Prof. R. Hiptmair
G. Alberti,
F. Leonardi

AS 2015

Numerical Methods for CSE

ETH Zürich
D-MATH

Problem Sheet 11

Problem 1 Efficient quadrature of singular integrands (core problem)

This problem deals with efficient numerical quadrature of non-smooth integrands with a special structure. Before you tackle this problem, read about regularization of integrands by transformation [1, Rem. 5.3.45].

Our task is to develop quadrature formulas for integrals of the form:

$$W(f) := \int_{-1}^{1} \sqrt{1-t^2}\, f(t)dt, \tag{66}$$

where f possesses an analytic extension to a complex neighbourhood of $[-1,1]$.

(1a) ⊡ The provided function

```
QuadRule gauleg(unsigned int n);
```

returns a structure `QuadRule` containing nodes (x_j) and weights (w_j) of a Gauss-Legendre quadrature (→ [1, Def. 5.3.28]) on $[-1,1]$ with n nodes. Have a look at the file `gauleg.hpp` and `gauleg.cpp`, and understand how the implementation works and how to use it.

HINT: Learn/remember how linking works in C++. To use the function `gauleg` (declared in `gauleg.hpp` and defined in `gauleg.cpp`) in a file `file.cpp`, first `include` the header file `gauleg.hpp` in the file `file.cpp`, and then compile and link the files `gauleg.cpp` and `file.cpp`. Using `gcc`:

```
g++ [compiler opts.] -c gauleg.cpp
g++ [compiler opts.] -c file.cpp
g++ [compiler opts.]  gauleg.o file.o -o exec_name
```

If you want to use `CMake`, have a look at the file `CMakeLists.txt`.

Solution: See documentation in `gauleg.hpp` and `gauleg.cpp`.

(1b) Study [1, § 5.3.37] in order to learn about the convergence of Gauss-Legendre quadrature.

(1c) Based on the function `gauleg`, implement a C++ function

```
template <class func>
double quadsingint(func&& f, unsigned int n);
```

that approximately evaluates (66) using $2n$ evaluations of f. An object of type `func` must provide an evaluation operator

```
double operator(double t) const;
```

For the quadrature error asymptotic exponential convergence to zero for $n \to \infty$ must be ensured by your function.

HINT: A C++ lambda function provides such operator.

HINT: You may use the classical binomial formula $\sqrt{1-t^2} = \sqrt{1-t}\sqrt{1+t}$.

HINT: You can use the template `quadsingint_template.cpp`.

Solution: Exploiting the hint, we see that the integrand is non-smooth in ± 1.

The first possible solution is the following (I): we split the integration domain $[-1,1]$ in $[0,1]$ and $[-1,0]$. Applying the substitution $s = \sqrt{1 \pm t}$ (sign depending on which part of the integrals considered), $t = \pm(s^2 - 1)$:

$$\frac{dt}{ds} = \pm 2s$$

$$\begin{aligned} W(f) := \int_{-1}^{1} \sqrt{1-t^2} f(t) dt &= \int_{-1}^{0} \sqrt{1-t^2} f(t) dt + \int_{0}^{1} \sqrt{1-t^2} f(t) dt \\ &= \int_{0}^{1} 2 \cdot s^2 \sqrt{2-s^2} f(-s^2+1) ds + \int_{0}^{1} 2 \cdot s^2 \sqrt{2-s^2} f(s^2-1) ds. \end{aligned}$$

Notice how the resulting integrand is analytic in a neighbourhood of the domain of integration because, for instant, $t \mapsto \sqrt{1+t}$ is analytic in a neighborhood of $[0,1]$.

Alternatively (II), one may use the trigonometric substitution $t = \sin s$, with $\frac{dt}{ds} = \cos s$

obtaining

$$W(f) := \int_{-1}^{1} \sqrt{1-t^2} f(t) dt$$
$$= \int_{-\pi/2}^{\pi/2} \sqrt{1-\sin^2 s}\, f(\sin s) \cos s \, ds = \int_{-\pi/2}^{\pi/2} \cos^2 s\, f(\sin s) ds.$$

This integrand is also analytic. The C++ implementation is in `quadsingint.cpp`.

(1d) ⚁ Give formulas for the nodes c_j and weights $\tilde{w}_j$ of a $2n$-point quadrature rule on $[-1,1]$, whose application to the integrand f will produce the same results as the function `quadsingint` that you implemented in (1c).

Solution: Let (x_j, w_j) be the original Gauss nodes and weights.

Using substitution (I): the nodes are mapped from x_j in $[-1,1]$ to c_j for $j \in 0, \ldots, 2n-1$ in $[-1,1]$ as follows:

$$c_j = \begin{cases} (x_j+1)^2/4 - 1 & j \text{ even} \\ -(x_j+1)^2/4 + 1 & j \text{ odd} \end{cases}$$

The weights $\tilde{w}_j, j = 0, \ldots, n-1$, become:

$$\tilde{w}_{2j} = \tilde{w}_{2j+1} = w_j x_j^2 \sqrt{2 - x_j^2}.$$

Using substitution (II): the nodes are mapped from x_j to c_j as follows:

$$c_j = \sin(x_j \pi/2)$$

The weights $\tilde{w}_j, j = 0, \ldots, n-1$, become:

$$\tilde{w}_j = w_j cos^2(x_j \pi/2)\pi/2.$$

(1e) ⚀ Tabulate the quadrature error:

$$|W(f) - \texttt{quadsingint(f,n)}|$$

for $f(t) := \frac{1}{2+\exp(3t)}$ and $n = 1, 2, ..., 25$. Estimate the $0 < q < 1$ in the decay law of exponential convergence, see [1, Def. 4.1.31].

Solution: The convergence is exponential with both methods. The C++ implementation is in `quadsingint.cpp`.

Problem 2 Nested numerical quadrature

A laser beam has intensity

$$I(x,y) = \exp(-\alpha((x-p)^2 + (y-q)^2))$$

on the plane orthogonal to the direction of the beam.

(2a) Write down the radiant power absorbed by the triangle

$$\triangle := \{(x,y)^T \in \mathbb{R}^2 \mid x \geq 0, y \geq 0, x + y \leq 1\}$$

as a double integral.

HINT: The radiant power absorbed by a surface is the integral of the intensity over the surface.

Solution: The radiant power absorbed by $\triangle$ can be written as:

$$\int_{\triangle} I(x,y)dxdy = \int_0^1 \int_0^{1-y} I(x,y)dxdy.$$

(2b) Write a C++ function

```
template <class func>
double evalgaussquad(double a, double b, func&& f, const
    QuadRule & Q);
```

that evaluates an the N-point quadrature for an integrand passed in `f` in $[a,b]$. It should rely on the quadrature rule on the reference interval $[-1,1]$ that supplied through an object of type `QuadRule`. (The vectors `weights` and `nodes` denote the weights and nodes of the reference quadrature rule respectively.)

HINT: Use the function `gauleg` declared in `gauleg.hpp` and defined in `gauleg.cpp` to compute nodes and weights in $[-1,1]$. See Problem 1 for further explanations.

HINT: You can use the template `laserquad_template.cpp`.

Solution: See `laserquad.cpp` and `CMakeLists.txt`.

(2c) Write a C++ function

```
template <class func>
double gaussquadtriangle(func&& f, int N)
```

for the computation of the integral

$$\int_{\triangle} f(x,y)dxdy, \tag{67}$$

using nested N-point, 1D Gauss quadratures (using the functions `evalgaussquad` of (2b) and `gauleg`).

HINT: Write (67) explicitly as a double integral. Take particular care to correctly find the intervals of integration.

HINT: Lambda functions of C++ are well suited for this kind of implementation.

Solution: The integral can be written as

$$\int_{\triangle} f(x,y)dxdy = \int_0^1 \int_0^{1-y} f(x,y)dxdy.$$

In the C++ implementation, we define the auxiliary (lambda) function f_y:

$$\forall y \in [0,1], f_y : [1, 1-y] \to \mathbb{R}, x \mapsto f_y(x) := f(x,y)$$

We also define the (lambda) approximated integrand:

$$g(y) := \int_0^{1-y} f_y(x)dx \approx \frac{1}{1-y} \sum_{i=0}^{N} w_i f_y\left(\frac{x_i+1}{2}(1-y)\right) =: \mathcal{I}(y), \tag{68}$$

the integral of which can be approximated, using a nested Gauss quadrature:

$$\int_{\triangle} f(x,y)dxdy = \int_0^1 \int_0^{1-y} f_y(x)dxdx = \int_0^1 g(y)dy \approx \frac{1}{2} \sum_{j=1}^{N} w_j \mathcal{I}\left(\frac{y_j+1}{2}\right). \tag{69}$$

The implementation can be found in `laserquad.cpp`.

(2d) $\boxdot$ Apply the function `gaussquadtriangle` of (2c) to the subproblem (2a) using the parameter $\alpha = 1, p = 0, q = 0$. Compute the error w.r.t to the number of nodes N. What kind of convergence do you observe? Explain the result.

HINT: Use the "exact" value of the integral 0.366046550000405.

Solution: As one expects from theoretical considerations, the convergence is exponential. The implementation can be found in `laserquad.cpp`.

Problem 3 Weighted Gauss quadrature

The development of an alternative quadrature formula for (66) relies on the Chebyshev polynomials of the second kind U_n, defined as

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

Recall the role of the orthogonal Legendre polynomials in the derivation and definition of Gauss-Legendre quadrature rules (see [1, § 5.3.25]).

As regards the integral (66), this role is played by the U_n, which are orthogonal polynomials with respect to a weighted L^2 inner product, see [1, Eq. (4.2.20)], with weight given by $w(\tau) = \sqrt{1-\tau^2}$.

(3a) Show that the U_n satisfy the 3-term recursion

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

for every $n \geq 1$.

Solution: The case $n = 0$ is trivial, since $U_0(t) = \frac{\sin(\arccos t)}{\sin(\arccos t)} = 1$, as desired. Using the trigonometric identity $\sin 2x = \sin x \cos x$, we have $U_1(t) = \frac{2\sin(\arccos t)}{\sin(\arccos t)} = 2\cos\arccos t = 2t$, as desired. Finally, using the identity $\sin(x+y) = \sin x \cos y + \sin y \cos x$, we obtain for $n \geq 2$

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

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.

(3b) ⚀ Show that $U_n \in \mathcal{P}_n$ with leading coefficient 2^n.

Solution: Let us prove the claim by induction. The case $n = 0$ is trivial, since $U_0(t) = 1$. 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.

(3c) ⚁ 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: With the substitution $t = \cos s$ we obtain

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

The claim immediately follows, as it was done in Problem Sheet 9, Problem 3.

(3d) ⚀ 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: From the definition of U_n we immediately find that the zeros are given by

$$\xi_j^n = \cos\left(\frac{j}{n+1}\pi\right), \qquad j = 1, \dots, n. \tag{70}$$

(3e) ⚃ Show that the choice of weights

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

ensures that the quadrature formula

$$Q_n^U(f) = \sum_{j=1}^{n} w_j f(\xi_j^n) \tag{71}$$

provides the exact value of (66) for $f \in \mathcal{P}_{n-1}$ (assuming exact arithmetic).

HINT: Use all the previous subproblems.

Solution: 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 $k = 0, \dots, n-1$. From (70) we readily derive

$$
\begin{aligned}
\sum_{j=1}^{n} w_j U_k(\xi_j^n) &= \sum_{j=1}^{n} \frac{\pi}{n+1} \sin^2\left(\frac{j}{n+1}\pi\right) \frac{\sin((k+1)\arccos \xi_j^n)}{\sin(\arccos \xi_j^n)} \\
&= \frac{\pi}{n+1} \sum_{j=1}^{n} \sin\left(\frac{j}{n+1}\pi\right) \sin((k+1)\left(\frac{j}{n+1}\pi\right)) \\
&= \frac{\pi}{2(n+1)} \sum_{j=1}^{n} \left(\cos(\frac{(k+1-1)j}{n+1}\pi) - \cos(\frac{(k+1+1)j}{n+1}\pi)\right) \\
&= \frac{\pi}{2(n+1)} \operatorname{Re}\left(\sum_{j=0}^{n} e^{\frac{kj}{n+1}\pi i} - e^{\frac{(k+2)j}{n+1}\pi i}\right) \\
&= \frac{\pi}{2(n+1)} \operatorname{Re}\left(\sum_{j=0}^{n} e^{\frac{kj}{n+1}\pi i} - \frac{1 - e^{i\pi(k+2)}}{1 - e^{i\pi(k+2)/(n+1)}}\right) \\
&= \frac{\pi}{2}\delta_{k0}.
\end{aligned}
$$

Finally, the claim follows from (3c), since $U_0(t) = 1$ and so the integral in (66) is nothing else than the weighted scalar product between U_k and U_0.

(3f) Show that the quadrature formula (71) gives the exact value of (66) even for every $f \in \mathcal{P}_{2n-1}$.

HINT: See [1, Thm. 5.3.21].

Solution: The conclusion follows by applying the same argument given in [1, Thm. 5.3.21] with the weighted L^2 scalar product with weight w defined above.

(3g) Show that the quadrature error

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

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

HINT: See [1, § 5.3.37].

Solution: 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 [1, § 5.3.37], we obtain the exponential decay, as desired.

(3h) Write a C++ function

```
template<typename Function>
double quadU(const Function &f, unsigned int 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: See file `quadU.cpp`.

(3i) Test your implementation with the function $f(t) = 1/(2+e^{3t})$ and $n = 1, \dots, 25$. Tabulate the quadrature error $E_n(f) = |W(f) - Q_n^U(f)|$ using the "exact" value $W(f) = 0.483296828976607$. Estimate the parameter $0 \le q < 1$ in the asymptotic decay law $E_n(f) \approx Cq^n$ characterizing (sharp) exponential convergence, see [1, Def. 4.1.31].

Solution: See file `quadU.cpp`. An approximation of q is given by $E_n(f)/E_{n-1}(f)$.

Problem 4 Generalize "Hermite-type" quadrature formula

(4a) Determine $A, B, C, x_1 \in \mathbb{R}$ such that the quadrature formula:

$$\int_0^1 f(x)dx \approx Af(0) + Bf'(0) + Cf(x_1) \tag{72}$$

is exact for polynomials of highest possible degree.

Solution: The quadrature is exact for every polynomial $p(x) \in \mathcal{P}^n$, if and only if it is exact for $1, x, x^2, \ldots, x^n$. If we apply the quadrature to the first monomials:

$$1 = \int_0^1 1dx = A \cdot 1 + B \cdot 0 + C \cdot 1 = A \tag{73}$$

$$\frac{1}{2} = \int_0^1 xdx = A \cdot 0 + B \cdot 1 + C \cdot x_1 = B + Cx_1 \tag{74}$$

$$\frac{1}{3} = \int_0^1 x^2dx = A \cdot 0 + B \cdot 0 + C \cdot x_1^2 = Cx_1^2 \tag{75}$$

$$\frac{1}{4} = \int_0^1 x^3dx = A \cdot 0 + B \cdot 0 + C \cdot x_1^3 = Cx_1^3 \tag{76}$$

$\Rightarrow B = \frac{1}{2} - Cx_1, C = \frac{1}{3x_1^2} \Rightarrow \frac{1}{4} = \frac{1}{3x_1^2}x_1^3 = \frac{1}{3}x_1, A = \frac{11}{27}$, i.e.

$$x_1 = \frac{3}{4}, C = \frac{16}{27}, B = \frac{1}{18}, A = \frac{11}{27}. \tag{77}$$

Then

$$\frac{1}{5} = \int_0^1 x^4dx \neq A \cdot 0 + B \cdot 0 + C \cdot x_1^4 = C \cdot x_1^4 = \frac{16}{27}\frac{81}{256}. \tag{78}$$

Hence, the quadrature is exact for polynomials up to degree 3.

(4b)

Compute an approximation of $z(2)$, where the function z is defined as the solution of the initial value problem

$$z'(t) = \frac{t}{1+t^2} \quad , \quad z(1) = 1 \; . \tag{79}$$

Solution: We know that

$$z(2) - z(1) = \int_1^2 z'(x)dx, \tag{80}$$

hence, applying (72) and the transformation $x \mapsto x + 1$, we obtain:

$$z(2) = \int_0^1 z'(x+1)dx + z(1) \approx \frac{11}{27} \cdot z'(1) + \frac{1}{18} \cdot z''(1) + \frac{16}{27} \cdot z'\left(\frac{7}{4}\right) + z(1). \tag{81}$$

With $z''(x) = -\frac{2 \cdot x}{(1+x^2)^2}$ and:

$$\begin{aligned}
z(1) &= 1, \\
z'(1) &= \frac{1}{1+1^2} = \frac{1}{2}, \\
z''(1) &= -\frac{2 \cdot 1}{(1+1^2)^2} = -\frac{1}{2}, \\
z'\left(\frac{7}{4}\right) &= \frac{(\frac{7}{4})}{1+(\frac{7}{4})^2} = \frac{28}{65},
\end{aligned}$$

we obtain

$$z(2) = \int_0^1 z'(x+1)dx + z(1) \approx \frac{11}{27} \cdot \frac{1}{2} - \frac{1}{18} \cdot \frac{1}{2} + \frac{16}{27} \cdot \frac{28}{65} + 1 = 1.43\ldots$$

For sake of completeness, using the antiderivative of z':

$$z(2) = \int_1^2 z'(x)dx + z(1) = \frac{1}{2}\log(x^2+1)|_1^2 + 1 = 1.45\ldots$$

Issue date: 26.11.2015

Hand-in: 03.12.2015 (in the boxes in front of HG G 53/54).

Version compiled on: December 7, 2015 (v. 1.0).