

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 2-20: Matrix-Vector Product in CRS format

Sparse matrices have to be stored in special formats in order to save memory and inform algorithms about the position of non-zero entries. This problem will examine the CRS format

This problem is related to [Lecture → § 2.7.1.4] and assumes familiarity with C++.

The following data structure is used to store an $m \times n$ -matrix in compressed row-storage (CRS) format:

```
struct CRSMatrix {
    unsigned int m;           // number of rows
    unsigned int n;           // number of columns
    std::vector<double> val;   // value array
    std::vector<unsigned int> col_ind; // same length as value array
    std::vector<unsigned int> row_ptr; // length m+1, row_ptr[m] ==
    val.size()
};
```

This format is defined by the relationship ("C++ indexing")

$$\text{val}[k] = a_{ij} \Leftrightarrow \begin{cases} \text{col_ind}[k] = j, \\ \text{row_ptr}[i] \leq k < \text{row_ptr}[i+1], \end{cases} \quad 0 \leq k \leq \text{val.size()},$$

for $i \in \{0, \dots, m-1\}, j \in \{0, \dots, n-1\}$.

(2-20.a) (40 min.) The function `crsmv` is supposed to return the product of a matrix in CRS format passed through `M` and of a vector given as argument `x`. Supplement the missing parts of the following listing code by writing valid C++ code into the boxes.

```
template <typename VECTORTYPE_I, typename VECTORTYPE_II>
VECTORTYPE_I crsmv(const CRSMatrix &M,
                  const VECTORTYPE_II &x) {
    assert((x.size() == M. )
           && "Size mismatch between x and M");
    VECTORTYPE_I y(M.m);
    for (int k = 0; k < ; ++k) {
        y[k] = 0;
        for (int j = M. ; j < ; ++j) {
            y[ ] += M. * x[ ];
        }
    }
    return y;
}
```

SOLUTION of (2-20.a):

The code processes the matrix row-wise (index variable `k`) and, thus, sequentially runs through the `val` array and the `col_ind` array (index variable `j`).

C++ code 2.20.1: Function `crsmv()`

```
2  template <typename VECTORTYPE_I, typename VECTORTYPE_II>
3  VECTORTYPE_I crsmv(const CRSMatrix &M, const VECTORTYPE_II &x) {
4      assert((x.size() == M.n) && "Size mismatch between x and M");
5      VECTORTYPE_I y(M.m);
6      for (int k = 0; k < M.m; ++k) {
7          y[k] = 0;
8          for (int j = M.row_ptr[k]; j < M.row_ptr[k + 1]; ++j) {
9              y[k] += M.val[j] * x[M.col_ind[j]];
10         }
11     }
12     return y;
13 }
```

**End Problem 2-20 , 40 min.**