

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 7-6: Weighted Gauss quadrature

In [Lecture → Rem. 7.4.3.10] we saw, how, in special cases, integrals with singular integrands can be transformed into integrals with smooth integrands, which allow much faster convergence of Gauss quadrature. This problem examines an alternative method: If the type of the singular behavior of the integrand is known, special quadrature rules, called weighted Gauss rules, can be designed that enjoy fast convergence.

This problem relies on the material of [Lecture → Section 7.4] and requests a little implementation in C++.

The development of an alternative quadrature formula for a particular singular integral relies on the Chebyshev polynomials of the second kind U_n , defined as

$$U_n(t) = \frac{\sin((n+1) \arccos t)}{\sin(\arccos t)}, \quad n \in \mathbb{N}.$$

The U_n are **orthogonal polynomials** with respect to a weighted L^2 inner product (see [Lecture → Eq. (6.3.2.3)]) with weight function given by $w(\tau) = \sqrt{1 - \tau^2}$.

(7-6.a) ☐ (10 min.) Recall the role of the orthogonal Legendre polynomials in the derivation and definition of Gauss-Legendre quadrature rules (see [Lecture → § 7.4.2.15]). ▲

(7-6.b) ☐ (15 min.) Show that the U_n satisfy the 3-term recursion

$$U_{n+1}(t) = 2tU_n(t) - U_{n-1}(t), \quad U_0(t) = 1, \quad U_1(t) = 2t,$$

for every $n \geq 1$.

SOLUTION of (7-6.b):

n = 0:

This case is trivial, since $U_0(t) = \frac{\sin(\arccos t)}{\sin(\arccos t)} = 1$, as desired.

n = 1:

Using the trigonometric identity $\sin 2x = \sin x \cos x$, we have

$$U_1(t) = \frac{2 \sin(\arccos t)}{\sin(\arccos t)} = 2 \cos \arccos t = 2t.$$

n ≥ 2:

Using the identity $\sin(x+y) = \sin x \cos y + \sin y \cos x$, we obtain:

$$\begin{aligned} U_{n+1}(t) &= \frac{\sin((n+1) \arccos t)t + \cos((n+1) \arccos t) \sin(\arccos t)}{\sin(\arccos t)} \\ &= U_n(t)t + \cos((n+1) \arccos t). \end{aligned}$$

Similarly, we have

$$\begin{aligned} U_{n-1}(t) &= \frac{\sin((n+1-1) \arccos t)}{\sin(\arccos t)} \\ &= \frac{\sin((n+1) \arccos t)t - \cos((n+1) \arccos t) \sin(\arccos t)}{\sin(\arccos t)} \\ &= U_n(t)t - \cos((n+1) \arccos t). \end{aligned}$$

Combining the last two equalities we obtain the desired 3-term recursion.



(7-6.c) □ (5 min.) Show that $U_n \in \mathcal{P}_n$ and has leading coefficient 2^n .

SOLUTION of (7-6.c):

Let us prove the claim by induction.

$n = 0$: This case is trivial, since $U_0(t) = 1$.

Induction hypothesis: Let us now assume that the statement is true for every $k = 0, \dots, n$ and let us prove it for $n + 1$.

In view of $U_{n+1}(t) = 2tU_n(t) - U_{n-1}(t)$, since by inductive hypothesis $U_n \in \mathcal{P}_n$ and $U_{n-1} \in \mathcal{P}_{n-1}$, we have that $U_{n+1} \in \mathcal{P}_{n+1}$. Moreover, the leading coefficient will be 2 times the leading order coefficient of U_n , namely 2^{n+1} , as desired. □



(7-6.d) □ (15 min.) Show that for every $m, n \in \mathbb{N}_0$, we have

$$\int_{-1}^1 \sqrt{1-t^2} U_m(t) U_n(t) dt = \frac{\pi}{2} \delta_{mn}.$$

SOLUTION of (7-6.d):

With the substitution $t = \cos s$ we obtain

$$\begin{aligned} \int_{-1}^1 \sqrt{1-t^2} U_m(t) U_n(t) dt &= \int_{-1}^1 \sqrt{1-t^2} \frac{\sin((m+1) \arccos t) \sin((n+1) \arccos t)}{\sin^2(\arccos t)} dt \\ &= \int_0^\pi \sin s \frac{\sin((m+1)s) \sin((n+1)s)}{\sin^2 s} \sin s ds \\ &= \int_0^\pi \sin((m+1)s) \sin((n+1)s) ds \\ &= \frac{1}{2} \int_0^\pi \cos((m-n)s) - \cos((m+n+2)s) ds. \end{aligned}$$



(7-6.e) □ (10 min.) What are the zeros ξ_j^n ($j = 1, \dots, n$) of U_n , $n \geq 1$? Give an explicit formula similar to the formula for the Chebyshev nodes in $[-1, 1]$.

SOLUTION of (7-6.e):

From the definition of U_n we immediately find that the zeros are given by

$$\xi_j^n = \cos\left(\frac{j}{n+1}\pi\right), \quad j = 1, \dots, n. \quad (7.6.1)$$



(7-6.f) (30 min.) [depends on Sub-problem (7-6.d)]

Show that the choice of weights

$$w_j = \frac{\pi}{n+1} \sin^2\left(\frac{j}{n+1}\pi\right), \quad j = 1, \dots, n,$$

ensures that the quadrature formula

$$Q_n^U(f) = \sum_{j=1}^n w_j f(\xi_j^n) \quad (7.6.2)$$

provides the exact value of the integral

$$I := \int_{-1}^1 \sqrt{1-t^2} f(t) dt$$

for $f \in \mathcal{P}_{n-1}$ (assuming exact arithmetic).

SOLUTION of (7-6.f):

Since U_k is a polynomial of degree exactly k , the set $\{U_k : k = 0, \dots, n-1\}$ is a basis of $\mathcal{P}_{n-1}$. Therefore, by linearity it suffices to prove the above identity for $f = U_k$ for every k . Fix $k = 0, \dots, n-1$. Setting $x = \pi/(n+1)$, from (7.6.1) we readily derive

$$\begin{aligned} \sum_{j=1}^n w_j U_k(\xi_j^n) &= \sum_{j=1}^n \frac{\pi}{n+1} \sin^2\left(\frac{j}{n+1}\pi\right) \frac{\sin((k+1) \arccos \xi_j^n)}{\sin(\arccos \xi_j^n)} \\ &= x \sum_{j=1}^n \sin(jx) \sin((k+1)jx) \\ &= \frac{x}{2} \sum_{j=1}^n (\cos((k+1-1)jx) - \cos((k+1+1)jx)) \\ &= \frac{x}{2} \operatorname{Re} \sum_{j=0}^n (e^{ikxj} - e^{i(k+2)xj}) \\ &= \frac{x}{2} \operatorname{Re} \left(\sum_{j=0}^n e^{ikxj} - \frac{1 - e^{i\pi(k+2)}}{1 - e^{i(k+2)x}} \right). \end{aligned}$$

Thus, for $k = 0$ we have

$$\sum_{j=1}^n w_j U_0(\xi_j^n) = \frac{x}{2} \operatorname{Re} \left(\sum_{j=0}^n 1 - \frac{1 - e^{2\pi i}}{1 - e^{2xi}} \right) = \frac{x}{2} \operatorname{Re}((n+1) - 0) = \frac{\pi}{2}.$$

On the other hand, if $k = 1, \dots, n-1$ we obtain

$$\sum_{j=1}^n w_j U_k(\xi_j^n) = \frac{x}{2} \operatorname{Re} \left(\frac{1 - e^{i\pi k}}{1 - e^{ikx}} - \frac{1 - e^{i\pi(k+2)}}{1 - e^{i(k+2)x}} \right) = \frac{(1 - (-1)^k)x}{2} \operatorname{Re} \left(\frac{1}{1 - e^{ikx}} - \frac{1}{1 - e^{i(k+2)x}} \right).$$

In view of the elementary equality $(a + ib)(a - ib) = a^2 + b^2$ we have $\operatorname{Re}(1/(a + ib)) = a/(a^2 + b^2)$. Thus

$$\operatorname{Re}\left(\frac{1}{1 - e^{ikx}}\right) = \operatorname{Re}\left(\frac{1}{1 - \cos(kx) - i \sin(kx)}\right) = \frac{1 - \cos(kx)}{(1 - \cos(kx))^2 + \sin(kx)^2} = \frac{1}{2}$$

Arguing in a similar way we have $\operatorname{Re}(1 - e^{i(k+2)x})^{-1} = 1/2$. Therefore for $k = 1, \dots, n-1$ we have

$$\sum_{j=1}^n w_j U_k(\xi_j^n) = \frac{(1 - (-1)^k)x}{2} \left(\frac{1}{2} - \frac{1}{2}\right) = 0.$$

To summarise, we have proved that

$$\sum_{j=1}^n w_j U_k(\xi_j^n) = \frac{\pi}{2} \delta_{k0}, \quad k = 0, \dots, n-1.$$

Finally, the claim follows from Sub-problem (7-6.d), since $U_0(t) = 1$.

(7-6.g)  Show that the quadrature formula (7.6.2) gives the exact value of (??) even for every $f \in \mathcal{P}_{2n-1}$.

HINT 1 for (7-6.g): Refer to [Lecture → Thm. 7.4.2.11].

SOLUTION of (7-6.g):

The conclusion follows by applying the same argument given in [Lecture → Thm. 7.4.2.11] with the weighted L^2 scalar product with weight w defined above.

(7-6.h)  Show that the quadrature error

$$|Q_n^U(f) - W(f)|$$

decays to 0 exponentially as $n \rightarrow \infty$ for every $f \in C^\infty([-1, 1])$ that admits an analytic extension to an open subset of the complex plane.

HINT 1 for (7-6.h): See [Lecture → § 7.4.3.2].

SOLUTION of (7-6.h):

By definition, the weights defined above are positive, and the quadrature rule is exact for polynomials up to order $2n-1$. Therefore, arguing as in [Lecture → § 7.4.3.2], we obtain the exponential decay, as desired.

(7-6.i)  Write a C++ function

```
template<typename Function>
double quadU(const Function & f, const unsigned n);
```

that gives $Q_n^U(f)$ as output, where f is an object with an evaluation operator, like a lambda function, representing f , e.g.

```
auto f = [] (double & t) { return 1/(2 + exp(3*t)); };
```

SOLUTION of (7-6.i):

The full code can be found in `quadU.cpp`.

C++ code 7.6.3: Solution for quadU.

```
2  template<typename Function>
3  double quadU(const Function& f, const unsigned int n) {
4      // Value of integral
5      double I = 0;
6
7      // TODO: (7-6.i) Integrate f using weighted Gauss quadrature
8      // START
9      for (unsigned int j = 0; j < n; ++j) {
10         // Weight
11         const double w =
12             M_PI / (n + 1) * std::pow(std::sin((j + 1) * M_PI / (n + 1)), 2);
13
14         // Node
15         const double xi = std::cos((j + 1.) / (n + 1) * M_PI);
16
17         I += w * f(xi);
18     }
19     // END
20     return I;
21 }
```



(7-6.j)  Devise a C++ function

```
void testQuadU(unsigned int nmax = 20);
```

that, for the function $f(t) = 1/(2 + e^{3t})$ and $n = 1, \dots, n_{\max}$ plots (employing `MATPLOTLIBCPP`) the quadrature error $E_n(f) = |W(f) - Q_n^U(f)|$ using the “exact” value $W(f) = 0.483296828976607$. Estimate the parameter $0 \leq q < 1$ in the asymptotic decay law $E_n(f) \approx Cq^n$ characterising (sharp) exponential convergence, see [Lecture → Def. 6.2.2.7].

SOLUTION of (7-6.j):

The full code can be found in `quadU.cpp`.

C++ code 7.6.4: Computation of convergence of quadU.

```
2  void testQuadU(unsigned int nmax = 20) {
3      // "Exact" value of integral
```

```

4  constexpr double exact = 0.483296828976607;
5  // Integrand
6  auto f = [](double t) { return 1. / (2. + std::exp(3. * t)); };
7
8  plt::figure();
9
10 // TODO: (7-6.j) Tabulate and plot the quadrature error.
11 // START
12 // memory allocation for plot
13 std::vector<double> err(nmax);
14 std::vector<int> num_pts(nmax);
15
16 std::cout << std::setw(5) << "n" << std::setw(20) << "Error" << std::setw(20)
17           << "q" << std::endl;
18
19 for (unsigned int n = 1; n <= nmax; ++n) {
20     // Compute error
21     err[n - 1] = std::abs(quadU(f, n) - exact);
22     num_pts[n - 1] = n;
23
24     if (n > 1) {
25         // Print table
26         std::cout << std::setw(5) << n << std::setw(20) << err[n - 1]
27                 << std::setw(20) << err[n - 1] / err[n - 2] << std::endl;
28     }
29 }
30 // Error plot rendered by matplotlibcpp
31 plt::semilogy(num_pts, err, "^", {"label", "error"});
32 plt::xlim(0.5, nmax + 0.5);
33 plt::xticks(num_pts);
34 plt::xlabel("No. of quadrature nodes");
35 plt::ylabel("|Error|");
36 plt::title("Convergence of quadU");
37 plt::grid("True");
38 // END
39 plt::savefig("./cx_out/convergence.png");
40 }

```

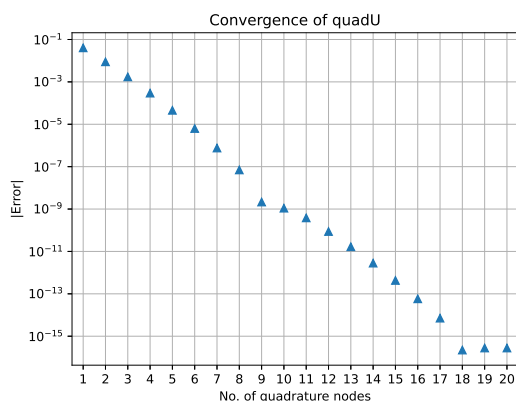


Fig. 64

An approximation of q is given by $E_n(f)/E_{n-1}(f)$:

n	Error	Approximated q
2	0.00171981	0.194545
3	0.000294532	0.171259
4	4.54948e-05	0.154465
5	6.30403e-06	0.138566
6	7.57006e-07	0.120083
7	6.99966e-08	0.0924651
8	2.14329e-09	0.0306199
9	1.09843e-09	0.512495
10	3.82291e-10	0.348035
11	8.71921e-11	0.228078
12	1.66109e-11	0.190509
13	2.80703e-12	0.168988
14	4.28158e-13	0.15253
15	5.80647e-14	0.135615
16	7.10543e-15	0.122371
17	2.77556e-16	0.0390625
18	2.77556e-16	1
19	2.77556e-16	1



End Problem 7-6 , 85 min.