Journal de Théorie des Nombres
de Bordeaux 00 (XXXX), 000–000

Computing $L_E'''(1)$ to high precision

par Mark Watkins

Abstract. We describe how to calculate the third central derivative $L_E'''(1)$ of an elliptic curve (or modular form) L-function to large precision, and carry this out to either 10025 or 1025 digits for a specific selection of 12 rank 5 elliptic curves, showing that the third central derivative is zero to the asserted precision in each case via an implementation using the Arb package of Johansson. Our target precisions are sufficiently low that asymptotically superior methods (of quasi-quadratic complexity) do not yet avail, and we found that local power series (which are quasi-cubic) performed much better in practice.

The computations for these 12 specific rank 5 curves were used in a companion paper for the proof of (e.g.) the lower bound

$$\sqrt{D}L_\chi(1) = \pi h_K \geq \min\bigl(10^{10000}\log D, (\log D)^3/10^8\bigr)$$

for imaginary quadratic fields $\mathbf{Q}(\sqrt{-D})$ with $D \geq 4\pi^2\exp(10^7)$ prime. There is also a version for composite D (which includes a product over $p|D$), and analogously for real quadratic fields.

Résumé. $\backslash tofrench${We describe how to calculate the third central derivative $L_E'''(1)$ of an elliptic curve (or modular form) L-function to large precision, and carry this out to either 10025 or 1025 digits for a specific selection of 12 rank 5 elliptic curves, showing that the third central derivative is zero to the asserted precision in each case via an implementation using the Arb package of Johansson. Our target precisions are sufficiently low that asymptotically superior methods (of quasi-quadratic complexity) do not yet avail, and we found that local power series (which are quasi-cubic) performed much better in practice.

The computations for these 12 specific rank 5 curves were used in a companion paper for the proof of (e.g.) the lower bound

$$\sqrt{D}L_\chi(1) = \pi h_K \geq \min\bigl(10^{10000}\log D, (\log D)^3/10^8\bigr)$$

for imaginary quadratic fields $\mathbf{Q}(\sqrt{-D})$ with $D \geq 4\pi^2\exp(10^7)$ prime. There is also a version for composite D (which includes a product over $p|D$), and analogously for real quadratic fields.}

1. Introduction and purpose

Recently we showed in [15] an improved (effective) lower bound for $L_\chi(1)$ based upon relating such an L-value to the precision to which we could

2010 *Mathematics Subject Classification.* Primary 11Y35; Secondary 11Y16, 11G40.
Mots-clefs. elliptic curve L-functions, L-function computations, central values of L-functions.

compute various third central derivatives of specific elliptic curves of rank 5. We list our 12 such curves E of interest in Table 1, with each conductor N_E, and the precision d to which we now verify that $|\Lambda_E'''(1)| \le 10^{-d}$ (and an attendant timing, for which see §1.5.1). Here the $E \otimes t$ notation indicates the quadratic twist of E by t.

curve label	conductor	set	d	time
$[0,0,1,-79,342]$	19047851	$\mathcal{E}_1^-$	10029	4.6 days
$[0,0,1,-169,930]$	64921931	$\mathcal{E}_1^-$	10029	7.7 days
$[0,1,1,-30,390]$	67445803	$\mathcal{E}_1^-$	10029	7.9 days
$[1,0,0,-22,219]$	20384311	$\mathcal{E}_1^+$	10029	4.6 days
$[0,1,1,-100,110]$	55726757	$\mathcal{E}_1^+$	10029	7.2 days
$[0,0,1,-139,732]$	59754491	$\mathcal{E}_1^+$	10029	7.5 days
$[0,-1,1,0,0] \otimes -25351367$	$11 \cdot 25351367^2$	$\mathcal{E}_2^-$	1033	21 days
$[1,-1,1,-1,0] \otimes -19502039$	$17 \cdot 19502039^2$	$\mathcal{E}_2^-$	1033	21 days
$[0,1,1,-9,-15] \otimes -16763912$	$19 \cdot 16763912^2$	$\mathcal{E}_2^-$	1033	12 days
$[0,1,1,-7,5] \otimes 6350941$	$91 \cdot 6350941^2$	$\mathcal{E}_2^+$	1033	16 days
$0,1,1,-10,10] \otimes 5467960$	$123 \cdot 5467960^2$	$\mathcal{E}_2^+$	1033	11 days
$0,1,1,-27,-55] \otimes 3217789$	$209 \cdot 3217789^2$	$\mathcal{E}_2^+$	1033	12 days

Table 1. Data about our 12 rank 5 curves

The minimal twists for $\mathcal{E}_2^-$ are in the isogeny classes 11a, 17a, 19a, while for $\mathcal{E}_2^+$ they are in 91b, 123a, 209a.

1.1. More precisely, writing $\Lambda_E(s) = (\sqrt{N_E}/2\pi)^s\Gamma(s)L_E(s)$ for the completed L-function of E, in [15] we showed the following result, where $\mathbf{Q}(\sqrt{\Delta})$ is the quadratic field of fundamental discriminant Δ with $D = |\Delta|$, and χ its associated quadratic character.

Theorem 1.1.1. *Suppose that* $|\Lambda_E'''(1)| \le 10^{-1025} \cdot (2\pi/\sqrt{N_E})$ *for the six elliptic curves in Table 1 for* $\mathcal{E}_2^\pm$*. Then for* $D \ge 4\pi^2 \exp(10^7)$ *we have*

$$\sqrt{D}L_\chi(1) \ge \min\bigl(10^{1000}\log D, (\log D)^3/10^{13}\bigr) \cdot \prod_{p|D}\Bigl(1 - \frac{\lfloor 2\sqrt{p}\rfloor}{p+1}\Bigr).$$

We have similar theorems for D that are coprime to the conductors of various rank 5 elliptic curves; here the conductors are smaller, and we extended our calculations to 10000 digits. As examples, we have the following.

Theorem 1.1.2. *Suppose* $|\Lambda_E'''(1)| \le 10^{-10025} \cdot (2\pi/\sqrt{N_E})$ *for the 3 elliptic curves in Table 1 for* $\mathcal{E}_1^-$*. When* $\gcd(D, 19047851 \cdot 64921931 \cdot 67445803) = 1$ *and* $D \ge 4\pi^2\exp(10^7)$*, for* $\Delta < 0$ *we have*

$$\sqrt{D}L_\chi(1) \ge \min\bigl(10^{10000}\log D, (\log D)^3/10^8\bigr) \cdot \prod_{p|D}\Bigl(1 - \frac{\lfloor 2\sqrt{p}\rfloor}{p+1}\Bigr).$$

Suppose that $|\Lambda_E'''(1)| \le 10^{-10025} \cdot (2\pi/\sqrt{N_E})$ for the 3 elliptic curves in Table 1 for $\mathcal{E}_1^+$. Then for $\gcd(D, 3089 \cdot 6599 \cdot 647 \cdot 86131 \cdot 409 \cdot 146099) = 1$ *and $D \ge 4\pi^2 \exp(10^7)$, for $\Delta > 0$ we have*

$$\sqrt{D} L_\chi(1) \ge \min\bigl(10^{10000} \log D, (\log D)^3/10^8\bigr) \cdot \prod_{p|D} \Bigl(1 - \frac{\lfloor 2\sqrt{p} \rfloor}{p+1}\Bigr).$$

Rather than notating with E, we shall instead refer to the associated weight 2 modular newform f, which exists by a theorem of Wiles [16].

Herein we give the details concerning how to compute the desired central Λ-derivatives to the indicated precision. We describe two different methods to compute $\Lambda_f'''(1)$ to high precision: multi-point evaluation of a polynomial that approximates the modular form, in conjunction with Poisson summation; and a more standard/direct method that weights the modular form coefficients by a suitable Mellin transform. Both theoretically take time quasi-quadratic[1] in the precision asymptotically – however, we find that neither is particularly fast in our ranges of 1000 or 10000 digits.

Instead, in practice we use a variant of the second method involving computations by local power series (using the recurrence relation for derivatives that is satisfied by the weighting function), which is much faster in practice.

1.2. In the first method, suggested to us by A. R. Booker, we write $\Lambda_f(s)$ in terms of an integral $\int_0^\infty y^s f(iy)\, \partial y/y$. We then apply Poisson summation with a sufficiently large stride T so that $\sum_n \Lambda_f'''(1 + 2\pi i n T)$ has only $n = 0$ contributing non-negligibly, while on the other side we have a sum over f evaluated at various points, where again one can restrict the number of such points by exponential decay, and moreover we can adequately approximate the modular form itself by a polynomial of suitable length. With a desired absolute precision of 10^{-d}, one ends up with a polynomial of length approximately $(\log 10^d)\sqrt{N_f}/2\pi$ to be evaluated at $\log(\sqrt{N_f} \log 10^d) \cdot (\log 10^d)/\pi^2$ points. While the resulting calculation indeed theoretically takes time quasi-quadratic in d via multi-point evaluation methods, an initial experiment (by Johansson, using Arb) found that in the range of 1000 digits the gains were rather small (less than 10%) over a simplistic quasi-cubic implementation, and that for each of the six curves in $\mathcal{E}_2^\pm$ it would take about 10 core-years to compute the third central derivative to 1000 digits of precision. We still gives some details of this method below, though we ultimately were unable to make much use of it in practice.

1.3. The second method ultimately boils down to computing a generalized integro-exponential function G_3 in a sum $\sum_n G_3(2\pi n/\sqrt{N_f}) c_f(n)/n$ over the Fourier coefficients $c_f(n)$, and this G_3 can be calculated in various ways via the theory of hypergeometric functions. The desired calculation

[1]This simply means that the time is $\ll_\epsilon d^{2+\epsilon}$ for all $\epsilon > 0$ as the digit precision d tends to ∞.

does not quite seem to follow "out of the box", as it might for the comparable computation of the first central derivative (when we would have the exponential integral rather than G_3).

1.3.1. For small n we can compute the G_3-function by an expansion about 0 as already noted by Buhler, Gross, and Zagier [1]. For large n, the theory of asymptotic expansions is not well-illuminated in the literature, but we can adapt Olver's error bounds for the confluent hypergeometric function of the second kind to our setting.

1.3.2. However, it turns out in any case that for n that are not too small it is much superior to compute many G_3-values *en masse* by employing local power series about a thin set of demarcation points, at which one can readily compute a large number of derivatives using the recursion formula for the derivatives of G_3 (which comes from the functional equation for $\Lambda_f(s)$). This turns out to be faster by a factor of 10 or more in practice.

1.4. It may thus seem that we spend an inordinate amount of effort herein describing methods that we do not use in the end; this is a valid criticism, but it still seems useful to have a proper analysis of the various schema, particularly as we do not use the asymptotically best method in practice.

Moreover, while it may be "well-known" to experts, I am not aware of any theoretical (let alone practical) analysis of the complexity of computing L-function derivatives to high precision that notes it is quasi-quadratic.

Our contribution thus has both theoretical and practical aspects.

1.5. Notation and terminology. We use ∂ in integrals instead of "d". We write $\Lambda_f(s) = \Gamma(s)(\sqrt{N_f}/2\pi)^s L_f(s)$ for the completed L-function of a modular newform of level N_f, where the modular form has the Fourier expansion $f(z) = \sum_n c_f(n)e^{2\pi i n z}$ and its L-function is $L_f(s) = \sum_n c_f(n)/n^s$. We use the Θ-notation for an error term with an implicit constant of 1.

We write $(n)_k^{\uparrow} = n(n+1)\cdots(n+k-1)$ for the rising factorial, so $(1)_k^{\uparrow} = k!$. Our notation for ${}_pF_q$ hypergeometric functions is a bit lazy, e.g. we will simply write ${}_1F_1(a, b, z)$ and ${}_2F_0(a, b, z)$, without encapsulating the indices according to the subscripts.

1.5.1. Our calculations used a desktop 3.6GHz i7-7700 and a cluster-based 3.1GHz E5-2687W. The former is about 60% faster[2] for our code, and we have rescaled the timings from the latter to match this. The numbers given in Table 1 are thus scaled for one core on the i7-7700. We mainly used the desktop for parameter tuning and the first curve in $\mathcal{E}_1^-$, and switched to the cluster for the rest.

[2]Around half of this is from the GMP compilation being faster, e.g. using the `mulx` instruction.

2. Methods to compute $\Lambda_f'''(1)$

We describe some assorted methods to compute the third central derivative of a (completed) modular form L-function to high precision. We then indicate which we found more satisfactory in practice.

For our computations, we used the rigourous arbitrary-precision package Arb [6] of Fredrik Johansson. This uses FLINT, MPFR, and GMP for its underlying arithmetic. One aspect of Arb is that it computes to relative precision, whereas for our computations we are typically more interested in absolute precision. For instance, if we want to compute a special function to absolute precision 10^{-1050} and the functional value is itself around 10^{-300}, we need only request a relative precision of about 750 digits from Arb.

2.1. We recall that $\Lambda_f^{(w)}(1)$ can be approximated by a method detailed by Buhler, Gross, and Zagier [1, §4] (see also Cremona [2, §2.13]). For integral $w \geq 0$ we have

$$\frac{\Lambda_f^{(w)}(1)}{w!} = [1+\epsilon_f(-1)^w]\cdot\frac{\sqrt{N_f}}{2\pi}\sum_{n=1}^{\infty}\frac{c_f(n)}{n}G_w\Big(\frac{2\pi n}{\sqrt{N_f}}\Big) \tag{1}$$

where $G_0(x)=e^{-x}$ and for $w \geq 1$ we have

$$G_w(x) = \frac{1}{(w-1)!}\int_1^\infty e^{-xy}(\log y)^{w-1}\frac{\partial y}{y},$$

which satisfies $G_w'(x) = -G_{w-1}(x)/x$. For $w=1$ this is simply a familiar exponential integral. Our main interest is for $w=3$. We shall assume throughout that f has odd parity ($\epsilon_f=-1$).

It is critical that $G_w(x) \sim e^{-x}/x^w$ decays exponentially as $x\to\infty$, implying we can calculate to d digits using approximately $(\sqrt{N_f}/2\pi)\cdot\log 10^d$ terms of the series,[3] so that the number of terms needed is linear in d.

However, the prototypical asymptotic expansion for G_w is not readily in a form suitable for fast computation (indeed, as Dokchitser [3, §4, Remark 5.4] notes, it is not known in full generality that a continued fraction expansion about ∞ even converges). For instance, we were unable to locate any specific statement in the L-function literature that says computing central derivatives should be quasi-quadratic in the precision, as follows once

[3]The $c_f(p)$ can be computed in time polynomial in $\log p$ via Schoof's algorithm [14], though in practice using Gelfond's method (dubbed "baby step, giant step" by Shanks) takes roughly $p^{1/4}$ time and suffices for our range. Most of the time goes into computing G_3 in any event, though with $\mathcal{E}_2^\pm$ it could not be said to be overly dominant. For instance, computing $c_f(p)$ for $p \leq 17\cdot 2^{30}$ took around 3.5 days for the last curve in Table 1, thus about 25% of the total time (and for the curves in $\mathcal{E}_2^\pm$ with an even twisting factor the split nears 50-50). Admittedly, perhaps because we erroneously thought that its time would be negligible, we did not seek out the fastest code for $c_f(p)$-computation, but simply used a reasonably fast implementation that readily meshed with the rest of our codebase.

we note (below) that G_w can be computed in quasi-linear time. Moreover, even with a method that is asymptotically good, there is no assurance how viable it will be in practical ranges of computation.

2.2. As noted to us by A. R. Booker, a somewhat different method to compute $\Lambda_f'''(1)$ is to exploit the equi-spacedness of the evaluation points of G_3 and use a multi-point polynomial evaluation scheme. The complexity analysis is nontrivial due to precision issues, and appears somewhat obscurely in work of Kirrinnis [8] (see also [9]).

This gives a theoretical quasi-quadratic complexity for computing the third central derivative, but preliminary tests by Johansson indicated that in the 1000-digit range (applicable for 6 of our curves) any speedup from divide-and-conquer methods would be essentially offset by poor numerical stability (necessitating higher intermediate precisions). We didn't pursue the matter much after this, but still outline the idea for completeness.

2.2.1. We first note that for $\sigma > 3/2$ we have

$$\begin{aligned}\int_0^\infty f(iy)y^s\,\frac{\partial y}{y} &= \int_0^\infty \sum_{n=1}^\infty c_f(n)e^{-2\pi n y}y^s\,\frac{\partial y}{y} = \sum_{n=1}^\infty c_f(n)\int_0^\infty e^{-t}\Bigl(\frac{t}{2\pi n}\Bigr)^s\frac{\partial t}{t}\\ &= \frac{\Gamma(s)}{(2\pi)^s}\sum_{n=1}^\infty \frac{c_f(n)}{n^s} = \frac{\Gamma(s)}{(2\pi)^s}L_f(s) = \frac{\Lambda_f(s)}{(\sqrt{N_f})^s}.\end{aligned}$$

This integral is everywhere convergent since f is a cusp form, and thus gives an analytic continuation of $\Lambda_f(s)$.

We then change variables with $y = e^u/\sqrt{N_f}$ and $s = 1+it$ to get

$$\Lambda_f(1+it) = \int_{-\infty}^\infty f\Bigl(\frac{ie^u}{\sqrt{N_f}}\Bigr)\cdot e^{u+itu}\,\partial u,$$

and differentiating with respect to t gives

$$\Lambda_f^{(w)}(1+it) = \int_{-\infty}^\infty f\Bigl(\frac{ie^u}{\sqrt{N_f}}\Bigr)\cdot u^w e^{u+itu}\,\partial u,$$

We then take $H(x) = \Lambda_f^{(w)}(1+2\pi i x)$ so that $H(x) = \int_{-\infty}^\infty \hat{H}(u)e^{2\pi i x u}\partial u$ with $\hat{H}(u) = f(ie^u/\sqrt{N_f})u^w e^u$. We then apply the Poisson summation formula in the form $\sum_n H(nT) = \sum_m \hat{H}(m/T)/T$ (for $T>0$) to get

$$\sum_{n=-\infty}^\infty \Lambda_f^{(w)}(1+2\pi i n T) = \frac{1}{T^{w+1}}\sum_{m=-\infty}^\infty m^w e^{m/T} f\Bigl(\frac{ie^{m/T}}{\sqrt{N_f}}\Bigr). \qquad (2)$$

2.2.2. We then take T large enough to ensure the $n \neq 0$ terms contribute negligibly due to the vertical decay of Λ. To reach d digits of precision, by Stirling's approximation this will have approximately $T \sim (\log 10^d)/\pi^2$. Similarly the m-sum is curtailed at length $\sim T\log(T\sqrt{N_f})$, and the modular form is suitably approximated by a polynomial of length $\sim (\pi/2)T\sqrt{N_f}$.

To make these explicit, we have the following Lemmata.

Lemma 2.2.3. *In $|\sigma - 1| \leq 1$ we have*

$$\big|\Lambda_f(s)\big| \leq |\Gamma(2+it)|\Big(\frac{\sqrt{N_f}}{2\pi}\Big)^2 \zeta(3/2)^2.$$

Proof. Follows from bounding $L_f(2+it)$ by $\zeta(3/2)^2$ and using convexity. □

Lemma 2.2.4. *For $t \geq 100$ we have*

$$\big|\Lambda_f'''(1+it)\big| \leq 3! \cdot \big|\Gamma\big(2+i(t-1)\big)\big|\Big(\frac{\sqrt{N_f}}{2\pi}\Big)^2 \zeta(3/2)^2 \leq 13N_f t^{3/2}e^{-\pi t/2}.$$

Proof. We draw a circle of radius 1 about $1+it$ and use the previous Lemma in conjunction with Cauchy's bound on derivatives. We then use Stirling's approximation for the second part of the result. □

Lemma 2.2.5. *Suppose d is given and $T \geq 100/2\pi$ is chosen so that*

$$\pi^2 T \geq \log 10^d + \log\big(410 N_f T^{3/2}\big). \tag{3}$$

Then

$$\sum_{n=-\infty}^{\infty} \Lambda_f'''(1+2\pi i n T) = \Lambda_f'''(1) + \Theta\big(10^{-d}\big).$$

Proof. By the previous Lemma we have

$$2\sum_{n=1}^{\infty}\big|\Lambda_f'''(1+2\pi i n T)\big| \leq 26N_f\sum_{n=1}^{\infty}(2\pi n T)^{3/2}e^{-\pi^2 n T} \leq 410 N_f T^{3/2} e^{-\pi^2 T}.$$

The given condition on T now implies the desired bound. □

2.2.6. We then turn to computation of the other side of (2), namely.

$$\frac{1}{T^{w+1}}\sum_{m=-\infty}^{\infty} m^w e^{m/T} f\Big(\frac{ie^{m/T}}{\sqrt{N_f}}\Big).$$

We can initially note that $\mathrm{Im}(z)f(z)$ is automorphic for $\Gamma_0(N_f)$, and since f has odd parity it is anti-invariant under the involution $z \to -1/(N_f z)$ of Atkin and Lehner. In particular this implies $e^{m/T}f(ie^{m/T}/\sqrt{N_f})$ is anti-invariant under $m \to -m$. When w is odd we can thus pair m and $-m$.

We again enumerate some simple Lemmata.

Lemma 2.2.7. *For $y \geq 1$ we have $\big|f(iy)\big| \leq 1.005e^{-2\pi y}$.*

Indeed, from $|c_f(n)| \leq n$ we have

$$\big|f(iy)\big| = \Bigl|\sum_{n=1}^{\infty} c_f(n)e^{-2\pi ny}\Bigr| \leq e^{-2\pi ny} + \sum_{n=2}^{\infty} ne^{-2\pi ny} \leq 1.005e^{-2\pi ny}.$$

We write $C = \sqrt{N_f}/2\pi$ as a shorthand.

Lemma 2.2.8. *Suppose d is given and $w = 3$, while κ satisfies $e^\kappa/C \geq 10$ and*

$$2.28C\kappa^w \exp(-e^\kappa/C) \leq T10^{-d}. \tag{4}$$

Then

$$\sum_{|m|\geq\kappa T+2} \frac{m^w}{T^{w+1}} e^{m/T} \Bigl| f\Bigl(\frac{ie^{m/T}}{\sqrt{N_f}}\Bigr)\Bigr| \leq 10^{-d}.$$

Proof. From the previous Lemma we have that the m-sum is bounded as

$$\leq \frac{2.01}{T} \sum_{m\geq\kappa T+2} \frac{m^w}{T^w} e^{m/T} \exp\Bigl(-\frac{2\pi e^{-m/T}}{\sqrt{N_f}}\Bigr).$$

Comparing to an integral with $u = m/T$ gives

$$\leq \frac{2.01}{T}\int_\kappa^\infty u^w e^u \exp(-2\pi e^u/\sqrt{N_f})\,\partial u \leq \frac{2.01C}{T}\int_{e^\kappa/C}^\infty (\log vC)^w e^{-v}\,\partial v,$$

When $\xi \geq 10$ and $b \leq 3$ we have $\int_\xi^\infty (\log v)^b e^{-v}\,\partial v \leq 1.13(\log\xi)e^{-\xi}$ so that this is

$$\leq \frac{2.28C}{T}\kappa^w \exp(-e^\kappa/C) \leq 10^{-d}.$$

Note that to first order we have $\kappa \sim \log(Cd\log 10)$. □

2.2.9. The above serves to curtail the m-sum, so that under the conditions on T and κ we have

$$\Lambda_f'''(1) = 2\sum_{m=1}^{\lfloor\kappa T+2\rfloor} \frac{m^3}{T^4} e^{m/T} f\Bigl(\frac{ie^{m/T}}{\sqrt{N_f}}\Bigr) + \Theta(2\cdot 10^{-d}), \tag{5}$$

and we now codify how to approximate f by a polynomial of suitable length.

We again use $|c_f(n)| \leq n$, and for $y \geq 1/\sqrt{N_f}$ and $\alpha \geq 2$ (say) we have

$$f(iy) = \sum_{n=1}^{\lfloor\alpha\sqrt{N_f}+2\rfloor} c_f(n)e^{-2\pi ny} + \Theta\Bigl(\int_{\alpha\sqrt{N_f}} ue^{-2\pi uy}\,\partial u\Bigr).$$

The error term is then estimated as

$$\leq \int_{\alpha/y} ue^{-2\pi uy}\,\partial u \leq \frac{\alpha}{2\pi}\frac{e^{-2\pi\alpha}}{y^2} \leq \frac{\alpha}{2\pi}e^{-2\pi\alpha}N_f.$$

We then take

$$\frac{\alpha}{2\pi}e^{-2\pi\alpha} \le \frac{1}{\kappa^4 e^\kappa N_f}10^{-d} \tag{6}$$

so that $\alpha \sim (\log 10^d)/2\pi$. With $y = e^{m/T}/\sqrt{N_f}$ we approximate f in (5).

Lemma 2.2.10. *Suppose that d and f are given, while T satisfies (3) and κ satisfies (4) and α satisfies (6). Then*

$$\Lambda_f'''(1) = 2\sum_{m=1}^{\lfloor \kappa T+2\rfloor}\frac{m^3}{T^4}e^{m/T}\sum_{n=1}^{\lfloor \alpha\sqrt{N_f}+2\rfloor}c_f(n)\exp\Bigl(-2\pi n\frac{e^{m/T}}{\sqrt{N_f}}\Bigr)+\Theta\bigl(3\cdot 10^{-d}\bigr).$$

Proof. The above choice (6) of α yields an error from replacing f by a finite sum in (5) of

$$\Theta\Bigl(2\sum_{m=1}^{\lfloor \kappa T+2\rfloor}\frac{m^3}{T^4}e^{m/T}\frac{1}{\kappa^4 e^\kappa}10^{-d}\Bigr)$$

Summing over m then gives the result. □

Writing $M = \lfloor \kappa T + 2\rfloor$ and $J = \lfloor \alpha\sqrt{N_f}+2\rfloor$, we then have

$$\Lambda_f'''(1) = 2\sum_{m=1}^{M}\frac{m^3}{T^4}e^{m/T}\sum_{n=1}^{J}c_f(n)\exp\Bigl(-2\pi\frac{e^{m/T}}{\sqrt{N_f}}\Bigr)^n+\Theta\bigl(3\cdot 10^{-d}\bigr),$$

so are evaluating a polynomial of length roughly $\alpha\sqrt{N_f} \sim (\log 10^d)\sqrt{N_f}/2\pi$ at $\kappa T \sim (\log Cd\log 10)\cdot(\log 10^d)/\pi^2$ points.

2.2.11. For our 12 curves and their desired d-values, in Table 2 we give acceptable values of $(T, \kappa, \alpha, M, J)$.

N_f	d	T	κ	α	M	J
19047851	10029	2344	16.591	3684	38891	16078394
64921931	10029	2344	17.205	3684	40330	29683517
67445803	10029	2344	17.224	3684	40375	30254997
20384311	10029	2344	16.625	3684	38971	16632889
55726757	10029	2344	17.128	3684	40150	27501193
59754491	10029	2344	17.163	3684	40232	28477701
$11\cdot 25351367^2$	1033	247	24.193	391	5977	32875660156
$17\cdot 19502039^2$	1033	247	24.148	391	5966	31439905986
$19\cdot 16763912^2$	1033	247	24.053	391	5943	28571229539
$91\cdot 6350941^2$	1033	247	23.865	391	5896	23688389302
$123\cdot 5467960^2$	1033	247	23.866	391	5896	23711260510
$209\cdot 3217789^2$	1033	247	23.601	391	5831	18188943051

TABLE 2. Parameters for our 12 rank 5 curves

Thus, for the latter 6 curves we would (roughly) need to evaluate polynomials of degree 6000 at $6000 \cdot P$ points where P is as large as 55000000. Johansson kindly tested Arb's speed of `arb_poly_evaluate_vec_fast` on 3500-bit examples of degree 6000 with 6000 points, and found that it took about 60s (compared to 72s with `arb_poly_evaluate_vec_iter`, which is based on Horner's method). This then gives around 10 core-years of computation time for the worst of our examples. Further, he notes that the observed slowness seems to be due to the extra precision needed in intermediate computations (to overcome numerical instability), indeed as a ballpark estimate one needs about $4 \cdot 6000$ extra bits of precision in degree 6000. Comparatively, the two methods each took about 2s for degree 1000, and for degree 3000 the "fast" method took 15s and the "iter" method 18s. Thus it seems that our precision is not sufficiently large with respect to the length of polynomial for the asymptotic gains to be seen in practice.

2.3. We now turn to a different method of computing $\Lambda_f'''(1)$, based on the above expansion (1) and computing the special function G_3 fast. After giving a vanilla description and its analysis, we will then note that we can exploit that we are computing many values of G_3 at equi-spaced points, which then allows a practical usage of local power series expansions.

We recall from above that $G_w'(x) = -G_{w-1}(x)/x$ with $G_0(x) = e^{-x}$ so $\theta_x^w G_w(x) = (-1)^w e^{-x}$ where $\theta_x = x\partial_x$, and $\theta_x^w G_w(x) = -\partial_x \theta_x^w G_w(x)$. This indicates that G_w is holonomic (it satisfies a linear differential equation with polynomial coefficients), and so it can be computed in quasi-linear time (in the precision) via binary splitting as detailed by van der Hoeven [5].[4] This again gives a quasi-quadratic complexity overall. However, this does not seem too practical for 1000 or 10000 digits.

2.3.1. Let us discuss some alternative schemes. Taking $w = 3$ for specificity, one can note [1, (13)ff] that

$$G_3(x) = \frac{1}{3!}\Bigl(\log\frac{1}{x} - \gamma\Bigr)^3 + \frac{\zeta(2)}{2}\Bigl(\log\frac{1}{x} - \gamma\Bigr) - \frac{\zeta(3)}{3} + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^3}\frac{x^n}{n!} \quad (7)$$

where $\gamma \approx 0.577$ is Euler's constant, and the polynomial in $(\log 1/x - \gamma)$ comes about from the Maclurin expansion of $\log\Gamma(1+s)$. Furthermore, the power series coefficients $u(n) = (-1)^n/n^3 n!$ here satisfy the recursion relation $(n+1)^4 \cdot u(n+1) + n^3 \cdot u(n) = 0$, and thus u is a hypergeometric sequence (holonomic of order 1). Indeed, by shifting $k = n-1$ the sum is

[4]Of course, when we increase the precision we also need to compute $G_3(x)$-values for larger x, but it appears that [5, Theorem 3] has adequate uniformity for such purposes, in conjunction with the subsequent discussion therein involving evaluation at arbitrary points.

(here $(a)_k^{\uparrow} = a(a+1)\cdots(a-k+1)$ is the rising factorial)

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)^3}\frac{x^{k+1}}{(k+1)!} = x\sum_{k=0}^{\infty} \frac{1}{(k+1)^4}\frac{(-x)^k}{k!} = x\sum_{k=0}^{\infty}\Bigl(\frac{(1)_k^{\uparrow}}{(2)_k^{\uparrow}}\Bigr)^4 \frac{(-x)^k}{k!}$$

which amounts to evaluating a ${}_4F_4$ hypergeometric function. As alluded to above, and noted by Johansson [7, §2.2], for large $|x|$ there are difficulties in obtaining useful error bounds in an asymptotic expansion for such general hypergeometric functions, and again the practicality of the method is not immediately clear.

For $x \approx 1$ the Arb function `arb_hypgeom_pfq()` readily computes to 1000 digits in about 550 microseconds; but with $x \approx 1000$ it is about 10 times slower. (Since we have $G_3(x) \sim e^{-x}/x^3$ as $x \to \infty$, we will need to consider $x \approx 1000 \log 10$).

2.3.2. We can also work in terms of generalized integro-exponential functions as catalogued by Milgram [11, (2.6b)], where for $G_3(x)$ we want the second s-derivative at $s = 1$ of $E_s(x) = x^{s-1}e^{-x}U(s,s,x)$, where U is the confluent hypergeometric function of the second kind (Tricomi's).

For small x we imitate Johansson's protocol [7, §6.2], relating $U(a,b,z)$ to ${}_1F_1$ hypergeometric functions via the connection formula (valid for $z \neq 0$)

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

where for $b \in \mathbf{Z}$ the right side is taken as a limit. We can then compute with power series, using here $a = b = 1+\delta+O(\delta^4)$ to ensure that we are able to recover the second s-derivative with $E_s(z)$.

In practice we found there was a slowdown by about a factor of 30 when computing this, compared to $U(a,b,z)$ itself. One reason for this is rectangular splitting was only used [7, §4.2.1] by Arb in a limited case due to earlier implementation aspects. Theoretically the second parameter derivative should be about 6 times slower than $U(a,b,z)$ to compute, and with Johansson's help we were able to achieve Arb computations within about a factor of 10. Still, this was significantly slower than the above direct evaluation of the ${}_4F_4$ function.

2.3.3. There is also the idea of using parameter derivatives in conjunction with an asymptotic expansion for large x. This is mentioned in [7, §6.1], but with no real application has never been implemented as such. Though we ultimately don't really need to use this in our computations, we still give a rudimentary analysis.[5]

[5]This is quite specific to our case, and in more generality there are other facets to consider.

For a and $b-a$ that are not nonpositive integers, we recall the standard expansion (as given by Olver [13, §7.10])

$$z^a U(a,b,z) = {}_2F_0\big(a, a-b+1, -1/z\big) = \sum_{k=0}^{N-1} \frac{(a)_k^{\uparrow}(a-b+1)_k^{\uparrow}}{k!(-z)^k} + \epsilon_N(a,b,z)$$

which converges as $|z| \to \infty$. In our case we have $a=b$ so $(a-b+1)_k^{\uparrow} = k!$, and rather than a or b we write s for this variable. We assume z is on the positive real axis, writing x for it, and that $\lambda = |s|/x < 1$ so that x is in region R_1 in Olver's schema [12, §7]. The error term is then bounded as (see [12, §5] or[13, §14.5])

$$\big|\epsilon_N^s(x)\big| \le \frac{2}{1-\lambda}\left|\frac{(s)_N^{\uparrow}}{x^N}\right| \exp\biggl(\frac{2}{1-\lambda}\Bigl(\frac{|s|}{2} + \frac{\lambda(1+\lambda/4)}{(1-\lambda)^2}\Bigr)\biggr).$$

We need to take derivatives with respect to the parameter s (see [7, §2.3]). As noted by Johansson [7, §6.1], one can use Cauchy's bound on derivatives. Choosing the convenient circle $|s-1| = 1/2$ and requiring $x \ge 15$ so that $\lambda \le 1/10$, we have

$$\big|\partial_s^l \epsilon_N^s(x)\big|_{s=1} \le 2^l l! \cdot \frac{20}{9}\left|\frac{(3/2)_N^{\uparrow}}{x^N}\right| \exp\biggl(\frac{20}{9}\Bigl(\frac{3}{4} + \frac{41}{324}\Bigr)\biggr) \le 2^l l! \cdot 13 \frac{(N+1)!}{x^N}.$$

Meanwhile, we can compute the truncated hypergeometric series (k-sum) with $s = 1 + \delta + O(\delta^3)$ as a power series, for which [7, §4.2] suggests that rectangular splitting (as x does not have a short binary representation) is the best practical method.

Finally, we should not forget that we ultimately want our result with an absolute precision (of around 1000 or 10000 digits), and the extra e^{-x} with $E_s(x) = x^{s-1}e^{-x}U(s,s,x)$ means that (e.g.) with $x \approx 500 \log 10$ we need only compute $U(s,s,x)$ itself to roughly 500 relative digits to reach 1000 absolute. Therefore, the smallest x to which we could (even theoretically) apply the asymptotic expansion is $x \approx (\log 10^d)/2$, or approximately half the x-values of interest.

We used this to compute $G_3(1500)$ to 1000 absolute digits in 500 microseconds and $G_3(1200)$ in 1250 microseconds, so indeed was significantly faster than the series expansion about 0 throughout the range where the asymptotic was applicable.

2.3.4. Another computational idea (indeed, the one we find most useful in practice) is to use a local power series expansion, exploiting that the derivatives of G_3 satisfy a recursion relation. Here the equi-spaced nature of the x comes into play somewhat, but moreso that the density of relevant x is large compared to the precision. Thus the method is more useful for the six curves in $\mathcal{E}_2^{\pm}$, though it finds applicability for the curves in $\mathcal{E}_1^{\pm}$ too.

We describe the idea for curves in $\mathcal{E}_2^\pm$, where $x_0 = 2\pi/\sqrt{N_f} \approx 1/10^7$, which is a fairly small step size. We then consider n of size 10^7 or more, compute $G_3(nx_0)$ and its first two derivatives by whatever method, and then use the recursion to get (say) the first 249 derivatives, or more precisely we will utilize $G_3^{(l)}(nx_0)x_0^l/l!$ for $0 \le l \le 249$. Then for small j we have

$$G_3\big((n+j)x_0\big) \approx \sum_{l=0}^{249} \frac{G_3^{(l)}(nx_0)}{l!}(x_0)^l j^l. \tag{8}$$

For each j we can then evaluate this in about 250 additions and multiplications (having a table of j^l-values to look up).[6]

We can expect that the derivative values $G_3^{(l)}(nx_0)$ are of size $l!/(nx_0)^l$, due to the singularity of G_3 at $x = 0$. Thus the error should be $(j/n)^{250}$ which for $n \approx 10^7$ allows $j \approx 10^3$ for our target precision of 1000 digits, and for $n \approx 10^9$ allows $j \approx 10^5$. There are some other small considerations, but this rough analysis largely holds true in practice. (One can also try to minimize the number of derivatives in balance with the amortized computation cost, though 249 is not too bad in any case).

We found that Arb does about 10^7 1000-digit additions per second, and about 10^6 1000-digit multiplications per second. Even in a vanilla version of the above description, it takes about 275 microseconds for each j, which is distinctly less than the 5000 microseconds needed to compute $G_3(1000)$ by the expansion about 0. Also, the cost of each "major evaluation" of $G_3(nx_0)$ and its first 249 derivatives is largely negligible due to the allowable size of j permitting effective amortization.

Explicitly, we note $\theta_x^3 G_3 = -e^{-x}$ so $\partial_x^{l+1}(\theta_x^3 G_3) = -\partial_x^l(\theta_x^3 G_3)$ for $l \ge 0$. We then write this in terms of G_3-derivatives, with $\partial_x^l(\theta_x^3 G_3)$ equal to

$$-(-1)^l e^{-x} = x^3 G_3^{(l+3)} + (3+3l)x^2 G_3^{(l+2)} + (3l^2+3l+1)xG_3^{(l+1)} + l^3 G_3^{(l)},$$

and thereby we can recursively compute $G_3^{(l+3)}$ at any given x-value.

Note that this use of such a recursion loses relative precision, and indeed fairly rapidly, for if we approximate the 0th-2nd derivatives with an error of ε, then the lth derivative of G_3 will have an error of roughly $3^l l!\varepsilon$. However, this is an illusory worry in our context, as the $l!$ in the denominator of the local power series cancels much of this already, and moreover the latter l-terms in the expansion don't need much relative precision anyway (much dominated by $(jx_0)^l$ being small).

For the initial values, one can compute $\theta_x^q G_3 = (-1)^q G_{3-q}$ for $q \le 2$ by either the series about $x = 0$ or the asymptotic method, and then use

[6] We can also compute the sum and difference of $G_3((n \pm j)x_0)$ each via power series of half the density, essentially saving a factor of 2. Another minor savings can come from shortening the sum for small j. Finally, in some ranges the j^l will have sufficiently short representations to expedite the multiplications.

$\theta_x G_3 = xG_3' = -G_2$ and $G_1 = \theta_x^2 G_3 = xG_3' + x^2 G_3'' = -G_2 + x^2 G_3''$ to recover G_3' and G_3'' at any given x.

Each step in the recursion takes 3 full multiplications and 3 multiplications by short integers, and then usage in the power series requires additional multiplication by $x_0^l/l!$. The other overhead cost of computing the initial values varies depending on x (again one could compute them by local power series, but we chose to use the other methods as a sanity check). For the curves in $\mathcal{E}_2^{\pm}$ our batch-size was $|j| \leq 2^{10}$, which easily assured that the overhead costs were small.

For curves in $\mathcal{E}_1^{\pm}$, where loosely we have $x_0 \approx 1/10^3$ and desire 10000 digits, the parameters are more delicate to work out. With $n \approx 10^7$ the expansion around 0 takes about 0.25s for 10000 digits, which is roughly equivalent to 7500 multiplications at this precision. If we take 2500 derivatives in a local power series we will win comfortably, especially with the various other factors to be gained (from pairing $\pm j$ and small j^l-representations). We were able to profitably use the local power series with n as small as 10^5 with a batch-size of $|j| \leq 100$.

3. Computational details for the 12 curves

We now give some computational details for the 12 curves in Table 1. This naturally splits into two cases, depending on whether the curve is in $\mathcal{E}_1^{\pm}$ or $\mathcal{E}_2^{\pm}$.

We might note in passing that each of the 12 curves in Table 1 does indeed have rank 5, as can be verified in a few seconds with a computer algebra system (using 2-descent and point searches); moreover, a difficult theorem of Kolyvagin [10] following work of Gross and Zagier [4] then implies that $\Lambda_E'(1) = 0$ for each curve. Thus it is indeed the third central derivative that is of most prominent interest.

3.1. First we give a rigourous bound for the truncation error when using local power series in (8). Recall that for $w \geq 1$ we have

$$G_w(x) = \frac{1}{(w-1)!}\int_1^\infty e^{-xy}(\log y)^{w-1}\frac{\partial y}{y},$$

and with $G_0(x) = e^{-x}$ this satisfies $G_w'(x) = -G_{w-1}(x)/x$. The integrand here (and thus G_w itself) is positive, so that $G_3'(x) = -G_2(x)/x$ is negative. We also have $|G_3(z)| \leq G_3(\operatorname{Re} z)$, so that by Cauchy's bound on derivatives we deduce the bound $\left|G_3^{(l)}(x)\right| \leq l! \cdot G_3(x-R)/R^l$ for any $0 < R < x$.

We do not try to optimize R precisely, but instead give a reasonable value of it for use in computation. Noting that G_3 has a mild increase (log singularity) near 0 and a decay like e^{-x}/x^3, if we want to bound the truncation error from a local power series with M terms around x, it is reasonable to take R of size $x(1 - 1/\sqrt{M})$.

In practice, for the curves in $\mathcal{E}_2^\pm$ we took R as $15x/16$, which implies the bound $\bigl|G_3^{(l)}(x)\bigr| \le l!(16/15x)^l G_3(x/16)$. The truncation error when using the local power series for $G_3((n+j)x_0)$ is thus bounded as

$$\le G_3\Bigl(\frac{nx_0}{16}\Bigr)\sum_{l=M}^{\infty}\Bigl(\frac{16j}{15n}\Bigr)^l \le 2G_3\Bigl(\frac{nx_0}{16}\Bigr)\cdot\Bigl(\frac{16j}{15n}\Bigr)^M,$$

the last step assuming $|16j/15n| \le 1/2$. With $\mathcal{E}_1^\pm$ we took R as $63x/64$.

3.2. We also need a rudimentary bound for the error in truncating the n-series in (1). We have $G_3(x) \le 2e^{-x}/x^3$ for $x \ge 10$. It is convenient to write the truncation limit as $M = \lambda\sqrt{N_f}/2\pi$ where $\lambda \ge 1000$, and using $|c_f(n)| \le n$ we find

$$\Biggl|\sum_{n\ge M}\frac{c_f(n)}{n}G_3\Bigl(\frac{2\pi n}{\sqrt{N_f}}\Bigr)\Biggr| \le 2\frac{\sqrt{N_f}}{2\pi}\int_\lambda \frac{e^{-u}}{u^3}\,\partial u \le 3\frac{\sqrt{N_f}}{2\pi}\frac{e^{-\lambda}}{\lambda^3}.$$

For a target precision of d digits (with $d \ge 1000$) we take $\lambda = d\log 10$.

3.3. Let us give some timings for an example curve in each class.

Consider the curve 11a twisted by -25351367 for which

$$x_0 = \frac{2\pi}{25351367\sqrt{11}} \approx 0.7472779\cdot 10^{-7}.$$

All of the computations will be to an absolute precision of 1060 digits, which suffices to have an adequate final error.

With $n = 12345678$ (so $nx_0 \approx 0.922565$), computing $G_3\bigl((n+j)x_0\bigr)$ for $|j| \le 1000$ takes about 1 second by using the series expansion about 0 and the computation of the ${}_4F_4$ hypergeometric function (as implemented in Arb). Meanwhile, the local power series method is about 6 times as fast.

Comparatively, with $n = 10\cdot 2^{30} + 12345678$ (which is $nx_0 \approx 803.36$) and $|j| \le 10^5$ it takes 24 seconds to compute all the G_3-values by the local power series method, while using the expansion about 0 takes around 35 times as long. Note that this is not yet to an applicable range for the asymptotic expansion, where we would require nx_0 to exceed $500\log 10 \approx 1151.3$ or so.

3.3.1. Consider the curve of conductor 67445803 in Table 1, so that we have $x_0 = 2\pi/\sqrt{N_f} \approx 0.765\cdot 10^{-3}$. It takes 3.6s to compute 200 values to 10000 digits at $n = 1234$ by the series about 0. This increases to 12.7s around $n = 1234567$ and 48.6s for $n = 12345678$.

Comparatively, it takes about 5.6s to compute 200 values via the local power series, and 50.2s to compute 2000 values. This is almost independent of the size of n (though we do need to ensure j/n is small enough). For large n this is thus about 10 times faster.

3.4. For the six curves in $\mathcal{E}_1^{\pm}$ we ultimately chose the following division: for $n \le 10^5$ we compute G_3 by the expansion about 0; for $10^5 \le n \le 3 \cdot 10^6$ we compute it by a local power series of length 3360 with $|j| \le 100$; and for $n \ge 3 \cdot 10^6$ we use a local power series of length 2520 with $|j| \le 300$.

The final timings are as given in Table 1. (We refer to our comments in §1.5.1 regarding processor types and rescalings). We do not doubt that optimizing parameters and various code improvements could accrue multiple factors of 10-20% here, and an overall speed-up by a factor of 2 would not be shocking.

Described in more raw terms, for instance for the curve of conductor 19047851, we wish to carry out about 16 million calculations of G_3, each two of which require about 2500 10000-digit multiplications, and we can carry out such arithmetic at about 32000 multiplications per second. This would crudely give a time of 625000s, and in practice we beat that by a factor of 3/2, presumably since many multiplications are by short representations.

3.4.1. For the six curves in $\mathcal{E}_2^{\pm}$ we used the expansion about 0 for $n \le 2^{24}$ and the local power series with $|j| \le 2^{10}$ for larger n. We adjusted the length of the local power series based upon n, with it starting at 252 for $n \approx 2^{24}$.

We can note that when the twisting parameter is even we have $c_f(n) = 0$ for even n, saving a factor of 2 in these cases.

The final timings are listed in Table 1. For these computations we used parallelization in an obvious manner, splitting up the n-values into (say) lots of size of 2^{30}. Again I expect reasonable improvements can be made time-wise, though not quite to the same extent as with the curves in $\mathcal{E}_1^{\pm}$.

As a rough analysis, for the twist of 11a we compute about 33 billion values of G_3, each two of which require 250 1000-digit multiplications, and the arithmetic can handle 10^6 of these per second. This gives an estimate a bit over 4 million seconds, and in practice we do about 3 times better, from short representations and also saving an additional factor by shortening the local power series as n increases.

3.4.2. Note that the asymptotic complexity of our method is actually quasi-cubic in the precision. Indeed, the number of G_3-evaluations grows linearly, the number of terms in the local power series essentially does also, and the time for each multiplication grows quasi-linearly (at least asymptotically). However, this still seems to be the superior method in practice, at least for the examples we considered here.

3.5. Let us explicitly state the results of our computations.

Theorem 3.5.1. *For each of the six curves in $\mathcal{E}_1^{\pm}$ we have $\Lambda_E'(1) = 0$ (by Kolyvagin's theorem) and $|\Lambda_E'''(1)| \le 10^{-10025} \cdot (2\pi/\sqrt{N_E})$.*

For each of the six curves in $\mathcal{E}_2^{\pm}$ we have $\Lambda_E'(1) = 0$ (by Kolyvagin's theorem) and $|\Lambda_E'''(1)| \le 10^{-1025} \cdot (2\pi/\sqrt{N_E})$.

References

[1] J. P. Buhler, B. H. Gross, D. B. Zagier, *On the conjecture of Birch and Swinnerton-Dyer for an elliptic curve of rank 3.* Math. Comp. **44** (1985), 473–481. `http://doi.org/10.1090/S0025-5718-1985-0777279-X`

[2] J. E. Cremona, *Algorithms for Modular Elliptic Curves.* Cambridge University Press, 1992. Second ed. 1997. `http://homepages.warwick.ac.uk/~masgaj/book/fulltext/index.html`

[3] T. Dokchitser, *Computing Special Values of Motivic L-functions.* Experiment. Math. **13** (2004), no. 2, 137–149. `http://projecteuclid.org/euclid.em/1090350929`

[4] B. H. Gross, D. B. Zagier, *Heegner points and derivatives of L-series.* Invent. Math. **84** (1986), no. 2, 225–320. `http://eudml.org/doc/143341`

[5] J. van der Hoeven, *Fast evaluation of holonomic functions.* Theor. Comp. Sci. **210** (1999), no. 1, 199–215. `http://doi.org/10.1016/S0304-3975(98)00102-9`

[6] F. Johansson, *Arb: efficient arbitrary-precision midpoint-radius interval arithmetic.* IEEE Trans. Comp., **66** (2017), no. 8, 1281–1292. `http://doi.org/10.1109/TC.2017.2690633`

[7] F. Johansson, *Computing hypergeometric functions rigourously.* INRIA preprint, 2016. `http://hal.inria.fr/hal-01336266`

[8] P. Kirrinnis, *Partial Fraction Decomposition in* $\mathbf{C}(z)$ *and Simultaneous Newton Iteration for Factorization in* $\mathbf{C}(z)$. J. Complexity **14** (1998), no. 3, 378–444. `http://doi.org/10.1006/jcom.1998.0481`

[9] A. Kobel, M. Sagraloff, *Fast Approximate Polynomial Multipoint Evaluation and Applications.* Preprint, 2013, 2016.

[10] В. А. Колывагин [V. A. Kolyvagin], Конечность $E(\mathbf{Q})$ и Ш$(E,\mathbf{Q})$ для подкласса крибых бейля. (Russian) [Finiteness of $E(\mathbf{Q})$ and Ш$(E,\mathbf{Q})$ for a subclass of Weil curves]. Изв. Акад. Наук УССР Сер. Мат. **52** (1988), no. 3, 522–540 (Russian). English translation: Math. USSR Izv. **32** (1989), 523–541. `http://mi.mathnet.ru/eng/izv1191` `http://doi.org/10.1070/IM1989v032n03ABEH000779`

———, *Euler systems.* In *The Grothendieck Festschrift* (Vol. II), ed. P. Cartier *et al.* Prog. in Math. **87** Birkhäuser Boston (1990), 435–483. `http://doi.org/10.1007/978-0-8176-4575-5`

[11] M. S. Milgram, *The generalized integro-exponential function.* Math. Comp. **44** (1985), 443–458. `http://doi.org/10.1090/S0025-5718-1985-0777276-4`

[12] F. W. J. Olver, *On the Asymptotic Solution of Second-Order Differential Equations Having an Irregular Singularity of Rank One, with an Application to Whittaker Functions.* J. SIAM Numer. Anal. Ser. B **2** (1965), 225–243. `http://doi.org/10.1137/0702017`

[13] F. W. J. Olver, *Asymptotics and Special Functions.* Academic Press, 1974. `http://doi.org/10.1016/C2013-0-11254-8` `http://doi.org/10.1201/9781439864548`

[14] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod p.* Math. Comp. **44** (1985), 483–494. `http://doi.org/10.2307/2007968`

[15] M. Watkins, *A new effective lower bound for* $L_\chi(1)$. Preprint, 2019.

———, *Class number problems for real quadratic fields with small fundamental unit.* Preprint, 2021.

[16] A. Wiles, *Modular elliptic curves and Fermat's Last Theorem.* Ann. of Math. (2) **141** (1995), no. 3, 443–551. `http://doi.org/10.2307/2118559`

School of Mathematics and Statistics, University of Sydney, AUSTRALIA
E-mail : `watkins@maths.usyd.edu.au`