scientific Computing I

Least Squares Problems and the QR Decomposition

Martin Köhler

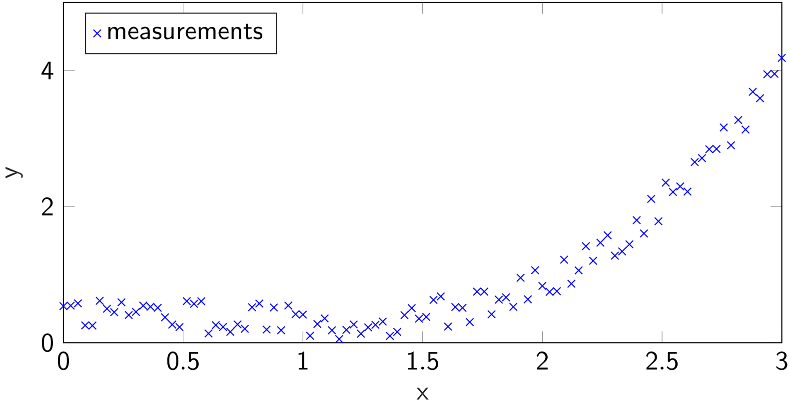
Computational Methods in Systems and Control Theory (CSC) Max Planck Institute for Dynamics of Complex Technical Systems

Winter Term 2024/2025

Motivation

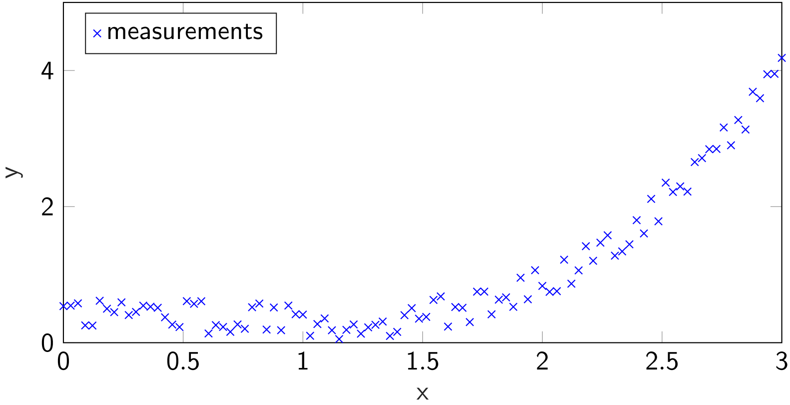
Motivation

We consider a set of 100 measurements (x_i, z_i) from some process:



Motivation

We consider a set of 100 measurements (x_i, z_i) from some process:



$\rightarrow$ Candidates: a quadratic or a cubic function.

Motivation

Goal

For a given set of of measurements (x_i, y_i) we need to find a function

$$f(x) = ax^2 + bx + c$$

or

$$f(x) = ax^3 + bx^2 + cx + d$$

such that either

$$f(x_i) = y_i, \quad \forall i$$

or

$$||\bar{y} - f(\bar{x})||_2 \to \min,$$

where $\bar{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ and $\bar{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}$.

Motivation

In case of a quadratic function $f(x) = ax^2 + bx + c$, we have 3 degrees of freedom (a, b, c). This yields a linear system

$$
\begin{aligned}
a\hat{x}_1^2 + b\hat{x}_1 + c &= \hat{y}_1 \\
a\hat{x}_2^2 + b\hat{x}_2 + c &= \hat{y}_2 \\
a\hat{x}_3^2 + b\hat{x}_3 + c &= \hat{y}_3
\end{aligned}
$$

Motivation

In case of a quadratic function $f(x) = ax^2 + bx + c$, we have 3 degrees of freedom (a, b, c). This yields a linear system

$$\begin{aligned} a\hat{x}_1^2 + b\hat{x}_1 + c &= \hat{y}_1 \\ a\hat{x}_2^2 + b\hat{x}_2 + c &= \hat{y}_2 \\ a\hat{x}_3^2 + b\hat{x}_3 + c &= \hat{y}_3 \end{aligned}$$

which can be rewritten to

$$\underbrace{\begin{bmatrix} \hat{x}_1^2 & \hat{x}_1 & 1 \\ \hat{x}_2^2 & \hat{x}_2 & 1 \\ \hat{x}_3^2 & \hat{x}_3 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}}_{v}.$$

Motivation

In case of a quadratic function $f(x) = ax^2 + bx + c$, we have 3 degrees of freedom $(a,\ b,\ c)$. This yields a linear system

$$\begin{aligned} a\hat{x}_1^2 + b\hat{x}_1 + c &= \hat{y}_1 \\ a\hat{x}_2^2 + b\hat{x}_2 + c &= \hat{y}_2 \\ a\hat{x}_3^2 + b\hat{x}_3 + c &= \hat{y}_3 \end{aligned}$$

which can be rewritten to

$$\underbrace{\begin{bmatrix} \hat{x}_1^2 & \hat{x}_1 & 1 \\ \hat{x}_2^2 & \hat{x}_2 & 1 \\ \hat{x}_3^2 & \hat{x}_3 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}}_{v}.$$

Now, the linear system can be solved for three arbitrary $(\hat{x}_1, \hat{y}_1)$, $(\hat{x}_2, \hat{y}_2)$, $(\hat{x}_3, \hat{y}_3)$ out of our set of measurements (x_i, y_i).

Motivation

We select $i = 1, 50, 100$ and obtain

$$A = \begin{bmatrix} 0 & 0 & 1.0000 \\ 2.2048 & 1.4848 & 1.0000 \\ 9.0000 & 3.0000 & 1.0000 \end{bmatrix}, \quad v = \begin{bmatrix} 0.5372 \\ 0.3533 \\ 4.1853 \end{bmatrix}$$

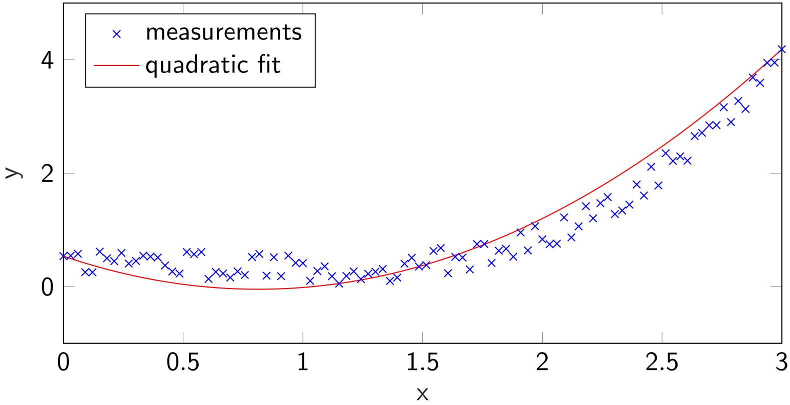
which yields

$$f(x) = 0.8843x^2 - 1.4370x + 0.5372.$$

and

$$||\bar{y} - f(\bar{x})||_2 = 3.3532$$

Motivation



Motivation

Second try: We assume that the connection between x and y is of cubic nature, thus we have to fit function $f(x) = ax^3 + bx^2 + cx + d$ with 4 degrees of freedom (a, b, c, d):

$$
\begin{aligned}
a\hat{x}_1^3 + b\hat{x}_1^2 + c\hat{x}_1 + d &= \hat{y}_1 \\
a\hat{x}_2^3 + b\hat{x}_2^2 + c\hat{x}_2 + d &= \hat{y}_2 \\
a\hat{x}_3^3 + b\hat{x}_3^2 + c\hat{x}_3 + d &= \hat{y}_3 \\
a\hat{x}_3^4 + b\hat{x}_3^4 + c\hat{x}_4 + d &= \hat{y}_4
\end{aligned}
$$

Motivation

Second try: We assume that the connection between x and y is of cubic nature, thus we have to fit function $f(x) = ax^3 + bx^2 + cx + d$ with 4 degrees of freedom (a, b, c, d):

$$
\begin{aligned}
a\hat{x}_1^3 + b\hat{x}_1^2 + c\hat{x}_1 + d &= \hat{y}_1 \\
a\hat{x}_2^3 + b\hat{x}_2^2 + c\hat{x}_2 + d &= \hat{y}_2 \\
a\hat{x}_3^3 + b\hat{x}_3^2 + c\hat{x}_3 + d &= \hat{y}_3 \\
a\hat{x}_3^4 + b\hat{x}_3^4 + c\hat{x}_4 + d &= \hat{y}_4
\end{aligned}
$$

We choose $i = 1, 33, 66, 100$ and obtain

$$
A = \begin{bmatrix} 0 & 0 & 0 & 1.0000 \\ 0.9118 & 0.9403 & 0.9697 & 1.0000 \\ 7.6418 & 3.8797 & 1.9697 & 1.0000 \\ 27.0000 & 9.0000 & 3.0000 & 1.0000 \end{bmatrix}, \quad v = \begin{bmatrix} 0.5372 \\ 0.4187 \\ 1.0653 \\ 4.1853 \end{bmatrix}.
$$

Motivation

This results in

$$f(x) = 0.26093x^3 - 037667x^2 - 0.00231x + 0.53723$$

with

$$||\bar{y} - f(\bar{x})||_2 = 2.3096$$

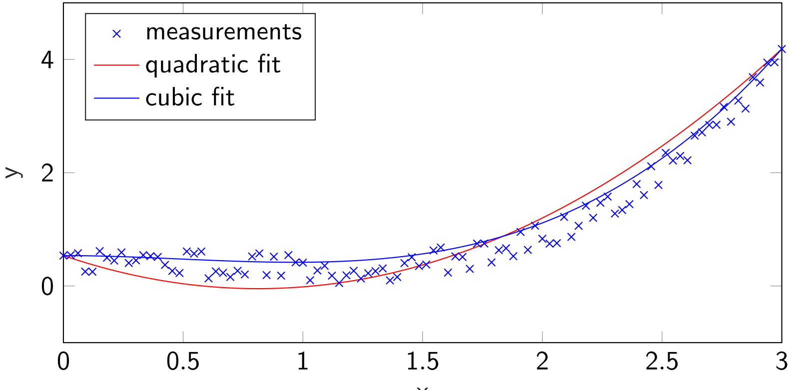
Motivation

This results in

$$f(x) = 0.26093x^3 - 037667x^2 - 0.00231x + 0.53723$$

with

$$||\bar{y} - f(\bar{x})||_2 = 2.3096$$



Motivation

Problem

We only use a small selection of measurements to fit the function. For a polynomial of degree p and n measurements, this will lead to linear system of dimension $n \times p$, which could not be solved with the LU decomposition.

E.g.:

$$\underbrace{\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_{y}$$

Motivation

Problem

We only use a small selection of measurements to fit the function. For a polynomial of degree p and n measurements, this will lead to linear system of dimension $n \times p$, which could not be solved with the LU decomposition.

E.g.:

$$\underbrace{\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ v_m \end{bmatrix}}_{y}$$

$\rightarrow$ With $n > p$ the system is over-determined, A^{-1} does not exist.

Motivation

Problem

We only use a small selection of measurements to fit the function. For a polynomial of degree p and n measurements, this will lead to linear system of dimension $n \times p$, which could not be solved with the LU decomposition.

E.g.:

$$\underbrace{\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{w} = \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_{y}$$

$\rightarrow$ With $n > p$ the system is over-determined, A^{-1} does not exist.

$\rightarrow$ Since the linear system is no longer uniquely defined, we search for the best fit in terms of

$$\min ||Aw - y||_2.$$

Least Squares

Least Squares

Definition 8.1

A linear system

$$Ax = b,$$

with $A \in \mathbb{R}^{n \times p}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^p$ is **over-determined** if $n > p$.

Least Squares

Definition 8.1

A linear system

$$Ax = b,$$

with $A \in \mathbb{R}^{n \times p}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^p$ is **over-determined** if $n > p$.

Definition 8.2

For a given over-determined linear system, the solution x which fulfills

$$\min ||Ax - b||_2$$

is called the **least squares solution**.

Least Squares

Since $||x||_2 \geqslant 0$, $\forall x$ we consider the equivalent function

$$\Phi(x) = \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}(Ax - b, Ax - b)_2.$$

Now, $\Phi(x)$ is differentiable and $\min \Phi(x)$ requires

$$\nabla\Phi(x) = A^T(Ax - b) = 0.$$

This leads to

$$\min \Phi(x) \Leftrightarrow A^T Ax = A^T b.$$

Least Squares

Since $||x||_2 \geqslant 0$, $\forall x$ we consider the equivalent function

$$\Phi(x) = \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}(Ax - b, Ax - b)_2.$$

Now, $\Phi(x)$ is differentiable and $\min \Phi(x)$ requires

$$\nabla\Phi(x) = A^T(Ax - b) = 0.$$

This leads to

$$\min \Phi(x) \Leftrightarrow A^T Ax = A^T b.$$

Remark 1

This approach is similar to the derivation of the Conjugate Gradient algorithm.

Least Squares

Since $||x||_2 \geqslant 0$, $\forall x$ we consider the equivalent function

$$\Phi(x) = \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}(Ax - b, Ax - b)_2.$$

Now, $\Phi(x)$ is differentiable and $\min \Phi(x)$ requires

$$\nabla\Phi(x) = A^T(Ax - b) = 0.$$

This leads to

$$\min \Phi(x) \Leftrightarrow A^T Ax = A^T b.$$

Remark 1

This approach is similar to the derivation of the Conjugate Gradient algorithm.

Remark 2

Solving a linear system with $A^T A$ instead of A means $\kappa_2(A) \rightsquigarrow \kappa_2(A^T A) \approx \kappa_2(A)^2$.

Least Squares

Lemma 8.3

Let $Q \in \mathbb{R}^{n\times n}$ *be an orthogonal matrix, i.e.* $Q^T Q = I$*, then it holds* $\forall x \in \mathbb{R}^n$*:*

$$\|x\|_2^2 = \|Qx\|_2^2 .$$

Least Squares

Lemma 8.3

Let $Q \in \mathbb{R}^{n\times n}$ *be an orthogonal matrix, i.e.* $Q^T Q = I$*, then it holds* $\forall x \in \mathbb{R}^n$*:*

$$\|x\|_2^2 = \|Qx\|_2^2 \,.$$

Proof.

$$\|Qx\|_2^2 = (Qx, Qx)_2 = (Qx)^T(Qx) = x^T Q^T Qx = (x, x)_2 = \|x\|_2^2$$

$\square$

Least Squares

New Idea

Due to Lemma 8.3 we can solve

$$\min \left\| Q^T A x - Q^T b \right\|_2$$

instead of

$$\min \left\| Ax - b \right\|_2$$

Thereby, Q is an orthogonal matrix and $Q^T A$ should have a "better" structure than A, e.g. it is upper triangular.

Least Squares

Theorem 8.4

Every matrix $A \in \mathbb{R}^{n \times p}$ *can be decomposed into*

$$QR = A,$$

where $Q \in \mathbb{R}^{n \times p}$ *is a orthogonal matrix and* $R \in \mathbb{R}^{p \times p}$ *is an upper triangular matrix.*

Least Squares

Theorem 8.4

Every matrix $A \in \mathbb{R}^{n \times p}$ *can be decomposed into*

$$QR = A,$$

where $Q \in \mathbb{R}^{n \times p}$ *is a orthogonal matrix and* $R \in \mathbb{R}^{p \times p}$ *is an upper triangular matrix.*

Lemma 8.5

Let $A \in \mathbb{R}^{n \times p}$ *has full column rank and the QR decomposition* $A = QR$*, then* R *has full rank and is non-singular.*

Least Squares

Theorem 8.4

Every matrix $A \in \mathbb{R}^{n\times p}$ can be decomposed into

$$QR = A,$$

where $Q \in \mathbb{R}^{n\times p}$ is a orthogonal matrix and $R \in \mathbb{R}^{p\times p}$ is an upper triangular matrix.

Lemma 8.5

Let $A \in \mathbb{R}^{n\times p}$ has full column rank and the QR decomposition $A = QR$, then R has full rank and is non-singular.

Remark 3

The QR decomposition is not unique.

Least Squares

Theorem 8.6

Every matrix $A \in \mathbb{R}^{n \times p}$ can be decomposed into a full QR decomposition

$$QR = A,$$

where $Q \in \mathbb{R}^{n \times n}$ is a orthogonal matrix and $R \in \mathbb{R}^{n \times p}$ with

$$Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \quad \textit{and} \quad R = \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix},$$

where $Q_1 \in \mathbb{R}^{n \times p}$, $Q_2 \in \mathbb{R}^{n \times (n-p)}$ and $\tilde{R} \in \mathbb{R}^{p \times p}$. For A and Q holds:

$$\operatorname{span} A = \operatorname{span} Q_1$$
$$Q_2^T A = 0$$

Least Squares

Solution of the Least Squares Problem

Let $\min \|Ax - b\|_2^2$ be a least squares problem, if A has full column rank, the solution x is given by

$$\min \Phi(x) \Leftrightarrow A^T Ax - A^T b = 0 \Leftrightarrow \qquad \begin{aligned} A^T Ax &= A^T b \\ (QR)^T QRx &= (QR)^T b \\ R^T Q^T QRx &= R^T Q^T b \\ R^T Rx &= R^T b \\ Rx &= b. \end{aligned}$$

Remark 4

The condition that A has full rank needs to be guaranteed by the choice of the basis functions.

QR Decompositon

Orthogonalization using Gram-Schmidt

QR Decompositon

Orthogonalization using Gram-Schmidt

We need an orthogonal basis $Q = \begin{bmatrix} q_1 & q_2 & \dots & q_p \end{bmatrix}$ for the columns of $A = \begin{bmatrix} a_1 & a_2 & \dots & a_p \end{bmatrix}$, i.e. for span A.

QR Decompositon

Orthogonalization using Gram-Schmidt

We need an orthogonal basis $Q = \begin{bmatrix} q_1 & q_2 & \dots & q_p \end{bmatrix}$ for the columns of $A = \begin{bmatrix} a_1 & a_2 & \dots & a_p \end{bmatrix}$, i.e. for span A.

Using a straight forward approach, we set

$$q_1 = \frac{a_1}{\left\|a_1\right\|_2} = \frac{a_1}{r_{11}}$$

and for $k = 2, \dots, p$ we can represent q_k as

$$q_k = \frac{1}{r_{kk}} \hat{q}_k = \frac{1}{r_{kk}} \left(a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right).$$

Multiplying $\hat{q}_k$ from right with q_j gives

$$\hat{q}_k^T q_j = a_k^T q_j - \sum_{i=1}^{k-1} r_{ik} q_i^T q_j$$

QR Decompositon

Orthogonalization using Gram-Schmidt

Since he have

$$q_i^T q_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

we get for $j < k$

$$\begin{aligned} \hat{q}_k^T q_j &= a_k^T q_j - \sum_{i=1}^{k-1} r_{ik} q_i^T q_j \\ 0 &= a_k^T q_j - r_{jk} q_j^T q_j = a_k^T q_j - r_{jk} \\ r_{jk} &= a_k^T q_j. \end{aligned}$$

QR Decompositon

Orthogonalization using Gram-Schmidt

Since he have

$$q_i^T q_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

we get for $j < k$

$$\begin{aligned} \hat{q}_k^T q_j &= a_k^T q_j - \sum_{i=1}^{k-1} r_{ik} q_i^T q_j \\ 0 &= a_k^T q_j - r_{jk} q_j^T q_j = a_k^T q_j - r_{jk} \\ r_{jk} &= a_k^T q_j. \end{aligned}$$

Now, we have

$$\begin{aligned} r_{kk} &= \|\hat{q}_k\|_2 \\ q_k &= \frac{1}{r_{kk}} \hat{q}_k \end{aligned}$$

QR Decompositon

Orthogonalization using Gram-Schmidt

Since he have

$$q_i^T q_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

we get for $j < k$

$$\begin{aligned} \hat{q}_k^T q_j &= a_k^T q_j - \sum_{i=1}^{k-1} r_{ik} q_i^T q_j \\ 0 &= a_k^T q_j - r_{jk} q_j^T q_j = a_k^T q_j - r_{jk} \\ r_{jk} &= a_k^T q_j. \end{aligned}$$

Now, we have

$$\begin{aligned} r_{kk} &= \|\hat{q}_k\|_2 \\ q_k &= \frac{1}{r_{kk}} \hat{q}_k \end{aligned}$$

This procedure is called **Gram-Schmidt-Orthogonalization.**

QR Decompositon

Orthogonalization using Gram-Schmidt

This procedure leads to an orthogonal matrix $Q \in \mathbb{R}^{n \times p}$ and an upper triangular matrix $R \in \mathbb{R}^{p \times p}$ with

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1p} \\ 0 & r_{22} & \dots & r_{2p} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & r_{pp} \end{bmatrix},$$

with $A = QR$.

Example 8.7

In case of our quadratic fitting, we obtain

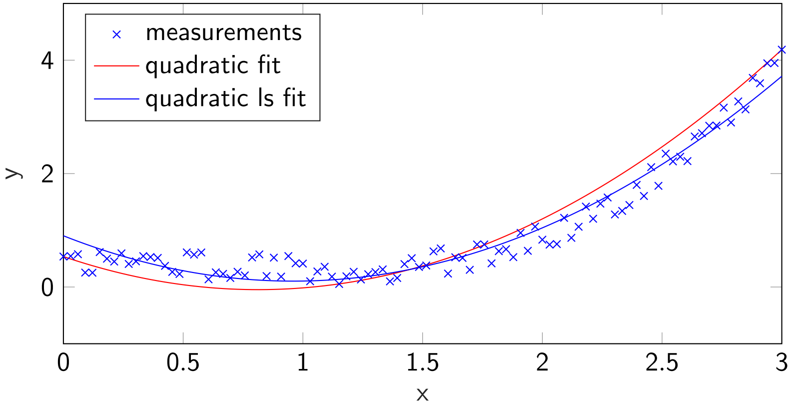
$$f(x) = 0.868x^2 - 1.6664x + 0.9034$$

with

$$\|\bar{y} - f(\bar{x})\|_2 = 2.2873$$

QR Decompositon

Orthogonalization using Gram-Schmidt



QR Decompositon

Example 8.8

In case of our cubic fitting, we obtain

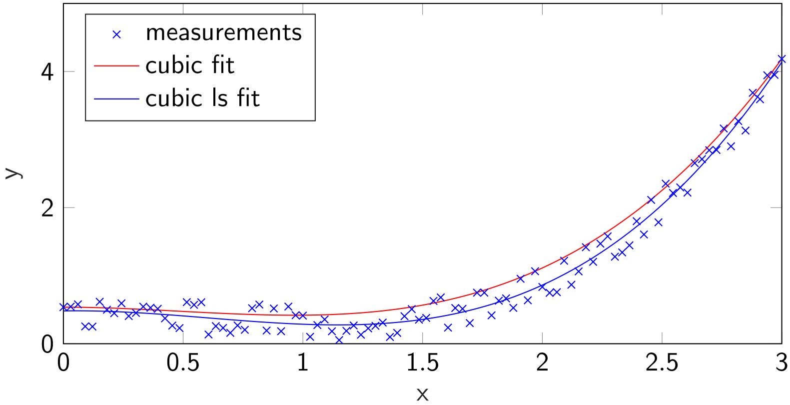
$$f(x) = 0.321939x^3 - 0.580698x^2 + 0.063395x + 0.481854$$

with

$$\|\bar{y} - f(\bar{x})\|_2 = 1.5393$$

QR Decompositon

Orthogonalization using Gram-Schmidt



QR Decompositon

For comparison:

- **Input function:** $f(x) = 0.3x^3 - 0.5x^2 + 0.5$ with random noise added
- **quadratic with 3 points**: $f(x) = 0.8843x^2 - 1.4370x + 0.5372$, $||\bar{y} - f(\bar{x})||_2 = 3.3532$
- **cubic with 4 points**: $f(x) = 0.26093x^3 - 037667x^2 - 0.00231x + 0.53723$, $||\bar{y} - f(\bar{x})||_2 = 2.3096$
- **quadratic with least squares**: $f(x) = 0.868x^2 - 1.6664x + 0.9034$, $||\bar{y} - f(\bar{x})||_2 = 2.2873$
- **cubic with least squares**: $f(x) = 0.321939x^3 - 0.580698x^2 + 0.063395x + 0.481854$, $||\bar{y} - f(\bar{x})||_2 = 1.5393$

QR Decompositon

For comparison:

- **Input function:** $f(x) = 0.3x^3 - 0.5x^2 + 0.5$ with random noise added
- **quadratic with 3 points**: $f(x) = 0.8843x^2 - 1.4370x + 0.5372$, $||\bar{y} - f(\bar{x})||_2 = 3.3532$
- **cubic with 4 points**: $f(x) = 0.26093x^3 - 037667x^2 - 0.00231x + 0.53723$, $||\bar{y} - f(\bar{x})||_2 = 2.3096$
- **quadratic with least squares**: $f(x) = 0.868x^2 - 1.6664x + 0.9034$, $||\bar{y} - f(\bar{x})||_2 = 2.2873$
- **cubic with least squares**: $f(x) = 0.321939x^3 - 0.580698x^2 + 0.063395x + 0.481854$, $||\bar{y} - f(\bar{x})||_2 = 1.5393$

But...

... the Gram-Schmidt procedure is numerically unstable.

QR Decompositon

Orthogonalization using Gram-Schmidt

The Gram-Schmidt procedure is a “left-looking” algorithm. For a column a_k it takes all previously computed columns q_i, $i < k$ and compute the influence on a_k, i.e.

$$r_{ik} = q_i^T a_k$$

and normalize the remaining vector afterwards with r_{kk}:

$$r_{kk} = \left\| \left(a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right) \right\|_2 .$$

A numerically more stable approach is, as soon as some q_k is known, remove its influence from the remaining columns a_j, $j > k$.

→ This leads to a “right-looking” variant, called **Modified-Gram-Schmidt (MGS)**.

QR Decompositon

Orthogonalization using Gram-Schmidt

Algorithm 8.1: Modified-Gram-Schmidt

Input: $A \in \mathbb{R}^{n \times p}$

Output: $Q \in \mathbb{R}^{n \times p}$, $R \in \mathbb{R}^{p \times p}$

for $k = 1 : p$ **do**

  $r_{kk} = \|a_k\|_2$;

  $q_k = \frac{1}{r_{kk}} a_k$;

  for $j = k + 1 : p$ **do**

    $r_{kj} = q_k^T a_j$;

    $a_j = a_j - r_{kj} q_k$;

QR Decompositon

Orthogonalization using Gram-Schmidt

Algorithm 8.1: Modified-Gram-Schmidt

Input: $A \in \mathbb{R}^{n \times p}$

Output: $Q \in \mathbb{R}^{n \times p}$, $R \in \mathbb{R}^{p \times p}$

for $k = 1 : p$ **do**

  $r_{kk} = \|a_k\|_2$;

  $q_k = \frac{1}{r_{kk}} a_k$;

  for $j = k + 1 : p$ **do**

    $r_{kj} = q_k^T a_j$;

    $a_j = a_j - r_{kj} q_k$;

Remark 5

- The algorithm takes $2np^2$ flops.
- A can be overwritten with Q but R needs to be stored separately, p^2 additional memory required.

Householder Transformation

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

Goal

We construct Q as a sequence of k orthogonal transformations P_k, $Q = P_1 P_2 \ldots$, where

- P_k can be stored with less then n memory and
- $P_k x$ costs less then $2n^2$ flops.

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

Goal

We construct Q as a sequence of k orthogonal transformations P_k, $Q = P_1 P_2 \dots$, where

- P_k can be stored with less then n memory and
- $P_k x$ costs less then $2n^2$ flops.

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

Goal

We construct Q as a sequence of k orthogonal transformations P_k, $Q = P_1 P_2 \ldots$, where

- P_k can be stored with less then n memory and
- $P_k x$ costs less then $2n^2$ flops.

$$P_1 A = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}$$

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

Goal

We construct Q as a sequence of k orthogonal transformations P_k, $Q = P_1 P_2 \ldots$, where

- P_k can be stored with less then n memory and
- $P_k x$ costs less then $2n^2$ flops.

$$P_2 P_1 A = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix}$$

QR Decompositon

Householder Transformation

We have to handle the following issues with the (Modified-)Gram-Schmidt procedure:

- $2np^2$ flops is very expensive for $p = n$ (LU: $\frac{2}{3}n^3$),
- p^2 extra memory required,
- stability issues, especially in the non-modified case.

Goal

We construct Q as a sequence of k orthogonal transformations P_k, $Q = P_1 P_2 \ldots$, where

- P_k can be stored with less then n memory and
- $P_k x$ costs less then $2n^2$ flops.

$$\underbrace{P_3 P_2 P_1}_{Q^T} A = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} = R$$

QR Decompositon

Householder Transformation

The goal can be fulfilled, if we obtain an orthogonal matrix $P \in \mathbb{R}^{n \times n}$ such that

$$Px = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The goal can be fulfilled, if we obtain an orthogonal matrix $P \in \mathbb{R}^{n \times n}$ such that

$$Px = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Definition 8.9

Let $v \in \mathbb{R}^n$, $v \neq 0$, then

$$P = I - \beta vv^T, \ \beta = \frac{2}{v^T v}$$

is called **Householder-Transformation**.

Theorem 8.10

Let $P \in \mathbb{R}^{n \times n}$ be a Householder-Transformation, then the following holds

- *P is orthogonal*
- *P is symmetric.*
- *Products of Householder-Transformations are orthogonal again.*

QR Decompositon

Householder Transformation

Theorem 8.10

Let $P \in \mathbb{R}^{n\times n}$ be a Householder-Transformation, then the following holds

- *P is orthogonal*
- *P is symmetric.*
- *Products of Householder-Transformations are orthogonal again.*

Regarding the goals:

- storing P costs $n+1$ memory,
- $Px = x - \beta vv^T x$ costs $4n$ flops.

QR Decompositon

Householder Transformation

How to chose v depending on x such that

$$Px = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}?$$

QR Decompositon

Householder Transformation

How to chose v depending on x such that

$$Px = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}?$$

We use Theorem 8.10 and Lemma 8.3 and we get from

$$||Px||_2 = ||x||_2$$

that

$$Px = \begin{bmatrix} \pm||x||_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

QR Decompositon

That means, we have to chose v such that

$$Px = \pm||x||_2 e_1.$$

which yields

$$Px = \left(I - \frac{2vv^T}{v^T v} \right) x = x - \frac{2v^T x}{v^T v} v.$$

QR Decompositon

That means, we have to chose v such that

$$Px = \pm||x||_2 e_1.$$

which yields

$$Px = \left(I - \frac{2vv^T}{v^T v}\right) x = x - \frac{2v^T x}{v^T v} v.$$

From this we know that $v \in \operatorname{span}\{x, e_1\}$, i.e. $v = x + \alpha e_1$.

With

$$v^T x = x^T x + \alpha x_1$$

and

$$v^T v = x^T x + 2\alpha x_1 + \alpha^2$$

QR Decompositon

That means, we have to chose v such that

$$Px = \pm ||x||_2 e_1.$$

which yields

$$Px = \left(I - \frac{2vv^T}{v^T v} \right) x = x - \frac{2v^T x}{v^T v} v.$$

From this we know that $v \in \operatorname{span}\{x, e_1\}$, i.e. $v = x + \alpha e_1$.

With

$$v^T x = x^T x + \alpha x_1$$

and

$$v^T v = x^T x + 2\alpha x_1 + \alpha^2$$

we obtain

$$\begin{aligned} Px &= \left(1 - 2 \frac{x^T x + \alpha x_1}{x^T x + 2\alpha x_1 + \alpha^2} \right) x - 2\alpha \frac{v^T x}{v^T v} e_1 \\ &= \frac{\alpha^2 - \|x\|_2^2}{x^T x + 2\alpha x_1 + \alpha^2} x - 2\alpha \frac{v^T x}{v^T v} e_1. \end{aligned}$$

QR Decompositon

Householder Transformation

Since we have to enforce

$$\frac{\alpha^2 - \|x\|_2^2}{x^T x + 2\alpha x_1 + \alpha^2} = 0$$

we set

$$\alpha = \pm \|x\|_2$$

and obtain

$$v = x \pm \|x\|_2 \, e_1 \quad \Rightarrow \quad Px = \mp \|x\|_2 \, e_1 .$$

QR Decompositon

Householder Transformation

Since we have to enforce

$$\frac{\alpha^2 - \|x\|_2^2}{x^T x + 2\alpha x_1 + \alpha^2} = 0$$

we set

$$\alpha = \pm \|x\|_2$$

and obtain

$$v = x \pm \|x\|_2 \, e_1 \quad \Rightarrow \quad Px = \mp \|x\|_2 \, e_1 .$$

Remark 6

- Since the sign of α can be selected freely, we choose the numerically more stable variant to avoid cancelation.
- v can be normalized, such that $v(1) = 1$.

QR Decompositon

Householder Transformation

Algorithm 8.2: Computation of a Householder Vector

Input: $x \in \mathbb{R}^n$

Output: $v \in \mathbb{R}^n$ with $v(1) = 1$, $\beta \in \mathbb{R}$ such that $Px = \pm \|x\|_2 \, e_1$

$\sigma = \|x(2:n)\|_2$;

$v = [1; x(2:n)]$;

if $\sigma = 0$ **and** $x_1 \geqslant 0$ **then** $\beta = 0$;

else if $\sigma = 0$ **and** $x_1 < 0$ **then** $\beta = 2$;

else

- $\mu = \sqrt{(\sigma + x(1)^2}$;
- **if** $x(1) \leqslant 0$ **then**
  - $v(1) = x(1) - \mu$;
- **else**
  - $v(1) = \frac{-\sigma}{x(1)+\mu}$;
- $\beta = \frac{2v(1)^2}{\sigma + v(1)^2}$;
- $v = \frac{1}{v(1)} v$;

QR Decompositon

Householder Transformation

Applying P to a vector is performed as

$$Px = (I - \beta v v^T)x = x - \beta v \underbrace{v^T x}_{\gamma} = x - (\beta\gamma)v$$

which is

- a scalar product with $2n$ flops and
- an `axpy` operation with $2n$ flops.

QR Decompositon

Householder Transformation

Applying P to a vector is performed as

$$Px = (I - \beta vv^T)x = x - \beta v \underbrace{v^T x}_{\gamma} = x - (\beta\gamma)v$$

which is

- a scalar product with $2n$ flops and
- an `axpy` operation with $2n$ flops.

Applying P to a matrix $A \in \mathbb{R}^{n\times m}$ is computed using

$$PA = (I - \beta vv^T)A = A - \beta v \underbrace{v^T A}_{w} = A - \beta vw$$

which consists of

- a matrix-vector product with $2mn$ flops and
- a rank-1 update with $2mn$ flops.

Householder QR Decompostion

QR Decompositon

Householder QR Decompostion

Using the Householder-Transformation we can create a sequence of P_1 such that we obtain an orthogonal Q and a upper triangular R.

First we compute P_1 from the first column a_1 and get

$$P_1 A = \left[\begin{array}{ccc} * & * & * \\ 0 & & \\ 0 & \multicolumn{2}{c}{\raisebox{1ex}{A_1}} \end{array}\right].$$

QR Decompositon

Householder QR Decompostion

Using the Householder-Transformation we can create a sequence of P_1 such that we obtain an orthogonal Q and a upper triangular R.

First we compute P_1 from the first column a_1 and get

$$P_1 A = \begin{bmatrix} * & * & * \\ 0 & & \\ 0 & \multicolumn{2}{c}{A_1} \end{bmatrix}.$$

Now we take the first column $\tilde{a}_1$ of A_1 and compute P_2:

$$\begin{bmatrix} 1 & \\ & P_2 \end{bmatrix} P_1 A = \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & A_2 \end{bmatrix}.$$

QR Decompositon

Householder QR Decompostion

Using the Householder-Transformation we can create a sequence of P_1 such that we obtain an orthogonal Q and a upper triangular R.

First we compute P_1 from the first column a_1 and get

$$P_1 A = \begin{bmatrix} * & * \quad * \\ 0 & \\ 0 & \raisebox{0.5em}{A_1} \end{bmatrix}.$$

Now we take the first column $\tilde{a}_1$ of A_1 and compute P_2:

$$\begin{bmatrix} 1 & \\ & P_2 \end{bmatrix} P_1 A = \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & A_2 \end{bmatrix}.$$

This repeats until the lower right block is of size 0 or upper triangular.

QR Decompositon

Householder QR Decompostion

Algorithm 8.3: Householder QR

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$
Output: $R \in \mathbb{R}^{p \times p}$, P_1, P_2, ..., P_p Householder-Transformations

for $j = 1, \ldots, p$ **do**
  Compute v_j, β_j from $A(j : n, j)$ using Algorithm 8.2;
  $A(j : n, j : p) = (I - \beta_j v_j v_j^T)A(j : n, j : p)$;

Algorithm 8.3: Householder QR

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$

Output: $R \in \mathbb{R}^{p \times p}$, P_1, P_2, ..., P_p Householder-Transformations

for $j = 1, \ldots, p$ **do**

  Compute v_j, β_j from $A(j:n, j)$ using Algorithm 8.2;

  $A(j:n, j:p) = (I - \beta_j v_j v_j^T) A(j:n, j:p)$;

Remark 7

- A is overwritten with R.
- v_j are normalized, i.e. $v_j(1) = 1$, thus they can be stored in the newly created zeros in the lower triangle of A.
- We need p memory locations to store β_j.
- It costs $2p^2(n - \frac{p}{3})$ flops. (If $p = n$, we have $\frac{8}{3}n^3$ flops)

QR Decompositon

Householder QR Decompostion – Where is Q?

The Householder-QR overwrites A with R and stores P_j as v_j and β_j and not explicitly as Q. Thus we have:

$$Q = P_1 \begin{bmatrix} 1 & \\ & P_2 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & P_3 \end{bmatrix} \dots = (I - \beta_1 v_1 v_1^T) \begin{bmatrix} 1 & \\ & (I - \beta_2 v_2 v_2^T) \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & (I - \beta_3 v_3 v_3^T) \end{bmatrix} \dots \quad (1)$$

QR Decompositon

Householder QR Decompostion – Where is Q?

The Householder-QR overwrites A with R and stores P_j as v_j and β_j and not explicitly as Q. Thus we have:

$$Q = P_1 \begin{bmatrix} 1 & \\ & P_2 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & P_3 \end{bmatrix} \dots = (I - \beta_1 v_1 v_1^T) \begin{bmatrix} 1 & \\ & (I - \beta_2 v_2 v_2^T) \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & (I - \beta_3 v_3 v_3^T) \end{bmatrix} \dots \quad (1)$$

Lemma 8.11

Let Q be given as sequence of Householder-Transformations $P_1, \dots P_p$ of a QR decomposition of $A \in \mathbb{R}^{n \times p}$ as in Eqn (1). Furthermore, let $C \in \mathbb{R}^{n \times m}$ be a matrix. Then the matrix-matrix products

$$QC \quad \text{and} \quad Q^T C$$

cost $mp(2n - p) = 2mnp - 2mp^2$ flops.

Remark 8

- In most applications, Q is not required explictly, only its application to a vector/matrix.
- As long $n \gg p$, using the factorized version is much cheaper than using a `gemm` operation ($\to 2n^2m$ flops)
- It is numerically stable and does not require pivoting, as long as A has full column rank.
- ... but the Algorithm is built on top of level-1 and level-2 operations.

QR Decompositon

Householder QR Decompostion

Remark 8

- In most applications, Q is not required explictly, only its application to a vector/matrix.
- As long $n \gg p$, using the factorized version is much cheaper than using a `gemm` operation ($\rightarrow 2n^2m$ flops)
- It is numerically stable and does not require pivoting, as long as A has full column rank.
- ... but the Algorithm is built on top of level-1 and level-2 operations.

Remark 9

The algorithms are available in LAPACK:

LARFG	compute a Householder-Transformation
LARF	apply a Householder-Transformation
GEQRF	compute the QR decomposition as in Algorithm 8.3
ORMQR	apply a factored Q to a right hand side
GELS	solve a least squares problem in a single step

Level-3 Algortihms

Problem

The computation of the Householder-Transformation and its application requires at most a rank-1 update and a matrix-vector product.

QR Decompositon

Level-3 Algortihms

Problem

The computation of the Householder-Transformation and its application requires at most a rank-1 update and a matrix-vector product.

Goal

We need to accumulate products of P_j without forming a single P_j explicitly:

- less then $\mathcal{O}(n^3)$ flops,
- less then n^2 memory.

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$P_1 P_2 = \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right)$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$\begin{aligned} P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\ &= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \end{aligned}$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$
\begin{aligned}
P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\
&= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T
\end{aligned}
$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$
\begin{aligned}
P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\
&= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \beta_1 & \beta_1 \beta_2 v_1^T v_2 \\ & \beta_2 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}^T}_{V^T}
\end{aligned}
$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$
\begin{aligned}
P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\
&= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \beta_1 & \beta_1 \beta_2 v_1^T v_2 \\ & \beta_2 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}^T}_{V^T} \\
&= I - \underbrace{VT}_{W} \underbrace{V^T}_{Y}
\end{aligned}
$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$\begin{aligned}
P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\
&= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \beta_1 & \beta_1 \beta_2 v_1^T v_2 \\ & \beta_2 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}^T}_{V^T} \\
&= I - \underbrace{VT}_{W} \underbrace{V^T}_{Y} \\
&= I - WY
\end{aligned}$$

QR Decompositon

Let $P_1 = I - \beta_1 v_1 v_1^T$ and $P_2 = I - \beta_2 v_2 v_2^T$:

$$
\begin{aligned}
P_1 P_2 &= \left(I - \beta_1 v_1 v_1^T\right)\left(I - \beta_2 v_2 v_2^T\right) \\
&= I \underbrace{-\beta_1 v_1 v_1^T}_{P_1} \underbrace{-\beta_2 v_2 v_2^T}_{P_2} + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \beta_1 & \\ & \beta_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T + \beta_1 \beta_2 v_1 v_1^T v_2 v_2^T \\
&= I - \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \beta_1 & \beta_1 \beta_2 v_1^T v_2 \\ & \beta_2 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} v_1 & v_2 \end{bmatrix}^T}_{V^T} \\
&= I - \underbrace{VT}_{W} \underbrace{V^T}_{Y} \\
&= I - WY
\end{aligned}
$$

Cost: one scalar product

QR Decompositon

Definition 8.12

Let P_1 and P_2 be two Householder-Transformations and $Q = P_1 P_2$ their product, then the representation

$$Q = I - VTV^T,$$

with T upper triangular, or

$$Q = I - WY,$$

with $W = VT$ and $Y = V^T$, is called **compact WY representation**.

Lemma 8.13

Let $Q \in \mathbb{R}^{n\times n}$ be a orthogonal matrix in compact WY representation $Q = I - VTV^T$ and $P = I - \beta ww^T$ a Householder-Transformation. Then the product $Q_+ = QP$ is given by

$$Q_+ = QP = I - V_+ T_+ V_+^T,$$

where

$$V_+ = \begin{bmatrix} V & w \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} T & -\beta TV^T w \\ & \beta \end{bmatrix}.$$

Lemma 8.13

Let $Q \in \mathbb{R}^{n\times n}$ be a orthogonal matrix in compact WY representation $Q = I - VTV^T$ and $P = I - \beta ww^T$ a Householder-Transformation. Then the product $Q_+ = QP$ is given by

$$Q_+ = QP = I - V_+ T_+ V_+^T,$$

where

$$V_+ = \begin{bmatrix} V & w \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} T & -\beta TV^T w \\ & \beta \end{bmatrix}.$$

Using the Lemma, we can subsequently accumulate a set of Householder Transformations into a matrix-valued object, but with increasing size of V (and T) this procedure gets more expensive.

QR Decompositon

Level-3 Algortihms

Algorithm 8.4: Computation of the WY representation

Input: $P_1, \dots, P_r$ Householder-Transformation with $v_1, \dots, v_r$ and $\beta_1, \dots, \beta_r$
Output: V and T such that $Q = P_1 P_2 \cdots P_r = I - VTV^T$

$V = v_1$;
$T = \beta_1$;
for $k = 2, \dots, r$ **do**
    $z = -\beta_k TV^T v_k$;
    $V = \begin{bmatrix} V & v_k \end{bmatrix}$;
    $T = \begin{bmatrix} T & z \\ & \beta_k \end{bmatrix}$;

- **Cost:** $2r^2 n - \frac{2}{3} r^3$ flops
- v_k and β_k are still accessible $\rightarrow$ use of level-2 alg. possible
- Application of Q, $QC = (I - VTV^T)C$:
  - 2 general matrix-matrix products
  - 1 triangular matrix-matrix product
- r^2 auxiliary memory for T

QR Decompositon

Level-3 Algortihms

We assume that $P_1, \ldots, P_p$ are from the Householder-QR Algorithm 8.3. Now we, have

$$
\begin{aligned}
P_1 &= I - \beta_1 v_1 v_1^T \\
P_2 &= \begin{bmatrix} 1 & \\ & I - \beta_2 v_2 v_2^T \end{bmatrix} \\
P_3 &= \begin{bmatrix} 1 & & \\ & 1 & \\ & & I - \beta_3 v_3 v_3^T \end{bmatrix} \\
&\vdots
\end{aligned}
$$

but is this compatible with Lemma 8.13?

QR Decompositon

Level-3 Algortihms

We assume that $P_1, \ldots, P_p$ are from the Householder-QR Algorithm 8.3. Now we, have

$$
\begin{aligned}
P_1 &= I - \beta_1 v_1 v_1^T \\
P_2 &= \begin{bmatrix} 1 & \\ & I - \beta_2 v_2 v_2^T \end{bmatrix} \\
P_3 &= \begin{bmatrix} 1 & & \\ & 1 & \\ & & I - \beta_3 v_3 v_3^T \end{bmatrix} \\
&\vdots
\end{aligned}
$$

but is this compatible with Lemma 8.13?

$$
P_2 = \begin{bmatrix} 1 & \\ & I - \beta_2 v_2 v_2^T \end{bmatrix} = I - \beta_2 \begin{bmatrix} 0 \\ v_2 \end{bmatrix} \begin{bmatrix} 0 \\ v_2 \end{bmatrix}^T
$$

QR Decompositon

If $P_1, \ldots, P_p$ are generated by the Householder-QR Algorithm 8.3, do we need

- 2 general matrix-matrix products, and
- 1 triangular matrix-matrix product

for computing QC or $Q^T C$?

QR Decompositon

If $P_1, \ldots, P_p$ are generated by the Householder-QR Algorithm 8.3, do we need

- 2 general matrix-matrix products, and
- 1 triangular matrix-matrix product

for computing QC or $Q^T C$?

We have v_k is of length $n - k + 1$, the top $p \times p$ part of the matrix

$$V = \begin{bmatrix} & & 0 & & 0 \\ & 0 & & & \\ v_1 & & 0 & \ldots & \vdots \\ & v_2 & & & \\ & & v_3 & & v_p \end{bmatrix}$$

is (unit) lower triangular.

$\rightarrow$ we have 3 triangular(-like) matrix-matrix products.

QR Decompositon

Level-3 Algortihms

With the help of Algorithm 8.4 we can create a level-3 enabled version of the Householder-QR decomposition. As in the LU decomposition, we assume a block size of r.

QR Decompositon

Level-3 Algortihms

With the help of Algorithm 8.4 we can create a level-3 enabled version of the Householder-QR decomposition. As in the LU decomposition, we assume a block size of r.

Algorithm 8.5: Level-3 Householder-QR (Variant 1)

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$, block size r
Output: $R \in \mathbb{R}^{p \times p}$, $Q = P_1 \dots P_p$ as $v_1, \dots, v_p$ and $\beta_1, \dots, \beta_p$

```
for k = 1, ..., p with step size r do
    τ = min(p − k + 1, r);
    Compute P_k, ... P_{k+τ−1} from A(k : n, k : k + τ − 1) using Algorithm 8.3;
    Compute V and T from P_k, ... P_{k+τ−1} from A(k : n, k : k + τ − 1) using Algorithm 8.4;
    Update A(k : n, k + τ : p) = (I − V T V^T) A(k : n, k + τ : p);
```

for $k = 1, \dots, p$ **with step size** r **do**
- $\tau = \min(p - k + 1, r)$;
- Compute $P_k, \dots P_{k+\tau-1}$ from $A(k:n, k:k+\tau-1)$ using Algorithm 8.3;
- Compute V and T from $P_k, \dots P_{k+\tau-1}$ from $A(k:n, k:k+\tau-1)$ using Algorithm 8.4;
- Update $A(k:n, k+\tau:p) = (I - VTV^T)A(k:n, k+\tau:p)$;

QR Decompositon

Level-3 Algortihms

With the help of Algorithm 8.4 we can create a level-3 enabled version of the Householder-QR decomposition. As in the LU decomposition, we assume a block size of r.

Algorithm 8.5: Level-3 Householder-QR (Variant 1)

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$, block size r
Output: $R \in \mathbb{R}^{p \times p}$, $Q = P_1 \dots P_p$ as $v_1, \dots, v_p$ and $\beta_1, \dots, \beta_p$

for $k = 1, \dots, p$ **with step size** r **do**
  $\tau = \min(p - k + 1, r)$;
  Compute $P_k, \dots P_{k+\tau-1}$ from $A(k : n, k : k + \tau - 1)$ using Algorithm 8.3;
  Compute V and T from $P_k, \dots P_{k+\tau-1}$ from $A(k : n, k : k + \tau - 1)$ using Algorithm 8.4;
  Update $A(k : n, k + \tau : p) = (I - VTV^T)A(k : n, k + \tau : p)$;

- v_j is stored in the lower part of A.
- R is the upper right $p \times p$ triangle of A.
- T is a temporary value of size $r \times r$.

Remark 10

Algorithm 8.5 has the following properties:

- The updates on A are performed as (triangular) matrix-matrix products.
- The output is compatible to the level-2 Householder-QR decomposition.
- The algorithm needs slightly more flops than the level-2 variant, but this is negligible for a moderate block size r.
- LAPACK **GEQRF** implements this approach.

Remark 10

Algorithm 8.5 has the following properties:

- The updates on A are performed as (triangular) matrix-matrix products.
- The output is compatible to the level-2 Householder-QR decomposition.
- The algorithm needs slightly more flops than the level-2 variant, but this is negligible for a moderate block size r.
- LAPACK **GEQRF** implements this approach.

$\rightarrow$ The algorithm has still a high portion of level-2 operations, especially since the computation of T is an extra step.

Remark 10

Algorithm 8.5 has the following properties:

- The updates on A are performed as (triangular) matrix-matrix products.
- The output is compatible to the level-2 Householder-QR decomposition.
- The algorithm needs slightly more flops than the level-2 variant, but this is negligible for a moderate block size r.
- LAPACK **GEQRF** implements this approach.

$\rightarrow$ The algorithm has still a high portion of level-2 operations, especially since the computation of T is an extra step.

$\rightarrow$ Unify the level-2 part (Step 2) and the computation of T (Step 3).

QR Decompositon

Level-3 Algortihms

Algorithm 8.6: Householder QR with T accumulation

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$
Output: $R \in \mathbb{R}^{p \times p}$, V, T such that $Q = I - VTV^T$

$V = [\ ]$;
$T = [\ ]$;
for $j = 1, \dots, p$ **do**
  Compute v_j, β_j from $A(j:n, j)$ using Algorithm 8.2;
  $z = -\beta_j T V^T v_j$;
  $V = \begin{bmatrix} V & v_j \end{bmatrix}$;
  $T = \begin{bmatrix} T & z \\ & \beta_j \end{bmatrix}$;
  $A(j:n, j:p) = (I - \beta_j v_j v_j^T) A(j:n, j:p)$;

Remark 11

- The integration of Algorithm 8.6 into Algorithm 8.5 in the foundation of the **GEQRT** routine in LAPACK.
- Algorithm 8.6 is implemented as **GEQRT2** in LAPACK.

QR Decompositon

Level-3 Algortihms

Question

Why is V and T only used in block of size r and not in the end for $Q = P_1 P_2 \cdots P_p = I - VTV^T$?

Question

Why is V and T only used in block of size r and not in the end for $Q = P_1 P_2 \cdots P_p = I - VTV^T$?

Example 8.14

Let $A \in \mathbb{R}^{n \times n}$ with $A = QR$, $Q = P_1 P_2 \cdots P_p$ as in Householder-QR decomposition, and $C \in \mathbb{R}^{n \times m}$, then the computation of

$$QC = P_1 \cdots P_p C$$

costs $2n^2 m$ flops, which is the cost of general matrix-matrix product. If Q is represented as $Q = I - VTV^T$, with $V, T \in \mathbb{R}^{n \times n}$, the evaluation of

$$QC = (I - VTV^T)C$$

costs $3n^2 m$ flops, since V and T are triangular matrices.

QR Decompositon

Level-3 Algortihms

→ The compact WY representation allows a level-3 enabled computation of the QR decomposition but applying Q seem to more efficient in terms of P_j.

QR Decompositon

Level-3 Algortihms

→ The compact WY representation allows a level-3 enabled computation of the QR decomposition but applying Q seem to more efficient in terms of P_j.

Solution

We group the Householder-Transformations $P_1, \ldots, P_p$ into k groups of size r:

→ $P_1, \ldots P_r \Rightarrow V_1, T_1$

→ $P_{r+1}, \ldots P_{2r} \Rightarrow V_2, T_2$

→ $P_{2r+1}, \ldots P_{3r} \Rightarrow V_3, T_3$

→ $\ldots$

and store them as

$$V = \begin{bmatrix} V_1 & V_2 & \ldots & V_k \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} T_1 & T_2 & \ldots & T_k \end{bmatrix}.$$

QR Decompositon

Level-3 Algortihms

This rearrangement of V_j and T_j leads to the following properties:

- T_j can be reused after computing the QR decomposition.
- $r \times p$ memory required for storing T.
- V_j is already stored in A for free.
- Applying $Q = (I - V_1 T_1 V_1^T)(I - V_2 T_2 V_2^T)\dots$ is still in $\mathcal{O}(2n^2 p)$ flops.
- QR decomposition, application of Q and solving with R are now in a level-3 enabled shape.
- Implemented as **GEMQRT** in LAPACK.

Next Problem

What if n is getting very large, such that the level-2 part in Algorithm 8.5 gains influence ?

Next Problem

What if n is getting very large, such that the level-2 part in Algorithm 8.5 gains influence ?

Idea

For each panel of block size r, we use the level-3 algorithm recursively again with block size $\frac{r}{2}$ until it is worth to switch back to the level-2 algorithm.

QR Decompositon

Level-3 Algortihms

Algorithm 8.7: Recursive compact *WY* Householder QR decomposition (**RQRT**)

Input: $A \in \mathbb{R}^{n \times p}$, $n \geqslant p$, threshold l, $1 \leqslant l \leqslant \frac{p}{2}$ for level-2

Output: $R \in \mathbb{R}^{p \times p}$, V, T such that $Q = I - VTV^T$

if $p \leqslant l$ **then**

  Compute V, T, R using Algorithm 8.6 (Level-2 with T accumulation);

else

  $p_1 = \lfloor \frac{p}{2} \rfloor$;

  Compute V_1, T_1, R_1 from $A(1:n, 1:p_1)$ using **RQRT** again.;

  $A(1:n, p_1+1:p) = (I - V_1 T_1 V_1^T) A(1:n, p_1+1:p)$;

  Compute V_2, T_2, R_2 from $A(p_1+1:n, p_1+1:p)$ using **RQRT** again.;

  $\tilde{T} = -T_1 V_1^T V_2 T_2$;

  $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$, $T = \begin{bmatrix} T_1 & \tilde{T} \\ & T_2 \end{bmatrix}$, $R = \begin{bmatrix} R_1 & A(1:p_1, p_1+1:p) \\ & R_2 \end{bmatrix}$;

QR Decompositon

Level-3 Algortihms

Algorithm 8.7: Recursive compact *WY* Householder QR decomposition (**RQRT**)

Input: $A \in \mathbb{R}^{n\times p}$, $n \geqslant p$, threshold l, $1 \leqslant l \leqslant \frac{p}{2}$ for level-2

Output: $R \in \mathbb{R}^{p\times p}$, V, T such that $Q = I - VTV^T$

```
if p ⩽ l then
    Compute V, T, R using Algorithm 8.6 (Level-2 with T accumulation);
else
    p1 = ⌊p/2⌋;
    Compute V1, T1, R1 from A(1 : n, 1 : p1) using RQRT again.;
    A(1 : n, p1 + 1 : p) = (I − V1 T1 V1^T) A(1 : n, p1 + 1 : p);
    Compute V2, T2, R2 from A(p1 + 1 : n, p1 + 1 : p) using RQRT again.;
    T~ = −T1 V1^T V2 T2;
    V = [V1 V2], T = [T1 T~; T2], R = [R1 A(1 : p1, p1 + 1 : p); R2];
```

$$p_1 = \lfloor \tfrac{p}{2} \rfloor; \quad A(1:n, p_1+1:p) = (I - V_1 T_1 V_1^T) A(1:n, p_1+1:p); \quad \tilde{T} = -T_1 V_1^T V_2 T_2;$$

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}, \quad T = \begin{bmatrix} T_1 & \tilde{T} \\ & T_2 \end{bmatrix}, \quad R = \begin{bmatrix} R_1 & A(1:p_1, p_1+1:p) \\ & R_2 \end{bmatrix};$$

- Algorithm 8.5 with **RQRT** for the panels and $l = 1$ gives LAPACK's **GEQRT**
- **RQRT** with $l = 1$ is available as **GEQRT3** in LAPACK

Alternative QR Variants

QR Decompositon

Alternative QR Variants – Givens-Rotation QR

Based on

$$\underbrace{\begin{bmatrix} c & -s \\ s & c \end{bmatrix}}_{G} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix},$$

where $r = \sqrt{a^2 + b^2}$ and

$$c \leftarrow \frac{a}{r}$$
$$s \leftarrow -\frac{b}{r}.$$

and $GG^T = I$.

QR Decompositon

Alternative QR Variants – Givens-Rotation QR

Based on

$$\underbrace{\begin{bmatrix} c & -s \\ s & c \end{bmatrix}}_{G} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix},$$

where $r = \sqrt{a^2 + b^2}$ and

$$c \leftarrow \frac{a}{r}$$
$$s \leftarrow -\frac{b}{r}.$$

and $GG^T = I$.

- numerically stable has the Householder Transformation
- can be use if only a few elements below the diagonal exists
- slow, level-3 formulations complicated or not available
- mostly used in the Hessenberg-QR, i.e. for matrices with one sub-diagonal
- Q needs to be setup explicitly or c,s need to be stored for each transformation

QR Decompositon

Alternative QR Variants – Tall-Skinny QR (TSQR)

The computation of a single Householder-Transformation gets slow if the vector v gets too long, e.g. $v \in \mathbb{R}^n$ with $n > 10^6$.

- often the case in real world parameter fitting problems, $n > 10^6$ and $p \approx 100$.
- large leading dimension causes a loss in data-locality when applying P.
- the vector v needs to be accesses several times when computing and applying $P \rightarrow$ many cache misses due to its length

QR Decompositon

Alternative QR Variants – Tall-Skinny QR (TSQR)

The computation of a single Householder-Transformation gets slow if the vector v gets too long, e.g. $v \in \mathbb{R}^n$ with $n > 10^6$.

- often the case in real world parameter fitting problems, $n > 10^6$ and $p \approx 100$.
- large leading dimension causes a loss in data-locality when applying P.
- the vector v needs to be accesses several times when computing and applying $P \rightarrow$ many cache misses due to its length

Basic Idea:

- split the rows into blocks of n_b rows.
- perform a Householder-QR Decomposition on the top block
- use Householder-Transformations to join the remaining blocks with the top block one by one.
- available as **LATSQR** in LAPACK

QR Decompositon

Alternative QR Variants – Communication Avoiding QR (CAQR)

The Tall-Skinny QR does a good job on matrices with large number of rows, but it is not well parallelizable.

The Communication Avoiding QR(CAQR) on a tall-and-skinny matrix ($n \gg p$) works as follows:

- split the rows into blocks of n_b rows.
- perform a Householder-QR Decomposition on **each** block (in parallel)
- perform a binary reduction: combine two neighboring blocks using Householder-Transforms and repeat this until one upper triangular block is left
- works in massive parallel environments

QR Decompositon

Alternative QR Variants – Tile-QR

The TSQR and the CAQR are designed for the $n \gg p$ case, for nearly square matrices $n \approx p$ the TSQR approach can be extended:

- the matrix is partitioned into blocks of $n_b \times p_b$,
- in each block-column a TSQR-performed and applied to the remaining block columns,
- parallelization via DAG/data dependencies easily possible,
- starting with 4 CPU cores, this is beneficial over a classical Householder-QR,
- implemented as **GEQRT** in PLASMA.

QR Decompositon

Alternative QR Variants – GPUs and Other

On GPUs:

- Hybrid CPU-GPU variants of the Householder-QR
- CAQR and TSQR get replaced by the approximate Householder QR (AHQR)

Sparse Matrices:

- similar problems as in LU/Cholesky decomposition: fill-in in Q and R
- clever reordering and graph theory necessary

Non-Householder based:

We use

$$A^T A = (QR)^T QR = R^T Q^T QR = R^T R,$$

where $R^T R$ is the Cholesky decomposition of $A^T A$ and Q is given as $Q = AR^{-1}$.