Verified computations of the polylogarithm

Matthias Goerner

Abstract. We review existing formulas to show how to compute a complex interval arithmetic polylogarithm for integral orders and related functions giving the (complex) volume of hyperbolic 3-manifolds.

1 Introduction

The polylogarithm is defined as the analytic continuation of the series

$$\mathrm{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n} \tag{1.1}$$

It is a generalization of the logarithm with $\mathrm{Li}_1(z) = -\ln(1-z)$. Li_2 is known as the dilogarithm - the only function with a "sense of humor" according to Zagier's excellent survey article [Zag].
We will review the necessary identities and series and give error estimates to implement a complex interval version of the polylogarithm for integral orders.
The input to such a polylogarithm is the integeral order n and a complex interval which is a rectangle $K = \left[\underline{a}, \overline{a}\right] + \left[\underline{b}, \overline{b}\right] i \subset \mathbb{C}$ in the complex plane where $\underline{a}, \overline{a}, \underline{b}$, and $\overline{b}$ are real numbers exactly representable by floating point numbers. A correct implementation must return another complex interval L such that for every $z \in K$, we have $\mathrm{Li}_n(z) \in L$ — at least as long as K does not intersect the branch cut at $[1, \infty)$, see Chapter **??**. The returned interval L should be reasonably small but does not need to be the smallest interval possible with this property.
We use methods similar to the ones in the Pari [Par] library, but deploy interval arithmetic. A technical report similar to this one is [Woo].

Before describing these methods, we review related functions that can be applied to compute volumes of hyperbolic 3-manifolds.

2 Semantics of complex interval arithmetics

2.1 Differences to MPFI. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. We say that an interval implementation of f is correct if it returns an interval J given some real input intervals $I_1, \dots I_n$ such that $f(x_1, \dots, x_n) \in J$ whenever $x_j \in I_j$ for $j = 1, \dots, k$. Note that this does not require the interval implementation to

(a) be deterministic, i.e., two calls of the same functions given the same inputs might return two different intervals

(b) preserve inclusions, i.e., if $I'_j \subset I_j$ for $j = 1, \dots, k$, then $J' \subset J$ where J' is returned when giving the I'_j as inputs.

(c) return the smallest possible interval that has as endpoints floating-point numbers of the given precision.

The MPFI library implements real interval arithmetics and most (if not all) of its function satisfy the strongest condition (c). Given how hard it is to achieve the analogous conditions for complex functions and intervals, a library for complex interval arithmetic might choose to implement functions in such a way that the returned intervals are still correct but only fulfill condition (a).

2.2 Interpretations of multi-valued functions. The MPFI library implements the $\arg(y, x)$ function which returns the signed angle of the x axis and the line segment from the origin to the point (x, y). If we give it as input intervals $[-1, -1]$ for x and $[-0.0625, 0.0625]$ for y, it returns $[\underline{-\pi}, \overline{\pi}]$.
In a complex analysis setting, it is more natural to think of a function such as log (whose imaginary part is given by arg) as being multi-valued. The log function is related to the exp function which is a covering map $\mathbb{C} \to \mathbb{C} \setminus 0$. A complex interval implementation can be thought of as returning a particular lift of a given complex interval in $\mathbb{C}\setminus 0$ to $\mathbb{C}$. This requires a choice and it is canonical to pick a lift that agrees with the logarithm's principal value if the complex interval avoids the branch cut at $(-\infty, 0)$.
Some examples of the results returned by the complex interval logarithm will make this more clear (note that (b) is not satisfied):

z	$\log(z)$
[-1,-1]+[0.0625, 0.0625]i	[0.00194, 0.00196] + [3.0791, 3.080]i
[-1,-1]+[0 , 0.0625]i	[0 , 0.00196] + [3.0791, 3.1416]i
[-1,-1]+[-0.0625, 0.0625]i	[0 , 0.00196] + [3.0791, 3.2041]i
[-1,-1]+[-0.0625, 0]i	[0 , 0.00196] + [3.1415, 3.2041]i
[-1,-1]+[-0.0625, -0.0625]i	[0.00194, 0.00196] + [-3.080 , -3.0791]i

This is justified by the fact that clients calling log or $\sqrt{z}$ (which can be implemented as $\exp(\log(z)/2)$) often care about the result only up to multiplies of $2\pi i$, respectively, sign.
Evaluating log on a complex interval containing 0 is not allowed which can be signaled by returning NaN (Not a Number).
Similarly, the polylogarithm has a singularity at 1 and the branch cut of polylogarithm is typically chosen to be $(1,\infty)$. For consistency, evaluating the polylogarithm on a complex interval containing 0 should not be allowed either, even though the singularity is such that the principal value of polylogarithm converges as $z \to 1$.

2.3 A subtle pitfall. Let z be some complex quantity such as the cross ratio of an ideal tetrahedron. While an algorithm using the complex interval library typically does not care which of the two values for $\sqrt{z}$ the library chooses, it might still require that every occurrence of $\sqrt{z}$ returns the same choice.

While algorithms implemented by a client of a complex interval library typically do not care about the choice of $\sqrt{z}$, they still might require that whenver

Mathematically, the following holds

$$c = \lambda_0\lambda_0' = \lambda_1\lambda_1'$$

One choice $\sqrt{c}$. Interval implementation: $\sqrt{\lambda_0\lambda_0'}$ and $\sqrt{\lambda_1\lambda_1'}$.

3 Related functions

The dilogarithm

$$\mathrm{Li}_2(z) = -\int_0^z \frac{\log(1-t)}{t}\,dt$$

can be used to compute the Lobachevsky function

$$L(\theta) = -\int_0^\theta \log|2\sin(t)|\,dt = \frac{1}{2}\mathrm{Im}\left(\mathrm{Li}_2\left(e^{2i\theta}\right)\right),$$

the Bloch-Wigner function

$$D(z) = \mathrm{Im}\left(\mathrm{Li}_2(z)\right) + \arg(1-z)\log|z|,$$

and Roger's dilogarithm

$$\mathscr{R}(z) = \mathrm{Li}_2(z) + \frac{1}{2}\log(z)\log(1-z)$$

which Neumann [**?**] uses to define

$$R(z;p,q) = \mathscr{R}(z) + \frac{\pi i}{2}\left(p\log(1-z) + q\log(z)\right) - \frac{\pi^2}{6}.$$

The Lobachevsky function and Bloch-Wigner function can be used to compute the volume of a hyperbolic ideal 3-simplex from its three dihedral angles, respectively, its cross-ratio as explained by Milnor [**?**]. Neumann's $R(z;p,q)$ relates to the extended Bloch group and can be used to compute the complex volume of a hyperbolic 3-manifold.

4 General strategy

If $n \leq 0$, Li_n is actually a rational function which we treat separately in Section **??**. For $n = 1$, we have $\mathrm{Li}_1 = -\ln(1-z)$, so we are only concerned about orders greater than 1 here.
If $|z| < 1$, we can use the defining series (1.1) and, if $|z| > 1$, Jonquière's inversion formula [Jon]

$$\mathrm{Li}_n(z) = -\left(\frac{(2\pi i)^n}{n!}B_n\left(\frac{\ln(z)}{2\pi i}\right) + (-1)^n\mathrm{Li}_n(1/z)\right) \tag{4.1}$$

reduces the problem to the case $|z| < 1$ where

$$B_n(x) = \sum_{k=0}^{n}\binom{n}{k}b_{n-k}x^k = -\sum_{k=0}^{n}\frac{n(n-1)\ldots(n-k)}{k!}\zeta(k-n+1)x^k$$

is the Bernoulli polynomial and b_i are the Bernoulli numbers.
The resulting series converges arbitrarily slow though when $|z|$ is close to 1. In this case, we can use the following series for $z = e^{\mu}$ with $|\mu| < 2\pi$

$$\mathrm{Li}_n(e^{\mu}) = \frac{\mu^{n-1}}{(n-1)!}\left[\sum_{h=1}^{n-1}\frac{1}{h} - \ln(-\mu)\right] + \sum_{k=0,k\neq n-1}^{\infty}\frac{\zeta(n-k)}{k!}\mu^k. \tag{4.2}$$

Section **??** discusses the interval implementation of these series giving error bounds. This also enables us to analyze the convergence speed of these series in Section **??**

which tells us which series to pick for a given z to make the computation of the polylogarithm faster.
If the given interval for z is so large that none of $|z|<1$, $|z|>1$, or $|\mu|<2\pi$ hold, an interval implementation can simply give $(-\infty,\infty)+(-\infty,\infty)i$ as result.

4.3 Remark. None of the series performs well (gaining less than 2 bits precision per term) when z is close to -0.3 (see Figure 6.2) and there is a potential optimization by using the duplication formula [**?**, **?**]

$$\mathrm{Li}_n(z) = 2^{1-n}\mathrm{Li}_n(z^2) - \mathrm{Li}_n(-z).$$

5 Error bounds on the series

5.1 Error bounds. Let $E[r] = [-r,r]+[-r,r]i$. For the series about 0 from Equation 1.1, we have $|t_k+1| < r|t_k|$ with $r=|z|$ for two subsequent terms. This means that the absolute value of the sum of the terms after t_u is bounded by a geometric series with common ratio r and we obtain

$$\mathrm{Li}_n(z) \in \sum_{k=1}^{u} t_k + E[F\cdot|t_u|] \text{ where } t_k = \frac{z^k}{k^n} \text{ and } F = \frac{r}{1-r} \text{ with } r=|z|. \tag{5.2}$$

Similarly, we have for the series about 1 from Equation 4.2

$$\mathrm{Li}_n(e^\mu) \in \frac{\mu^{n-1}}{(n-1)!}\left[\sum_{k=1}^{n-1}\frac{1}{k} - \ln(-\mu)\right] + \sum_{k=0}^{n-2} t_k + t_n + \sum_{k=n+1}^{u} t_k + E[F\cdot|t_u|] \tag{5.3}$$

$$\text{where } t_k = \frac{\zeta(n-k)}{k!}\mu^k \text{ and } F = \frac{r}{1-r} \text{ with } r = |\mu/(2\pi)|^2$$

when u and n have the opposite parity and $u \geq n+1$. It follows from Riemann's functional equation

$$\zeta(s) = 2^s\pi^{s-1}\sin\left(\frac{\pi s}{2}\right)\Gamma(1-s)\zeta(1-s), \tag{5.4}$$

that every other term $t_{n+2}, t_{n+4}, \ldots$ in the last sum is zero and that the non-trivial terms in the last sum are given

$$t_k = 2(2\pi)^{n-1}(-1)^{\left(\frac{k-n+1}{2}\right)}\frac{\zeta(k-n+1)}{k(k+1)\cdots(k-n+1)}\left(\frac{\mu}{2\pi}\right)^k \text{ for } k = n+1, n+3, \ldots .$$

Since the Riemann ζ-function is descreasing on $(1,\infty)$, we also see that $|t_{k+2}| < r|t_k|$ where $r=|\mu/(2\pi)|^2$ and thus obtain the above error bound $F\cdot|t_u|$.

5.5 Interval arithmetic implementations. To be able to use (5.2) or (5.3), we first need to use interval arithmetics to verify that $|z| < 1$, respectively, $|\log(z)| < 2\pi$.
For all the operations on the right hand side of these two formulas, we apply the corresponding interval versions except for the term $E[F \cdot |t_u|]$ which we need to replace by $E[\overline{F \cdot |t_u|}]$ where $\overline{F \cdot |t_u|}$ is obtained by applying interval arithmetic to compute an interval for $F \cdot |t_u|$ and picking the right endpoint of that interval (since we don't use the left endpoint of the interval, we can save some work by computing only $\overline{F \cdot |t_u|}$ by proper rounding).
More technical notes: An interval implementation could evaluate the right hand side of (5.2) or (5.3) for larger and larger u until the resulting complex interval is not descreasing in size anymore. We also get tighter complex intervals as result by doing intermediate computations in slightly higher precision. Subsequent evaluations of polylogarithms by (5.3) can be sped up significantly by caching the values of the Riemann ζ-function.

6 Convergence speed

We gain about $-\log_2(|z|)$ bits of precision per term when using Equation 5.2 and $-\log_2(1/|z|)$ when combining that equation with the inversion formula. Similarly, we gain about $-\log_2((|\log(z)|/2\pi)^2)$ bits of precision for each non-trivial term when using Equation 5.3.
Thus, we prefer using Equation 5.3 when

$$(|\log(z)|/2\pi)^2 < \min(|z|, 1/|z|). \tag{6.1}$$

Figure 6.2 shows the convergence speed of series (4.2) and the region where this series converges faster than (1.1). The convergence is slowest near -0.3 where both series gain only about 1.7 bits per term.
When we test for (6.1) in code, we do not need to use interval arithmetic or higher precision to verify it since the wrong choice will still be correct, just slower to compute (as long as z in the domain where the chosen series still converges).

7 Non-positive integral orders

For $n \geq 0$, the polylogarithm Li_{-n} is obtained by applying $(n+1)$ times the operator $z\frac{\partial}{\partial z}$ to $\mathrm{Li}_1(z) = -\ln(1-z)$. It is thus a rational function given by

$$\mathrm{Li}_{-n}(z) = \sum_{k=0}^{n} k! \left\{ \begin{matrix} n+1 \\ k+1 \end{matrix} \right\} \left(\frac{z}{1-z}\right)^{k+1}$$

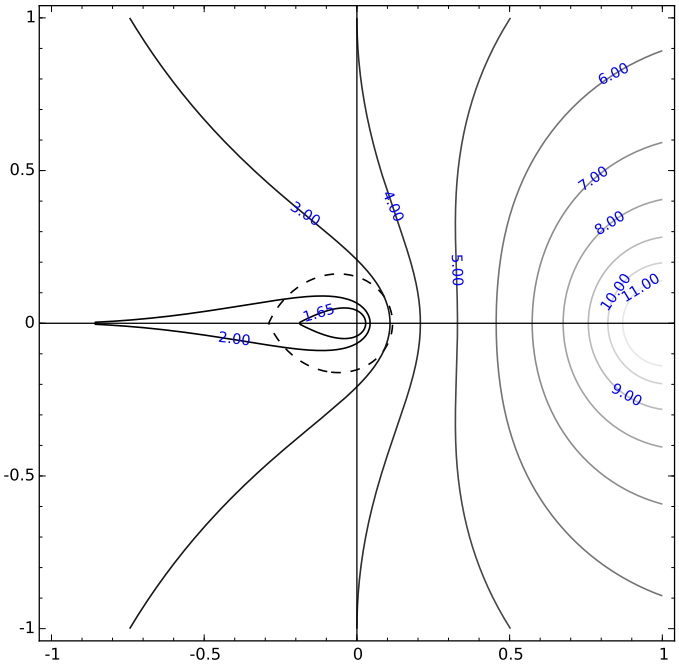

Figure 6.2. Bits precision per non-trivial term t_k in (4.2). The dashed line encloses the region where series (1.1) converges faster than (4.2). The picture is invariant under inversion $z \mapsto 1/z$.

where $\left\{ \begin{matrix} n \\ k \end{matrix} \right\}$ are the Stirling numbers of second kind given explicitly by

$$\left\{ \begin{matrix} n \\ k \end{matrix} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \tag{7.1}$$

or recursively by

$$\left\{ \begin{matrix} n+1 \\ 0 \end{matrix} \right\} = 0, \quad \left\{ \begin{matrix} n+1 \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n \\ k-1 \end{matrix} \right\} \quad \text{for} \quad k > 0$$

and the initial conditions

$$\left\{ \begin{matrix} 0 \\ 0 \end{matrix} \right\} = 1, \quad \left\{ \begin{matrix} 0 \\ k \end{matrix} \right\} \quad \text{for} \quad k > 0.$$

Note that the recursive definition is easy enough for a numerical implementation and also preferable over (7.1) since it avoids the alternating sum that could potentially accumulate larger errors.

8 Bernoulli numbers and Riemann ζ-function

8.1 Bernoulli numbers. The Bernoulli numbers can be defined recursively:

$$b_m = 1 - \sum_{k=0}^{m-1} \binom{m}{k} \frac{b_k}{m-k+1} \quad \text{for} \quad m \geq 0. \tag{8.2}$$

This also allows to compute the Riemann ζ-function for odd negative integers and even positive integers

$$\frac{b_{2n}}{2n} = -\zeta(1-2n) = (-1)^{n+1} \frac{2(2n-1)!}{(2\pi)^{2n}} \zeta(2n) \quad \text{for} \quad n \geq 1.$$

The latter equation directly follows from Riemann's functional equation 5.4 which also implies that ζ is zero at the even negative integers. Furthermore, we have $\zeta(0) = -1/2$.

For our purposes, we can use (8.2) to compute the Bernoulli numbers. Pari employs a different algorithm, but it has different goals: compute a Bernoulli number b_m for much higher m with high enough precision to obtain a rational representation of b_m and without computing all the pervious Bernoulli numbers. Pari's algorithm computes ζ using the Euler product which actually converges quickly if m is large and uses that von Staudt and Clausen give an expression for the numerator of b_m, see [McG, Ste].

8.3 Riemann ζ-function at positive odd integers. To compute higher polylogarithms, we still need a way to compute the values of ζ at the odd positive integers. Ramanujan gave a formula for these:

$$(-\alpha)^{-n} \left[\frac{\zeta(2n+1)}{2} + \sum_{k=1}^{\infty} \frac{k^{-2n-1}}{e^{2\alpha k}-1} \right] - \beta^{-n} \left[\frac{\zeta(2n+1)}{2} + \sum_{k=1}^{\infty} \frac{k^{-2n-1}}{e^{2\beta k}-1} \right] = 2^{2n} \sum_{k=0}^{n+1} (-1)^k \frac{b_{2k}}{(2k)!} \frac{b_{2n+2-2k}}{(2n+2-2k)!} \alpha^k \beta^{n+1-k} \tag{8.4}$$

where n is a positive integer and $\alpha\beta = \pi^2, \alpha, \beta > 0$, proven, for example, by Berndt [Ber]. If $(-\alpha)^{-n} - \beta^{-n} \neq 0$, we can solve for and quickly compute $\zeta(2n+1)$. In particular, for n odd and $\alpha = \beta = \pi$, we obtain

$$\zeta(4m-1) = -\frac{1}{2}(2\pi)^{4m-1} \sum_{k=0}^{2m} (-1)^k \frac{b_{2k}}{(2k)!} \frac{b_{4m-2k}}{(4m-2k)!} - 2 \sum_{k=1}^{\infty} \frac{1}{k^{4m-1}\left(e^{2\pi k}-1\right)}.$$

for $m \geq 1$ which converges quickly ($e^{2\pi} = 535.49...$). For n even, we can take the derivative of equation (8.4) and again set $\alpha = \beta = \pi$ to obtain

$$\zeta(4m+1) = \frac{(2\pi)^{4m+1}}{2m} \sum_{k=0}^{m} (-1)^k (2m+1-2k) \frac{b_{2k}}{(2k)!} \frac{b_{4m+2-2k}}{(4m+2-2k)!} - 2 \sum_{k=1}^{\infty} \frac{e^{2\pi k}(1+\pi k/m)-1}{k^{4m+1}(e^{2\pi k}-1)^2}. \tag{8.5}$$

These equations are also used by pari and, according to Cohen [Coh], can also be proved "by computing the period functions associated to the Eisenstein series of weight $4m+2$ on $\mathrm{PSL}(2,\mathbb{Z})$".

9 STUFF

```
C=contour_plot(
    lambda x,y:-log(abs(log(x+y*I))/6.283)/0.3465735+log(abs(x+y*I))/0.693
    (-1,1),(-1,1),
    axes=True,contours=[0],linestyles='dashed',
    fill=False,plot_points=300)
C2=contour_plot(
    lambda x,y:-log(abs(log(x+y*I))/6.283)/0.3465735,
    (-1,1),(-1,1),
    axes=True,contours=[1.65,2,3,4,5,6,7,8,9,10,11],
    fill=False,plot_points=300, labels=True)
C3=C+C2
```

References

[Ber] B. C. Berndt. Modular transformations and generalizations of several formulae of Ramanujan. *Rocky Mountain Journal of Mathematics* **7** (1977), 147–190. https://projecteuclid.org/download/pdf_1/euclid.rmjm/1250130041

[Coh] H. Cohen. High precision computation of Hardy-Littlewood constants. http://www.math.u-bordeaux1.fr/~hecohen/.

[Jon] A. Jonquière. Note sur la série $\sum_{n=1}^{\infty} \frac{x^n}{n^s}$. *Bulletin de la Société Mathématique de France* **17** (1889), 142–152.

[McG] K. J. McGown. Computing Bernoulli Numbers Quickly, 2005. http://wstein.org/projects/168/kevin_mcgown/bernproj.pdf.

[Ste] W. Stein. Computing Bernoulli Numbers, 2006. http://wstein.org/talks/bernoulli/current.pdf.

[Par] The PARI Group, Bordeaux. *PARI/GP version* 2.7.5, 2015. available from http://pari.math.u-bordeaux.fr/.

[Woo] D. Wood. The Computation of Polylogarithms. Technical Report 15-92*, University of Kent, Computing Laboratory, University of Kent, Canterbury, UK, June 1992. http://www.cs.kent.ac.uk/pubs/1992/110

[Zag] D. Zagier. The Dilogarithm Function. In *Number Theory, Physics, and Geometry II*, pages 3–35. Springer-Verlag, 2007. http://people.mpim-bonn.mpg.de/zagier/files/doi/10.1007/978-3-540-30308-4_1/fulltext.pdf

Matthias Goerner
email: enischte@gmail.com
http://unhyperbolic.org/