

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

Chapter 2

Direct Methods for Linear Systems of Equations

Problem 2-1: Solving a small linear system

In this problem, you will use EIGEN to solve a simple linear algebra exercise.

This problem is meant to practise the solution of linear systems of equations in EIGEN, see [Lecture → § 2.5.0.4].

We have the following information: Daniel buys 3 mangoes, 1 kiwi, 7 lychees and 2 bananas. Peter buys 2 mangoes and 1 pineapple. Julia needs 3 pomegranates and 1 mango for her fruit juice, whereas Ralf needs 5 kiwis and 1 banana for his fruit salad. Marcus buys 1 pineapple and 2 bananas. Manfred buys 20 lychees and 1 kiwi. Daniel pays CHF 11.10, Peter pays CHF 17.00 (and so spends all his pocket money), Julia pays CHF 6.10, Ralf pays CHF 5.25, Marcus pays CHF 12.50 and Manfred pays CHF 7.00.

(2-1.a)  Study [Lecture → § 2.5.0.8] and  [EIGEN documentation](#) to learn how to solve linear systems using direct elimination solvers of EIGEN. 

(2-1.b) Write a C++ function

```
Eigen::VectorXd fruitPrice();
```

that returns the prices of the fruits in the following order:

Mango – Kiwi – Lychee – Banana – Pomegranate – Pineapple

SOLUTION of (2-1.b):

From the information given above we have to extract a linear system of equations, whose solution gives the prices of the fruits. This linear system is evident from Code 2.1.1.

The following code uses a particular of the many direct solvers provided by EIGEN. You may use most other from the table in  [EIGEN documentation](#), especially **Eigen::PartialPivLU**, which is the standard choice.

C++ code 2.1.1: Code

```
2 Eigen::VectorXd fruitPrice() {  
3     Eigen::VectorXd x = Eigen::VectorXd::Zero(6);  
4 }
```

```

5  // TODO: (2-1.b) Determine the fruit prices using Eigen.
6  // START
7  // Initialize the matrix
8  Eigen::MatrixXd A(6, 6);
9  // See "Advanced Initialization" in the Eigen doc
10 A << 3, 1, 7, 2, 0, 0, 2, 0, 0, 0, 0, 1, 1, 0, 0, 0, 3, 0, 0, 5, 0, 1, 0, 0,
11      0, 0, 0, 2, 0, 1, 0, 1, 20, 0, 0, 0;
12
13 // Initialize the vector
14 Eigen::VectorXd b(6);
15 b << 11.10, 17.00, 6.10, 5.25, 12.50, 7.00;
16
17 // Initialize the solver; other decompositions also possible
18 Eigen::ColPivHouseholderQR<Eigen::MatrixXd> solver;
19 // Decompose A
20 solver.compute(A);
21 // Solve the LSE
22 x = solver.solve(b);
23
24 // Notice how we split up initialization, decomposition and solving.
25 // This means we can use the same solver for multiple matrices and
26 // the same decompositions for multiple LSE with the same matrix.
27 // These three steps can be combined:
28 Eigen::VectorXd x2 = A.colPivHouseholderQr().solve(b);
29
30 // The result is the same:
31 assert((x - x2).norm() < 1e-10);
32 // END
33 return x;
34 }

```

Get it on  GitLab (fruits.hpp).



End Problem 2-1