

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 3

Direct Methods for Linear Least Squares Problems

Problem 3-1: Hidden linear regression

This problem is about a hidden linear regression: in fact, at first glance it will seem that we are dealing with a nonlinear system of equations. However, we will be able to reduce the system to a linear form.

Linear regression was introduced in [Lecture → Ex. 3.0.1.1] and [Lecture → Ex. 3.0.1.4]. You have to be familiar with normal equations, see [Lecture → Section 3.1.2] and, in particular, [Lecture → Ex. 3.1.2.4]. This problem also addresses the solution of linear least-squares problems in EIGEN, see [Lecture → Code 3.2.0.1] and [Lecture → Code 3.3.4.2].

We consider the function $f(t) = \alpha e^{\beta t}$ with unknown parameters $\alpha > 0$, $\beta \in \mathbb{R}$. Given are measurements (t_i, y_i) , $i = 1, \dots, n$, $2 < n \in \mathbb{N}$. We want to determine $\alpha, \beta \in \mathbb{R}$ such that $f(t)$ fulfills the following condition “to the best of its ability” (in the least square sense):

$$f(t_i) = y_i, \quad \text{with } y_i > 0, i = 1, \dots, n. \quad (3.1.1)$$

(3-1.a)  (10 min.) Which overdetermined system of *nonlinear* equations directly stems from (3.1.1)?

SOLUTION of (3-1.a):

The resulting system of nonlinear equations is:

$$y_i = f(t_i) = \alpha e^{\beta t_i}, \quad i = 1, \dots, n.$$

The non-linearity is due to the occurrence of parameter β as exponent of e .



(3-1.b)  (15 min.) In which overdetermined system of *linear* equations can you transform the nonlinear system derived in (3-1.a)?

HINT 1 for (3-1.b): Use the inverse function of the exponential.



SOLUTION of (3-1.b):

Define $g(t) := \log f(t)$. This is well defined, as $f(t) > 0 \forall t$.

With $a := \log \alpha$ we can then obtain:

$$g(t) = \log f(t) = \log \alpha + \log e^{\beta t} = a + \beta t.$$

Moreover, we impose $b_i := \log y_i$.

The new problem is therefore expressed by the following **overdetermined linear system**:

$$\mathbf{Ax} = \mathbf{b}, \quad \text{with } \mathbf{A} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_n \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} a \\ \beta \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \log y_1 \\ \log y_2 \\ \vdots \\ \log y_n \end{bmatrix}.$$

▲

(3-1.c) 🕒 (15 min.) Determine the **normal equations** for the overdetermined linear system of (3-1.b).

HINT 1 for (3-1.c): Apply the definition of normal equations given in [Lecture → Section 3.1.2]. ┐

SOLUTION of (3-1.c):

We determine the parameters of the normal equations $\mathbf{A}^\top \mathbf{Ax} = \mathbf{A}^\top \mathbf{b}$. We obtain:

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} n & \sum_{i=1}^n t_i \\ \sum_{i=1}^n t_i & \sum_{i=1}^n t_i^2 \end{bmatrix}, \quad \mathbf{A}^\top \mathbf{b} = \begin{bmatrix} \sum_{i=1}^n b_i \\ \sum_{i=1}^n t_i b_i \end{bmatrix}. \quad (3.1.2)$$

Once we have found the least squares solution $\mathbf{x} = \begin{bmatrix} a \\ \beta \end{bmatrix}$, we just need to replace $\alpha = e^a$ and β in function f .

▲

(3-1.d) 🕒 (20 min.) Consider the data $(t_i, y_i) \in \mathbb{R}^2, i = 1, \dots, m$, stored in two vectors $\mathbf{t}$ and $\mathbf{y}$ of equal length. In the file `adaptedlinreg.hpp` implement a C++ function

```
Eigen::VectorXd linReg(const Eigen::VectorXd &t, const
Eigen::VectorXd &y);
```

that uses the **method of normal equations** discussed in [Lecture → Section 3.2] to solve the 1D linear regression problem as described in [Lecture → Ex. 3.0.1.1] in least-squares sense. This function should return the estimated parameters α and β in a vector of length 2.

SOLUTION of (3-1.d):

The solution is a straightforward adaptation of [Lecture → Code 3.2.0.1]. We start with initializing the matrix $\mathbf{A}$. Alternatively, we could just initialize $\mathbf{A}^\top \mathbf{A} \in \mathbb{R}^{2,2}$ as given in (3.1.2).

C++ code 3.1.3: Solution of (3-1.d)

```
2 Eigen::VectorXd linReg(const Eigen::VectorXd &t, const Eigen::VectorXd &y) {
```

```

3 | // TODO: (3-1.d) Use the method of normal equations to solve the 1D
  | linear
4 | // regression problem in least-square sense.
5 | // Note: The tests anticipate the outputs in the order (alpha,
  | beta), and not
6 | // (beta, alpha).
7 |
8 | // START
9 | assert(t.size() == y.size() && "t and y must have same size");
10 |
11 | // Coefficient matrix of overdetermined linear system.
12 | Eigen::MatrixXd A(t.size(), 2);
13 | A.col(0).setOnes();
14 | A.col(1) = t;
15 |
16 | // Normal equation
17 | Eigen::MatrixXd lhs = A.transpose() * A; // Left-hand side
18 | Eigen::VectorXd rhs = A.transpose() * y; // Right-hand side
19 |
20 | // Intercept and slope.
21 | Eigen::VectorXd x = lhs.lu().solve(rhs);
22 |
23 | // END
24 |
25 | return x;
26 | }

```

Get it on  GitLab (adaptedlinreg.hpp).



(3-1.e)  (15 min.) [depends on Sub-problem (3-1.d), Sub-problem (3-1.a)]

Now consider the data (t_i, y_i) stored in two vectors $\mathbf{t}$ and $\mathbf{f}$ of equal length. Implement a C++ function

```

Eigen::VectorXd expFit(const Eigen::VectorXd &t, const
Eigen::VectorXd &y);

```

that returns a least squares estimate of α and β , given the nonlinear relationship (3.1.1) between $\mathbf{t}$ and $\mathbf{f}$:

HINT 1 for (3-1.e): Use the function `linReg` from (3-1.d). Take care to give the correct input $\mathbf{y}$ to `linReg` and to extract the right parameters α and β from its output.

SOLUTION of (3-1.e):

C++ code 3.1.4: Solution of (3-1.e)

```

2 | Eigen::VectorXd expFit(const Eigen::VectorXd &t, const Eigen::VectorXd &y) {
3 | // TODO: (3-1.e) Implement least square estimate of alpha and beta
  | using
4 | // the previously implemented function linReg().
5 | // Note: You don't need to have implemented linReg() to solve this
  | subproblem.
6 | // The tests will compile and use the master solution for linReg().
7 | // Note: The tests anticipate the outputs in the order (alpha,
  | beta), and not
8 | // (beta, alpha).

```

```
9
10 // START
11 assert(t.size() == y.size() && "t and y must have same size");
12
13 // Transform to a linear equation
14 // b := log(y) = x(0) + x(1)*t
15 // We use array() for coefficient-wise log() and matrix() to
16 // transform it
17 // back to a vector.
18 Eigen::VectorXd b = y.array().log().matrix();
19
20 // Solve and transform back to the nonlinear equation
21 // y = exp(x(0)) * exp(x(1)*t) = alpha*exp(beta*t)
22 Eigen::VectorXd x = linReg(t, b);
23 x(0) = exp(x(0)); // Eigen's coefficient-wise exp()
24
25 // END
26
27 return x;
}
```

Get it on  [GitLab](#) (adaptedlinreg.hpp).



End Problem 3-1 , 75 min.