Fast and rigorous numerical integration

Fredrik Johansson (LFANT)

Comité des projet
Inria Bordeaux Sud-Ouest
27 February 2018

Ball arithmetic

Floating-point arithmetic:

3.14159265358979323846264338327950288 42

3.141592653589793238462648118509314556 ???

Ball arithmetic

Floating-point arithmetic:

`3.1415926535897932384626433832795028842`

`3.14159265358979323846264`48118509314556 ???

Ball arithmetic:

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

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

Ball arithmetic

Floating-point arithmetic:

```
3.1415926535897932384626433832795028842
3.1415926535897932384626448118509314556
```
???

Ball arithmetic:

```
[3.1415926535897932384626433832795028842 +/- 1.65e-38]
[3.1415926535897932384626 +/- 4.82e-24]
```

Rigorous approach to numerical computing
Automatic tracking of error bounds
Fast for high precision computations

The Arb library (`http://arblib.org`)

Real and complex numbers, polynomials, matrices, transcendental functions ... **everything with rigorous error bounds and arbitrary precision**

The Arb library (`http://arblib.org`)

Real and complex numbers, polynomials, matrices, transcendental functions ... **everything with rigorous error bounds and arbitrary precision**

About the project:

- C library, 150 000 lines of code, developed since 2012
- Latest version 2.13.0 (February 23, 2018)
- Free software (GNU LGPL)
- Portable, available in common package managers

The Arb library (`http://arblib.org`)

Real and complex numbers, polynomials, matrices, transcendental functions ... **everything with rigorous error bounds and arbitrary precision**

About the project:

- C library, 150 000 lines of code, developed since 2012
- Latest version 2.13.0 (February 23, 2018)
- Free software (GNU LGPL)
- Portable, available in common package managers

Software using Arb:

- Sage (`http://www.sagemath.org/`)
- Nemo – computer algebra in Julia (`http://nemocas.org/`)
- several others!

Rigorous numerical integration

Problem: approximate $\int_a^b f(x)dx$ with (provable) error $< \varepsilon$

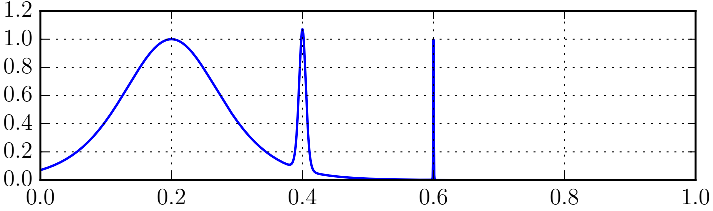
Support arbitrary precision (e.g. 1000 digits)

Applications and domains (not an exhaustive list):

- Complex analysis: integral transforms, differentiation, counting zeros and poles
- Proving inequalities needed in mathematical theorems
- Evaluating probability distributions
- Analytic number theory
- Computational geometry
- Dynamical systems

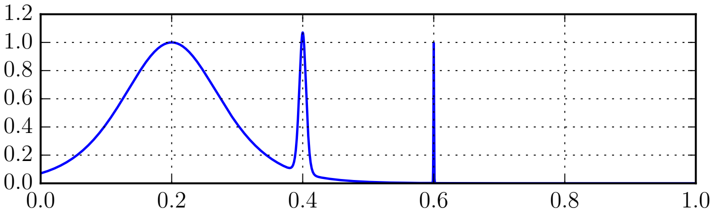
A test integral (from Cranley and Patterson, 1971)

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



A test integral (from Cranley and Patterson, 1971)

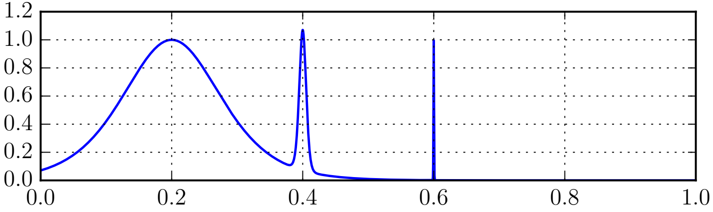
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



Mathematica: 0.209736

A test integral (from Cranley and Patterson, 1971)

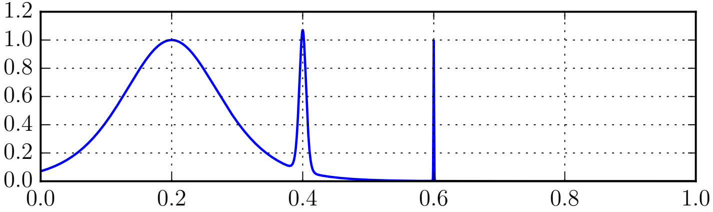
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



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

A test integral (from Cranley and Patterson, 1971)

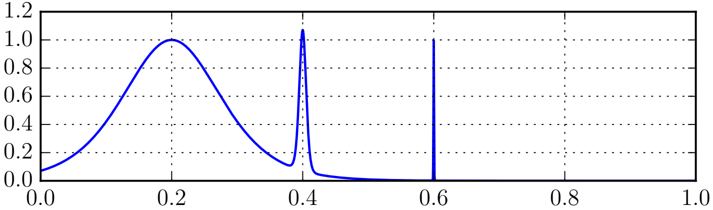
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



Mathematica:	0.209736
Octave:	0.209736, error estimate 10^{-9}
Sage/GSL:	0.209736, error estimate 10^{-14}

A test integral (from Cranley and Patterson, 1971)

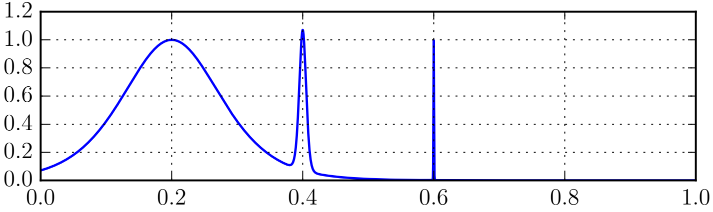
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



Mathematica:	0.209736
Octave:	0.209736, error estimate 10^{-9}
Sage/GSL:	0.209736, error estimate 10^{-14}
SciPy:	0.209736, error estimate 10^{-9}

A test integral (from Cranley and Patterson, 1971)

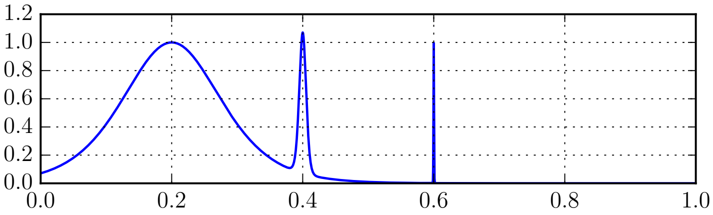
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



Mathematica:	0.209736
Octave:	0.209736, error estimate 10^{-9}
Sage/GSL:	0.209736, error estimate 10^{-14}
SciPy:	0.209736, error estimate 10^{-9}
mpmath:	0.209819

A test integral (from Cranley and Patterson, 1971)

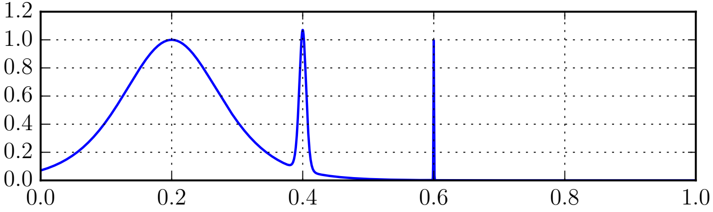
$$I = \int_0^1 \left(\frac{1}{\cosh^2(10(x-0.2))} + \frac{1}{\cosh^4(100(x-0.4))} + \frac{1}{\cosh^6(1000(x-0.6))} \right) dx$$



Mathematica:	0.209736
Octave:	0.209736, error estimate 10^{-9}
Sage/GSL:	0.209736, error estimate 10^{-14}
SciPy:	0.209736, error estimate 10^{-9}
mpmath:	0.209819
Pari/GP:	0.211316

A test integral (from Cranley and Patterson, 1971)

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



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

Results with Arb

64-bit precision – 0.005 seconds:

```
[0.21080273550054928 +/- 4.43e-18]
```

Results with Arb

64-bit precision – 0.005 seconds:

```
[0.21080273550054928 +/- 4.43e-18]
```

333-bit precision – 0.04 seconds:

```
[0.21080273550054927737564325570572915436090918643678119034
785050587872061312814550020505868926155764 +/- 3.73e-99]
```

Results with Arb

64-bit precision – 0.005 seconds:

```
[0.21080273550054928 +/- 4.43e-18]
```

333-bit precision – 0.04 seconds:

```
[0.21080273550054927737564325570572915436090918643678119034
7850505878720613128145500205058689261557764 +/- 3.73e-99]
```

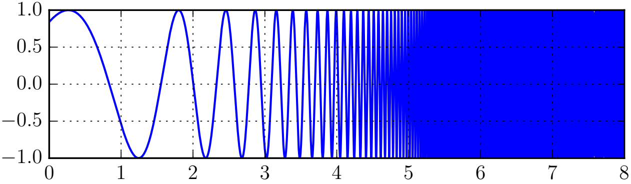
3333-bit precision – 9 seconds:

```
[0.2108027355005492773756432557057291543609091864367811903478505058787206
1312814550020505868926155764182569304879671206001843928909018111331144790
...
7257121373225380650636758727660502248127426772955849986119442410239414975
142681856841572013309327434544120526904793978249363773 +/- 1.39e-1001]
```

Another example (Rump, 2010)

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

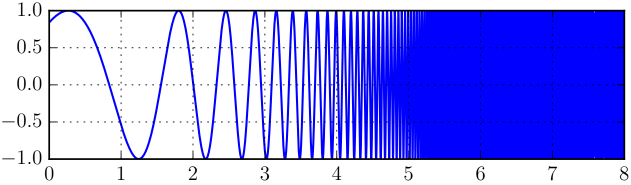
950 sign changes



Another example (Rump, 2010)

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

950 sign changes



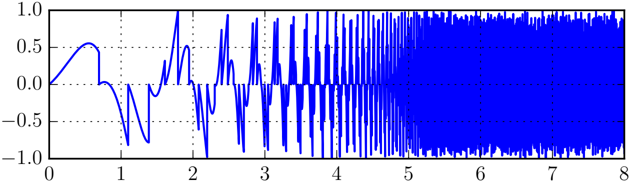
64-bit precision: 0.005 s `[0.34740017265725 +/- 3.94e-15]`

333-bit precision: 0.02 s `[0.34740017265...  +/- 5.98e-96]`

3333-bit precision: 1 s `[0.34740017265...  +/- 2.95e-999]`

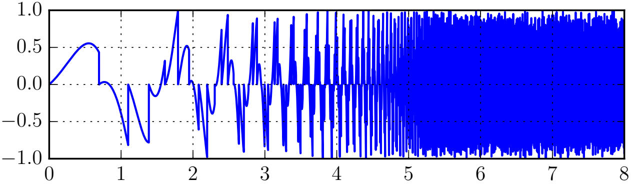
A monster

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



A monster

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



64-bit precision: 0.15 s `[+/- 2.47e+4]`

64-bit precision: 9 s `[0.0986517044784 +/- 4.74e-14]`

333-bit precision: 521 s

`[0.098651704478365206119658249764859856504169620792384491451091906830826680482290609839624064582 4 +/- 6.78e-95]`

Sage interface

```
sage: C = ComplexBallField(333)
sage: C.integral(lambda x, _: sin(x+exp(x)), 0, 8)
[0.34740017265724780787951215911989312465745625486618018
388549271361674821398878532052968510434660 +/- 5.97e-96]
```

Will be available in the next version of Sage (wrapper code by Marc Mezzarobba and Vincent Delecroix).

What's inside the engine

Petras (2002): *Self-validating integration and approximation of piecewise analytic functions*:

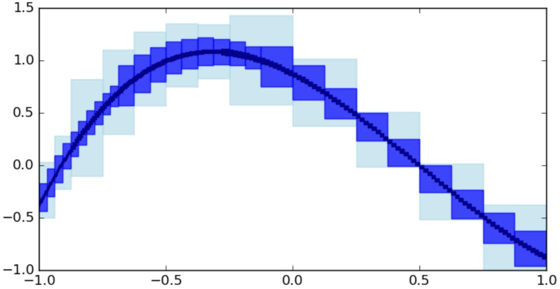
Combine:

- Space adaptivity (bisection of $[a, b]$).
- Degree adaptivity (n-point Gauss-Legendre quadrature with variable n). Error bounds for Gauss-Legendre quadrature via complex magnitudes.

With some adaptations for arbitrary-precision ball arithmetic!

Brute force rigorous 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!

Rigorous 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

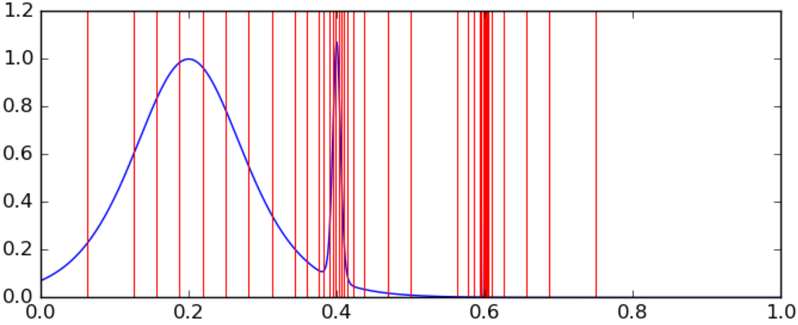
$$\left| \int_{-1}^{1} f(x)dx - \sum_{k=1}^{n} w_k f(x_k) \right| \leq \frac{M}{\rho^{2n}} \cdot C_\rho$$

$X = 1.25, Y = 0.75, \rho = 2.00$ $\qquad$ $X = 2.00, Y = 1.73, \rho = 3.73$

Fast convergence ($O(p)$ evaluations) for smooth integrands, but must be combined with splitting!

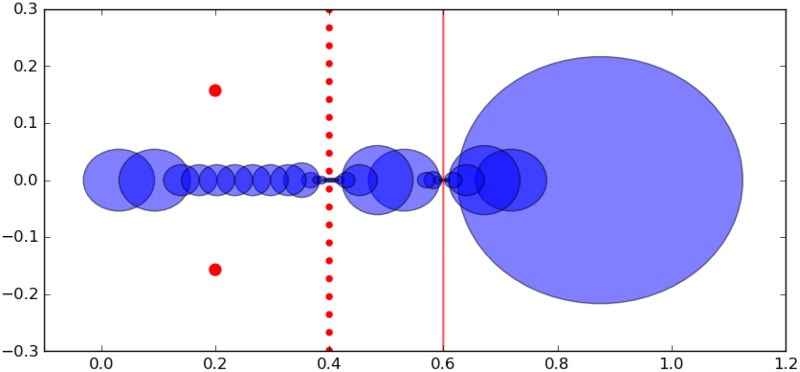
Adaptive subdivision performed by Arb

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



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

Adaptive subdivision, complex view

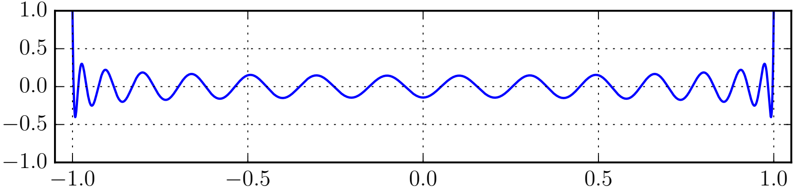


Blue ellipses used for error bounds on the subintervals

Red dots: poles of the integrand

Gauss-Legendre quadrature nodes and weights

The nodes $x_1 \ldots x_n$ are roots of the Legendre polynomial $P_n(x)$.



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

Joint work with Marc Mezzarobba: we greatly speeded up precomputation of the Gauss-Legendre nodes and weights (example: 20-60 s $\rightarrow$ 1-3 s for 1000-digit integration).

Further reading

F.J., *Numerical integration in arbitrary-precision ball arithmetic*, https://arxiv.org/abs/1802.07942

F.J. and M. Mezzarobba, *Fast and rigorous arbitrary-precision computation of Gauss-Legendre quadrature nodes and weights*, https://arxiv.org/abs/1802.03948