Algorithmic Number Theory Symposium XVII, Groningen, July 6-10, 2026

NEWTON STRATA REALIZATION FOR HYPERSURFACES VIA EXPLICIT p-ADIC COHOMOLOGY

RYAN BATUBARA, JACK J GARZELLA, YONGYUAN HUANG,
AND MAXIMUS MELLBERG

Abstract. Let X be a smooth projective hypersurface over a finite field k of characteristic p. We address the problem of practically computing the zeta function $Z(X,T)$ of X (equivalently, the point counts $\#X(\mathbb{F}_q)$, where $q=p^n$), and we focus on the case when $7 \leq p < 50$. We use the theoretical framework of the variant of Kedlaya's algorithm in [AKR10], and we use the technique of *controlled reduction* as described in [Cos15]. We define an optimization problem that abstracts the key bottleneck in the implementation of controlled reduction. An algorithm that solves this problem is called a *reduction policy*. We present three reduction policies with different advantages and disadvantages. We also present a high-performance implementation of controlled reduction that contains GPU-optimized linear algebra code and a data structure for linear recurrences that the authors hope can be used to study further reduction policies. Our algorithms get state-of-the-art performance in many cases; for example, we beat [Tui16] or [Kyn22] on many examples of quintic curves, while also being able to compute zeta functions of cubic fourfolds when $p = 7, 11$. We also have the first (to our knowledge) systematic computations of zeta functions of quintic surfaces. We use our implementation to deduce many new explicit examples of varieties with specified Newton polygons, including cubic fourfolds which are neither ordinary nor supersingular, quartic K3 surfaces of various Artin-Mazur heights, and quintic surfaces of all possible domino numbers.

1. Introduction

1.1. **Background.** Let $k = \mathbb{F}_q$ and X/k be a smooth proper k-scheme of finite type of dimension n. The *zeta function* of X is the generating series of the number of $\mathbb{F}_{q^r}$–points on X defined by

$$Z(X,T) \coloneqq \exp\left(\sum_{r=1}^{\infty} \frac{T^r}{r} \#X(\mathbb{F}_{q^r})\right);$$

it is an invariant of X of substantial interest in number theory and arithmetic geometry. The well-celebrated Weil conjectures [Wei49] proven by Dwork–Grothendieck–Deligne [Dwo60, Gro66, Del74, Del80] encapsulate many key properties of $Z(X,T)$. For example, $Z(X,T)$ is a rational function of the form $Z(X,T) = \prod_{i=0}^{2n} P_i(x)^{(-1)^{i+1}}$ for $P_i \in \mathbb{Z}[T]$. Let $P_i = \sum_j a_{ij} x^j$. We define the *i-th Newton polygon* $\mathrm{newt}_i(X)$ of X to be the lower convex hull of the points $(j, v_p(a_{ij}))$; it is another invariant of X which is coarser than its zeta function but also of wide interest. In many cases (such as when X is a curve, abelian variety, or projective hypersurface), there is only one i for which the factor P_i is nontrivial, in which case we say that $\mathrm{newt}_i(X)$

Date: June 5, 2026.

is *the* Newton polygon of X and denote it by $\mathrm{newt}(X)$. A celebrated theorem of Mazur [Maz73] gives a lower bound for $\mathrm{newt}_i(X)$ in terms of another lower convex hull, which is called the *Hodge polygon* and is denoted $\mathrm{hodge}(X)$.

1.2. **Motivation.** We recall some of the reasons for interest in $\mathrm{newt}(X)$.

1.2.1. *Other invariants of X.* Many other arithmetic invariants of X are encoded in $\mathrm{newt}(X)$. If X is a curve of genus g or abelian variety of dimension g, the *p-rank* of X is exactly the length of the slope zero segment of $\mathrm{newt}(X)$. Furthermore, X is called *ordinary* if its Newton polygon equals its Hodge polygon, and *almost ordinary* if it has the (in this case, unique) lowest Newton polygon that is not ordinary. In fact, being ordinary is equivalent to having p-rank g, and being almost ordinary is equivalent to having p-rank $g-1$. If X is a K3 surface, then by Mazur's theorem and the Weil conjectures, the first slope of the Newton polygon of X must be equal to $1-\frac{1}{h}$, for $h \in \{1 \dots 10\} \cup \infty$. This h is called the *Artin-Mazur height* of X [AM77]. More generally, if X is any surface, we may characterize its *domino number* [JR03, Jos20] as the smallest vertical distance between its Newton polygon and its Hodge polygon. We say that X is *Hodge-Witt* if its domino number is zero. For example, a K3 surface is not Hodge-Witt if and only if it has height ∞. Note that many of the aforementioned notions first appeared with other definitions, but the Newton polygon provides a clean way to discuss all of them in a unified way.

These constructions can be extended to other families of varieties. To define an analogue of the p-rank for X a cubic threefold, we must consider the length of the slope one segment of $\mathrm{newt}(X)$; the p-rank of a cubic threefold coincides with the p-rank of its intermediate Jacobian [CG72]. For X a cubic fourfold, we may define a "height" of X such that the first slope of the Newton polygon is $2-\frac{1}{h}$, where $h \in \{1 \dots 10\} \cup \infty$. We call this the *Newton height* of the cubic fourfold. One of the key motivations for this work is to lay the groundwork for the development of the theory of heights of cubic fourfolds.

1.2.2. *F-singularities and cubic hypersurfaces.* Newton polygons also carry information about singularities. We refer the reader to [MP] for an overview of the theory of F-singularities. We say a ring R of characteristic p is *F-split* if the Frobenius map $F\colon R \to R$ splits in the category of R-modules. F-splitness is a "mild singularities" condition; that is, if R is regular, then R is F-split, while the converse does not hold [MP, §1-2]. We say that the scheme $\operatorname{Spec} R$ is F-split if R is. If X is a smooth projective K3 surface, that is, $\omega_X \cong \mathcal{O}_X$, we have:

Proposition 1.1. *Choose some R such that $X = \operatorname{Proj} R$. Let h be the Artin-Mazur height of X. Then the following are equivalent:*

(i) $X = \operatorname{Proj} R$ *is ordinary,*
(ii) $h = 1$*, and*
(iii) $\operatorname{Spec} R$ *is F-split.*

We see that the Newton polygon of X gives information about the F-singularities of (any) affine cone over that variety; we may ask for analogues of Proposition 1.1 for other families of hypersurfaces. Recently, Kawakami and Witaszek [KW24b] have introduced the notion of *higher F-injectivity*, which gives a natural analogue of F-splitness. The following is a folklore question, first asked by Witaszek:

Question 1.2. *Let X be a smooth projective cubic hypersurface of dimension $d \geq 3$, and choose R such that $X = \operatorname{Proj} R$. Let h be the Newton height of X. Are the following equivalent?*

(i) $X = \operatorname{Proj} R$ *is ordinary;*
(ii) $h = 1$*; and*
(iii) $\operatorname{Spec} R$ *is* $1 - F$*-injective in the sense of Kawakami and Witaszek.*

Motivated by Question 1.2 and the desire to test conjectures with examples, we ask the following much simpler question.

Question 1.3. *Can we produce explicit examples of cubic threefolds (resp. cubic fourfolds) X which have interesting Newton polygons? Especially, can we find explicit examples which are ordinary, and examples which have Newton polygons that are as close to ordinary as possible without being ordinary?*

Question 1.3 may be called the *Newton strata realization problem*, a reference to the fact that Newton polygons stratify families of varieties in characteristic p (see [dJO00]). The purpose of this paper is to provide a general (computational) methodology for answering questions such as Question 1.3. See Section 6.6 for the prior work on this question for cubic fourfolds.

1.3. **Kedlaya-type algorithms.** One immediate way of accessing the Newton polygons of X is via explicitly computing $Z(X, T)$. Given an arbitrary smooth projective k-scheme X of finite type, explicitly computing $Z(X, T)$ remains an open problem in general. There has been a myriad of papers written on computing the zeta functions of varieties of particular types, most of which use cohomological methods. For example, the work of Schoof [Sch95] may be interpreted as using mod-ℓ étale cohomology, the work of Sperber-Voight [SV13] uses Dwork cohomology, and the work of Harvey, Sutherland, Costa, and others [HS14, HS16a, HMS16, Sut20, AFP22, CHS22] uses Čech cohomology (via formulas for the Hasse-Witt/Cartier-Manin matrix). The landmark work of [Ked01] for hyperelliptic curves initiated the study of computations of $Z(X, T)$ using rigid-cohomological methods. Kedlaya's algorithm has since been generalized and improved in various ways. We highlight a few relevant developments. In [AKR10], Abbott, Kedlaya, and Roe generalize Kedlaya's algorithm to projective hypersurfaces. Harvey [Har07] then improves the complexity in p to $p^{1/2}$ using a technique known as *controlled reduction* in the literature. Costa and Harvey [CH, Cos15] provide algorithms and an implementation for computing $Z(X, T)$ in the case of projective hypersurfaces; Costa, Harvey, and Kedlaya [CHK19] further generalize these algorithms to hypersurfaces in toric varieties, under the condition that the equations of these varieties satisfy a nondegeneracy condition (see Definition 3.1). Such *Kedlaya-type* algorithms remain the state of the art for the computation of $Z(X, T)$ when $\dim X > 1$.

1.4. **Anatomy of a Kedlaya-type algorithm.** Let $H^i_{rig}(X)$ denote the rigid cohomology groups of X. Each Kedlaya-type algorithm works to compute the action of Frob_q on $H^i_{rig}(X)$ by the standard method of linear algebra: write an explicit basis for $H^i_{rig}(X)$, write a formula for the action of Frob_q on an arbitrary element $\omega \in H^i_{rig}(X)$, compute the action of Frob_q on each of the basis vectors, and write $\operatorname{Frob}_q(\omega)$ as a p-adic linear combination of the basis elements using cohomological relations via some sort of a reduction algorithm.

There are two main difficulties in any Kedlaya-type algorithms:

(a) The formula for the action of Frob_q on $H^i_{rig}(X)$ is an overconvergent p-adic power series with infinitely many nonzero terms. One may apply the reduction algorithm to each term in the series, but a computer can only do this for a finite subset of the terms.
(b) Applying the cohomological relations naively involves a lot of polynomial arithmetic, which makes the algorithm impractical in most cases.

Section 2.5 summarizes the approach of [AKR10] to address the first problem. With regards to the second problem, Harvey and Costa in [Har07, Cos15, CH] pioneered the method of *controlled reduction*, which we recall in Section 3.1.

1.5. **Our contributions.** Let X be a smooth projective hypersurface. We answer questions akin to Question 1.3 by developing and implementing fast algorithms for calculating $Z(X,T)$ and deduce $\mathrm{newt}(X)$ from the zeta function. In particular, we use the machinery of *controlled reduction* due to Harvey and Costa. Previous work on Kedlaya-type algorithms and controlled reduction has focused on the complexity properties of the algorithms involved. We take a different perspective and instead focus on practicalities in the implementation. In particular, the bottleneck of most controlled reduction algorithms is a series of matrix-vector multiplications which take place over $\mathbb{Z}/p^M\mathbb{Z}$ for some "precision" M. This is also a bottleneck for practical implementations of [Har15], which is theoretically very different but algorithmically similar. Much work has gone into making the size of these matrices as small as possible and to computing $Z(X,T)$ for very large primes p [HS14, CHK19]. Furthermore, prior implementations are typically tailored to the specific variety under study, making them difficult to reuse for other inputs.

In this work, for X a projective hypersurface, we describe an algorithm to compute $Z(X,T)$ that is equivalent to controlled reduction as presented in [Cos15, Proposition 1.15]. However, we give a mathematical formalism for a key implementation detail of controlled reduction algorithms, which we call the *reduction policy*. We describe the choices in Costa's implementations [Cosa, CHK], and provide two reduction policies which perform better on our examples; one which is generic, and one which is suited to hypersurfaces of low degree. We also develop a data structure which encapsulates the functionality necessary for another important implementation detail, called a *linear recurrence of matrices*. We provide several implementations, each suited to various X. Finally, we provide a few GPU-accelerated linear algebra utilities, including a matrix multiplication modulo p^m for $p^m < 2^{25}$ (which is a wrapper around Nvidia's CUBLAS, similar to the Magma implementation), and a matrix multiplication modulo p^m for $p^m < 2^{53}$, which uses Karatsuba's multiplication algorithm.

All of our results are obtained on small computers, each with no more than 32 GB of memory. Furthermore, given the motivations in F-singularities, we focus on medium-low characteristic, whereas previous work have mostly focused on optimizing performance for large primes; in this paper, all computations have $7 \leq p < 50$, though our algorithms often work beyond these cases. In particular, we achieve the first systematic computations (to the knowledge of the authors) on cubic fourfolds in characteristics 7 and 11, and quintic surfaces in small characteristics. We also achieve state-of-the-art or competitive performance across cubic threefolds, quartic and quintic surfaces, and smooth plane quintics; see Section 7 for details.

With this performance, our main results are as follows.

Theorem 1.4. *There exist*

(1) *cubic fourfolds with height 2 in characteristic* $p = 7, 11$,
(2) *cubic threefolds that are ordinary and almost ordinary in characteristics* $p = 7, 11, 13$,
(3) *quartic K3 surfaces of height up to 4 in characteristic* $p = 17, 19, 23$,
(4) *quintic surfaces of all possible domino numbers in characteristic* $p = 7, 13$,
(5) *plane quintics of all possible p-ranks for* $p = 7, 11, 13$,
(6) *plane sextics of all p-ranks* ≥ 6 *for* $p = 7, 11, 13$, *and*
(7) *plane heptics which are ordinary and almost ordinary for* $p = 7, 11, 13, 17, 19, 23, 31$.

Our proof is constructive; in particular, we have explicit examples for all of the above and we have calculated the full zeta function for each of our examples. All examples can be found in [BGHM26b].

We implement our algorihtms in the Julia programming language, making use of the OSCAR computer algebra system [OSC26, DEF+25], especially Nemo [FHHJ17] and FLINT [FLI26]. The algorithms described in this article as well as computations verifying Theorem 1.4 is available at [BGHM26a] and [BGM26].

1.6. **Contents.** We conclude this introduction with an outline of the paper. After recalling the theoretical background for computing the zeta function of smooth projective hypersurfaces using explicit p-adic cohomological methods in Section 2, we dissect the components of controlled reduction in Section 3 and formally define an optimization problem that models controlled reduction in Section 3.4. We devote Section 4 to describe three different reduction policies and Section 5 to present the GPU-accelerated linear algebra algorithms and data structure for computing linear recurrence of matrices that we implement. Section 6 demonstrates the practical application of our work in finding varieties with Newton polygons previously unknown in the literature, and Section 7 compares the performance of our algorithms to existing implementations.

1.7. **Acknowledgements.** The authors thank Simon Branhorst, Edgar Costa, Max Horn, and Kiran S. Kedlaya for helpful conversations and feedback, Youran Geng, Devanshi Jain, and Thomas Spraling for preliminary work during an early stage of this project, Daniel Bragg for a correction on his work with Perry, Jeff Achter for pointing out the argument in Section 6.5, and the UCSD research cluster for use of their machines. The second author was partially supported by NSF Graduate Research Fellowship Grant No. 2038238 and a fellowship from the Sloan Foundation.

2. Computing the Zeta Functions of Projective Hypersurfaces

We recall the necessary theoretical background (see, e.g., [AKR10, §2] and the references therein for further details) and establish our notations.

2.1. **Some p-adic cohomology theories.** Suppose $q = p^s$ for some prime number p. Let $\mathbb{Q}_q$ be the unramified extension of $\mathbb{Q}_p$ of degree n, and $\mathbb{Z}_q$ be the ring of integers of $\mathbb{Q}_q$. Let $\mathcal{X} = \mathbf{P}^n_k$, and $\mathcal{Z}$ be a smooth projective hypersurface of degree d in $\mathcal{X}$ defined by a homogeneous polynomial $f_{\mathcal{Z}} \in k[x_0, \ldots, x_n]_d$. Berthelot [Ber96, Ber97] has developed a Weil cohomology theory known as *rigid cohomology*, which contravariantly associates finite dimensional $\mathbb{Q}_q$-vector spaces $H^i_{\mathrm{rig}}(\cdot)$ to a variety. Let $\mathcal{U} = \mathcal{X} \backslash \mathcal{Z}$. For $m \geq 1$, let Homog_m denote the free $\mathbb{Z}_q$-module

of homogeneous polynomials in $\mathbb{Z}_q[x_0, \ldots, x_n]$ of degree m; by convention, we let $\mathrm{Homog}_0 = \{1\}$ and $\mathrm{Homog}_m = \emptyset$ for $m < 0$. After choosing an arbitrary lift $f \in \mathrm{Homog}_d$ of $f_{\mathcal{Z}}$, let Z be the hypersurface cut out by f in $X = \mathbf{P}^n_{\mathbb{Z}_q}$ and define $U = X \backslash Z$. For smooth projective hypersurfaces, the existence of such a lift is guaranteed by lifting the coefficients of $f_{\mathcal{Z}}$.

Proposition 2.1. *With notation as above, let* $Q(T) = \det(1 - q^{-1} T \,\mathrm{Frob}_q \,|H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q}))$. *Then*

$$Z(\mathcal{Z}, T) = \frac{Q(T)^{(-1)^n}}{\prod_{i=0}^{n-1}(1 - q^i T)}.$$

Proof. See [Cos15, §1.2.1] for a concise proof and [AKR10, §2] for details of p-adic cohomological calculations. □

Hence it suffices to compute $Q(T)$ in order to compute the zeta functions of smooth projective hypersurfaces.

2.2. **Griffiths-Dwork Construction.** [AKR10, §3.2], based on Griffiths's work [Gri69, Page 469-472], gives an explicit construction of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ as follows. Let $\Omega \coloneqq \sum_{i=0}^n (-1)^i x_i dx_0 \wedge \cdots \wedge \widehat{dx_i} \wedge \cdots \wedge dx_n$, where $\widehat{dx_i}$ means the dx_i is omitted. We have that $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ is isomorphic to the $\mathbb{Q}_q$-vector space generated by $\{g\Omega/f^m | m \geq 1, g \in \mathrm{Homog}_{dm-n-1}\}$ modulo the relations

$$\left\{ \left(f \frac{\partial g}{\partial x_i} - mg \frac{\partial f}{\partial x_i} \right) \frac{\Omega}{f^{m+1}} : 1 \leq m \leq n, 0 \leq i \leq n, g \in \mathrm{Homog}_{dm-n} \right\}. \tag{1}$$

We refer to (1) as the *Griffiths-Dwork relations*. Hence every element $\omega \in H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ can be represented by $g\Omega/f^m$ with m minimal and $g \in \mathrm{Homog}_{dm-n-1}$; we refer to such an integer m as the *pole order* of ω. In fact, the basis elements of pole order m span the m-th graded piece of the Hodge filtration of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ [Gri69, §8]. Thus, the maximum pole order in any basis of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ is n.

As [AKR10, Remark 3.2.5] suggests and also with the anatomy of a Kedlaya-type algorithm described in 1.4 in mind, Griffiths's description of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ gives a natural way of computing a p-adic approximation of the *Frobenius matrix*, i.e. the matrix representation F of $\mathrm{Frob}_q \,|H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ with respect to an explicitly computable basis:

(1) For $m = 1, \ldots, n$, descend a basis for Homog_{dm-n-1} to a basis for $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$.
(2) Compute the action of Frobenius on basis elements of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_q})$ expressed as truncated power series.
(3) Apply the relations (1) to rewrite each term in the power series from (2) as a $\mathbb{Q}_p$-linear combination of forms $g\Omega/f^m$ with $m \leq n$, i.e., in the span of the basis from (1). This step is commonly referred to as *pole reduction*.

We assume $q = p$ from now on and denote Frob_p by σ. By convention, the natural numbers include 0. For $\beta = (\beta_0, \ldots, \beta_n) \in \mathbb{N}^{n+1}$, let $x^\beta \coloneqq x_0^{\beta_0} \cdots x_n^{\beta_n}$. We also let $\partial_i f \coloneqq \partial f / \partial x_i$.

2.3. **Computing the basis of cohomology.** For the sake of computational efficiency, we prefer having a *monomial* basis for $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$. Following Griffiths's

construction, computing a monomial basis B for $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$ amounts to computing a monomial basis for the cokernel of the following linear transformation:

$$M_k : (\mathrm{Homog}_{k-(d-1)})^{\oplus n+1} \to \mathrm{Homog}_k$$
$$(\mu_0, \ldots, \mu_n) \mapsto \sum_{i=0}^{n} \mu_i \partial_i f$$

for $k = dm - n - 1$, where $m = 1, \ldots, n$. (Note that $\partial_i f \in \mathrm{Homog}_{d-1}$, so $\mu_i \partial_i f \in \mathrm{Homog}_{k-(d-1)+(d-1)} = \mathrm{Homog}_k$.)

2.4. **Action of Frobenius.** We can express the action of Frobenius σ on a basis element $x^\beta \Omega / f^m$ of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$ via a series expansion as follows [Cos15, §1.2.3]:

$$\sigma\left(\frac{x^\beta \Omega}{f^m}\right) = p^n \frac{x^{p(\beta+1)-1}\Omega}{f^{pm}} \sum_{i \geq 0} \binom{-m}{i} \left(\frac{\sigma(f) - f^p}{f^p}\right)^i. \tag{2}$$

The number of terms on the right hand side of (2) visibly depends on p, and this is one of the primary reasons that the run-time of the algorithm from [AKR10] is exponential in p. Using an insight of Harvey [Har07, Proposition 4.1], one can rewrite (2) in a way such that the number of terms in the series expansion of $\sigma(x^\beta \Omega / f^m)$ does not depend on p.

Proposition 2.2. *Let $x^\beta \Omega / f^m$ be an integral basis element of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$. Let $N \coloneqq N_m$, the series order, and $s = N + m - 1$ and $C_{i,\alpha}$ be the coefficient of x^α in f^i. For $0 \leq i < N$, define*

$$D_{i,m} \coloneqq \sum_{i=j}^{N-1} (-1)^{i+j} \binom{-m}{i} \binom{i}{j}.$$

We have

$$p^{m-n-1} \sigma\left(\frac{x^\beta \Omega}{f^m}\right) \equiv \sum_{j=0}^{N-1} \sum_{|\alpha| = dj} p^{m-1} (D_{j,m} C_{j,\alpha} \bmod p^s) \frac{x^{p(\beta+\alpha+1)}\Omega}{f^{p(m+j)} x^1} \bmod p^{r_m}.$$

Proof. See [Cos15, Lemma 1.10]. □

Remark 2.3. The notations N_m and r_m will be explained in 2.5.

2.5. **Precision.** Since the coefficients of the Frobenius matrix F are in $\mathbb{Q}_p$, we need to compute the entries of F to some sufficient finite precision such that one can *provably* recover the correct integer coefficients of $Q(T)$ from a p-adic approximation of F.

Definition 2.4. For $x, \overline{x} \in \mathbb{Q}_p$, we say an approximation $\overline{x}$ of x has *absolute p-adic precision* N if $v_p(x - \overline{x}) = N$. Suppose $x = p^k u$ with $k = v_p(x)$ and $u \in \mathbb{Z}_p^\times$. The *relative p-adic precision* of the approximation $x + p^N \mathbb{Z}_p$ is $\rho(x; N) \coloneqq \max\{0, N - v_p(x)\}$; equivalently, $\rho(x; N)$ is the largest natural number r such that $u = p^{-k} x$ is determined modulo p^r, i.e. we know the first r p-adic digits of u.

There are two aspects to the precision problem:

(1) the p-adic precision needed in an approximation of F to uniquely determine its characteristic polynomial, and

(2) the number of terms needed in the series expansion of the Frobenius action on differentials (c.f. 2) in order to obtain a specific precision on the resulting Frobenius matrix.

The first question is essentially answered by the Riemann hypothesis; see, e.g. [Ked12, Lemma 1.2.3 and §4.3]. A helpful ingredient here is a classical result known as "Newton over Hodge" due to Mazur [Maz73], which relates the Hodge filtration to p-divisibility of the Frobenius matrix. In summary, Mazur's theorem says that $p^{n-m}|p^{-1}\sigma\left(\frac{x^\beta\Omega}{f^m}\right)$, where n is as before the dimension of the ambient projective space, and $1 \leq m \leq n$; also see [Ked12, §3.3] for more details. Combined with [Ked12, Lemma 1.2.3], one can deduce the *relative p-adic precision* r_m that one needs to compute for $\sigma(x^\beta\Omega/f^m)$. We refer to r_m as the *relative precision* for the basis element $x^\beta\Omega/f^m$, and r_m depends both on the pole order m as well as the characteristic of the base field.

On the other hand, the analysis of how many terms in the series expansion we need in order to obtain the desired relative precision is a very delicate matter, and is one of the core difficulties in the study of explicit p-adic cohomological methods. The best known theoretical result in this direction is attributed to [AKR10, Proposition 3.4.6 and Proposition 3.4.9]. We let N_m denote the number of terms in the series expansion of $\sigma(x^\beta\Omega/f^m)$ in (2) that we need to compute $\sigma(x^\beta\Omega/f^m)$ to relative precision r_m; we refer to N_m as the *series order* of the basis element $x^\beta\Omega/f^m$, and N_m depends on m, r_m, and p. [AKR10, Algorithm 3.4.10] gives an algorithm for computing N_m; to the best knowledge of the authors, this algorithm has not been implemented. For the cases implemented by [Cos15], we use the values precomputed there. For the other cases we study (i.e. cubic hypersurfaces), we provide an implementation that calculates these precisions for an arbitrary projective hypersurface. We record the values of $(N_1, \ldots, N_n)$ we get for various values of p we consider: for cubic surfaces, $p = 7, 11$, we have $(N_1, \ldots, N_3) = (1, 3, 2)$ and we get $(1, 2, 1)$ for $p = 13$. For cubic threefolds, we get $(N_1, \ldots, N_4) = (2, 5, 5, 4)$ for $p = 7, 11, 13$. Finally, for cubic fourfolds we get $(N_1, \ldots, N_5) = (1, 5, 5, 5, 4)$ for $p = 7$, and $(1, 5, 5, 4, 3)$ for $p = 11, 13$.

Finally, for each $m = 1, \ldots, n$, given r_m and N_m, let $s_m \coloneqq N_m + m - 1$, we define the *algorithm precision* to be $M \coloneqq \max_{i=1,\ldots,n}\{r_m + v_p((ps_m - 1)) - m + 1\}$. More specifically, this means that we are doing linear algebra over the ring $\mathbb{Z}_p/p^M\mathbb{Z}_p \cong \mathbb{Z}/p^M\mathbb{Z}$ in each of the step in the controlled reduction algorithm (c.f. [Cos15, 1.5.1 Step 1]). The definition of M will become apparent later.

3. Pole reduction

3.1. **Controlled Reduction.** To express $p^{-1}\sigma(x^\beta\Omega/f^m)$ as a $\mathbb{Q}_p$-linear combination of basis elements B of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$, we need to employ certain pole reduction algorithms to reduce the pole order of elements in the cohomology group using the Griffiths-Dwork relations (1). A straight-forward application of (1) is done in [AKR10], but it can often turn a sparse polynomial into a dense polynomial. We now recall the method of *controlled reduction* [Cos15, §1.4], a variation of [AKR10] that improves the time dependency on p from that of p^n in [AKR10] to $p^{1+\epsilon}$. Controlled reduction is a family of algorithms, which, generally speaking, works by

applying the Griffiths-Dwork relations (1) in a clever way that reduces the computation from polynomial arithmetic to matrix-vector multiplication and matrix addition[1].

Before we state the main result, we recall a technical definition.

Definition 3.1. For $S \subseteq \{0, \dots, n\}$, let $J_S \coloneqq \langle \partial_i f \rangle_{i \in S} \oplus \langle x_i \partial_i f \rangle_{i \notin S}$. We say that the hypersurface defined by the homogenous polynomial f is *S-smooth* if J_S defines the empty scheme. We say f is *nondegenerate* if f is $\emptyset$-smooth.

Remark 3.2. When $S = \{0, \dots, n\}$, f is S-smooth is equivalent to f is smooth.

Let $|\cdot|$ denote the (restriction of the) ℓ_1 *norm* on $\mathbb{N}^n$, i.e., for $u = (u_0, \dots, u_n) \in \mathbb{N}^n, |u| = u_0 + \ldots + u_n$.

Proposition 3.3 (Controlled Reduction)**.** *Assume f is S-smooth with $d \geq \#S$. Let $x^S \coloneqq \prod_{i \in S} x_i$. Let $u = (u_i) \in \mathbb{N}^{n+1}$ and $v = (v_i) \in \mathbb{N}^{n+1}$ with $|v| = d$. Suppose further that for $i \in S$, if $v_i = 0$, then $u_i = 0$. There is a $\mathbb{Z}_p$-linear map*

$$R_{u,v} : \mathrm{Homog}_{dn-n} \to \mathrm{Homog}_{dn-n}$$

such that for any positive integer m and $g \in \mathrm{Homog}_{dn-n}$,

$$m \frac{x^{u+v} g \Omega}{x^S f^{m+1}} \equiv x^u \frac{R_{u,v}(g)\Omega}{x^S f^m} \text{ in } H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p}). \tag{3}$$

Furthermore, setting $u = (x_0, \dots, x_n)$, the linear map can be represented as a $\binom{dn}{n} \times \binom{dn}{n}$ matrix with entries being linear or constant polynomials in $\mathbb{Z}_p[x_0, \dots, x_n]$.

Proof. See [Cos15, Proposition 1.15]. □

We make explicit the definition of $R_{u,v}(g)$ and how to compute a matrix representation of $R_{u,v}$ following the proof given in [Cos15, Proposition 1.15]. Under the hypothesis in Proposition 3.3, we can find $g_i \in \mathrm{Homog}_{dn-n-|S|+1}$ such that

$$\frac{x^v g}{x^S} = \sum_{i \in S} g_i \partial_i f + \sum_{i \notin S} x_i g_i \partial_i f.$$

We define $R_{u,v}$ by

$$R_{u,v}(g) \coloneqq x^S \left(\sum_{i \in S} \frac{u_i g_i + x_i \partial_i g_i}{x_i} + \sum_{i \notin S} [(u_i + 1) g_i + x_i \partial_i g_i] \right),$$

Remark 3.4. We refer to the vector v in $R_{u,v}$ as the *direction of reduction*, c.f. Section 4.

Remark 3.5. The implementation [Cosb] only supports the case when $S = \{0, \dots, n\}$, making it unusable for cubic threefolds and fourfolds. [CHK], which uses an analogue of [Cos15, Proposition 1.18] for controlled reduction as opposted to [Cos15, Proposition 1.15], can handle cubic fourfolds in large enough characteristic, but the implementation requires a lot of memory (from a larger row reduction).

[1] In theory, it reduces the problem to polynomial evaluation and matrix-vector multiplication, and one uses the fast-evaluation techniques of [BGS07] to reduce the fast evaluation to matrix addition.

For each fixed constant vector $v \in \mathbb{N}^{n+1}$, let $u = (u_0, \dots, u_n) \in \mathbb{N}^{n+1}$. The $R_{u,v}$ linear map can a priori be represented by a matrix of linear polynomials in $\mathbb{Z}_p[x_0, \dots, x_n]$; however, for better computational efficiency, we instead think of $R_{u,v}$ as being algorithmically represented by $n+2$ constant coefficient matrices $A_0, \dots A_{n+1}$ with $R_{u,v} = \sum_{i=0}^{n} A_i u_i + A_{n+1}$.

In order to apply Proposition 3.3 to reduce the pole order of a monomial differential $x^\beta \Omega / f^m$ in practice, we need a consistent way of choosing $g \in \mathrm{Homog}_{dn-n}$ and some $u \in \mathbb{N}^{n+1}$ such that

$$x^\beta = x^u g. \tag{4}$$

In our implementation, as in Costa's implementation of [Cos15], the vector u in (4) is determined by a straight-forward algorithm as follows.

Algorithm 1 Tweak(I, m): remove m units from the front of integer vector I

Require: I, a vector of nonnegative integers; $m \geq 0$
Ensure: I' after removing an integer vector J with $|J| = \min(m, |I|)$ from the front

```
count ← 0
o ← m
I' ← copy(I)
while m > 0 do
    for i ← 1 to length(I') do
        if I'[i] > 0 then
            I'[i] ← I'[i] − 1
            m ← m − 1
            break
        end if
    end for
    if count > length(I') · o then
        return I'
    end if
    count ← count + 1
end while
return I'
```

The output of Tweak$(\beta, dn-n)$ gives the u in (4).

3.2. **Evaluation of $R_{u,v}$-matrices.** When we apply controlled reduction for the purpose of reducing the pole order of differentials via the $R_{u,v}$-matrices in practice, we write $u = (x_0, \dots, x_n) + yv$ for some fixed constant vectors $(x_0, \dots, x_n), v \in \mathbb{N}^{n+1}$, and variable $y \in \mathbb{N}$. Let $M(y) \coloneqq R_{(x_0,\dots,x_n)+yv,v} = Ay + B$, where A and B are constant coefficient matrices. To reduce the pole order of a differential ω by k at a time for some $k \in \mathbb{N}$ by applying Proposition 3.3, we need to evaluate the expression

$$M(0) \cdots M(k) w, \tag{5}$$

where w is the vector of coefficients of some g as in (3). The question of how one chooses v will be addressed in Section 4. For now, we focus on how to efficiently evaluate (5). A priori, (5) can be evaluated via k matrix multiplications of matrices

of size $\binom{dn}{n} \times \binom{dn}{n}$ and a matrix-vector multiplication. Instead, (5) can be evaluated via $O(k)$ matrix-vector multiplications and $O(k)$ matrix additions as follows.

Algorithm 2 Fast Evaluation of (5)

```
M ← kA + B
for i = k downto 0 do
    w ← Mw
    M ← M − A
end for
return w
```

We introduce a customized data structure for an efficient implementation of Algorithm 2 in Section 5.1.

3.3. **Outline of the pole reduction process.** For each element in a monomial basis B of $H^n_{\mathrm{dR}}(U_{\mathbb{Q}_p})$ as described in Section 2.3, recall that our objective is to express (2) as a $\mathbb{Z}_p$-linear combination of elements of B by means of a pole reduction procedure (elements in B have pole order at most n). There are two main steps for how this is done in controlled reduction:

(0) Apply Algorithm 1 to express the terms in the right hand side of (2) in the format of the left hand side of (3).
(1) Iteratively apply the $R_{u,v}$-matrices to reduce the pole order of the terms in (2) to exactly n.
(2) Directly apply the Griffiths-Dwork relation (1) as in [AKR10] to express the output from Step 1 as a linear combination of elements of B.

Most of the computational complexities lie in Step 2, which we repackage as an abstract optimization problem in Section 3.4 and to which we propose various solutions in Section 4.

3.4. **The Abstract Reduction Problem.** We define an optimization problem that models the problem of controlled reduction. Recall our convention from earlier that the natural numbers include zero.

Throughout the following, we will consider various elements $u \in \mathbb{N}^{n+1}$, which correspond to exponent vectors of terms from Proposition 3.3. $|u|$ always denotes ℓ_1 norm. We fix $d, N \in \mathbb{N}$. We call d the *degree* or *step size* and N the *target norm*. Our first goal is to axiomatize the possible choices for v in 3.3.

Definition 3.6. Let $V_{\mathrm{all}}(u,k)$ be the set $\{v \in \mathbb{N}^{n+1} \colon |v| = d, u - kv \in \mathbb{N}^{n+1}\}$, where $k \geq 1$. That is, the set of all elements of ℓ_1 norm d, with the condition that the entries of u remain nonnegative after subtracting by v k times.

Remark 3.7. The assumptions of Proposition 3.3 introduce restrictions on what v can be, in terms of the components of u. Thus, we cannot always choose an arbitrary $v \in V_{\mathrm{all}}(u,k)$.

Definition 3.8. A *direction choice collection* $V(u,k)$ is, for any (u,k) with $k \geq 1$ and $u \in \mathbb{N}^{n+1}$ such that $N = |u| - md$ for some $m \geq 1$, a set $V(u,k) \subseteq V_{\mathrm{all}}(u,k)$, for which $V(u,1)$ is always nonempty.

Remark 3.9. The condition $N = |u| - md$ comes from Proposition 2.2, which gives rise to terms of this form. One could make minor modifications to instead use the condition that $0 < |u| - md < N$ for some m, but we do not pursue this as our applications to hypersurfaces always satisfy Definition 3.8.

Remark 3.10. Note that $V(u,k)$ necessarily depends on N, and (via $V_{\text{all}}(u,k)$) also d. However, since we regard both N and d as fixed, we supress this from our notation.

Example 3.1. By restricting to u such that $N = |u| - md$, $V_{\text{all}}(u,k)$ itself is a direction choice collection.

Example 3.2. Let $V_{\text{full}}(u,k)$ be the subset of $V_{\text{all}}(u,k)$ with the additional condition that if $u_i \neq 0$, then $v_i \neq 0$.

Remark 3.11. For projective hypersurfaces where the degree d equals or exceeds the number of homogeneous variables $n+1$, the direction choice collection $V_{\text{full}}(u,k)$ is always sufficient for Proposition 3.3. However, for hypersurfaces where $d < n+1$, this no longer holds, so we must introduce a set S as in Proposition 3.3.

Example 3.3. Let $S \subseteq \{0,\dots,N\}$. Let $V_S(u,k)$ bet the subset of $V_{\text{all}}(u,k)$ with the additional condition that for $i \in S$, if $u_i = 0$, then $v_i = 0$. Note that $V_{\text{full}}(u,k) = V_{\{0,\dots,N\}}(u,k)$.

We are now ready to introduce the central concept of this section, which we term the *abstract reduction game*; this will be formulated as a one-player game.

Definition 3.12. An *Abstract Reduction Game* is a one-player game with the following inputs and rules. We start with U, a finite list (i.e. multiset, not a set) whose elements lie in $\mathbb{N}^n$ with the following property: for every $u \in U$, $N = |u| - md$ for some $m \in \mathbb{N}$. We are also given a direction choice collection $V(u,k)$ for every $u \in U$. Moreover, we are given constants $D \ll T \ll M$.

On each turn, the player does one of following two moves, where each move accumulates a cost, and the problem is to minimize the cost.

(1) Choose $u \in U$, $k \geq 1$, and $v \in V(u,k)$. Replace $u \in U$ by $u - kv$ at a cost of $Mk + T$; this move is called a *reduction chunk of size* k.
(2) Remove all duplicates in U at a cost of D per duplicate. This operation may not be done twice in a row. This move is called *combining like terms*.

Throughout the game, all u must have $N \leq |u|$; that is, a reduction chunk of size k is not allowed if it would make $|u| < N$. The game ends when all $u \in U$ satisfy $|u| = N$.

Remark 3.13. The interpretation of the constants is as follows: M denotes the cost of reducing one term by one pole order. This is dominated, in practice, by the cost of a single matrix-vector multiplication (there is also a matrix subtraction, but in practice the memory bottleneck of matrix-vector multiplication takes a lot more time, even though technically both have $O(s^2)$ operations, where s is the side length of the matrices). T is the startup cost to compute the A and B matrices as described in Section 3.2. This consists of matrix additions and scalar multiplications, which like matrix subtractions, are much faster than a matrix-vector multiplication. Finally, D is the cost of adding two vectors which correspond to duplicates from U, which in theory is a single vector addition, but in practice again memory overhead dominates.

Theorem 3.14. *Any abstract reduction game ends.*

Proof. Given an abstract reduction game with list U, we define $|U| \coloneqq \sum_{u \in U} |u|$. We denote the state of U after the i-th move by U_i. If we perform a reduction chunk of size k during the i-th move, then we have $|U_i| = |U_{i-1}| - kd$. If we combine like terms, then in the worst case, $|U_i|$ stays fixed, since no duplicates were removed. Thus, the sequence $(|U_i|)_i$ is a monotonically nonincreasing quantity that must decrease every other move and it is also bounded below by N. Therefore, the sequence $(|U_i|)_i$ converges (and thus is eventually constant as it is an integer sequence) and no more moves are possible hereafter.

Furthermore, since $V(u, 1)$ is always nonempty as long as there is some u with $|u| > N$, we can always perform a reduction chunk of size 1 unless the game has already been won. Thus, every game terminates with every $u \in U$ having $|u| = N$. □

The goal of a player/algorithm is to finish the game with minimal cost. One must balance the savings of taking the first move with large k (thus avoiding the startup cost S) with the savings of combining like terms in the second move. In general, we fix the reduction direction choices $V(u, k)$ once and for all, and expect to solve the problem for many choices of U. Given a fixed choice of $V(u, k)$ (for example, $V_{\text{all}}(u, k)$), a *reduction policy* is a program that plays the absract reduction game. The optimization problem is to find a good policy that runs with (close to) minimal cost for the U that appear in practice. We discuss various versions of reduction policies in Section 4.

Remark 3.15. When doing controlled reduction, we always take the *degree* d to be the degree of the defining equation of the hypersurface X, and the *target norm* N to be n, the dimension of the ambient projective space of X. The input U is obtained via Algorithm 1.

4. Reduction Policies

We now describe three practical implementations of reduction policies. Following the tradition of previous implementations like [Cos15] and [CHK], all of our policies begin with the following steps:

(1) Choose a $u \in U$ to reduce.
(2) Choose $v \in V(u, 1)$.
(3) Choose $k \in \mathbb{N}$ such that v remains in $V(u, k)$.

The v is always chosen using Algorithm 3, a greedy algorithm, which always outputs $v \in V(u, 1)$. Recall that for all $v \in V(u, 1)$, $|v| = d$, by the setup of Section 3.4.

Observe that Algorithm 3 guarantees that v satisfies the assumption of Proposition 3.3.

Remark 4.1. The reduction policy structure of choosing a $v \in V(u, 1)$ given each u is not required and possibly not optimal. It would be interesting to explore this in future work.

4.1. p-chunk reduction.

Our first reduction policy, which we call the *p-chunk reduction*, is the one implemented in [Cos15] and [CHK]. It takes advantage of the fact that for all $u, u' \in U$, we have $|u| \equiv |u'| \bmod p$, which follows from Equation (2). Let $L \coloneqq \max\{|u| : u \in U\}$ and $U_{\max} = \{u \in U \mid |u| = L\}$. The p-chunk reduction policy goes as follows:

Algorithm 3 Greedy algorithm to choose v

Require: Inputs: $d \in \mathbb{N}$, $u \in \mathbb{N}^{n+1}$, $S \subseteq \{0, \dots, n\}$
Ensure: Output: $v \in V(u, 1)$

```
v ← [0, ..., 0]
i ← d
for s ∈ S do
    if u[s] ≠ 0 then
        v[s] ← 1
        i ← i − 1
        if i = 0 then
            break
        end if
    end if
end for
while i > 0 do
    for m = 0 upto n do
        if v[m] < u[m] then
            v[m] = v[m] + 1
            i = i − 1
        end if
    end for
end while
return w
```

(1) For every $u \in U_{\max}$, choose a v using Algorithm 3 to reduce u by a chunk of size p. If $|u| = p$, then reduce by a chunk of size $p - n$.
(2) Combine like terms.
(3) Recalculate L and $U_{\max}$.
(4) Repeat steps 1-3 until the game described in Section 3.4 is won.

Towards the end of the reduction, many like terms will be combined as there are fewer possibilities for u when it has smaller norm.

The p-chunk reduction policy is most effective when p is very large, since if p is small the start-up costs will accumulate and make the algorithm slow.

4.2. **Depth-first reduction.** Our second reduction policy, which we call *depth-first reduction*, works as follows. For each $u \in U$,

(1) Choose a v by Algorithm 3.
(2) Find the largest k such that v is still in $V_{\text{full}}(u, k)$. Then reduce u by a chunk of size k.
(3) Repeat steps 1-2 until $|u| = N$.

Unlike p-chunk reduction, this strategy never combines like terms.

Since the depth-first policy prioritizes long reduction chunks over combining like terms, it works best when the size of U is small. For example, at smaller primes, Equation (2) will have fewer cross terms. Also, if the equation f of the hypersurface is sparse, then Equation (2) will have fewer terms and the performance of depth-first reduction will be improved. This policy outperforms p-chunk reduction in our situations, for example in the case of quartic K3 surfaces of medium-low characteristic.

4.3. **Variable-by-variable reduction.** The third reduction policy that we have implemented, *variable-by-variable reduction*, aims to balance the approaches of the previous two. The key idea is to reduce u one coordinate at a time, and then combine like terms after each coordinate becomes zero. However, if the S from Proposition 3.3 is not $\{n\}$ or $\emptyset$, then the assumptions of 3.3 will require that more than one coefficient of v is nonzero. As such, we assume that X is $\{n\}$-smooth in the sense of Definition 3.1. By applying a change of variables, this suffices if X is i-smooth for any $i \in \{0, \ldots, n\}$. Recall also that for $S \subseteq S' \subseteq \{0, \ldots, n\}$, S-smooth implies S'-smooth, so this also covers the $\emptyset$-smooth case. In the $\{n\}$-smooth case, the first for loop of Algorithm 3 does not affect its output. The policy works by:

(1) For each u, choose a v by Algorithm 3.
(2) Choose k to be the largest k such that $v \in V(u, k)$. Reduce u by a chunk of size k.
(3) Repeat steps 1-2 until the n-th component u_n of u is zero.
(4) Now that all u have $u_n = 0$, combine like terms
(5) Repeat steps 1-4 for $u_{n-1}, \ldots u_0$ until the game ends.

This policy works better when U is more dense, so there are likely to be many $u \in U$ with the same first k entries. This is advantages for smaller primes p, where U is more dense. It also works better when the degree of the equation f is low; note that for the k-th entry, step (3) happens at most d times for each u. The biggest advantage of variable-by-variable policy is the memory requirements - since each u is processed with the same v simultaneously, one is only required to store a single $R_{u,v}$ matrix in memory at a time. This turns out to be a decisive advantage for GPU computations, which tend to be bottlenecked by the amount of memory on the GPU. We find that on cubic threefolds and cubic fourfolds, variable-by-variable outperforms both other strategies.

4.4. **Outlook.** The authors do not suspect that our reduction policies are the most optimal. In particular, one thing that all of our reduction policies have in common is that they do not inspect U at all. We hope that the framework of the abstract reduction problem inspires future work on more efficient reduction policies, which may lead to significant speedups. We hope to address this in future work.

5. Algorithms and Data Structures

A fast implementation of controlled reduction relies on efficient implementations of many auxiliary algorithms that are not, a priori, related to controlled reduction. In particular, to implement controlled reduction, one must at least have (1) a matrix multiplication mod p^n, and (2) fast evaluation of linear recurrences of polynomials (with coefficients that are matrices with entries mod p^n). Furthermore, fast implementations of these algorithms are not always widely available or immediately usable, though if we are on the CPU, the FLINT library [FLI26] has fast implementations of matrix multiplication. We provide implementations of two variants of matrix multiplication, and we introduce data structures for managing linear recurrences. Our implementations are released as open-source software available at [BGM26].

5.1. **Partially Evaluated (linear) Polynomials–a novel data structure for fast evaluation of matrices of polynomials.** We introduce an abstract data structure that can be used to encapsulate the evaluation procedure described in

Section 3.2. Let A be a (possibly noncommutative) ring. For us, A is always $M_n(B)$ for some commutative ring B. We consider an element $R \in A[u_0, \dots, u_n, v_0, \dots, v_n]$ of degree 1. We let $V \in \mathbb{N}^{n+1}$ be some finite set of values. For the evaluation in Section 3.2, we need to store polynomials $R' = R(v_0, \dots, v_n) \in A[u_0, \dots, u_n]$ for each value $v \in V$ with $|v| = d$. Creating R can be an expensive operation, while storing an $R(v_0, \dots, v_n)$ can take a lot of memory. Finally, we must manage the process of evaluating the matrix $R'(x_0 + yv_0, \dots, x_n + yv_n)$ to get the matrices A and B from Section 3.2; this also is potentially an expensive operation.

A **Partially Evaluated Polynomial** (PEP) is a key-value data structure which stores evaluations of R for various fixed values $v \in V$, manages the storage of the R', and contains methods for evaluation of linear recurrences as in 3.2. A PEP data structure has has three main operations:

- `compute`, a function provided by the user which inputs a fixed $v_0 \in \mathbb{N}^{n+1}$ and returns a list of the coefficients of $R(u, v_0) \in A[u_0, \dots, u_n]$
- `get`, which gets $R' = R(u, v)$.
- `eval_to_linear!`, which takes the output of `get` and an $x = (x_0, \dots, x_n)$ and evaluates $R(x + yv, v) \in A[y]$ as in Section 3.2. This can then be used for fast evaluation as in Algorithm 2.

PEP structures generally have some sort of underlying dictionary and are distinguished by how they manage the creation of key-value pairs. We provide an abstract Julia type, `AbstractPEP`, as a supertype for different PEP implementations. We provide five such implementations:

- `EagerPEP`, which is backed by a dictionary and `compute`s all possible keys upon initialization (optionally, in parallel).
- `LazyPEP`, which is backed by a dictionary and `compute`s keys lazily.
- `LRULazyPEP`, which is backed by an LRU Cache of a fixed size, and computes keys lazily.
- `LRUCachePEP`, which assumes the output coefficients are expected to be in GPU memory. It is backed by an internal PEP on the CPU, and an LRU cache whose entries are in GPU memory, and it manages moving entries to and from the GPU.
- `LFUDACachePEP`, which gives the same behavior as `LRUCachePEP` but uses an LFUDA cache.

Julia's JIT-compiler and dynamic dispatch make PEP objects more or less interchangeable, allowing one to rapidly test different memory management strategies. The optimal choice of PEP object tends to depend heavily on the choice of reduction policy; for example, we anecdotally observed that the LFUDA backing had fewer cache misses with the depth-first policy (Section 4.2) than LRU, while the LRU backing had fewer misses with the variable-by-variable policy (Section 4.3). As another anecdote, the `LRULazyPEP` became useful for our computations with cubic fourfolds, when our CPU memory became a constraint (in addition to GPU memory, which was also a constraint).

5.2. `CuModMatrix`**: mod** N **matrix multiplication on GPUs.** To the authors' best knowledge, an open-source finite-field linear algebra GPU implementation using Julia and CUDA.jl was not available during the development of this work. To this end, we provide a (dense) GPU matrix Julia datatype for mod N linear algebra, `CuModMatrix`, implemented using CUDA.jl. `CuModMatrix` implements

standard matrix operations (matrix addition and multiplication, inverse, LU decomposition, etc.). Each `CuModMatrix` has an element type which must be a Julia integer of float type (e.g. `Int8` or `Float64`) that backs the elements of the matrix. If the element type is a floating point type, the mantissa is used to efficiently obtain a smaller integer type (e.g. `Float64` becomes `Int53`, `Float32` becomes `Int24`). We prefer these floating-point-backed types, as we can use Nvidia's CUBLAS [Nvi24].

We use the method of [BGLM23, Algorithm 1] for mod N matrix multiplication. This involves breaking the matrix into vertical stripes, whose lengths are determined by the number of possible operations without overflow. The algorithm uses CUBLAS to multiply the stripes and reduces the entries modulo N.

5.3. `KaratsubaMatrix`**: Mod N matrix multiplication on GPUs via Karatsuba multiplication.** Often, the necessary p-adic precision for controlled reduction is too big to do matrix multiplications within the `Float64` datatype (i.e. `Int53`) which is the largest one available for use with cuBLAS. For cubic threefolds over $\mathbb{F}_{11}$, for example, the precision analysis of [AKR10] indicates that we do matrix multiplications $\bmod 11^8$. However, we have that $\left\lfloor \frac{2^{53}}{(11^8-1)^2} \right\rfloor \approx \lfloor 0.19 \rfloor = 0$, meaning that we cannot justify even a single multiplication.

To overcome this, we use Karatsuba multiplication. To multiply two matrices $A, B \bmod N$, we write $A = A_1 + MA_2$ and $B = B_2 + MB_2$, where we require $N|M^2$. Then

$$AB = A_1B_1 + M(A_2B_1 + A_1B_2) + M^2A_2B_2.$$

We can re-express the term $A_2B_1 + A_1B_2$ as the difference between products $(A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2$, so computing the multiplication AB now requires only 3, instead of 4, matrix multiplications. We may want to choose M to be 11^4 in the above example, in which case we can discard the last term $M^2A_2B_2$. This approach is considered in [Bao24].

We provide a GPU-accelerated `KaratsubaMatrix` type that also supports basic operations such as addition, subtraction, negation, and scalar multiplication.

6. Examples

In this section, we survey prior work in the literature and discuss the methodologies adopted for finding all of our explicit examples. The equations for each example can be found in [BGHM26b].

6.1. **Methodology.** There are two principal methods for finding explicit examples of hypersurfaces with interesting Newton polygons. The scripts that implement the methods described below are available at [BGHM26c].

6.1.1. *Method 1: Fully random guessing.* We randomly choose equations of hypersurfaces, computing the zeta function of each, hoping to find one with the desired Newton polygon. The idea is to exploit the fact that Newton strata jump in codimension 1 [dJO00], so we expect that by randomly sampling the closure of some stratum, finding a hypersurface in a stratum one jump higher has probability roughly $1/p$. Thus, if a strata has codimension n, the chance of finding such a hypersurface has probability $1/p^n$. We can formally justify these expectations using the Lang-Weil estimate [LW54], though that bound is not so tight as to determine what happens practically. This method is more effective the smaller p is.

6.1.2. *Method 2: Random deformations of a fixed example.* In this method, we fix a hypersurface and add random monomials to it. We try to fix a hypersurface which has a Newton polygon as high as possible, ideally supersingular. For example, we may choose the Fermat hypersurface or a member of a Dwork-type pencil. The idea is to exploit a phenomenological observation: equations with higher Newton polygons are often similar-looking to other equations with higher Newton polygons. This phenomenon has not been fully explored in the literature; however, there are theoretical reasons to expect such similarities. Since the Newton strata are locally closed strata in the moduli space, they are defined by polynomial equations. If the defining equations of these strata have low degree, that could lead to "similar-looking equations" among various points of the strata. For example, if we consider quartic K3 surfaces, one can in principle compute the equation of the $h = 2$ Newton stratum using Fedder's criterion [MP, §2], which has degree $p - 1$. We leave the computation of explicit equations for these strata as future work.

6.2. **Curves.** Let $g \geq 1$, $\mathcal{M}_g$ be the moduli space of genus g curves, and $\mathcal{A}_g$ be the moduli space of g-dimensional principally polarized abelian varieties. One is generally interested in how the *open Torelli locus*, i.e. the image of $\mathcal{M}_g$ under the Torelli morphism $\tau_g : \mathcal{M}_g \to \mathcal{A}_g$ sending a curve to its Jacobian, intersects various strata of $\mathcal{A}_g$, in particular the p-rank strata and Newton polygon strata. For the former, Achter and Pries [AP08] has shown that there exists a curve $C/\overline{\mathbb{F}}_p$ of genus g and p-rank f for all $0 \leq f \leq g$; the same result holds for hyperelliptic curves as well [AP11]. We are interested in whether the result of Achter and Pries holds over $\mathbb{F}_p$ for plane curves. We study quintics, sextics, and heptics, and use a combination of Method 1 and Method 2. We find

(1) Plane quintics of all possible p-ranks for $p = 7, 11, 13$. See [BGHM26b].
(2) Plane sextics of all p-ranks ≥ 6 for p $= 7, 11, 13$. See [BGHM26b].
(3) Plane heptics which are ordinary and almost ordinary for $p = 7, 11, 13, 17, 19, 23, 31$. See [BGHM26b].

6.3. **Quartic K3 surfaces.** Let X be a K3 surface; the Hodge numbers of $H^2_{rig}(X)$ are $1, 20, 1$. Thus, the only possibilities for the Newton polygon are:

(1) A straight line of slope 1 for 22 units, i.e. the supersingular case
(2) Lines connecting the vertices $(0, 0), (h, h - 1), (22 - h, 22 - h - 1), (22, 22)$ for some $h \in \{1, \ldots, 10\}$.

The h in possibility (2) is called the *Artin-Mazur height* of X. If X is supersingular, we say that $h = \infty$. Alternatively, note that the first slope of the Newton polygon is $\frac{h-1}{h} = 1 - \frac{1}{h}$.

Over an algebraically closed field k of characteristic p, K3 surfaces of all heights exist by [Tae16]. Quartic K3 surfaces of any height exist over $\mathbb{F}_2$ by [KS16], over $\mathbb{F}_3$ by [KTY22], and over $\mathbb{F}_5$ and $\mathbb{F}_7$ by [BGP25]. Over $\mathbb{F}_{11}$ and $\mathbb{F}_{13}$, [BGP25] provides examples of heights less than or equal to five. We obtain examples for quartic K3 surfaces of heights up to and including 4 over $\mathbb{F}_p$ for $p = 17, 19, 23$. See [BGHM26b].

Here we use Method 1 to obtain our result; we use Fedder's criterion [MP, §2] to rule out surfaces of height 1, so we have to make fewer random guesses to find surfaces of higher height.

6.4. **Quintic Surfaces.** Let X be any algebraic variety over a perfect field k of characteristic p. Then X is *Hodge–Witt* if the cohomology groups $H^j(X, W\Omega^i_X)$

are finitely generated W-modules for all i and j (where $W\Omega^i_X$ are the sheaves appearing in the de Rham–Witt complex). More generally, Illusie and Raynaud [IR83] introduce the notion of the *domino numbers* $T^{i,j}$ associated to a variety in characteristic p. If X is a projective surface, then by [Jos20, §2.23, §2.37], the only interesting domino for X is $T^{0,2}$; all others are either zero or related to this one by a duality theorem. Furthermore, we have $0 \leq T^{0,2} \leq \dim H^2(X, \mathcal{O}_X)$, and $T^{0,2}$ is given as a formula depending on the Hodge numbers and slopes of the Newton polygon. We may visually understand this formula as follows: if we take the vertical distance between the Newton polygon and Hodge polygon, the domino number $T^{0,2}$ is the minimum such difference in the slope 1 part of the Hodge polygon. If X is a quintic surface, then $\dim H^2(X, \mathcal{O}_X) = 4$ and so $T^{0,2} \in \{0, 1, 2, 3, 4\}$; X is Hodge–Witt if and only if $T^{0,2} = 0$.

In characteristics $7, 13$, and 17, the Fermat quintic is supersingular. By applying Method 2 to the Fermat quintic, we obtain examples of quintic surfaces of all possible domino numbers in these characteristics. However, in characteristic 11, the Fermat quintic is ordinary, and the authors do not know of a supersingular example. Moreover, given the large number of possible Newton polygons and the large amount of time that computing the zeta function of a single dense example takes (about one hour), Method 1 does not seem viable for quintic surfaces.

6.5. **Cubic Threefolds.** The Hodge numbers of a cubic threefold are $0, 5, 5, 0$. Thus, their possible Newton polygons are the same as for curves of genus 5, with all slopes increased by one. In characteristic $p \neq 2$, one may deduce the existence of ordinary cubic threefolds over $\overline{F}_p$ as a consequence of [Ach14, Proposition 4.1]. Breifly, the closure of the locus of intermediate Jacobians in the moduli of abelian fivefolds contains the locus Jacobians of hyperelliptic curves of genus 5. Since there are ordinary hyperelliptic curves, and ordinary is an open condition on the moduli of abelian varieties, one deduces that there must be ordinary points in the locus of intermediate Jacobians, and thus cubic threefolds. The authors are not aware of any other results on the realization of Newton strata of cubic threefolds, even over $\overline{\mathbb{F}}_p$, in the literature.

We find cubic threefolds in characteristic $p = 7, 11$ and 13 which are ordinary and almost ordinary, using Method 1. See [BGHM26b].

6.6. **Cubic Fourfolds.** Let X be a cubic fourfold. The Hodge numbers of $H^4_{rig}(X)$ are $0, 1, 21, 1, 0$; hence the possibilities for the Newton polygon of X are the same as for a K3 surface, except that the slopes are all increased by one. Thus, primitive cohomology "looks like" the cohomology of a K3 surface that has been Tate twisted by 1. As such, we may define the *Newton height* of X analogously as $h = \frac{1}{2-\text{newt}_1}$, or $2 - \frac{1}{h} = \text{newt}_1$, where newt_1 is the slope of the first segment in the Newton polygon of X. Using Method 1, we find cubic fourfolds of heights 1 and 2 in characteristic $p = 7$. See [BGHM26b].

There is some existing literature on examples of cubic fourfolds. [AKPW25] computes all cubic fourfolds over $\mathbb{F}_2$ up to isomorphism, and verifies that all possible Newton polygons occur, which fully resolves Question 1.3 over $\mathbb{F}_2$. Furthermore, Bragg and Perry, in an unpublished work [BP], calculate the Newton height of the Dwork pencil of cubic fourfolds $x_0^3 + x_1^3 + x_2^3 + x_3^3 + x_4^3 + x_5^3 + \lambda\big(x_0x_1x_2 + x_3x_4x_5\big)$. They determine that for all p, there exists a λ_b for which the corresponding fourfold X_{λ_b} is supersingular, and a λ_0 for which X_{λ_0} is ordinary. Then they deduce by a

TABLE 1. Algorithms for plane quintics over $\mathbb{F}_p$. Times in one Apple M3 Pro CPU core-seconds.

p	4.2	[Tui16]	[Kyn22]
11	0.911	8.24	1.32
13	0.683	1.3	1.26
17	0.899	2.21	1.38
19	0.987	2.36	1.43
23	1.157	2.66	1.45
29	3.898	3.13	0.57
31	1.488	3.37	0.65
37	1.692	18.72	0.76
41	4.679	4.22	0.72

deformation argument that there exist cubic fourfolds of all heights over $k = \overline{\mathbb{F}}_p$. However, this argument is nonconstructive, and so it only answers Question 1.3 for the ordinary case.

7. Performance Comparisons

In this section, we compare the practical performance of our reduction policies to existing implementations in various settings. Although our ultimate goal is to compute zeta functions of higher dimensional hypersurfaces, the method of controlled reduction *can* be used to compute the zeta function of smooth plane curves. We provide some performance comparisons for completeness sake. For smooth plane quintics (curves of genus 6), we compare the performance of our implementation of the depth-first reduction policy 4.2 together with fast evaluation (c.f. Algorithm 2) on a single CPU against Tuitman's algorithm [Tui16], a Kedlaya-type algorithm, as well as Kyng's algorithm [Kyn22], an implementation of Harvey's trace formula method [Har15], both of which are available in Magma [BCP97]. The running times for each of the aforementioned algorithms on smooth plane quintics defined by random dense polynomials $f \in \mathbb{F}_p[x_0, x_1, x_2]_5$ can be found in Table 1. We note that Kyng's algorithm is considered to be the state of the art for computing the zeta functions of higher genus (possibly singular) plane curves.

[CHK] is the current state the art for computing zeta functions of projective hypersurfaces beyond curves. In Table 2, we see that our implementation using the variable-by-variable reduction policy 4.3 obtains performance surpassing [CHK] on quartic surfaces with six terms. Multithreading, which is well-supported in our implementation, speeds up computation further. [CHK] only handles cubic threefolds and fourfolds over $\mathbb{F}_p$ for $p \geq 13$, cannot compute random quintic surfaces, and runs out of memory (under 32GB) on random cubic fourfolds; our implementation handles all of these cases, with random quintic surfaces taking about one Apple M3 Max core-hour. Using the variable-by-variable policy 4.3, random cubic fourfolds over $\mathbb{F}_7$ take 2–3 hours on an Intel i5-8400 with one Nvidia RTX 3070 GPU, and the Fermat cubic fourfold takes 15–20 minutes.

References

[Ach14] Jeffrey D. Achter, *On the Abelian fivefolds attached to cubic surfaces*, Mathematical Research Letters **20** (2014), no. 5, 805 – 824.

TABLE 2. Algorithms for degree d surfaces in $\mathbf{P}^3$ over $\mathbb{F}_p$ with six terms. Times in one AMD Ryzen 5 2600 core-seconds.

	Cubic Surfaces $(n=3,d=3)$		Quartic Surfaces $(n=3,d=4)$	
p	4.3	[CHK]	4.3	[CHK]
7	0.15	-	7.22	-
11	0.13	0.13	9.71	12.79
13	0.08	0.07	7.85	10.70
17	0.09	0.07	10.29	12.98
19	0.09	0.07	11.11	13.94
23	0.09	0.07	4.69	7.14
29	0.10	0.07	5.28	6.93
31	0.10	0.07	5.56	7.34

[AFP22] Sualeh Asif, Francesc Fité, and Dylan Pentland, *Computing l-polynomials of Picard curves from Cartier-Manin matrices*, Mathematics of Computation **91** (2022), 943–971.

[AKPW25] Asher Auel, Avinash Kulkarni, Jack Petok, and Jonah Weinbaum, *A census of cubic fourfolds over $\mathbb{F}_2$*, Math. Comp. **94** (2025), no. 354, 2089–2112.

[AKR10] Timothy G. Abbott, Kiran S. Kedlaya, and David Roe, *Bounding Picard numbers of surfaces using p-adic cohomology*, Arithmetics, geometry, and coding theory (AGCT 2005), Sémin. Congr., vol. 21, Soc. Math. France, Paris, 2010, pp. 125–159.

[AM77] Michael Artin and Barry Mazur, *Formal groups arising from algebraic varieties*, Ann. Sci. École Norm. Sup. (4) **10** (1977), no. 1, 87–131.

[AP08] Jeffrey D. Achter and Rachel Pries, *Monodromy of the p-rank strata of the moduli space of curves*, Int. Math. Res. Not. IMRN (2008), no. 15, Art. ID rnn053, 25.

[AP11] ———, *The p-rank strata of the moduli space of hyperelliptic curves*, Adv. Math. **227** (2011), no. 5, 1846–1872.

[Bao24] Chengyang Bao, *Computing Crystalline Deformation Rings via the Taylor-Wiles-Kisin Patching Method*, Phd thesis, University of Chigago, June 2024, Available at `https://doi.org/10.6082/uchicago.12410`.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, version 2.28-3 obtained via `https://magma.maths.usyd.edu.au/`.

[Ber96] Pierre Berthelot, *Cohomologie rigide et cohomologie rigide à supports propres. première partie*, Prépublication IRMAR 96-03, 1996, Unpublished preprint, 89 pp.

[Ber97] ———, *Finitude et pureté cohomologique en cohomologie rigide*, Invent. Math. **128** (1997), no. 2, 329–377, With an appendix in English by Aise Johan de Jong.

[BGHM26a] Ryan Batubara, Jack Garzella, Yongyuan Huang, and Maximus Mellberg, *DeRham.jl: Algorithms for point counts and cohomology of algebraic varieties*, `https://github.com/UCSD-computational-number-theory/DeRham.jl`, 2026, GitHub repository, accessed 2026-05-12.

[BGHM26b] ———, *NEWTON STRATA REALIZATION FOR HYPERSURFACES VIA EXPLICIT p-ADIC COHOMOLOGY, Zenodo Database*, January 2026, Available at `https://doi.org/10.5281/zenodo.18364679`.

[BGHM26c] ———, *NewtonExperiments.jl: Experiment scripts for computing zeta functions and newton polygons*, `https://github.com/UCSD-computational-number-theory/NewtonExperiments`, 2026, GitHub repository, accessed June 4, 2026.

[BGLM23] Jérémy Berthomieu, Stef Graillat, Dimitri Lesnoff, and Theo Mary, *Modular Matrix Multiplication on GPU for Polynomial System Solving*, ACM Commun. Comput. Algebra **57** (2023), no. 2, 35–38.

[BGM26] Ryan Batubara, Jack Garzella, and Maximus Mellberg, *GPUFiniteFieldMatrices.jl: Computational linear algebra library*, `https://github.com/UCSD-computational-number-theory/GPUFiniteFieldMatrices.jl`, 2026, GitHub repository, accessed 2026-05-12.

[BGP25] Ryan Batubara, Jack J Garzella, and Alex Pan, *K3 surfaces of any Artin-Mazur height over $\mathbb{F}_5$ and $\mathbb{F}_7$ via quasi-f-split singularities and GPU acceleration*, 2025, `https://arxiv.org/abs/2502.12428`.

[BGS07] Alin Bostan, Pierrick Gaudry, and Éric Schost, *Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator*, SIAM J. Comput. **36** (2007), no. 6, 1777–1806.

[BP] Daniel Bragg and Alex Perry, *Construction of Supersingular cubic fourfolds*, Personal Communication.

[CG72] C. Herbert Clemens and Phillip A. Griffiths, *The Intermediate Jacobian of the Cubic Threefold*, Annals of Mathematics **95** (1972), no. 2, 281–356.

[CH] Edgar Costa and David Harvey, *Controlled Reduction*, Unpublished Manuscript.

[CHK] Edgar Costa, David Harvey, and Kiran S. Kedlaya, *ToricControlledReduction*, `https://github.com/edgarcosta/ToricControlledReduction`.

[CHK19] ———, *Zeta functions of nondegenerate hypersurfaces in toric varieties via controlled reduction in p-adic cohomology*, Proceedings of the Thirteenth Algorithmic Number Theory Symposium, Open Book Ser., vol. 2, Math. Sci. Publ., Berkeley, CA, 2019, pp. 221–238.

[CHS22] Edgar Costa, David Harvey, and Andrew V. Sutherland, *Counting points on smooth plane quartics*, Research in Number Theory **9** (2022), no. 1, 1.

[Cosa] Edgar Costa, *controlledreduction*, `https://github.com/edgarcosta/controlledreduction`.

[Cosb] ———, *pycontrolledreduction*, `https://github.com/edgarcosta/pycontrolledreduction`.

[Cos15] ———, *Effective computations of Hasse-Weil zeta functions*, Ph.D. thesis, New York University, New York, 2015, Available at: `https://edgarcosta.org/assets/articles/EdgarCosta-PhDthesis.pdf`.

[CSS+10] Leandro F. Cupertino, Anderson P. Singulani, Cleomar P. da Silva, Marco Aurelio C. Pacheco, and Ricardo Farias, *LU Decomposition on GPUs: The Impact of Memory Access*, 2010 22nd International Symposium on Computer Architecture and High Performance Computing Workshops, 2010, pp. 19–24.

[DEF+25] Wolfram Decker, Christian Eder, Claus Fieker, Max Horn, and Michael Joswig (eds.), *The Computer Algebra System OSCAR: Algorithms and Examples*, 1 ed., Algorithms and Computation in Mathematics, vol. 32, Springer, 2025.

[Del74] Pierre Deligne, *La conjecture de Weil. I*, Inst. Hautes Études Sci. Publ. Math. (1974), no. 43, 273–307.

[Del80] ———, *La conjecture de Weil. II*, Inst. Hautes Études Sci. Publ. Math. (1980), no. 52, 137–252.

[dJO00] Aise Johan de Jong and Frans Oort, *Purity of the stratification by Newton polygons*, J. Amer. Math. Soc. **13** (2000), no. 1, 209–241.

[Dwo60] Bernard Dwork, *On the rationality of the zeta function of an algebraic variety*, American Journal of Mathematics **82** (1960), no. 3, 631–648.

[FHHJ17] Claus Fieker, William Hart, Tommy Hofmann, and Fredrik Johansson, *Nemo/hecke: Computer algebra and number theory packages for the julia programming language*, Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation (New York, NY, USA), ISSAC '17, ACM, 2017, pp. 157–164.

[FLI26] The FLINT Team, *FLINT: Fast Library for Number Theory*, 2026, Version 3.5.0, `https://flintlib.org`.

[Gri69] Phillip A. Griffiths, *On the periods of certain rational integrals. I, II*, Ann. of Math. (2) **90** (1969), 460–495; **90 (1969), 496–541**.

[Gro66] Alexander Grothendieck, *Formule de Lefschetz et rationalité des fonctions L*, Séminaire Bourbaki, années 1964/65–1965/66, exposés 277–312, Séminaire Bourbaki, no. 9, Société Mathématique de France, 1966, Exposé 279, pp. 41–55.

[Har77] Robin Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics, vol. No. 52, Springer-Verlag, New York-Heidelberg, 1977.

[Har07] David Harvey, *Kedlaya's algorithm in larger characteristic*, Int. Math. Res. Not. IMRN (2007), no. 22, Art. ID rnm095, 29.

[Har15] ———, *Computing zeta functions of arithmetic schemes*, Proc. Lond. Math. Soc. (3) **111** (2015), no. 6, 1379–1401.

[HMS16] David Harvey, Maike Massierer, and Andrew V. Sutherland, *Computing l -series of geometrically hyperelliptic curvesof genus three*, LMS Journal of Computation and Mathematics **19** (2016), no. A, 220–234.

[HS14] David Harvey and Andrew V. Sutherland, *Computing Hasse–Witt matrices of hyperelliptic curves in average polynomial time*, LMS Journal of Computation and Mathematics **17** (2014), no. A, 257–273.

[HS16a] ———, *Computing Hasse-Witt matrices of hyperelliptic curves in average polynomial time, II*, Frobenius Distributions:: Lang-Trotter and Sato-Tate Conjectures, Contemp. Math., vol. 663, Amer. Math. Soc., Providence, RI, 2016, pp. 127–149.

[HS16b] ———, *Computing Hasse-Witt matrices of hyperelliptic curves in average polynomial time, II*, Frobenius distributions: Lang-Trotter and Sato-Tate conjectures, Contemp. Math., vol. 663, Amer. Math. Soc., Providence, RI, 2016, pp. 127–147.

[IR83] Luc Illusie and Michel Raynaud, *Les suites spectrales associées au complexe de de Rham-Witt*, Publications Mathématiques de l'IHÉS **57** (1983), 73–212 (fr).

[Jos20] Kirti Joshi, *Crystalline aspects of geography of low dimensional varieties I: numerology*, European Journal of Mathematics **6** (2020), 1111–1175.

[JR03] Kirti Joshi and C. S. Rajan, *Frobenius splitting and ordinarity*, International Mathematics Research Notices **2003** (2003), no. 2, 109–121.

[Kat89] Kazuya Kato, *Logarithmic structures of Fontaine-Illusie*, Algebraic analysis, geometry, and number theory (Baltimore, MD, 1988), Johns Hopkins Univ. Press, Baltimore, MD, 1989, pp. 191–224.

[Ked01] Kiran S. Kedlaya, *Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology*, J. Ramanujan Math. Soc. **16** (2001), no. 4, 323–338.

[Ked06] ———, *Fourier transforms and p-adic 'Weil II'*, Compos. Math. **142** (2006), no. 6, 1426–1450.

[Ked12] ———, *Effective p-adic cohomology for cyclic cubic threefolds*, Computational algebraic and analytic geometry, Contemp. Math., vol. 572, Amer. Math. Soc., Providence, RI, 2012, pp. 127–171.

[KS16] Kiran S. Kedlaya and Andrew V. Sutherland, *A census of zeta functions of quartic K3 surfaces over* $\mathbb{F}_2$, LMS J. Comput. Math. **19** (2016), 1–11.

[KTY22] Tatsuro Kawakami, Teppei Takamatsu, and Shou Yoshikawa, *Fedder type criteria for quasi-F-splitting*, 2022, `https://arxiv.org/abs/2204.10076`.

[KW24a] Tatsuro Kawakami and Jakub Witaszek, *Higher F-injective singularities*, 2024.

[KW24b] Tatsuro Kawakami and Jakub Witaszek, *Higher F-injective singularities*, arXiv e-prints (2024), arXiv:2412.08887.

[Kyn22] Madeleine Kyng, *Computing zeta functions of algebraic curves using Harvey's trace formula*, Res. Number Theory **8** (2022), no. 4, Paper No. 100, 17.

[Laz04] Robert Lazarsfeld, *Positivity in algebraic geometry. I*, Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge. A Series of Modern Surveys in Mathematics [Results in Mathematics and Related Areas. 3rd Series. A Series of Modern Surveys in Mathematics], vol. 48, Springer-Verlag, Berlin, 2004, Classical setting: line bundles and linear series.

[LW54] Serge Lang and André Weil, *Number of Points of Varieties in Finite Fields*, American Journal of Mathematics **76** (1954), no. 4, 819–827.

[Maz72] B. Mazur, *Frobenius and the Hodge filtration*, Bull. Amer. Math. Soc. **78** (1972), 653–667.

[Maz73] ———, *Frobenius and the Hodge filtration (estimates)*, Ann. of Math. (2) **98** (1973), 58–95.

[MP] Linquan Ma and Thomas Polstra, *F-singularities, a commutative algebra approach*, `https://www.math.purdue.edu/~ma326/F-singularitiesBook.pdf`. Accessed: Sept 2024.

[Nvi24] Nvidia, *cuBLAS*, 2024, `https://developer.nvidia.com/cublas`. Accessed: Nov 2024.

[OSC26] *OSCAR – Open Source Computer Algebra Research system, Version 1.7.3*, 2026.

[Sch95] René Schoof, *Counting points on elliptic curves over finite fields*, vol. 7, 1995, Les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993), pp. 219–254.

[Shi02] Atsushi Shiho, *Crystalline fundamental groups. II. Log convergent cohomology and rigid cohomology*, J. Math. Sci. Univ. Tokyo **9** (2002), no. 1, 1–163.

[ST18] Ananth N. Shankar and Jacob Tsimerman, *Unlikely intersections in finite characteristic*, Forum Math. Sigma **6** (2018), Paper No. e13, 17.

[Sut20] Andrew V. Sutherland, *Counting points on superelliptic curves in average polynomial time*, ANTS XIV **4** (2020), 403–422.

[SV13] Steven Sperber and John Voight, *Computing zeta functions of nondegenerate hypersurfaces with few monomials*, LMS J. Comput. Math. **16** (2013), 9–44.

[Tae16] Lenny Taelman, *K3 surfaces over finite fields with given L-function*, Algebra & Number Theory **10** (2016), no. 5, 1133–1146 (English).

[Tui16] Jan Tuitman, *Counting points on curves using a map to* $\mathbf{P}^1$, Math. Comp. **85** (2016), no. 298, 961–981.

[Wei49] André Weil, *Numbers of solutions of equations in finite fields*, Bull. Amer. Math. Soc. **55** (1949), 497–508.

Ryan Batubara, Department of Mathematics, University of California San Diego, 2985 Muir Ln, La Jolla, CA 92093, USA
Email address: rbatubara@ucsd.edu

Jack J Garzella, Department of Mathematics, University of California San Diego, 2985 Muir Ln, La Jolla, CA 92093, USA
Email address: jgarzell@ucsd.edu

Yongyuan Huang, Department of Mathematics, University of California San Diego, 2985 Muir Ln, La Jolla, CA 92093, USA
Email address: yoh011@ucsd.edu

Maximus Mellberg, Department of Mathematics, University of California San Diego, 2985 Muir Ln, La Jolla, CA 92093, USA
Email address: mmellber@ucsd.edu