HYPERGEOMETRIC L-FUNCTIONS IN AVERAGE POLYNOMIAL TIME, II

EDGAR COSTA, KIRAN S. KEDLAYA, AND DAVID ROE

ABSTRACT. We describe an algorithm for computing, for all primes $p \leq X$, the trace of Frobenius at p of a hypergeometric motive over $\mathbb{Q}$ in time quasilinear in X. This involves computing the trace modulo p^e for suitable e; as in our previous work treating the case $e = 1$, we combine the Beukers–Cohen–Mellit trace formula with average polynomial time techniques of Harvey and Harvey–Sutherland. The key new ingredient for $e > 1$ is an expanded version of Harvey's "generic prime" construction, making it possible to incorporate certain p-adic transcendental functions into the computation; one of these is the p-adic Gamma function, whose average polynomial time computation is an intermediate step which may be of independent interest. We also provide an implementation in Sage and discuss the remaining computational issues around tabulating hypergeometric L-series.

1. Introduction

We continue the investigation begun in [CKR20] of computational aspects of L-functions associated to *hypergeometric motives* in the sense of [RR22]. These L-functions are easily accessed via the Beukers–Cohen–Mellit trace formula [BCM15] together with the p-adic expression of Gauss sums via the Gross-Koblitz formula [GK79]. While the trace formula has $O(p)$ terms, the main result of [CKR20] gives a way to amortize the cost over p; that is, one obtains an efficient algorithm for computing, for all primes $p \leq X$, the mod-p reduction of the trace of Frobenius at p of a fixed hypergeometric motive in time quasilinear in X.

Here we do the same for the mod-p^e reduction for any positive integer e, answering a question raised at the end of [CKR20]. Since the trace is an integer in a known range (see Remark 4.11), this yields an algorithm for computing the exact trace. (See Definition 4.1 for terminology.)

Theorem 1.1. *Algorithm 3, on input of a Galois-stable hypergeometric datum $(\alpha, \beta) \in \mathbb{Q}^r \times \mathbb{Q}^r$, a parameter $z \in \mathbb{Q} \setminus \{0, 1\}$, and a positive integer X, computes the hypergeometric trace $H_p \left(\begin{matrix} \alpha \\ \beta \end{matrix} \middle| z \right)$ for all primes $p \leq X$ (excluding tame and wild*

Date: May 2024.

2020 *Mathematics Subject Classification.* 11Y16, 33C20 (primary), and 11G09, 11M38, 11T24 (secondary).

Costa and Roe were supported by the Simons Collaboration on Arithmetic Geometry, Number Theory, and Computation via Simons Foundation grant 550033. Kedlaya was supported by NSF (DMS-1802161, 2053473), UC San Diego (Warschawski Professorship), and the Simons Foundation (Simons Fellows program, 2023–2024); he was also hosted by HIM (Bonn) during summer 2023 and IAS (Princeton) during fall 2023. We thank Will Sawin and Drew Sutherland for helpful discussions.

Submitted to Algor. Num. Th. Symp.

primes). Assuming that $r = O(\log X)$ and the bitlength of z is $O(r^2 \log X)$, the time and space complexities are respectively bounded by

$$O(r^5 X (\log X)^3) \text{ bit operations and } O(r^5 X (\log X)^2) \text{ bits.}$$

As in [CKR20], the general strategy is to amortize the computational work over primes using *average polynomial time* techniques of Harvey [Har14; Har15] and Harvey–Sutherland [HS14; HS16]. In [CKR20], this amortization is achieved by expressing the trace formula in terms of a series of matrix products of a special form: we have a sequence of matrices defined over $\mathbb{Z}$, and the desired quantity is obtained by truncating the product at an index depending (linearly) on p and reducing modulo p. One then uses an *accumulating remainder tree*[1] to compute the products and remainders in essentially linear time.

Since the expression we use for hypergeometric traces involves the p-adic Gamma function Γ_p, we first describe average polynomial time algorithms to compute this function. These reduce to applications of remainder trees quite close to the original work of Costa–Gerbicz–Harvey [CGH14] that inspired Harvey's work on L-functions, which may be of independent interest.

Our main algorithm uses a variant of Harvey's *generic prime* construction [Har15, §4.4]. In its simplest form, this consists of using matrices over the truncated polynomial ring $\mathbb{Z}[P]/(P^e)$, arranged in such a way that truncating the matrix product at an index depending on p and specializing along the map $\mathbb{Z}[P]/(P^e) \to \mathbb{Z}/(p^e)$ taking P to p yields the desired result. In practice, such a product can be handled by encoding it as a product of block matrices over $\mathbb{Z}$; see Example 2.8.

Here, we adapt the construction of [CKR20] to use a block lower triangular matrix to record both a product over $\mathbb{Z}[P]/(P^e)$ and the sum of the partial products. Two new complications arise for $e > 1$: we must perform an additional transformation on the product before adding it to the sum, and we must also incorporate the contribution of certain structural constants depending on p, which we treat as further unknowns. (These unknowns are derived from the series expansions for Γ_p and from the difference between the evaluation point z and the $(p-1)$-st root of unity in its mod-p residue disc.)

As in [CKR20], we have implemented the algorithm described above (for arbitrary e) in Sage [Sag24] with some low-level code written in Cython, including a wrapper to Sutherland's C library rforest (wrapped in Cython) to implement Theorem 2.1. Some sample timings are included in Table 1, including comparisons with Sage and Magma [BCP97]; see §6 for explanation. These confirm that the quasilinear complexity shows up in practice, not just asymptotically. Our code, which also includes the algorithm from [CKR20] for $e = 1$, is available on GitHub [CKR23] at

`https://github.com/edgarcosta/amortizedHGM`.

The broader context of our work is the desire to tabulate hypergeometric L-functions at scale in the L-Functions and Modular Forms Database (LMFDB) [LMFDB], in part to investigate the *murmurations* phenomenon for these L-functions [HLOP22; LOP24]. Our timings suggest that this prospect is within reach; we discuss this in §7. In particular, while there should exist an analogue of Theorem 1.1 for p^f-Frobenius traces for any fixed $f > 1$, it does not seem to be needed in practice.

[1] In practice one uses accumulating remainder *forests* for improved efficiency, especially with regard to memory usage. As we are using the technique as a black box, we ignore this distinction.

2. Accumulating remainder trees and generic primes

We use accumulating remainder trees to amortize the computation of the trace formula, following [CKR20]. As we use this construction as a black box, we recall only the structure of the input and output and the overall complexity estimates.

Theorem 2.1. *Fix a positive integer e. Suppose we are given*

- *a list of $r \times r$ matrices $A_0, \dots, A_{b-1}$ over $\mathbb{Z}$,*
- *a list of primes $p_1, \dots, p_c$, and*
- *a list of distinct cut points $0 \le b_1, \dots, b_c \le b$.*

Let B be an upper bound on the bit size of $\prod_{j=1}^c p_j$ and H an upper bound on the bit size of any p_i^e or any entry of A_i. Assume also that $\log r = O(H)$ and $r = O(\log b)$. Then there is an algorithm that computes

$$C_n := A_0 \cdots A_{b_n - 1} \bmod p_n^e \qquad (1 \le n < c)$$

with time complexity

$$O(r^2(eB + bH)\log(eB + bH)\log b)$$

and space complexity

$$O(r^2(eB + bH)\log b).$$

Proof. This follows from [HS16, Thm. 3.2][2] (an improvement of [HS14, Thm. 4.1]) via a change of notation as in [CKR20, Definition 3.1, Algorithm 2]. □

Example 2.2. In practice, we will apply Theorem 2.1 in a restricted fashion.

- The matrix A_i will be the specialization of a single matrix A over $\mathbb{Z}[k]$ at $k = i$, whose entries have degree $O(d)$ and coefficients of bit size $O(d \log X)$.
- The primes p_i will all lie in a fixed arithmetic progression bounded by X.
- The cut point b_i will be a linear function of p_i (up to rounding) whose coefficients are $O(1)$.

We may then take $b = O(X)$, $B = O(X)$, and $H = O((d+e)\log X)$. In particular, $eB + bH = O((d+e)X \log X)$; assuming that $d + e = O(X)$ and $r = O(\log X)$, the time and space complexities in Theorem 2.1 become respectively

$$O((d+e)r^2 X(\log X)^3) \qquad \text{and} \qquad O((d+e)r^2 X(\log X)^2).$$

Example 2.3. One basic instantiation of Example 2.2 is to batch-compute the quantities $(\lceil \gamma p \rceil - 1)! \pmod{p^e}$ for some $\gamma \in (0,1] \cap \mathbb{Q}$. For instance, the case $e = 2, \gamma = 1$ is the focus of [CGH14].

Example 2.4. Let j be a positive integer and choose $\gamma \in (0,1]$. We may also use Example 2.2 to batch-compute the quantities

$$H_{j,\gamma}(p) \pmod{p^e}, \qquad H_{j,\gamma}(p) := \sum_{i=1}^{\lceil \gamma p \rceil - 1} i^{-j} \tag{2.5}$$

by interpreting them as

$$H_{j,\gamma}(p) = S(p)_{21}/S(p)_{11}, \qquad S(p) := \prod_{i=1}^{\lceil \gamma p \rceil - 1} \begin{pmatrix} i^j & 0 \\ 1 & i^j \end{pmatrix}.$$

[2] The complexities in *loc. cit.* are stated in terms of the complexity of multiplying two n-bit integers. Per [HH21] we take this to be $O(n \log n)$.

Submitted to Algor. Num. Th. Symp.

Remark 2.6. In Example 2.4, we can also write

$$H_{j,\gamma}(p) = (VS(p))_{11}/(VS(p))_{12}, \qquad V := \begin{pmatrix} 0 & 1 \end{pmatrix}.$$

Prepending V to the product yields some significant computational savings, by reducing the size of the intermediate products in the remainder tree computation.

Notation 2.7. We write $f^{[h]}$ for the coefficient of x^h in the polynomial $f(x)$.

Example 2.8. The paradigm of Example 2.2 excludes the possibility of computing expressions involving p other than via the cut point. Harvey's "generic prime" construction circumvents this issue by instead computing over $\mathbb{Z}[x]/(x^e)$, where x is specialized to p as a postprocessing step. We use a variant of this idea in §5.

In lieu of implementing Theorem 2.1 with $\mathbb{Z}$ replaced by $\mathbb{Z}[x]/(x^e)$, we encode a matrix product over $\mathbb{Z}[x]/(x^e)$ via a lower triangular block representation

$$f \mapsto (f^{[h_1-h_2]})_{h_1,h_2=1,\dots,e}. \tag{2.9}$$

One could alternatively represent polynomials via Schönhage's method, i.e., giving their evaluations at a large power of 2 [Sch82]. In theory, this would improve our runtime by replacing matrix multiplication with integer multiplication. However, in §5 we will manipulate triangular matrices in a way that seems incompatible with this strategy.

Remark 2.10. Example 2.4 can be interpreted either as computing $\prod_{i=1}^{\lceil \gamma p \rceil - 1}(i^j + x) \pmod{(x^2, p^e)}$ as in Example 2.8, or as a "hypergeometric" construction using

$$\prod_{i=1}^{\lceil \gamma p \rceil -1} \begin{pmatrix} g(i-1) & 0 \\ 1 & f(i) \end{pmatrix} \text{ to compute } \sum_{i=1}^{\lceil \gamma p \rceil -1} \prod_{j=1}^{i-1} \frac{f(j)}{g(j)},$$

taking $f(i) := i^j, g(i) := (i+1)^j$. The latter form also covers the construction in [CKR20] and inspires, but does not directly encompass, the construction in §5.6.

Remark 2.11. In many cases, Theorem 2.1 is used in a "projective" manner, in that the matrix product is only needed up to scalar multiples (e.g., all cases covered by Remark 2.10, and the construction of §5.6). It may be possible to exploit this to reduce the complexity of some intermediate matrices, but we did not pursue this.

To illustrate the potential savings, let us make these common factors explicit in Example 2.4. In the ring $\mathbb{Z}[x]/(x^2)$, $\prod_{i=1}^k (i^j + x)$ is divisible by $\prod_{i=1}^k \tilde{f}(i)$ where

$$\tilde{f}(i) = \begin{cases} (i/p)^j & \text{if } i = p^e \text{ for some prime } p \text{ and some } e > 0, \\ i^j & \text{otherwise.} \end{cases}$$

3. The p-adic Gamma function

We next recall some properties of the p-adic Gamma function, and then give an average polynomial time algorithm to compute series expansions of it at rational arguments. This may be of independent interest.

To avoid some complications, notably in (3.7), we assume $p > 2$.

Definition 3.1. The (Morita) p-adic Gamma function is the unique continuous function $\Gamma_p \colon \mathbb{Z}_p \to \mathbb{Z}_p^\times$ which satisfies

$$\Gamma_p(n+1) = (-1)^{n+1} \prod_{\substack{i=1 \\ (i,p)=1}}^{n} i = (-1)^{n+1} \frac{\Gamma(n+1)}{p^{\lfloor n/p \rfloor} \Gamma(\lfloor n/p \rfloor + 1)} \tag{3.2}$$

for all $n \in \mathbb{Z}_{\geq 0}$. It satisfies the functional equations

$$\Gamma_p(x+1) = \omega(x)\Gamma_p(x), \qquad \omega(x) := \begin{cases} -x & \text{if } x \in \mathbb{Z}_p^\times, \\ -1 & \text{if } x \in p\mathbb{Z}_p, \end{cases} \tag{3.3}$$

$$\Gamma_p(x)\Gamma_p(1-x) = (-1)^{x_0}, \tag{3.4}$$

in which $x_0 \in \{1, \dots, p\}$ is congruent to $x \in \mathbb{Z}_p$ mod p. There is also an analogue of the Gauss multiplication formula [Rob00, §VII.1.3]; it may be possible to use this to streamline our algorithms, but we have not pursued this.

Definition 3.5. As in [Rob00, §VII.1.6] or [Rod07, §6.2], for any $a \in \mathbb{Z}_p$ the restriction of Γ_p to the disc $a + p\mathbb{Z}_p$ admits a series expansion

$$\Gamma_p(x+a) = \sum_{i=0}^{\infty} c_i x^i \qquad (c_i \in \mathbb{Z}_p) \tag{3.6}$$

In particular, Γ_p is Lipschitz continuous with $C = 1$, i.e.,

$$|\Gamma_p(x) - \Gamma_p(y)|_p \leq |x-y|_p\,. \tag{3.7}$$

Definition 3.8. Let $\log\colon \mathbb{Z}_p^\times \to \mathbb{Z}_p$ denote the p-adic logarithm, which vanishes on roots of unity and is given by the usual power series on $1 + p\mathbb{Z}_p$. Then (3.6) immediately implies that $\log \Gamma_p$ admits a series expansion around any $a \in \mathbb{Z}_p$ with coefficients in $\mathbb{Z}_p$.

For $\gamma \in (0,1]$ and $x \in p\mathbb{Z}_p$, with $H_{j,\gamma}(p)$ as defined in (2.5), (3.3) implies that

$$\begin{aligned} \log \frac{\Gamma_p(x + \lceil \gamma p \rceil)}{\Gamma_p(\lceil \gamma p \rceil)} &= \log \frac{\Gamma_p(x)}{(\lceil \gamma p \rceil - 1)!} \prod_{i=0}^{\lceil \gamma p \rceil - 1} \frac{\Gamma_p(x+i+1)}{\Gamma_p(x+i)} \\ &= \log \Gamma_p(x) + \sum_{i=1}^{\lceil \gamma p \rceil - 1} \log\left(1 + \frac{x}{i}\right) \\ &= \log \Gamma_p(x) - \sum_{j=1}^{\infty} \frac{(-x)^j}{j} H_{j,\gamma}(p). \end{aligned} \tag{3.9}$$

Remark 3.10. We will use a naïve quadratic estimate for the runtime of computing the p-adic logarithm and exponential; while asymptotically these require quasilinear time [Ber08, §7, 8], [CMTV21, §3], we will consider input sizes much smaller than the asymptotic crossover and so the quadratic estimates are more accurate in practice. Similar considerations will apply to integer multiplication outside of Theorem 2.1.

We first give an amortized computation of the expansion of Γ_p around 0. It is equivalent, and will be convenient for later, to work instead with $\log \Gamma_p$.

Theorem 3.11. *Let T be the set of primes. Then Algorithm 1 computes the series expansions of $\log \Gamma_p(py)$ in $\mathbb{Z}[y]/(p^e)$ for all primes $p \in T_X := T \cap [e+1, X]$, with respective time and space complexities*

$$O(e^2 X (\log X)^3 + (e^4 \log e) X) \qquad \text{and} \qquad O(eX(\log X)^2 + e^2 X).$$

Proof. To see that the output is correct, fix a prime $p \geq e+1$. For $j \not\equiv 0 \pmod{p-1}$, $H_{j,1}(p) \equiv 0 \pmod{p}$ and so $(py)^j H_{j,1}(p) \equiv 0 \pmod{p^{j+1}}$. Hence the terms $j > e-2$

Submitted to Algor. Num. Th. Symp.

Algorithm 1: Expansion of $\log \Gamma_p(x)$ modulo p^e (Theorem 3.11)

1. Apply Theorem 2.1 to compute $(p-1)! \pmod{p^e}$ for all $p \in T_X$.
2. For $j = 1, \dots, e-2$, apply Theorem 2.1 as in Example 2.4 to compute $H_{j,1}(p) = \sum_{i=1}^{p-1} i^{-j} \pmod{p^{e-j}}$ for all $p \in T_X$.
3. Let A be the upper triangular $(e-1) \times (e-1)$ matrix with $A_{ij} = \binom{j}{i-1}$ for $1 \le i \le j \le e-1$. For each $p \in T_X$, form the vectors $v, w \in (\mathbb{Z}/p^e\mathbb{Z})^{e-1}$ given by
$$v_j = \begin{cases} \log(-(p-1)!) & \text{if } j = 1 \\ \frac{(-1)^j}{j-1} p^{j-1} H_{j-1,1}(p) & \text{if } j > 1, \end{cases} \qquad w = A^{-1}v.$$
Then $\log \Gamma_p(py) \equiv \sum_{j=1}^{e-1} w_j y^j \pmod{p^e}$ in $\mathbb{Z}[y]$.

do not contribute modulo p^e in (3.9), so we may rewrite the latter as

$$\begin{aligned} \log \Gamma_p(p(y+1)) - \log \Gamma_p(py) &\equiv \log\bigl(-(p-1)!\bigr) - \sum_{j=1}^{e-2} \frac{(-py)^j}{j} H_{j,1}(p) \pmod{p^e} \\ &\equiv \sum_{j=1}^{e-1} v_j y^{j-1} \pmod{p^e}. \end{aligned}$$

Therefore, the right hand side is the image of $\log \Gamma_p(py)$ under the difference operator, which is represented by A. Since A has diagonal entries $1, \dots, e-1$, it is invertible over $\mathbb{Z}_p$. Since $\Gamma_p(0) = 1$, $A^{-1}v$ contains the coefficients of the expansion of $\log \Gamma_p(py)$ in y.

For the complexity estimate, we may assume $e < X$ as otherwise there is nothing to do. We cover Steps 1 and 2 by applying Theorem 2.1 $e-1$ times as in Example 2.2 with a degree bound of $O(e)$. This costs $O(e^2 X (\log X)^3)$ time and $O(eX(\log X)^2)$ space, plus $O(e^2 X)$ space to record the results. In Step 3, for each of $O(X/\log X)$ primes p, we perform one logarithm and $O(e^2)$ multiplications of an integer in $\mathbb{Z}/(p^e)$ with an entry of A^{-1} of bitsize[3] $O(e \log e)$; as per Remark 3.10 each multiplication costs $O(e^2 \log e \log X)$ time. □

Remark 3.12. One can extend Algorithm 1 to cover $p > \frac{e}{2}$ using the fact that $\log \Gamma_p$ is an odd series [Rob00, §VII.1.5, Theorem]. For smaller p, we can replace the use of interpolation in step 2 with a direct application of (3.9) to solve for the coefficients of $\Gamma_p(x)$. As we will assume $p > e$ later, we did not implement these steps.

We next expand around other values $\gamma \in (0,1) \cap \mathbb{Q}$. For these values, it is more useful to retain neither $\Gamma_p(py + \gamma)$ nor its logarithm, but something in between.

Theorem 3.13. *Fix integers $e, d \ge 2$. Let S_d be the set of $\gamma \in \mathbb{Q} \cap (0,1)$ of the form $\frac{c}{d}$ with $\gcd(c,d) = 1$. Let T be the set of primes not dividing d. Then Algorithm 2 computes, for all $\gamma \in S_d$ and all $p \in T_X := T \cap [e+1, X]$, some quantities*

- *$c_{\gamma,p} \in \mathbb{Z}/(p^e)$ and*
- *$s_{\gamma,p}(py) \in \mathbb{Z}[y]/(p^e)$*

such that
$$\Gamma_p(py + \gamma) \equiv c_{\gamma,p} \exp s_{\gamma,p}(py) \pmod{p^e},$$

[3]This estimate follows by expressing A^{-1} in terms of Bernoulli numbers.

Algorithm 2: Expansion of $\log \Gamma_p(x+\gamma)$ modulo p^e (Theorem 3.13)

1 Use Algorithm 1 to compute $\log \Gamma_p(py) \pmod{p^e}$ for all $p \in T_X$.
2 For each $\gamma \in S_d$, use Theorem 2.1 to compute $\Gamma_p(\lceil \gamma p \rceil) = \pm(\lceil \gamma p \rceil - 1)! \pmod{p^e}$ for all $p \in T_X$.
3 For each $\gamma \in S_d$, for $j = 1, \dots, e-1$, apply Theorem 2.1 as in Example 2.4 to compute $H_{j,\gamma}(p) \pmod{p^{e-j}}$ for all $p \in T_X$.
4 For each $p \in T_X$ and $\gamma \in S_d$, use Step 1 and (3.9) to compute $\log \frac{\Gamma_p(py+\lceil \gamma p \rceil)}{\Gamma_p(\lceil \gamma p \rceil)}$ as an element of $\mathbb{Z}[y]/(p^e)$.
5 For each $p \in T_X$ and $\gamma \in S_d$, set $b := -(\gamma d)/p \pmod d$. Set $c_{\gamma,p} := \Gamma_p\big(\lceil \frac{b}{d} p \rceil\big)$ and $s_{\gamma,p}(py) := \log \Gamma_p\big(py + \lceil \frac{b}{d} p \rceil\big)/\Gamma_p\big(\lceil \frac{b}{d} p \rceil\big)\big|_{y=y-\frac{b}{d}}$.

with respective time and space complexities bounded by

$$O(de^2 X (\log X)^3 + de^4(\log d + \log e) X) \qquad \textit{and} \qquad O(eX(\log X)^2 + de^2 X).$$

Proof. To see that the output is correct, we first show that $py+\gamma = p(y-b/d)+\lceil \frac{b}{d}p\rceil$. Set $a := \lceil \frac{b}{d} p \rceil$, $a' := \gamma \pmod p$, and define $b' \in \mathbb{Z}$ by $\gamma d = a'd - b'p$; then $\frac{b'}{d} \in S_d$, $a' = \frac{b'}{d}p + \gamma = \lceil \frac{b'}{d} p \rceil$, and $\gamma d \equiv -b'p \pmod d$. We deduce that $a = a', b = b'$ and hence $a \equiv \gamma \pmod p$, and $b = d(a-\gamma)/p$. We thus reduce to (3.9) and Theorem 3.11.

For the complexity estimate, we may again assume $e < X$. We cover Step 1 using Theorem 3.11. We cover Steps 2 and 3 by applying Theorem 2.1 $O(de)$ times as in Example 2.2 with a degree bound of $O(e)$. This costs $O(de^2 X(\log X)^3)$ time and $O(eX(\log X)^2)$ space, plus $O(de^2 X)$ space to record the results; this cost also covers Step 4. In Step 5, for each of $O(d)$ values of b and $O(X/\log X)$ primes p, we perform a polynomial substitution using $O(e^2)$ multiplications of an element of $\mathbb{Z}/(p^e)$ by a rational of bitsize $O(e \log d)$; as per Remark 3.10 each multiplication costs $O(e^2 \log d \log X)$ time. ∎

Remark 3.14. In practice, we obtain a speedup by a factor of 2 in Algorithm 2 by working over $S_d \cap (0, \frac{1}{2}]$ in Steps 1–4. In Step 5, if $b > \frac{d}{2}$, in light of (3.4) we may set $c_{\gamma,p} := (-1)^{\lceil \gamma p \rceil} c_{1-\gamma,p}^{-1}$ and $s_{\gamma,p}(x) := -s_{1-\gamma,p}(-x)$.

Remark 3.15. Note that computing $H_{1,\gamma}(p) \pmod{p^e}$ via Example 2.4 yields $(\lceil \gamma p \rceil - 1)! \pmod{p^e}$ as a byproduct. In practice, in Step 2 of Algorithm 1 we compute $H_{1,1}$ mod p^e rather than p^{e-1}, and we skip Step 1. We similarly modify Steps 2 and 3 of Algorithm 2 when $e > 1$.

4. The Beukers–Cohen–Mellit trace formula

We summarize [CKR20, §2.2] primarily for the purpose of setting notation.

Definition 4.1. A *hypergeometric datum* is a pair of disjoint tuples $\alpha = (\alpha_1, \dots, \alpha_r)$ and $\beta = (\beta_1, \dots, \beta_r)$ valued in $\mathbb{Q} \cap [0, 1)$. Such a pair is *Galois-stable* (or *balanced*) if any two reduced fractions with the same denominator occur with the same multiplicity.

For the rest of the paper, fix a Galois-stable[4] hypergeometric datum α, β and some $z \in \mathbb{Q} \setminus \{0,1\}$. We say that a prime p is *wild* if it divides the denominator of some α_j or β_j; *tame* if it is not wild but divides the numerator or denominator of z or the numerator of $z-1$; and *good* otherwise.

Definition 4.2. The *zigzag function* $Z_{\alpha,\beta} : [0,1] \to \mathbb{Z}$ is defined by

$$Z_{\alpha,\beta}(x) := \#\{j \colon \alpha_j \leq x\} - \#\{j \colon \beta_j \leq x\}.$$

In terms of the zigzag function, the *minimal motivic weight* is given by

$$\begin{aligned} w &= \max\{Z_{\alpha,\beta}(x) \colon x \in [0,1]\} - \min\{Z_{\alpha,\beta}(x) \colon x \in [0,1]\} - 1 \\ &= \max\{Z_{\alpha,\beta}(x) \colon x \in \alpha\} - \min\{Z_{\alpha,\beta}(x) \colon x \in \beta\} - 1. \end{aligned} \tag{4.3}$$

Write $\{x\} := x - \lfloor x \rfloor$ for the fractional part of $x \in \mathbb{Q}$ and $\#S$ for the cardinality of a set S. Set

$$\eta_m(x_1, \dots, x_r) := \sum_{j=1}^{r} \left(\left\{ x_j + \tfrac{m}{1-p} \right\} - \{x_j\} \right), \tag{4.4}$$

$$\xi_m(\beta) := \#\{j : \beta_j = 0\} - \# \left\{ j : \beta_j + \tfrac{m}{1-p} = 0 \right\}, \tag{4.5}$$

$$D := \tfrac{1}{2}(w + 1 - \#\{j : \beta_j = 0\}). \tag{4.6}$$

Definition 4.7. For p prime, define a p-adic analogue of the Pochhammer symbol by setting

$$(x)^*_m := \frac{\Gamma_p\left(\left\{ x + \frac{m}{1-p} \right\} \right)}{\Gamma_p(\{x\})}. \tag{4.8}$$

Let $[z] \in \mathbb{Z}_p$ be the unique $(p-1)$-st root of unity congruent to z modulo p; note that this should not be confused with the notation $f^{[h]}$ defined in Notation 2.7. As in [Wat15, § 2], for p good we write

$$H_p \left(\begin{matrix} \alpha \\ \beta \end{matrix} \middle| z \right) := \frac{1}{1-p} \sum_{m=0}^{p-2} (-p)^{\eta_m(\alpha) - \eta_m(\beta)} p^{D + \xi_m(\beta)} \left(\prod_{\gamma \in \beta}^{\gamma \in \alpha} (\gamma)^*_m \right) [z]^m, \tag{4.9}$$

where $\prod_{\gamma \in \beta}^{\gamma \in \alpha} f(\gamma)$ is shorthand for $\prod_{j=1}^{r} \frac{f(\alpha_j)}{f(\beta_j)}$.

By combining [BCM15, Theorem 1.5] (an adaptation of [Kat90, §8.2]) with the Gross–Koblitz formula [Rob00, §VII.2.6] as described in [Wat15], one establishes the following.

Theorem 4.10. *Assume*[5] *that* $0 \notin \alpha$. *Then there exists a motive* $M_z^{\alpha,\beta}$ *over* $\mathbb{Q}$, *pure of weight* w *and dimension* r, *such that for each good prime* p, $M_z^{\alpha,\beta}$ *has good reduction at* p *and*

$$H_p \left(\begin{matrix} \alpha \\ \beta \end{matrix} \middle| z \right) = \mathrm{Tr}(\mathrm{Frob} \,|\, M_z^{\alpha,\beta}) \in \mathbb{Z} \cap [-rp^{w/2}, rp^{w/2}].$$

Remark 4.11. The bound on $H_p \left(\begin{matrix} \alpha \\ \beta \end{matrix} \middle| z \right)$ implies that for $p > 4r^2$, $H_p \left(\begin{matrix} \alpha \\ \beta \end{matrix} \middle| z \right)$ is determined by its reduction modulo p^e for $e = \lceil (w+1)/2 \rceil$.

[4] Without the Galois-stable condition, much of this discussion carries over, but the resulting motives are defined not over $\mathbb{Q}$ but some cyclotomic field.

[5] This point was neglected in [CKR20].

Remark 4.12. The definition of $M_z^{\alpha,\beta}$ does not itself require $0 \notin \alpha$, only the validity of the trace formula as written. In general, there is an isomorphism $M_z^{\alpha,\beta} \cong M_{1/z}^{\beta,\alpha}$, which in the case $0 \notin \alpha, \beta$ corresponds to a symmetry in (4.9): the substitution $[z] \mapsto [1/z]$ carries the summand indexed by m to the summand indexed by $p-1-m$. The term-by-term equality can be seen from (3.3), taking care about signs.

When $0 \in \alpha$, we currently compute Frobenius traces by applying Theorem 4.10 to $M_{1/z}^{\beta,\alpha}$. It would be preferable to adapt (4.9) to handle this case directly, so as to free up the swap of α and β for other uses (see Remark 5.45).

5. Computation of hypergeometric traces

We next exhibit an algorithm (Algorithm 3) that, on input of α, β, X, e, computes $H_p\left(\begin{matrix}\alpha \\ \beta\end{matrix}\middle| z\right) \pmod{p^e}$ for all primes $p \leq X$ excluding tame[6] and wild primes. The complexity analysis of this algorithm will yield Theorem 1.1; see §5.7.

It is harmless to also exclude a finite set of small good primes, as they can be handled easily by directly computing (4.9) modulo a suitable power of p. We will restrict attention to p with

$$\max\{e, d(d-1)\} < p \leq X, \tag{5.1}$$

where d is the maximum of the denominators of $\alpha \cup \beta$.

To help navigate some heavy notation, we summarize it in the following table.

Symbol	Reference	Symbol	Reference	Symbol	Reference
$f^{[h]}$	Notation 2.7	$\Gamma_p(x)$	Def. 3.1	$\omega(x)$	(3.3)
α, β, r, z	Def. 4.1	$\{x\}, w, \eta_m, \xi_m, D$	Def. 4.2	H_p, $[z]$, $\prod_{\gamma\in\beta}^{\gamma\in\alpha}$	Def. 4.7
a_i, b_i, r_i, c	(5.8)	$f_{i,c}(k), g_{i,c}(k)$	(5.24)	P_m, P'_m	(5.2)
$A_{i,c}(k)$	(5.39)	γ_i, m_i	(5.3)	$Q_{h_1,h_2}(k)$	(5.38)
$c_{i,h}(p)$	(5.32)	$\gamma_{i,c}$	(5.9)	$R_i(x)$	(5.30)
δ_{h_1,h_2}	(5.39)	$h_c(\gamma, \gamma_i)$	(5.10)	σ_i, τ_i	(5.4)
$\epsilon_c(\gamma, \gamma_i)$	(5.13)	$\iota(x, y)$	(5.7)	$S_i(p)$	(5.22)
$e_i, e'_i, \overline{\sigma}_i, \overline{\tau}_i$	(5.5)	k, m	(5.11)		

5.1. **Breaking the sum into ranges.** We start with a high-level breakdown of the algorithm, in which e plays only a minor role. For $m = 0, \dots, p-2$, define

$$P_m := [z]^m \prod_{\gamma\in\beta}^{\gamma\in\alpha} (\gamma)^*_m, \qquad P'_m := (-p)^{\eta_m(\alpha)-\eta_m(\beta)} p^{D+\xi_m(\beta)} P_m. \tag{5.2}$$

Let $0 = \gamma_0 < \cdots < \gamma_s = 1$ be the distinct elements in $\alpha \cup \beta \cup \{0, 1\}$. Write m_i for $\lfloor \gamma_i(p-1) \rfloor$; by (5.1), we also have

$$0 = m_0 < \cdots < m_s = p-1. \tag{5.3}$$

By [CKR20, Lemma 4.2], there exist integers σ_i, τ_i for $0 \leq i \leq s-1$ such that

$$\frac{P'_m}{P_m} = \begin{cases} \tau_i & \text{if } m = m_i, \\ \sigma_i & \text{if } m_i < m < m_{i+1}; \end{cases} \tag{5.4}$$

[6] We can also handle tame primes where $z \in \mathbb{Z}_p^\times$. In particular, we can take $z = 1$.

in fact we can choose $e_i, e'_i \in \{1, \dots, e\}$ and $\overline{\sigma}_i, \overline{\tau}_i \in \{-1, 0, 1\}$ such that for all p,

$$\sigma_i \equiv p^{e-e_i}\overline{\sigma}_i \pmod{p^e}, \qquad \tau_i \equiv p^{e-e'_i}\overline{\tau}_i \pmod{p^e}. \tag{5.5}$$

In this notation, we summarize the method in Algorithm 3.

Algorithm 3: Computation of $H_p\left(\begin{matrix}\alpha \\ \beta\end{matrix}\middle| z\right)$ for good p satisfying (5.1)

1. Apply Algorithm 2 to each d dividing the common denominator of γ_i and γ_j for some $i, j \in \{0, \dots, s\}$ (not necessarily distinct).
2. For each $i \in \{0, \dots, s-1\}$ for which $\overline{\tau}_i \neq 0$, for each p, compute $\overline{\tau}_i P_{m_i} \pmod{p^{e'_i}}$ as indicated in §5.3.
3. For each $i \in \{0, \dots, s-1\}$ for which $\overline{\sigma}_i \neq 0$, for each p, compute the quantities $c_{i,h}(p) \pmod{p^{e_i - h}}$ as indicated in (5.32).
4. For each $i \in \{0, \dots, s-1\}$ for which $\overline{\sigma}_i \neq 0$, compute $\overline{\sigma}_i \sum_{m=m_i+1}^{m_{i+1}-1} P_m \pmod{p^{e_i}}$ in terms of the quantities $c_{i,h}(p)$ as indicated in §5.6.
5. For each p, compute $H_p\left(\begin{matrix}\alpha \\ \beta\end{matrix}\middle| z\right) \pmod{p^e}$ by rewriting (4.9) in the form

$$H_p\left(\begin{matrix}\alpha \\ \beta\end{matrix}\middle| z\right) \equiv \frac{1}{1-p}\sum_{i=0}^{s-1}\left(p^{e-e'_i}\overline{\tau}_i P_{m_i} + p^{e-e_i}\overline{\sigma}_i \sum_{m=m_i+1}^{m_{i+1}-1} P_m\right) \pmod{p^e}. \tag{5.6}$$

5.2. **Residue classes.** We next separate primes into residue classes modulo the denominator of γ_i. Following [CKR20, Lemma 4.1], define

$$\iota(x, y) := \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{if } x > y. \end{cases} \tag{5.7}$$

Write $\gamma_i = \frac{a_i}{b_i}$ and define an integer $r_i \in \{0, \dots, b_i - 1\}$ by

$$a_i(p-1) = m_i b_i + r_i; \tag{5.8}$$

for $p \equiv c \pmod{b_i}$ with $c \in (\mathbb{Z}/b_i\mathbb{Z})^\times$, r_i is the residue of $a_i(c-1)$ mod b_i. We have

$$m_i = \gamma_i(p-1) - \gamma_{i,c}, \qquad \gamma_{i,c} := \tfrac{r_i}{b_i}. \tag{5.9}$$

For the remainder of §5, we fix an index $i \in \{0, \dots, s-1\}$ and a quantity $c \in (\mathbb{Z}/b_i\mathbb{Z})^\times$, and limit attention to primes $p \equiv c \pmod{b_i}$.

For $\gamma \in \alpha \cup \beta$, we analyze $\left\{\gamma + \frac{m}{1-p}\right\}$ in terms of

$$h_c(\gamma, \gamma_i) := \gamma - \gamma_i + \iota(\gamma, \gamma_i) - \gamma_{i,c} \in (-1, 1]. \tag{5.10}$$

For $m = m_i + k$ with $1 \leq k \leq m_{i+1} - m_i$, by [CKR20, (5.11)] we have

$$\begin{aligned} \left\{\gamma + \tfrac{m}{1-p}\right\} &= \gamma + \tfrac{m}{1-p} + \iota(\gamma, \gamma_i) \\ &= h_c(\gamma, \gamma_i) + (k - p\gamma_{i,c})\tfrac{1}{1-p} \\ &= h_c(\gamma, \gamma_i) + k + (k - \gamma_{i,c})\tfrac{p}{1-p}. \end{aligned} \tag{5.11}$$

Submitted to Algor. Num. Th. Symp.

For $k = 0$, we instead have

$$\left\{\gamma + \tfrac{m_i}{1-p}\right\} = h_c(\gamma, \gamma_i) - \tfrac{p}{1-p}\gamma_{i,c} - \epsilon_c(\gamma, \gamma_i) \tag{5.12}$$

$$\text{where } \epsilon_c(\gamma, \gamma_i) := \iota(\gamma, \gamma_i) - \iota(\gamma, \gamma_i - \tfrac{1}{p-1}\gamma_{i,c}) = \begin{cases} 1 & \text{if } \gamma_i = \gamma \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

(We would also have $\epsilon_c(\gamma, \gamma_i) = 1$ if $\gamma_i - \frac{1}{p-1}\gamma_{i,c} < \gamma < \gamma_i$, but this would imply $\lfloor \gamma(p-1) \rfloor = \lfloor \gamma_i \rfloor$ in violation of (5.3).)

We make explicit a point that was elided in [CKR20, Lemma 5.10].

Lemma 5.14. *Let $m_i \leq m < m_{i+1}$ and $\gamma \in \alpha \cup \beta$, and suppose that $\left\{\gamma + \frac{m}{1-p}\right\} \equiv 0 \pmod{p}$. Then either*

$$m = m_i \text{ and } h_c(\gamma, \gamma_i) = \epsilon_c(\gamma, \gamma_i), \text{ or}$$
$$m = m_{i+1} - 1 \text{ and } h_c(\gamma, \gamma_{i+1}) = \epsilon_c(\gamma, \gamma_{i+1}) + 1.$$

Proof. If $m = m_i$, then by (5.12), $\left\{\gamma + \frac{m}{1-p}\right\} \equiv h_c(\gamma, \gamma_i) - \epsilon_c(\gamma, \gamma_i) \pmod{p}$. Since $\epsilon_c(\gamma, \gamma_i) = 1$ implies $\gamma = \gamma_i$ and hence $h_c(\gamma, \gamma_i) = 1 - \gamma_{i,c} \geq 0$, $h_c(\gamma, \gamma_i) - \epsilon_c(\gamma, \gamma_i)$ is in $\mathbb{Q} \cap (-1, 1]$ with denominator at most $d(d-1)$, and so by (5.1) can only be divisible by p if it is zero.

Now assume that $m_i < m < m_{i+1}$ and $\left\{\gamma + \frac{m}{1-p}\right\} \equiv 0 \pmod{p}$. Write $\gamma + \iota(\gamma, \gamma_i) = \frac{a}{d}$ with $0 \leq a < 2d$ and $\gcd(a, d) = 1$. By our hypothesis plus (5.11),

$$\frac{a}{d} = \gamma + \iota(\gamma, \gamma_i) = \left\{\gamma + \tfrac{m}{1-p}\right\} + \frac{m}{p-1}.$$

Thus,

$$a + md = (a - t)p \tag{5.15}$$

for some $t \in \mathbb{Z}_{\geq 0}$, and $t \leq a$ since $a + md$ is non-negative. Furthermore, since every divisor of $a + md$ is prime to d, we see that $\gcd(a - t, d) = 1$, which by Definition 4.1 ensures that $\frac{a-t}{d} = \gamma_j + \delta$ for some $\delta \in \{0, 1\}$ and some $j \in \{0, \dots, s\}$. Rewriting (5.15) in the form $(a - t)(p - 1) = md + t$, we get

$$(\gamma_j + \delta)(p - 1) = m + \tfrac{t}{d}.$$

Taking floors on both sides we obtain

$$m = \delta(p - 1) + m_j - \lfloor \tfrac{t}{d} \rfloor.$$

Since $0 \leq t \leq a < 2d$, this gives a contradiction unless $t \geq d$ and either $\delta = 0$, $m = m_{i+1} - 1$, and $m_j = m_{i+1}$ or $\delta = 1$, $m = m_s - 1$, and $m_j = 0$; in either case, (5.3) implies $\gamma_j + \delta = \gamma_{i+1}$. Then (5.15) becomes

$$\gamma + \iota(\gamma, \gamma_i) = \frac{a}{d} = \gamma_{i+1}p - m_{i+1} + 1 = \gamma_{i+1} + \gamma_{i+1,c} + 1.$$

Since $\iota(\gamma, \gamma_i) = \iota(\gamma, \gamma_{i+1}) - \epsilon_c(\gamma, \gamma_{i+1})$, we obtain $h_c(\gamma, \gamma_{i+1}) - \epsilon_c(\gamma, \gamma_{i+1}) = 1$. ∎

5.3. **Computation of P_{m_i}.** We next elaborate Step 2 of Algorithm 3. Similar ideas will also be used in Step 3; see §5.5.

To begin with, compute $z^m \pmod{p^e}$ by repeated squaring. Since $\log[z] = 0$, we may then obtain $[z]^m \pmod{p^e}$ by writing

$$\left(\frac{[z]}{z}\right)^m = \exp\left(\frac{m}{1-p}\log z^{p-1}\right) \equiv \sum_{h=0}^{e-1} \frac{1}{h!}\left(\frac{1}{1-p}\log z^{p-1}\right)^h m^h \pmod{p^e}. \tag{5.16}$$

Next, use the output of Step 1 of Algorithm 3 to recover

$$\prod_{\gamma\in\beta}^{\gamma\in\alpha} \Gamma_p(\gamma) \pmod{p^e} \tag{5.17}$$

and, for each $\gamma \in \alpha \cup \beta$, a constant $c_{i,\gamma,p}$ and a series $s_{i,\gamma,p}(x)$ such that for all $x \equiv 0 \pmod p$,

$$c_{i,\gamma,p} \exp s_{i,\gamma,p}(x) \equiv \Gamma_p(x + \{h_c(\gamma,\gamma_i)\}) \pmod{p^e}. \tag{5.18}$$

Next, use (3.3), Lemma 5.14, and (5.18), keeping in mind that $h_c(\gamma,\gamma_i) - \epsilon_c(\gamma,\gamma_i) \in (-1,1]$, to obtain an analogous representation of

$$\prod_{\gamma\in\beta}^{\gamma\in\alpha} \Gamma_p\left(x + h_c(\gamma,\gamma_i) - \epsilon_c(\gamma,\gamma_i)\right) \pmod{p^e}. \tag{5.19}$$

Finally, compute P_{m_i} as $z^{m_i} \pmod{p^e}$, times (5.16) evaluated at $m = m_i$, times (5.19) evaluated at $x = \gamma_{i,c}\frac{p}{1-p}$, divided by (5.17).

Remark 5.20. For the most part, in practice we make these computations modulo $p^{e_i'}$ rather than p^e. The sole exception is (5.17), which we use again in (5.32). That said, the balanced condition ensures that (5.17) is a fourth root of unity [Rob00, §VII.1.3, Lemma], so it suffices to compute it modulo p.

5.4. **Factorization of the quotient.** Before continuing through the remaining steps of Algorithm 3, we give a high-level description of what these steps are doing, then link this back to [CKR20].

We will define in §5.6 a block triangular matrix $A_{i,c}(k)$ over $\mathbb{Z}[k]$ for which

$$A_{i,c}(1)\cdots A_{i,c}(k) = \begin{pmatrix} \Delta & 0 \\ \Sigma & \Pi \end{pmatrix}, \tag{5.21}$$

where Δ is a scalar matrix, $\Delta^{-1}\Sigma$ "records" $\overline{\sigma}_i \sum_{j=1}^k P_{m_i+j} \pmod{p^{e_i}}$ and $\Delta^{-1}\Pi$ "records" $P_{m_i+k+1} \pmod{p^{e_i}}$ in a sense to be specified later. We then apply Theorem 2.1 and Example 2.2, noting the dependence of $m_{i+1} - m_i$ on p, to compute

$$S_i(p) := A_{i,c}(1)\cdots A_{i,c}(m_{i+1} - m_i - 1) \pmod{p^{e_i}} \tag{5.22}$$

for all p at once, then extract the desired sum from $S_i(p)$ for each p separately. (As in Remark 2.11, we are using the matrix product in a "projective" fashion.)

Let us recall how this was done for $e_i = 1$ in [CKR20]. For $m := m_i + k$, write (5.23)

$$\frac{P_m}{P_{m_i+1}} = [z]^{k-1} \prod_{\gamma\in\beta}^{\gamma\in\alpha} \frac{\Gamma_p\left(\left\{\gamma + \frac{m}{1-p}\right\}\right)}{\Gamma_p\left(\left\{\gamma + \frac{m_i+1}{1-p}\right\}\right)} = [z]^{k-1} \prod_{\gamma\in\beta}^{\gamma\in\alpha} \frac{\Gamma_p\left(h_c(\gamma,\gamma_i) + k + \frac{(k-\gamma_{i,c})p}{1-p}\right)}{\Gamma_p\left(h_c(\gamma,\gamma_i) + 1 + \frac{(1-\gamma_{i,c})p}{1-p}\right)}.$$

As in [CKR20, Definition 5.7], choose a positive integer b to ensure that

$$f_{i,c}(k) := b \prod_{j=1}^{r} (h_c(\alpha_j, \gamma_i) + k), \qquad g_{i,c}(k) := b \prod_{j=1}^{r} (h_c(\beta_j, \gamma_i) + k) \tag{5.24}$$

belong to $\mathbb{Z}[k]$. By (3.3), Lemma 5.14, and (5.23),

$$\frac{P_{m_i+k+1}}{P_{m_i+k}} \equiv z \frac{f_{i,c}(k)}{g_{i,c}(k)} \pmod p \qquad (k = 1, \dots, m_{i+1} - m_i - 1). \tag{5.25}$$

Write $z = \frac{z_f}{z_g}$ in lowest terms, then set

$$A_{i,c}(k) := \begin{pmatrix} z_g g_{i,c}(k) & 0 \\ \overline{\sigma}_i z_g g_{i,c}(k) & z_f f_{i,c}(k) \end{pmatrix} \equiv (\text{scalar}) \begin{pmatrix} 1 & 0 \\ \overline{\sigma}_i & \frac{P_{m+1}}{P_m} \end{pmatrix} \pmod p; \tag{5.26}$$

we then have

$$\overline{\sigma}_i \sum_{m=m_i+1}^{m_{i+1}-1} P_m \equiv P_{m_i+1} \frac{S_i(p)_{21}}{S_i(p)_{11}} \pmod p. \tag{5.27}$$

Remark 5.28. In [CKR20], the matrices $S_i(p)$ are chained together into a single product, interleaved with connecting matrices $T_i(p)$ to account for the summands P'_{m_i}. We originally implemented a similar approach for $e > 1$; while this saves some work at certain stages, it forces some intermediate computations to be done modulo p^e rather than p^{e_i}, and this is disadvantageous especially with regard to memory usage. The present approach also allows more of the work to be treated as a precomputation; see §6.

5.5. **More factorization of the quotient.** To upgrade the previous discussion to handle $e_i > 1$, we make an additional separation of factors in (5.23), in order to decouple the effect of shifting the argument of Γ_p by a multiple of p (accounted for by Step 3 of Algorithm 3, described below) from the effect of shifting the argument by 1 (accounted for by Step 4 of Algorithm 3, described in §5.6).

We first observe that in the ring $(\operatorname{Frac} \mathbb{Z}_p[k])[x]/(x^{e_i})$, we have

$$\frac{f_{i,c}(x+k)}{g_{i,c}(x+k)} = \prod_{\gamma\in\beta}^{\gamma\in\alpha} \frac{\Gamma_p(x + h_c(\gamma,\gamma_i) + k + 1)}{\Gamma_p(x + h_c(\gamma,\gamma_i) + k)}. \tag{5.29}$$

If we then define the power series

$$R_i(x) := \prod_{\gamma\in\beta}^{\gamma\in\alpha} \frac{\Gamma_p\left(x + h_c(\gamma,\gamma_i) + 1\right)}{\Gamma_p\left(h_c(\gamma,\gamma_i) + 1\right)}, \tag{5.30}$$

then we can rewrite (5.23) as

$$\frac{P_{m_i+k}}{P_{m_i+1}} = \left(\tfrac{[z]}{z}\right)^{k-1} \frac{R_i((k-\gamma_{i,c})\,\frac{p}{1-p})}{R_i((1-\gamma_{i,c})\,\frac{p}{1-p})} \cdot \prod_{j=1}^{k-1} z^{\frac{f_{i,c}(x+j)}{g_{i,c}(x+j)}}\Big|_{x=(k-\gamma_{i,c})\frac{p}{1-p}}. \tag{5.31}$$

The terms in (5.31) not involving j depend on k in a usefully simple way: there exist quantities $c_{i,h}(p) \in \mathbb{Z}/p^{e_i-h}\mathbb{Z}$ for $h = 0, \dots, e_i - 1$ such that for all k,

(5.32)
$$\left(\tfrac{[z]}{z}\right)^{k-1} \frac{R_i((k-\gamma_{i,c})\,\frac{p}{1-p})}{R_i((1-\gamma_{i,c})\,\frac{p}{1-p})} P_{m_i+1} \equiv \sum_{h=0}^{e-1} c_{i,h}(p) \left((k-\gamma_{i,c})\tfrac{p}{1-p}\right)^h \pmod{p^{e_i}}.$$

Remark 5.33. In lieu of (5.31) one could try to use the factorization

$$\frac{P_{m_i+k}}{P_{m_i+1}} = \prod_{j=1}^{k-1} z^{\frac{f_{i,c}(x+j)}{g_{i,c}(x+j)}}\Big|_{x=(1-\gamma_{i,c})\frac{p}{1-p}} \cdot \left(\frac{[z]}{z}\right)^{k-1} \cdots, \tag{5.34}$$

but this does not achieve the requisite decoupling; in particular the second factor does not admit a useful representation in terms of k.

Step 3 of Algorithm 3 is to compute the $c_{i,h}(p)$ following the approach of §5.3. First, use (3.3), Lemma 5.14, and (5.18), now noting that $h_c(\gamma, \gamma_i) + 1 \in (0, 2]$, to obtain a representation in the form $c \exp(s(x))$ of

$$\prod_{\gamma \in \beta}^{\gamma \in \alpha} \Gamma_p\left(x + h_c(\gamma, \gamma_i) + 1\right) \pmod{p^{e_i}}. \tag{5.35}$$

Then compute (5.32) as $z^{m_i+1} \pmod{p^{e_i}}$, times (5.16) evaluated at $m = m_i + k$, times (5.35) evaluated at $x = (k - \gamma_{i,c})\frac{p}{1-p}$, divided by (5.17).

5.6. **Form of the matrix product.** We now perform Step 4 of Algorithm 3, using (5.32) to express the desired sum in terms of a matrix product. Rewrite (5.31) as

$$P_{m_i+k} \equiv \sum_{h=0}^{e_i-1} c_{i,h}(p) \left((k - \gamma_{i,c})\frac{p}{1-p}\right)^h \cdot \prod_{j=1}^{k-1} z^{\frac{f_{i,c}(x+j)}{g_{i,c}(x+j)}}\Big|_{x=(k-\gamma_{i,c})\frac{p}{1-p}} \tag{5.36}$$

$$\equiv \sum_{h_1=0}^{e_i-1} \sum_{h_2=h_1}^{e_i-1} c_{i,h_1}(p) Q_{h_1,h_2}(k) \left(\frac{p}{1-p}\right)^{h_2} \pmod{p^{e_i}}, \tag{5.37}$$

where using the notation $f^{[h]}$ from Notation 2.7 we write

$$Q_{h_1,h_2}(k) := (k - \gamma_{i,c})^{h_2} \left(\prod_{j=1}^{k-1} z^{\frac{f_{i,c}(x+j)}{g_{i,c}(x+j)}}\right)^{[h_2-h_1]}. \tag{5.38}$$

This prompts us to define the block matrix $A_{i,c}(k)$ with $e_i \times e_i$ blocks as follows, where δ_{h_1,h_2} is the Kronecker delta and the scalar is chosen to clear denominators (see Remark 5.45):

$$A_{i,c}(k) := (\text{scalar}) \begin{pmatrix} \delta_{h_1,h_2} & 0 \\ \overline{\sigma}_i (k - \gamma_{i,c})^{e_i - h_2} \delta_{h_1,h_2} & \left(z^{\frac{f_{i,c}(x+k)}{g_{i,c}(x+k)}}\right)^{[h_1-h_2]} \end{pmatrix}; \tag{5.39}$$

note that the factors of $k - \gamma_{i,c}$ appear in the bottom left rather than the bottom right. As promised, we apply Theorem 2.1 and Example 2.2 to compute the products $S_i(p)$ as in (5.22); writing $S_i(p)$ as a block matrix in the notation of (5.21), we have

$$(\Delta^{-1}\Sigma)_{h_1,h_2} = \overline{\sigma}_i \sum_{k=1}^{m_{i+1}-m_i-1} Q_{e_i-h_1,e_i-h_2}(k). \tag{5.40}$$

In terms of the $1 \times e_i$ row vectors v, w given by

$$v_j := c_{i,e_i-j}(p), \qquad w_j := \left(\tfrac{p}{1-p}\right)^{e_i-j}, \tag{5.41}$$

by (5.37) we have

$$\overline{\sigma}_i \sum_{m=m_i+1}^{m_{i+1}-1} P_m \equiv \frac{1}{\Delta_{e_i,e_i}} (v\Sigma w^T)_{11}. \tag{5.42}$$

Remark 5.43. As in Remark 2.6, in practice we achieve a constant factor speedup by computing not $S_i(p)$ but $VS_i(p)$ where V is the matrix consisting of the last $e_i + 1$ rows of the $2e_i \times 2e_i$ identity matrix.

Remark 5.44. According to (5.42), for $j = 1, \dots, e_i$ we only need to compute columns j and $e_i + j$ of $S_i(p)$ modulo p^j. However, in practice there seems to be little overhead incurred by computing all of $S_i(p)$ modulo p^{e_i}.

Remark 5.45. The scalar in (5.39) can be bounded by $b_i^{e_i-1} g_{i,c}(k) \operatorname{rad}(g_{i,c}(k))^{e_i-1}$, where $\operatorname{rad}(g_{i,c}(k))$ is the radical of $g_{i,c}(k)$. The latter arises from clearing denominators in the series expansion of $g_{i,c}(x+k)^{-1} \pmod{x^{e_i}}$; e.g., for $e_i = 2$,

$$A_{i,c}(k) = (\text{scalar}) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \overline{\sigma}_i(k-\gamma_{i,c}) & 0 & z\frac{f_{i,c}(k)}{g_{i,c}(k)} & 0 \\ 0 & \overline{\sigma}_i & z\frac{f'_{i,c}(k)}{g_{i,c}(k)} - z\frac{f_{i,c}(k)g'_{i,c}(k)}{g_{i,c}(k)^2} & z\frac{f_{i,c}(k)}{g_{i,c}(k)} \end{pmatrix}. \tag{5.46}$$

It is tempting to try to restructure the computation so that the top left corner of $A_{i,c}(k)$ is used to track the product over $g_{i,c}(x+k)$. This is complicated by the need to sum $(l-\gamma_{i,c})^{h_1} \left(\prod_{j=1}^k z\frac{f_{i,c}(x+j)}{g_{i,c}(x+j)}\right)^{[h_2]}$ for $h_1 > h_2$ to incorporate the $c_{i,h}(p)$; it would also mean retaining all of the rows of the product, rather than only e_i+1 of them as in Remark 5.43.

In any case, the dependence on $\operatorname{rad}(g_{i,c}(k))$ means that our algorithm performs much better in cases where β has many repeated entries. In particular, it is sometimes very profitable to swap α and β when possible (see Remark 4.12).

5.7. **Complexity estimates.** We conclude by analyzing the complexity of Algorithm 3. This will imply Theorem 1.1 by taking $e = \lceil (w+1)/2 \rceil \le r-1$ and invoking Remark 4.11.

We first note that $\varphi(b_i) \le r$ and so[7] $b_i = O(r \log\log r)$. From this we see that the sum of all integers that occur as $\operatorname{lcm}(b_i, b_j)$ for some i,j is $O(r^2(\log\log r)^2)$. Applying Theorem 3.13, we bound the time complexity of Step 1 by

$$O(r^2(\log\log r)^2 e^2 X(\log X)^3 + r^2(\log\log r)^2 e^4(\log r + \log e)X)$$

and the space complexity by

$$O(eX(\log X)^2 + r^2(\log\log r)^2 e^2 X).$$

Steps 2 and 3 include no amortized steps, so their space costs are negligible. For each of $O(X/\log X)$ primes p, we perform $O(r+e^2)$ operations in $\mathbb{Z}/(p^e)$. At a time cost of $O(e^2(\log X)^2)$ apiece per Remark 3.10, this runs to $O((re^2+e^4)X\log X)$.

Step 4 is dominated by $O(r)$ applications of Theorem 2.1 via Example 2.2. Since the matrix $A_{i,c}(k)$ has size $2e_i \times 2e_i$ and its entries have degree $O(e_i r)$ (the factor of e_i coming from Remark 5.45), we bound the time and space costs by

$$O(e^3 r^2 X(\log X)^3) \quad \text{and} \quad O(e^3 r^2 X(\log X)^2)$$

[7] By [RS62, Theorem 15], $\varphi(n) \ge \frac{n}{2\log\log n + 3/\log\log n}$ for $n > 2$.

provided that $e = O(r)$, $r = O(\log X)$, and the bitsize of z is $O(er \log X)$. Combining these estimates (and replacing e with r) yields Theorem 1.1.

6. Implementation notes and sample timings

We have implemented Algorithm 3 in Sage, using Cython for certain inner loops and for a wrapper to Sutherland's rforest C library. While it would be natural to implement a multithreaded approach in certain steps (particularly the calculations not amortized over p), we have not implemented this.

We report timings in Table 1 and divide our algorithm into three phases:

(1) Step 1 of Algorithm 3. This phase is independent of z and depends only mildly on α and β.
(2) Steps 2 and 3 of Algorithm 3, in both cases assuming $z = 1$. This phase includes no amortized steps and is independent of z.
(3) Step 4 of Algorithm 3, plus adjustment of the results of Steps 2 and 3 to account for z.

When possible, we also include comparison timings with the built-in functions for computing $H_p\left(\begin{matrix}\alpha\\ \beta\end{matrix}\middle| z\right)$ in Sage and Magma, denoted "Sage(p)" and "Magma(p)" in the table. These are not amortized, and so these runtimes are quasilinear in X^2 rather than X. These builtin functions also compute Frobenius traces at higher prime powers q (see [CKR20, (2.22)] for the analogous formula), which are needed for tabulation of L-functions when $r > 2$ (see §7); these are denoted "Sage(q)" and "Magma(q)" in the table, and the runtimes are quasilinear in $X^{3/2}$.

For $e = 1$, we also include a comparison with the "chained product" approach of [CKR20] (see Remark 5.28). The two approaches perform comparably for individual calculations, but for bulk calculations Algorithm 3 is clearly superior.

We use the following hypergeometric data (computing the weight w as in (4.3)):

e	r	w	α	β
1	2	1	$(\frac{1}{4}, \frac{3}{4})$	$(\frac{1}{6}, \frac{5}{6})$
1	4	1	$(\frac{1}{10}, \frac{3}{10}, \frac{7}{10}, \frac{9}{10})$	$(\frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6})$
2	4	3	$(\frac{1}{4}, \frac{1}{3}, \frac{2}{3}, \frac{3}{4})$	$(\frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6})$
3	6	5	$(\frac{1}{5}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{5}, \frac{4}{5})$	$(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6})$
4	8	7	$(\frac{1}{5}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{4}{5})$	$(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6}, \frac{5}{6})$

We take the specialization point $z = \frac{314}{159}$; note that in Magma one must input z^{-1} instead of z.

Timings are reported in 5.1GHz Intel i9-12900K core-seconds, running Sage 10.1 and Magma 2.28-5. Memory usage was limited to 20GB.

7. Tabulation of L-functions

As mentioned in the introduction, the broader context for our work is the desire to tabulate hypergeometric L-functions at scale in LMFDB. This problem naturally breaks down as follows.

- For a fixed hypergeometric L-function of motivic weight w, we may compute p-Frobenius traces for small p using (4.9) and for larger p up to some cutoff X using our algorithm, in both cases excluding tame and wild primes.

Table 1. Timings, see §6 for explanation.

$e=1, r=2$	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}	2^{25}
[CKR20]	0.06	0.11	0.21	0.48	1.20	2.84	6.80	16.0	37.6	88.5	226
Phase (1)	0.08	0.13	0.24	0.49	0.98	2.26	8.15	17.9	32.5	65.6	157
Phase (2)	0.03	0.05	0.10	0.19	0.38	0.71	3.20	5.54	9.44	17.1	28.2
Phase (3)	0.04	0.07	0.12	0.24	0.48	1.02	2.30	5.44	11.2	24.8	63.3
Total	0.15	0.25	0.46	0.92	1.84	3.98	13.6	28.9	53.1	107	248
Sage(p)	40.6	150	572								
Magma(p)	36.7	147	604								

$e=1, r=4$	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
[CKR20]	0.10	0.15	0.24	0.39	0.82	1.86	4.28	10.3	24.7	55.0	127
Phase (1)	0.06	0.08	0.11	0.19	0.34	0.66	1.34	3.18	9.94	17.6	38.6
Phase (2)	0.02	0.03	0.06	0.11	0.24	0.45	0.87	1.67	3.42	6.85	16.6
Phase (3)	0.03	0.04	0.07	0.14	0.25	0.54	1.17	2.73	6.32	14.3	31.9
Total	0.11	0.16	0.25	0.44	0.83	1.65	3.38	7.58	19.7	38.7	87.1
Sage(p)	4.33	15.5	56.0	223	879						
Magma(p)	3.11	11.8	44.9	181	749						
Sage(q)	0.03	0.08	0.20	0.46	1.34	3.61	9.54	28.7	69.1	193	534
Magma(q)	0.10	0.28	0.73	1.76	5.62	15.2	39.8	111	325		

$e=2, r=4$	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
Phase (1)	0.12	0.23	0.41	3.28	1.59	3.33	10.7	19.4	42.9	94.8	203
Phase (2)	0.06	0.11	0.21	0.37	0.68	1.27	2.36	4.39	8.57	17.2	37.9
Phase (3)	0.08	0.14	0.21	0.47	1.00	2.19	5.04	12.3	28.6	67.7	147
Total	0.26	0.48	0.84	4.12	3.27	6.79	18.0	36.1	80.0	179	388
Sage(p)	119	469									
Magma(p)	17.3	65.7	454								
Sage(q)	0.11	0.23	0.48	1.40	3.67	9.58	26.1	68.4	188	518	
Magma(q)	0.32	0.86	2.93	10.3	29.2	81.1	249	762			

$e=3, r=6$	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
Phase (1)	0.16	0.30	0.61	1.02	2.12	4.46	10.0	25.0	54.1	123	283
Phase (2)	0.05	0.08	0.16	0.38	0.59	1.26	2.33	4.69	8.67	17.2	33.8
Phase (3)	0.11	0.23	0.32	0.72	1.98	4.60	11.4	27.6	71.4	158	374
Total	0.31	0.62	1.09	2.11	4.69	10.3	23.8	57.3	134	298	691
Sage(p)	54.9	210	793								
Magma(p)	11.2	42.8	162	626	2437						
Sage(q)	0.06	0.12	0.28	0.57	1.64	4.31	11.3	30.7	165	869	2825
Magma(q)	0.27	0.68	1.98	4.37	14.5	37.2	100	291	907	2951	

$e=4, r=8$	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
Phase (1)	0.45	0.72	1.23	2.35	4.86	10.7	23.5	57.9	130	304	697
Phase (2)	0.09	0.17	0.32	0.60	1.24	2.33	4.38	8.38	16.8	32.4	65.5
Phase (3)	0.19	0.40	0.67	1.78	4.47	10.9	26.1	63.9	163	365	865
Total	0.74	1.28	2.22	4.72	10.6	23.9	54.0	130	310	702	1629
Sage(p)	76.5	291	1088								
Magma(p)	12.8	49.4	189	745	3789						
Sage(q)	0.07	0.20	0.31	1.74	7.75	24.6	66.6	184	494		
Magma(q)	0.30	0.81	2.32	5.09	21.2	56.4	155	463			

- For prime powers $q \leq X$, excluding powers of tame and wild primes, as noted in §6 we may use Sage or Magma to compute the q-Frobenius trace. (We may also skip $q = p^f$ for $f > r/2$ by virtue of the local functional equation, but this is a minor point because the dominant case is $f = 2$.) While it is likely possible to to reduce the complexity from $X^{3/2}$ to X either by adapting our present approach (as suggested already in [CKR20]) or implementing the algorithm indicated in [Ked22], the timings in §6 diminish the urgency of this.
- Magma can compute Euler factors and conductor exponents at tame primes (modulo a tractable conjecture). The computational difficulty is negligible.[8]
- For Euler factors and conductor exponents at wild primes, even the conjectural picture remains incomplete, but see [RR22] for a partial description. Given enough Fourier coefficients at good primes, one can empirically verify a complete guess for the conductor, the global root number, and all bad Euler factors using the approximate functional equation, as in [Dok04] or [FKL19].

Work in this direction is ongoing, and we expect to have many examples of hypergeometric L-functions available in LMFDB in the near future.

References

[Ber08] Daniel J. Bernstein. "Fast multiplication and its applications". In: *Algorithmic number theory: lattices, number fields, curves and cryptography*. Vol. 44. Math. Sci. Res. Inst. Publ. Cambridge Univ. Press, Cambridge, 2008, pp. 325–384 (↑ 5).

[BCM15] Frits Beukers, Henri Cohen, and Anton Mellit. "Finite hypergeometric functions". In: *Pure Appl. Math. Q.* 11.4 (2015), pp. 559–589 (↑ 1, 8).

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system. I. The user language". In: *J. Symbolic Comput.* 24.3-4 (1997). Computational algebra and number theory (London, 1993), pp. 235–265 (↑ 2).

[CMTV21] Xavier Caruso, Marc Mezzarobba, Nobuki Takayama, and Tristan Vaccon. "Fast evaluation of some p-adic transcendental functions". https://hal.science/hal-03263044v1. 2021 (↑ 5).

[CGH14] Edgar Costa, Robert Gerbicz, and David Harvey. "A search for Wilson primes". In: *Math. Comp.* 83.290 (2014), pp. 3071–3091 (↑ 2, 3).

[CKR23] Edgar Costa, Kiran Kedlaya, and David Roe. "Amortized Hypergeometric Motives Repository". https://github.com/edgarcosta/amortizedHGM. [Online; accessed 25 February 2020]. 2023 (↑ 2).

[CKR20] Edgar Costa, Kiran S. Kedlaya, and David Roe. "Hypergeometric L-functions in average polynomial time". In: *ANTS XIV—Proceedings of the Fourteenth Algorithmic Number Theory Symposium*. Vol. 4. Open Book Ser. Math. Sci. Publ., Berkeley, CA, 2020, pp. 143–159 (↑ 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18).

[Dok04] Tim Dokchitser. "Computing special values of motivic L-functions". In: *Experiment. Math.* 13.2 (2004), pp. 137–149 (↑ 18).

[8] Note that identifying *all* tame primes requires an integer factorization, but here we only need to know which primes $\leq X$ are tame.

[FKL19] David W. Farmer, Sally Koutsoliotas, and Stefan Lemurell. "Varieties via their L-functions". In: *J. Number Theory* 196 (2019), pp. 364–380 (↑ 18).

[GK79] Benedict H. Gross and Neal Koblitz. "Gauss sums and the p-adic Γ-function". In: *Ann. of Math. (2)* 109.3 (1979), pp. 569–581 (↑ 1).

[Har14] David Harvey. "Counting points on hyperelliptic curves in average polynomial time". In: *Ann. of Math. (2)* 179.2 (2014), pp. 783–803 (↑ 2).

[Har15] David Harvey. "Computing zeta functions of arithmetic schemes". In: *Proc. Lond. Math. Soc. (3)* 111.6 (2015), pp. 1379–1401 (↑ 2).

[HH21] David Harvey and Joris van der Hoeven. "Integer multiplication in time $O(n \log n)$". In: *Ann. of Math. (2)* 193.2 (2021), pp. 563–617 (↑ 3).

[HS14] David Harvey and Andrew V. Sutherland. "Computing Hasse-Witt matrices of hyperelliptic curves in average polynomial time". In: *LMS J. Comput. Math.* 17.suppl. A (2014), pp. 257–273 (↑ 2, 3).

[HS16] David Harvey and Andrew V. Sutherland. "Computing Hasse-Witt matrices of hyperelliptic curves in average polynomial time, II". In: *Frobenius distributions: Lang-Trotter and Sato-Tate conjectures*. Vol. 663. Contemp. Math. Amer. Math. Soc., Providence, RI, 2016, pp. 127–147 (↑ 2, 3).

[HLOP22] Yang-Hui He, Kyu-Hwan Lee, Thomas Oliver, and Alexey Pozdnyakov. "Murmurations of elliptic curves". arXiv:2204.10140v1. 2022 (↑ 2).

[Kat90] Nicholas M. Katz. "Exponential sums and differential equations". Vol. 124. Annals of Mathematics Studies. Princeton University Press, Princeton, NJ, 1990, pp. xii+430 (↑ 8).

[Ked22] Kiran S. Kedlaya. "Frobenius structures on hypergeometric equations". In: *Arithmetic, geometry, cryptography, and coding theory 2021*. Vol. 779. Contemp. Math. Amer. Math. Soc., [Providence], RI, 2022, pp. 133–158 (↑ 18).

[LOP24] Kyu-Hwan Lee, Thomas Oliver, and Alexey Pozdnyakov. "Murmurations of Dirichlet characters". arXiv:2307.00256v3. 2024 (↑ 2).

[LMFDB] The LMFDB Collaboration. "The L-functions and Modular Forms Database". `http://www.lmfdb.org`. [Online; accessed 25 January 2024]. 2020 (↑ 2).

[Rob00] Alain M. Robert. "A course in p-adic analysis". Vol. 198. Graduate Texts in Mathematics. Springer-Verlag, New York, 2000, pp. xvi+437 (↑ 5, 6, 8, 12).

[RR22] David P. Roberts and Fernando Rodriguez Villegas. "Hypergeometric motives". In: *Notices Amer. Math. Soc.* 69.6 (2022), pp. 914–929 (↑ 1, 18).

[Rod07] Fernando Rodriguez Villegas. "Experimental number theory". Vol. 13. Oxford Graduate Texts in Mathematics. Oxford University Press, Oxford, 2007, pp. xvi+214 (↑ 5).

[RS62] J. Barkley Rosser and Lowell Schoenfeld. "Approximate formulas for some functions of prime numbers". In: *Illinois J. Math.* 6 (1962), pp. 64–94 (↑ 15).

[Sag24] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.1)*. http://www.sagemath.org. 2024 (↑ 2).

[Sch82] Arnold Schönhage. "Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients". In: *Computer algebra (Marseille, 1982)*. Vol. 144. Lecture Notes in Comput. Sci. Springer, Berlin-New York, 1982, pp. 3–15 (↑ 4).

[Wat15] Mark Watkins. "Hypergeometric motives over $\mathbb{Q}$ and their L-functions". http://magma.maths.usyd.edu.au/~watkins/papers/known.pdf. [Online; accessed 25 January 2024]. 2015 (↑ 8).