

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 2-9: Partitioned Matrix

Based on the block view of matrix multiplication presented in [Lecture → § 1.3.1.13], we looked at *block elimination* for the solution of block partitioned linear systems of equations in [Lecture → § 2.6.0.2]. Also of interest are [Lecture → Rem. 2.3.2.19] and [Lecture → Rem. 2.3.2.17] where LU-factorisation is viewed from a block perspective. Closely related to this problem is [Lecture → Ex. 2.6.0.5], which you should study again as warm-up to this problem.

This problem practises block Gaussian elimination and is related to [Lecture → Section 2.6]

Let the matrix $\mathbf{A} \in \mathbb{R}^{n+1, n+1}$ be block partitioned according to

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{u}^T & 0 \end{bmatrix}, \quad (2.9.1)$$

where $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^n$, and $\mathbf{R} \in \mathbb{R}^{n \times n}$ is *upper triangular* and *regular*.

(2-9.a) ☞ (10 min.) Give a necessary and sufficient condition for the *triangular* matrix $\mathbf{R}$ to be invertible.

HINT 1 for (2-9.a): Conceive a simple algorithm for solving a triangular linear system, and find out for which coefficient matrices it will run without break-down. These matrices must then be invertible.

Alternatively, you may appeal to the criteria of [Lecture → Thm. 2.2.1.4].

SOLUTION of (2-9.a):

There are many ways to come to the conclusion that all diagonal entries must be different from zero:

- The matrix $\mathbf{R}$ being upper triangular has determinant $\det(\mathbf{R}) = \prod_{i=1}^n (\mathbf{R})_{i,i}$, which implies that all the diagonal elements must be non-zero for $\mathbf{R}$ to be invertible.
- Alternatively, you may argue that backward elimination for solving the LSE $\mathbf{R}\mathbf{x} = \mathbf{b}$ will always produce a solution, if division by the diagonal elements of $\mathbf{R}$ can be carried out.
- It is easy to verify (how?) that all columns of $\mathbf{R}$ are linearly independent, if and only if the diagonal entries do not vanish.

▲

(2-9.b) ☞ (15 min.) Determine expressions for the sub-vector $\mathbf{z} \in \mathbb{R}^n$ and the last component $\xi \in \mathbb{R}$ of the solution vector of the linear system of equations

$$\begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{u}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \xi \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \beta \end{bmatrix} \quad (2.9.2)$$

for arbitrary $\mathbf{b} \in \mathbb{R}^n, \beta \in \mathbb{R}$.

Use *block Gaussian elimination* as presented in [Lecture → § 2.6.0.2].

SOLUTION of (2-9.b):

From the first n equations of (2.9.2) we deduce

$$\mathbf{z} = \mathbf{R}^{-1}(\mathbf{b} - \xi \mathbf{v}),$$

from which we find, using the bottom equation of (2.9.2)

$$\mathbf{u}^\top \mathbf{R}^{-1}(\mathbf{b} - \zeta \mathbf{v}) = \beta \Rightarrow \zeta = \frac{\beta - \mathbf{u}^\top \mathbf{R}^{-1} \mathbf{b}}{-\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v}},$$

assuming that the denominator does not vanish.

Alternatively, applying the block Gaussian elimination as in [Lecture → § 2.6.0.2], we obtain:

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \zeta \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{-1}(\mathbf{b} - \mathbf{v}s^{-1}b_s) \\ s^{-1}b_s \end{bmatrix}$$

with $s := -(\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v})$, $b_s := (\beta - \mathbf{u}^\top \mathbf{R}^{-1} \mathbf{b})$. Of course, we have to assume $s \neq 0$.

▲

(2-9.c) ☐ (10 min.) [depends on Sub-problem (2-9.b)]

Show that $\mathbf{A}$ is regular if and only if $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} \neq 0$.

SOLUTION of (2-9.c):

Recall that the matrix $\mathbf{R}$ is known to be invertible. The square matrix $\mathbf{A}$ is regular, if the corresponding linear system has a unique solution for every right hand side vector. In the previous sub-problem, it was shown that the linear system is equivalent to the equations

$$\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{b} - \zeta \mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} = \beta, \quad \mathbf{z} = \mathbf{R}^{-1}(\mathbf{b} - \zeta \mathbf{v}).$$

If $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} \neq 0$, then ζ is uniquely determined by the former equation, and $\mathbf{z}$ by the latter. If $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} = 0$ and $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{b} = \beta$, then the equations are satisfied for any ζ (the linear system has an infinite number of solutions). If $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} = 0$ and $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{b} \neq \beta$, then the former equation is not satisfied (the linear system has no solution). This proves that $\mathbf{A}$ is regular if and only if $\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} \neq 0$.

An alternative proof can be based on the **block LU decomposition** [Lecture → Rem. 2.3.2.19]:

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{u}^\top & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{u}^\top \mathbf{R}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{O} & -\mathbf{u}^\top \mathbf{R}^{-1} \mathbf{v} \end{bmatrix},$$

and the upper triangular factor is invertible, once the bottom-right diagonal element does not vanish, cf. Sub-problem (2-9.a).

▲

(2-9.d) ☐ (25 min.) [depends on Sub-problem (2-9.b)]

Implement the C++ function

```
void solveelse (
    const Eigen::MatrixXd & R,
    const Eigen::VectorXd & v,
    const Eigen::VectorXd & u,
    const Eigen::VectorXd & bb,
    Eigen::VectorXd & x);
```

for the efficient computation of the solution of $\mathbf{Ax} = \mathbf{b}$ (with $\mathbf{A}$ as in (2.9.1)), which is to be returned in the argument $\mathbf{x}$. Perform size checks on input matrices and vectors.

Use the decomposition from Sub-problem (2-9.b). You can rely on the `triangularView()` function to inform EIGEN of the triangular structure of $\mathbf{R}$, see [Lecture → Code 1.2.1.6].

SOLUTION of (2-9.d):

It is important to communicate to EIGEN that $\mathbf{R}$ is upper triangular so that the library knows that it can skip the costly elimination step.

C++11-code 2.9.3: Block Gauss based solution

```

2 void solveElse(const Eigen::MatrixXd& R, const Eigen::VectorXd& v,
3               const Eigen::VectorXd& u, const Eigen::VectorXd& bb,
4               Eigen::VectorXd& x) {
5     // size of R, which is size of u, v, and size of bb is n+1
6     const unsigned int n = R.rows();
7
8     // TODO: (2-9.d)
9     // i) Use assert() to check that R, v, u, bb all have the
10    // appropriate sizes.
11    // ii) Use (3-9.b) to solve the LSE.
12    // Hint: Use R.triangularView<Eigen::Upper>() to make use of the
13    // triangular
14    // structure of R.
15    // START
16    assert(n == R.cols() && n == u.size() && n == v.size() &&
17           n + 1 == bb.size() && "Size mismatch!");
18
19    const Eigen::TriangularView<const Eigen::MatrixXd, Eigen::Upper>& triR =
20        R.triangularView<Eigen::Upper>();
21
22    // s is the Schur complement and, in this case, is a scalar
23    // also b_s is a scalar
24    // snv = s^{-1}, b_s as in lecture notes
25    // sinvbs = s^{-1} * b_s
26    const double sinv = -1. / u.dot(triR.solve(v));
27    const double bs = (bb(n) - u.dot(triR.solve(bb.head(n))));
28    const double sinvbs = sinv * bs;
29
30    // Stack the vector (z, \xi)^T =: x
31    x = Eigen::VectorXd::Zero(n + 1);
32    x << triR.solve(bb.head(n) - v * sinvbs), sinvbs;
33    // END
34 }

```

Get it on  GitLab (blockdecomp.hpp).



(2-9.e)  (20 min.) [depends on Sub-problem (2-9.d)]

Code a C++ function

```

bool testSolveLSE(const Eigen::MatrixXd & R,
                  const Eigen::VectorXd & v, const Eigen::VectorXd & u,
                  const Eigen::VectorXd & b, Eigen::VectorXd & x);

```

that tests your implementation of `solveLSE` by comparison with a standard LU-solver provided by EIGEN. Return **true**, if the results “are the same”.

HINT 1 for (2-9.e): In order to obtain information about direct elimination solvers available in EIGEN consult  [EIGEN documentation](#). ┘

HINT 2 for (2-9.e): Heed the warning of [Lecture → Rem. 1.5.3.15]: Never test whether a computed results of real type agrees exactly with zero. ┘

SOLUTION of (2-9.e):

C++11-code 2.9.4: LU-decomposition based solution

```

2  bool testSolveLSE(const Eigen::MatrixXd& R, const Eigen::VectorXd& v,
3                  const Eigen::VectorXd& u, const Eigen::VectorXd& b,
4                  Eigen::VectorXd& x) {
5      bool areTheSame = false;
6
7      // TODO: (2-9.e)
8      // i) Create the system matrix A and solve the LSE, using
9      // an Eigen LU-solver. Store the solution in x.
10     // ii) Solve the LSE with solveLSE(), and calculate the
11     // difference between this solution and x. Return true
12     // if and only if the norm of this difference is close
13     // enough to zero.
14     // START
15     // size of R, which is size of u, v, and size of bb is n+1
16     const unsigned int n = R.rows();
17     assert(n == R.cols() && n == u.size() && n == v.size() && n + 1 == b.size() &&
18            "Size mismatch!");
19
20     Eigen::MatrixXd A(n + 1, n + 1);
21     // Build matrix using blocks
22     A << R, v, u.transpose(), 0;
23
24     // Solve system using LU decomposition
25     x = A.partialPivLu().solve(b);
26
27     // Solve system using solveLSE()
28     Eigen::VectorXd y;
29     solveLSE(R, v, u, b, y);
30
31     // We choose an acceptable tolerance.
32     constexpr double tolerance = 1e-8;
33     areTheSame = ((x - y).norm() < tolerance * b.norm());
34
35     // END
36     return areTheSame;
37 }

```

Get it on  [GitLab \(blockdecomp.hpp\)](#).



(2-9.f)  (10 min.) [depends on Sub-problem (2-9.d)]

What is the asymptotic complexity of your implementation of `solveLSE()` in terms of problem size parameter $n \rightarrow \infty$?

SOLUTION of (2-9.f):

The complexity is $O(n^2)$. The backward substitution for $\mathbf{R}^{-1}\mathbf{x}$ is $O(n^2)$, vector dot product and subtraction of vectors is $O(n)$, so that the complexity is dominated by the backward substitution: $O(n^2)$.



End Problem 2-9, 90 min.