Note on fast polylogarithm computation

R. E. Crandall
18 Jan 2006

Abstract: The polylogarithm function $\text{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n}$, manifestly convergent for $|z| \leq 1$, integer $n > 1$, is sometimes numerically/symbolically relevant for $|z| > 1$, i.e. the analytic continuation may be required. By exploiting analytic symmetry relations, we give, for integer $n$, simple and efficient algorithms for complete continuation in complex $z$. 
1. Nomenclature and relations.

The definition
\[ \text{Li}_n(z) := \sum_{k=1}^{\infty} \frac{z^k}{k^n} \] (1.1)
allows rapid computation for small \(|z|\)—one may sum directly. For \(z\) barely inside, or on the unit circle, transformations allow rapid convergence. Outside the unit circle, there are two difficulties: First, there is no absolute convergence, and second, cuts in the complex plane must be carefully considered. So for example, it is known that
\[ \text{Li}_2 \left( \frac{1}{2} \right) = \frac{\pi^2}{12} - \frac{1}{2} \log^2 2, \]
as may be verified numerically by direct summation of (1.1), with a precision gain of about 1 bit per summand. However, it is also known that the analytic continuation has
\[ \text{Li}_2(2) = \frac{\pi^2}{4} - i\pi \log 2, \]
even though the sum (1.1) cannot be performed directly. Incidentally, all along the cut \(z \in [1, \infty]\) there is a discontinuity in the correct analytic continuation, exemplified (for \(\epsilon > 0\)) by
\[ \text{Li}_2(2 + i\epsilon) = \frac{\pi^2}{4} + i\pi \log 2, \]
and in general
\[ \text{Disc Li}_s(z) = 2\pi i \frac{\log^{s-1} z}{\Gamma(s)} , \]
with \(\Im (\text{Li})\) always being split equally across the cut—thus we know exactly the imaginary part of any \(\text{Li}_n(z)\) on the real ray \(z \in [1, \infty]\); said part is \((i/2)\text{Disc}\. This discontinuity relation is quite useful in checking of any software.

There are relations that allow analytic continuation, namely these (references [1], [2], but see analytic corrections of the classical work in [3]):
\[ \text{Li}_s(z) + \text{Li}_s(-z) = 2^{1-s} \text{Li}_s(z^2), \] (1.2)
true for all complex \(s, z\), and for \(n\) integer, and complex \(z\),
\[ \text{Li}_n(z) + (-1)^n \text{Li}_n(1/z) = -\frac{(2\pi i)^n}{n!} B_n \left( \frac{\log z}{2\pi i} \right) - 2\pi i \Theta(z) \frac{\log^{n-1} z}{(n-1)!} , \] (1.3)
where \(B_n\) is the standard Bernoulli polynomial and \(\Theta\) is a domain dependent step function: \(\Theta(z) := 1\), if \(\Im(z) < 0\) or \(z \in [1, \infty]\), else \(\Theta = 0\). That is, the final term in (1.3) is included when and only when \(z\) is in the lower open half-plane union the real cut \([1, \infty]\).
Another relation we shall use is an expansion for constrained values of \( \log z \), this time for integers \( n > 1 \),

\[
\text{Li}_n(z) = \sum_{m=0}^{\infty} \frac{\zeta(n-m)}{m!} \log^m z + \frac{\log^{n-1} z}{(n-1)!} (H_{n-1} - \log(-\log z)),
\]

(1.4)

valid for \( |\log z| < 2\pi \). Here, the \( \sum' \) notation means we avoid the singular \( \zeta(1) \) summand, and \( H_q := \sum_{k=1}^{q} 1/k \) with \( H_0 := 0 \) being the harmonic numbers.

The final relation we shall need for a comprehensive algorithm is, for any complex \( z \) but for \( n = 0, -1, -2, -3, \ldots \),

\[
\text{Li}_n(z) = (-n)!(-\log z)^{n-1} - \sum_{k=0}^{\infty} \frac{B_{k-n+1}}{k!(k-n+1)} \log^k z.
\]

(1.5)

The central idea of the algorithms to follow is to employ analytic relations to render \( |\log z| < 2\pi \), so that either (1.4) or (1.5) applies efficiently.

2. Explicit algorithms for complete analytic continuation.

Some instances of \( \text{Li}_n \) with integer \( n \) are elementary, as

\[
\text{Li}_n(1) = \zeta(n),
\]

(2.1)

\[
\text{Li}_n(-1) = -(1 - 2^{1-n}) \zeta(n),
\]

\[
\text{Li}_0(z) = \frac{z}{1-z}, \; z \neq 1,
\]

\[
\text{Li}_1(z) = -\log(1-z), \; z \neq 1,
\]

\[
\text{Li}_{-1}(z) = \frac{z}{(1-z)^2},
\]

and generally \( \text{Li}_n \) is a rational-polynomial function of \( z \) for \( n \leq 0 \) (however, the algorithm following simply provides a sufficient approximation to such representations without expanding out the requisite rational form).
Algorithm 2.1 \((\text{poly}(n, z))\): Computation of \(L_n(z)\) for any \(n \in \mathbb{Z}, z \in \mathbb{C}\). It is always assumed that \(-\pi < \arg z \leq \pi\), whence the analytic continuation with proper branch cut behavior is assured in the algorithm’s return value. This algorithm resolves \(L_i\) at the worst-case rate of about 1 precision bit per loop iteration.

0) For \(D\)-decimal-digit precision, choose summation limit \(L := \lceil D \log_2 10 \rceil\), where we define functions

\[
F_n^{(0)}(L, z) := \text{R.H.S. of (1.1) through summation limit } L;
\]

\[
F_n^{(1)}(L, z) := \text{R.H.S. of (1.4) through summation limit } L;
\]

\[
F_n^{(-1)}(L, z) := \text{R.H.S. of (1.5) through summation limit } L;
\]

\[
G_n(L, z) := \text{R.H.S. of (1.3)};
\]

1) if \(z = \pm 1\) or \(n = -1, 0, 1\) return result of (2.1);

2) if \(|z| \leq 1/2\) return \(F_n^{(0)}(L, z)\);

3) if \(|z| \geq 2\) return \(G_n(L, z) - (-1)^n F_n^{(0)}(L, \frac{1}{z})\);

4) (Here, we have \(|z| \in (1/2, 2)\), and \(n < -1\) or \(n > 1\)) return \(F_n^{(\text{sign}(n))}(L, z)\);

When analytic continuation is not necessary, say when \(z \in [-1, 1]\) is real and \(n > 1\), one may use the following, real-arithmetic algorithm:

Algorithm 2.2 \((\text{polyreal}(n, z))\): Computation of \(L_i(n)(z)\) for \(n > 1\) and real \(z \in [-1, 1]\). This algorithm resolves \(L_i\) at the worst-case rate of about 2 precision bits per loop iteration. Note that only real arithmetic is required.

0) For \(D\)-decimal-digit precision, choose summation limit \(L := \lceil D \log_4 10 \rceil\), where we define functions

\[
F_n^{(0)}(L, z) := \text{R.H.S. of (1.1) through summation limit } L;
\]

\[
F_n^{(1)}(L, z) := \text{R.H.S. of (1.4) through summation limit } L;
\]

1) function \(\text{polyreal}(n, z)\) {
    if \(z = 1\) return \(\zeta(n)\);
    if \(z = -1\) return \(-(1 - 2^{1-n})\zeta(n)\);
    if \(|z| < 1/4\) return \(F_n^{(0)}(L, z)\);
    if \(z < 0\) return \(2^{1-n}\text{polyreal}(n, z^2) - \text{polyreal}(n, -z)\);
    return \(F_n^{(1)}(L, z)\); // All arithmetic in (1.4) is real.
}

Note that the recursion (internal call on “if \(z < 0\)”) can only happen at most once, because both \(-z, z^2\) are positive on the conditional.
3. Enhancements and extensions

Though Algorithm 2.1 achieves about 1 precision bit per summand in any of the $F_n$ evaluations, somewhat more acceleration can be obtained by adjusting the interval endpoints of $(1/2, 2)$ to say $(r_1, r_2)$. Convergence of any part of Algorithm 2.1 is assured if both $|\log r_k| < \pi \sqrt{3}$.

One enhancement is to use some recursion on the quadratic relation (1.2), for said relation can improve the convergence rate in some cases. In fact, full recursion is certainly a possibility in the sense of

**Algorithm 3.1** (*polyrec(n, z)*): Recursive algorithm for $\text{Li}_n(z)$. We refer to the overall function of Algorithm 2.1 as poly, and define a new function based on analytic relation (1.2).

0) Choose a threshold $r_2 > 1$, for example $r_2 := 2$;
1) function $\text{polyrec}(n, z)$ {
   if($|z| < r_2)$ return $\text{poly}(n, z)$; // This can be refined just to invoke parts of poly.
   return $2^{n-1}(\text{polyrec}(\sqrt{z}) + \text{polyrec}(-\sqrt{z}))$;
}

The attractive simplicity of this recursion must be weighed against the number of calls required to reduce the effective $z$ parameter down to the size of $r_2$ via the nested square roots. Still, some partial recursion of this kind should accelerate step (4) of Algorithm 2.1 by reducing the log $z$ magnitude for the requisite summations.

Finally, it is possible to parallelize these algorithms to obtain $\text{Li}_n(z)$ on a set $\{z_1, z_2, \ldots \}$. Such parallelization is called for in experimental mathematics work, where say a numerical integral having polylogarithms in its integrand is to be resolved.
References

