

MA5233 Computational Mathematics

Lecture 4: QR Factorisation

Simon Etter



2019/2020

QR Factorisation

QR factorisation

- ▶ An alternative way to solve linear systems.
- ▶ A basic building block in many more advanced algorithms.

Outline

- ▶ QR factorisation: definition and applications.
- ▶ Algorithms:
 - ▶ Gram-Schmidt orthogonalisation
 - ▶ Householder reflection
 - ▶ Givens rotation
- ▶ `qr()` in Julia.
- ▶ Complexity and stability.

QR Factorisation

QR factorisation

Any $A \in \mathbb{K}^{m \times n}$ can be written as $A = QR$, where

- ▶ $Q \in \mathbb{K}^{m \times m}$ is orthogonal ($Q^H Q = I$), and
- ▶ $R \in \mathbb{K}^{m \times n}$ is upper triangular ($R(i, j) = 0$ for $i > j$).

Applications

- ▶ Solve linear systems (assuming $m = n$):

$$QRx = b \iff x = R^{-1}Q^H b.$$

- ▶ Determine orthogonal basis (see next slide).

QR Factorisation

Key property of QR factorisation

Q factor provides an orthogonal basis for range of A .

More precisely:

$$A[:, 1] = Q[:, 1] R[1, 1],$$

$$A[:, 2] = Q[:, 1] R[1, 2] + Q[:, 2] R[2, 2],$$

$$A[:, 3] = Q[:, 1] R[1, 3] + Q[:, 2] R[2, 3] + Q[:, 3] R[3, 3]$$

...

Conclusion: $\text{span } A[:, 1 : k] = \text{span } Q[:, 1 : k]$.

QR Factorisation

Illustration of QR decomposition

Case $m > n$:

$$\left(\begin{array}{c} \text{gray rectangle} \end{array} \right) = \left(\begin{array}{c} \text{gray square} \quad \text{red rectangle} \end{array} \right) \left(\begin{array}{c} \text{upper triangular triangle} \end{array} \right)$$

Case $m < n$:

$$\left(\begin{array}{c} \text{gray rectangle} \end{array} \right) = \left(\begin{array}{c} \text{gray square} \end{array} \right) \left(\begin{array}{c} \text{trapezoidal shape} \end{array} \right)$$

Red part of Q factor is redundant since it gets multiplied by zeros in R .
QR factorisation without red part is known as “thin” QR.

QR Factorisation

Algorithms for computing the QR factorisation

- ▶ Gram-Schmidt orthogonalisation
- ▶ Householder reflections
- ▶ Givens rotations

All of these have strengths and weaknesses.
We next go through each of them.

QR Factorisation

Gram-Schmidt theorem

Let $q_1, \dots, q_k \in \mathbb{K}^n$ be orthonormal and $a_{k+1} \in \mathbb{K}^n$ be linearly independent of the q_k . Then,

$$\tilde{q}_{k+1} := a_{k+1} - \sum_{\ell=1}^k q_{\ell}^H a_{k+1} q_{\ell}$$

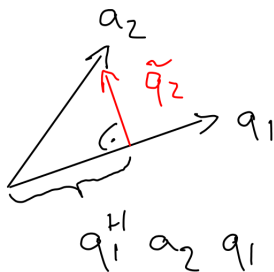
is orthogonal to q_1, \dots, q_k .

Proof.

$$q_{\ell}^H \tilde{q}_{k+1} = q_{\ell}^H a_{k+1} - q_{\ell}^H q_{\ell} q_{\ell}^H a_{k+1} = 0.$$

QR Factorisation

Gram-Schmidt theorem, pictorial version



QR Factorisation

QR factorisation via Gram-Schmidt theorem

Rewrite Gram-Schmidt formula

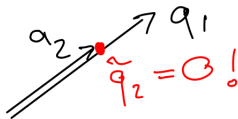
$$\tilde{q}_{k+1} = a_{k+1} - \sum_{\ell=1}^k q_{\ell}^H a_{k+1} q_{\ell}$$

in the form

$$a_{k+1} = \underbrace{\|\tilde{q}_{k+1}\|_2}_{R[k+1,k+1]} q_{k+1} + \sum_{\ell=1}^k \underbrace{q_{\ell}^H a_{k+1}}_{R[\ell,k+1]} q_{\ell}.$$

QR Factorisation

Breaking the Gram-Schmidt algorithm



QR Factorisation

Classical Gram-Schmidt

$$\tilde{q}_{k+1}^{(0)} := a_{k+1}, \quad \tilde{q}_{k+1}^{(\ell+1)} := \tilde{q}_{k+1}^{(\ell)} - q_{\ell}^H a_{k+1} q_{\ell}$$

Modified Gram-Schmidt

$$\tilde{q}_{k+1}^{(0)} := a_{k+1}, \quad \tilde{q}_{k+1}^{(\ell+1)} := \tilde{q}_{k+1}^{(\ell)} - q_{\ell}^H \tilde{q}_{k+1}^{(\ell)} q_{\ell}$$

Modified Gram-Schmidt is more numerically stable.

QR Factorisation

Example

Classical Gram-Schmidt:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & \varepsilon_{\text{mach}} \varepsilon^{-1} & \varepsilon_{\text{mach}} \varepsilon^{-1} \\ 1 & \varepsilon_{\text{mach}} \varepsilon^{-1} & \varepsilon_{\text{mach}} \varepsilon^{-1} \\ 0 & 1 & \varepsilon_{\text{mach}} \varepsilon^{-2} \\ 0 & 0 & 1 \end{pmatrix}$$

Modified Gram-Schmidt:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & \varepsilon_{\text{mach}} \varepsilon^{-1} & \varepsilon_{\text{mach}} \varepsilon^{-1} \\ 1 & \varepsilon_{\text{mach}} \varepsilon^{-1} & \varepsilon_{\text{mach}} \varepsilon^{-1} \\ 0 & 1 & \varepsilon_{\text{mach}}^2 \varepsilon^{-2} \\ 0 & 0 & 1 \end{pmatrix}$$

QR Factorisation

Householder reflection theorem

Assume $u \in \mathbb{K}^n$ such that $\|u\| = 1$. Then,

$$Q := I - 2uu^H$$

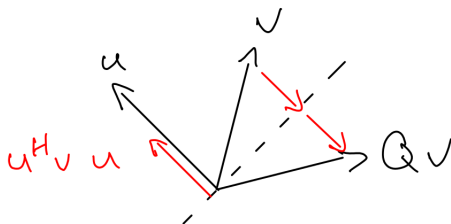
is an orthogonal matrix, and $Q^2 = I$.

Proof.

$$Q^H Q = Q^2 = (I - 2uu^H)(I - 2uu^H) = I - 4uu^H + 4u(u^H u)u^H = I.$$

QR Factorisation

Householder reflection theorem, pictorial version



QR Factorisation

QR factorisation via Householder reflections

Idea: choose u such that $Qa_1 = \pm\alpha e_1$. Then,

$$Q \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \end{pmatrix} = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix}.$$

How to find $Q = I - 2uu^H$?

QR Factorisation

Determining Householder u

- ▶ Reflections preserve length; hence $\alpha = \|a_1\|$.
- ▶ $a_1 - u u^H a_1 = \pm \|a_1\| e_1$ implies $u \propto a_1 \mp \|a_1\| e_1$.
- ▶ Choose $Qa_1 = -\text{sign}(a_{11}) \|a_1\| e_1$ to avoid cancellation in u .
 a_{11} denotes first component of a_1 .



QR Factorisation

Givens rotation

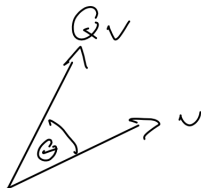
$$Q := \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \quad \text{represents a rotation by } \theta.$$

Example

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -a_1 \end{bmatrix}$$

QR Factorisation

Serious example



QR factorisation via Givens rotation

Similar idea as in Householder case:

$$\begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \end{pmatrix} \rightarrow \begin{pmatrix} x & x & x \\ 0 & x & x \\ x & x & x \end{pmatrix} \rightarrow \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix} \rightarrow \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{pmatrix}$$

QR Factorisation

QR factorisation in Julia

`F = qr(A)` computes “factorisation object”.

- ▶ Access factors through `F.Q`, `F.R`, `F.p` (vector) and `F.P` (matrix).
- ▶ `F.Q * F.U == A[:,F.p] == A * F.P`.
- ▶ `x = F\b` computes solution.

Computational cost of QR factorisation

$\mathcal{O}(mn \min\{m, n\})$ floating-point operations.

Stability of Householder QR factorisation

`Q,R` computed by Householder QR satisfy

$$QR = A + \delta A \quad \text{where} \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

Hence, Householder QR factorisation is backwards stable.

QR Factorisation

References and further reading

- ▶ G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press (1996),
- ▶ L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (1997),
- ▶ J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (1997),
doi:10.1137/1.9781611971446
- ▶ N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics (2002),
doi:10.1137/1.9780898718027