

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 8-5: Order- p convergent iterations

In [Lecture → Section 8.2.2] we investigated the speed of convergence of iterative methods for the solution of a general non-linear problem $F(\mathbf{x}) = 0$ and introduced the notion of **convergence of order $p \geq 1$** , see [Lecture → Def. 8.2.2.10]. This problem highlights the fact that for $p > 1$ convergence may not be guaranteed, even if the error norm estimate of [Lecture → Def. 8.2.2.10] may hold for some $\mathbf{x}^* \in \mathbb{R}^n$ and all iterates $\mathbf{x}^{(k)} \in \mathbb{R}^n$.

Problem with a theoretical focus and a little C++ coding. Relies on the concepts introduced in [Lecture → Section 8.2.2]

Given $\mathbf{x}^* \in \mathbb{R}^n$, suppose that a sequence $\mathbf{x}^{(k)}$ satisfies [Lecture → Def. 8.2.2.10]:

$$\exists C > 0: \quad \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq C \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p \quad \forall k \quad \text{and} \quad p > 1. \quad (8.5.1)$$

(8-5.a) 🕒 (20 min.) Determine $\epsilon_0 > 0$ as large as possible such that we have

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\| \leq \epsilon_0 \quad \implies \quad \lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*. \quad (8.5.2)$$

In other words, ϵ_0 tells us which distance of the initial guess from $\mathbf{x}^*$ still guarantees **local convergence**.

HINT 1 for (8-5.a): Study the estimates of [Lecture → § 8.4.3.4].

SOLUTION of (8-5.a):

We have to ensure that the norm of the iteration error tends to zero:

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^* \iff \lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0.$$

Thus, we seek an upper bound $B(k)$ for $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|$:

$$\begin{aligned} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| &\leq C \|\mathbf{x}^{(k-1)} - \mathbf{x}^*\|^p \\ &\leq C \cdot C^p \|\mathbf{x}^{(k-2)} - \mathbf{x}^*\|^{p^2} \\ &\leq C \cdot C^p \cdot C^{p^2} \|\mathbf{x}^{(k-3)} - \mathbf{x}^*\|^{p^3} \\ &\vdots \\ &\leq C \cdots C^{p^{k-1}} \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^{p^k} \\ &= C^{\sum_{i=0}^{k-1} p^i} \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^{p^k} \\ &\stackrel{\text{geom. series}}{=} C^{\frac{p^k - 1}{p-1}} \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^{p^k} \\ &\leq C^{\frac{p^k - 1}{p-1}} \epsilon_0^{p^k} = \underbrace{C^{\frac{1}{1-p}}}_{\text{const.}} \cdot \left(C^{\frac{1}{p-1}} \epsilon_0 \right)^{p^k} = B(k). \end{aligned}$$

For what values of ϵ_0 will we have $\lim_{k \rightarrow \infty} B(k) = 0$? We can read this off the expression we found for $B(k)$:

$$\lim_{k \rightarrow \infty} B(k) = 0 \iff C^{\frac{1}{p-1}} \epsilon_0 < 1 \implies 0 < \epsilon_0 < C^{\frac{1}{1-p}}. \quad (8.5.3)$$

See also [Lecture → Eq. (8.4.3.9)].



(8-5.b) (15 min.) Provided that $\|\mathbf{x}^{(0)} - \mathbf{x}^*\| < \epsilon_0$ is satisfied with ϵ_0 from Sub-problem (8-5.a), determine the minimal iteration step $k_{\min} = k_{\min}(\epsilon_0, C, p, \tau)$ such that $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < \tau$.

SOLUTION of (8-5.b):

Using the previous upper bound and the condition τ , we obtain: $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq C^{\frac{1}{1-p}} \cdot \left(C^{\frac{1}{p-1}} \epsilon_0\right)^{p^k} < \tau$. Solving for the minimal k (and calling the solution $k_{\min}$), with the additional requirement that $k \in \mathbb{N}$, we obtain:

$$\begin{aligned} \ln(C^{\frac{1}{1-p}}) + \underbrace{p^k \cdot \ln(C^{\frac{1}{p-1}} \epsilon_0)}_{<0} &< \ln \tau, \\ \Rightarrow k &> \ln \left(\frac{\ln(\tau) + \frac{1}{p-1} \ln(C)}{\ln(C^{\frac{1}{p-1}} \epsilon_0)} \right) \cdot \frac{1}{\ln(p)}, \\ \Rightarrow k_{\min} &= \left\lceil \ln \left(\frac{\ln(\tau) + \frac{1}{p-1} \ln(C)}{\ln(C^{\frac{1}{p-1}} \epsilon_0)} \right) \cdot \frac{1}{\ln(p)} \right\rceil \in \mathbb{N}, \end{aligned} \quad (8.5.4)$$

where $\lceil \cdot \rceil$ stands for the next larger integer. A similar, slightly simplified formula is [Lecture → Eq. (8.4.3.10)].



(8-5.c) (30 min.) Implement a C++ function

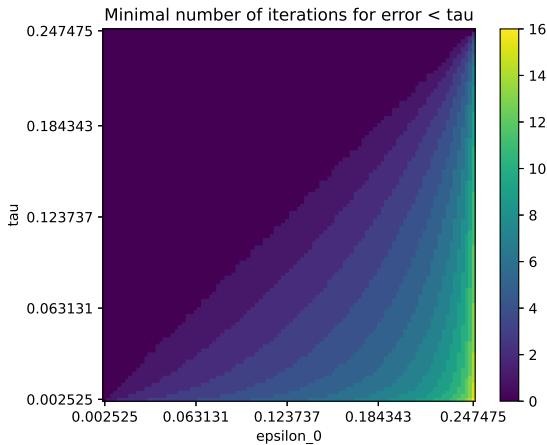
```
void kminplot(void);
```

that uses `MATPLOTLIBCPP`'s `plt::imshow()` to create a image of $k_{\min}(\epsilon_0, \tau)$ for the values $p = 1.5$, $C = 2$ and

$$\epsilon_0 \in \left[0, C^{\frac{1}{1-p}}\right]^2, \quad \tau \in [10^{-5}, 10^{-1}].$$

Use a logarithmic axis scale for τ .

SOLUTION of (8-5.c):



◁ Image of the minimum number $k_{\min}$ of iteration according to (8.5.4) as a function of ϵ_0 and the error reduction factor τ .

We see that the number of iterations increases only mildly as τ is reduced, but rather strongly depends on ϵ_0 as this values approaches the threshold $C^{\frac{1}{1-p}}$ found in Sub-problem (8-5.a).

A similar plot is given in [Lecture → Fig. 300].

Fig. 89

C++-code 8.5.5: Solution of Sub-problem (8-5.c)

```

2 void kminplot() {
3     constexpr double p = 1.5;
4     constexpr double C = 2.;
5     const double eps_max = std::pow(C, 1. / (1. - p));
6     constexpr unsigned int ngp = 100; // number of grid points
7
8     plt::figure();
9
10    // TODO: (8-5.c) Plot kmin using matplotlibcpp. To that end, start
11    // by creating
12    // a mesh using the provided meshgrid function in meshgrid.hpp.
13    // START
14    Eigen::VectorXd eps_lin = Eigen::VectorXd::LinSpaced(ngp, 0, eps_max);
15    Eigen::VectorXd tau_lin = eps_lin;
16    // we need an open interval, i.e. remove first and last element
17    Eigen::VectorXd eps_lin_seg = eps_lin.segment(1, eps_lin.size() - 2);
18    Eigen::VectorXd tau_lin_seg = tau_lin.segment(1, tau_lin.size() - 2);
19    Eigen::MatrixXd eps_msh, tau_msh;
20    meshgrid(eps_lin_seg, tau_lin_seg, eps_msh, tau_msh);
21
22    // apply formula to mesh
23    Eigen::MatrixXd kmin;
24    Eigen::MatrixXd num = tau_msh.array().log();
25    num += (1. / (p - 1.)) * std::log(C) *
26           Eigen::MatrixXd::Ones(tau_msh.rows(), tau_msh.cols());
27    Eigen::MatrixXd den = (eps_msh * std::pow(C, 1. / (p - 1.))).array().log();
28    kmin = (num.cwiseQuotient(den)).array().log() / std::log(p);
29
30    // Consider only gridpoints where eps is larger than tau
31    for (unsigned int i = 0; i < kmin.rows(); ++i) {
32        for (unsigned int j = 0; j < kmin.cols(); ++j) {
33            kmin(i, j) = std::ceil(kmin(i, j));
34            if (i > j) {
35                kmin(i, j) = 0;
36            }
37        }
38    }
39
40    std::vector<double> ticks(5);
41    std::vector<std::string> xlabels(5), ylabels(5);
42    for (unsigned int i = 0; i < 5; ++i) {
43        ticks[i] = i * (ngp - 3) / 4;
44        Eigen::Index id = (Eigen::Index) ticks[i];

```

```
44     xlabel[i] = std::to_string(eps_lin_seg(id));
45     ylabel[i] = std::to_string(tau_lin_seg(id));
46 }
47
48 plt::imshow(kmin, {{ "cmap", "viridis"}, {"origin", "lower"}});
49 plt::colorbar();
50 plt::title("Minimal number of iterations for error < tau");
51 plt::xlabel("epsilon_0");
52 plt::ylabel("tau");
53 plt::xticks(ticks, xlabel);
54 plt::yticks(ticks, ylabel);
55
56 // END
57 plt::savefig("./cx_out/k_min.png");
58 }
```

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



End Problem 8-5, 65 min.