

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 6-5: Chebychev interpolation of analytic functions

This problem concerns Chebychev interpolation as discussed in [Lecture → Section 6.2.3] and takes a close look at the interpolation error estimates derived for analytic interpolands in [Lecture → Section 6.2.2.3].

This problem is closely connected to [Lecture → Rem. 6.2.3.26] and involves implementation in C++ based on EIGEN. You are expected to be able to compute with complex numbers.

Using techniques from complex analysis, notably the residue theorem [Lecture → Thm. 6.2.2.50], in class we derived an expression for the interpolation error [Lecture → Eq. (6.2.2.57)] and from it an error bound [Lecture → Eq. (6.2.2.58)], as much sharper alternative to [Lecture → Thm. 6.2.2.15] and [Lecture → Lemma 6.2.2.20] for *analytic* interpolands. The bound tells us that for all $t \in [a, b]$

$$|f(t) - L_{\mathcal{T}}f(t)| \leq \left| \frac{w(x)}{2\pi i} \int_{\gamma} \frac{f(z)}{(z-t)w(z)} dz \right| \leq \frac{|\gamma| \max_{a \leq \tau \leq b} |w(\tau)| \max_{z \in \gamma} |f(z)|}{2\pi \min_{z \in \gamma} |w(z)| d([a, b], \gamma)},$$

where $d([a, b], \gamma)$ is the geometric distance of the integration contour $\gamma \subset \mathbb{C}$ from the interval $[a, b] \subset \mathbb{C}$ in the complex plane. The contour γ must be contractible in the domain D of analyticity of f and must wind around $[a, b]$ exactly once, see [Lecture → Fig. 212].

Now we consider the interval $[-1, 1]$. Following [Lecture → Rem. 6.2.3.26], our task is to find an upper bound for this expression, in the case where f possesses an analytical extension to a complex neighbourhood of $[-1, 1]$.

For the analysis of the Chebychev interpolation of analytic functions we used the elliptical contours, see [Lecture → Fig. 226],

$$\gamma_{\rho}(\theta) := \cos(\theta - i \log(\rho)), \quad \forall 0 \leq \theta \leq 2\pi, \quad \rho > 1. \quad (6.5.1)$$

(6-5.a)  (20 min.) Implement a C++ function

double lengthEllipse(double rho, unsigned int N);

that approximates the length of the ellipse γ_{ρ} by sampling (6.5.1) for $N \in \mathbb{N}$ (equidistant) arguments θ and returns the result.

SOLUTION of (6-5.a):

The ellipse is approximated by a polygon, whose length is considered an approximation of the length of the ellipse. Recall that $|w - z|$ gives the distance of two complex numbers $w, z \in \mathbb{C}$ considered as points in the plane $\mathbb{C} \cong \mathbb{R}^2$.

C++11-code 6.5.2: Code for lengthEllipse.

```

2 double lengthEllipse(double rho, unsigned int N) {
3     double length = 0.0;
4     // TODO: (6-5.a) Approximate the length of ellipse given rho and N
5     // equidistant points (in theta)
6     // START
7     const complex i(0, 1); // imaginary unit
8     // Define gamma_rho
9     auto f = [rho, i](double theta) { return cos(theta - i * log(rho)); };
10    Eigen::VectorXd eval_points = Eigen::VectorXd::LinSpaced(N + 1, 0, 2 * M_PI);

```

```

11 | // Calculating distance between last and current point on ellipse
12 | // for all points
13 | complex last = f(eval_points(0));
14 | for (unsigned int i = 0; i < N; ++i) {
15 |     complex current = f(eval_points(i + 1));
16 |     length += std::abs(current - last);
17 |     last = current;
18 | }
19 |
20 | // Alternative method using an approximation over the integral of
21 | // the length
22 | // of  $\gamma'_p$ .
23 | //  $df = |\gamma'_p(\theta)|$ 
24 | double length_alt = 0.0;
25 | auto df = [rho](long double theta) {
26 |     return 0.5 * sqrt(rho * rho + 1 / (rho * rho) - 2 * cos(2 * theta));
27 | };
28 | double dtheta = 2 * M_PI / N;
29 | double theta_i = 0;
30 | for (unsigned int i = 0; i < N; ++i) {
31 |     length_alt += df(theta_i);
32 |     theta_i += dtheta;
33 | }
34 | length_alt *= dtheta;
35 | // END
36 | return length;

```

Get it on  GitLab (chebyApprox.hpp).



From now on we consider the S -curve function (the logistic function):

$$f(t) := \frac{1}{1 + e^{-3t}}, \quad t \in \mathbb{R} \text{ } \textit{bdot} \quad (6.5.3)$$

(6-5.b)  (20 min.) Determine the maximal domain of analyticity of the extension of f to the complex plane $\mathbb{C}$.

HINT 1 for (6-5.b): Consult [Lecture $\rightarrow$ Rem. 6.2.2.67].

SOLUTION of (6-5.b):

f is analytic in

$$D := \mathbb{C} \setminus \left\{ \frac{2}{3}\pi i k - \frac{1}{3}\pi i \mid k \in \mathbb{Z} \right\}.$$

In fact, $g(t) := 1 + \exp(-3t)$ is an entire function, whereas $h(x) := \frac{1}{x}$ is analytic in $\mathbb{C} \setminus \{0\}$. Therefore, using [Lecture $\rightarrow$ Thm. 6.2.2.68], f is analytic in

$$\mathbb{C} \setminus \{z \in \mathbb{C} \mid g(z) = 0\} =: \mathbb{C} \setminus S,$$

where $S \subset \mathbb{C}$ is the “domain of singularity”. To find S let $z := a + ib$, $a, b \in \mathbb{R}$ and compute

$$\begin{aligned} -1 = \exp(z) &= \exp(a + ib) = \exp(a) \exp(ib) && \Leftrightarrow a = 0, b \in 2\pi\mathbb{Z} + \pi \\ -1 = \exp(z) &&& \Leftrightarrow z \in i(2\pi\mathbb{Z} + \pi) \\ -1 = \exp(-3z) &&& \Leftrightarrow z \in \frac{i(2\pi\mathbb{Z} - \pi)}{3}. \end{aligned}$$



(6-5.c) 🕒 (15 min.) [depends on Sub-problem (6-5.b)]

Find the least upper bound for the set of ellipse parameters

$$\{\rho > 1 : \gamma_\rho \subset D\},$$

where $D \subset \mathbb{C}$ is the domain of analyticity found in Sub-problem (6-5.b).

HINT 1 for (6-5.c): In [Lecture → Rem. 6.2.3.26] you can find alternative formulas for γ_ρ .

SOLUTION of (6-5.c):

The ellipse must be restricted such that the minor axis has length $\leq 2\pi/3$ (2 times the smallest point, in absolute value, where f is not-analytic). Since this corresponds to the imaginary part of $\gamma_\delta(\theta)$, when $\theta = \pi/2$, we find:

$$\begin{aligned} \cos(\pi/2 - i \log(\rho_{\max})) &= 1/3\pi i \\ \Updownarrow \\ \sinh(\log(\rho_{\max})) &= \pi/3 \Leftrightarrow \rho_{\max} = \exp(\sinh^{-1}(\pi/3)). \end{aligned}$$



(6-5.d) 🕒 (20 min.) [depends on Sub-problem (6-5.c)]

Write a C++ function

```
std::pair<double, double> bestBound();
```

that computes an approximation M of

$$\min_{\rho > 1} \frac{|\gamma_\rho| \max_{z \in \gamma_\rho} |f(z)|}{\pi d([-1, 1], \gamma_\rho)}, \quad (6.5.4)$$

by sampling 1000 equidistant ρ -values in the interval $(1, \rho_{\max}]$, where $\rho_{\max}$ is the value obtained in Sub-problem (6-5.c). For sampling along the integration path γ_ρ use 1000 points corresponding to equidistant θ -values, cf. Sub-problem (6-5.a). The function should return the pair $(M, \rho) \in \mathbb{R}^2$, where ρ stands for the ρ -value for which the minimum is attained.

The distance of $[a, b]$ from γ_ρ is formally defined as

$$d([a, b], \gamma) := \inf\{|z - t| \mid z \in \gamma, t \in [a, b]\}. \quad (6.5.5)$$

HINT 1 for (6-5.d): Apply geometric reasoning to establish that the distance of γ_ρ and $[-1, 1]$ is $\frac{1}{2}(\rho + \rho^{-1}) - 1$. ┘

SOLUTION of (6-5.d):

C++11-code 6.5.6: Code for bestBound.

```

2  std::pair<double, double> bestBound() {
3      const double maxRho = exp(asinh(M_PI / 3.));
4      constexpr unsigned int N = 1000;
5      double m = 0.0, r = 0.0; // min and rho for output
6      // TODO: (6-5.d) Compute (6.5.4) with d from (6.5.5) and
7      //  $|\gamma_\rho|$  from (6-5.a)
8      // START
9      // define f and gamma
10     const complex i(0, 1); // imaginary unit
11     auto f = [](complex z) { return 1. / (1. + std::exp(-3. * z)); };
12     auto gamma = [i](double rho, double theta) {
13         return std::cos(theta - i * std::log(rho));
14     };
15
16     m = std::numeric_limits<double>::max(); // save min in here
17     // discretize rhos and remove first (open interval!)
18     Eigen::VectorXd rhos = Eigen::VectorXd::LinSpaced(N + 1, 1, maxRho).head(N);
19     r = rhos(0); // save argmin here
20
21     // discretize the ellipse
22     Eigen::VectorXd thetas = Eigen::VectorXd::LinSpaced(N, 0, 2 * M_PI);
23
24     // loop over all rhos and get compute expression
25     for (unsigned int k = 0; k < rhos.size(); ++k) {
26         const double rho = rhos(k); // current rho
27         // get values on the ellipse
28         Eigen::VectorXcd z(thetas.size());
29         for (unsigned int j = 0; j < thetas.size(); ++j) {
30             z(j) = gamma(rho, thetas(j));
31         }
32
33         // sup of f of  $\gamma_\rho$ 
34         const Eigen::VectorXcd fz = z.unaryExpr(f);
35         const double M = lengthEllipse(rho, N) * fz.cwiseAbs().maxCoeff() *
36             M_1_PI; // nominator of expression
37
38         // Distance is the one between -1 and the minimal real part of the
39         // ellipse
40         // (i.e. semi-major-axis - 1):
41         const double d = (rho + 1. / rho) / 2. - 1; // denominator of expression
42
43         if (M / d < m) {
44             r = rho; // update rho
45         }
46         m = std::min(m, M / d); // update minimum
47     }
48     // END
49     return std::make_pair(m, r);
50 }

```

Get it on  GitLab (chebyApprox.hpp).



(6-5.e) 🕒 (15 min.) [depends on Sub-problem (6-5.d)]

Based on the result of Sub-problem (6-5.d) and [Lecture → Rem. 6.2.3.26], give an “optimal” bound for

$$\|f - L_n f\|_{L^\infty([-1,1])},$$

where $L_n : C^0([-1,1]) \rightarrow \mathcal{P}_n$ is the operator of **Chebyshev interpolation**, that is, polynomial Lagrange interpolation based on Chebyshev nodes [Lecture → Eq. (6.2.3.10)], on $[-1,1]$ into the space of polynomials of degree $\leq n$.

SOLUTION of (6-5.e):

Let M be the approximation of the minimum computed in Sub-problem (6-5.d) and ρ the associated ellipse parameter. Then we deduce from [Lecture → Eq. (6.2.3.28)] that

$$\|f - L_n f\|_{L^\infty([-1,1])} \lesssim \frac{M}{\rho^{n+1} - 1}.$$

From Sub-problem (6-5.d) we learned that $M \approx 10.0753$ and $\rho \approx 1.91355$. Applying this to the above bound, we find that

$$\|f - L_n f\|_{L^\infty([-1,1])} \lesssim \frac{10.0753}{1.91355^{n+1} - 1}. \quad (6.5.7)$$



(6-5.f) 🕒 (20 min.) [depends on Sub-problem (6-5.e)]

Implement a C++ function

```
std::pair<std::vector<double>, std::vector<double>> >
compareErrorAndBound(unsigned int n_max);
```

that computes and tabulates

1. estimates for $\|f - L_n f\|_{L^\infty([-1,1])}$ obtained by sampling in 10000 equidistant points $\in [-1,1]$,
2. the bounds derived in Sub-problem (6-5.e),

for polynomial degrees $n = 4, \dots, n_{\max}$, where $n_{\max}$ is passed through the argument `n_max`.

You may rely on the provided function `intpolyval` (cf. [Lecture → ??]).

HINT 1 for (6-5.f): The function `intpolyval` has the signature:

```
void intpolyval(const VectorXd& t, const VectorXd& y, const
VectorXd& x, VectorXd& p);
```

Where (t, y) is the interpolation data and p is used to save the values of the interpolant evaluated in x .

SOLUTION of (6-5.f):

C++11-code 6.5.8: Code for compareErrorAndBound.

```

2  std::pair<std::vector<double>, std::vector<double>> compareErrorAndBound(
3      unsigned int n_max) {
4      std::vector<double> upperBound, errInf;
5      constexpr unsigned N = 10000;
6
7      // TODO: (6-5.f) Tabulate the L-inf error and the bounds.
8      // START
9      auto m_rho = bestBound();
10     const double m = m_rho.first;
11     const double rho = m_rho.second;
12
13     // Test for degrees 4 to n_max
14     Eigen::VectorXd degrees = Eigen::VectorXd::LinSpaced(n_max - 3, 4, n_max);
15
16     for (unsigned int j = 0; j < degrees.size(); ++j) {
17         upperBound.emplace_back(m / (std::pow(rho, degrees(j) + 1) - 1));
18     }
19
20     // Compute actual error for any degree
21     auto f = [](double x) { return 1. / (1 + std::exp(-3 * x)); };
22
23     for (unsigned int l = 0; l < degrees.size(); ++l) {
24         const int n = degrees(l);
25         // Cheby nodes and value of f at the nodes
26         Eigen::ArrayXd k = Eigen::ArrayXd::LinSpaced(n + 1, 0, n);
27         Eigen::VectorXd t = ((2 * k + 1) / (2 * (n + 1)) * M_PI).cos().matrix(),
28             y = t.unaryExpr(f);
29         // Evaluate error at a couple of points
30         Eigen::VectorXd x = Eigen::VectorXd::LinSpaced(N, -1, 1),
31             F = x.unaryExpr(f);
32
33         // Use barycentric formula (in intpolyval.hpp)
34         Eigen::VectorXd LnF;
35         intpolyval(t, y, x, LnF);
36         errInf.push_back((F - LnF).cwiseAbs().maxCoeff());
37     }
38     // END
39     return {errInf, upperBound};
40 }

```

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

Remark. Algorithms related to Chebychev interpolation can usually be implemented very efficiently by relying on FFT, see [Lecture → Section 6.2.3.3]. In this problem, the focus is not on efficient implementation and so we do not pursue this.

Degree n	$\ f - L_n f\ _{L^\infty([-1,1])}$	Bound (6.5.7)
4	0.0122079	0.408628
5	0.00263694	0.209489
6	0.00202698	0.1084
7	0.000423312	0.0563594
8	0.000330751	0.0293744
9	6.79868e-05	0.0153294
10	5.3577e-05	0.00800517
11	1.09199e-05	0.00418183
12	8.64874e-06	0.00218495
13	1.75395e-06	0.00114171
14	1.3936e-06	0.000596614
15	2.8172e-07	0.000311775
16	2.24322e-07	0.000162928
17	4.52499e-08	8.51438e-05
18	3.60853e-08	4.4495e-05
19	7.26804e-09	2.32526e-05
20	5.80245e-09	1.21515e-05

The actual interpolation error is still far below the bound, which is hardly surprising, because the path integral has been estimated fairly crudely.



(6-5.g) 🕒 (20 min.) Rely on pullback to $[-1, 1]$ to discuss how the error bounds in [Lecture → Eq. (6.2.3.28)] will change qualitatively when we consider Chebychev interpolation on $[-a, a]$, $a > 0$, instead of $[-1, 1]$, whilst keeping the function f fixed.

SOLUTION of (6-5.g):

Let $\Phi : [-1, 1] \rightarrow [-a, a]$ the unique affine mapping of the two intervals to each other, which means $\Phi(t) = at$, which means $(\Phi^* f)(t) = f(\frac{t}{a})$. Thus, the rescaled function $\Phi^* f$ will have a *different domain of analyticity* D_a compared to f . In fact, its domain of analyticity will be $D_a = D/a$.

Thus, the larger a , the closer the pole of $\Phi^* f$ will move to $[-1, 1]$, the more the choice of the ellipses is restricted (i.e. $\rho_{\max}$ becomes smaller). This will result in a larger bound.



End Problem 6-5, 130 min.