A Comparison of Three High-Precision Quadrature Schemes

David H. Bailey
Lawrence Berkeley National Laboratory
Berkeley, CA, USA 94720
dhbailey@lbl.gov

Xiaoye S. Li
Lawrence Berkeley National Laboratory, Berkeley
Berkeley, CA, USA 94720
xsli@lbl.gov

Experimental Math Workshop, March 29-30, 2004

Outline

- History and motivation
- ARPREC – a new freely available arbitrary precision package
- Three high precision quadrature schemes

  Gaussian, Error function, Tanh-sinh
- Comparison results

History of numerical quadrature

- 1670: Newton devises what is now known as Newton-Coates integration.
- 1740: Thomas Simpson develops Simpson's rule.
- 1820: Gauss develops Gaussian quadrature.
- 1900s: Adaptive quadrature, Romberg integration, Clenshaw-Curtis integration, others.
- 1990: *Maple* and *Mathematica* feature built-in numerical quadrature schemes.

Almost all published works focus on obtaining results accurate to 15 digits or less, the limit of present-day IEEE "double" arithmetic.

Why high precision quadrature?

- Recognize an unknown definite integral in analytic terms, using high-precision quadrature and PSLQ integer relation detection.

  Example [Jon Borwein, Greg Fee and D.H. Bailey]
  If

$$C(a) = \int_0^1 \frac{\arctan(\sqrt{x^2+a^2})\,dx}{\sqrt{x^2+a^2}(x^2+1)}$$

  then

$$\begin{aligned} C(0) &= \pi\log 2/8 + G/2 \\ C(1) &= \pi/4 - \pi\sqrt{2}/2 + 3\sqrt{2}\arctan(\sqrt{2})/2 \\ C(\sqrt{2}) &= 5\pi^2/96 \end{aligned}$$

  where G is Catalan's constant (the third result appeared in the *MAA Monthly*, Aug/Sept 2002). These particular results have now led to several general results, including:

$$\int_0^\infty \frac{\arctan(\sqrt{x^2+a^2})\,dx}{\sqrt{x^2+a^2}(x^2+1)} = \frac{\pi}{2\sqrt{a^2-1}}\left[2\arctan(\sqrt{a^2-1}) - \arctan(\sqrt{a^4-1})\right]$$

Why high precision quadrature?

- Identification of infinite series

Recently Borwein and Roland Girgensohn of Zentrum Mathematik in Germany were able to derive an analytic representation for sums of the form

$$b_2(k) = \sum_{n\geq 1} \frac{n^k}{\binom{2n}{n}}$$

$$b_3(k) = \sum_{n\geq 1} \frac{n^k}{\binom{3n}{n}2^n}$$

by transforming them into the integral expressions

$$b_2(k) = \sum_j c_j^k(2) \int_0^1 [1 - x(1-x)]^{-j}\, dx$$

$$b_3(k) = \sum_j c_j^k(3) \int_0^1 [1 - x^2(1-x)]^{-j}\, dx$$

where the c function involves certain combinatorial values. They then identified these two integrals by using a high-precision quadrature routine, similar to those described here, combined with an integer relation detection facility.

Why not *Maple* and *Mathematica*?

The integrals

$$I_1 = \int_0^1 \frac{t^2 \ln(t)\, dt}{(t^2-1)(t^4+1)}$$
$$I_2 = \int_0^{\pi/4} \frac{t^2\, dt}{\sin^2(t)}$$
$$I_3 = \int_0^{\pi} \frac{x \sin x\, dx}{1+\cos^2 x}$$

are successfully evaluated by *Maple* and *Mathematica*, but the results are lengthy expressions involving advanced functions and complex entities (in one case, over 100 lines long!).

Using the techniques in this paper, we were able to determine that:

$$I_1 = \pi^2(2-\sqrt{2})/32$$
$$I_2 = -\pi^2/16 + \pi \ln(2)/4 + G$$
$$I_3 = \pi^2/4,$$

High precision quadrature facility

Goals

- Given only the function definition in a user program
- No *a priori* knowledge of the specific function to be integrated
- No bounds on the magnitude of the function or any of its derivatives
- Does not rely on symbolic manipulation
- Suitable for parallel implementation

Three schemes in consideration

- Gaussian quadrature
- Error function quadrature
- Tanh-sinh quadrature

Last two based on the Euler-MacLaurin Summation Formula

The arbitrary precision (ARPREC) package

- Improved upon Bailey's MPFUN.
- Implemented in C++ for high performance and broad portability; thread-safe.
- Fortran-90 modules that permit Fortran-90 programs to utilize the package with only very minor changes to source code.
- Arbitrary precision integer, floating and complex datatypes, and dynamically changing precision.
- Support for datatypes with differing precision levels.
- Inter-operability with conventional integer and floating-point datatypes.
- Common transcendental functions (sqrt, exp, sin, erf, etc).
- Quadrature routines (for numerical integration).
- PSLQ routines (for integer relation detection).
- Special routines for extra-high precision (> 1000 digits) computation.

Experimental Mathematician's Toolkit:

- An interactive utility incorporating the above facilities.

More on ARPREC implementation

- An n-"digits" number A is represented in an array of $(n+5)$-long words $(a_k, 1 \le k \le n+5)$:
  - a_1 : array size
  - a_2 : number of mantissa words, with sign
  - a_3 : exponent
  - $(a_4 : a_{n+3})$: mantissa words
  - a_{n+4}, a_{n+5} : 2 scratch words for rounding purpose

  $A = \pm(a_4\beta^{a_3} + a_5\beta^{a_3-1} + \cdots + a_{n+3}\beta^{a_3-n+1})$
- Exploit IEEE 64-bit FP word: use 48 bits for a_i, so $\beta = 2^{48}$, $a_i \in [0, 2^{48})$
- Multiplication employs $double \times double$ arithmetic for each component

  Can do 32 rows of mults before releasing carries $(32 = 2^5, 5 + 48 = 53)$
- Taylor series for transcendental functions, with proper argument reduction
- FFT-based multiplication for extra-high precision numbers
- Some routines (e.g., `sqrt`) use Newton algorithm
- 30-40% faster than MPFUN for low level arithmetics

Gaussian quadrature

An integral on $[-1, 1]$ is approximated as the sum

$$\int_{-1}^{1} f(x)\,dx \;\approx\; \sum_{j=0}^{n} w_j f(x_j)$$

- abscissas x_j are roots of the n-th degree Legendre polynomial $P_n(x)$ on $[-1, 1]$
- weights $w_j = \frac{-2}{(n+1)P_n'(x_j)P_{n+1}(x_j)}$

How are they computed?

- abscissas x_j are computed using a Newton iteration scheme, with starting value $\cos[\pi(j - 1/4)/(n + 1/2)]$ [G. Rybicki].
- $P_n(x)$ is computed using an n-long iteration of the recurrence

  $$(k+1)P_{k+1}(x) \;=\; (2k+1)xP_k(x) - kP_{k-1}(x), \text{ for } k \geq 2,$$

  with $P_0(x) = 0,\; P_1(x) = 1.$
- $P_n'(x)$ is computed as

  $$P_n'(x) \;=\; n(xP_n(x) - P_{n-1}(x))/(x^2 - 1)$$

Drawback: cost increases quadratically with n

High precision implementation

Multiple "levels" (phases) are employed ... Starting with $n_0 = 3$, and with $n_k = 2 \cdot n_{k-1}$, up to level m. Total $n = \Sigma_{k=0}^{m} 3 \cdot 2^k \approx 6 \cdot 2^m$.
(we set $m = 9$ in the experiments.)

- Initialization
  The Newton iteration for x_j is accelerated by using a dynamic level of precision: start with 64-bit, roughly doubles the precision for each level up.
- Integral evaluation
  Start with the first level, continue with additional levels but evaluate function only at the odd-indexed abscissas, until:
  - either exhaust the set of pre-computed abscissas and weights,
  - or accuracy satisfied.

The Euler-Maclaurin formula

Let $h = (b-a)/n$ and $x_j = a + jh$. Then

$$\int_a^b f(x)\,dx = h\sum_{j=1}^{n-1} f(x_j) - \frac{h}{2}\left(f(a)+f(b)\right) - \sum_{i=1}^{m}\frac{h^{2i}B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right) - E$$

The error term E is smaller than $h^{2m+1}/(2m+2)!$ times a constant (certain definite integral that is independent of n and h.)

Special cases

- $f(x)$ and all of its derivatives are zero at a and b. (e.g., a smooth, bell-shaped function)

  $$\int_a^b f(x)\,dx = h\sum_{j=1}^{n-1} f(x_j) - E$$

  That is, in a simple step-function approximation, error goes to zero more rapidly than any power of h, as h is decreased.

- Infinite interval, where $f(x)$ and all its derivatives tend to zero for large $|x|$, the E-M formula also applicable

  $$\int_{-\infty}^{\infty} f(x)\,dx = h\sum_{j=-\infty}^{\infty} f(x_j) - E$$

Error function and tanh-sinh quadratures

In E-M, change of variable $x = g(t)$, $g(t)$ is a monotonic function satisfying:

- $\lim_{t\to\infty} g(t) = 1$, $\lim_{t\to-\infty} g(t) = -1$,
- $g'(t)$ and higher derivatives approach to zero for large $|t|$.

Then, for $h > 0$, we have

$$\int_{-1}^{1} f(x)\,dx = \int_{-\infty}^{\infty} f(g(t))g'(t)\,dt \approx h \sum_{-N}^{N} w_j f(x_j), \text{ where } , w_j = g'(hj), x_j = hj$$

1. Error function quadrature
   - $g(t) = \mathrm{erf}(t) = (2/\sqrt{\pi}) \int_0^t e^{-x^2}\,dx \ \in (-1,1)$,
   - $g'(t) = 2/\sqrt{\pi} \cdot e^{-t^2}$, like probability density function

   For many $f(x) \in C^\infty(-1,1)$, the integrand $f(g(t))g'(t)$ is a nice bell-shaped curve for which the E-M formula applies.

2. Tanh-sinh quadrature
   - $g(t) = \tanh(\pi/2 \cdot \sinh t) \ \in (-1,1)$,
   - $g'(t) = \pi/2 \cdot \sinh t / \cosh^2(\pi/2 \cdot \sinh t)$

Convergence and error estimation

- For bounded and well behaved functions on a finite closed interval, all three quadrature schemes exhibit quadratic convergence: after a few initial levels, subsequent levels produce approximations about twice the number of correct digits as the previous level.
- Error estimation attempts to predict the error of the current iteration based on the error improvement from the previous two iterations. The following works well in practice:
  Let S_k be the computed integral for levels k to n, then the estimated error E_l at level l is given by

$$E_l = \begin{cases} 1 & \text{if } l \leq 2 \\ 0 & \text{or } S_l = S_{l-1} \\ 10^d & \text{otherwise} \end{cases}$$

where, $d = \min[0, \max(d_1^2/d_2, 2d_1, d_3)]$,
$d_1 = \log_{10} |S_l - S_{l-1}|$,
$d_2 = \log_{10} |S_l - S_{l-2}|$,
$d_3 = \log_{10} |\varepsilon \cdot \max_{j \in \text{level } l} |f(x_j)|$,
$\varepsilon =$ "machine epsilon" of the multiprecision system.

Experiments

- User working precision set to 400 digits
- Target accuracy set to 10^{-390}
- 9 levels of abscissas and weights pre-computed
- Termination criteria
  - case 1: maximum level (9) is reached, or
  - case 2: estimated error E_l achieved target accuracy, or
  - case 3: magnitude of the function near an endpoint is too large, so the summands are not sufficiently small to ensure full accuracy (higher numeric precision is required)

Test problems

Set 1. Well-behaved continuous functions on finite itervals:

$$1: \int_0^1 t\log(1+t)\,dt = 1/4 \qquad 2: \int_0^1 t^2\arctan t\,dt = (\pi-2+2\log 2)/12$$

$$3: \int_0^{\pi/2} e^t\cos t\,dt = (e^{\pi/2}-1)/2 \qquad 4: \int_0^1 \frac{\arctan(\sqrt{2+t^2})}{(1+t^2)\sqrt{2+t^2}}\,dt = 5\pi^2/96$$

Set 2. Continuous functions on finite itervals, but with a vertical derivative at an endpoint:

$$5: \int_0^1 \sqrt{t}\log t\,dt = -4/9 \qquad 6: \int_0^1 \sqrt{1-t^2}\,dt = \pi/4$$

Set 3. Functions on finite intervals with an integrable singularity at an endpoint:

$$7: \int_0^1 \frac{t}{\sqrt{1-t^2}}dt = 1 \qquad 8: \int_0^1 \log t^2\,dt = 2$$

$$9: \int_0^{\pi/2} \log(\cos t)\,dt = -\pi\log(2)/2 \qquad 10: \int_0^{\pi/2} \sqrt{\tan t}\,dt = \pi\sqrt{2}/2$$

Set 4. Functions on an infinite interval:

$$11: \int_0^\infty \frac{1}{1+t^2}\,dt = \pi/2 \qquad 12: \int_0^\infty \frac{e^{-t}}{\sqrt{t}}\,dt = \sqrt{\pi}$$

$$13: \int_0^\infty e^{-t^2/2}\,dt = \sqrt{\pi/2}$$

Set 5. Oscillatory functions on an infinite interval:

$$14: \int_0^\infty e^{-t}\cos t\,dt = 1/2 \qquad 15: \int_0^\infty \frac{\sin t}{t}\,dt = \pi/2$$

Performance of quadrature routines

	QUADGS			QUADERF			QUADTS		
Prob.	Level	Time	Error	Level	Time	Error	Level	Time	Error
Init	9	2778.29		9	131.80		9	45.46	
1	6	8.72	10^{-406}	9	57.43	10^{-406}	7	13.69	10^{-390}
2	6	8.86	10^{-406}	9	36.17	10^{-406}	8	21.86	10^{-406}
3	5	4.16	10^{-405}	9	44.06	10^{-405}	7	12.01	10^{-405}
4	6	8.78	10^{-405}	9	92.48	10^{-406}	8	38.43	10^{-406}
5	9	78.00	10^{-11}	9	68.15	10^{-406}	7	16.08	10^{-406}
6	9	3.65	10^{-12}	9	3.94	10^{-406}	7	0.90	10^{-392}
7	9	4.39	10^{-4}	8	2.39	10^{-210}	6	0.55	10^{-196}
8	9	75.84	10^{-6}	9	65.58	10^{-405}	7	15.29	10^{-400}
9	9	99.83	10^{-7}	9	69.96	10^{-403}	7	17.97	10^{-390}
10	9	31.68	10^{-4}	8	7.49	10^{-203}	6	2.36	10^{-194}
11	7	0.61	10^{-405}	9	3.04	10^{-255}	9	2.59	0
12	9	47.55	10^{-4}	9	18.88	10^{-132}	9	26.09	10^{-203}
13	9	41.74	10^{-353}	9	11.30	10^{-91}	9	17.97	10^{-241}
14	9	71.69	10^{-126}	9	26.87	10^{-68}	9	44.39	10^{-165}
15	5/9	34.45	10^{-19}	9/9	41.14	10^{-17}	7/9	31.66	10^{-19}

Summary

- For well-behaved continuous functions on finite intervals, Gaussian quadrature is clearly the most efficient.
  - For Set 1, it is 10x faster than Error function, and 4-5x faster than Tanh-sinh
- When the function has a "blow-up" singularity or vertical derivative at an endpoint, Gaussian quadrature fails.
- A major drawback of Gaussian quadrature is very long initialization time, growing as n^2, where n is the number of grid points.
- Error function quadrature performs very well, even in cases where the function is not well behaved at endpoints.
- Tanh-sinh quadrature also performs well when the function is not well behaved, and has the lowest overall run times.
- For problems 7 & 10, both error function and tanh-sinh quadratures stopped because of case 3 termination criterion.

Future work

- Parallelization on SMP, fairly straightforward
- Extend to 2D and 3D quadratures, also amenable to parallel computing