

ETH Lectures 401-0663-00L Numerical Methods for Computer Science
401-2673-00L Numerical Methods for CSE

Homework Problems and Projects

Prof. R. Hiptmair, SAM, ETH Zurich

Autumn Term 2021, Version of October 2, 2022
(C) Seminar für Angewandte Mathematik, ETH Zürich

[Link](#) to the current version of this homework collection

0.1 General Information

0.1.1 Weekly Homework Assignments

All problems will be published in this single “.pdf” file. Every week, we publish a list of problem numbers on the lecture Moodle page. These are the current assignments for that week. Details are given on the lecture Moodle page.

0.1.2 Importance of Homework

Homework assignments are **not** mandatory. However, it is **very important** that you constantly exercise with the material you learn. “Solving” the homework assignments one week before the main exam by looking at the solutions will likely result in failure.

We provide hints and solutions from the beginning. It is your responsibility to look at the solutions only if you are stuck with a difficult problem and you tried to find a solution for a sufficient amount of time.

Most homework problems come with an estimate for the time required for its solution. These times are what we expect a bright and well-prepared student to require for that problem during an exam.

0.1.3 Corrections and Grading of Assignments

Homework assignments will not be graded.

However, you may submit your handwritten solutions by handing them to your tutor directly or uploading them via the Moodle upload interface, see the lecture Moodle page for instructions. The assistants will

Problem 2-4: Cholesky and QR decomposition

This problem is about the Cholesky and QR decomposition and the relationship between them. The Cholesky decomposition is explained in [Lecture → § 2.8.0.13]. The QR decomposition is explained in [Lecture → Section 3.3.3]. Please study these topics before tackling this problem.

This problem relies on both [Lecture → § 2.8.0.13] and [Lecture → Section 3.3.3]

(2-4.a)  Show that, for every matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ such that $\text{rank}(\mathbf{A}) = n$, the product matrix $\mathbf{A}^\top \mathbf{A}$ admits a Cholesky decomposition.

HINT 1 for (2-4.a): For a Cholesky decomposition to be possible, show that $\mathbf{A}^\top \mathbf{A}$ is symmetric and positive definite. 

SOLUTION of (2-4.a):

$\mathbf{A}^\top \mathbf{A}$ has to be symmetric and positive definite.

$(\mathbf{A}^\top \mathbf{A})^\top = \mathbf{A}^\top (\mathbf{A}^\top)^\top = \mathbf{A}^\top \mathbf{A}$, which proves the symmetry.

$\text{rank}(\mathbf{A}) = n$ implies that $\mathbf{A}$ is injective. Hence, $\forall \mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} \neq \mathbf{0}$, it holds that $\mathbf{A}\mathbf{v} \neq \mathbf{0} \implies \mathbf{v}^\top \mathbf{A}^\top \mathbf{A} \mathbf{v} = |\mathbf{A}\mathbf{v}|_2^2 > 0$, which means that $\mathbf{A}^\top \mathbf{A}$ is positive definite.



(2-4.b)  The following C++ functions with EIGEN are given:

C++11-code 2.4.1: QR-decomposition via Cholesky decomposition

```

2 void CholeskyQR(const Eigen::MatrixXd& A, Eigen::MatrixXd& R,
3               Eigen::MatrixXd& Q) {
4     Eigen::MatrixXd AtA = A.transpose() * A;
5     Eigen::LLT<Eigen::MatrixXd> L = AtA.llt();
6     R = L.matrixL().transpose();
7     Q = R.transpose()
8         .triangularView<Eigen::Lower>()
9         .solve(A.transpose())
10        .transpose();
11    // .triangularView() template member only accesses the triangular
12    // part of a dense matrix and allows to easily solve linear problem
13 }

```

Get it on  GitLab (choleskyQR.hpp).

C++11-code 2.4.2: Standard way to do QR-decomposition in EIGEN

```

2 void DirectQR(const Eigen::MatrixXd& A, Eigen::MatrixXd& R,
3              Eigen::MatrixXd& Q) {
4     const size_t m = A.rows();
5     const size_t n = A.cols();
6
7     Eigen::HouseholderQR<Eigen::MatrixXd> QR = A.householderQr();
8     Q = QR.householderQ() * Eigen::MatrixXd::Identity(m, std::min(m, n));
9     R = Eigen::MatrixXd::Identity(std::min(m, n), m) *
10        QR.matrixQR().triangularView<Eigen::Upper>();

```

```

11 | // If A: m x n, then Q: m x m and R: m x n.
12 | // If m > n, however, the extra columns of Q and extra rows of R are
13 | // not needed. Matlab returns this "economy-size" format calling
14 | // "qr(A,0)", which does not compute these extra entries. With the code above, Eigen
15 | // is smart enough to not compute the discarded vectors.
16 | }

```

Get it on  GitLab (choleskyQR.hpp).

Prove that, for every matrix $\mathbf{A}$, CholeskyQR and DirectQR produce the *same* output matrices $\mathbf{Q}$ and $\mathbf{R}$, if there were no roundoff errors (Such functions are called **algebraically equivalent**).

HINT 1 for (2-4.b): If you test the code, ignore the signs of the entries in the resulting matrices $\mathbf{Q}$ and $\mathbf{R}$. The differences are due to the implementation of EIGEN. They are not however caused by the fact that EIGEN might use a permutation in the QR decomposition: the class we employ, `Eigen::HouseholderQR`, does not make use of them. (`Eigen::ColPivHouseholderQR` and `Eigen::FullPivHouseholderQR` do, on the other hand.)

Moreover, both functions only compute the same *economy-size* output of QR decompositions in MATLAB (`qr(A,0)`), which holds for $m > n$. Indeed, if $\mathbf{A} \in \mathbb{R}^{m,n}$, then orthogonal $\mathbf{Q}$ is a square matrix in $\mathbb{R}^{m,m}$ and upper triangular $\mathbf{R}$ is in $\mathbb{R}^{m,n}$. However, if $m > n$, we can obtain an equivalent QR decomposition for $\mathbf{A}$ by removing extra right columns of $\mathbf{Q}$ and extra bottom rows of $\mathbf{R}$, to end up with $\mathbf{Q} \in \mathbb{R}^{m,n}$ and $\mathbf{R} \in \mathbb{R}^{n,n}$. Both CholeskyQR and DirectQR only compute the entries which are needed. The former because it relies on a Cholesky decomposition of $\mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{n,n}$. The latter because the underlying EIGEN QR decomposition is already optimized when one imposes that $\mathbf{Q} \in \mathbb{R}^{m,n}$ and $\mathbf{R} \in \mathbb{R}^{n,n}$. $\lrcorner$

SOLUTION of (2-4.b):

For CholeskyQR to return the same $\mathbf{R}$ and $\mathbf{Q}$ of a QR decomposition using a Cholesky decomposition, we need to verify three properties:

- $\mathbf{R}$ is upper triangular.
- $\mathbf{Q}$ is orthogonal.
- $\mathbf{QR} = \mathbf{A}$

This is done in the following:

- $\mathbf{R}$ is upper triangular from definition of Cholesky decomposition.
- Given that $\mathbf{Q} = (\mathbf{R}^{-\top} \mathbf{A}^\top)^\top = \mathbf{AR}^{-1}$ and, from the Cholesky decomposition, $\mathbf{R}^\top \mathbf{R} = \mathbf{A}^\top \mathbf{A} \Rightarrow \mathbf{R}^{-\top} \mathbf{A}^\top \mathbf{AR}^{-1} = \mathbf{I}$, we have that $\mathbf{Q}^\top \mathbf{Q} = \mathbf{R}^{-\top} \mathbf{A}^\top \mathbf{AR}^{-1} = \mathbf{I}$. This means that $\mathbf{Q}$ is orthogonal.
- $\mathbf{QR} = (\mathbf{R}^{-\top} \mathbf{A}^\top)^\top \mathbf{R} = \mathbf{AR}^{-1} \mathbf{R} = \mathbf{A}$

▲

(2-4.c)  Let `EPS` denote the machine precision. Why does the function CholeskyQR from Subproblem (2-4.b) fail to return the correct result for $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ \frac{1}{2}\text{EPS} & 0 \\ 0 & \frac{1}{2}\text{EPS} \end{bmatrix}$?

HINT 1 for (2-4.c): Review [Lecture $\rightarrow$ Ex. 3.2.0.4]! $\lrcorner$

HINT 2 for (2-4.c): Are the hypothesis needed by the Cholesky decomposition satisfied? ┘

SOLUTION of (2-4.c):

$\mathbf{A}$ has rank 2, but $\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} 1 + \frac{1}{4}\epsilon^2 & 1 \\ 1 & 1 + \frac{1}{4}\epsilon^2 \end{bmatrix} \approx \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ has rank 1 in machine arithmetic. Hence, $\mathbf{A}^\top \mathbf{A}$ is symmetric but not positive definite: the hypothesis needed for the Cholesky decomposition are not satisfied.



End Problem 2-4