Computing Stieltjes constants using complex integration

Fredrik Johansson

Computing Stieltjes constants using complex integration

Fredrik Johansson*

Abstract

The Stieltjes constants γ_n are the coefficients appearing in the Laurent series of the Riemann zeta function at $s = 1$. We give a simple and efficient method to compute a p-bit approximation of γ_n with rigorous error bounds. Starting from an integral representation due to Blagouchine, we shift the contour to eliminate cancellation. The integral is then evaluated numerically in ball arithmetic using the Petras algorithm, with the use of a Taylor expansion for bounds near the saddle point. This appears to be the first algorithm for Stieltjes constants with uniformly low complexity with respect to both n and p. An implementation is provided in the Arb library. We can, for example, compute γ_n to 1000 digits in a minute for any $n \leq 10^{100}$.

1 Introduction

The Stieltjes constants are the real numbers γ_n defined by

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

where $\zeta(s)$ is the Riemann zeta function. Also known as generalized Euler constants, they include the Euler-Mascheroni constant $\gamma_0 = \gamma \approx 0.577216$ as a special case.

The numbers γ_n with $n \leq 8$ were computed to nine decimal places by Jensen in 1887. Many authors have followed up on this work using an array of techniques. Fundamentally, any method to compute $\zeta(s)$ can be adapted to γ_n by taking derivatives; for example, as discussed by Liang and Todd [15], Jensen's calculations were based on the limit formula

$$\zeta(s) = \lim_{N \to \infty} \sum_{k=1}^{N} \frac{1}{k^s} - \frac{N^{1-s}}{1-s} \quad \left(\gamma_n = \lim_{N \to \infty} \left\{ \sum_{k=1}^{N} \frac{(\log(k))^n}{k} - \frac{(\log(N))^{n+1}}{n+1} \right\} \right),$$

while Gram in 1895 expressed γ_n in terms of derivatives of the Riemann ξ-function which he evaluated using integer zeta values. Liang and Todd themselves proposed computing γ_n via the Euler-Maclaurin summation formula for $\zeta(s)$, or as an alternative, via the application of Euler's series transformation to the alternating zeta function. Bohman and Fröberg [4] later refined the limit formula technique.

Keiper [12] proposed an algorithm based on the approximate functional equation for computing the Riemann ξ-function, which upon differentiation yields derivatives of $\xi(s)$ as integrals involving Jacobi theta functions. The Stieltjes constants are then recovered by a power series transformation. Keiper's algorithm is used for numerical evaluation of Stieltjes constants in Mathematica [18].

Kreminski [14] used a version of the limit formula combined with Newton-Cotes quadrature to estimate the resulting sums, and computed accurate values of γ_n up to

*LFANT – INRIA – IMB, Bordeaux, France (fredrik.johansson@gmail.com)

$n = 3200$. More recently, Johansson [7] combined the Euler-Maclaurin formula with fast power series arithmetic, proved rigorous error bounds for this method, and performed the most extensive computation of Stieltjes constants to date resulting in 10000-digit values of γ_n for all $n \leq 10^5$.

Even more recently, Adell and Lekuona [1] have used probability densities for binomial processes to obtain new rapidly convergent series for γ_n in terms of Bernoulli numbers.

The drawback of all the methods mentioned so far is that the complexity to compute γ_n is at least linear in n. In most cases, the complexity is actually at least quadratic in n since the formulas tend to have a high degree of cancellation necessitating use of $\Omega(n)$-digit arithmetic. For the same reason, the space complexity is also usually quadratic in n, at least in the most efficient forms of the algorithms.

The fast Euler-Maclaurin method [7] does allow computing the first n Stieltjes constants simultaneously to precision $p = \Theta(n)$ in $n^{2+o(1)}$ time, which is quasi-optimal. However, this is not ideal if we only need $p = O(1)$ or a single γ_n.[1]

This leads to the question of whether we can compute γ_n quickly for any n; ideally, in time depending only polynomially on $\log(n)$. If we assume that the precision p is fixed, then any asymptotic formula $\gamma_n \sim G(n)$ where $G(n)$ is an easily computed function should do the job. Various asymptotic estimates and bounds for the Stieltjes constants have been published, going back at least to Berndt [2], but the explicit computations by Kreminski and others showed that these estimates were far from precise. A breakthrough finally came in 2011 when Knessl and Coffey [13] obtained an asymptotic formula that not only precisely describes the growth rate of γ_n but also explains the local oscillations (and semi-regular sign changes). The Knessl-Coffey formula can be stated as

$$\gamma_n \sim \frac{B}{\sqrt{n}} e^{nA} \cos(an + b) \tag{1}$$

in terms of the slowly varying functions

$$A = \frac{1}{2}\log(u^2+v^2) - \frac{u}{u^2+v^2}, \quad B = \frac{2\sqrt{2\pi}\sqrt{u^2+v^2}}{[(u+1)^2+v^2]^{1/4}},$$

$$a = \operatorname{atan}\left(\frac{v}{u}\right) + \frac{v}{u^2+v^2}, \quad b = \operatorname{atan}\left(\frac{v}{u}\right) - \frac{1}{2}\left(\frac{v}{u+1}\right),$$

where v is the unique solution of $2\pi \exp(v \tan v) = n \cos(v)/v$ with $0 < v < \pi/2$, and $u = v\tan(v)$. In (1), the "$\sim$" symbol signifies asymptotic equality as long as the cosine factor is bounded away from zero. More recently, Fekih-Ahmed [6] has given an alternative asymptotic formula with similar accuracy to (1), and Paris [16] has extended (1) to an asymptotic series with higher order correction terms, permitting the determination of several digits of γ_n for moderately large n.

From a computational point of view, these formulas have three drawbacks. First, being asymptotic in nature, they only provide a fixed level of accuracy for a fixed n, so a different method must be used for small n and high precision p. Second, the high-order terms in Paris's expansion are quite complicated to compute. Third, explicit error bounds are not currently known.

[1] It is of course also interesting to consider the complexity of computing a single γ_n value to variable accuracy p. For $n \geq 1$, the complexity is $\Omega(p^2)$ with all known methods (although the fast Euler-Maclaurin method amortizes this to $p^{1+o(1)}$ per coefficient when computing $n = \Theta(p)$ values simultaneously). The exception is γ_0 which can be computed in time $p^{1+o(1)}$ by exploiting its role as a hypergeometric connection constant [5].

In this work, we give an algorithm that permits computing γ_n to p-bit accuracy efficiently for any combination of n and p. We also obtain rigorous error bounds, in the sense of ball arithmetic [19]; that is, we compute a provably correct enclosure $\gamma_n \in [m \pm r]$. We have implemented the algorithm in the Arb library [8] as the `acb_dirichlet_stieltjes` method.

The Knessl-Coffey, Fekih-Ahmed and Paris formulas were derived using the standard technique of applying saddle point analysis to a suitable contour integral. A natural approach to construct an algorithm with the desired properties is to take a similar integral representation and perform numerical integration instead of developing an asymptotic expansion symbolically. The integral representations used in the aforementioned works do not appear to be convenient for this purpose, since they involve nonsmooth functions (periodic Bernoulli polynomials) or require a summation over several integrals. A better starting point seems to be the Jensen-Franel formula (discussed by Blagouchine [3])

$$\gamma_n = \frac{1}{i} \int_0^\infty \frac{dz}{e^{2\pi z} - 1} \left[\frac{(\log(1 - iz))^n}{1 - iz} - \frac{(\log(1 + iz))^n}{1 + iz} \right], \quad n \geq 1 \tag{2}$$

which essentially is a differentiated form of the Abel-Plana summation formula applied to $\zeta(s)$. We will use the variant

$$\gamma_n = -\frac{\pi}{2(n+1)} \int_{-\infty}^\infty \frac{\left(\log\left(\frac{1}{2} + iz\right)\right)^{n+1}}{\cosh^2(\pi z)} dz \tag{3}$$

given in [3], which is more convenient since there is no removable singularity at the origin that needs to be analyzed.

The formula (2) has already used as the basis for a numerical implementation of Stieltjes constants in the mpmath library [10], but the algorithm as implemented in mpmath loses accuracy for large n (for example, $\gamma_{10^4} \approx -2.21 \cdot 10^{6883}$ but mpmath 1.0 computes $-1.25 \cdot 10^{6800}$). In this work, we use an approximate steepest descent contour together with validated integration in ball arithmetic to ensure that the computation is both accurate (indeed, fully rigorous) and efficient for large n.

2 Contour

We denote the integrand in (3) by

$$f(z) = \left(\log\left(\tfrac{1}{2} + iz\right)\right)^{n+1} \operatorname{sech}^2(\pi z). \tag{4}$$

Due to symmetry, we only need to evaluate the half of the integral $I_n = \int_0^\infty f(z) dz$, giving $\gamma_n = -\pi \operatorname{Re}[I_n]/(n+1)$. We approximate I_n by the truncated integral $\int_0^N f(z) dz$. If $N \geq 4$ and $N \geq n \geq 1$, then the tail $T_N = \int_N^\infty f(z) dz$ satisfies the bound

$$\begin{aligned}
|T_N| &\leq \int_N^\infty 4e^{-2\pi z} \left|\log\left(\tfrac{1}{2} + iz\right)\right|^{n+1} dz \\
&\leq 4e^{-2\pi N} \int_0^\infty e^{-2\pi z} \log^{n+1}(2N + 2z) dz \\
&\leq 4e^{-2\pi N} \log^{n+1}(2N) \int_0^\infty e^{-2\pi z} \left(1 + \frac{\log(2N + 2z) - \log(2N)}{\log(2N)}\right)^{n+1} dz \\
&\leq 4e^{-2\pi N} \log^{n+1}(2N) \int_0^\infty e^{-2\pi z} \left(1 + \frac{\log(1 + z/N)}{\log(2N)}\right)^{n+1} dz
\end{aligned}$$

$$\leq 4e^{-2\pi N}\log^{n+1}(2N)\int_0^\infty e^{-2\pi z}\left(1+\frac{z}{N\log(2N)}\right)^{n+1}dz$$
$$\leq 4e^{-2\pi N}\log^{n+1}(2N)\int_0^\infty \exp\left(-2\pi z+\frac{n+1}{2N}z\right)dz$$
$$\leq e^{-2\pi N}\log^{n+1}(2N).$$

We used $N \geq n \geq 1$ in the last line and $N \geq 4$ in the line prior.

We can select N by starting with $N = \max(n, 4)$ and repeatedly doubling N until $|T_N| \leq 2^{-p-20}$, say. This bound does not need to be tight since the integration algorithm, described later, discards negligible segments cheaply through bisection.

For small n, we can integrate $\int_0^N f(z)dz$ directly using a precision of about p bits. For large n, the integrand oscillates on the real line and a higher working precision must be used due to cancellation. The amount of cancellation can be calculated accurately by computing the maximum value of $|f(z)|$ on $[0, n]$ numerically and comparing this to the asymptotic formula (1). For example, we need about 30 extra bits for $n = 10^3$, 1740 bits when $n = 10^6$, and $4 \cdot 10^5$ bits when $n = 10^9$.

For n larger than about 10^3 where the extra precision would start to become a concern, we avoid the cancellation by shifting the main part of the integral to a horizontal line segment in the lower half plane. That is, we compute four segments

$$\int_0^N f(z)dz = \left[\int_0^M + \int_M^{M+Ci} + \int_{M+Ci}^{N+Ci} + \int_{N+Ci}^N\right] f(z)dz$$

where $C < 0$ is chosen to approximately minimize the peak magnitude of $f(t + Ci)$ on $M \leq t \leq N$. The left point $M > 0$ just serves to avoid the poles of the integrand on the imaginary axis; we can for instance take $M = 10$.

The point C is determined by $C = \mathrm{Im}(\omega)$ where ω is an approximation of the saddle point of $f(z)$ in the fourth quadrant. To this end, we write the integrand as

$$f(z) = \exp\left(g(z)\right) h(z) \tag{5}$$

where

$$g(z) = (n+1)\log\left(\log\left(\tfrac{1}{2}+iz\right)\right) - 2\pi z \tag{6}$$

and

$$h(z) = (1+\tanh(\pi z))^2. \tag{7}$$

The saddle point equation $g'(\omega) = 0$ is equivalent to

$$(n+1) + 2\pi i\left(\tfrac{1}{2}+i\omega\right)\log\left(\tfrac{1}{2}+i\omega\right) = 0$$

which can be solved in closed form using the Lambert W function, giving

$$\omega = \frac{i}{2}\left(1-\frac{2u}{W_0(u)}\right), \quad u = \frac{(n+1)i}{2\pi}.$$

Numerical tests confirm that the principal branch W_0 of the Lambert W function gives the correct saddle point and that there is essentially no cancellation with this contour. The path does not exactly pass through the saddle point of $f(z)$, but since $h(z)$ is exponentially close to a constant, the perturbation is negligible. The deviation between the straight-line path through the saddle point and the actual steepest descent contour also has negligible impact on the numerical stability.

We note that the complex Lambert W function can be computed rigorously using the methods described in [9]. However, it is not actually necessary to compute ω rigorously for this application since the integration follows a connected path and ball arithmetic will account for the actual cancellation; it is sufficient to use a floating-point approximation for ω with heuristic accuracy of about $\log_2(n)$ bits. For example, 53-bit machine arithmetic is sufficient up to about $n = 10^{15}$.

3 Integration and bounds near the saddle point

The main task of integrating $f(z)$ along one or four segments in the plane is not difficult in principle, since $f(z)$ is analytic (and non-oscillatory) in a neighborhood of each segment. Constructing a reliable and fast algorithm, in particular for extremely large n, does nevertheless require some attention to detail.

A first important observation is that the computations must be done with a working precision of about $p+\log_2(n)$ bits for p-bit accuracy, due to the sensitivity of the integrand. In other words, we lose about $\log_2(n)$ bits to the exponents of the floating-point numbers when evaluating exponentials.

To obtain rigorous error bounds and ensure rapid convergence with a minimum of manual analysis, we use the self-validating Petras algorithm [17] which was recently adapted for arbitrary-precision ball arithmetic and implemented in the Arb library [11]. The Petras algorithm combines Gauss-Legendre quadrature with adaptive bisection. Given a segment $[\alpha, \beta]$, the algorithm first evaluates the direct enclosure $(\beta - \alpha)f([\alpha, \beta])$ and uses this if the error is negligible (which in this application always occurs near the tail ends of the integral when $p \ll n$). Otherwise, it bounds the error of d-point quadrature $\int_\alpha^\beta f(z)dz \approx \sum_{k=1}^{d} w_k f(z_k)$ in terms of the magnitude $|f(z)|$ on a Bernstein ellipse E around $[\alpha, \beta]$: if $f(z)$ is analytic on E with $\max_{z\in E}|f(z)|\leq V$, the error is bounded by Vc/ρ^d where c and ρ only depend on E and $[\alpha, \beta]$. If $f(z)$ has poles or branch cuts on E or if the quadrature degree d determined by this bound would have to be larger than $O(p)$ to ensure a relative error smaller than 2^{-p}, the segment $[\alpha, \beta]$ is bisected and the same procedure is applied recursively.

The pointwise evaluations $w_k f(z_k)$ pose no problem: here we simply evaluate (4) in ball arithmetic as written. The remaining problem is to bound $f(z)$ on wide intervals representing z, which is needed both for the direct enclosures on subintervals $f([\alpha, \beta])$ and for the bounds on ellipses. We remark that the complex ball arithmetic in Arb actually uses rectangles with midpoint-radius real and imaginary parts rather than complex disks, so ellipses will always be represented by enclosing rectangles (with up to a factor $\sqrt{2}$ overestimation), but this detail is immaterial to the principle of the algorithm.

We can bound the integrand on wide ellipses (or enclosing rectangles) by evaluating (4) or (5)–(7) directly in interval or ball arithmetic. However, used alone, this results at best in $n^{1/2+o(1)}$ complexity as $n \to \infty$.[2] The explanation for this phenomenon is that $f(z)$ is a quotient of two functions $f_1(z) = (\log(\frac{1}{2} + iz))^{n+1}$, $f_2(z) = \cosh^2(\pi z)$ that individually vary rapidly near the saddle point, i.e.

$$\frac{f_1(z+\varepsilon)}{f_1(z)} \sim \frac{f_2(z+\varepsilon)}{f_2(z)} \sim e^{2\pi\varepsilon} \tag{8}$$

while f_1/f_2 is nearly constant. Direct evaluation fails to account for this correlation (which is an example of the dependency problem in interval arithmetic). Although $f(z)$ is

[2] In fact, the complexity becomes $n^{1+o(1)}$ when using ball arithmetic with a fixed precision for the radii (30 bits in Arb). The $n^{1/2+o(1)}$ estimate holds when the endpoints are tracked accurately.

nearly constant close to the saddle point, direct upper bounds for $|f(z)|$ are exponentially sensitive to the width of input intervals, and this forces the integration algorithm to bisect down to subsegments of width $O(1)$ around the saddle point. Since the peak of the integrand around the saddle point has an effective width of $O(n^{1/2})$, the integration algorithm has to bisect down to $O(n^{1/2})$ subsegments before converging.

To improve the bound and reduce the complexity, we use the standard trick of Taylor expanding with respect to a symbolic perturbation ε.

We use the decomposition (5)–(7). The function $h(z)$ is easy to bound: for example, if $\mathrm{Re}(z) \geq 1$, then $|h(z)| < 4.015$. We now turn to bounding $\exp(g(z))$. Given z contained in a complex ball with midpoint m and radius r, we seek to bound the absolute value of

$$\exp\left(g(m+\varepsilon)\right) = \exp\left(g(m)\right) \exp\left(g'(m)\varepsilon + \int_0^{\varepsilon} g''(m+t)(\varepsilon - t)dt\right) \tag{9}$$

for all $|\varepsilon| \leq r$. This gives us

$$|f(z)| < 4.015\,|\exp\left(g(m)\right)| \exp\left(|g'(m)|r + \tfrac{1}{2}Gr^2\right) \tag{10}$$

where $\max_{|u-m|\leq r} |g''(u)| \leq G$. To evaluate the bound (10), we compute $g(m)$ and $g'(m)$ explicitly, using

$$g'(m) = \frac{i(n+1)}{(1/2+im)\log(1/2+im)} - 2\pi.$$

The behavior near the saddle point is now captured precisely by the cancellation in $g'(m)$. At least $\log_2(n)$ bits of precision must be used to evaluate $\exp(g(m))$ (to ensure that the magnitude of the integrand near the peak is approximated accurately) and also to evaluate $g'(m)$ (to ensure that the remainder after the catastrophic cancellation is evaluated accurately). Finally, to compute G, we evaluate

$$g''(z) = \frac{(n+1)\left(1 + \frac{1}{\log(t)}\right)}{t^2 \log(t)}, \quad t = \frac{1}{2} + iz$$

directly over the complex ball representing z (as a minor optimization, we can compute lower bounds for $|t|$ and $|\log(t)|$). This completes the algorithm.

3.1 Final complexity

If the precision goal p is fixed (or grows sufficiently slowly compared to n), then we can argue heuristically that the bit complexity of computing γ_n to p-bit accuracy with this algorithm is $\log^{2+o(1)} n$. This estimate accounts for the bisection depth around the saddle point as well as the extra precision of $\log_2(n)$ bits necessary to manipulate exponents and evaluate the integrand accurately. The logarithmic complexity agrees well with the actual timings (presented in the next section).

We stop short of attempting a formal complexity analysis, which would require more detailed calculations and careful accounting for the accuracy of the enclosures in ball arithmetic as well as details about the integration algorithm. We have deliberately delegated as much work as possible to a general-purpose integration algorithm in order to minimize the analysis necessary for a complete implementation. However, in future work, it would be interesting to pursue such analysis not just for this specific problem, but more generally for evaluating classes of parametric integrals using the combination of saddle point analysis and numerical integration.

If we on the other hand fix n and consider varying p, then the asymptotic bit complexity is of course $p^{2+o(1)}$ since Gaussian quadrature uses $O(p)$ evaluations of the integrand on a fixed segment and $O(\log(p))$ segments are sufficient.

A minor technical point is that we cannot make any a priori statements about the relative error of γ_n since we do not have lower bounds for $|\gamma_n|$. However, it is reasonable to consider the accuracy goal $\frac{B}{\sqrt{n}} e^{nA} 2^{-p}$ with respect to the nonoscillatory envelope ignoring the cosine factor in (1). In our algorithm, the local oscillation corresponding to the cosine factor appears when extracting the imaginary part after performing the integration.

4 Benchmark results

Table 1 shows the time in seconds to evaluate γ_n at a precision p of 64 bits (about 18 digits), 333 bits (about 100 digits) and 3333 bits (just more than 1000 digits) on an Intel Core i5-4300U CPU. The implementation automatically chooses the internal working precision so that the ball computed for γ_n is accurate to about p bits.

Table 1: Time in seconds to compute γ_n to p-bit accuracy.

n	$p = 64$	$p = 333$	$p = 3333$
1	0.0011	0.0089	2.7
10	0.0020	0.032	6.6
10^2	0.0032	0.030	3.5
10^3	0.0064	0.10	7.5
10^4	0.0043	0.045	19.8
10^5	0.0043	0.026	27.8
10^6	0.0066	0.026	18.1
10^{10}	0.0087	0.031	32.6
10^{15}	0.014	0.061	7.0
10^{30}	0.087	0.22	16.7
10^{60}	0.26	0.86	30.9
10^{100}	0.76	1.5	57.2

As expected, the running time only depends weakly on n. The performance is also reasonable for large p. The timings fluctuate slightly rather than increasing monotonically with n; this is just an artifact of the adaptive integration algorithm.

We show the computed values of a few large Stieltjes constants. The following significands are correctly rounded to 100 digits (with at most 0.5 ulp error):

$\gamma_{10^5} \approx 1.99192730631254109565822724315685892052116597775331132587597552593617125927222717691432066619096522 5 \cdot 10^{83432}$

$\gamma_{10^{10}} \approx 7.58836212371310519482240337991254869217504103245097004705409333849242397478392791499204665451855077 9 \cdot 10^{12397849705}$

$\gamma_{10^{15}} \approx 1.84410172558473229070326955983513648856757465533155879218608594850254260862772177902307157373202222 1 \cdot 10^{1452992510427658}$

$\gamma_{10^{100}} \approx 3.18743141870239927999741646992711665139430991088384692250710626598304893415593755966828802263230609 5 \cdot 10^{e}$,

$e = 2346394292277254080949367838399091160903447689869837385205779111579215664052158234417125417543348369 4$

As a sanity check, γ_{10^5} agrees with the previous record Euler-Maclaurin computation [7], and the first 3-4 digits in all cases agree with the Knessl-Coffey formula (1).

A quick comparison with the existing Euler-Maclaurin implementation in Arb shows that computing a single γ_n value using integration is faster than computing $\gamma_0, \ldots, \gamma_n$ simultaneously using the Euler-Maclaurin formula roughly when $n > \max(100, p/2)$. (In fact, the method `acb_dirichlet_stieltjes` automatically switches to the Euler-Maclaurin formula when this is faster, but we disabled this feature for the timings so that only numerical integration was tested.) At this point, we do not have an optimized implementation of the Euler-Maclaurin algorithm for a single γ_n value, which would be useful for n from about 10 to 10^3.

A few possible optimizations of the integration algorithm are worth pointing out:

- For evaluating a range of values γ_n with $A \le n \le B$ simultaneously, one could perform a single vector-valued integration and recycle the evaluations of $\log(1/2+iz)$ and $\operatorname{sech}^2(\pi z)$. In the range up to about $B = 10^5$, it would be interesting to compare this approach to simultaneous evaluation with the Euler-Maclaurin formula.
- Parallelization of a single γ_n computation is trivial by splitting the d evaluations in Gaussian quadrature into d/r evaluations for each of the r worker threads. This should give a nearly linear speedup when $r \ll d$. We did not try to implement this technique for the present computations since only a dual-core machine was available.
- The adaptive integration strategy in Arb can probably be improved, which should give a small constant factor speedup. The working precision could also likely be reduced by a preliminary rescaling near the saddle point.

5 Discussion

This study was done for two purposes: first, to develop working code for Stieltjes constants as part of the collection of rigorous special function routines in the Arb library, and second, to test the integration algorithm [17, 11] for a family of integrals involving large parameters. We do not have a concrete application in mind for the code, but the Stieltjes constants are potentially useful in various types of analytic computations involving the Riemann zeta function, and large-n evaluation can be useful for testing the accuracy of asymptotic formulas for Stieltjes constants and related quantities.

The technique of evaluating parametric integrals by integrating along a steepest descent contour is, of course, well established in the literature on computational methods for special functions. The main contribution of this work is the humble observation that the particular integral (3) with a simple saddle point analysis leads to a complete and efficient algorithm for Stieltjes constants. The heavy lifting can be done by the integration algorithm, requiring only an elementary pen-and-paper analysis of the integrand. The same technique should be effective for many other number sequences and special functions given by similar integral representations.

An immediate continuation of this work would be to compute the generalized Stieltjes constants $\gamma_n(a)$ appearing in the Laurent expansion of the Hurwitz zeta function $\zeta(s,a) = \sum_{k=0}^{\infty}(a+k)^{-s}$ at $s = 1$, for instance via the formula

$$\gamma_n(a) = \left(\frac{1}{2a} - \frac{\log(a)}{n+1}\right)\log^n(a) - i\int_0^\infty \frac{dz}{e^{2\pi z}-1}\left[\frac{\log^n(a-iz)}{a-iz} - \frac{\log^n(a+iz)}{a+iz}\right].$$

We would need some more analysis to remove the singularity at $z = 0$, and further analysis would be needed to optimize the algorithm as the parameter a varies.

References

[1] J. Adell and A. Lekuona. Fast computation of the Stieltjes constants. *Mathematics of Computation*, 86(307):2479–2492, 2017.

[2] B. C. Berndt. On the Hurwitz zeta-function. *The Rocky Mountain Journal of Mathematics*, 2(1):151–157, 1972.

[3] I. V. Blagouchine. A theorem for the closed-form evaluation of the first generalized Stieltjes constant at rational arguments and some related summations. *Journal of Number Theory*, 148:537–592, 2015.

[4] J. Bohman and C. E. Fröberg. The Stieltjes function - definition and properties. *Mathematics of Computation*, 51(183):281–289, 1988.

[5] R. P. Brent and E. M. McMillan. Some new algorithms for high-precision computation of Euler's constant. *Mathematics of Computation*, pages 305–312, 1980.

[6] L. Fekih-Ahmed. A new effective asymptotic formula for the Stieltjes constants. *arXiv preprint arXiv:1407.5567*, 2014.

[7] F. Johansson. Rigorous high-precision computation of the Hurwitz zeta function and its derivatives. *Numerical Algorithms*, 69:253–270, 2015.

[8] F. Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66:1281–1292, 2017.

[9] F. Johansson. Computing the Lambert W function in arbitrary-precision complex interval arithmetic. *arXiv preprint arXiv:1705.03266*, 2017.

[10] F. Johansson. mpmath: a Python library for arbitrary-precision floating-point arithmetic, 2017. Version 1.0.

[11] F. Johansson. Numerical integration in arbitrary-precision ball arithmetic. *arXiv preprint arXiv:1802.07942*, 2018.

[12] J. B. Keiper. Power series expansions of Riemann's ξ function. *Mathematics of Computation*, 58(198):765–773, 1992.

[13] C. Knessl and M. Coffey. An effective asymptotic formula for the Stieltjes constants. *Mathematics of Computation*, 80(273):379–386, 2011.

[14] R. Kreminski. Newton-Cotes integration for approximating Stieltjes (generalized Euler) constants. *Mathematics of Computation*, 72(243):1379–1397, 2003.

[15] J. J. Y. Liang and J. Todd. The Stieltjes constants. *Journal of Research of the National Bureau of Standards*, 76:161–178, 1972.

[16] R. B. Paris. An asymptotic expansion for the Stieltjes constants. *arXiv preprint arXiv:1508.03948*, 2015.

[17] K. Petras. Self-validating integration and approximation of piecewise analytic functions. *Journal of Computational and Applied Mathematics*, 145(2):345–359, 2002.

[18] Wolfram Research. Some notes on internal implementation. Wolfram Language & System Documentation Center, 2018. https://reference.wolfram.com/language/tutorial/SomeNotesOnInternalImplementation.html.

[19] J. van der Hoeven. Ball arithmetic. Technical report, HAL, 2009. hal-00432152.