Computing special functions using integral representations

Fredrik Johansson

Certified and Symbolic-Numeric Computation
ENS Lyon

May 25, 2023

Introduction

Numerical integration is a classical and powerful technique for computing special functions like

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

Requirements:

- I want rigorous error bounds
- Parameters may be inexact ($s = [0.3 \pm 10^{-13}]$)
- I want to easily obtain, say, 10 – 10,000 digits
- I want robust code, suitable for general-purpose math software

In this talk, I will discuss general principles and describe where and how integration is currently used in the Arb library.

Why use integration?

Two ways to compute functions defined by integrals:

1. Use numerical integration (quadrature)
2. Use series expansions derived from integral representations

$$\int f(t)dt = f(t_0) \cdot [C_0 + C_1 + \ldots]$$

Usually (2) leads to faster algorithms (for example, the terms have nice recurrence relations).

However, (1) may be much more straightforward, considering error bounds, cancellation, divergence, branch cuts, ...

Example: one of Carlson's elliptic integrals

$$R_J(a,b,c,d) \;=\; \frac{3}{2}\int_0^{\infty} \frac{dt}{\sqrt{a+t}\sqrt{b+t}\sqrt{c+t}\,(d+t)}$$

Expansion algorithm: defining $\lambda = \sqrt{a}\sqrt{b} + \sqrt{b}\sqrt{c} + \sqrt{a}\sqrt{c}$, iterate

$$R_J(a,b,c,d) \;=\; \tfrac{1}{4} R_J\left(\tfrac{a+\lambda}{4}, \tfrac{b+\lambda}{4}, \tfrac{c+\lambda}{4}, \tfrac{d+\lambda}{4}\right) \;+\; \text{arctangent}$$

until $a \approx b \approx c \approx d$. Then $R_J(a, a+\varepsilon, \ldots) =$ hypergeometric series.

Example: one of Carlson's elliptic integrals

$$R_J(a,b,c,d) \;=\; \frac{3}{2}\int_0^{\infty} \frac{dt}{\sqrt{a+t}\sqrt{b+t}\sqrt{c+t}\,(d+t)}$$

Expansion algorithm: defining $\lambda = \sqrt{a}\sqrt{b} + \sqrt{b}\sqrt{c} + \sqrt{a}\sqrt{c}$, iterate

$$R_J(a,b,c,d) \;=\; \tfrac{1}{4} R_J\!\left(\tfrac{a+\lambda}{4}, \tfrac{b+\lambda}{4}, \tfrac{c+\lambda}{4}, \tfrac{d+\lambda}{4}\right) \;+\; \text{arctangent}$$

until $a \approx b \approx c \approx d$. Then $R_J(a, a+\varepsilon, \ldots) =$ hypergeometric series.

This is only valid for certain parameters.
For example, it is sufficient that $\mathrm{Re}(a), \mathrm{Re}(b), \mathrm{Re}(c), \mathrm{Re}(d) > 0$.

Arb 2.17 (always using the above algorithm):
$R_J(-1-0.5i, -10-6i, -10-3i, -5+10i) \approx 0.345986 + 0.399031i$

Arb 2.18 (using integration when the preconditions do not hold):
$R_J(-1-0.5i, -10-6i, -10-3i, -5+10i) \approx 0.128471 + 0.102176i$

Example: incomplete beta function with large parameters

A user reported that Arb struggles to evaluate
$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1}, \quad x = \frac{4999}{10000},\ a = b = 10^5.$

Arb 2.21 (only using series expansions):

```
prec = 64:     [+/- inf]
prec = 128:    [+/- inf]
prec = 256:    [+/- inf]
...
prec = 65536:  [+/- inf]
prec = 131072: [+/- inf]
prec = 262144: [0.46436508135202051998898147610...]
```

Arb 2.22 (using numerical integration in appropriate cases):

```
prec = 64:  [0.4643650813520 +/- 4.92e-14]
prec = 128: [0.46436508135202051998898147610644 +/- 3.74e-33]
```

General principles

Choice of integral and path:

- Handling singularities
- Avoiding cancellation / oscillation

Choice of integration algorithm:

- Gaussian quadrature
- Trapezoidal and double exponential quadrature
- Taylor series

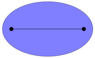
Implementation issues:

- Error bounds, stability, efficiency, code complexity

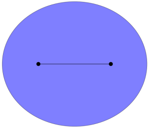
Gauss-Legendre quadrature

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

$$\left| \int_{-1}^{1} f(x) dx - \sum_{k=1}^{n} w_k f(x_k) \right| \leq \frac{M}{\rho^{2n}} \cdot \frac{4.27}{1 - \rho^{-1}}.$$



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

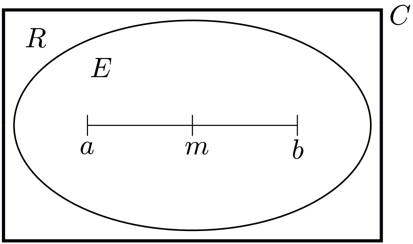


$X = 2.00, Y = 1.73, \rho = 3.73$

There is a rigorous, arbitrary-precision, adaptive implementation in Arb since 2017, accepting "black box" integrands f.

Implementation in Arb: bounds and adaptivity

- To bound $|f(z)|$ on E, we may evaluate f on a rectangle R. This can be done in low-precision interval arithmetic.

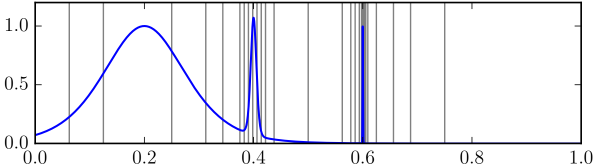


- Note: it suffices to bound $|f(z)|$ on the contour C.
- The integrand must check "$f(z)$ is analytic on R".
- Arb tests a set of semi-axes $X \sim 2^k|b-a|$ and degrees $n = 1, 2, 4, 6, 8, \ldots \sim 2^{\ell/2}$, choosing the smallest n such that $|I_n - I| < \varepsilon_{\text{tol}}$. If no good (X, n) is found, $[a, b]$ is bisected.

Adaptive subdivision

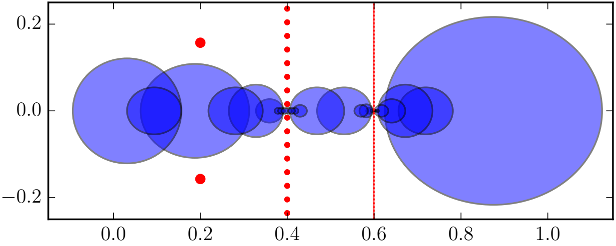
$$\int_0^1 \operatorname{sech}^2(10(x-0.2)) + \operatorname{sech}^4(100(x-0.4)) + \operatorname{sech}^6(1000(x-0.6))\; dx$$

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



Complex ellipses
used for bounds

Red dots = poles



Magnitude bound gotchas

Wrapping and dependency problems

- Evaluating expressions like $f(z) = g(z)/h(z)$ or $g(z) - h(z)$ naively in interval arithmetic can give extremely poor upper bounds, resulting in slow convergence
- Solution: when the input z is given by a wide interval, use a short Taylor expansion or rewrite symbolically
- Use $\exp(-z)$ instead of $1/\exp(z)$

Troublesome branch cuts

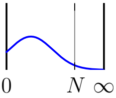
- Near $(-\infty, 0)$, replace $\log(z) \to \log(-z) - \pi i$, etc.

Tails and endpoint singularities

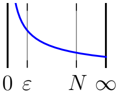
If $|a|$, $|b|$ or $|f| \to \infty$, we can no longer get automatic error bounds out of Gauss-Legendre quadrature + interval arithmetic.

Solutions:

- Truncation $\int_0^\infty f(x)dx \approx \int_\varepsilon^N f(x)dx$



Exponential decay

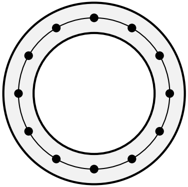


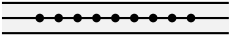
Algebraic blow-up/decay

- Gauss-Jacobi, Gauss-Laguerre, etc.

Either way, manual or symbolic preprocessing is needed.

Gauss's main competitor: the trapezoidal rule





$$\int_{|z|=1} f(t)dt \approx \frac{2\pi}{N} \sum_{k=1}^{N} f(e^{2\pi ik/N})$$

$$|I_N - I| \leq \frac{4\pi M}{r^N - 1}$$

$$M = \max_{1/r<|z|<r} |f(z)|$$

$$\int_{-\infty}^{\infty} f(t)dt \approx h \sum_{k=-\infty}^{\infty} f(hk)$$

$$|I_h - I| \leq \frac{2M}{e^{2\pi a/h} - 1}$$

$$M = \max_{|y|\leq a} \int_{-\infty}^{+\infty} |f(x+iy)|dx$$

For other contours and bounds, see:

- Trefethen and Weideman, *The exponentially convergent trapezoidal rule*, 2014
- P. Molin, *L'intégration numérique par la méthode double-exponentielle*, 2016

Double exponential (tanh-sinh) quadrature

What is the optimal rate of decay for using the trapezoidal rule to compute $\int_{-\infty}^{\infty} f(t)dt$? Under certain assumptions,

$$|f(t)| < \exp(-|t|) \implies |I_N - I| < \exp(-cN^{1/2})$$

$$|f(t)| < \exp(-\exp(|t|)) \implies |I_N - I| < \exp(-cN/\log(N))$$

This leads, for example, to the "tanh-sinh rule"

$$\int_{-1}^{1} f(t)dt = \sum_{k=-\infty}^{\infty} w_k f(x_k), \quad x_k = \tanh(\tfrac{\pi}{2}\sinh(kh)).$$

This method is remarkably robust for functions with endpoint singularities, e.g. $f(t) = (1-t)^{\alpha}(1+t)^{\beta}g(t)$.

Disadvantages of double exponential quadrature

- Less efficient than Gauss-Legendre on compact intervals.
- Not locally adaptive: efficiency deteriorates with singularities close to the path.
- Can be tricky to find a suitable variable transformation (passing through saddle points, avoiding singularities, etc.).
- Can be tricky to bound the error.

For these reasons, I am not using the double exponential method anywhere in Arb. However, there are certainly situations where it would be useful (some examples later).

Implementation example: hypergeometric functions

Arb uses numerical integration in some cases to compute hypergeometric functions via the representations

$$ {}_1F_1(a,b,z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)} \int_0^1 e^{zt} t^{a-1} (1-t)^{b-a-1} dt, $$

$$ U(a,b,z) = \frac{1}{\Gamma(a)} \int_0^\infty e^{-zt} t^{a-1} (1+t)^{b-a-1} dt, $$

$$ {}_2F_1(a,b,c,z) = \frac{\Gamma(a)}{\Gamma(b)\Gamma(c-b)} \int_0^1 t^{b-1} (1-t)^{c-b-1} (1-zt)^{-a} dt. $$

By extension: modified Bessel functions $I_\nu(x), K_\nu(x)$ and incomplete gamma and beta functions $\Gamma(s,x), \gamma(s,x), I_x(a,b)$.

Currently only real parameters are considered.

Practical issue: finding a good tolerance

Consider the ${}_1F_1$ integral: $I = \int_0^1 \exp(g(t))dt$ where

$$g(t) = zt + (a-1)\log(t) + (b-a-1)\log(1-t)$$

$$g'(t) = t + \frac{a-1}{t} - \frac{b-a-1}{1-t}$$

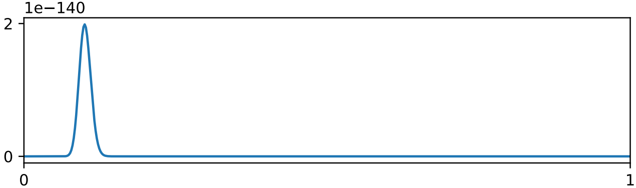
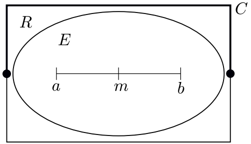


Figure: $g(t), \quad a = 100, \quad b = 1000, \quad z = 10$

If the peak is narrow, numerical integration with a relative tolerance becomes inefficient. We can use $I \approx \exp(g(t_{\max}))$ to find an accurate *absolute* tolerance.

Practical issue: local error bounds



Evaluating $g(R)$ naively gives poor bounds. Ditto for $g(m) + g'(R)(R - m)$ and $\ldots + \frac{1}{2}g''(m)(R - m)^2$.

What I found to work is first-order Taylor expansions on C, using

$$\mathrm{Re}(g(u + vi)) = h(u, v),$$

$$\frac{d}{du}h(u, v) = z + \frac{u(a-1)}{u^2 + v^2} + \frac{(u-1)(b-a-1)}{v^2 + (1-u)^2}, \text{etc.}$$

Using machine-precision interval arithmetic + a few subdivisions, this works well up to $a, b, z \approx 10^{15}$.

Implementation example: Laurent coefficients of $\zeta(s)$

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

$\gamma_{10^{100}} \approx 3.187 \cdot 10^{234639429227725408094936783839909116090344768986983738520577}$

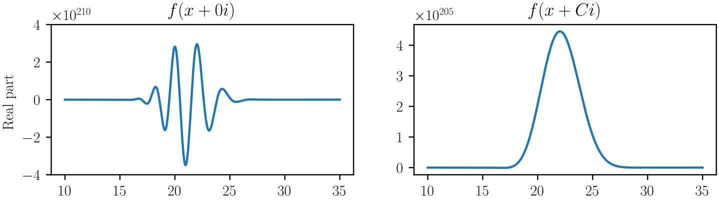


Figure: The integrand with $n = 500$

Piecewise linear path through the saddle point + integrand bounds of the type $\exp(g(m \pm r)) \leq \exp(g(m)) \exp(g'(m)r + Cr^2)$.

What about something more complex? Bessel functions?

For $\mathrm{Re}(z) > 0$,

$$J_\nu(z) = \frac{1}{2\pi i} \int_{-\infty - i\pi}^{-\infty + i\pi} \exp(g(t))dt, \quad g(t) = -z\sinh(t) + \nu t.$$

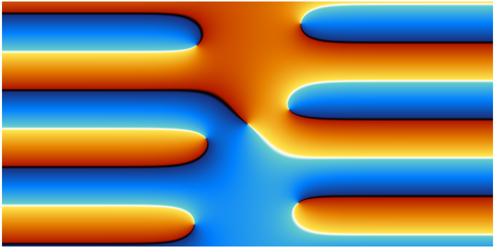


Figure: $\mathrm{sgn}(g(t))$ on $t \in [-20, 20] + [-10, 10]i$; $\nu = 200 + 100i, z = 50 - 20i$

All cases of $J_\nu(z)$, $Y_\nu(z)$, $I_\nu(z)$, $K_\nu(z)$, $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$ can be expressed using similar integrals.

What about something more complex? Bessel functions?

There is a lot of literature on asymptotic expansions, but it seems difficult to cover all (large, complex) combinations of ν, z.

In principle, it should be possible to cover all cases using numerical integration with an approximate steepest-descent contour.

There is a sketch of an algorithm in Jentschura and Lötstedt (2012), but it appears to be buggy.

Counterexample: $\nu = 200 + 100i, z = 50 - 20i$. Here J & L seemingly want to go through both saddle points

- $t_+ \approx +2.12 + 0.86i, \quad |\exp(g(t_+))| \approx 6.47 \cdot 10^{+60}$
- $t_- \approx -2.12 - 0.86i, \quad |\exp(g(t_-))| \approx 1.55 \cdot 10^{-61}$

but $J_\nu(z) \approx 1.33 \cdot 10^{-63} + 3.89 \cdot 10^{-63} i$.

What about something more complex? Bessel functions?

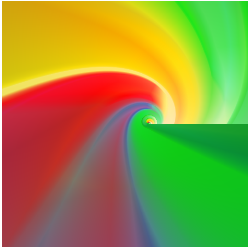
The integrand already has double exponential decay, so the trapezoidal rule is useful at least in some cases, e.g.

$$K_\nu(x) = \int_0^\infty e^{-x\cosh(t)} \cosh(\nu t)dt.$$

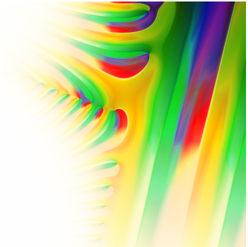
Implementation example: the Lerch transcendent

$$\Phi(z,s,a) = \sum_{n=0}^{\infty} \frac{z^n}{(n+a)^s}, \quad |z| < 1$$

$$\zeta(s) = \Phi(1,s,1), \quad \zeta(s,a) = \Phi(1,s,a), \quad \mathrm{Li}_s(z) = z\Phi(z,s,1)$$



$\Phi(z, 1+3i, 2-i)$ on
$s \in [-5,5] + [-5,5]i$



$\Phi(-0.75i, s, 1-0.5i)$ on
$s \in [-20,20] + [-20,20]i$

New function in Arb 2.23. Algorithms for $\zeta(s)$ etc. can be generalized to $\Phi(z,s,a)$, but this would have been a lot of work.

Analytically continuing $\Phi(z, s, a)$

For $\mathrm{Re}(a) > 0$ and $z \notin [1, \infty)$ (Laplace integral):

$$\Phi(z, s, a) = \frac{1}{\Gamma(s)} \int_0^\infty \frac{t^{s-1} e^{-at}}{1 - ze^{-t}} dt, \quad s \in \{1, 2, 3, \ldots\}$$

Hankel integral:

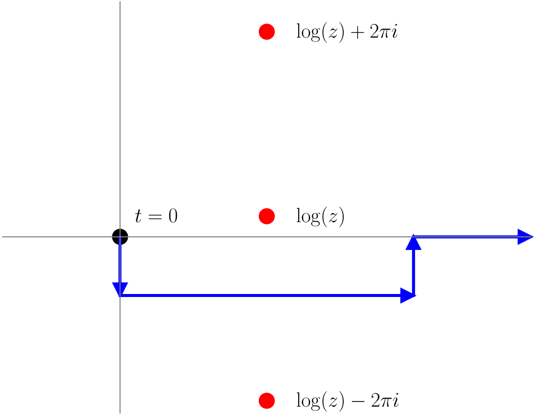
$$\Phi(z, s, a) = -\frac{\Gamma(1-s)}{2\pi i} \int_H \frac{(-t)^{s-1} e^{-at}}{1 - ze^{-t}} dt, \quad s \notin \{1, 2, 3, \ldots\}$$

To remove the restriction on a, we can use

$$\Phi(z, s, a) = z^n \Phi(z, s, a + n) + \sum_{k=0}^{n-1} \frac{z^k}{(k+a)^s}.$$

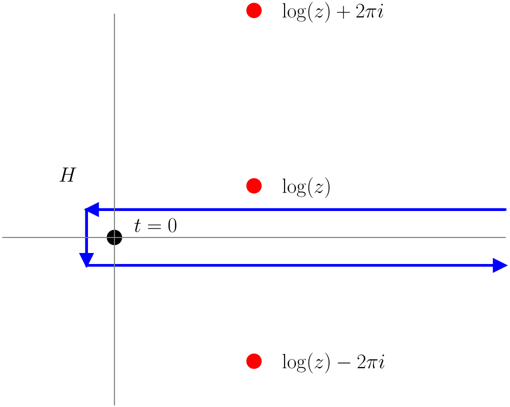
To remove the restriction on z, we can change the path.

Avoiding poles: the Laplace integral



The integrand has poles at $t = \log(z) + 2\pi i k$.

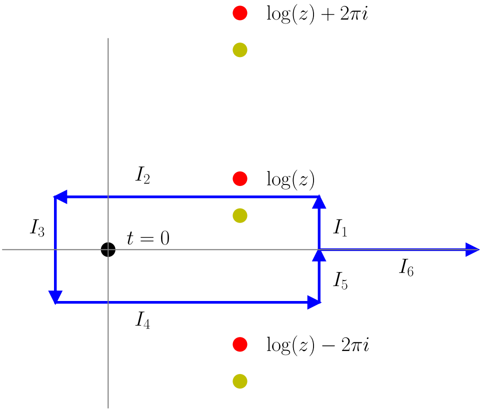
Avoiding poles: the Hankel integral



The integrand has poles at $\log(z) + 2\pi ik$ and a singularity at $t = 0$.

Note: to avoid the branch cut for $(-t)^{s-1}$ in the Gauss-Legendre bounds, we use t^{s-1} for $\text{Re}(t) > 0$ and $(-t)^{s-1}$ for $\text{Re}(t) < 0$

Avoiding poles: the Hankel integral



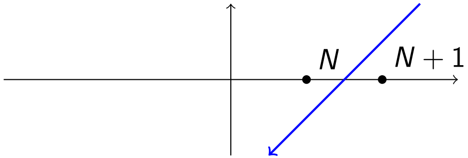
If a pole is too close to the real axis (yellow dot in the figure), integrate around it and subtract the residue.

Large parameters: the Riemann-Siegel formula

I have made no attempt to optimize $\Phi(z,s,a)$ for large parameters. In general, this looks complicated.

The most interesting case is when $\mathrm{Im}(s) \to \infty$. Here, we could use (various versions of) the *Riemann-Siegel formula*. For the classical case of $\zeta(s)$ it involves the following:

$$\zeta_{RS}(s) = \sum_{n=1}^{N} \frac{1}{n^s} + \int_{N \swarrow N+1} \frac{z^{-s} e^{\pi i z^2}}{e^{\pi i z} - e^{-\pi i z}} dz, \quad N = \lfloor \sqrt{\mathrm{Im}(s)/(2\pi)} \rfloor$$



Large parameters: the Riemann-Siegel formula

Usually one derives an asymptotic series for the integral. There is an implementation for $\zeta(s)$ in Arb, but the terms and error bounds are quite messy (Arias de Reyna, 2011).

Recently, Sandeep Tyagi has found[1] an effective way to apply double exponential quadrature directly to the integral.

This method allows computing $\zeta(s)$, $L(s, \chi)$, $\Phi(z, s, a)$ etc. with arbitrary precision and is remarkably simple and efficient. It remains to work out complete, rigorous error bounds.

[1] Sandeep Tyagi (2022), *Double Exponential method for Riemann Zeta, Lerch and Dirichlet L-functions*, `https://arxiv.org/abs/2203.02509`

Taylor series and the bit-burst algorithm

Using high-order Taylor expansions to integrate $\int_a^b f(t)dt$ is most useful for holonomic integrands f, where the "bit-burst algorithm" can be applied to compute D digits in time $D^{1+o(1)}$.

Arb uses the bit-burst algorithm for the following functions:

- Elementary functions
- $\operatorname{erf}(z) = (2/\sqrt{\pi}) \int_0^z e^{-t^2} \, \mathrm{d}t$, $\operatorname{erfc}(z)$, $\operatorname{erfi}(z)$
- $\Gamma(s, z)$ in some cases
- The dilogarithm $\operatorname{Li}_2(z) = -\int_0^z \log(1-t)/t \, dt$
- Indirectly, Dirichlet L-functions for special values

For general holonomic functions, see Marc Mezzarobba's implementation in `ore_algebra`.

Wishlist

- Optimizations for “low” precision (around machine precision).
- Automatic code generation, symbolic precomputation.
- Double exponential quadrature with semi-automatic error bounds.
- Robust implementations of standard integrals (e.g. Bessel functions) for large complex parameters.