Accurate Multiple-Precision Gauss-Legendre Quadrature

Laurent Fousse
Université Henri-Poincaré Nancy 1
laurent@komite.net

Abstract

Numerical integration is an operation that is frequently available in multiple precision numerical software packages. The different quadrature schemes used are considered well studied but the rounding errors that result from the computation are often neglected, and the actual accuracy of the results are therefore seldom rigorously proven.

We propose an implementation of the Gauss-Legendre quadrature scheme with bounded error: given a bound on the derivatives of a function we are able to compute an interval containing the true value of the integral, in arbitrary precision. The error analysis is given as well as experimental error measurements and timings, and a complete quadrature example.

1. Introduction

Numerical integration is readily available in most multiple precision numerical computation software (e.g. Pari/GP, MuPAD, Mathematica, Maple, ...). In those systems the precision can usually be tuned by the user for each computation (it is generally understood as the "working precision" but it may also be the number of digits displayed, when these two values differ). It is however not necessarily clear how many, if any, of the displayed digits are correct. As a concrete example we ask Maple 10 the value of $I = \int_{17}^{42} e^{-x^2} \log x \mathrm{d}x$ with the default precision of 10 digits:

```
> evalf(Int(exp(-x^2)*ln(x), x=17..42));
                          -126
                0.2604007480 10
```

We may want to ask for a second value with a greater precision of 20 digits and we would get $v_2 = 0.34288028340847034512 \cdot 10^{-126}$ which has no common digit with the previous value $v_1 = 0.2604007480 \cdot 10^{-126}$. As we will see later increasing the precision did in fact worsen the result.

This experiment is a blunt reminder of the lack of clear semantics for floating-point computations beyond the basic operations covered by the IEEE 754 standard [6]. As soon as computations are composed or transcendental functions like the sine function are used nothing is guaranteed by the IEEE 754 standard, and multiple-precision arithmetics is not covered either. This is however not an excuse to rely only on heuristics to compute accurately, and it is still possible to obtain guaranteed results.

Several approaches were made to overcome these shortcomings when computing integrals. One can mention the use of adaptive quadrature functions with an automatic adjustment of the integration step to each subinterval (in MuPAD [9]), or dynamic error control (of simple or multiple integrals [1, 7]). However well these techniques may work in practice, they rely on heuristics to provide an accurate answer to an integration problem.

Our work differs from these approaches in that we seek to give a proven bound on the error that takes into account all sources of errors, including the rounding errors. What we compute is in fact an interval containing the result of the integral, and with a proper choice of parameters one can use our algorithm to increase arbitrarily the precision on the result.

This paper is organized as follows. We first recall briefly the Gauss-Legendre integration from a mathematical point of view, as well as some definitions and properties of floating-point arithmetics. In Section 3 we will describe the algorithms used to compute the Legendre polynomials and the coefficients of the method, which do not depend on the function to integrate and can therefore be precomputed for several functions.

We follow with our main result in Section 4: our quadrature algorithm (Algorithm 2) along with its error analysis and an error bound summarized in Theorem 3. We give a complete example of use of our algorithm in Section 5.

2. Reminders

2.1. Gauss-Legendre Rule

We give a description of the Gauss-Legendre quadrature method. It is a member of the Gaussian family of quadrature methods which is more generally studied in [2]. Algorithms for orthogonal polynomials and gaussian quadrature may be found in [4] also.

In this paper, $f : [a, b] \to \mathbb{R}$ is the C^∞ function we want to integrate on a finite domain $[a, b]$ and n is the number of points of the Gauss-Legendre method. Let

$$I = \int_a^b f(x)\mathrm{d}x$$

be the exact value of the integral. We define the inner product of f and g on $[a, b]$ for the admissible weight function w as

$$< f, g >= \int_a^b w(x)f(x)g(x)\mathrm{d}x.$$

This leads to the definition of a sequence of orthogonal polynomials $(p_i)_{i\geq 0}$ such that:

$$\forall i \in \mathbb{N}, \deg(p_i) = i$$

$$\forall (i, j) \in \mathbb{N}^2, < p_i, p_j >= \delta_{i,j}$$

where $\delta_{i,j}$ is Kronecker's delta. For fixed $n > 0$, p_n has n distinct roots in (a, b) which we name $x_0 < x_1 < \ldots < x_{n-1}$. The Gauss quadrature method associated to the weight function w on $[a, b]$ is the interpolatory method at evaluation points $(x_i)_{0\leq i<n}$ such that

$$\int_a^b w(x)p(x)\mathrm{d}x = \sum_{i=0}^{n-1} w_i p(x_i)$$

holds for every polynomial p of degree up to $n - 1$ (this is enough to define the weights w_i although the method will be shown to integrate accurately polynomials of degree up to $2n - 1$).

The Gauss-Legendre quadrature method is the Gauss method for the weight function $w = 1$. Additionally the Legendre polynomials $(P_n)_{n\geq 0}$ are usually defined on $[-1, 1]$ and normalized such that $P_n(1) = 1$ and we will follow this custom here.

2.2. Legendre Polynomials

In the rest of this paper P_n is the Legendre polynomial of degree n defined on $[-1, 1]$ as usual. The quadrature method on $[a, b]$ is derived from the quadrature method on $[-1, 1]$ from a shifting and scaling in the polynomial. We will mostly focus on $[-1, 1]$ but the results will be given for the integration interval $[a, b]$, with the details of the translation omitted.

We denote by $x'_0 < x'_1 < \ldots < x'_{n-1}$ the roots of P_n on $[-1, 1]$ and use the notation $x_0 < x_1 < \ldots < x_{n-1}$ for the translated roots on $[a, b]$.

Like other orthogonal polynomial sequences, the polynomials $(P_n)_{n\geq 0}$ satisfy a recurrence relationship:

$$\left\{ \begin{array}{rcl} P_0(X) & = & 1 \\ P_1(X) & = & X \\ (n+1)P_{n+1}(X) & = & (2n+1)XP_n(X) - nP_{n-1}(X). \end{array} \right. \quad (1)$$

From (1) we deduce that P_n has only monomials of degree the parity of n and has rational coefficients. We recall Rodrigues' representation:

$$P_n = \frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}((x^2 - 1)^n)$$

which shows that we can use 2^n as common denominator for the polynomial's coefficients. Thus P_n can be written

$$P_n(X) = \left\{ \begin{array}{ll} 2^{-n}Q_n(X^2) & \text{if } n \text{ is even} \\ 2^{-n}XQ_n(X^2) & \text{otherwise.} \end{array} \right.$$

The problem of computing P_n is reduced to the one of computing Q_n, which has integer coefficients. The procedure is detailed in Algorithm 1.

Algorithm 1 Computation of the Legendre Polynomials

INPUT: $n \geq 2$.
OUTPUT: Q_n.
$Q_0 \leftarrow 1$
$Q_1 \leftarrow 2$
$p \leftarrow 0$ ▷ holds the parity of the polynomial currently computed
for $i \leftarrow 2$ to n **do**
  $Q_p \leftarrow -4(i-1)Q_p + 2(2i-1)X^{1-p}Q_{1-p}$
  $Q_p \leftarrow \frac{1}{i}Q_p$ ▷ exact integer divisions
  $p \leftarrow 1 - p$
end for
return Q_{1-p}

2.3. Mathematical error

In this section we give the bound on the mathematical error made with the Gauss-Legendre quadrature method. A generic proof for any weight function w can be found in [2].

Theorem 1. *The Gauss-Legendre method on $[a, b]$ with n points is exact for polynomials of degree $\leq 2n - 1$.*

Theorem 2. *Let M_{2n} be a bound of $|f^{(2n)}|$ on $[a, b]$, then the error of the Gauss-Legendre integration of f on $[a, b]$ with infinite precision is bounded by*

$$\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}M_{2n}.$$

We will use in Section 5 the composition of the Gauss-Legendre quadrature method: for an order of composition m and an integration domain $[a, b]$ the composed Gauss-Legendre method is the application of the Gauss-Legendre method on each m intervals $\{[a, a + \frac{b-a}{m}], [a + \frac{b-a}{m}, a + 2\frac{b-a}{m}], \ldots, [b - \frac{b-a}{m}, b]\}$. The error of the composed method on $[a, b]$ with infinite precision is therefore bounded by

$$\frac{(b-a)^{2n+1}(n!)^4}{m^{2n}(2n+1)[(2n)!]^3}M_{2n}.$$

2.4. Floating-point Arithmetics

For the error analysis of Algorithm 2, we need a few useful lemmas concerning the "ulp calculus", as well as some definitions. The floating-point numbers are represented with radix 2 (this could be generalized for any radix but radix 2 is simpler and is natural on computers). For this section, p is the working precision, and we assume all floating-point numbers are normalized, which means in our notations that the exponent range is unbounded. We denote by $\circ(x)$ the floating-point number rounded to nearest in precision p of a given real value x.

Definition 1 (Exponent, Unit in the last place)**.** *For a non-zero real number x we define $E(x) := 1 + \lfloor \log_2 |x| \rfloor$, such that $2^{E(x)-1} \leq |x| < 2^{E(x)}$, and $\mathrm{ulp}(x) := 2^{E(x)-p}$.*

For a real $x \neq 0$ and a working precision p we always have $2^{p-1}\mathrm{ulp}(x) \leq |x| < 2^p\mathrm{ulp}(x)$. If x is a floating-point number, then $\mathrm{ulp}(x)$ is the weight of the least significant bit — zero or not — in the p-bit mantissa of x. For all real x, $\mathrm{ulp}(x)$ is always greater than zero by definition.

Lemma 1. *If $c \neq 0$ and $x \neq 0$ then $c \cdot \mathrm{ulp}(x) < 2 \cdot \mathrm{ulp}(cx)$.*

Lemma 2. *Assuming no underflow occurs then in all rounding modes for a non zero real x we have: $\mathrm{ulp}(x) \leq \mathrm{ulp}(\circ(x))$, where $\circ(x)$ is the rounding of x in the chosen mode with an unbounded exponent range.*

Lemma 3. *Let x a non-zero real and $\circ(x)$ its rounding to nearest on p bits. Then $|x| \leq (1 + 2^{-p})|\circ(x)|$.*

Lemma 4. *Let a and b be two non-zero floating-point numbers of the same sign and precision p then in all rounding modes*

$$\mathrm{ulp}(a) + \mathrm{ulp}(b) \leq \frac{3}{2}\mathrm{ulp}(\circ(a+b)).$$

PROOF: It suffices to consider the case where a and b are positive. The definition of ulp gives:

$$2^{p-1}\mathrm{ulp}(a) \leq a < 2^p\mathrm{ulp}(a),$$

$$2^{p-1}\mathrm{ulp}(b) \leq b < 2^p\mathrm{ulp}(b)$$

thus

$$2^{p-1}[\mathrm{ulp}(a) + \mathrm{ulp}(b)] \leq a + b < 2^p[\mathrm{ulp}(a) + \mathrm{ulp}(b)].$$

If $\mathrm{ulp}(a) = \mathrm{ulp}(b)$ we get

$$2^p\mathrm{ulp}(a) \leq a + b < 2^{p+1}\mathrm{ulp}(a)$$

and therefore $\mathrm{ulp}(\circ(a+b)) \geq \mathrm{ulp}(a+b) \geq 2\mathrm{ulp}(a) = \mathrm{ulp}(a) + \mathrm{ulp}(b)$ (Lemma 2) and the lemma holds.

Otherwise we can assume without loss of generality that $\mathrm{ulp}(a) > \mathrm{ulp}(b)$, that is $\mathrm{ulp}(a) \geq 2 \cdot \mathrm{ulp}(b)$. We deduce:

$$\mathrm{ulp}(a) + \mathrm{ulp}(b) \leq \frac{3}{2}\mathrm{ulp}(a),$$

and together with $\mathrm{ulp}(\circ(a+b)) \geq \mathrm{ulp}(a+b) \geq \mathrm{ulp}(a)$ (Lemma 2) this concludes the proof. □

Lemma 5. *For x and y real numbers and using rounding to nearest in precision p we have*

$$|\circ(\circ(x) \cdot \circ(y)) - xy| \leq \frac{5}{2}\mathrm{ulp}(\circ(\circ(x) \cdot \circ(y))).$$

3. Pre-computations

In the integration algorithm the evaluation points and the weights of the method do not depend on the function to integrate and their computation can thus be shared among several executions of the algorithm. We will now explain how these quantities are computed.

3.1. Evaluation points

Computing the roots $(x'_i)_{0 \leq i < n}$ of P_n reduces to the computation of the roots of Q_n. Let $m = \lfloor \frac{n}{2} \rfloor$ and $u_0 < u_1 < \ldots < u_{m-1}$ be the roots of Q_n, we have:

$$\left\{x'_i, x'_{n-1-i}\right\} = \left\{\pm\sqrt{u_i}\right\}, \quad 0 \leq i < m$$

and $x'_m = 0$ if n is odd.

The process of computing the roots of Q_n involves two steps:

1. root isolation, that is finding m intervals that contain exactly one positive root of Q_n each,
2. root refinement.

The root isolation is made using Uspensky's algorithm as described in [11]. The input of the algorithm is $Q_n(x)$, and the output is a sequence of m intervals of the form $\frac{c_i}{2^{l_i}}$ where c_i and l_i are integers and such that $[\frac{c_i}{2^{l_i}}, \frac{c_i+1}{2^{l_i}}]$ contains exactly one root of Q_n, namely u_i. At this step, $\log_2(c_i)$ bits of u_i are known.

We use the interval Newton iteration described in [10] for the root refinement. Since this method computes each root in interval arithmetics, it is computable to arbitrary precision with a known bound on the error.

We denote by $\widehat{x}$ the value actually computed (i.e., with all rounding errors) for a given "exact" value x, as would be computed with an infinite precision from the beginning of the algorithm. For technical reasons in the error analysis we need to have the quantities $v_i = \frac{1+x'_i}{2}$ computed as rounded to the nearest floating-point number:

$$|\widehat{v_i} - v_i| \leq \frac{1}{2}\mathrm{ulp}(\widehat{v_i}),$$

$$\widehat{x_i} = \circ(\circ(\widehat{v_i} \cdot \widehat{(b-a)}) + \widehat{a}).$$

We will assume that $b-a$ as well as a were computed as rounded to nearest of the correct value. The error analysis for the translated points on $[a,b]$ gives:

$$|\circ(\widehat{v_i}\cdot\widehat{b-a})-v_i\cdot(b-a)|\leq\frac{5}{2}\mathrm{ulp}(\circ(\widehat{v_i}\cdot\widehat{b-a}))$$

using Lemma 5, then

$$\begin{aligned}|\widehat{x_i}-x_i| &\leq \frac{1}{2}\mathrm{ulp}(\widehat{x_i})+\frac{5}{2}\mathrm{ulp}(\circ(\widehat{v_i}\cdot\widehat{b-a}))+\frac{1}{2}\mathrm{ulp}(\widehat{a})\\ &\leq \frac{17}{4}\mathrm{ulp}(\widehat{x_i}). \quad \text{[Lemma 4]}\end{aligned}$$

3.2. Weights

The weights $(w_i)_{0\leq i<n}$ satisfy the equation

$$\int_{-1}^{1}p(x)\mathrm{d}x=\sum_{i=0}^{n-1}w_ip(x_i)$$

for every polynomial of degree $\leq 2n-1$ (see Section 2.3). For $i\in[0,n-1]$ we write $L_i(x)=\prod_{j\neq i}(x-x_j)$. Notice that $L_i(x)=\frac{P_n(x)}{(x-x_i)P_n'(x_i)}$. L_i' has degree $n-2$ so by definition $<L_i',P_n>=\int_{-1}^{1}P_n(x)L_i'(x)\mathrm{d}x=0$:

$$0=[P_n(x)L_i(x)]_{-1}^{1}-\int_{-1}^{1}P_n'(x)L_i(x)\mathrm{d}x.$$

$P_n'L_i$ has degree $2n-1$ so it is integrated exactly by the method:

$$\frac{P_n^2(1)+P_n^2(-1)}{(1-x_i)P_n'(x_i)}=\sum_{j=0}^{n-1}w_jP_n'(x_j)L_i(x_j).$$

From Equation (1) we can see that $|P_n(\pm1)|=1$. Moreover $L_i(x_j)=\delta_{i,j}$ so

$$w_i=\frac{2}{(1-x_i^2)P_n'^2(x_i)}. \tag{2}$$

Since we can compute x_i to arbitrary precision, we can use Equation (2) to compute w_i with arbitrary accuracy as well. Recall that P_n' is known exactly and we can get an error bound on $P_n'(x_i)$ (known as *running error*) using algorithm 5.1 from [5, p. 95]. In the rest of this paper we will assume that each w_i is computed as the rounded to nearest of the exact value:

$$|\widehat{w_i}-w_i|\leq\frac{1}{2}\mathrm{ulp}(\widehat{w_i}).$$

Algorithm 2 Gauss-Legendre integration

INPUT: $\widehat{a},\widehat{b-a},(\widehat{w_i}),f,(\widehat{v_i}),n$ ▷ where w_i are the weights and v_i is defined in §2.4.
OUTPUT: $\widehat{I}$, a p-bit approximation of $\int_a^b f(x)\mathrm{d}x$ with error bounded by Theorem 3.

1: **for** $i\leftarrow 0$ to $n-1$ **do**
2: $\quad t\leftarrow\circ((\widehat{b-a})\cdot\widehat{v_i})$
3: $\quad\widehat{x_i}\leftarrow\circ(t+\widehat{a})$
4: $\quad\widehat{f_i}\leftarrow\circ(f(\widehat{x_i}))$
5: $\quad\widehat{y_i}\leftarrow\circ(\widehat{f_i}\cdot\widehat{w_i})$
6: **end for**
7: $\widehat{S}\leftarrow\mathrm{sum}(\widehat{y_i},i=0\ldots n-1)$ ▷ with Demmel and Hida algorithm [3]
8: $\widehat{D}\leftarrow\circ(\widehat{b-a})/2$
9: **return** $\circ(\widehat{D}\widehat{S})=\widehat{I}$

4. Integration Algorithm

In order to provide an error bound on the numerical result given by the Gauss-Legendre method, we will have a step-by-step look into Algorithm 2.

In addition to the parameters of Algorithm 2 we need an upper bound M_{2n} of $|f^{(2n)}|$ on $[a,b]$; p is the working precision expressed in the number of bits of the mantissa, $\widehat{a}$ and $\widehat{b-a}$ are given as the rounded to nearest floating-point number in the desired precision; M_1 an upper bound of $|f'|$ on $[a,b]$. We will now prove our main theorem:

Theorem 3. *Let $\delta_{\widehat{y_i}}=\frac{11}{4}\mathrm{ulp}(\widehat{y_i})+6M_1\widehat{w_i}\ \mathrm{ulp}(\widehat{x_i})$, where $\widehat{y_i}$, $\widehat{w_i}$ and $\widehat{x_i}$ are defined in Algorithm 2. When computing the numerical quadrature of f using Algorithm 2 with $p\geq 2$ the total error on the result is bounded by:*

$$B_{total}=\frac{21}{4}\mathrm{ulp}(\widehat{I})+\frac{5n}{4}\widehat{D}\cdot\max(\delta_{\widehat{y_i}})+\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}M.$$

In the total error bound $B_{\text{total}}=B_{\text{math}}+B_{\text{rounding}}$ we will distinguish between the bound on the mathematical error B_{math} given in Section 2.3, and the bound on the rounding errors B_{rounding}.

Algorithm 2 can be analyzed in several steps:

1. The computation of $f(x_i)$. We assume we have an implementation of f with an error bounded by 1 ulp on the result with precision p.

   Such implementations of mathematical functions in arbitrary precision with bounded error on the result and even correct rounding for all rounding modes defined in the IEEE 754 standard can be found for example in MPFR [13] for non-trivial functions like exp, sin, arctan and numerous others. The task of providing an implementation with correct rounding (or weaker, with bounded error) for arbitrary functions f is make

possible by the strong semantics of floating-point operations in MPFR but it is not necessarily easy. With the already estimated error on $\widehat{x}_i$ we have:

$$|f(\widehat{x}_i) - f(x_i)| = |f'(\theta_i)(\widehat{x}_i - x_i)|,$$

for some $\theta_i \in [\min(x_i, \widehat{x}_i), \max(x_i, \widehat{x}_i)]$ and with an upper bound on f' we can bound this error absolutely. Let $\widehat{f}_i = \circ(f(\widehat{x}_i))$ be the floating-point number computed. At this step we now have:

$$\begin{aligned}\delta_{\widehat{f}_i} = |\widehat{f}_i - f(x_i)| &\leq |f'(\theta_i)(\widehat{x}_i - x_i)| + \mathrm{ulp}(\widehat{f}_i) \\ &\leq \frac{17}{4} M_1 \cdot \mathrm{ulp}(\widehat{x}_i) + \mathrm{ulp}(\widehat{f}_i).\end{aligned}$$

2. Computation of the $y_i = f(x_i) \cdot w_i$. The accumulated error so far:

$$\begin{aligned}|\widehat{y}_i - y_i| &\leq \frac{1}{2}\mathrm{ulp}(\widehat{y}_i) + |\widehat{f}_i\widehat{w_i} - f(x_i)w_i| \\ &\leq \frac{1}{2}\mathrm{ulp}(\widehat{y}_i) + \\ &\quad \widehat{f}_i|\widehat{w_i} - w_i| + w_i|\widehat{f}_i - f(x_i)| \\ &\leq \frac{1}{2}\mathrm{ulp}(\widehat{y}_i) + \frac{1}{2}\widehat{f}_i\mathrm{ulp}(\widehat{w_i}) + w_i\delta_{\widehat{f}_i} \\ &\leq \frac{3}{2}\mathrm{ulp}(\widehat{y}_i) + \\ &\quad w_i\left[\frac{17}{4}M_1 \cdot \mathrm{ulp}(\widehat{x}_i) + \mathrm{ulp}(\widehat{f}_i)\right] \\ &\quad \text{[Lemmas 1 and 2]} \\ &\leq \frac{3}{2}\mathrm{ulp}(\widehat{y}_i) + \\ &\quad (1+2^{-p})\widehat{w_i}\frac{17}{4}M_1 \cdot \mathrm{ulp}(\widehat{x}_i) + \\ &\quad (1+2^{-p})\widehat{w_i}\mathrm{ulp}(\widehat{f}_i) \\ &\quad \text{[Lemma 3]} \\ &\leq (\frac{7}{2} + 2^{1-p})\mathrm{ulp}(\widehat{y}_i) + \\ &\quad (1+2^{-p})M_1\widehat{w_i}\frac{17}{4}\mathrm{ulp}(\widehat{x}_i) \\ &\quad \text{[Lemmas 1 and 2]} \\ &\leq (\frac{7}{2} + 2^{1-p})\mathrm{ulp}(\widehat{y}_i) + \\ &\quad \left(\frac{17}{4} + 17 \cdot 2^{-p-2}\right) M_1\widehat{w_i}\mathrm{ulp}(\widehat{x}_i) \\ &= \delta_{\widehat{y}_i}.\end{aligned}$$

Remark: when bounding the error on $\widehat{x}_i$, $\widehat{f}_i$ as well as $\widehat{y}_i$, the term with $\mathrm{ulp}(\widehat{x}_i)$ vanishes if the error on $\widehat{x}_i$ is zero. One can easily show with our assumption that no underflow occurs, and that if $\widehat{x}_i = 0$ then the error on $\widehat{x}_i$ is zero (i.e., $x_i = 0$) and the ill-defined quantity $\mathrm{ulp}(\widehat{x}_i)$ vanishes. For the error bound we keep track of only $\max(\delta_{\widehat{y}_i})$.

3. Summation of the y_i's: this is done with Demmel and Hida summation algorithm [3], which guarantees an error of at most 1.5 ulp on the final result. This algorithm uses a larger working precision $p' \approx p + \log_2(n)$. Let $S = \sum_{i=0}^{n-1} y_i$.

$$|\widehat{S} - S| \leq \frac{3}{2}\mathrm{ulp}(\widehat{S}) + n \cdot \max(\delta_{\widehat{y}_i}).$$

4. Multiplication by $\frac{b-a}{2}$: $I = \frac{b-a}{2}S$. We note $D = \frac{b-a}{2}$ and assume as before that the input $\widehat{b-a}$ was computed as the rounded to nearest of its exact value. Since the division by 2 is exact in binary we have:

$$\begin{aligned}|\widehat{D} - D| &\leq \frac{1}{2}\mathrm{ulp}(\widehat{D}) \\ |\widehat{I} - I| &\leq \frac{1}{2}\mathrm{ulp}(\widehat{I}) + |\widehat{S}\widehat{D} - SD| \\ &\leq \frac{1}{2}\mathrm{ulp}(\widehat{I}) + \frac{1}{2}|\widehat{S}|\mathrm{ulp}(\widehat{D}) + D|\widehat{S} - S| \\ &\leq \frac{3}{2}\mathrm{ulp}(\widehat{I}) + D\frac{3}{2}D\mathrm{ulp}(\widehat{S}) + \\ &\quad nD \cdot \max(\delta_{\widehat{y}_i}) \\ &\quad \text{[Lemmas 1 and 2]} \\ &\leq \frac{3}{2}\mathrm{ulp}(\widehat{I}) + (1+2^{-p})\widehat{D}\frac{3}{2}\mathrm{ulp}(\widehat{S}) + \\ &\quad (1+2^{-p})\widehat{D}n \cdot \max(\delta_{\widehat{y}_i}) \\ &\quad \text{[Lemma 3]} \\ &\leq (\frac{9}{2} + 3 \cdot 2^{-p})\mathrm{ulp}(\widehat{I}) + \\ &\quad n(1+2^{-p})\widehat{D} \cdot \max(\delta_{\widehat{y}_i}). \\ &\quad \text{[Lemmas 1 and 2]}\end{aligned}$$

Corollary 1. *If we assume furthermore that the sign of f does not change on $[a, b]$, then we have the following bound:*

$$\begin{aligned}B'_{total} = &\frac{161}{4}\mathrm{ulp}(\widehat{I}) + \frac{425}{64}nM_1\widehat{D}\max(\widehat{w_i}\mathrm{ulp}(\widehat{x}_i)) \\ &+ \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}M_{2n}.\end{aligned}$$

PROOF: Let us assume for example that $f \geq 0$, knowing that the Gauss-Legendre weights are positive we have

$$\forall i \in [0, n-1], \widehat{y}_i = \circ(\widehat{w_i} \cdot \widehat{f}_i) \geq 0$$

so

$$\mathrm{ulp}(\widehat{y}_i) \leq 2^{1-p}\widehat{y}_i.$$

Let $\tilde{S} = \sum_{i=0}^{n-1} \widehat{y_i}$, we know that

$$\begin{aligned}
|\tilde{S} - \widehat{S}| &\leq \frac{3}{2}\mathrm{ulp}(\widehat{S}) \\
\tilde{S} &\leq (1 + 3 \cdot 2^{-p})\widehat{S} \\
L &= \sum_{i=0}^{n-1}(\frac{7}{2} + 2^{1-p})\mathrm{ulp}(\widehat{y_i}) \\
&\leq (\frac{7}{2} + 2^{1-p})2^{1-p}\sum_{i=0}^{n-1}\widehat{y_i} \\
&\leq 2^{1-p}(\frac{7}{2} + 2^{1-p})(1 + 3 \cdot 2^{-p})\widehat{S} \\
&\leq (7 + 2^{2-p})(1 + 3 \cdot 2^{-p})\mathrm{ulp}(\widehat{S}).
\end{aligned}$$

From this we get the following bound on the error on $\widehat{S}$:

$$\begin{aligned}
|\widehat{S} - S| \leq &\left(\frac{3}{2} + (7 + 2^{1-p})(1 + 3 \cdot 2^{-p})\right)\mathrm{ulp}(\widehat{S}) \\
&+nM_1\left(\frac{17}{4} + 17 \cdot 2^{-p-2}\right)\max(\widehat{w_i}\mathrm{ulp}(\widehat{x_i}))
\end{aligned}$$

and substituting this expression in the bound of $|\widehat{I} - I|$ above yields the announced result. □

5. Experiments: a complete example

Algorithm 2 was implemented using the MPFR library [13]. In addition to the result of the integration, the program gives the error bounds B_{math} and B_{rounding} on the mathematical and rounding errors, respectively.

We give now as an example how to use our algorithm to compute an accurate value for the integral given in the introduction, namely:

$$I = \int_{17}^{42} e^{-x^2}\log x \mathrm{d}x.$$

Let $f(x) = e^{-x^2}\log x$. We need to provide a bound on the derivatives of f on $[a, b] = [17, 42]$. This bound should be sharp enough as it will dictate the order n of the method used for a given target precision (see [12, p. 181]).

We note

$$\begin{aligned}
g(x) &= e^{-x^2} \\
h(x) &= \log x.
\end{aligned}$$

Leibniz's formula gives

$$f^{(n)}(x) = \sum_{i=0}^{n}\binom{n}{i}\frac{\mathrm{d}^i}{\mathrm{d}x^i}g(x)\frac{\mathrm{d}^{n-i}}{\mathrm{d}x^{n-i}}h(x).$$

For $i \geq 1$ we can write

$$h^{(i)}(x) = (-1)^{i+1}(i-1)!x^{-i}.$$

The derivatives of g need more work, but we can write

$$g^{(i)}(x) = G_i(x)e^{-x^2}$$

where $G_i(x)$ is a polynomial and

$$\begin{aligned}
G_0 &= 1 \\
G_{i+1} &= -2xG_i(x) + G_i'(x) \text{ for } i \geq 0. \qquad (3)
\end{aligned}$$

From Equation (3) we see that G_i is an integer polynomial of degree i and has only monomials of the same parity as i. Furthermore the leading coefficient of G_i is $(-2)^i$.

We will now prove by recurrence that for $i \geq 0$ the coefficients of G_i are bounded in absolute value by $(i+1)!$.

The property is true for $G_0(x) = 1$. Assume the property true for some $i \geq 0$ and write

$$\begin{aligned}
G_i(x) &= \sum_{j=0}^{i} a_j x^j \\
G_{i+1}(x) &= \sum_{j=0}^{i+1} b_j x^j.
\end{aligned}$$

For $j \leq i-1$ we have

$$\begin{aligned}
b_j &= -2a_{j-1} + (j+1)a_{j+1} \\
|b_j| &\leq 2(i+1)! + (j+1)(i+1)! \\
&\leq (j+3)(i+1)! \\
&\leq (i+2)!.
\end{aligned}$$

Since $b_i = 0$ and $|b_{i+1}| = 2^{i+1} < (i+2)!$ the property holds for $i+1$.

For $n \geq 0$ and $x \in [17, 42]$ we know that

$$|G_n(x)| \leq n \cdot (n+1)!x^n.$$

We may now bound $|f^{(n)}|$ as follows:

$$\begin{aligned}
|f^{(n)}(x)| &\leq |G_n(x)|e^{-x^2}\log x \\
&\quad + \sum_{i=0}^{n-1}\binom{n}{i}|G_i(x)|e^{-x^2}(n-i-1)!x^{i-n} \\
&\leq n \cdot (n+1)!x^n e^{-x^2}\log x + \\
&\quad n!\sum_{i=0}^{n-1}\frac{i(i+1)}{n-i}x^{2i-n}e^{-x^2} \\
&\leq n \cdot n!e^{-x^2}\left((n+1)x^n\log x + (n-1)x^{n-2}\right).
\end{aligned}$$

In particular the following bound is valid for $x \in [17, 42]$:

$$|f^{(n)}| \leq n \cdot n!e^{-17^2}\left((n+1)42^n\log\ 42 + (n-1)42^{n-2}\right).$$

Using this bound we have computed the value of I with our algorithm and several choices of working precisions p:

p	m	n_{opt}	p_{pre}	p_{meas}	t	τ
53	16	20	27	37	8	50
113	16	35	87	103	16	80
200	16	54	174	193	96	55
500	32	80	474	498	404	34
1000	32	142	974	998	620	37
2000	32	254	1974	1994	2952	44
5000	32	556	4974	4995	32818	51

Figure 1. Optimized order n_{opt}, predicted precision p_{pred} and measured precision p_{meas} for different working precisions p in bits and orders m of composition. The timings t are given in ms (with τ the percentage of total time taken by the weights computation) and were done on a 2.4GHz AMD Opteron™ 250 processor.

53 bits and 113 bits to reproduce the double and quad precision, and precisions 200, 500, 1000, 2000 and 5000 bits to observe the behaviour of our algorithm in higher precision. For several orders m of composition doubling at each step, we seek to find the smallest value of the number of points n for which the bound B_{math} on the mathematical error is smaller than the bound B_{rounding} on the rounding errors (see Figure 1). This value of n is considered optimal in the sense that increasing it will decrease B_{math} with no benefit in the guaranteed accuracy since B_{rounding} will increase, and using a smaller value of n means that we are using too high a working precision. For each set of parameters we give the number of good bits predicted by the software, and the number of bits actually correct, as measured against a value that is assumed to be accurate to a precision higher than what we will require afterwards. This reference value was computed with a precision $p = 5200$ bits using the 911-points Gauss-Legendre quadrature composed 8 times. For this set of parameters our algorithm gives

$$\begin{aligned} B_{\text{math}} &\leq 2^{-5599} \\ B_{\text{rounding}} &\leq 2^{-5594} \\ B_{\text{total}} &\leq 2^{-5593} \end{aligned}$$

and a value $v \approx 1.011 \cdot 2^{-421}$ in binary, so the computed value is accurate to about $5593 - 421 = 5172$ bits of relative precision, which is enough for our experiments.

The result of this experiment is given in Figure 1. For a given working precision p we noticed that for several orders of composition m the number of predicted good bits is the same (when we pick the optimal order n of the method) so we kept only the line with the best running time.

In order to study how good the different error bounds are, we chose to compute I with a working precision of $p = 1000$ bits and an order of composition $m = 8$ and vary the number of points n of the method.

The results are given in Figure 2 for a comparison of the predicted error bound and the measured error, and Figure 3 for a comparison of the rounding error bound and the mathematical error bound.

Looking at Figure 1 we see that when we use the optimal number of points n, the accuracy actually achieved is very close to the working precision: in other words, almost all bits are correct (except for $p = 53$ bits). The gap between the number of bits predicted to be correct and the number of bits measured to be correct (what we may call our "pessimism factor") is stable at about 25 bits. We may consider for example that for a working precision of 2000 bits a loss of 20 in the number of predicted good bits (i.e., 1% of the working precision) is satisfactory.

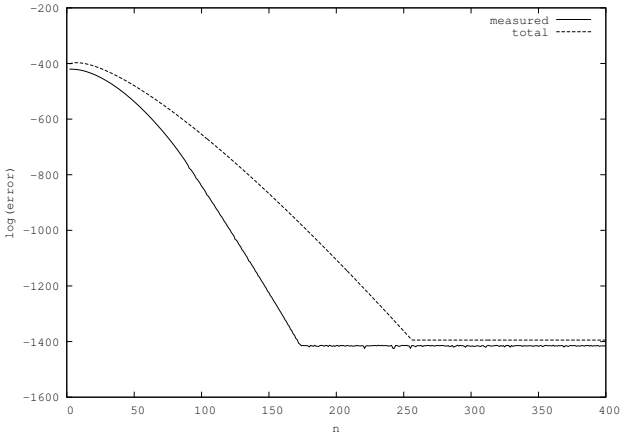

Figure 2. The bound on the total error B_{total} and the measured error when computing I with $m = 8$ and 1000 bits of precision, for several numbers of points n of the method.

Looking at Figures 3 and 2 we observe that our pessimism stems from the bound on the mathematical error B_{math}. As soon as $B_{\text{math}} \leq B_{\text{rounding}}$ the number of predicted good bits follows closely the number of bits measured correct. Our interpretation is that the estimation of the rounding error bound is quite good. Because of the overestimation of the mathematical error, our algorithm finds the value $n_{\text{opt}} = 254$ where $n \approx 175$ would have been enough. Considering the cost of computing the coefficients of the Gauss-Legendre method which is quadratic in n, we may again consider the performance of the experiment to be satisfactory, considering how little work was needed to establish B_{math}.

For the parameters used in Figure 1 the coefficients computing time is about half of the full running time (Figure 4) of the quadrature algorithm, which is expected since we kept only the best composition order for each precision. We tried only powers of 2 as composition orders, but it is expected that the percentage is closer to 50% when the experiment is done over all possible (m, n) parameters. It is also

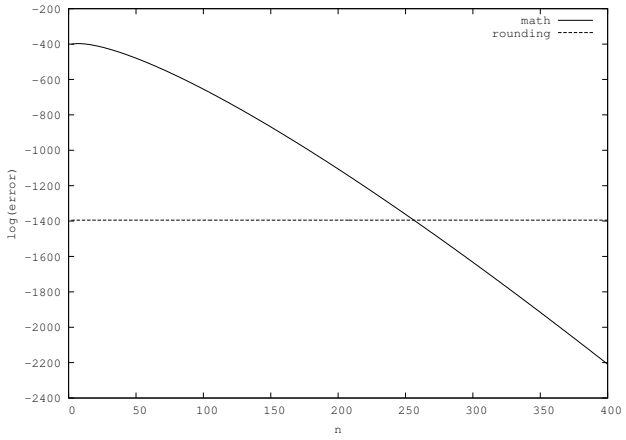

Figure 3. The bounds B_{rounding} on the rounding error and B_{math} on the mathematical error when computing I with $m = 8$ and 1000 bits of precision, for several numbers of points n of the method.

possible to use precomputed values for these coefficients.

As for the actual value of I computed, we get

$$I \approx 0.256572850056 \cdot 10^{-126}$$

which means that the first value v_1 given by Maple 10 had one correct digit out of ten displayed.

Our source code will be released under the GNU LGPL within a few months.

6. Conclusion

The Gauss-Legendre quadrature scheme provides a robust numerical integration algorithm, in the sense that an increase in the order of the method results usually in an increase in the accuracy of the results. This is not true of the Newton-Cotes quadrature scheme for example, where the stability suffers from coefficients of different signs for $n \geq 8$, if the working precision is not increased accordingly.

Providing the function f is sufficiently smooth on a finite integration domain $[a, b]$ and bounds on its derivatives are known, we were able in this paper to propose a quadrature algorithm with a complete error analysis. Our bound on the final error is valid for any precision or order of the method, and since it is an actual bound and not a mere estimate we do in fact compute an interval containing the true value of the integral.

As future work we consider an adaptation of our error bound when using an adaptive quadrature scheme. If the bounds on the derivatives of f are known not only globally for the whole interval but more precisely for sub-intervals, we may be able to use automatically a higher composition order on specific sub-intervals, as needed. We are also interested to see how this kind of error bounds could be given for the double exponential integration [8].

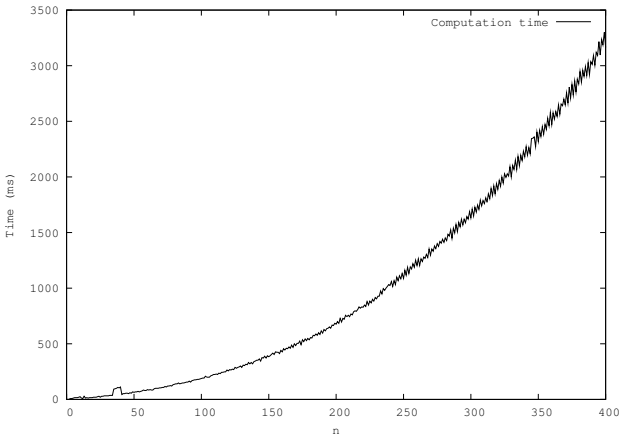

Figure 4. Total computing time in milliseconds when computing I with $m = 8$ and 1000 bits of precision, on a 2.4GHz AMD Opteron™ 250 processor.

References

[1] D. H. Bailey, X. S. Li, and K. Jeyabalan. A comparison of three high-precision quadrature schemes. *Experimental Mathematics*, 14(3):317–329, 2005.

[2] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Academic Press, New York, 2nd edition, 1984.

[3] J. Demmel and Y. Hida. Accurate and efficient floating point summation. *SIAM J. Sci. Comput.*, 25(4):1214–1248, 2003.

[4] W. Gautschi. Algorithm 726; ORTHPOL–a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules. *ACM Trans. Math. Softw.*, 20(1):21–62, 1994.

[5] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2nd edition, 2002.

[6] IEEE standard for binary floating-point arithmetic. Technical Report ANSI-IEEE Standard 754-1985, New York, 1985. approved March 21, 1985: IEEE Standards Board, approved July 26, 1985: American National Standards Institute, 18 pages.

[7] F. Jézéquel, F. Rico, J.-M. Chesneaux, and M. Charikhi. Reliable computation of a multiple integral involved in the neutron star theory. *Math. Comput. Simul.*, 71(1):44–61, 2006.

[8] M. Mori. Discovery of the Double Exponential Transformation and Its Developments. *Publ. RIMS*, 41:897–935, 2005.

[9] W. Oevel. Numerical computations in MuPAD 1.4. *mathPAD*, 8(1), 1998.

[10] N. Revol. Interval newton iteration in multiple precision for the univariate case. *Numerical Algorithms*, 34(2-4):417–426, dec 2003.

[11] F. Rouillier and P. Zimmermann. Efficient isolation of a polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[12] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer Verlag, 3rd edition, 2002.

[13] The Spaces project. The MPFR library, version 2.2.0. `http://www.mpfr.org/`, 2005.