Ball arithmetic as a tool in computer algebra

Fredrik Johansson

Inria Bordeaux

Maple Conference, Waterloo, ON, Canada

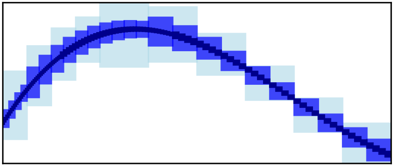
October 16, 2019

Interval arithmetic versus ball arithmetic

[Joris van der Hoeven, 2009]

Intervals $[a, b]$: *better for subdivision of space*

$$[2.0, 3.0] \to [2.0, 2.5] \cup [2.5, 3.0]$$



Balls $[m \pm r]$: *better for approximation of numbers*

$$\pi \in [3.14159265358979323846264338328 \pm 1.07 \cdot 10^{-30}]$$

Software

Interval arithmetic

- IntpakX (Maple)
- XSC (C, Pascal)
- Boost (C++)
- INTLAB (Matlab)
- MPFI (C)
- (many others...)

Ball arithmetic

- Mathemagix (C++)
- Arb (C) – started in 2012 as an extension of FLINT

Example: the partition function $p(n)$

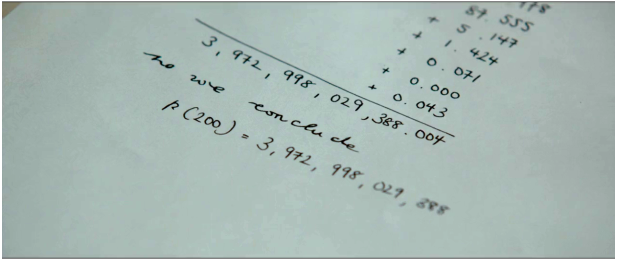
$$p(4) = 5 \quad \text{since} \quad (4) = (3+1) = (2+2) = (2+1+1) = (1+1+1+1)$$

Example: the partition function $p(n)$

$$p(4) = 5 \quad \text{since} \quad (4) = (3+1) = (2+2) = (2+1+1) = (1+1+1+1)$$

Fast computation: the Hardy-Ramanujan-Rademacher formula

$$p(n) = \sum_{k=1}^{\infty} A_k(n) \frac{\sqrt{k}}{\pi\sqrt{2}} \cdot \frac{d}{dn} \left[\frac{\sinh\left(\frac{\pi}{k}\sqrt{\frac{2}{3}\left(n - \frac{1}{24}\right)}\,\right)}{\sqrt{n - \frac{1}{24}}} \right]$$



Scene from *The Man Who Knew Infinity*, 2015

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES®

founded in 1964 by N. J. A. Sloane

A110375 Numbers n such that Maple 9.5, Maple 10, Maple 11 and Maple 12 give the wrong answers for the number of partitions of n. 2

11269, 11566, 12376, 12430, 12700, 12754, 15013, 17589, 17797, 18181, 18421, 18453, 18549, 18597, 18885, 18949, 18997, 20865, 21531, 21721, 21963, 22683, 23421, 23457, 23547, 23691, 23729, 23853, 24015, 24087, 24231, 24339, 24519, 24591, 24627, 24681, 24825, 24933, 25005, 25023, 25059, 25185, 25293, 27020 (list; graph; refs; listen; history; text; internal format)

OFFSET 1,1

COMMENTS

Based on various postings on the Web, sent to N. J. A. Sloane by R. J. Mathar. Thanks to several correspondents who sent information about other versions of Maple.
Mathematica 6.0, DrScheme and pari-2.3.3 all give the correct answers.
Ramanujan's congruence says that numbpart(5*k+4)==0 mod 5, so numbpart(11269)=...851==1 mod 5 can't be correct. [Robert Gerbicz, May 13 2008]

LINKS

Table of n, a(n) for n=1..44.
Author?, Concerning this sequence

EXAMPLE

```
From PARI, the correct answer:
numbpart(11269)
231139177231303975514411787649455628959060199360109972557851519105155176\
803182158917958749053182741632480330718 50
From Maple 11, incorrect:
combinat[numbpart](11269);
231139177231303975514411787649455628959060199360109972557851519105155176\
803182158917958749053182741632480330718 51
On the other hand, the old Maple 6 gives the correct answer.
```

Implementation of $p(n)$ in Arb

Correctness

- Arithmetic error in $\sum_{k=1}^{N} T(k)$
- Truncation error: $\left|\sum_{k=N+1}^{\infty} T(k)\right| \leq \varepsilon$
- $[\ldots 375.000 \pm 0.001] \to \ldots 375$

Implementation of $p(n)$ in Arb

Correctness

- Arithmetic error in $\sum_{k=1}^{N} T(k)$
- Truncation error: $\left|\sum_{k=N+1}^{\infty} T(k)\right| \le \varepsilon$
- $[\ldots 375.000 \pm 0.001] \to \ldots 375$

Performance

- Asymptotically optimal: $\tilde{O}(n^{1/2})$
- $p(10^{10})$: 10^5 digits, 0.2 seconds
- $p(10^{20})$: 10^{10} digits, 200 hours

Implementation of $p(n)$ in Arb

Correctness

- Arithmetic error in $\sum_{k=1}^{N} T(k)$
- Truncation error: $\left|\sum_{k=N+1}^{\infty} T(k)\right| \leq \varepsilon$
- $[\ldots 375.000 \pm 0.001] \to \ldots 375$

Performance

- Asymptotically optimal: $\tilde{O}(n^{1/2})$
- $p(10^{10})$: 10^5 digits, 0.2 seconds
- $p(10^{20})$: 10^{10} digits, 200 hours

Lessons

- Can focus more on mathematical aspects of algorithms, less on numerical issues

Examples of projects using Arb

- Numerical evaluation and analytic continuation of D-finite functions [Mezzarobba, 2016–]
- Computing period matrices and the Abel-Jacobi map of superelliptic curves [Molin and Neurohr, 2017]
- New upper bound for the de Bruijn-Newman constant [D.H.J. Polymath, 2019]

  Very impressed by the robustness of your software as well as by the tremendous speed gains that it has brought us (up till a factor 20-30 faster than pari/gp). So far we haven't encountered a single bug in the software. Well done!

How to use Arb

C directly

```
arb_t x, y;
arb_init(x); arb_init(y);

arb_const_pi(x, 53);            // x = pi to 53 bits
arb_add_ui(y, x, 1, 53);        // y = x + 1
arb_printn(y, 20, 0);           // output

arb_clear(x); arb_clear(y);
```

High-level interfaces

- Julia, Python, SageMath
- Anything with a C interface . . . Maple?

New (since 2017) features in Arb

Improved linear algebra

- F. J., *Faster arbitrary-precision dot product and matrix multiplication*, 2019

Numerical integration

- F. J., *Numerical integration in arbitrary-precision ball arithmetic*, 2018
- F. J. and M. Mezzarobba, *Fast and rigorous arbitrary-precision computation of Gauss-Legendre quadrature nodes and weights*, 2018

New and improved special functions

Improved linear algebra

1. Faster arithmetic
   CPU time to multiply two real 1000×1000 matrices

	$p = 53$	$p = 106$	$p = 212$	$p = 848$
BLAS	0.08			
QD		11	111	
MPFR	36	44	110	293
Arb	3.6	5.6	8.2	27

2. New and improved features (solving, eigenvalues)
3. Both ball arithmetic and ordinary floating-point versions

Dot product

$$\sum_{k=1}^{N} a_k b_k, \qquad a_k, b_k \in \mathbb{R} \text{ or } \mathbb{C}$$

Kernel in basecase ($N \lesssim 10$ to 100) algorithms for:

- Matrix multiplication
- Triangular solving, recursive LU factorization
- Polynomial multiplication, division, composition
- Power series operations

New low-level implementation: up to $4\times$ faster arbitrary-precision arithmetic!

Dot product as an atomic operation

The old way:

```
arb_mul(s, a, b, prec);
for (k = 1; k < N; k++)
    arb_addmul(s, a + k, b + k, prec);
```

The new way:

```
arb_dot(s, NULL, 0, a, 1, b, 1, N, prec);
```

(More generally, computes $s = s_0 + (-1)^c \sum_{k=0}^{N-1} a_{k\cdot\text{astep}} b_{k\cdot\text{bstep}}$)

`arb_dot` – ball arithmetic, real
`acb_dot` – ball arithmetic, complex
`arb_approx_dot` – floating-point, real
`acb_approx_dot` – floating-point, complex

Matrix multiplication (large N)

Same ideas as polynomial multiplication in Arb

1. $[A \pm a][B \pm b]$ via three multiplications AB, $|A|b$, $a(|B|+b)$
2. Split + scale matrices into blocks with uniform magnitude
3. Multiply blocks of A, B exactly over $\mathbb{Z}$ using FLINT
4. Multiply blocks of $|A|$, b, a, $|B|+b$ using hardware FP

Matrix multiplication (large N)

Same ideas as polynomial multiplication in Arb

1. $[A \pm a][B \pm b]$ via three multiplications AB, $|A|b$, $a(|B|+b)$
2. Split + scale matrices into blocks with uniform magnitude
3. Multiply blocks of A, B exactly over $\mathbb{Z}$ using FLINT
4. Multiply blocks of $|A|$, b, a, $|B|+b$ using hardware FP

Where is the gain?

- Integers and hardware FP have less overhead
- Multimodular arithmetic (60-bit primes in FLINT)
- Strassen $O(N^{2.81})$ matrix multiplication in FLINT

More than 10$\times$ speedup!

Approximate / certified linear algebra

Three approaches to linear solving $Ax = b$:

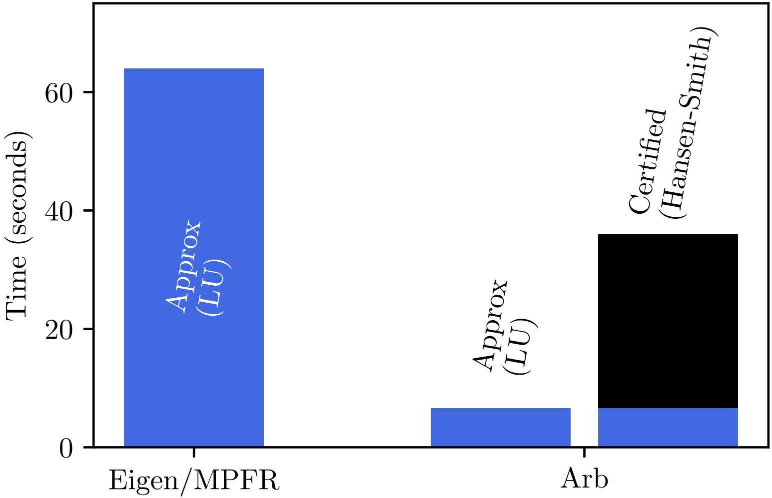
- Gaussian elimination in floating-point arithmetic
  Stable if A is well-conditioned
- Gaussian elimination in interval/ball arithmetic
  Unstable even if A is well-conditioned (loses $O(N)$ digits)
- Approximate solution + certification
  $3.141 \;\rightarrow\; [3.141 \pm 0.001]$

Example: Hansen-Smith algorithm

1. Compute $R \approx A^{-1}$ approximately
2. Solve $(RA)x = Rb$ in interval/ball arithmetic

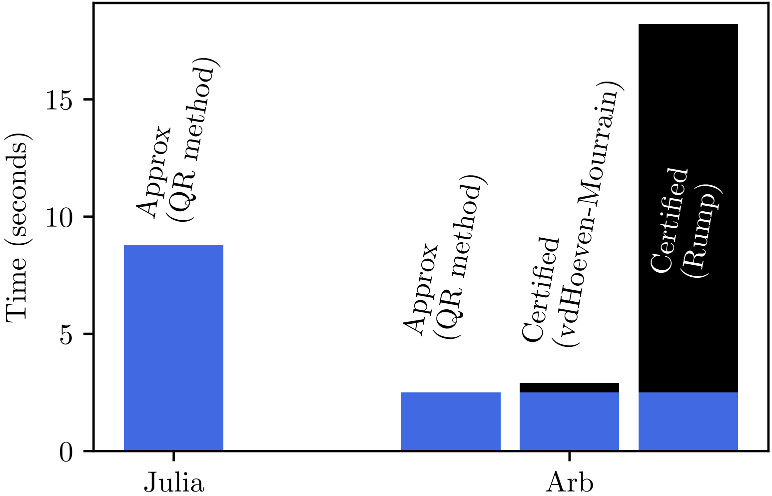
Linear solving

Solving a dense real linear system $Ax = b$ ($N = 1000$, $p = 212$)



Eigenvalues

Computing all eigenvalues and eigenvectors of a nonsymmetric complex matrix ($N = 100$, $p = 128$)

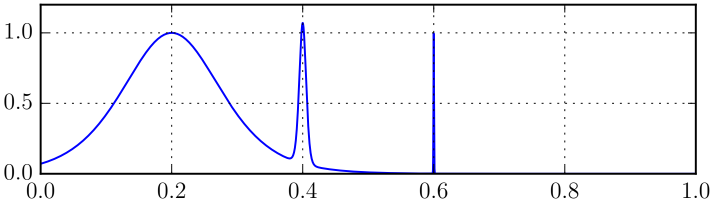


Numerical integration

$$\int_a^b f(x)dx \ = ?$$

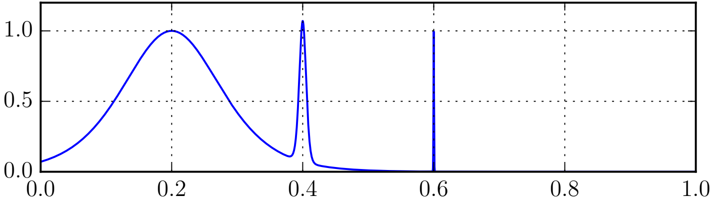
Example: smooth spikes (Cranley and Patterson, 1971)

$$\int_0^1 \mathrm{sech}^2(10(x-0.2)) + \mathrm{sech}^4(100(x-0.4)) + \mathrm{sech}^6(1000(x-0.6))\ dx$$



Example: smooth spikes (Cranley and Patterson, 1971)

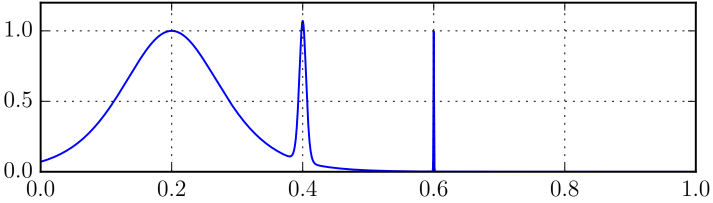
$$\int_0^1 \operatorname{sech}^2(10(x-0.2)) + \operatorname{sech}^4(100(x-0.4)) + \operatorname{sech}^6(1000(x-0.6))\ dx$$



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

Example: smooth spikes (Cranley and Patterson, 1971)

$$\int_0^1 \operatorname{sech}^2(10(x-0.2)) + \operatorname{sech}^4(100(x-0.4)) + \operatorname{sech}^6(1000(x-0.6)) \; dx$$



Arb, 64-bit precision:

```
[0.21080273550054928 +/- 4.55e-18]      # time 0.003 s
```

333-bit precision:

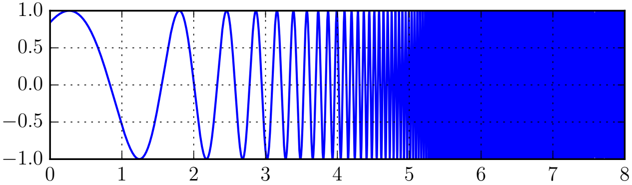
```
[0.21080273550054927375 6... +/- 3.67e-99]   # 0.02 s
```

3333-bit precision:

```
[0.210802735500549277356... +/- 1.39e-1001]  # 5.3 s
```

Example: violent oscillation (Rump, 2010)

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

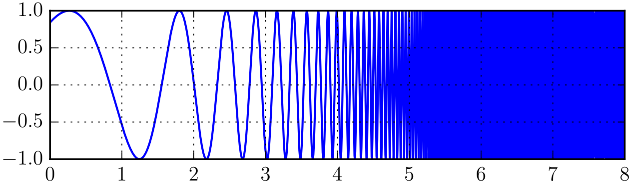


S. Rump noticed that MATLAB's `quad` returned the incorrect `0.2511` after 1 second of computation.

Rump's INTLAB gives `[0.34740016, 0.34740018]` in about 1 s

Example: violent oscillation (Rump, 2010)

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



S. Rump noticed that MATLAB's `quad` returned the incorrect `0.2511` after 1 second of computation.

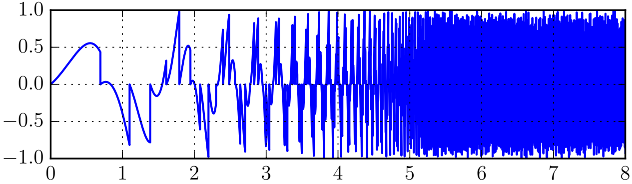
Rump's INTLAB gives `[0.34740016, 0.34740018]` in about 1 s

Arb at 64, 333, and 3333 bits:

```
[0.34740017265725 +/- 3.34e-15]   # 0.004 s
[0.34740017265... +/- 5.31e-96]   # 0.01 s
[0.34740017265... +/- 2.41e-999]  # 1 s
```

Example: a monster

$$\int_0^8 (e^x - \lfloor e^x \rfloor) \sin(x + e^x) dx$$ – now with 2979 discontinuities!

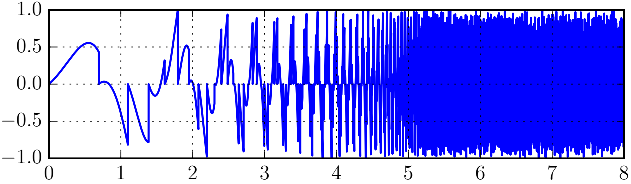


64-bit precision:

```
[+/- 5.45e+3]                      # time 0.14 s
```

Example: a monster

$$\int_0^8 (e^x - \lfloor e^x \rfloor) \sin(x + e^x) dx$$ – now with 2979 discontinuities!



64-bit precision:

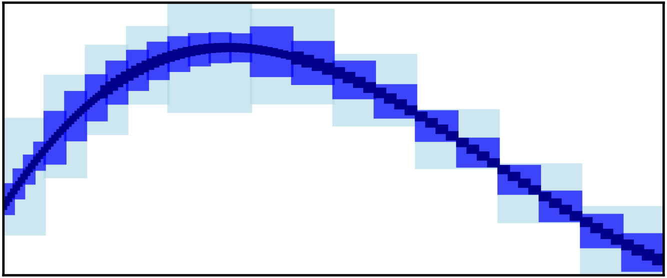
```
[+/- 5.45e+3]                       # time 0.14 s
[0.0986517044784 +/- 4.46e-14]      # time 5 s
```

333-bit precision:

```
[0.09865170447836520611965824976485985650416962079238449145
10919068308266804822906098396240645824 +/- 6.28e-95]  # 268 s
```

Brute force interval integration

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

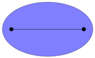


This is simple and general, but we need $2^{O(p)}$ evaluations to achieve p-bit accuracy!

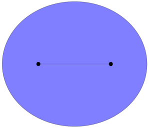
Efficient integration of piecewise analytic functions

- Gauss-Legendre quadrature with error bounds

$$\left|\int_a^b f(x)dx - \sum_{k=1}^n w_k f(x_k)\right| \le \frac{M}{\rho^{2n}} \cdot |b-a| C_\rho, \quad |f(z)| \le M$$



$\rho = 2.00$



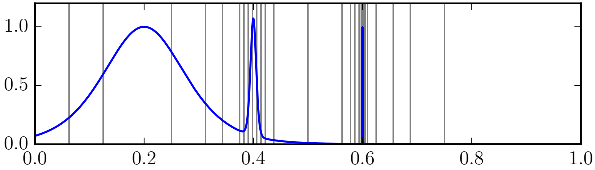
$\rho = 3.73$

- If there are singularities too close to $[a, b]$, bisect (Petras algorithm)
- Fast computation of high-degree, high-precision Gauss-Legendre nodes and weights

Adaptive subdivision

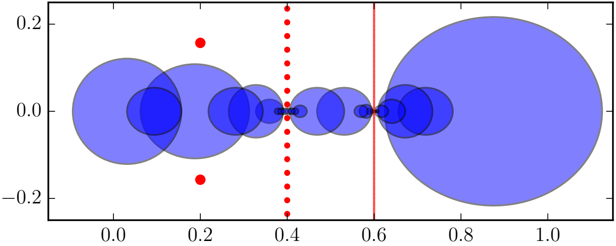
$$\int_0^1 \operatorname{sech}^2(10(x-0.2)) + \operatorname{sech}^4(100(x-0.4)) + \operatorname{sech}^6(1000(x-0.6))\ dx$$

Arb chooses
31 subintervals,
narrowest is 2^{-11}

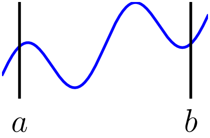


Complex ellipses
used for bounds

Red dots = poles

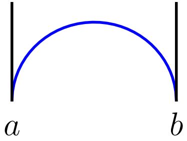


Typical proper integrals



Analytic around $[a, b]$

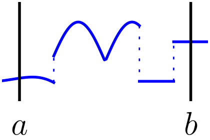
Complexity: $O(p)$ evaluations



Bounded algebraic-type singularities

Example: $\sqrt{1 - x^2}$

Complexity: $O(p^2)$



Piecewise analytic functions*

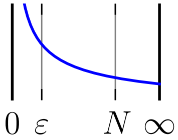
Examples: $\lfloor x \rfloor, \operatorname{sgn}(x), |x|, \max(f(x), g(x))$

Complexity: $O(p^2)$

* Trick: extend piecewise real functions to the complex plane.
Discontinuities $\to$ branch cuts.

Typical improper integrals ($|a|$, $|b|$ or $|f| \to \infty$)

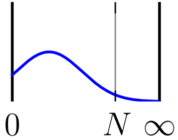
Manual truncation required, e.g. $\int_0^\infty f(x)dx \approx \int_\varepsilon^N f(x)dx$



Algebraic blow-up or decay

Examples: $\int_0^1 \frac{dx}{\sqrt{x}}$, $\int_0^1 \log(x)dx$, $\int_0^\infty \frac{dx}{1+x^2}$

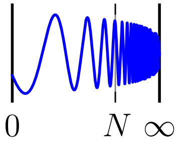
Complexity: $O(p^2)$



Exponential decay

Example: $\int_0^\infty e^{-x}\sin(x)dx$

Complexity: $O(p \log p)$



Essential singularity with slow decay

Example: $\int_1^\infty \frac{\sin(x)}{x} dx$

Complexity: $2^{O(p)}$

Applications of integration in ball arithmetic?

- Computing areas and volumes of bizarre things (duh)
- Special functions

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

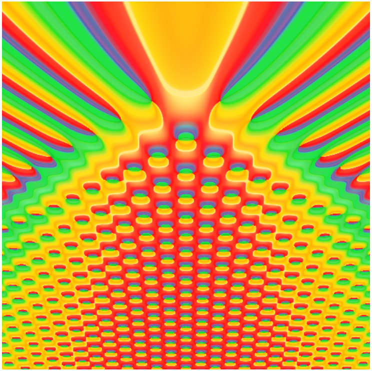
- (Inverse) Laplace/Fourier/Mellin transforms
- Taylor/Laurent/Fourier coefficients
- Counting zeros and poles

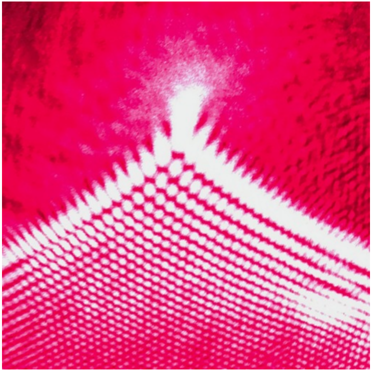
$$N - P = \frac{1}{2\pi i} \oint_C \frac{f'(z)}{f(z)} dz$$

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

Example: diffraction catastrophe integrals

$$P(x,y) = \int_{-\infty}^{\infty} e^{i(t^4+yt^2+xt)}\,dt = 2\int_{0}^{\infty} e^{-t^4+at^2+b}\cosh(ct)dt$$

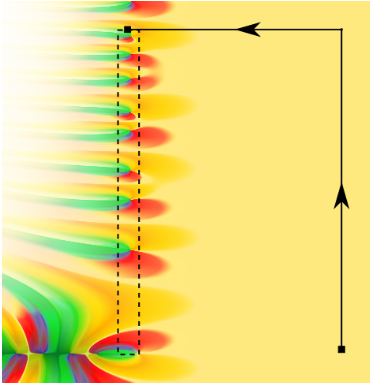




Left: 512×512 image rendered in 15 minutes with Arb ($|x| \leq 12.5$, $-20 \leq y \leq 5$). Using doubling precision (30, 60, ... bits). Near the bottom, $p = 120$ is required.

Right: photo of a cusp caustic produced by illuminating a flat surface with a laser beam through a droplet of water (image credit: Dan Piponi, CC-BY-SA)

Example: zeros of the Riemann zeta function



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

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

$$\frac{1}{\pi} \operatorname{Im} \left[\int_{1+\varepsilon}^{1+\varepsilon+Ti} \frac{\zeta'(s)}{\zeta(s)} ds + \int_{1+\varepsilon+Ti}^{\frac{1}{2}+Ti} \frac{\zeta'(s)}{\zeta(s)} ds \right]$$

T	p	Time (s)	Eval	Sub	$N(T)$
10^3	32	0.51	1219	109	`[649.00000 +/- 7.78e-6]`
10^6	32	16	5326	440	`[1747146.00 +/- 4.06e-3]`
10^9	48	1590	8070	677	`[2846548032.000 +/- 1.95e-4]`

Example: an integral with a large parameter

[F. J., I. Blagouchine, *Computing Stieltjes constants using complex integration*, 2019]

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

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

$\gamma_{10^{100}}(1) \in [3.18743141870239927999741646993 \pm 2.89 \cdot 10^{-30}] \cdot 10^{e}$

$e = 2346394292277254080949367838399091160903447689869$
$8373852057791115792156640521582344171254175433483694$

Some pen-and-paper analysis (steepest descent contour, tight enclosures near saddle point) needed for large n

Recent new and improved special functions

- Lambert W function (all branches)
- Coulomb wave functions
- Riemann zeta zeros (contributed by D.H.J. Polymath)
  - The zero with index $n = 10^9$ in 0.1 seconds
- Airy function zeros
- Dirichlet L-functions (contributed by P. Molin)
- Stieltjes constants
- Tighter enclosures for elementary functions
- Optimized evaluation of Legendre polynomials

Ad: *The Mathematical Functions Grimoire*

`http://fungrim.org`

An attempt to make a better (at least for me)

1. reference work
2. software library for symbolic computation

for special functions

---

grimoire = book of magic formulas

Thank you!