Hypergeometric functions in Arb

Fredrik Johansson
LFANT, INRIA Bordeaux

Journées FastRelax, 2016-05-25
LAAS-CNRS, Toulouse

What is Arb?

C library for arbitrary-precision mid-rad interval arithmetic

$$[3.14159265358979323846264 \pm 4.03 \cdot 10^{-25}]$$

Supports complex numbers, polynomials, power series, matrices, special functions. Designed for applications in number theory and computer algebra.

- `http://fredrikj.net/arb/`
- Free software (GNU LGPL)
- Dependencies: GMP/MPIR, MPFR, FLINT
- Works on common 32-bit and 64-bit platforms, threadsafe, extensively tested

In SageMath

Arb is now a standard package in SageMath, with a partial interface (thanks to Marc Mezzarobba, Clemens Heuberger).

```
sage: R = ComplexBallField(128)

sage: R("0.7", "1.3").bessel_J(0)
[1.266213309539262969656642862781658302 2 +/- 3.39e-38] +
[-0.523931733454760573436091557399835881 8 +/- 2.48e-38]*I

sage: R("0.7 +/- 1e-6", "1.3 +/- 1e-7").bessel_J(0)
[1.26621 +/- 3.97e-6] + [-0.52393 +/- 2.64e-6]*I}
```

Some more (experimental) wrappers...

`https://gist.github.com/fredrik-johansson/` - arbmpfr.c

- Re-implements all MPFR math functions (correctly rounded)
- Revealed three bugs in MPFR (sqrt, zeta, jn/yn)
- No rounding fast paths (e.g. $\tanh(x)$, x large)
- No special values (signed zero, nan, inf, etc.)

`https://github.com/fredrik-johansson/arbcmath`

- Wraps Arb transcendentals for the C99 `double complex` type, with guaranteed $\approx$ 53-bit relative accuracy
- No special values, options...

`https://github.com/klkuhlm/unconfined`

- Fortran code for aquifer simulations, calling Arb for accurate quad-precision complex Bessel functions

Nemo - `http://nemocas.org/` - Julia computer algebra package

Generalized hypergeometric function

$$ {}_pF_q(a_1 \dots a_p; b_1 \dots b_q; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!} $$

$$ {}_p\tilde{F}_q(a_1, \dots, a_p; b_1, \dots, b_q; z) = \frac{{}_pF_q(a_1, \dots, a_p; b_1, \dots, b_q; z)}{\Gamma(b_1) \cdots \Gamma(b_q)} $$

$$ (a)_k = a(a+1)(a+2) \cdots (a+k-1) $$

$p \leq q$: exponential type (entire function)
$p = q + 1$: logarithmic/algebraic type (radius of convergence 1)
$p > q + 1$: asymptotic (divergent) series

Done: complete implementation for $p \leq 2, q \leq 1$.

Partial implementation for arbitrary p, q (convergent cases only).

Completed cases of ${}_1F_1$ and friends

Function	Notation
Confluent hyp. functions	${}_1F_1(a;b;z), {}_1\tilde{F}_1(a;b;z),$
Confluent hyp. functions	$U(a,b,z)$ $(\simeq {}_2F_0)$
Confl. hyp. limit functions	${}_0F_1(b;z), {}_0\tilde{F}_1(b;z)$
Bessel functions	$J_\nu(z),\ Y_\nu(z),\ I_\nu(z),\ K_\nu(z)$
Airy functions	$\mathrm{Ai}(z), \mathrm{Ai}'(z),\ \mathrm{Bi}(z), \mathrm{Bi}'(z)$
Error functions	$\mathrm{erf}(z),\ \mathrm{erfc}(z),\ \mathrm{erfi}(z)$
Fresnel integrals	$S(z),\ C(z)$
Incomplete gamma functions	$\Gamma(s,z), \gamma(s,z), P(s,z), Q(s,z), \gamma^*(s,z)$
Generalized exp. integral	$E_\nu(z)$
Exp. and log. integrals	$\mathrm{Ei}(z),\ \mathrm{li}(z),\ \mathrm{Li}(z)$
Trigonometric integrals	$\mathrm{Si}(z),\ \mathrm{Ci}(z),\ \mathrm{Shi}(z),\ \mathrm{Chi}(z)$
Laguerre, Hermite functions	$L_\nu^\mu(z),\ H_\nu(z)$

To be added (easy): parabolic cylinder functions, Whittaker functions, Coulomb wave functions

Completed cases of ${}_2F_1$ and friends

Function	Notation
Gauss hypergeometric function	${}_2F_1(a,b;c;z), {}_2\tilde{F}_1(a,b;c;z)$
Chebyshev functions	$T_\nu(z), U_\nu(z)$
Jacobi functions	$P_\nu^{\alpha,\beta}(z)$
Gegenbauer (ultraspherical) functions	$C_\nu^\mu(z)$
Legendre functions	$P_\nu^\mu(z), Q_\nu^\mu(z), \mathcal{P}_\nu^\mu(z), \mathcal{Q}_\nu^\mu(z)$
Spherical harmonics	$Y_n^m(\theta,\varphi)$
Incomplete beta functions	$B(a,b;z), I(a,b;z)$
Complete elliptic integrals	$K(z), E(z)$

What is done, exactly?

Complete:

- Arbitrary-precision evaluation for complex parameters and argument, with error bounds
- Fast evaluation for large/near-singular argument z
- Generally faster than previous nonrigorous implementations (Mathematica, Maple, mpmath, others?)

Incomplete:

- Optimizations for low precision and some special cases
- Asymptotics for large parameters (say $|a| \gg 10^4$)
- Tight output intervals, given wide input intervals (in general)
- Intervals around some special parameter values (e.g. $J_\nu(z)$, $\nu = [-1 \pm 0.01]$)
- Series expansions of all functions (work in progress)

How to compute hypergeometric functions

If the defining series

$$ {}_pF_q(a_1 \dots a_p; b_1 \dots b_q; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!} $$

converges rapidly, use

$$ S \pm R \qquad \underbrace{S = \sum_{k=0}^{N-1} T(k)}_{\text{Using interval arithmetic}} \qquad \underbrace{\left| \sum_{k=N}^{\infty} T(k) \right| \leq R}_{\text{Upper bound}} $$

Error bounds

$$\left| \sum_{k=N}^{\infty} T(k) \right| = |T(N)| \, (1 + \varepsilon)$$

Convergent cases: bound $(1 + \varepsilon)$ by a geometric series. With $T(k+1)/T(k) = z \prod_i (a_i + k) / \prod_i (b_i + k)$, for large N use

$$\frac{|a+k|}{|b+k|} = \left| 1 + \frac{a-b}{b+k} \right| \leq 1 + \frac{|a-b|}{|b+N|}$$

and

$$\left| \frac{1}{b+k} \right| \leq \frac{1}{|b+N|}.$$

Asymptotic series ${}_2F_0$: general, effective bounds by Olver (1965).

Parameter derivatives

For expressions such as

$$f(a, z) = \frac{g(a, z)}{\sin(\pi a)}$$

with $a \in \mathbb{Z}$, we need to compute $\lim_{\varepsilon \to 0} f(a + \varepsilon, z)$.

Solution: use arithmetic in $\mathbb{C}[[\varepsilon]]/\langle \varepsilon^n \rangle$.

In particular, we need to evaluate ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; z)$ with $a_i, b_i, z \in \mathbb{C}[[\varepsilon]]/\langle \varepsilon^n \rangle$, usually with $n = 2$.

Error bounds follow the same principles as in the $n = 1$ case.

Algorithms to evaluate truncated hypergeometric series

Depends on the precision and on the bit lengths of the inputs.

Forward recurrence
Low output precision

Binary splitting
Short parameters, short argument: $J_3(3.25)$.

Rectangular splitting
Short parameters, long argument: $J_3(\pi)$. (Could also be done, but less common: $J_\pi(3.25)$)

Fast multipoint evaluation
Long parameters, long argument: $J_\pi(\pi)$

Hypergeometric series as a matrix product

$$T(k) = \frac{x^k}{k!}, \quad T(k+1) = (x/(k+1))\,T(k)$$

$$S(N) = \sum_{k=0}^{N-1} \frac{x^k}{k!}, \quad S(k+1) = T(k) + S(k)$$

$$\begin{bmatrix} T(k+1) \\ S(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} x/(k+1) & \\ 1 & 1 \end{bmatrix}}_{M(k)} \begin{bmatrix} T(k) \\ S(k) \end{bmatrix}$$

With $N \approx \infty$,

$$\begin{bmatrix} 0 \\ \exp(x) \end{bmatrix} \approx M(N-1)M(N-2)\cdots M(1)M(0) \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Binary splitting

Divide and conquer: if all input entries are "small", the bit complexity will be $\widetilde{O}(N)$.

Example: computing 16!

```
1 2    3 4    5 6    7 8    9 10    11 12    13 14    15 16
2      12     30     56     90      132      182      240
24            1680          11880            43680
40320                       518918400
20922789888000
```

Fast multipoint evaluation

With $N = 100$

1. Compute $R(k) = M(k+9)M(k+8)\cdots M(k+0)$ as a matrix with entries in $\mathbb{C}[k]$
2. Evaluate $R(k)$ simultaneously at $k = 0, 10, 20, \ldots, 90$
3. Multiply the evaluated matrices

Complexity is $\widetilde{O}(N^{1/2})$ arithmetic operations in $\mathbb{C}$ (uses FFT based fast polynomial arithmetic).

Drawbacks: reduced numerical stability, high overhead

Rectangular splitting

Evaluate $\sum_{i=0}^{N-1} \square z^i$ using $O(N)$ cheap arithmetic operations and $O(N^{1/2})$ expensive arithmetic operations:

With $N = 16$:

$$
\begin{array}{llllllllllll}
(& \square & + & \square\, z & + & \square\, z^2 & + & \square\, z^3 &) & & + \\
(& \square & + & \square\, z & + & \square\, z^2 & + & \square\, z^3 &) & z^4 & + \\
(& \square & + & \square\, z & + & \square\, z^2 & + & \square\, z^3 &) & z^8 & + \\
(& \square & + & \square\, z & + & \square\, z^2 & + & \square\, z^3 &) & z^{12} &
\end{array}
$$

Argument transformations

To cover all $z \in \mathbb{C}$ efficiently, we need series expansions at $z = 0$ and $z = \infty$, and also at $z = 1$ when $p = q + 1$.

The cases ${}_pF_q$ with $p \leq 2, q \leq 1$ are the easiest to deal with, because the series expansions at the singular points can always be written in terms of ${}_pF_q$'s with $p \leq 2, q \leq 1$.

With functions of higher order, the expansions are no longer hypergeometric (in general), and the more powerful framework of D-finite functions is required.

Connection formulas for confluent hyp. functions

Large $|z|$: ${}_0F_1 \to {}_1F_1 \to U(= {}_2F_0)$
Small $|z|$: $U \to {}_1F_1$ or ${}_0F_1$

$$U^*(a,b,z) = z^a U(a,b,z) = {}_2F_0(a, a-b+1, -1/z)$$

$$U(a,b,z) = \frac{\Gamma(1-b)}{\Gamma(a-b+1)} {}_1F_1(a,b,z) + \frac{\Gamma(b-1)}{\Gamma(a) z^{b-1}} {}_1F_1(a-b+1, 2-b, z)$$

$$\frac{{}_1F_1(a,b,z)}{\Gamma(b)} = \frac{(-z)^{-a}}{\Gamma(b-a)} U^*(a,b,z) + \frac{z^{a-b} e^z}{\Gamma(a)} U^*(b-a,b,-z)$$

$${}_0F_1(b,z) = e^{-2\sqrt{z}} {}_1F_1\left(b - \frac{1}{2}, 2b-1, 4\sqrt{z}\right)$$

Example: modified Bessel function of the second kind

Case 1: $|z| \approx \infty$: asymptotic series

$$K_a(z) = \left(\frac{\pi}{2z}\right)^{1/2} e^{-z} U^*(a+\tfrac{1}{2}, 2a+1, 2z), \quad U^* \sim {}_2F_0(\ldots, -\tfrac{1}{2z})$$

Case 2: $|z| \approx 0$ and $a \notin \mathbb{Z}$: convergent series

$$K_a(z) = \frac{1}{2}\frac{\pi}{\sin(\pi a)}\left[\left(\frac{z}{2}\right)^{-a} {}_0\widetilde{F}_1\left(1-a, \frac{z^2}{4}\right) - \left(\frac{z}{2}\right)^{a} {}_0\widetilde{F}_1\left(1+a, \frac{z^2}{4}\right)\right]$$

Case 3: $|z| \approx 0$ and $a \in \mathbb{Z}$: parameter limit

$$\lim_{\varepsilon \to 0} K_{a+\varepsilon}(z)$$

as in (Case 2), but with $\mathbb{C}[[\varepsilon]]/\langle\varepsilon^2\rangle$ arithmetic

Where to use what formula?

Assume small parameters:

$$ {}_1F_1(a, b, z) \approx {}_1F_1(1, 1, z) = e^z $$

Naive analysis:

- $|z^k/k!| \to 0$, so the convergent series can give ∞ accurate bits
- The minimum of $|k!/z^k|$ is $|z|!/|z|^{|z|} \approx e^{-|z|}$, so the asymptotic series can give $|z|/\log(2)$ accurate bits
- Algorithm: use the asymptotic series when $|z| > \log(2)p$

Optimizing for accuracy

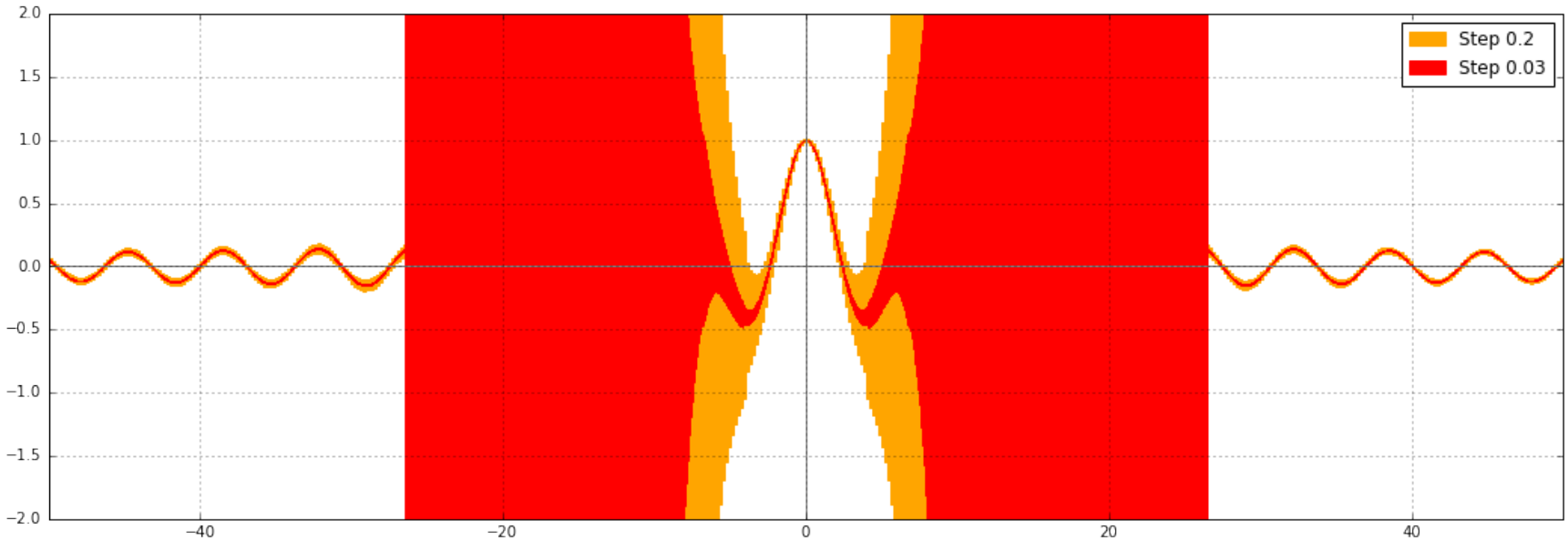
$|{}_1F_1(a, b, z)| \approx |e^z|$, but the terms in the convergent series grow to size $e^{|z|}$.

We lose $c = (|z| - \mathrm{Re}(z))/\log(2)$ bits to cancellation, and must increase the working precision.

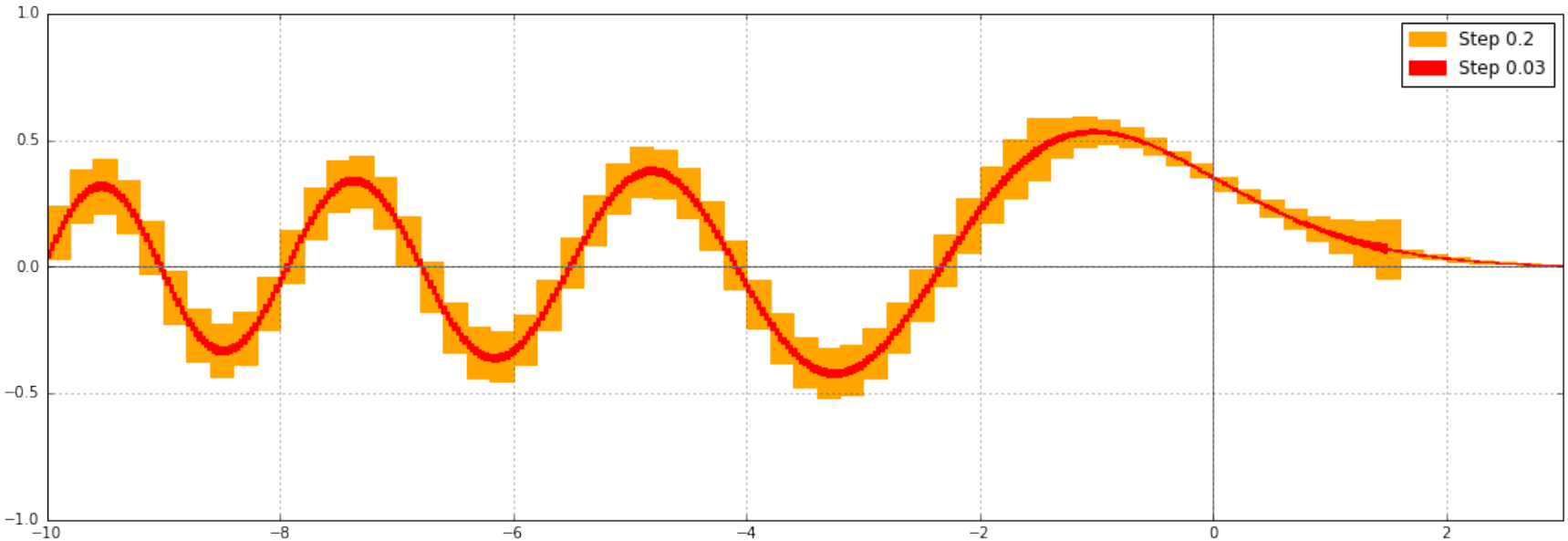
This limits the accuracy when z is approximate!

Better method in special cases: evaluate at midpoint of z, bound propagated error using the function derivative.

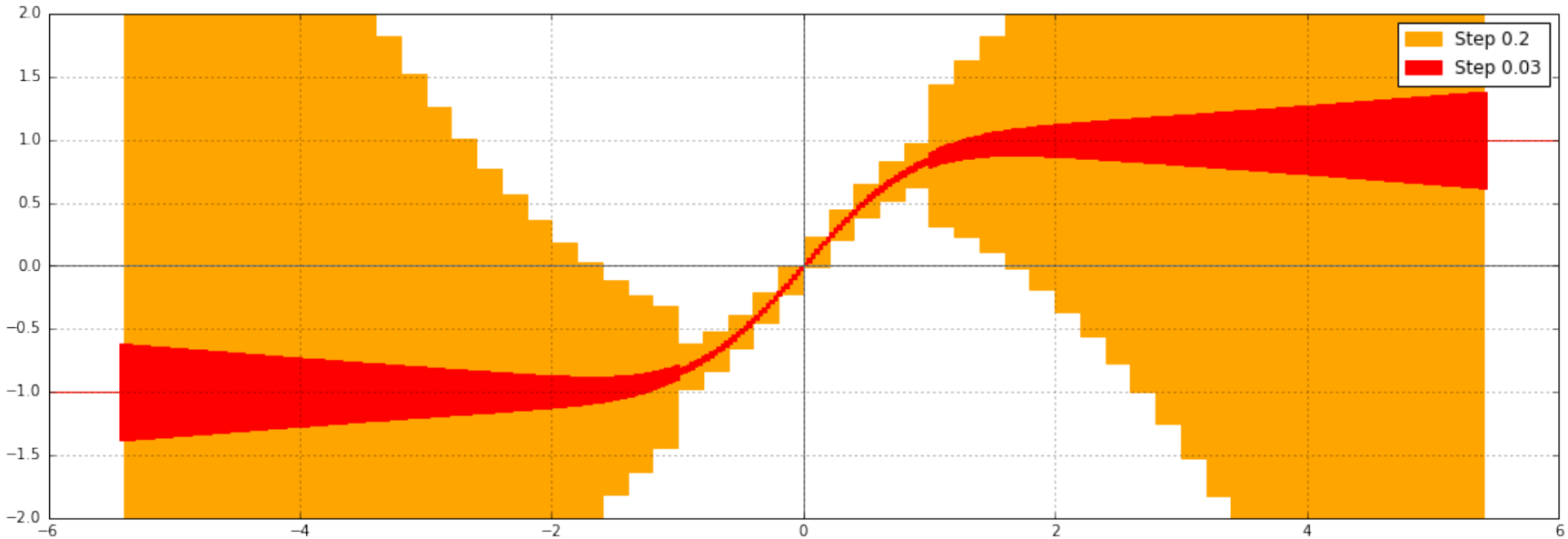
Bessel function $J_0(x)$ - not yet optimized



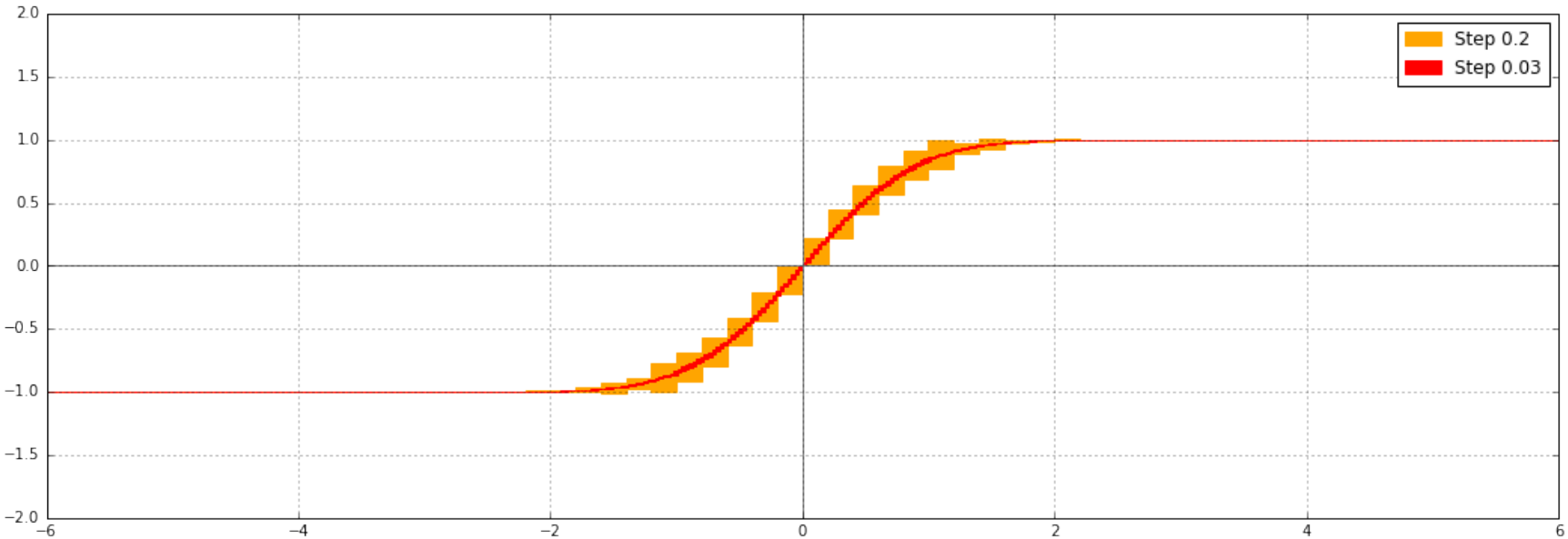
Airy function Ai(x) - optimized



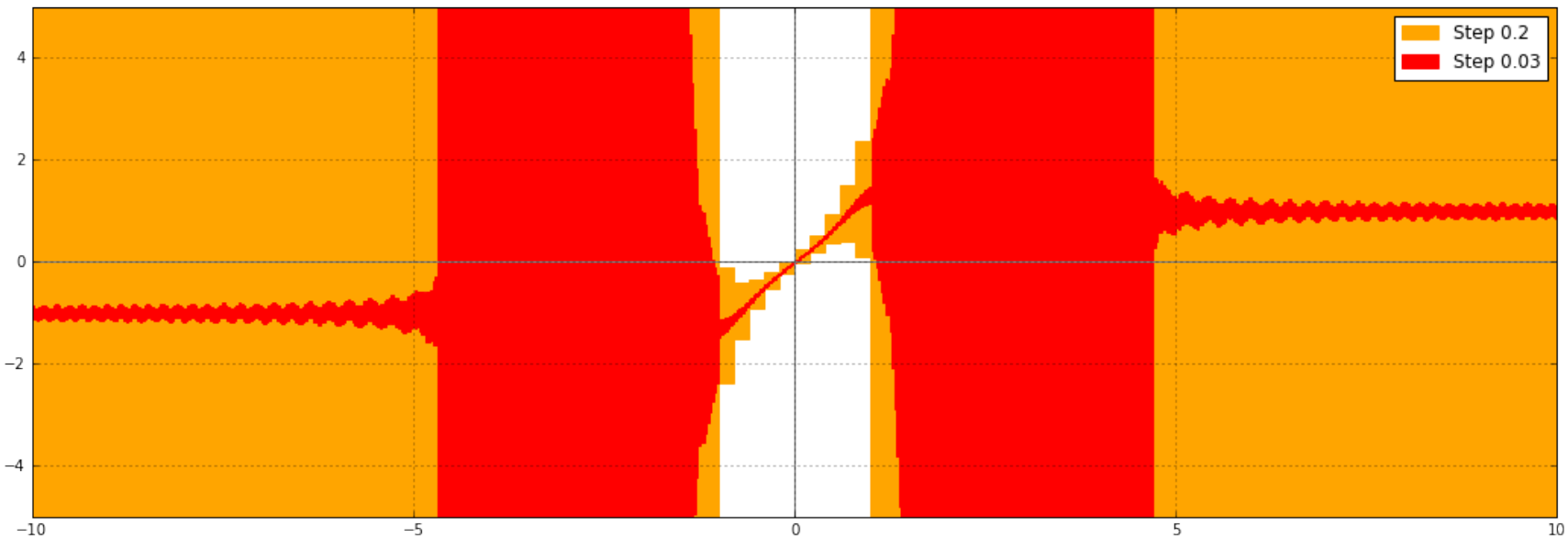
erf(x) - before optimizing



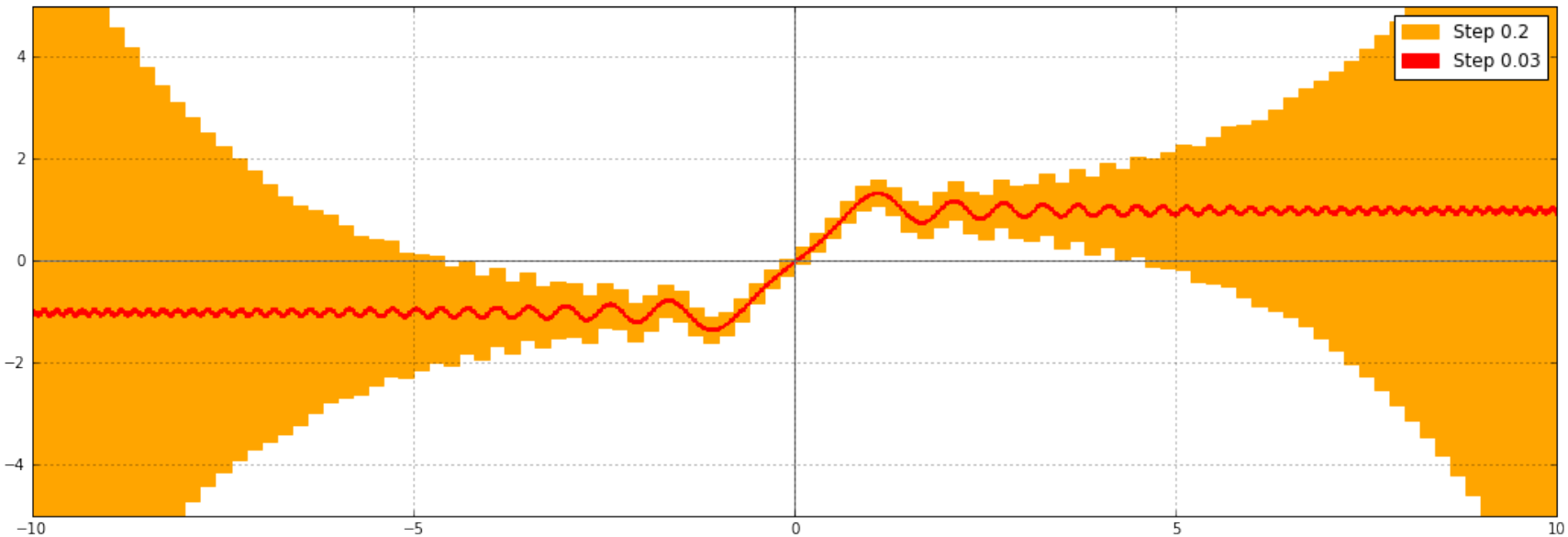
$\mathrm{erf}(x)$ - optimized



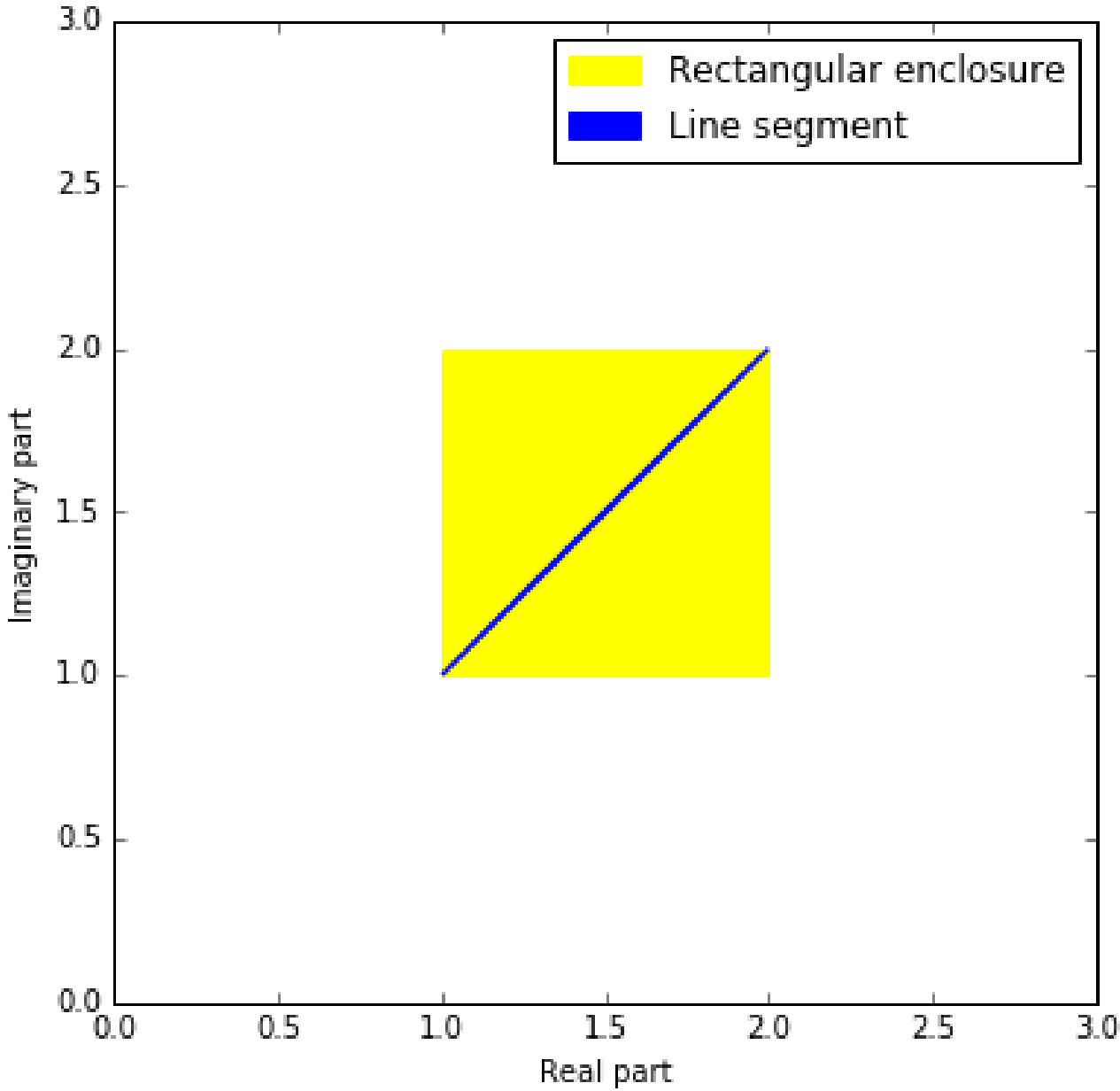
$\mathrm{erf}((1+i)x)$ - before optimizing



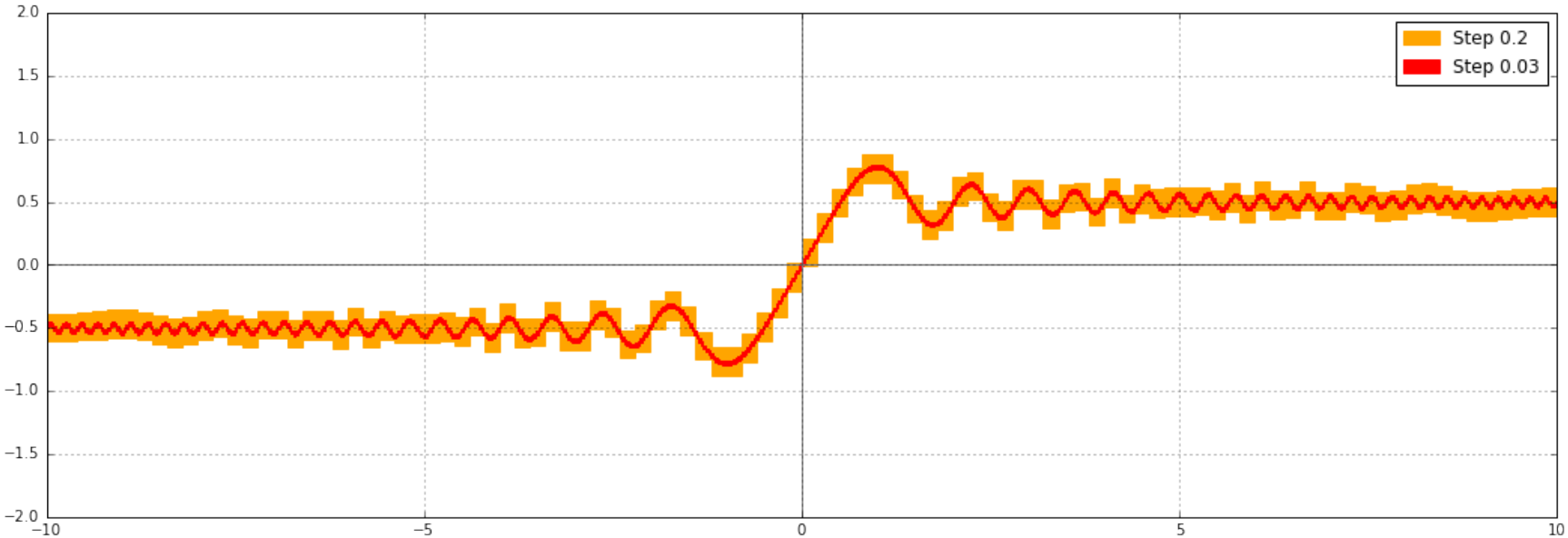
$\mathrm{erf}((1+i)x)$ - optimized



The problem with the diagonal



Fresnel integral $C(x)$ - optimized



$$\frac{1}{2}(1+i)\,\mathrm{erf}\left(\frac{1}{2}\sqrt{\pi}(1\pm i)z\right) = C(z) \pm iS(z)$$

Connection formulas for ${}_2F_1$

Linear fractional transformations:

$$\frac{z}{z-1}, \quad \frac{1}{z}, \quad \frac{1}{1-z}, \quad 1-z, \quad 1-\frac{1}{z}$$

Examples:

$${}_2F_1(a,b;c;z) = (1-z)^{-a}\,{}_2F_1\left(a, c-b; c; \frac{z}{z-1}\right)$$

$$\begin{aligned}\frac{\sin(\pi(b-a))}{\pi}{}_2\tilde{F}_1(a,b,z) = &\frac{(-z)^{-a}}{\Gamma(b)\Gamma(c-a)}{}_2\tilde{F}_1\left(a, a-c+1, a-b+1, \frac{1}{z}\right)\\ &- \frac{(-z)^{-b}}{\Gamma(a)\Gamma(c-b)}{}_2\tilde{F}_1\left(b, b-c+1, b-a+1, \frac{1}{z}\right)\end{aligned}$$

(As a limit when $b-a \in \mathbb{Z}$.)

Corner case of ${}_2F_1$

All transformations are ineffective near $z = \exp(\pm\pi i/3)$

Solution: use Taylor series to analytically continue the solution of the ODE: $f(z) = {}_2F_1(a, b, c, z_0 + z)$ satisfies

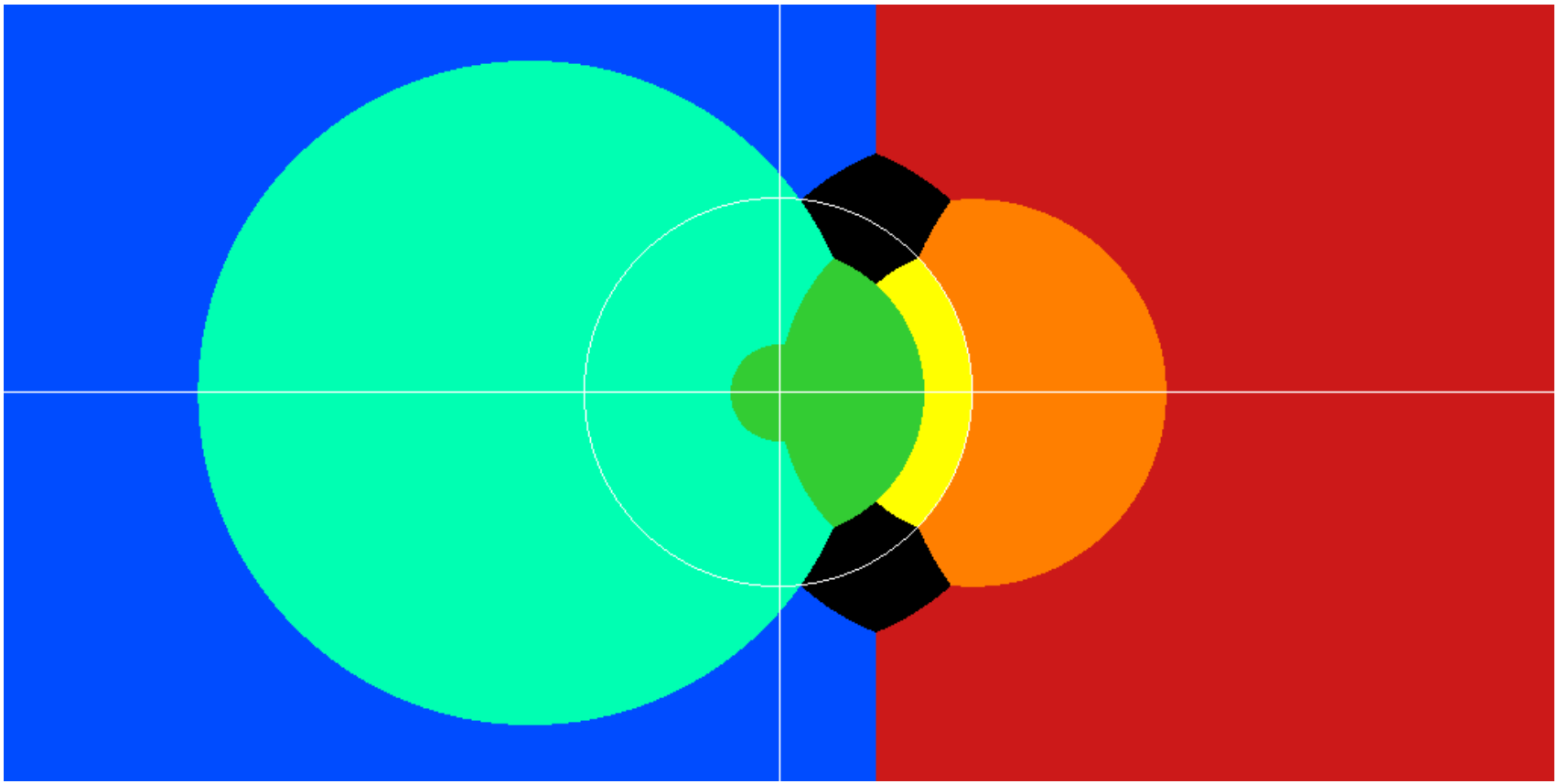
$$(z+z_0)(z+z_0-1)f''(z)+[(a+b+1)(z+z_0)-c]f'(z)+abf(z) = 0.$$

Error bounds by the Cauchy-Kovalevskaya majorant method.

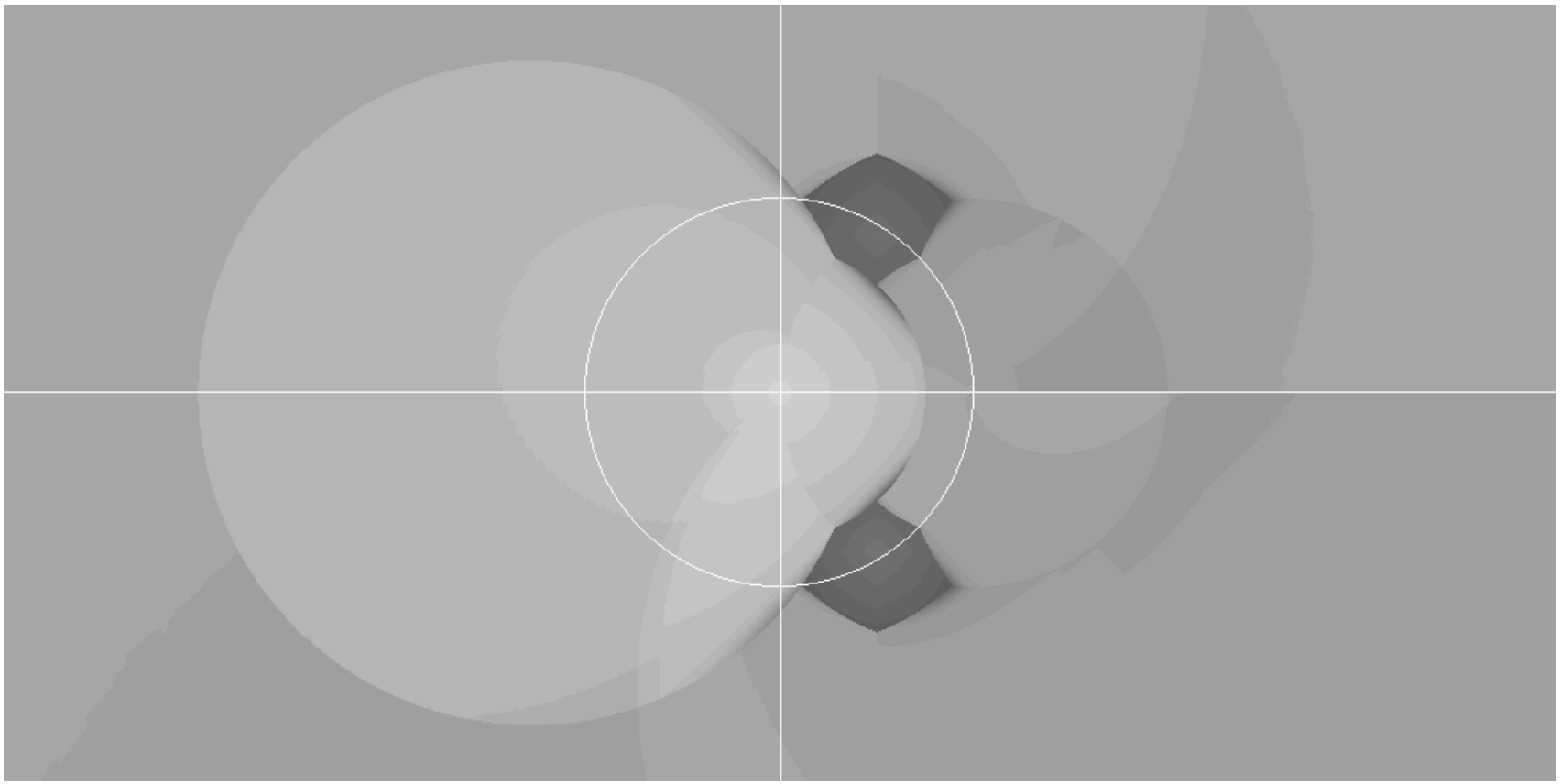
Example path:

$$0 \quad \to \quad 0.375 \pm 0.625i \quad \to \quad 0.5 \pm 0.8125i \quad \to \quad z$$

Covering the domain of ${}_2F_1$



Evaluating ${}_2F_1$ - accuracy



Test problem: ${}_2F_1(0.2 + 0.3i, 0.4 + 0.5i, 0.5 + 0.6i, z)$.
The parameters are 64-bit intervals.
Near $\exp(\pm\pi i/3)$: 23 bits lost; elsewhere: 5-12 bits lost

Some benchmark(et)ing

1. Small arguments, different precision and input types (binary splitting, rectangular splitting, fast multipoint evaluation, power series)
2. Large arguments
3. Large parameters

Low-precision parameters

Test case	Digits	Mathematica	mpmath	Arb
$J_3(3.25)$	10	24	26	6
	100	48	60	24
	1000	320	730	130
	10000	9 200	87 000	1 300
	100000	590 000	15 000 000	22 000
$J_3(\pi)$	10	23	32	7
	100	56	65	26
	1000	800	740	270
	10000	110 000	87 000	11 000
	100000	17 000 000	15 000 000	760 000

Timings in μs.

Full-precision parameters

Test case	Digits	Mathematica	mpmath	Arb
$J_\pi(\pi)$	10	58	120	14
	100	220	180	80
	1000	34 000	2 800	2 400
	10000	8 500 000	1 700 000	220 000
	100000	-	-	36 000 000
${}_0F_1(b,z)$	10	26	31	6
$b = \pi + 1$	100	84	52	41
$z = -\pi^2/4$	1000	1 800	1 100	1 200
	10000	350 000	230 000	78 000
	100000	61 000 000	52 000 000	12 000 000

Timings in μs.

Parameter limit

(Arb uses $\mathbb{C}[[\varepsilon]]/\langle\varepsilon^2\rangle$ arithmetic internally.)

Test case	Digits	Mathematica	mpmath	Arb
$K_3(3.25)$	10	68	1 200	42
	100	160	7 100	140
	1000	1 700	1 400 000	950
	10000	140 000	-	16 000
	100000	19 000 000	-	410 000
$K_3(\pi)$	10	69	1 200	44
	100	160	7 100	150
	1000	2 600	1 400 000	1 600
	10000	350 000	-	54 000
	100000	52 000 000	-	3 300 000

Timings in μs.

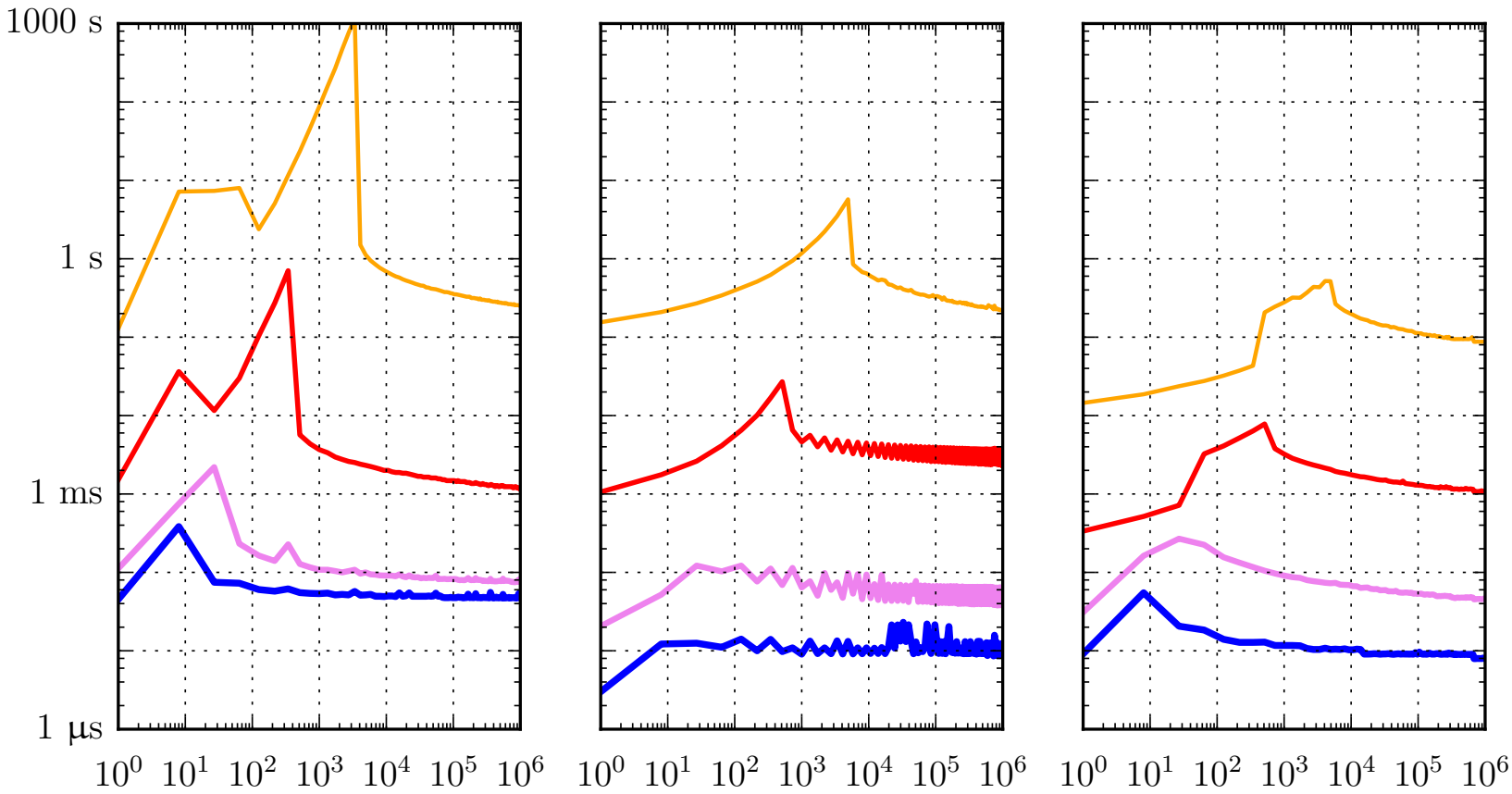
High-order parameter derivatives

Time in seconds to compute $\partial_\nu^{(n)} J_\nu(\pi)$ at $\nu = 1$ to 1000 digits.

n	Mathematica `N[]`	Maple `fdiff()`	Arb
1	73	2.3	0.0010
2	124	1.2	0.0043
3	> 3600	2.8	0.0067
10		51	0.019
20		384	0.018
30		1082	0.023
100			0.29
1000			7.3
10000			1739

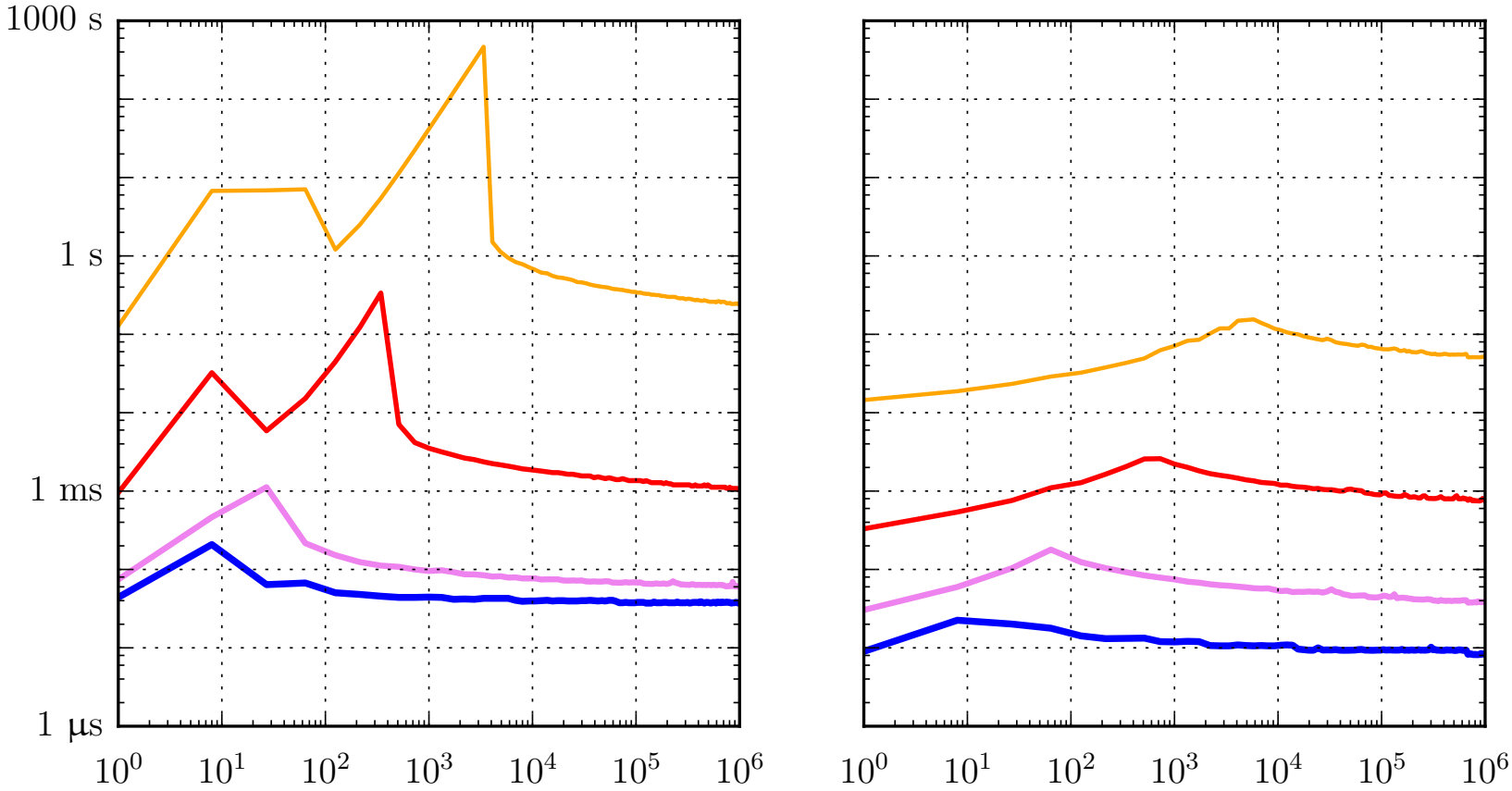
Arb: evaluating $((\pi/2)^{1+x}/\Gamma(2+x)){}_0F_1(2+x, -(\pi/2)^2)$ in $\mathbb{C}[[x]]/\langle x^{n+1}\rangle$ and repeatedly doubling the precision.

Large arguments: $J_0(\pi x)$ on $1 \le x \le 10^6$ (oscillatory)



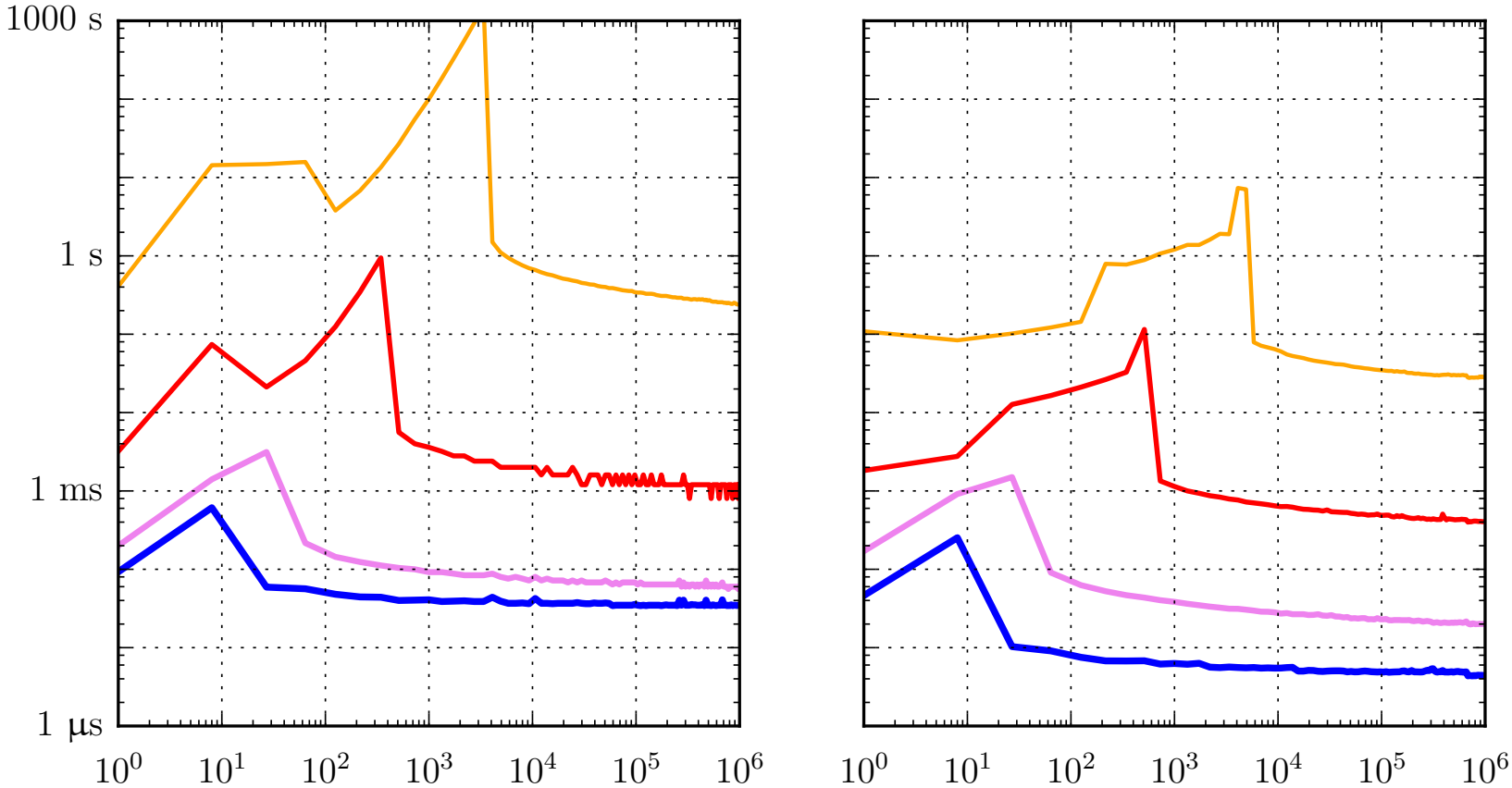
d-digit accuracy, for $d = 10$, 100, 1000, 10000.
Left to right: Mathematica, MPFR, Arb.

Large arguments: $I_0(\pi x)$ on $1 \le x \le 10^6$ (non-oscillatory)



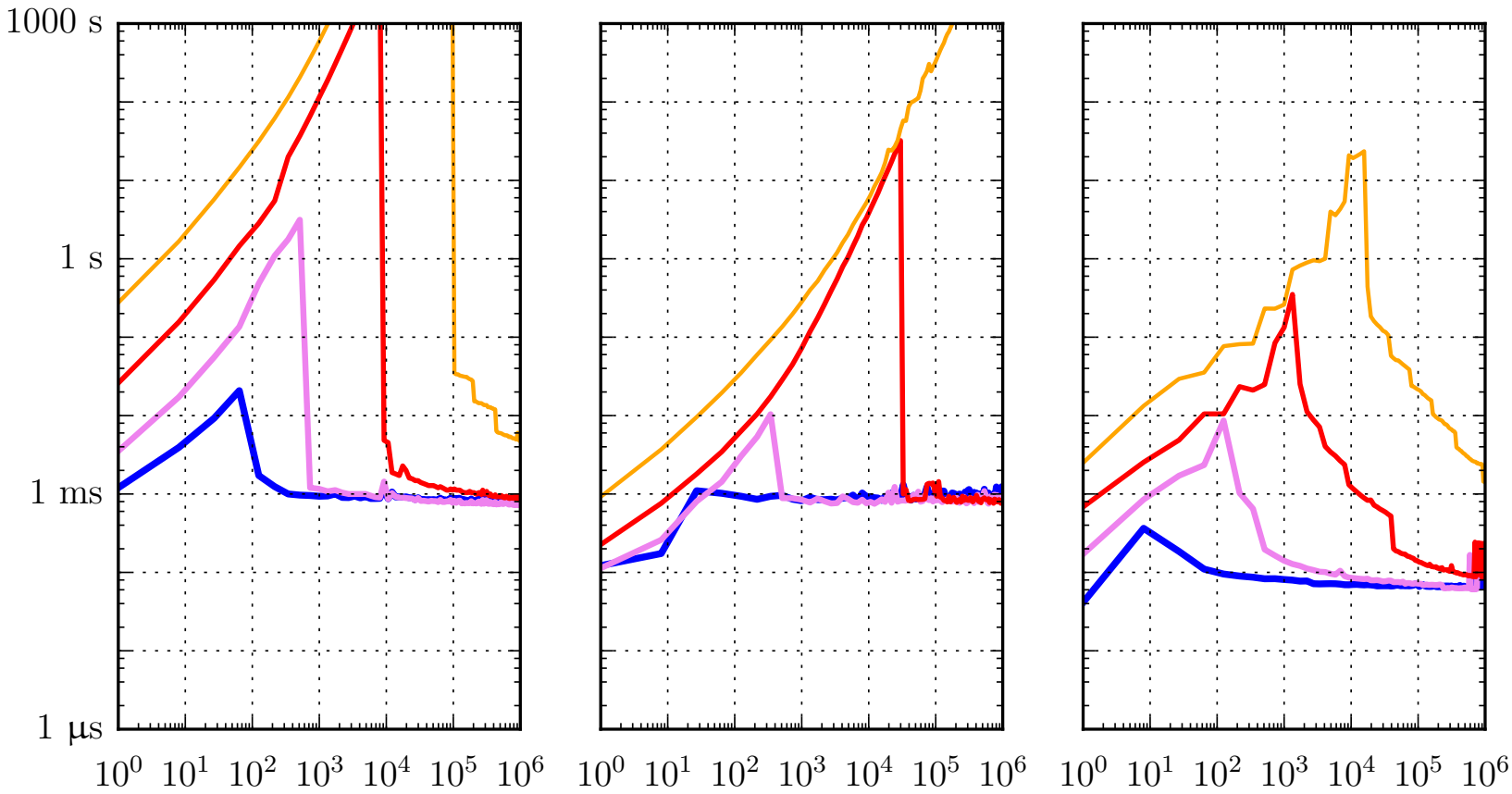
d-digit accuracy, for $d = 10$, 100, 1000, 10000.
Left: Mathematica, right: Arb.

Large arguments: $K_0(\pi x)$ on $1 \le x \le 10^6$



d-digit accuracy, for $d = 10$, 100, 1000, 10000.
Left: Mathematica, right: Arb.

Large parameters: ${}_1F_1(Ni, 1+i, e^{\pi i/3}\pi x)$ on $1 \le x \le 10^6$



10-digit accuracy, for $N = 10$, 100, 1000, 10000.
Left to right: Mathematica, mpmath, Arb.

Parameter asymptotics

How to compute ${}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; z)$ when some parameters a_i, b_i are huge?

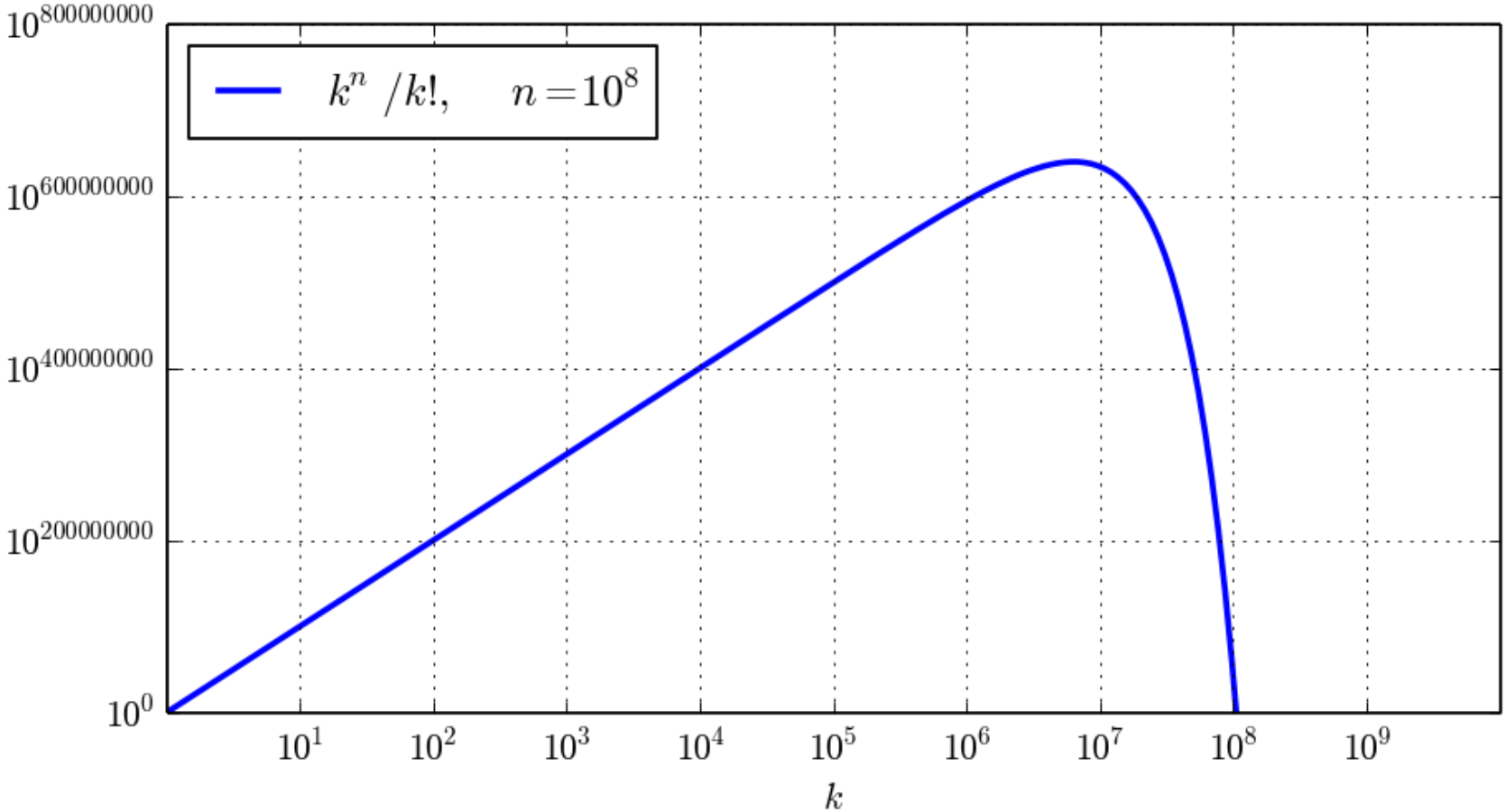
Would like complexity polynomial in $\log|a_i|, \log|b_i|$.

Simplest case: a_i, b_i, z positive real numbers.

Toy problem: *Bell numbers*

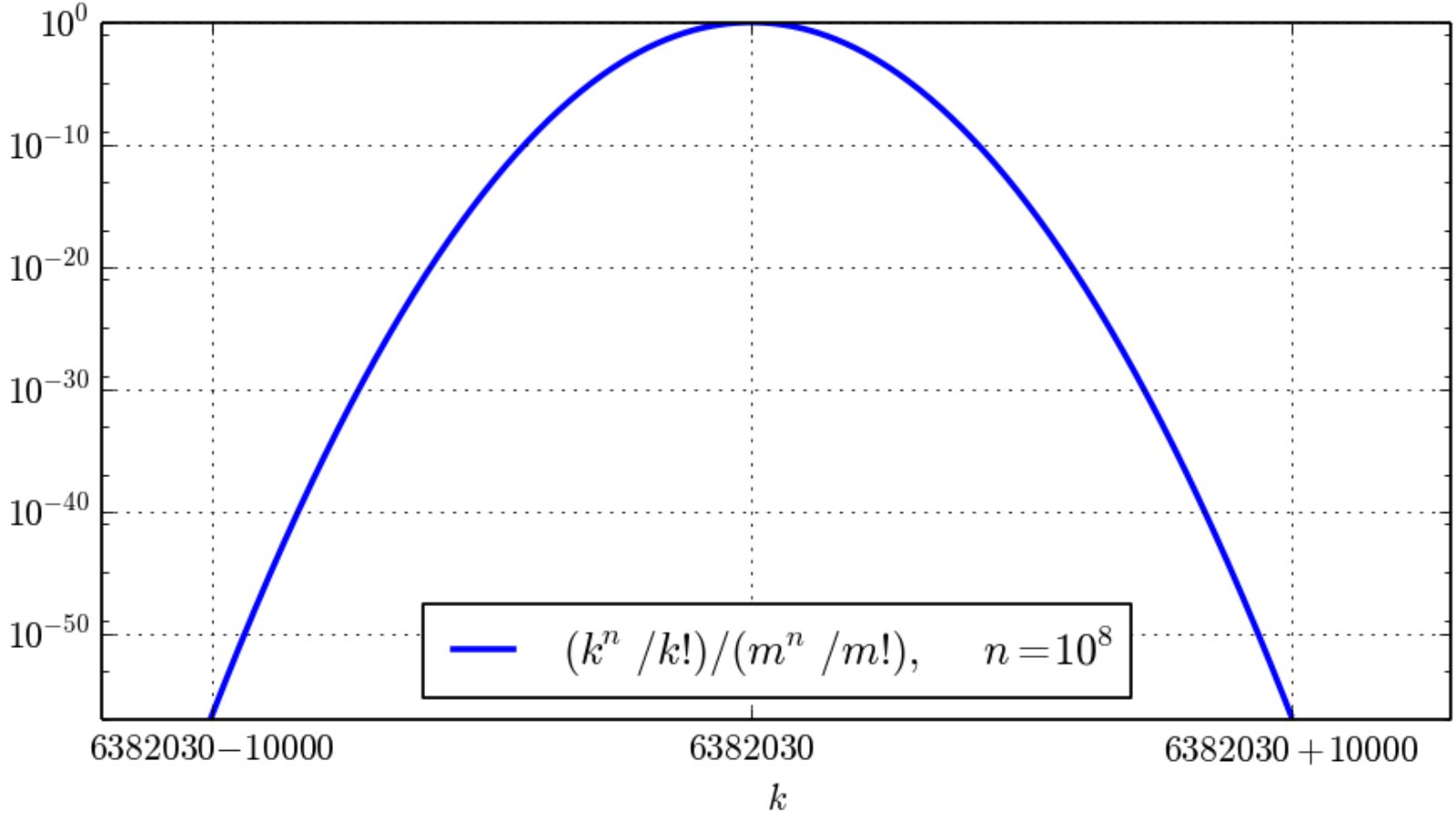
$$B_n = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!} = \frac{1}{e}\left[{}_n\widetilde{F}_n(1, \ldots, 1; 0, \ldots, 0; 1) - 1\right]$$

Bell numbers



The terms peak at $m \approx n/W(n)$. With $n = 10^8$, $m = 6382030$.

The peak zoomed-in



Effective width: $n^{1/2} = 10^4$

Numerical summation

Expand $T(k) = \frac{k^n}{k!} = \frac{k^n}{\Gamma(k+1)}$ in a Taylor series at $k = m$.
Interchange order of summation

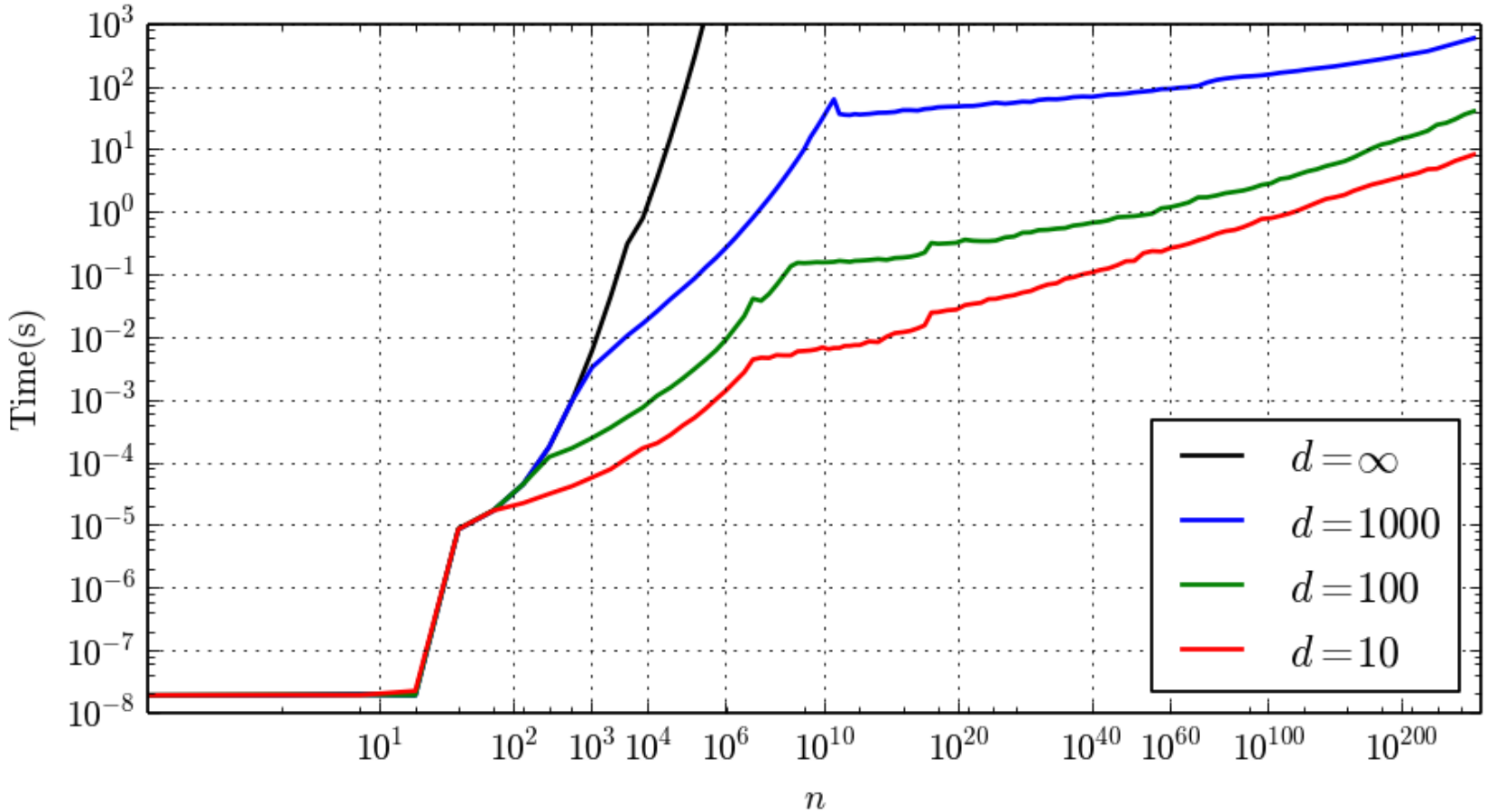
$$\sum_{x=-r}^{r} T(m+x) \approx \sum_{x=-r}^{r} \sum_{i=0}^{N-1} \frac{T^{(i)}(m)}{i!} x^i = \sum_{i=0}^{N-1} \frac{T^{(i)}(m)}{i!} \sum_{x=-r}^{r} x^i$$

$$\sum_{x=a}^{b} x^i = \frac{B_{i+1}(b+1) - B_{i+1}(a)}{i+1}$$

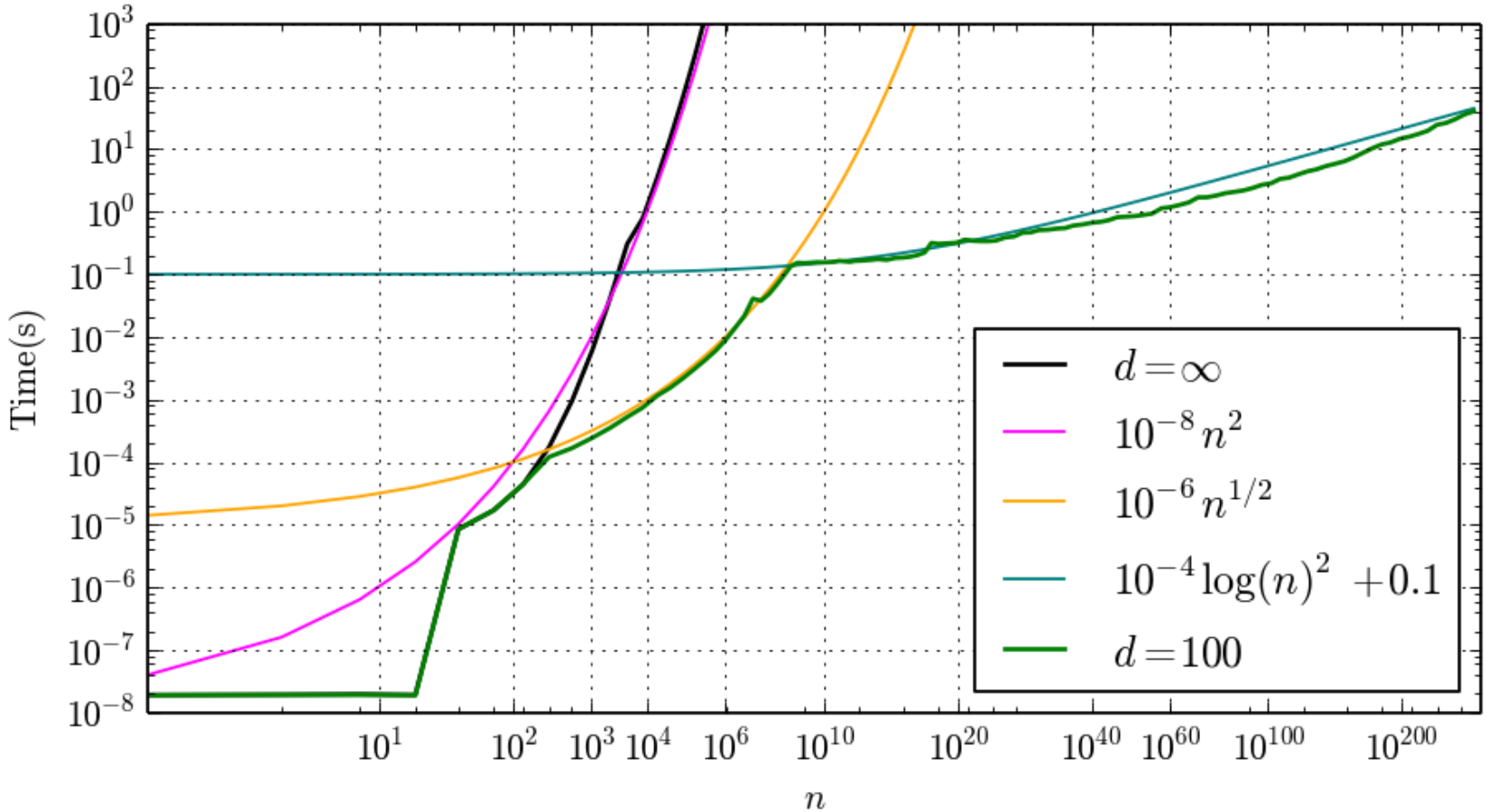
where $\frac{te^{at}}{e^t-1} = \sum_{i=0}^{\infty} B_i(a) \frac{t^i}{i!}$ are Bernoulli polynomials.

A bit of analysis gives error bounds.

Complexity of computing Bell numbers



Complexity of computing Bell numbers



$B_{10^{100}}$ (to 1000 digits)

2.937547415006984869398920080106574949387725148180178490086522814709141082608913334784314134261851806223288894965794626715751776559182743757556852732343576879261884716788878943894806030247861419977972636782577595059940422792450624457536478638747888223064139604389988573291994536205377887499420168234396826778585822533942208369004319687423225479298171524242007012081418958404496161123201363892701142452059296721108807672668911650015364013105771825261645179180673429674151635617615563878966244703690884073862745390209414198878538959777371930234417863546023805302182867063815217446678742129927289601876020900512672752037690916240689650852398468230120896448383085739867131963399875467499177815304267568743466285245985492231607242112207837333676860456717485594671823500441075731431468000467316898340014880056349953760540802402051823297138757018725522572436781900033913367936582937731151484061913797597293903657024242508217267788016980545827744432850309183533085494184420427943318934912041080220073271857 1 $\cdot$ 10^{E}
$\pm$ 4.37 $\cdot$ 10^{E-1000}

E = 97215757485769623537866302743421135921800685850493045081613407617889686798761838992941681528875583562 9

What about negative/complex parameters?

Look at what numerics people are doing. Integral representations + contour deformations seem to be a popular way to avoid cancellation issues.

Possible to use rigorous numerical integration methods?