Numerical integration in arbitrary-precision ball arithmetic

Fredrik Johansson
(LFANT, Bordeaux)

AriC seminar, LIP, ENS Lyon
22 March 2018

Ball arithmetic

Floating-point arithmetic:

`3.1415926535897932384626433832795028842`

`3.1415926535897932384626448118509314556` ???

Ball arithmetic:

`[3.1415926535897932384626433832795028842 +/- 1.65e-38]`

`[3.14159265358979323846264 +/- 4.82e-24]`

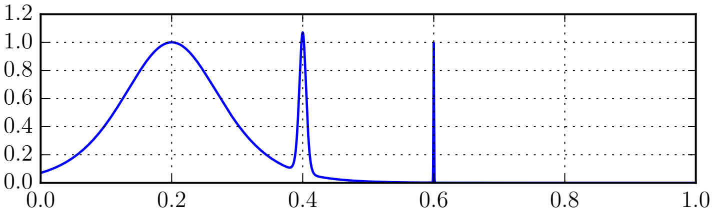
Arb – `http://arblib.org`

- Open source C library
- New code for numerical integration since November 2017 (paper: `https://arxiv.org/abs/1802.07942`)

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

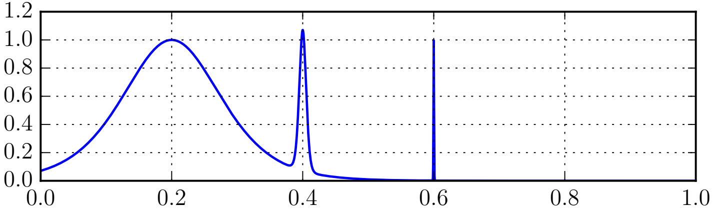
(Test problem by Cranley and Patterson, 1971)



Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)

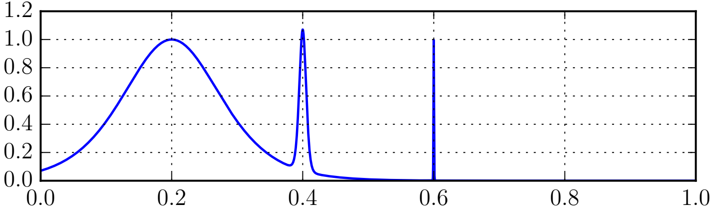


Mathematica `NIntegrate`: 0.209736

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)



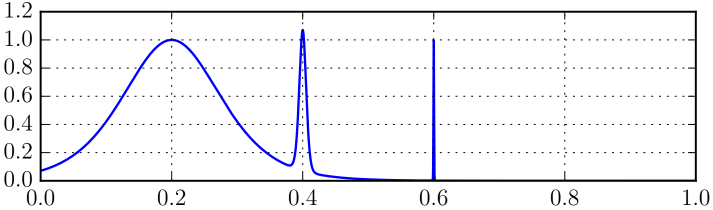
Mathematica `NIntegrate`: 0.209736

Octave `quad`: 0.209736, error estimate 10^{-9}

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)

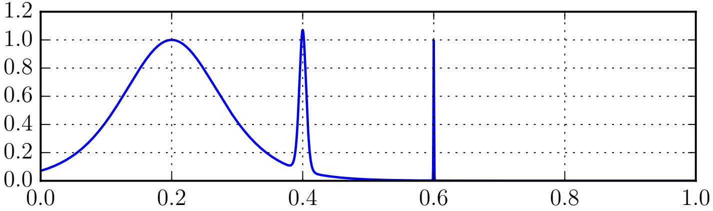


Mathematica `NIntegrate`:	0.209736
Octave `quad`:	0.209736, error estimate 10^{-9}
Sage `numerical_integral`:	0.209736, error estimate 10^{-14}

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)

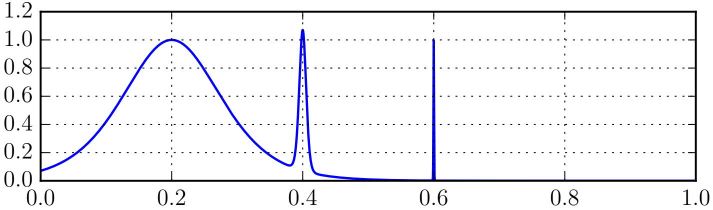


Mathematica `NIntegrate`:	0.209736
Octave `quad`:	0.209736, error estimate 10^{-9}
Sage `numerical_integral`:	0.209736, error estimate 10^{-14}
SciPy `quad`:	0.209736, error estimate 10^{-9}

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)

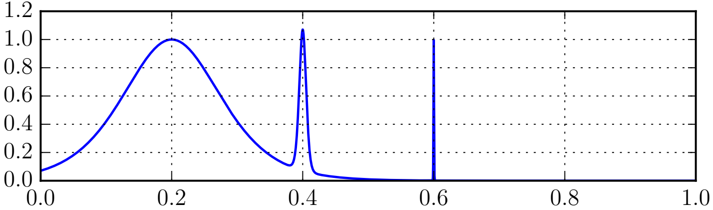


Mathematica `NIntegrate`:	0.209736
Octave `quad`:	0.209736, error estimate 10^{-9}
Sage `numerical_integral`:	0.209736, error estimate 10^{-14}
SciPy `quad`:	0.209736, error estimate 10^{-9}
mpmath `quad`:	0.209819

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)

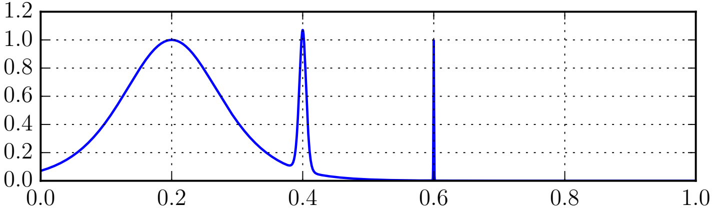


Mathematica `NIntegrate`:	0.209736
Octave `quad`:	0.209736, error estimate 10^{-9}
Sage `numerical_integral`:	0.209736, error estimate 10^{-14}
SciPy `quad`:	0.209736, error estimate 10^{-9}
mpmath `quad`:	0.209819
Pari/GP `intnum`:	0.211316

Example: a nice and smooth function

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$

(Test problem by Cranley and Patterson, 1971)



Mathematica `NIntegrate`:	0.209736
Octave `quad`:	0.209736, error estimate 10^{-9}
Sage `numerical_integral`:	0.209736, error estimate 10^{-14}
SciPy `quad`:	0.209736, error estimate 10^{-9}
mpmath `quad`:	0.209819
Pari/GP `intnum`:	0.211316
Actual value:	0.210803

Results with Arb

64-bit precision:

```
[0.21080273550054928 +/- 4.43e-18]      # time 0.005 s
```

333-bit precision:

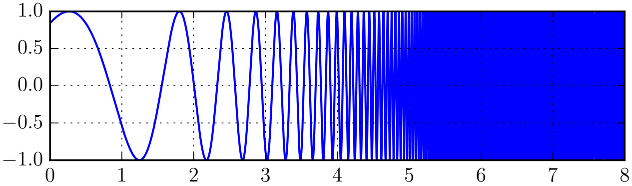
```
[0.2108027355005492773756... +/- 3.73e-99]    # 0.04 s
```

3333-bit precision:

```
[0.2108027355005492773756... +/- 1.39e-1001]  # 9 s
                                            (11 s first time)
```

Another example: violent oscillation

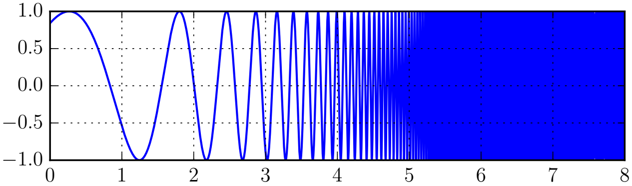
$$\int_0^8 \sin(x+e^x)\,dx$$



- S. Rump (2010) noticed that MATLAB's `quad` returned the incorrect `0.2511` after 1 second of computation
- Rump's INTLAB computes the correct enclosure `[0.34740016, 0.34740018]` in about 1 s
- Mahboubi, Melquiond & Sibut-Pinote (2016):
  1 digit in 80 s and 4 digits in 277 s with CoqInterval

Another example: violent oscillation

$$\int_0^8 \sin(x+e^x)dx$$



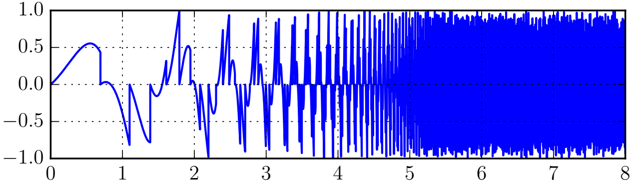
- S. Rump (2010) noticed that MATLAB's `quad` returned the incorrect `0.2511` after 1 second of computation
- Rump's INTLAB computes the correct enclosure `[0.34740016, 0.34740018]` in about 1 s
- Mahboubi, Melquiond & Sibut-Pinote (2016): 1 digit in 80 s and 4 digits in 277 s with CoqInterval

Arb (64, 333, 3333 bits):

```
[0.34740017265725 +/- 3.94e-15]   # 0.005 s
[0.34740017265... +/- 5.98e-96]   # 0.02 s
[0.34740017265... +/- 2.95e-999]  # 1 s  (5 s first time)
```

Yet another example: a monster

$$\int_0^8 (e^x - \lfloor e^x \rfloor) \sin(x + e^x)\, dx$$ – now with 2980 discontinuities!



64-bit precision:

```
[+/- 2.47e+4]                       # time 0.15 s, aborted
[0.0986517044784 +/- 4.74e-14]     # time 9 s
```

333-bit precision:

```
[0.09865170447836520611965824976485985650416962079238449145
10919068308266804822906098396240645824 +/- 6.78e-95]  # 521 s
```

Sage interface to Arb

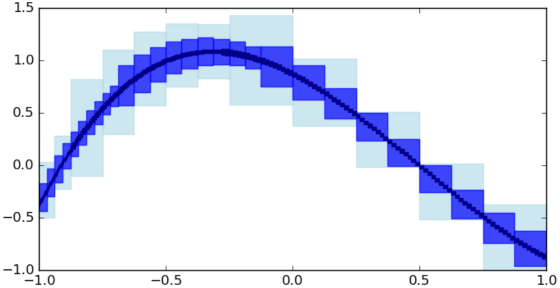
(Thanks to Clemens Heuberger, Marc Mezzarobba, Vincent Delecroix)

```
sage: C = ComplexBallField(333)
sage: C.integral(lambda x, _: sin(x+exp(x)), 0, 8)
[0.347400172657247807879512159119893124657456254866180183
88549271361674821398878532052968510434660 +/- 5.97e-96]
```

- Some overhead: 0.04 s via Sage, 0.02 s using C directly
- Integration will be available in Sage 8.2
- Some functions not yet wrapped
- Also: Julia interface in Nemo (`http://nemocas.org/`)

Brute force interval integration

$$\int_a^b f(x)dx \in (b-a)f([a,b]) \quad + \quad \text{adaptive subdivision of } [a,b]$$



Simple and general method, but need $2^{O(p)}$ evaluations of f for p-bit accuracy when used alone!

Methods with high order convergence

If f is analytic, we can achieve p-bit accuracy with a degree $n = O(p)$ approximation:

Methods of approximation:

- Taylor series truncated to order n
- Quadrature rule with n evaluation points

Error bounds:

- Using derivatives $f^{(n)}$ on $[a, b]$
- Using $|f|$ on a complex domain around $[a, b]$

Practical tradeoff: power series extension (automatic differentiation) vs complex extension of f.

Quadrature rules

Assume f analytic (without singularities close to $[-1,1]$).

$$\int_{-1}^{1} f(x)dx \approx \sum_k w_k f(x_k)$$

Gauss-Legendre

- x_k = roots of Legendre polynomial $P_n(x)$, w_k from $P_n'(x_k)$

Clenshaw-Curtis

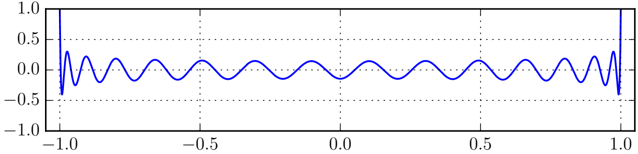
- x_k = Chebyshev nodes $\cos(\pi k/n)$, w_k from FFT
- $\approx 2n$ points equivalent to n-point GL quadrature

Double exponential

- x_k, w_k from change of variables $x = \tanh(\frac{1}{2}\pi \sinh t)$ and trapezoidal approximation $\int_{-\infty}^{\infty} g(t)dt \approx h\sum_{k=-n}^{n} g(hk)$
- $\approx 5n$ points equivalent to n-point GL quadrature

Fast and rigorous computation of GL rules

F.J. and M. Mezzarobba, `https://arxiv.org/abs/1802.03948`



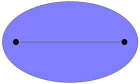
$$P_{30}(x) = \frac{1}{67108864}\Big(7391536347803839x^{30} - 54496920530418135x^{28} + \ldots$$
$$+ 10529425731825x^6 - 347123925225x^4 + 4508102925x^2 - 9694845\Big)$$

- Evaluation of P_n on $[-1,1]$ combining hypergeometric series + rectangular splitting + asymptotic series + three-term recurrence (with error analysis)
- Speedup: $20-60$ s $\to$ $1-3$ s for 1000-digit integration
- With $n \sim p$, complexity is $\widetilde{O}(n^3)$ ($\widetilde{O}(n^2)$ possible in theory)

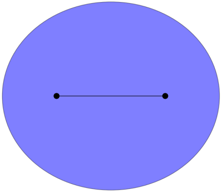
Error bounds for Gauss-Legendre quadrature

If f is analytic with $|f(z)| \leq M$ on an ellipse E with foci $-1, 1$ and semi-axes X, Y with $\rho = X + Y > 1$, then

$$\left| \int_{-1}^{1} f(x)dx - \sum_{k=1}^{n} w_k f(x_k) \right| \leq \frac{M}{\rho^{2n}} \cdot C_\rho$$



$X = 1.25, Y = 0.75, \rho = 2.00$



$X = 2.00, Y = 1.73, \rho = 3.73$

Fast convergence when no singularities are close to $[a, b]$, but should be combined with subdivision otherwise!

Adaptive integration algorithm

1. Compute $(b - a)f([a, b])$. If the error is $\leq \varepsilon$, done!
2. Compute $|f(z)|$ and check analyticity of f on some ellipse E around $[a, b]$. If the error of Gauss-Legendre quadrature is $\leq \varepsilon$, compute it – done!
3. Split at $m = (a + b)/2$ and integrate on $[a, m]$, $[m, b]$ recursively.

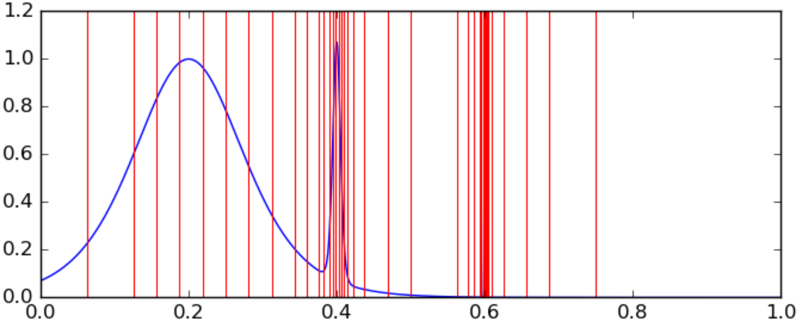
Knut Petras (*Self-validating integration and approximation of piecewise analytic functions*, 2002) pointed out that this guarantees rapid convergence for a large class of functions.

Choosing the quadrature degree n for $[a, b]$

- Try a sequence of increasing ellipses E with convergence rate ρ^{-2n}, $\rho = 3.73, 7.87, \ldots \sim 2^{2^i}, 2^i < p$
- For each tentative E, bound $|f(z)|$ on E and compute the n that would be needed; choose the smallest such n
- If no admissible n is found, GL quadrature is not used and we proceed to bisect $[a, b]$
- $n \leq 0.5p + 60$ by default (can be changed by user)
- n is chosen among $1, 2, 4, 6, 8, 12, 16, 22, 32, 46, \ldots \approx 2^{j/2}$

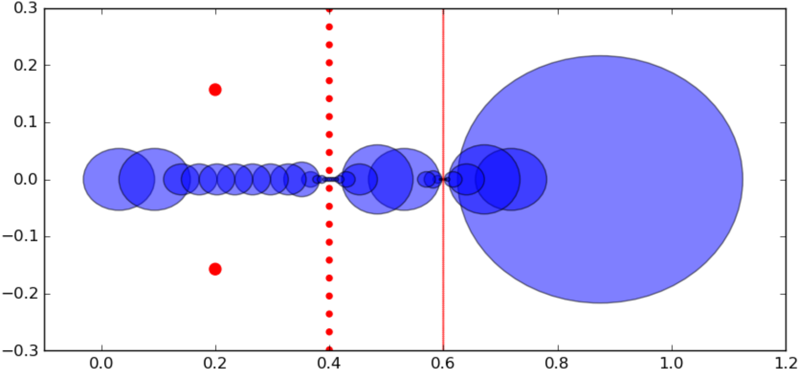
Adaptive subdivision performed by Arb

$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



49 terminal subintervals (smallest width 2^{-12})

Adaptive subdivision, complex view



Blue ellipses used for error bounds on the subintervals

Red dots: poles of the integrand

Some benchmarking

Heuristic arbitrary-precision code, used for comparison:

- Pari/GP: double exponential quadrature, non-adaptive (option to split uniformly into 2^i subintervals)
- mpmath: double exponential quadrature, degree-adaptive (but no bisection)
- Red = default settings give an inaccurate answer (I changed settings as necessary where this happened)

Timings in seconds for Pari/GP, mpmath, Arb.

Statistics for the Arb integration:

- Sub = number of terminal subintervals
- Eval = total number of function evaluations

Smooth integrands

p	Pari/GP	mpmath	Arb	Sub	Eval	Pari/GP	mpmath	Arb	Sub	Eval
	$I_0 = \int_0^1 1/(1+x^2)dx$					$I_1 = \int_0^1 \sum_{k=1}^{3} \operatorname{sech}^{2k}(10^k(x-0.2k))\, dx$				
32	0.00039	0.00057	0.000025	2	32	0.54	1.9	0.0030	49	795
64	0.00039	0.0011	0.000036	2	52	0.54	5.0	0.0051	49	1299
333	0.0043	0.0058	0.00018	2	188	12	38	0.038	49	4891
3333	1.0	0.13	0.014	2	2056	3385	-	8.7	49	48907
	$I_2 = \int_0^\pi x\sin(x)/(1+\cos^2(x))dx$					$I_3 = \int_0^{1000} W_0(x)dx$				
32	0.00077	0.0021	0.00033	14	229	0.0037	0.012	0.00041	12	163
64	0.00077	0.0046	0.00054	14	373	0.0037	0.032	0.00093	12	273
333	0.0088	0.037	0.0040	14	1401	0.052	0.25	0.0099	12	1109
3333	2.2	4.4	1.0	14	14401	11	25	1.3	12	12043
	$I_4 = \int_0^{100} \sin(x)dx$					$I_5 = \int_0^8 \sin(x+e^x)\, dx$				
32	0.0012	0.0019	0.000047	1	53	0.063	0.23	0.0048	33	2115
64	0.0012	0.0014	0.000074	1	72	0.063	0.25	0.0055	27	2307
333	0.015	0.018	0.00030	1	139	0.22	0.58	0.017	22	4028
3333	2.0	0.71	0.032	1	526	14	12	1.1	8	10417
	$I_6 = \int_{-1}^1 e^{-x}\operatorname{erf}\left(\sqrt{1250}\, x + \frac{3}{2}\right) dx$					$I_7 = \int_1^{1+1000i} \Gamma(x)dx$				
32	0.024	0.018	0.0025	7	297	0.031	0.028	0.00076	11	103
64	0.024	0.057	0.0055	6	438	0.054	0.093	0.0035	12	280
333	0.50	0.22	0.047	4	791	0.65	1.1	0.081	14	1304
3333	173	466	5.7	2	2923	561	847	48	14	16535

Endpoint singularities and infinite intervals

Convergence requires $|a|, |b|, |f| < \infty$. Can use manual truncation, e.g. $\int_0^\infty f(x)dx \approx \int_\varepsilon^N f(x)dx$ otherwise.

Integral	Problem	Truncation	Evaluations
$\int_0^\infty e^{-x}dx$	Exponential decay	$N \approx p\log(2)$	$O(p \log p)$
$\int_0^1 \sqrt{1-x^2}dx$	Branch point (f finite)	Not needed	$O(p^2)$
$\int_0^\infty \frac{dx}{1+x^2}$	Algebraic decay	$N \approx 2^p$	$O(p^2)$
$\int_0^1 \log(x)dx$	Branch point (f infinite)	$\varepsilon \approx 2^{-p}$	$O(p^2)$

- Manual truncation is not an ideal solution, but the algorithm is at least robust enough to work with large N or small ε
- $O(p^2)$ cost can be avoided with exponential change of variables
- Future improvement: automatic algorithm, provided that user supplies extra “global” information, e.g. $|f(x)| < x^\alpha e^{\beta x^\gamma}$

Endpoint singularities and infinite intervals

p	Pari/GP	mpmath	Arb	Sub	Eval	Pari/GP	mpmath	Arb	Sub	Eval
	$E_0 = \int_0^1 \sqrt{1-x^2} dx$					$E_1 = \int_0^\infty 1/(1+x^2)\, dx$				
32	0.00041	0.00055	0.00022	22	234	0.00060	0.0010	0.00079	94	997
64	0.00041	0.00067	0.00057	44	674	0.00060	0.0012	0.0022	190	2887
333	0.0044	0.0060	0.015	223	12687	0.0068	0.011	0.048	997	51900
3333	0.94	0.18	6.6	2223	1.2 M	1.7	0.24	27	9997	4.7 M
	$E_2 = \int_0^1 \log(x)/(1+x) dx$					$E_3 = \int_0^\infty \operatorname{sech}(x)\, dx$				
32	0.00081	0.00080	0.00042	34	361	0.0011	0.0019	0.00017	9	144
64	0.00081	0.00094	0.0012	67	1026	0.0011	0.0043	0.00032	10	251
333	0.011	0.011	0.038	336	19254	0.013	0.098	0.0030	14	1277
3333	1.7	1.08	106	3336	1.8 M	3.5	3.3	0.95	17	16593
	$E_4 = \int_0^\infty e^{-x^2+ix} dx$					$E_5 = \int_0^\infty e^{-x} \operatorname{Ai}(-x)\, dx$				
32	0.0014	0.0067	0.00011	1	71	-	0.19	0.0028	4	269
64	0.0014	0.016	0.00018	1	98	-	0.91	0.012	9	842
333	0.017	0.13	0.0016	2	397	-	26	0.94	124	24548
3333	4.7	7.1	0.47	4	3894	-	10167	502	1205	0.7 M

Branch cuts

```
def my_sqrt(z, analytic):
    if analytic and not (z.real() > 0 or z.imag() != 0):
        return CBF(NaN)
    else:
        return sqrt(z)
```

$$\int_1^2 \sqrt{z}\, dz$$

```
sage: CBF.integral(lambda z, _: sqrt(z), 1, 2)    # WRONG!
[1.219007822860045 +/- 7.96e-16]
sage: CBF.integral(my_sqrt, 1, 2)                  # correct
[1.21895141649746 +/- 3.73e-15]
```

$$\int_{-1-i}^{-1+i} \sqrt{z}\, dz$$

```
sage: CBF.integral(my_sqrt, -1 + CBF(i), -1 - CBF(i))
[+/- 1.14e-14] + [-0.4752076627926 +/- 5.18e-14]*I
```

Piecewise and discontinuous functions

Functions like $\lfloor x \rfloor$ and $|x|$ on $\mathbb{R}$ can be extended to piecewise holomorphic functions on $\mathbb{C}$.

$$f(x) = |x| \to f(x+yi) = \sqrt{(x+yi)^2} = \begin{cases} x+yi & x > 0 \\ -(x+yi) & x < 0 \end{cases}$$

(discontinuous at $x = 0$)

$f(x) = \lfloor x \rfloor \to f(x+yi) = \lfloor x \rfloor$ (discontinuous at $x \in \mathbb{Z}$)

Note: this trick does not work for $\int_a^b |f(z)| dz$ where f is a *complex* function. However, if we have a decomposition $f(z) = g(z) + h(z)i$, we can use $|f(z)| = \sqrt{g(z)^2 + h(z)^2}$.

Integrals with discontinuities in f or f'

In D_0, $p(x) = x^4 + 10x^3 + 19x^2 - 6x - 6$
In D_3, $u(x) = (x - \lfloor x \rfloor - \frac{1}{2})$, $v(x) = \max(\sin(x), \cos(x))$

p	Time	Sub	Eval	Time	Sub	Eval
	$D_0 = \int_0^1 \lvert p(x) \rvert \, e^x \, dx$			$D_1 = \int_0^{100} \lceil x \rceil \, dx$		
32	0.00058	38	412	0.0054	2208	6622
64	0.0016	70	1093	0.014	5536	16606
333	0.049	339	18137	0.12	33512	100534
3333	101	3339	1624951	1.6	345512	1036534
	$D_2 = \int_{-1-i}^{-1+i} \sqrt{x} \, dx$			$D_3 = \int_0^{10} u(x) v(x) dx$		
32	0.00064	68	506	0.011	699	5891
64	0.0021	132	1462	0.035	1437	19653
333	0.067	670	28304	1.4	7576	436 K
3333	35	6670	2669940	2805	76101	42 M

High accuracy with mpmath or Pari/GP is not possible without manually splitting at the singular points.

Practical issues: tolerances and work limits

The user specifies:

- Working precision p
- Absolute and relative tolerances ε_{abs} and ε_{rel}

Configurable work limits:

- Maximum quadrature degree (default: $O(p)$)
- Number of calls to the integrand (default: $O(p^2)$)
- Number of queued subintervals (default: $O(p)$)
- Use stack (default) or global priority queue for the list of subintervals generated by bisection

Relative tolerance

Goal: error $\leq \max(\varepsilon_{\text{abs}}, M\varepsilon_{\text{rel}})$, where $M = |\int_a^b f(x)dx|$.

- This is just a guideline for the algorithm, and the width of the output interval can be larger
- $\varepsilon_{\text{abs}} = \varepsilon_{\text{rel}} = 2^{-p}$ works well for most applications
- Can set $\varepsilon_{\text{abs}} = 0$ to force relative tolerance

Problem: the algorithm does not know M in advance.

- Too large estimate: the final result will have a large error
- Too small estimate: we waste time on small parts
- If the user has a good guess for M, setting $\varepsilon_{\text{abs}} \approx \varepsilon_{\text{rel}} M$ is more efficient than $\varepsilon_{\text{abs}} = 0$

Example: tiny and huge integrals

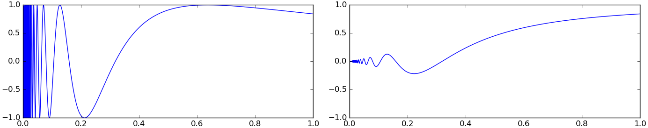
```
sage: C = ComplexBallField(64)

sage: f = lambda x, _: exp(-1000+x)*sin(10*x)
sage: C.integral(f, 0, 1)
[+/- 4.09e-434]                           # time 0.013 ms
sage: C.integral(f, 0, 1, abs_tol=0)
[1.57452858697275 8e-435 +/- 7.36e-451]    # time 1.1 ms
sage: C.integral(f, 0, 1, abs_tol=exp(-1000)/2^64)
[1.574528586972758e-435 +/- 7.27e-451]    # time 0.38 ms


sage: f = lambda x, _: exp(1000+x)*sin(10*x)
sage: C.integral(f, 0, 1)
[6.1110291670932 2e+433 +/- 1.98e+418]     # time 1.1 ms
sage: C.integral(f, 0, 1, abs_tol=exp(1000)/2^64)
[6.1110291670932 2e+433 +/- 1.95e+418]     # time 0.39 ms
```

Example: too much oscillation

$$I_1 = \int_0^1 \sin(1/x)\,dx, \qquad I_2 = \int_0^1 x\sin(1/x)\,dx$$



Default options, $p = 64$:

```
[+/- 1.27], [+/- 1.12]                    # time 0.2 s
```

With $\varepsilon_{\text{abs}} = 10^{-6}$:

```
[0.504 +/- 2.68e-4], [0.37853 +/- 6.35e-6]   # 0.01 s, 0.001 s
```

With priority queue instead of stack:

```
[0.504 +/- 7.88e-4], [0.3785300 +/- 3.17e-8]  # 0.01 s
```

With priority queue, work limits increased to 10^7:

```
[0.504067 +/- 2.78e-7], [0.3785300171242 +/- 5.75e-14] # 17 s
```

Applications: complex analysis

- (Inverse) Laplace/Fourier/Mellin transforms
- Taylor/Laurent/Fourier coefficients:

$$f(z) = \sum_{n=-\infty}^{\infty} c_n (z-a)^n, \quad c_n = \frac{1}{2\pi i} \oint_C \frac{f(z)}{(z-a)^{n+1}} dz$$

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \quad c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx$$

- Counting zeros and poles:

$$N - P = \frac{1}{2\pi i} \oint_C \frac{f'(z)}{f(z)} dz$$

- Acceleration of series (Euler-Maclaurin summation. . .)

Applications: computing special functions

Examples of integral representations:

$$\Gamma(s,z) = \int_z^\infty t^{s-1} e^{-t} dt$$

$$J_\nu(z) = \frac{1}{\pi} \int_0^\pi \cos(z \sin\theta - \nu\theta)\, d\theta - \frac{\sin(\nu\pi)}{\pi} \int_0^\infty e^{-z \sinh t - \nu t} dt$$

Benefits of direct integration:

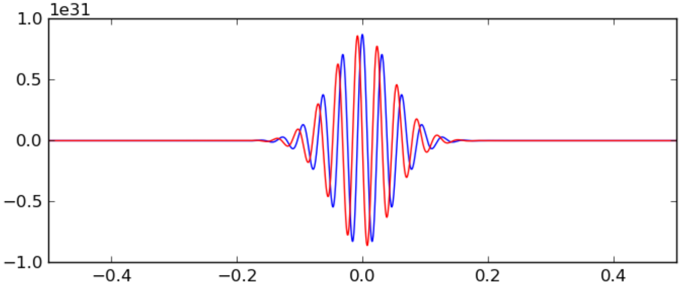
- Useful especially with large parameters (faster convergence, less cancellation than series expansions)
- Possibility to deform path (steepest descent method, analytic continuation)
- Automatic error bounds from integration algorithm

Example: Laurent series of elliptic functions

$$\wp(z;\tau) = \sum_{n=-2}^{\infty} a_n(\tau) z^n, \quad a_n = \frac{1}{2\pi i}\int_\gamma \frac{\wp(z)}{z^{n+1}}dz$$

Fix $\tau = i \Rightarrow \wp(z)$ has poles at $z = M + Ni \quad (M, N \in \mathbb{Z})$.

Pick $\gamma =$ square of width 1 centered on $z = 0$.
One segment ($n = 100$):



Example: Laurent series of elliptic functions

Time per integral ($n \leq 100$):
64 bits: 0.05 seconds
333 bits: 0.8 seconds
3333 bits: 120 seconds

Results with 333-bit precision:

```
a[-2]  = [1.0000000000000000 ... 00000 +/- 3.57e-98] + [+/- 1.89e-98]*I
a[-1]  =                             [+/- 4.11e-98] + [+/- 2.57e-98]*I
a[0]   =                             [+/- 1.02e-97] + [+/- 5.39e-98]*I
a[1]   =                             [+/- 1.41e-97] + [+/- 1.35e-97]*I
a[2]   = [9.453636006461692  ... 52235 +/- 4.44e-97] + [+/- 2.48e-97]*I
a[3]   =                             [+/- 4.47e-97] + [+/- 4.60e-97]*I
...
a[94]  = [380.00000000000135 ... 63746 +/- 9.24e-70] + [+/- 8.27e-70]*I
a[95]  =                             [+/- 1.37e-69] + [+/- 1.37e-69]*I
a[96]  =                             [+/- 2.93e-69] + [+/- 2.91e-69]*I
a[97]  =                             [+/- 5.81e-69] + [+/- 5.82e-69]*I
a[98]  = [395.9999999999996482...46383 +/- 2.90e-68] + [+/- 1.17e-68]*I
a[99]  =                             [+/- 2.32e-68] + [+/- 2.32e-68]*I
a[100] =                             [+/- 4.95e-68] + [+/- 4.95e-68]*I
```

Example: zeros of the Riemann zeta function

Number of zeros of $\zeta(s)$ on $R = [0, 1] + [0, T]i$:

$$N(T) = 1 + \frac{1}{2\pi i} \int_\gamma \frac{\zeta'(s)}{\zeta(s)} ds$$

$\gamma =$ contour around R (plus small excursion around $s = 1$)

Applying the functional equation gives a more useful formula:

$$N(T) = 1 + \frac{\theta(T)}{\pi} + \frac{1}{\pi} \operatorname{Im} \left[\int_{1+\varepsilon}^{1+\varepsilon+Ti} \frac{\zeta'(s)}{\zeta(s)} ds + \int_{1+\varepsilon+Ti}^{\frac{1}{2}+Ti} \frac{\zeta'(s)}{\zeta(s)} ds \right]$$

Can take ε large, e.g. $\varepsilon = 100$.

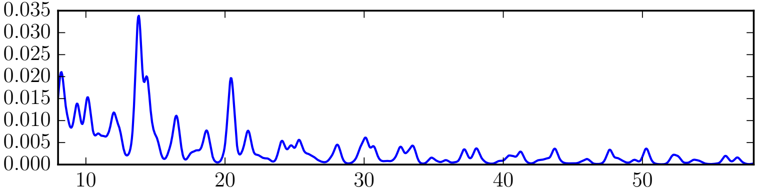
Example: zeros of the Riemann zeta function

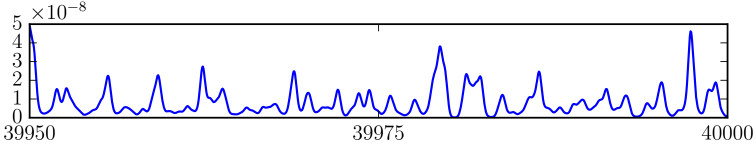
T	p	Time	Eval	Sub	$N(T)$
10^2	32	0.044	261	24	`[29.00000 +/- 1.94e-6]`
10^3	32	0.51	1219	109	`[649.00000 +/- 7.78e-6]`
10^4	32	13	6901	621	`[10142.0000 +/- 4.25e-5]`
10^5	32	12	4088	353	`[138069.000 +/- 3.10e-4]`
10^6	32	16	5326	440	`[1747146.00 +/- 4.06e-3]`
10^7	48	42	4500	391	`[21136125.0000 +/- 5.53e-5]`
10^8	48	210	6205	533	`[248008025.0000 +/- 8.09e-5]`
10^9	48	1590	8070	677	`[2846548032.000 +/- 1.95e-4]`

Using $\varepsilon_{\text{abs}} = 10^{-6}$

Example: $|\zeta(s)|$-integrals (from Harald Helfgott)

$$\int_{-\frac{1}{4}+8i}^{-\frac{1}{4}+40000i} \left| \frac{F_{19}(s+\frac{1}{2})F_{19}(s+1)}{s^2} \right| |ds|, \quad F_N(s) = \zeta(s) \prod_{p \le N} (1-p^{-s})$$





We compute Taylor models $f = g + hi + \varepsilon$ on subsegments $[a, a + 0.5]$, and integrate $\sqrt{g^2 + h^2}$.

Example: Stieltjes constants

$$\zeta(s) = \frac{1}{s-1} + \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \gamma_n (s-1)^n$$

$$\gamma_n = -\frac{\pi}{2(n+1)} \int_{-\infty}^{\infty} \frac{\left(\log\left(\frac{1}{2} + ix\right)\right)^{n+1}}{\cosh^2(\pi x)} dx$$

Some pen-and-paper analysis needed for large n:

- Contour is deformed to go near the saddle point
- Tight enclosures near the saddle point

$\gamma_{10^{100}} \approx 3.18743141870239927999741646992711665139430991088384692250710626598304893415593755966828802 \cdot 10^e$

$e = 2346394292277254080949367838399091160903447689869837385205779111579215664052158234417125417543348369\text{4}$

Conclusions

The Petras algorithms in ball arithmetic works very well in practice for arbitrary-precision rigorous integration.

Todo:

- Methods for improper integrals
- More intelligent adaptive strategies
- Compare Gauss-Legendre vs Taylor series
- Applications (special functions, etc.)