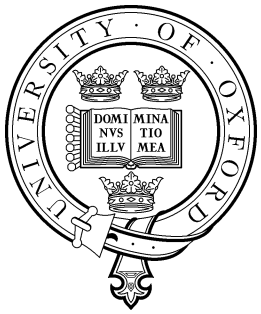
Practical improvements to the deformation method for point counting



Sebastian Pancratz

Hertford College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2012

Acknowledgements

I would like to start by expressing my gratitude towards my supervisor Alan Lauder. While encouraging me to work on the topic of this thesis he has given me a lot of freedom, allowing me to combine the theoretical aspects of this subject with my interest in programming.

In addition, I would also like to thank George Walker for his guidance and help throughout the initial period of my DPhil research. During the final two years, Jan Tuitman's joining of the research group accelerated my progress significantly.

In the context of my computational work, I would like thank Bill Hart and Fredrik Johansson, co-developers of the software package FLINT, for the tremendous amount of effort in producing the FLINT 2 series and the many stimulating discussions. I would also like extend my thanks to William Stein for the many invitations to workshops related to the software package SAGE.

Finally, I would like to thank the European Research Council, who supported this work financially through the grant awarded to Alan Lauder.

Abstract

Title Practical improvements to the deformation method for point counting

Name Sebastian Pancratz

College Hertford College

Degree Doctor of Philosophy in Mathematics

Term Trinity 2012

In this thesis we investigate practical aspects related to point counting problems on algebraic varieties over finite fields. In particular, we present significant improvements to Lauder's deformation method for smooth projective hypersurfaces, which allow this method to be successfully applied to previously intractable instances.

Part I is dedicated to the deformation method, including a complete description of the algorithm but focussing on aspects for which we contribute original improvements. In Chapter 3 we describe the computation of the action of Frobenius on the rigid cohomology space associated to a diagonal hypersurface; in Chapter 4 we develop a method for fast computations in the de Rham cohomology spaces associated to the family, which allows us to compute the Gauss–Manin connection matrix. We conclude this part with a small selection of examples in Chapter 6.

In Part II we present an improvement to Lauder's fibration method. We manage to resolve the bottleneck in previous computation, which is formed by so-called polynomial radix conversions, employing power series inverses and a more efficient implementation.

Finally, Part III is dedicated to a comprehensive treatment of the arithmetic in unramified extensions of $\mathbf{Q}_p$, which is connected to the previous parts where our computations rely on efficient implementations of p-adic arithmetic. We have made these routines available for others in FLINT as individual modules for p-adic arithmetic.

Contents

Part I

The deformation method

Chapter 1

Introduction to point counting

1.1 Varieties over finite fields and zeta functions

Let X denote an algebraic variety over the finite field $\mathbf{F}_q$ with $q = p^a$ elements. In particular, X is also defined over every algebraic extension $\mathbf{F}_{q^i}$, for $i \geq 0$, and we can let $N_i = |X(\mathbf{F}_{q^i})|$ denote the number of points of X over $\mathbf{F}_{q^i}$. The *zeta function* of X is then defined as the formal power series in $\mathbf{Q}[[T]]$,

$$Z(X,T) = \exp\Big(\sum_{i=0}^{\infty} \frac{N_i T^i}{i}\Big).$$

In the above form, the power series $Z(X,T)$ is represented by an infinite series and it is unclear whether this can be explicitly computed by an algorithm viz. a *finite* routine. This issue was resolved by Dwork [25] in 1960:

Theorem 1.1.1. Let X be an algebraic variety over the finite field $\mathbf{F}_q$. Then the zeta function $Z(X,T)$ is a quotient of two polynomials with integer coefficients.

Moreover, Bombieri [8] provided an explicit bound for the total degree of the rational function representing $Z(X,T)$. This implies that we can, at least in principle, compute the zeta function by computing a finite number of the quantities N_i by directly enumerating the points $|X\big(\mathbf{F}_{q^i}\big)|$. Dwork's theorem is included in a group of results known as the *Weil conjectures*, statements conjectured by Weil [65] in 1949 which have subsequently been proved by Dwork, Grothendieck, Deligne, et al. (see [34, Appendix C]).

We follow Kedlaya [45, Theorem 1.2.1] in the statement of the Weil conjectures:

Theorem 1.1.2. Let X be a separated scheme of finite type over the finite field $\mathbf{F}_q$. Then the zeta function of X is the power series representation of a rational function T. Moreover, if X is smooth and proper over $\mathbf{F}_q$, then there is a unique way to write

$$Z(X,T) = \prod_{i=0}^{2\dim(X)} P_i(T)^{(-1)^{i+1}} \tag{1.1}$$

for some polynomials $P_i(T) \in \mathbf{Z}[T]$ with $P_i(0) = 1$, satisfying the following conditions.

(i) We have

$$P_i(q^{-i}/T) = \pm q^{-i\deg(P_i)/2} T^{-\deg(P_i)} P_i(T),$$

with the sign being + whenever i is odd. In other words, the roots of P_i are invariant under the map $r \mapsto q^{-i}/r$, and if i is odd then the multiplicities of $\pm q^{-i/2}$ are even.

(ii) The complex roots of P_i all have absolute value $q^{-i/2}$.

(iii) If $X \cong \mathfrak{X}_{\mathbf{F}_q}$ for some smooth proper scheme $\mathfrak{X}$ over the local ring $R = \mathfrak{o}_{K,\mathfrak{p}}$, for some number field K and some prime ideal $\mathfrak{p}$ of $\mathfrak{o}_K$ with residue field $\mathbf{F}_q$, then for any embedding $K \hookrightarrow \mathbf{C}$,

$$\deg(P_i) = \dim_{\mathbf{C}} H^i\big((\mathfrak{X} \times_R \mathbf{C})^{\mathrm{an}}, \mathbf{C}\big).$$

In other words, $\deg(P_i)$ equals the i-th Betti number of $\mathfrak{X} \times_R \mathbf{C}$.

A discussion of the proof of this theorem is significantly beyond the scope of this thesis. As a starting point for further details, we refer the reader to Hartshorne [34, Appendix C] and Osserman [52].

Dwork's theorem on the rationality of the zeta function implies that $Z(X,T)$ is already defined by a finite initial segment of the sequence $(N_i)_{i\geq 0}$. In particular, the problem of computing the zeta function can be formulated as follows:

Problem 1.1.3. Given an algebraic variety X defined by a finite list of multivariate polynomials in $\mathbf{F}_q[x_0, \ldots, x_n]$, efficiently compute the rational function in $\mathbf{Q}(T)$ representing $Z(X, T)$.

1.2 Point counting algorithms

Computational routines addressing Problem 1.1.3 are known as *point counting* algorithms. Perhaps the most naive approach is to explicitly exhibit the points of X over $\mathbf{F}_{q^i}$ by an exhaustive search for sufficiently many $i \geq 0$, determining $Z(X, T)$. For example, in the case of an elliptic curve E over a finite field $\mathbf{F}_q$, the zeta function is given by

$$Z(E, T) = \frac{1 - a_q T + qT^2}{(1 - T)(1 - qT)}$$

where $a_q = q + 1 - |E(\mathbf{F}_q)|$. However, this approach is only feasible for the smallest of instances of Problem 1.1.3. More sophisticated algorithms have been developed, and we refer the reader to surveys by Chambert-Loir [14] on point counting algorithms in general and Kedlaya [42] on p-adic methods in particular. In the following, we only mention a few different approaches.

In 1985, Schoof [58] proposed an ℓ-adic algorithm for computing the zeta function of an elliptic curve E. The main idea behind this approach is to compute the trace $a_q \bmod \ell$ of the Frobenius endomorphism on $H^1_{\text{ét}}(E, \mathbf{F}_\ell)$ for sufficiently many primes ℓ and then determine the integer a_q using the Chinese remainder theorem. Schoof's algorithm has subsequently been improved by Atkin [2, 3] and Elkies [29], leading to a practical algorithm with runtime polynomial in $\log q$. It has also been generalised, for example to higher genus curves by Pila [54], however, the runtime of these algorithms remains exponential in the genus.

Another group of algorithms applicable primarily to elliptic curves is based on the *canonical lift* of a curve from $\mathbf{F}_q$ to the ring $\mathbf{Z}_q$, which is the ring of integers of the unique unramified extension of $\mathbf{Q}_p$ with residue field $\mathbf{F}_q$. The first of these algorithms is Satoh's method [56] presented in 2000, which has runtime polynomial in the extension degree $\log_p q$ but linear in the prime p.

Finally, there is a class of so-called p-adic algorithms based on Kedlaya's algorithm [41], which directly computes a p-adic approximation to the action of Frobenius on the rigid cohomology spaces associated to hyperelliptic curves in odd characteristic. For a fixed prime p, the runtime of Kedlaya's original algorithm is polynomial in the extension degree $\log_p q$ and the genus g. However, there have been a number of improvements. Harvey [35] improved the roughly linear dependence on the prime from p to $\sqrt{p}$. Denef and Vercauteren [20] included the case $p = 2$, and Castryck, Denef and Vercauteren [12] extended the algorithm to so-called nondegenerate curves.

Lauder's *deformation method* presented in [47] applies to smooth projective hypersurfaces, additionally making use of the theory of deformation due to Dwork [26]. The runtime of this algorithm is polynomial in the prime p, the extension degree $\log_p q$, and the genus g. Subsequently, Gerkmann [30] improved a number of p-adic bounds and developed a better implementation. Moreover, the practicality of the deformation method has also been demonstrated for various classes of curves. Hubrechts [36, 37] used this algorithm to compute the zeta function of hyperelliptic curves, Castryck, Hubrechts and Vercauteren [13] considered a slightly more general class of $C_{a,b}$-curves, and Tuitman [62] extended this work to nondegenerate curves.

Finally, we mention Lauder's *fibration method* introduced in [49], which proceeds recursively on the dimension of a hypersurface. This approach has been developed further by Walker [64].

1.3 This thesis

We present our results pertaining to the main goal of this thesis in Part I, which is to demonstrate the practicality of Lauder's deformation method for a wide class of smooth projective hypersurfaces. We build on the work of Lauder [47] and Gerkmann [30], resolving bottlenecks in previous computations by employing both theoretical advances as well as improved implementation details. In Chapter 2, we provide an overview of the deformation method and briefly revisit some of the underlying theoretical machinery and results. As there are no original contributions in this chapter we omit many technical details and do not include any proofs. The deformation method relies on the

computation of the action of Frobenius on a fibre in the family of hypersurfaces for which this computation is easy and we address this issue in Chapter 3. We present a novel algorithmic approach in the case of diagonal hypersurfaces, which is vastly superior to previous computations in practice, together with complementing theoretical results and a complexity analysis. In Chapter 4, we consider another major step in the deformation method, namely the computation of the Gauss–Manin connection matrix. Existing implementations [47, 30, 45] make use of Gröbner base computations for rational functions in order to carry out effective computations in de Rham cohomology spaces, which leads to poor practical performance and limits the deformation method to hypersurfaces defined by sparse polynomials. We reduce the impact of this step by treating it as an explicit finite-dimensional linear algebra problem. We collect the remaining steps of the deformation method in Chapter 5 as we do not present any original theoretical contributions. Our practical improvements are due to theoretical advances by Kedlaya and Tuitman [46] and improved implementation details. Finally, we present a number of examples in Chapter 6, hoping to convince the reader of the practical relevance of this work. In particular, in Section 6.7 we demonstrate that we can compute the zeta function of an arbitrary smooth projective quartic surface over a finite field of small characteristic in a matter of hours.

In Part II, we present a constant factor improvement to the radix conversion of polynomials, which is a key step and the main bottleneck in current implementations [49, 64] of Lauder's fibration method. Our brief review of the problem in Chapter 7 is followed by a presentation of suitable algorithms in Chapter 8. We observe a vast practical improvement when reconsidering an example due to Lauder in Chapter 9.

The practical improvements that we have observed in the first part of this thesis rely on an efficient implementation of the underlying p-adic arithmetic, which we address in Part III. In fact, we go beyond treating merely the functions utilised in the deformation method and develop a complete module for unramified p-adic arithmetic, which we have implemented in the C library FLINT [33].

Chapter 2

Introduction to the deformation method

2.1 Introduction

In this section we present the problem that we are concerned with in the first part of this thesis and carefully set up the notation that we are using throughout. We begin by giving a brief overview.

The aim of the deformation method as introduced by Lauder [48] following a paper by Dwork [26] is to compute the zeta function of a smooth hypersurface, which we call X_{t_1} for reasons that will become clear in a moment, in projective n-space over a finite field of characteristic p. This is facilitated by computing the action of the Frobenius morphism on the associated rigid cohomology spaces $H^\bullet_{\mathrm{rig}}(X_{t_1})$. The main idea is to embed this hypersurface in a family of smooth projective hypersurfaces X/S defined over a finite field k, where $S \subset \mathbf{A}^1_k(t)$, containing a fibre X_{t_0} for which it is easier to compute the Frobenius action. Lifting the family X/S to a family $\mathcal{X}/\mathcal{S}$ in characteristic zero, we can utilise a certain p-adic differential equation involving the Frobenius matrix and the algebraic Gauss–Manin connection. Moreover, due to a comparison theorem of Baldassarri and Chiarellotto [5], in this part of the algorithm we can work with the algebraic de Rham cohomology spaces $H^\bullet_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S})$ of the generic fibres, which are computationally more accessible. Solving this differential equation

and using the Frobenius action on $H^{\bullet}_{\mathrm{rig}}(X_{t_0})$ as the initial condition, by an appropriate evaluation we can compute the Frobenius matrix for $H^{\bullet}_{\mathrm{rig}}(X_{t_1})$ and hence recover the zeta function of the fibre X_{t_1}.

2.2 Families of smooth, projective hypersurfaces

For the remaining part of this thesis, we restrict ourselves to the case where the family X/S is defined over a prime field, $t_0 = 0$, and the initial fibre X_0 is diagonal. Specifically, we establish our set-up closely following Tuitman [62, §3.6] and using the same choice of resultant as Lauder [50, §2.3.2].

Notation 2.2.1. Let $\mathbf{F}_q$ be the finite field with $q = p^a$ elements, let $\mathbf{Q}_q$ denote the unique unramified extension of $\mathbf{Q}_p$ with residue field $\mathbf{F}_q$, and let $\mathbf{Z}_q$ denote its ring of integers.

We let $P \in \mathbf{Z}[t][x_0, \ldots, x_n]$ be a homogeneous polynomial of degree d with $n, d \geq 1$. We define $r(t) \in \mathbf{Z}[t]$ as the *Macaulay resultant* of the partial derivatives of P as in [51, Page 7, Chapter I.6]. We can now define two smooth $\mathbf{Z}_q$-schemes $\mathcal{X}$ and $\mathcal{S}$ as

$$\begin{aligned} \mathcal{X} &= \mathrm{Spec}\big(\mathbf{Z}_q[t, r(t)^{-1}][x_0, \ldots, x_n]/(P)\big), \\ \mathcal{S} &= \mathrm{Spec}\big(\mathbf{Z}_q[t, r(t)^{-1}]\big) \end{aligned}$$

together with the obvious smooth projective morphism $\mathcal{X} \to \mathcal{S}$. We also define the generic fibres $\mathfrak{X}$ and $\mathfrak{S}$ of $\mathcal{X}$ and $\mathcal{S}$ as

$$\begin{aligned} \mathfrak{X} &= \mathrm{Spec}\big(\mathbf{Q}_q[t, r(t)^{-1}][x_0, \ldots, x_n]/(P)\big), \\ \mathfrak{S} &= \mathrm{Spec}\big(\mathbf{Q}_q[t, r(t)^{-1}]\big), \end{aligned}$$

respectively. Moreover, we let X and S denote the reductions of $\mathcal{X}$ and $\mathcal{S}$ modulo the prime p. Finally, we define complements $\mathcal{U}$, $\mathfrak{U}$, and U of $\mathcal{X}$, $\mathfrak{X}$, and X, respectively, in the ambient projective n-space.

We assume that $r(0) \neq 0 \pmod{p}$ and hence that $\mathcal{X}_0$ is a smooth, projective hypersurface, which we will moreover assume to be diagonal, and that $t_1 \in S$ is a non-zero

point.

Notation 2.2.2. Furthermore, as an artefact of our particular choice of $\mathbf{Q}(t)$-vector space basis for the cohomology space $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ in Definition 4.3.3, we assume that $d \geq 2$ whenever n is odd and $d \geq 3$ whenever n is even.

Remark 2.2.3. By considering the diagonal fibre X_0, which is a smooth, projective hypersurface over $\mathbf{F}_p$, we see that our assumptions imply $p \nmid d$.

We will now briefly describe the actions of the Gauss–Manin connection and Frobenius on the vector spaces $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ and $H^n_{\mathrm{rig}}(U/S)$, following Tuitman [62, §3.4.2, §3.6.1] and Walker [64, §3.2.2.2]. As we will need this later, we begin by defining the notion of *overconvergent* power series:

Definition 2.2.4. Given indeterminates $z_1, \ldots, z_k$, the ring $\mathbf{Q}_q\langle z_1, \ldots, z_k\rangle^\dagger$ of *overconvergent* power series consists of power series of the form

$$\sum_{i_1,\ldots,i_k \geq 0} a_{i_1,\ldots,i_k} z_1^{i_1} \cdots z_k^{i_k}$$

such that there exist real numbers $\epsilon > 0$ and ν such that, for all $i_1, \ldots, i_k \geq 0$,

$$\mathrm{ord}_p(a_{i_1,\ldots,i_k}) \geq \epsilon(i_1 + \cdots + i_k) + \nu.$$

Now suppose that $(v_1, \ldots, v_b)$ is a basis for the finite-dimensional module $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ over $\mathbf{Q}_q[t, r(t)^{-1}]$. Moreover, we suppose this is also a basis for the module $H^n_{\mathrm{rig}}(U/S)$ over $\mathbf{Q}_q\langle t, r(t)^{-1}\rangle^\dagger$, slightly abusing notation by relying on a comparison theorem due to Baldassarri and Chiarellotto [5]. The Gauss–Manin connection

$$\nabla \colon H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S}) \to \Omega^1_{\mathfrak{S}} \otimes_{\mathbf{Q}_q[t,r(t)^{-1}]} H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$$

is Leibniz-linear, that is, it is additive and satisfies the Leibniz rule

$$\nabla(av) = da \otimes v + a\nabla(v)$$

for all $a \in \mathbf{Q}_q[t, r(t)^{-1}]$ and $v \in H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$, and we represent its action by a matrix M

over $\mathbf{Q}_q[t, r(t)^{-1}]$ with

$$\nabla(v_j) = dt \otimes \sum_{i=1}^{b} M_{ij} v_i.$$

As the connection ∇ can be defined solely in algebraic terms, it also induces an action on $H^n_{\mathrm{rig}}(U/S)$.

We note that in the examples we consider the polynomial P is a polynomial over the integers and the family X is defined over base field $\mathbf{F}_p$ already, and hence that the entries of M can be chosen to lie in $\mathbf{Q}[t, r(t)^{-1}]$. We present further details of this construction and give an explicit routine for the computation of the matrix M in Chapter 4.

Now let F_p denote the standard lift of the p-th power Frobenius from $\mathbf{F}_q$ to $\mathbf{Q}_q[t]$, which is equal to $\sigma \in \mathrm{Gal}(\mathbf{Q}_q/\mathbf{Q}_p)$ on $\mathbf{Q}_q$ and is extended to $\mathbf{Q}_q[t]$ by sending t to t^p, where σ is the unique lift of the p-th power Frobenius map to $\mathbf{Q}_q$. The canonical Frobenius map $\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$

$$\mathcal{F}_p \colon F_p^* H^n_{\mathrm{rig}}(U/S) \to H^n_{\mathrm{rig}}(U/S)$$

is σ-semilinear as a map on $H^n_{\mathrm{rig}}(U/S)$. We represent the action of $p^{-1}\,\mathcal{F}_p$ by a matrix Φ such that

$$p^{-1}\,\mathcal{F}_p(v_j) = \sum_{i=1}^{b} \Phi_{ij} v_i,$$

where the entries of Φ are in fact overconvergent power series (as proven by Berthelot [7, Théorème 5]), that is, they lie in the ring $\mathbf{Q}_q\langle t, r(t)^{-1}\rangle^\dagger$ defined as $\mathbf{Q}_q\langle t, z\rangle^\dagger/(1-zr(t)^{-1})$.

This is explained further e.g. by Lauder [49, §3.5]. It means that the matrix Φ converges on $\mathbf{P}^1(\mathbf{Q}_q)$ with disks of radius strictly less than 1 removed around the roots of $r(t)$ and ∞. Thus, the entries of Φ can be approximated modulo any given power of p by rational functions with denominators some power of $r(t)$. We provide explicit estimates for this convergence in Section 5.2.

As we do not present any original, theoretical improvements to this part of the deformation method, that is, the computation of the action of Frobenius on the generic fibre, we provide fewer details.

Finally, we note that the Gauss–Manin connection and the action of Frobenius satisfy

the following commutative diagram,

$$\begin{array}{ccc} H^n_{\mathrm{rig}}(U/S) & \xrightarrow{\quad\nabla\quad} & \Omega^1_S \otimes H^n_{\mathrm{rig}}(U/S) \\ \Big\downarrow \mathcal{F}_p & & \Big\downarrow dF_p \otimes \mathcal{F}_p \\ H^n_{\mathrm{rig}}(U/S) & \xrightarrow{\quad\nabla\quad} & \Omega^1_S \otimes H^n_{\mathrm{rig}}(U/S) \end{array}$$

where $dF_p(dt) = d(t^p) = pt^{p-1}dt$. This implies that the matrices M and Φ satisfy the p-adic differential equation given by

$$\Big(\frac{d}{dt} + M\Big)\Phi = pt^{p-1}\Phi\sigma(M).$$

Further details regarding this can be found in [30, §5].

Finally, we will be able to compute the zeta function of the hypersurface X_{t_1} from the action of $q^{-1}\,\mathcal{F}_q$ on $H^n_{\mathrm{rig}}(U_{t_1})$, using the following theorem:

Theorem 2.2.5. The zeta function of the hypersurface X_{t_1} is of the form

$$Z(X_{t_1}, T) = \frac{p(T)^{(-1)^n}}{(1-T)(1-qT)\cdots(1-q^{n-1}T)},$$

where $p(T) = \det\big(1 - Tq^{-1}\,\mathcal{F}_q\,|H^n_{\mathrm{rig}}(U_{t_1})\big)$ is a polynomial defined over the integers.

While the result is well known in the area, a convenient reference is [30, Theorem 3.1].

2.3 Outline of the algorithm

We now provide an outline of the deformation method. The list below includes nearly full details of the computation, omitting only the specific choice of precisions N_0, N_1, N_2, N_3, N_3', N_4, and K_1, K_2.These are provided and justified in Chapter 5.

Step *I*. Let Φ_0 denote the matrix for the action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_0)$, which has entries in $\mathbf{Q}_p$. We compute an approximation to p-adic precision N_4.

Step *II*. We compute the matrix M, which denotes the connection matrix on $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ and has entries in $\mathbf{Q}(t)$.

Step III. Let C denote the matrix over $\mathbf{Q}[[t]]$ for the local expansion of the solution to the p-adic differential equation $\big(\frac{d}{dt} + M\big)C = 0$. We compute an approximation modulo t^{K_2} to p-adic precision N_3. We also compute an approximation for the matrix $C(t^p)^{-1}$ modulo t^{K_2} to p-adic precision N_3'.

Step IV. Let $\Phi(t)$ denote the matrix for the action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_t)$, which satisfies the equation $\Phi(t) = C(t)\Phi_0 C(t^p)^{-1}$ with $\Phi(0) = \Phi_0$ and has entries in $\mathbf{Q}_p\langle t, r(t)^{-1}\rangle^{\dagger}$. We compute an approximation modulo t^{K_2} to p-adic precision N_2.

Step V. Let Φ_{t_1} denote the matrix for the action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_{t_1})$, which has entries in $\mathbf{Q}_q$. Firstly, we compute an approximation to $\Psi(t) = r(t)^{K_1}\Phi(t)$ modulo t^{K_2} to p-adic precision N_2. We can then compute an approximation for Φ_{t_1} to p-adic precision N_2 as $r(\hat{t}_1)^{-K_1}\Psi(\hat{t}_1)$.

Step VI. Let $\Phi_{t_1}^{(q)}$ denote the matrix for the action of $q^{-1}\,\mathcal{F}_q$ on $H^n_{\mathrm{rig}}(U_{t_1})$, which satisfies $\Phi_{t_1}^{(q)} = \Phi_{t_1}\sigma(\Phi_{t_1})\cdots\sigma^{a-1}(\Phi_{t_1})$ and has entries in $\mathbf{Q}_q$. We compute an approximation to this matrix with p-adic precision N_1.

Step VII. Let $p(T) = \det\big(1 - Tq^{-1}\,\mathcal{F}_q\,|H^n_{\mathrm{rig}}(U_{t_1})\big)$, which is an integer polynomial. We recover this exactly by computing an approximation to the reverse characteristic polynomial of $\Phi_{t_1}^{(q)}$ to p-adic precision N_0.

Chapter 3

Frobenius on the diagonal fibre

In this chapter we discuss the action of Frobenius on the diagonal fibre. We first describe the general method based on an explicit formula by Dwork [28, §4], following the expositions in the work of Lauder [48, §6] and Gerkmann [30, §4.4]. Then, we present a new improvement in Theorem 3.2.2, showing that a certain expression is defined already over $\mathbf{Q}_p$ instead of a ramified extension as observed by Gerkmann. Furthermore, Theorems 3.2.9 and 3.2.13 state the expression lies in $\mathbf{Z}_p$ and provide computationally suitable reformulations. Our presentation of an explicit algorithm for the computation of the action of Frobenius is followed by Theorems 3.3.4 and 3.3.5 providing a complexity analysis. To conclude this chapter, we illustrate the practical relevance of these improvements with three example computations.

3.1 Introduction

In order to simplify our notation, we temporarily suppress the earlier setup of a family of hypersurfaces. Thus, we consider a smooth, projective, diagonal hypersurface $\mathcal{X}$ over $\mathbf{Z}_p$ given by a polynomial $P \in \mathbf{Z}[x_0, \ldots, x_n]$ of homogeneous degree d and its reduction X over $\mathbf{F}_p$ given by

$$\tilde{P}(x_0, x_1, \ldots, x_n) = a_0 x_0^d + a_1 x_1^d + \cdots + a_n x_n^d = 0$$

where $a_0, a_1, \ldots, a_n \in \mathbf{F}_p^\times$ and $p \nmid d$. We also define the generic fibre $\mathfrak{X}$ as well as affine complements $\mathcal{U}$, U and $\mathfrak{U}$ of $\mathcal{X}$, X and $\mathfrak{X}$, respectively. We choose the following basis $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ for $H^n_{\mathrm{dR}}(\mathfrak{U})$ where, for $k = 1, \ldots, n$,

$$B_k = \{x^i : i \in \mathbf{N}_0^{n+1}, |i| = kd - (n+1) \text{ and } 0 \leq i_j < d-1, \text{ for all } j\},$$
$$\mathcal{B}_k = \{x^i \Omega / P^k : x^i \in B_k\},$$

with $x^i = x_0^{i_0} \cdots x_n^{i_n}$, and Ω is the n-form defined by

$$\Omega = \sum_{i=0}^{n} (-1)^i x_i dx_0 \wedge \cdots \wedge \widehat{dx_i} \wedge \cdots \wedge dx_n.$$

Further details of this construction can be found in Section 4.2. Finally, we recall the condition that $d \geq 2$ whenever n is odd and $d \geq 3$ whenever n is even. Our goal is to compute the matrix Φ representing the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U) \cong H^n_{\mathrm{dR}}(\mathfrak{U})$ to p-adic precision N.

We now describe an explicit formula for the $\dim H^n_{\mathrm{dR}}(\mathfrak{U})$ non-zero coefficients of the matrix Φ due to Dwork [28], see e.g. Lauder [48, §6.1]. We work over the ramified extension $\mathbf{Q}_p(\pi)$ where $\pi^{p-1} = -p$, and we normalise the valuation such that $\mathrm{ord}_p(\pi) = (p-1)^{-1}$.

Let $u = (u_0, \ldots, u_n)$ and $v = (v_0, \ldots, v_n)$ be tuples such that $x^u, x^v \in B_1 \cup \cdots \cup B_n$, and let u' and v' denote integers such that $du' = \sum_{i=0}^n (u_i + 1)$ and similarly for v'. Finally, for $m \geq 0$, let λ_m denote the coefficient of z^m in the expansion of $\exp \pi(z - z^p)$ and define products $(w)_r = \prod_{j=0}^{r-1}(w+j)$ for $w \in \mathbf{Q}$ and $r \geq 0$. We introduce terms $\alpha_{u,v}$,

$$\alpha_{u,v} = \pi^{v'-u'} \prod_{i=0}^{n} \sum_{m,r} \lambda_m (u_i/d)_r (-1)^r \pi^{-r} \hat{a}_i^{m-r} \tag{3.1}$$

where $\hat{a}_i$ is the Teichmüller lift of $a_i \in \mathbf{F}_p$ to $\mathbf{Q}_p$ and the summation indices $m, r \geq 0$ satisfy $pu_i - v_i = d(m - pr)$.

Theorem 3.1.1. Continuing with the notation set out above, let ω_1 and ω_2 denote the forms in $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ corresponding to u and v, respectively. Then $p^{-1}\mathcal{F}_p(\omega_1) = 0$

unless, for all $i = 0, \ldots, n$, $p(u_i + 1) \equiv v_i + 1 \pmod{d}$. In this case,

$$p^{-1}\,\mathcal{F}_p(\omega_1) = (-1)^{u'+v'}\frac{(v'-1)!}{(u'-1)!}p^n\alpha_{u+1,v+1}^{-1}\omega_2$$

where $u + 1 = (u_0 + 1, \ldots, u_n + 1)$ and similarly for $v + 1$.

Proof. See Dwork [28, §4] for Dwork's original work, or Lauder [48, §6.1]. Both references also treat the more general case when $\mathcal{X}$ is a smooth, projective, diagonal hypersurface over $\mathbf{Z}_q$, for q a prime power. □

3.2 Improvements

From the previous description, it appears that this computation genuinely has to take place over the extension field $\mathbf{Q}_p(\pi)$. This is, however, not the case as we will show in this section. We are able to show that the terms $\alpha_{u+1,v+1}$ are p-adic integers and provide expressions for these that are more suitable from a computational perspective.

Lemma 3.2.1. Let $\pi^{p-1} = -p$ and, for $m \geq 0$, let λ_m be the coefficient of z^m in the power series expansion of $\exp \pi(z - z^p)$ in $\mathbf{Q}_p[[z]]$. Then

$$\pi^{-(m \bmod (p-1))}\lambda_m = (-1)^{\lfloor m/(p-1)\rfloor}\sum_{k=0}^{\lfloor m/p\rfloor} p^{\lfloor m/(p-1)\rfloor - k}\frac{1}{(m-pk)!k!},$$

where $m \bmod (p-1)$ denotes the remainder of m upon Euclidean division by $p - 1$.

Proof. From

$$\exp \pi(z - z^p) = \sum_{n=0}^{\infty}\frac{\pi^n}{n!}(z - z^p)^n = \sum_{n=0}^{\infty}\frac{\pi^n}{n!}\sum_{k=0}^{n}\binom{n}{k}(-1)^k z^{n+k(p-1)}$$

we find that

$$\begin{aligned}\lambda_m &= \sum_{n=0}^{\infty}\frac{\pi^n}{n!}\sum_{\substack{0\leq k\leq n\\ n+k(p-1)=m}}\binom{n}{k}(-1)^k \\ &= \sum_{\substack{\lceil m/p\rceil \leq n\leq m\\ n\equiv m \pmod{p-1}}}\frac{\pi^n}{n!}\binom{n}{(m-n)/(p-1)}(-1)^{(m-n)/(p-1)}\end{aligned}$$

To simplify this further, let $m = q(p-1)+r$ and $n = q'(p-1)+r'$, noting that $r = r'$ and also setting $k = (m-n)/(p-1) = q - q'$,

$$\begin{aligned}\lambda_m &= \sum_{\substack{\lceil m/p\rceil \le n \le m \\ n \equiv m \pmod{(p-1)}}} \pi^r p^{q'} (-1)^{q'+k} \frac{1}{(n-k)!k!} \\ &= (-1)^q \pi^r \sum_{k=0}^{\lfloor m/p \rfloor} p^{q-k} \frac{1}{(m-pk)!k!}.\end{aligned}$$

The expression for $\pi^{-(m \bmod (p-1))}\lambda_m$ now follows. □

Theorem 3.2.2. Continuing with the notation set up in this chapter, let $u, v \in \mathbf{Z}^{n+1}$ be such that $x^u, x^v \in B_1 \cup \cdots \cup B_n$ and satisfy, for all i, $p(u_i+1) \equiv v_i + 1 \pmod{d}$. Then

$$\alpha_{u+1,v+1} = (-p)^{u'} \prod_{i=0}^{n} \hat{a}_i^{(p(u_i+1)-(v_i+1))/d} \sum_{m,r} \Big(\frac{u_i+1}{d}\Big)_r \sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{r-k}}{(m-pk)!k!}.$$

where $m, r \ge 0$ satisfy $p(u_i+1) - (v_i+1) = d(m-pr)$.

Proof. We begin from the definition of $\alpha_{u+1,v+1}$ and use Lemma 3.2.1 to find that $\alpha_{u+1,v+1}$ is equal to

$$\pi^{v'-u'} \prod_{i=0}^{n} \sum_{m,r} (-1)^{\lfloor m/(p-1)\rfloor} \pi^{m \bmod (p-1)} \Big(\sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{\lfloor m/(p-1)\rfloor - k}}{(m-pk)!k!} \Big) \Big(\frac{u_i+1}{d}\Big)_r (-1)^r \pi^{-r} \hat{a}_i^{m-r}.$$

We now write $m = \lfloor m/(p-1)\rfloor\,(p-1) + \big(m \bmod (p-1)\big)$ and simplify by $\pi^{p-1} = -p$, so this expression is equal to

$$\pi^{v'-u'} \prod_{i=0}^{n} \sum_{m,r} \pi^{m-r} \Big(\sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{-k}}{(m-pk)!k!} \Big) \Big(\frac{u_i+1}{d}\Big)_r (-1)^r \hat{a}_i^{m-r}.$$

Using that $m - r = (p-1)r + \big(p(u_i+1) - (v_i+1)\big)/d$, we arrive at

$$\pi^{v'-u'} \prod_{i=0}^{n} \pi^{(p(u_i+1)-(v_i+1))/d} \hat{a}_i^{(p(u_i+1)-(v_i+1))/d} \sum_{m,r} \Big(\sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{r-k}}{(m-pk)!k!} \Big) \Big(\frac{u_i+1}{d}\Big)_r$$

and finally

$$(-p)^{u'}\prod_{i=0}^{n}\hat{a}_i^{(p(u_i+1)-(v_i+1))/d}\sum_{m,r}\Big(\frac{u_i+1}{d}\Big)_r\sum_{k=0}^{\lfloor m/p\rfloor}\frac{p^{r-k}}{(m-pk)!k!},$$

as required. □

In particular, Theorem 3.2.2 implies that $\alpha_{u+1,v+1}\in\mathbf{Q}_p$. Our next aim is to obtain expressions for $\alpha_{u+1,v+1}$ which are defined over $\mathbf{Z}_p$. But before progressing, we collect a few intermediate results which we will refer to later.

Proposition 3.2.3. Let $u,v\in B_1\cup\cdots\cup B_n$, $i\in\{0,\dots,n\}$ and let $m,r\geq 0$ satisfy $d(m-pr)=p(u_i+1)-(v_i+1)$. Then

$$0\leq m-pr\leq\frac{p(d-1)-1}{d}.$$

In particular, $m=m(r)=pr+d^{-1}\big(p(u_i+1)-(v_i+1)\big)\geq 0$.

Proof. This can be easily verified using that $0\leq u_i,v_i\leq d-2$ and $m-pr\in\mathbf{Z}$. □

Proposition 3.2.4. Let $u,v\in B_1\cup\cdots\cup B_n$, $i\in\{0,\dots,n\}$ and let $m,r\geq 0$ satisfy $d(m-pr)=p(u_i+1)-(v_i+1)$. Then

$$r-\left\lfloor\frac{m}{p}\right\rfloor\geq 0.$$

Proof. Using the previous proposition,

$$r-\left\lfloor\frac{m}{p}\right\rfloor=-\left\lfloor\frac{m-pr}{p}\right\rfloor\geq-\left\lfloor\frac{p(d-1)-1}{pd}\right\rfloor=-1+\left\lceil\frac{p+1}{pd}\right\rceil=0$$

as $p\geq 2$, $d\geq 2$. □

Proposition 3.2.5. For all integers $u,d\geq 1$ and $r\geq 0$ with $p\nmid d$,

$$\operatorname{ord}_p\Big(\frac{u}{d}\Big)_r\geq\frac{r}{p-1}-\big\lfloor\log_p(r)+1\big\rfloor.$$

Proof. Let $s_p(r)$ denote the sum of digits in the p-adic expansion of r and observe that $s_p(r) \leq (p-1)\left\lfloor \log_p(r)+1 \right\rfloor$. Using the fact that $\mathrm{ord}_p\big((u/d)_r\big) \geq \mathrm{ord}_p(r!)$ from Clark [15, Page 265, Case 3] it follows that

$$\mathrm{ord}_p\Big(\frac{u}{d}\Big)_r \geq \mathrm{ord}_p(r!) = \frac{r-s_p(r)}{p-1} \geq \frac{r}{p-1} - \left\lfloor \log_p(r)+1 \right\rfloor$$

as required. □

3.2.1 The case $p = 2$

We first note that in the case when $p = 2$, the expression for λ_m in Lemma 3.2.1 can be simplified.

Corollary 3.2.6. Let $p = 2$, let $\pi^{p-1} = -p$ and, for $m \geq 0$, let λ_m be the coefficient of z^m in the power series expansion of $\exp \pi(z - z^p)$ in $\mathbf{Q}_p[[z]]$. Then

$$\lambda_m = (-1)^m \sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{m-k}}{(m-pk)!k!}.$$

Lemma 3.2.7. Let $p = 2$ and define a sequence $\big(\mu_m^{(2)}\big)$ by

$$\mu_m^{(2)} = \sum_{k=0}^{\lfloor m/2 \rfloor} \frac{2^{\lfloor 3m/4 \rfloor - \nu_m - k}}{(m-2k)!k!}$$

where ν_m is equal to one whenever $m = 3, 7$ and zero otherwise, and we more simply write $\mu_m = \mu_m^{(2)}$ whenever this does not cause confusion. Then $\mu_m \in \mathbf{Z}_p$ for all $m \geq 0$.

Proof. In the two cases $m = 3, 7$ we explicitly compute the values of μ_m as $4/3$ and $232/315$. Now suppose that $m \neq 3, 7$. From Corollary 3.2.6 we obtain that

$$\mathrm{ord}_2\big(\mu_m\big) = \lfloor 3m/4 \rfloor - m + \mathrm{ord}_2(\lambda_m).$$

Using the bound $\mathrm{ord}_p(\lambda_m) \geq \big((p-1)/p^2\big)m$ from Dwork [27, Pages 55–57], we obtain the lower bound

$$\mathrm{ord}_2\big(\mu_m\big) \geq \lfloor 3m/4 \rfloor - m + \left\lceil \frac{m}{4} \right\rceil = 0.$$ □

Remark 3.2.8. The expression μ_m, $m \neq 3, 7$, can be computed efficiently modulo $p^{\tilde{N}}$ via

$$\mu_m = \frac{2^{\lfloor 3m/4 \rfloor}}{m!} \sum_{k=0}^{\lfloor m/2 \rfloor} \frac{m!}{2^k (m-2k)! k!}$$

using only one p-adic inversion. We observe that the summands are integers of size $\mathcal{O}(m \log m)$ bits, which allows us to avoid performing intermediate reductions modulo $p^{\tilde{N}}$.

Theorem 3.2.9. Let $p = 2$. Continuing with the notation set up in this chapter, let $u, v \in \mathbf{Z}^{n+1}$ be such that $x^u, x^v \in B_1 \cup \cdots \cup B_n$ and satisfy, for all i, $p(u_i + 1) \equiv v_i + 1 \pmod{d}$. Then $\alpha_{u+1,v+1}$ can be expressed as

$$\alpha_{u+1,v+1} = (-2)^{u'} \prod_{i=0}^{n} \sum_{m,r} \Big(\frac{u_i + 1}{d} \Big)_r 2^{-\lfloor (m+1)/4 \rfloor + \nu_m} \mu_m, \tag{3.2}$$

and $\alpha_{u+1,v+1}$ is a p-adic integer.

Proof. The expression for $\alpha_{u+1,v+1}$ is an immediate consequence of Theorem 3.2.2 and Lemma 3.2.7, together with Proposition 3.2.3 implying $0 \leq m - 2r \leq 1$ and hence $r = \lfloor m/2 \rfloor$. It remains to prove that $\alpha_{u+1,v+1}$ is a p-adic integer. Following Lemma 3.2.7, it suffices to show that the valuation of the factor

$$\Big(\frac{u_i + 1}{d} \Big)_r 2^{-\lfloor (m+1)/4 \rfloor + \nu_m}$$

in each summand is non-negative. From the proof of Proposition 3.2.5 we obtain that

$$\operatorname{ord}_p \bigg(\Big(\frac{u_i + 1}{d} \Big)_r 2^{-\lfloor (m+1)/4 \rfloor + \nu_m} \bigg) \geq \operatorname{ord}_p \Big(\Big\lfloor \frac{m}{2} \Big\rfloor ! \Big) - \Big\lfloor \frac{m+1}{4} \Big\rfloor + \nu_m. \tag{3.3}$$

Applying Proposition 3.2.5, we see that this is bounded below by

$$\Big\lfloor \frac{m}{2} \Big\rfloor - \Big\lfloor \frac{m+1}{4} \Big\rfloor - \lfloor \log_2 m \rfloor$$

which is non-negative whenever $m \geq 12$. In the remaining cases $m = 0, \ldots, 11$, we explicitly verify that the lower bound in (3.3) is non-negative. □

Remark 3.2.10. We observe that in equation (3.2) the exponent $-\lfloor (m+1)/4 \rfloor + \nu_m$ is non-positive for each value $m \geq 0$, which suggests the computational question of how to determine the sequence of factors

$$\Big(\frac{u_i + 1}{d}\Big)_r 2^{-\lfloor (m+1)/4 \rfloor + \nu_m}$$

to p-adic precision $\tilde{N}$ without intermediate precision loss. To simplify our notation slightly, suppose we wish to compute the sequence

$$f_r = 2^{-\lfloor (m+1)/4 \rfloor + \nu_m} \prod_{j=0}^{r-1} (u + jd)$$

for all $r \geq 0$, where $m(r) = m(0) + 2r$, $m(0) \in \{0, 1\}$, u is a positive integer and d is a positive odd integer. Explicitly, we find that this sequence satisfies

$$\begin{aligned}
f_0 &= 1, \\
f_1 &= u, \\
f_5 &= \begin{cases} \frac{1}{4}\big(u(u+d)(u+2d)(u+3d)(u+4d)\big) & \text{if } m(0) = 0, \\ \frac{1}{8}\big(u(u+d)(u+2d)(u+3d)(u+4d)\big) & \text{otherwise,} \end{cases} \\
f_r &= f_{r-2} \frac{(u + (r-2)d)(u + (r-1)d)}{2}
\end{aligned}$$

for all remaining $r \geq 0$. Since the product in the numerator is an even integer, it is clear that this computation can be carried out without precision loss despite reducing previous values of f_r modulo $p^{\tilde{N}}$.

3.2.2 The case of odd primes

Lemma 3.2.11. Let $p \geq 3$ be an odd prime and define a sequence $\big(\mu_m^{(p)}\big)$ by

$$\mu_m^{(p)} = \sum_{k=0}^{\lfloor m/p \rfloor} \frac{p^{\lfloor m/p \rfloor - k}}{(m - pk)!k!},$$

where we write $\mu_m = \mu_m^{(p)}$ when the prime can be identified from the context. Then $\mu_m \in \mathbf{Z}_p$ for all $m \geq 0$.

Proof. It is clear that $\mu_m \in \mathbf{Q}$. From Lemma 3.2.1 we observe that

$$\lambda_m = \pi^m p^{-\lfloor m/p \rfloor} \mu_m.$$

Using the bound $\mathrm{ord}_p(\lambda_m) \geq \big((p-1)/p^2\big)m$ from Dwork [27, Pages 55–57], it follows that

$$\mathrm{ord}_p(\mu_m) \geq \frac{p-1}{p^2} m + \left\lfloor \frac{m}{p} \right\rfloor - \frac{m}{p-1}$$

Let us write $m = qp + r$ with $0 \leq r \leq p-1$. As the valuation of μ_m is an integer, it suffices to show that, for $q \geq 0$,

$$\frac{p-1}{p} q + q - \frac{qp+p-1}{p-1} > -1,$$

which is equivalent to

$$\Big(\frac{p-1}{p} + 1 - \frac{p}{p-1}\Big) q > 0.$$

Dividing by q and multiplying by $p(p-1)$, we obtain the equivalent condition

$$p^2 - 3p + 1 > 0$$

which holds true provided that $p \geq 3$. □

Remark 3.2.12. The expression μ_m can be computed efficiently modulo $p^{\tilde{N}}$ via

$$\mu_m = \frac{p^{\lfloor m/p \rfloor}}{m!} \sum_{k=0}^{\lfloor m/p \rfloor} \frac{m!}{p^k (m-pk)! k!}$$

using only one p-adic inversion. We observe that the summands are integers of size $\mathcal{O}(m \log m)$ bits, which allows us to avoid performing intermediate reductions modulo $p^{\tilde{N}}$.

Theorem 3.2.13. Let $p \geq 3$. Continuing with the notation set up in this chapter, let $u, v \in \mathbf{Z}^{n+1}$ be such that $x^u, x^v \in B_1 \cup \cdots \cup B_n$ and satisfy, for all i, $p(u_i + 1) \equiv v_i + 1$

(mod d). Then

$$\alpha_{u+1,v+1} = (-p)^{u'} \prod_{i=0}^{n} \hat{a}_i^{(p(u_i+1)-(v_i+1))/d} \sum_{m,r} \Big(\frac{u_i+1}{d}\Big)_r p^{r-\lfloor m/p \rfloor} \mu_m \tag{3.4}$$

where $m, r \geq 0$ satisfy $p(u_i+1) - (v_i+1) = d(m-pr)$. In particular, $\alpha_{u+1,v+1} \in \mathbf{Z}_p$.

Proof. This follows from Theorem 3.2.2, Proposition 3.2.4 and Lemma 3.2.11. □

3.3 Description of the algorithm

Up to this point, the double sum over m, r in our expressions for $\alpha_{u+1,v+1}$ has been an infinite sum. We now present a convergence result, which will allow us to derive a finite expression for $\alpha_{u+1,v+1}$ modulo $p^{\tilde{N}}$.

Lemma 3.3.1. Given integers $b, c \geq 2$ and defining $x = c + \log_b c + 1$ we have that, for all real numbers $y \geq x$,

$$y - \log_b y \geq c.$$

Proof. We first note that the function $y \mapsto y - \log_b y$ is increasing for $y \geq 2$ because it has derivative $1 - \log_b(e)/y > 0$. Thus, it suffices to verify the result for x. Indeed, as $c \geq 2$ we have that

$$\log_b c + 1 \leq c$$

and hence

$$c + \log_b c + 1 \leq b^{\log_b c + 1}$$

Taking logarithms to base b and rearranging, we obtain

$$c + \log_b c + 1 - \log_b\big(c + \log_b c + 1\big) - c \geq 0$$

as required. □

Proposition 3.3.2. Continuing with the notation set up in this chapter, in order to compute $\alpha_{u+1,v+1} \bmod p^{\tilde{N}}$ it suffices to restrict the inner sum to pairs $m, r \geq 0$ such that $m \leq M = M(\tilde{N}, p)$ where

$$M = \left\lfloor p^2 \Big(\frac{\tilde{N}}{p-1} + \log_p \Big(\frac{\tilde{N}}{p-1} + 2 \Big) + 3 \Big) \right\rfloor .$$

Proof. This follows from [48, §6.2] and Lemma 3.3.1. □

Finally, we describe how to compute the matrix Φ representing the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U)$ using our previous results.

Proposition 3.3.3. The valuation of $(u'-1)!\alpha_{u+1,v+1}$ is bounded from above by

$$\mathrm{ord}_p\big((u'-1)!\alpha_{u+1,v+1}\big) \leq \mathrm{ord}_p\big((n-1)!\big) + n + \delta$$

where $\delta = \mathrm{ord}_p\big((n-1)!\big) + (n+1)\left\lfloor \log_p(n-1) \right\rfloor$.

Proof. Recall from Gerkmann [30, Lemma 3.3] that the valuations of the entries of the matrix Φ are bounded from below by $-\delta$. Thus, by Theorem 3.1.1,

$$-\delta \leq \mathrm{ord}_p\big((v'-1)!\big) + n - \mathrm{ord}_p\big((u'-1)!\alpha_{u+1,v+1}\big)$$

Noting that $dv' = \sum_{i=0}^{n}(v_i+1) \leq nd$ and hence $v' \leq n$, the result follows. □

This allows us to formalise the procedure for computing the entries of Φ modulo p^N in Algorithm 3.1 below.

Theorem 3.3.4. The time complexity of Algorithm 3.1 is given by

$$p\tilde{N}^2 M\big(p\tilde{N}\log(p\tilde{N})\big) + d^n n\big(M(\log d) + (\tilde{N} + \log p)M(\tilde{N}\log p)\big)$$

where $M(-)$ denotes the complexity of integer multiplication and $\tilde{N}$ is $\mathcal{O}(N + n\log_p n)$.

Proof. We consider each of the steps in Algorithm 3.1. We can ignore the computational effort in Step (i), but observe that $\tilde{N} \in \mathcal{O}(N + n\log_p n)$, $M \in \mathcal{O}(p\tilde{N})$, and $R \in \mathcal{O}(\tilde{N})$.

Algorithm 3.1 Compute the matrix for $p^{-1}F_p$ on $H^n_{\mathrm{dR}}(\mathfrak{U})$

Input: Polynomial $a_0x_0^d + \cdots + a_nx_n^d$, prime p, precision $N \geq 0$.
Output: Matrix for $p^{-1}F_p$ on $H^n_{\mathrm{dR}}(\mathfrak{U})$ modulo p^N.
procedure DiagFrob$(n, d, p, N, a_0, \ldots, a_n)$

(i) Determine the numbers $C = C(n,p) = n+2\,\mathrm{ord}_p((n-1)!)+(n+1)\left\lfloor \log_p(n-1)\right\rfloor$, $\tilde{N} = N - n + 2C$, $M = M(\tilde{N}, p)$, and $R = R(\tilde{N}, p) = \lfloor M/p \rfloor$.

(ii) Precompute the Teichmüller lifts $\hat{a}_0, \ldots, \hat{a}_n$, and the sequences $(d^{-r})_{r=0}^{R}$, and $(\mu_m)_{m=0}^{M}$ to precision $\tilde{N}$.

(iii) For each monomial $u = (u_0, \ldots, u_n) \in B_1 \cup \cdots \cup B_n$, determine the unique monomial $v = (v_0, \ldots, v_n)$ such that $v_i = p(u_i+1) - 1 \bmod d$. Use the following steps to compute the entry at position (u, v) in the matrix.

(iv) Compute the expression $x_1 = (-1)^{u'+v'}(v'-1)!p^n$ as an exact integer and observe that $\mathrm{ord}_p(x_1) \geq n$.

(v) Compute the expression $x_2 = (u'-1)!\alpha_{u+1,v+1}$ to precision $\tilde{N}$ using equation (3.2) or (3.4) depending on whether p is even or odd.

(vi) Compute the inverse x_2^{-1} to precision $N - n$.

(vii) Finally, compute the product $x_1x_2^{-1}$ to precision N.

In Step (ii), we compute $n+1$ Teichmüller lifts with an overall complexity of $\mathcal{O}(n(\log p)M(\tilde{N}\log p))$. The computation of the sequence $(d^{-r})_{r=0}^{R}$ requires one reduction of d modulo $p^{\tilde{N}}$, one p-adic inversion and $R-1$ products to precision $\tilde{N}$, yielding $\mathcal{O}\big(M(\log d)+(\log\log p)M(\log p)+\tilde{N}M(\tilde{N}\log p)\big)$. Finally, we carry out the computation of $(\mu_m)_{m=0}^{M}$ following Remark 3.2.12, which requires time $\mathcal{O}\big(p\tilde{N}^2M(p\tilde{N}\log(p\tilde{N}))\big)$.

The following Steps (iii) through (vii) are executed $\dim H^n_{\mathrm{rig}}(U)$ times, where this dimension is $\mathcal{O}(d^n)$. The time complexity of Step (iii) is $\mathcal{O}\big(nM(\log\max\{p,d\})\big)$. We observe that we can ignore Step (iv). Step (v) involves an $(n+1)$-fold product of series with $\mathcal{O}(R)$ terms modulo $p^{\tilde{N}}$, where each summand requires an absolutely bounded number of products, as well as $n+1$ exponentiations of Teichmüller lifts with exponents given by $d^{-1}\big(p(u_i+1)-(v_i+1)\big) < p$. Computing the exponents has complexity $\mathcal{O}(M(\log\max\{p,d\}))$ and we note that we may ignore the update of the term $\big((u_i+1)/d\big)_r$ throughout the summation assuming we have computed the reduction of $d \bmod p^{\tilde{N}}$ once and for all earlier. Therefore, the complexity of each invocation of Step (v) is $\mathcal{O}\big(nM(\log d) + n(R+\log p)M(\tilde{N}\log p)\big)$. The p-adic inverse in Step (vi) requires time $\mathcal{O}\big((\log\log p)M(\log p) + M(\tilde{N}\log p)\big)$. Finally, we can ignore the product in Step (vii). Thus, the aggregate time complexity of Steps (iii) through (vii) is given

by $\mathcal{O}\big(d^n n\big(M(\log d) + (\tilde{N} + \log p)M(\tilde{N}\log p)\big)\big)$. □

Although the current implementation of Algorithm 3.1 performs very well in practical applications, its time complexity is quasi-cubic in the p-adic precision N, which is *not* optimal. Both of these aspects are due to the use of the sequence $(\mu_m)_{m=0}^M$ defined over $\mathbf{Z}_p$ instead of the coefficients $(\lambda_m)_{m=0}^M$ defined over $\mathbf{Q}_p(\pi)$. However, we can achieve a better time complexity by utilising fast exponentials of power series:

Theorem 3.3.5. There exists an algorithm for computing the matrix for the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U)$ in time complexity

$$(M\log M\log\log M)(p\log p)M(\tilde{N}\log p) + d^n n\big(M(\log d) + \tilde{N}(p\log p)M(\tilde{N}\log p)\big)$$

where $M(-)$ denotes the complexity of integer multiplication and $\tilde{N} \in \mathcal{O}(N + n\log_p n)$, $M \in \mathcal{O}(p\tilde{N})$.

Proof. The key idea is to slightly modify Algorithm 3.1 and use equation (3.1), working directly with the sequence $(\lambda_m)_{m=0}^M$. As we will be using operations in the totally ramified extension $\mathbf{Q}_p(\pi)$ to p-adic precision $\tilde{N}$, we remark that the cost of an arithmetic operation in this ring is $\mathcal{O}\big((p\log p)M(\tilde{N}\log p)\big)$, achieved by polynomial multiplication based on the fast Fourier transform.

We first observe that the valuation of the summands in equation (3.1) can be bounded by $\mathrm{ord}_p\big(\lambda_m(u_i/d)_r(-1)^r\pi^{-r}\hat{a}_i^{m-r}\big) \geq -1/2$. As the only term with negative valuation is π^{-r} and $R \in \mathcal{O}(\tilde{N})$, it suffices to precompute the sequences $(\lambda_m)_{m=0}^M$ and $(d^{-r})_{r=0}^R$ to p-adic precision $\tilde{N}$. While the computation of the latter remains unchanged, the computation of the former can be improved significantly using fast exponentials of power series in $\mathbf{Q}_p(\pi)[[z]]$ as described by Bernstein [6, §9.3]. This allows for computing the sequence $(\lambda_m)_{m=0}^M$ in $\mathcal{O}\big((M\log M\log\log M)\big)$ operations in $\mathbf{Q}_p(\pi)$ to precision $\tilde{N}$. The only remaining change to our analysis of Algorithm 3.1 occurs in Step (v), where for each matrix entry we have to consider $\mathcal{O}(nR)$ multiplications in $\mathbf{Q}_p(\pi)$ instead of $\mathbf{Z}_p$. □

Remark 3.3.6. The complexity estimate in Theorem 3.3.5 is an improvement over that presented by Lauder [47, §10].

3.4 Examples

We conclude this chapter by presenting three examples, demonstrating the relevance of the improvements in practical computations. The first two allow us to compare our implementations with timings reported by Gerkmann [30], and the third example features an increase in various parameters.

Firstly, let us consider the surface given by

$$x_0^4 + x_1^4 + x_2^4 + x_3^4 = 0$$

over $\mathbf{F}_3$ and aim to compute the matrix Φ to precision $N = 374$.

We find that we need to compute the expression $\alpha_{u+1,v+1}$ to precision $\tilde{N} = 377$ and are led to compute the terms μ_m for $m \leq 1768$.

In [30, §7.4], Gerkmann considers this example and reports a runtime of 45.26 minutes. In contrast, we are able to compute this matrix in 0.55 seconds, of which 0.39 seconds are spent in the precomputation of the sequence (μ_m), an improvement of a factor of nearly $5,000$.

Secondly, we consider a quintic surface over $\mathbf{F}_2$,

$$x_0^5 + x_1^5 + x_2^5 + x_3^5 = 0,$$

and we aim to compute the matrix Φ to precision $N = 355$. This leads us to compute $\alpha_{u+1,v+1}$ to precision $\tilde{N} = 370$, which requires us to compute μ_m up to $M = 1528$.

Gerkmann [30, §7.6] reports a runtime of 64.13 minutes. In contrast, we are able to compute this matrix in 0.32 seconds, of which 0.27 seconds are spent in the precomputation of the sequence (μ_m), an improvement by a factor of over $12,000$.

For the third and final example, we consider the sextic threefold over $\mathbf{F}_{97}$ given by

$$x_0^6 + 2x_1^6 + 3x_2^6 + 4x_3^6 + 5x_4^6$$

and compute the matrix Φ to precision $N = 500$. This leads to the other parameters $\tilde{N} = 504$ and $M = 87033$. We also observe that $\dim H^n_{\mathrm{dR}}(\mathfrak{U}) = 520$. The precomputation

requires 0.01 seconds for the sequence d^{-r} and 9.67 hours for the sequence μ_m. After this, we can determine the entries of the Frobenius matrix in another 30.86 seconds.

Chapter 4

Constructing the Gauss–Manin connection

In this chapter we return to the relative situation of a family of smooth projective hypersurfaces. Specifically, we recall that given a family X of smooth projective hypersurfaces over a finite field k we consider a lift $\mathfrak{X}$ in $\mathbf{P}^n(K) \times \mathbf{A}^1(K)$ defined by $P \in K[t][x_0, \ldots, x_n]$ where K is a field of characteristic zero. The goal of this chapter is to describe a practical routine for computing the action of the Gauss–Manin connection, viewed as a Leibniz-linear map on $H^n_{\mathrm{dR}}(\mathfrak{U})$ where $\mathfrak{U}$ is the complement of $\mathfrak{X}$.

The material in this chapter is arranged as follows. We begin by introducing the Gauss–Manin connection in a scheme-theoretic set-up following Katz and Oda [40]. Specialising to the case of hypersurfaces, we then follow Abbott, Kedlaya, and Roe [1] in the description of arithmetic in algebraic de Rham cohomology. This leads us to a complete description of the algorithm, including critical details on sparse matrix techniques used to accelerate the computations. Finally, we present computational examples, comparing the runtimes achieved by our approach to previous work where possible.

4.1 Scheme-theoretic introduction

In this section, we give a brief introduction to the Gauss–Manin connection in the algebraic sense, following Katz and Oda [40]. We consider two smooth schemes $\mathfrak{X}$ and

$\mathfrak{S}$ over a field K of characteristic zero together with a smooth K-morphism $\pi\colon \mathfrak{X} \to \mathfrak{S}$.

Definition 4.1.1. A *connection* on a quasi-coherent $\mathcal{O}_{\mathfrak{S}}$-module $\mathcal{E}$ is defined to be a homomorphism ρ of abelian sheaves

$$\rho\colon \mathcal{E} \to \Omega^1_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{E}$$

such that

$$\rho(se) = s\rho(e) + ds \otimes e$$

for sections s and e of $\mathcal{O}_{\mathfrak{S}}$ and $\mathcal{E}$, respectively, over an open subset of $\mathfrak{S}$.

Remark 4.1.2. We observe that this can be extended to the whole de Rham complex as follows. For every $i \in \mathbf{N}$, given $\omega \in \Omega^i_{\mathfrak{S}/K}$ and $e \in \mathcal{E}$ let $\omega \wedge \rho(e)$ denote the image of $\omega \otimes \rho(e)$ under the map

$$\Omega^i_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \left(\Omega^1_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{E}\right) \to \Omega^{i+1}_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{E}, \quad \omega \otimes (\tau \otimes e) \mapsto (\omega \wedge \tau) \otimes e.$$

We then obtain a homomorphism of abelian sheaves ρ_i given by

$$\rho_i\colon \Omega^i_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{E} \to \Omega^{i+1}_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{E}, \quad \omega \otimes e \mapsto d\omega \otimes e + (-1)^i \omega \wedge \rho(e).$$

Definition 4.1.3. The complex $\Omega^\bullet_{\mathfrak{X}/K}$ admits a decreasing filtration given by

$$F^j = \mathrm{Im}\Big(\Omega^{\bullet-j}_{\mathfrak{X}/K} \otimes_{\mathcal{O}_{\mathfrak{X}}} \pi^*\big(\Omega^j_{\mathfrak{S}/K}\big) \to \Omega^\bullet_{\mathfrak{X}/K}\Big).$$

Forming a spectral sequence $\{E_r, d_r\}_{r\geq 0}$ as in Griffiths and Harris [31, §3.5, p. 440], we find that

$$E_1^{i,j} = \Omega^i_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{H}^j_{dR}(\mathfrak{X}/\mathfrak{S}).$$

The *(algebraic) Gauss–Manin connection* ∇ is now defined as the differential $d_1^{0,j}$, i.e.,

$$\nabla = d_1^{0,j}\colon \mathcal{H}^j_{dR}(\mathfrak{X}/\mathfrak{S}) \to \Omega^1_{\mathfrak{S}/K} \otimes_{\mathcal{O}_{\mathfrak{S}}} \mathcal{H}^j_{dR}(\mathfrak{X}/\mathfrak{S}).$$

Remark 4.1.4. From the description by Katz and Oda [40] and following the practical

description by Kedlaya [43], we can describe the action of the Gauss–Manin connection further in the case when $\mathfrak{S}$ is affine, say $\mathfrak{S} = \operatorname{Spec}(A)$. In this case, we may apply the global section functor $\Gamma(\mathfrak{S}, -)$ throughout and consider

$$\nabla \colon H^j_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S}) \to \Omega^1_{A/K} \otimes_A H^j_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S}).$$

The action of ∇ can be computed as follows. Given $\omega \in H^j_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S})$, arbitrarily lift this to $\tilde{\omega} \in \Omega^j_{\mathfrak{X}/K}$. After computing the exterior derivative, which decomposes as $d_{\mathfrak{X}/K} = d_{\mathfrak{S}} + d_{\mathfrak{X}/\mathfrak{S}}$, the image of ω under the Gauss–Manin connection is given as the projection of $d_{\mathfrak{X}/K}(\tilde{\omega})$ onto $\Omega^1_{A/K} \otimes_A H^j_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S})$.

4.2 Hypersurfaces in projective space

In this section we make the results from the previous section concrete, specialising to the case of hypersurfaces in projective space. In the development of a computationally feasible description of the Gauss–Manin connection in this situation, we follow Abbott, Kedlaya and Roe [1, §3.2] when computing in de Rham cohomology and Kedlaya [43, §3.2] when expressing the action of the Gauss–Manin connection. We begin by recalling the notation relevant in this chapter.

Notation 4.2.1. We consider a non-singular hypersurface $\mathfrak{X}$ in $\mathbf{P}^n(K) \times \mathbf{A}^1(K)$, where K is a field of characteristic zero and $n \geq 2$, defined by a homogeneous polynomial $P \in K[t][x_0, \ldots, x_n]$ of degree d. We let $\mathfrak{U}$ denote the open complement $\mathbf{P}^n(K) \times \mathbf{A}^1(K) - \mathfrak{X}$.

Moreover, we may also view this as two smooth K-schemes $\mathfrak{X}$ and $\mathfrak{S}$ together with a smooth proper K-morphism $\mathfrak{X} \to \mathfrak{S}$, where we consider an affine subspace $\mathfrak{S} = \operatorname{Spec}(A)$ of the t-line[1]. We denote the fibre of $\mathfrak{X}$ above t in $\mathfrak{S}$ by $\mathfrak{X}_t$.

Let us consider a smooth fibre $\mathfrak{X}_t$. The embedding $\mathfrak{X}_t \hookrightarrow \mathbf{P}^n(K)$ induces maps $H^i_{\mathrm{dR}}(\mathbf{P}^n(K)/K) \to H^i_{\mathrm{dR}}(\mathfrak{X}_t/K)$, which by the Lefschetz hyperplane theorem [31, §1.2, p. 156] are bijective for $0 \leq i \leq n-2$ and injective for $i = n-1$. A direct computation [1, Corollary 3.1.4] shows that $H^i_{\mathrm{dR}}(\mathbf{P}^n(K))$ has, as a K-vector space, dimension 1 if $i =$

[1] We typically choose $A = K[t, r(t)^{-1}]$ where $r(t)$ is the Macaulay resultant of $\mathfrak{X}$ in order to exclude non-singular fibres. But it is harmless to exclude a finite number of additional fibres if this is convenient.

$0, 2, \ldots, 2n$ and 0 otherwise. Using Poincaré duality, it then follows that, for all $0 \leq i \leq 2n-2$ with $i \neq n-1$,

$$\dim_K H^i_{\mathrm{dR}}(\mathfrak{X}_t/K) = \begin{cases} 1 & \text{if } i \text{ is even,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

In particular, $H^{n-1}_{\mathrm{dR}}(\mathfrak{X}_t/K)$ is the only cohomology group that remains to be determined. Defining the complement $\mathfrak{U}_t = \mathbf{P}^n(K) - \mathfrak{X}_t$, from [32, (10.16)], we have one of the following two exact sequences

$$0 \to H^n_{\mathrm{dR}}(\mathfrak{U}_t/K) \to H^{n-1}_{\mathrm{dR}}(\mathfrak{X}_t/K) \to 0, \tag{4.2}$$

$$0 \to H^n_{\mathrm{dR}}(\mathfrak{U}_t/K) \to H^{n-1}_{\mathrm{dR}}(\mathfrak{X}_t/K) \to H^{n+1}_{\mathrm{dR}}(\mathbf{P}^n(K)/K) \to 0, \tag{4.3}$$

as n is even or odd, respectively.

Now define the n-form

$$\Omega = \sum_{i=0}^{n} (-1)^i x_i dx_0 \wedge \cdots \wedge \widehat{dx_i} \wedge \cdots \wedge dx_n. \tag{4.4}$$

A calculation as in [32, §4] then shows that the above cohomology group $H^n_{\mathrm{dR}}(\mathfrak{U}_t/K)$ is isomorphic as a K-vector space to the quotient of the group of n-forms $Q\Omega/P^k$ with $k \in \mathbf{N}$ and $Q \in K[x_0, x_1, \ldots, x_n]$ homogeneous of degree $kd - (n+1)$ by the subgroup generated by

$$\frac{(\partial_i Q)\Omega}{P^k} - k\frac{Q(\partial_i P)\Omega}{P^{k+1}} \tag{4.5}$$

for all $0 \leq i \leq n$, where here and in the following ∂_i denotes the partial derivative operator with respect to x_i.

Since the fibres of the relative de Rham cohomology $H^i_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S})$ can be identified with the de Rham cohomology $H^i_{\mathrm{dR}}(\mathfrak{X}_t/K)$ of the fibres, it follows that we can compute the action of the Gauss–Manin connection $\nabla\colon H^{n-1}_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S}) \to \Omega^1_{A/K} \otimes_A H^{n-1}_{\mathrm{dR}}(\mathfrak{X}/\mathfrak{S})$ via the induced map $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S}) \to \Omega^1_{A/K} \otimes_A H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$, which by abuse of notation we shall refer to as ∇, too. Let $\omega \in H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$, where we may assume it is of the form $Q\Omega/P^k$ as described above. Since $\mathfrak{S}$ is an affine curve, we have that $\Omega^1_{A/K}$ is free of

rank one, generated by the symbol dt. Then the action of ∇ is given by

$$\nabla\colon \omega \mapsto d_{\mathfrak{U}}(\omega) = d_{\mathfrak{S}}(\omega) + d_{\mathfrak{U}/\mathfrak{S}}(\omega) = \frac{\partial}{\partial t}\omega \wedge dt \tag{4.6}$$

where the term $d_{\mathfrak{U}/\mathfrak{S}}(\omega)$ vanishes by definition of Ω. We will describe how a unique representative can be obtained for the right-hand side in the following two sections.

4.3 Computing in de Rham cohomology

This section is devoted to an in-depth description of the computation in de Rham cohomology, exploiting the vector space isomorphism given in the previous section. We begin by setting up the notation and then describe the so-called *reduction of poles* procedure in some generality. A particular problem, that of finding the coordinates of an element of a multivariate polynomial ideal, is treated as a black box, but we turn to it in the second subsection where we specialise to the case when the family of projective hypersurfaces contains a diagonal fibre. Finally, we provide further details on the matrix computations involved in the third subsection.

4.3.1 Reduction of poles

This section is based on [1, Remark 3.2.5], describing a *reduction of poles* procedure also referred to as the Griffiths–Dwork method. We continue with the same Notation 4.2.1, but since for large parts of the discussion it will not matter that $K(t)$ is a function field, we also define $L = K(t)$.

From the description of $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ in the introduction, with its elements being represented by n-forms $Q\Omega/P^i$ for $i \in \mathbf{N}$, it is clear that it can be equipped with a filtration whose ith part consists of all elements which can be represented by n-forms as above with $\deg Q = kd - (n+1)$ for $1 \leq k \leq i+1$.

We can obtain a basis for $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ respecting this filtration as follows. For every $k \in \mathbf{N}$, we find an independent set B_k of polynomials of degree $kd-(n+1)$ generating the quotient of the space of all such polynomials by the Jacobian ideal $(\partial_0 P, \ldots, \partial_n P)$. This yields a generating set $\bigcup_{k\in\mathbf{N}} \mathcal{B}_k$ for $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ where $\mathcal{B}_k = \{Q\Omega/P^k : Q \in B_k\}$.

However, it follows from a theorem of Macaulay [32, §4, (4.11)] that in fact the set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ already forms a generating set. In the next subsection, we shall exhibit an explicit basis of monomials in the case where the family of projective hypersurfaces contains a diagonal fibre.

Now, to obtain a unique representative for the class of $Q\Omega/P^k$ in terms of the basis elements in $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$, we express Q in terms of $\partial_0 P, \ldots, \partial_n P$ as well as elements of B_k, and then iteratively reduce the pole order of the first part according to the relations given by the expressions from equation (4.5). Assume that we have at our disposal a routine DECOMPOSE which, given a polynomial Q of degree $kd - (n+1)$ in the ideal generated by the Jacobian ideal $(\partial_0 P, \ldots, \partial_n P)$ together with the set B_k, returns an expression of the form $Q = Q_0 \partial_0 P + \cdots + Q_n \partial_n P + \gamma_k$ with $Q_0, \ldots, Q_n$ homogeneous polynomials in $L[x_0, \ldots, x_n]$ and γ_k in the L-span of B_k. The reduction of poles procedure can then be formalised as in Algorithm 4.1 below, which we will refer to as REDUCE. In this generality, the correctness of the algorithm depends on a theorem of Macaulay [32, §4, (4.11)].

4.3.2 Reduction of poles using linear algebra

In this subsection, we specialise to the case of a smooth family of projective hypersurfaces containing a diagonal fibre. We exhibit a basis of monomials $B_1 \cup \cdots \cup B_n$ such that the corresponding set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ forms a basis for $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ and re-express the problem of decomposing a homogeneous polynomial from the previous section in the language of linear algebra. From this description, we furnish an explicit reduction procedure in terms of matrices. The approach is based on a generalisation of Sylvester matrices from two polynomials to $n+1$ polynomials, following Macaulay [51].

The decomposition problem from the previous section can be formulated as follows:

Problem 4.3.1. Given a homogeneous multivariate polynomial $Q \in L[x_0, \ldots, x_n]$ of degree $kd - (n+1)$, for some $k \in \mathbf{N}$, we try to find homogeneous polynomials $Q_0, \ldots, Q_n$ in $L[x_0, \ldots, x_n]$ such that

$$Q \equiv Q_0 \partial_0 P + \cdots + Q_n \partial_n P \tag{4.7}$$

Algorithm 4.1 Reduce $Q\Omega/P^k$ in $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$

Input: P is a homogeneous polynomial in $L[x_0,\dots,x_n]$, $\mathrm{char}(L)=0$, of degree d defining a smooth hypersurface; B_k, for $1\le k\le n$, is a basis for all homogeneous polynomials of degree $kd-(n+1)$ modulo $(\partial_0 P,\dots,\partial_n P)$; Q is homogeneous of degree $kd-(n+1)$.

Output: Polynomials γ_i in the L-span of B_i, for $1\le i\le n$, with $Q\Omega/P^k \equiv \gamma_1\Omega/P^1 + \cdots + \gamma_n\Omega/P^n$.

procedure REDUCE($P, B_1,\dots,B_n, Q$)
  while $k \ge n+1$ **do**
    $Q_0,\dots,Q_n \leftarrow$ DECOMPOSE($Q, \partial_0 P,\dots,\partial_n P, B_k$)
    $k \leftarrow k-1$
    $Q \leftarrow k^{-1}\sum_{i=0}^n \partial_i Q_i$
  $\gamma_{k+1},\dots,\gamma_n \leftarrow 0$
  while $Q \notin B_k$ **do**
    $Q_0,\dots,Q_n \leftarrow$ DECOMPOSE($Q, \partial_0 P,\dots,\partial_n P, B_k$)
    $\gamma_k \leftarrow Q - \sum_{i=0}^n Q_i\partial_i P$
    $k \leftarrow k-1$
    $Q \leftarrow k^{-1}\sum_{i=0}^n \partial_i Q_i$
  if $Q \neq 0$ **then**
    $\gamma_k \leftarrow Q$
    $k \leftarrow k-1$
  $\gamma_1,\dots,\gamma_k \leftarrow 0$
  return $\gamma_1,\dots,\gamma_n$

modulo the L-span of B_k.

Remark 4.3.2. Immediately, we see that, for each $0\le i\le n$, either Q_i is identically zero or has degree $(k-1)d-n$ since $\partial_i P$ is homogeneous of degree $d-1$ and also the elements of B_k have degree $kd-(n+1)$.

For the remaining part of this section we consider the following basis sets B_k, for $k\in\mathbf{N}$, which as before induce a generating set $\mathcal{B}_1\cup\cdots\cup\mathcal{B}_n$ of $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ as an L-vector space respecting the aforementioned filtration:

Definition 4.3.3. For $k\in\mathbf{N}$, we define the following sets of monomials,

$$F_k = \{x^i : i\in\mathbf{N}_0^{n+1}, |i| = kd-(n+1)\},$$
$$B_k = \{x^i : i\in\mathbf{N}_0^{n+1}, |i| = kd-(n+1) \text{ and } i_j < d-1 \text{ for } 0\le j\le n\},$$

where $x^i = x_0^{i_0}\cdots x_n^{i_n}$ and $|i| = i_0+\cdots+i_n$. Moreover, we let $\mathcal{F}_k$ denote the L-vector space spanned by the n-forms $x^i\Omega/P^k$ for $x^i\in F_k$ and $\mathcal{B}_k$ denote the set of n-forms

$x^i\Omega/P^k$ with $x^i \in B_k$.

We shall defer the proof of the statement that the corresponding set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ indeed forms a basis of $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$, at least in the case when the family of hypersurfaces given by P contains a diagonal fibre, until the end of this section.

Remark 4.3.4. In order to ensure that our computations are not vacuous, we assume that $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ is non-zero. After setting

$$\ell = \left\lceil \frac{n+1}{d} \right\rceil, \quad u = \left\lfloor \frac{(n+1)(d-1)}{d} \right\rfloor = n+1-\ell,$$

we note that B_k is non-empty if and only if $\ell \leq k \leq u$. The assumption is thus that $\ell \leq u$, which is equivalent to the statement that $d \geq 2$ whenever n is odd and $d \geq 3$ whenever n is even.

When considering Problem 4.3.1, it turns out that, when the family of hypersurfaces given by P contains a diagonal fibre, we can place further restrictions on the polynomials $Q_0, \ldots, Q_n$. We thus consider the following problem instead:

Problem 4.3.5. Given a homogeneous multivariate polynomial $Q \in L[x_0, \ldots, x_n]$ of degree $kd-(n+1)$, for some $k \in \mathbf{N}$, we try to find homogeneous polynomials $Q_0, \ldots, Q_n$ in $L[x_0, \ldots, x_n]$ such that

$$Q \equiv Q_0\partial_0 P + \cdots + Q_n\partial_n P \tag{4.8}$$

modulo the L-span of B_k. Moreover, for each $1 \leq j \leq n$, the polynomial Q_j may only contain non-zero coefficients for monomials of degree $(k-1)d-n$ that are not divisible by any of the monomials $x_0^{d-1}, \ldots, x_{j-1}^{d-1}$.

Definition 4.3.6. For $k \in \mathbf{N}$, we define the following sets of monomials in $L[x_0, \ldots, x_n]$. Let $R_k = F_k - B_k$ be the set of monomials of total degree $kd-(n+1)$ and partial degree at least $d-1$ with respect to some of the $n+1$ variables. Let $C_k^{(0)}$ be the set of monomials of total degree $(k-1)d-n$, and then inductively, for $j = 1, \ldots, n$, define $C_k^{(j)}$ to be the set of monomials in $C_k^{(j-1)}$ except for those divisible by x_{j-1}^{d-1}. Moreover, we define the

multi-set C_k as the disjoint union of $C_k^{(0)}, \ldots, C_k^{(n)}$. We shall write an element of this multi-set as (j, g), referring to a monomial g in $C_k^{(j)}$.

With this set-up, the following theorem provides a solution to Problem 4.3.5 in the cases that we are interested in:

Theorem 4.3.7. Suppose that the family of smooth projective hypersurfaces given by the polynomial P in $L[x_0, \ldots, x_n]$ contains a diagonal fibre. For $k \in \mathbf{N}$ and $0 \leq j \leq n$, let $U_k^{(j)}$ be the L-vector space of polynomials with basis $C_k^{(j)}$, and let U_k denote the cartesian product $U_k = U_k^{(0)} \times \cdots \times U_k^{(n)}$. Moreover, let V_k and W_k be the L-vector spaces of polynomials with bases F_k and R_k, respectively, and let $\pi \colon V_k \to W_k$ denote the linear map that sends the elements of B_k to zero and the elements of R_k to themselves. Then the L-linear map

$$\phi_k \colon U_k \to W_k, (Q_0, \ldots, Q_n) \mapsto \pi\big(Q_0 \partial_0 P + \cdots + Q_n \partial_n P\big) \tag{4.9}$$

is an isomorphism of L-vector spaces.

Proof. We first show that, for all $k \in \mathbf{N}$, the multi-sets R_k and C_k have the same cardinality:

We construct the following bijection $R_k \to C_k$, representing the monomials by their exponent tuple. Let $i = (i_0, \ldots, i_n)$ be in R_k. If $i_0 \geq d - 1$, we define the image as $(i_0 - d - 1, i_1, \ldots, i_n) \in C_k^{(0)}$. More generally, if $i_0 < d-1, \ldots, i_{j-1} < d-1$ and $i_j \geq d-1$, the image is $(i_0, \ldots, i_{j-1}, i_j - d - 1, i_{j+1}, \ldots, i_n) \in C_k^{(j)}$. It is easy to verify that this map is indeed a bijection.

In order to establish that the map $\phi_k \colon U_k \to W_k$ is an isomorphism of L-vector spaces, we now exhibit its matrix with respect to the given basis:

Let $k \in \mathbf{N}$ and suppose that R_k and C_k are non-empty, that is to say, $k \geq n/d+1$. We define the auxiliary matrix Δ_k with row and column index sets R_k and C_k, respectively, as follows. Given $f \in R_k$ and $(j, g) \in C_k$, we set the corresponding entry in Δ_k to be the monomial coefficient of f/g in $\partial_j P$ if g divides f and 0 otherwise. It is immediate that Δ_k is the matrix representing ϕ_k with respect to the bases C_k and R_k of U_k and W_k, respectively.

The assumption that the family $\mathfrak{X}$ of projective hypersurfaces given by P contains a diagonal hypersurface means that for some t_0 the fibre $\mathfrak{X}_{t_0}$ is given by an equation of the form

$$P_{t_0}(x_0, \ldots, x_n) = a_0 x_0^d + \cdots a_n x_n^d = 0$$

with $a_0, \ldots, a_n \in K^\times$.

We aim to show that the determinant of Δ_k is non-zero in L. Since the specialisation to the diagonal fibre viz. evaluation of the matrix at $t = t_0$ commutes with computing the determinant, it suffices to show that the determinant of $(\Delta_k)\big|_{t=t_0}$ is non-zero in K.

Since, for $0 \leq j \leq n$, $\partial_j P_{t_0}(x_0, \ldots, x_n) = d a_j x_j^{d-1}$, there is precisely one non-zero entry in each column of Δ_k. Namely, in column $(j, g) \in C_k$ and row $g x_j^{d-1} \in R_k$ there is the non-zero entry $d\alpha_j$. Thus, the determinant of $(\Delta_k)\big|_{t=t_0}$ is given by the product of all its non-zero entries, which implies that it is non-zero, concluding the proof. □

In principle, by including any of the numerous methods available for solving linear equations, we are in a position to furnish a routine DECOMPOSE, which we formalise in Algorithm 4.2.

Algorithm 4.2 Obtain co-ordinates for Q in the Jacobian ideal modulo basis elements

---

Input:
- Q is a homogeneous polynomial of degree $kd - (n+1)$.
- $\partial_0 P, \ldots, \partial_n P$ are the partial derivatives of a homogeneous polynomial P in $K[t][x_0, \ldots, x_n]$ of degree d, which defines a smooth family of projective hypersurfaces containing a diagonal fibre, where K is a field of characteristic zero.
- B_k is the set of all monomials of total degree $kd-(n+1)$ and partial degree less than $d-1$.

Output: Homogeneous polynomials $Q_0, \ldots, Q_n$ such that $Q \equiv Q_0 \partial P_0 + \cdots + Q_n \partial_n P$ modulo the $K(t)$-span of B_k.

The multi-sets R_k and C_k are as in Definition 4.3.6, the matrix Δ_k is as in the proof of Theorem 4.3.7.

procedure DECOMPOSE$(Q, \partial_0 P, \ldots, \partial_n P, B_k)$

Step I. Let b be the vector of length $|R_k|$ such that the entry corresponding to the monomial $x^i \in R_k$ is the coefficient of x^i in Q.

Step II. Let v be the unique vector of length $|C_k|$ satisfying $\Delta_k v = b$. From the description of C_k as a disjoint union, we can write v accordingly as $\left(v^{(0)}, \ldots, v^{(n)}\right)$ where, for $0 \leq j \leq n$, $v^{(j)}$ is a vector of length $|C_k^{(j)}|$.

Step III. For $j = 0, \ldots, n$, set $Q_j = \sum_{g \in C_k^{(j)}} v_g^{(j)} g$, where $v_g^{(j)}$ is the entry in $v^{(j)}$ corresponding to the monomial $g \in C_k^{(j)}$.

Step IV. **return** $Q_0, \ldots, Q_n$

We conclude this section by establishing that the set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ is indeed a basis for $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$, as claimed earlier, using the reduction of poles procedure.

Theorem 4.3.8. Suppose that the family of projective hypersurfaces given by P contains a diagonal fibre. Then the set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ as in Definition 4.3.3 is a basis for the L-vector space $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$.

Proof. We know that $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ is spanned by the classes of the n-forms $Q\Omega/P^k$ for all homogeneous polynomials Q of degree $kd-(n+1)$ and $k \in \mathbf{N}$. By a theorem of Macaulay [32, §4, (4.11)], we may assume that $1 \leq k \leq n$, that is to say, any class in $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$ can be represented by an n-form with a pole of order at most n.

Without loss of generality, we may thus start the reduction of poles procedure with a homogeneous polynomial Q of degree $(n+1)d-(n+1)$. Then, since $B_{n+1} = \emptyset$ and $R_{n+1} = F_{n+1}$, Theorem 4.3.7 shows that there exist homogeneous polynomials $Q_0, \ldots, Q_n$ either zero or homogeneous of degree $nd-(n+1)$ such that $Q = Q_0\partial_0 P + \cdots + Q_n\partial_n P$. Continuing with the reduction of poles procedure as described in Algorithm 4.1, we obtain an expression for $Q\Omega/P^{n+1}$ as an L-linear combination of elements in $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$. This shows that this set spans the vector space $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$.

To see that this set is linearly independent, note that it contains only monomials whose partial degrees are strictly less than $d-1$. However, since P is a homogeneous polynomial of degree d and the family of hypersurfaces contains a diagonal fibre, it follows that, for each $0 \leq i \leq n$, the partial derivative $\partial_i P$ is a homogeneous polynomial of degree $d-1$ and contains precisely one monomial term with partial degree equal to $d-1$, namely that of the monomial x_i^{d-1}. It follows that the elements of $B_1 \cup \cdots \cup B_n$ cannot be reduced further modulo the Jacobian ideal and hence that the set $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_n$ is linearly independent. □

4.3.3 Sparse matrix techniques

In this subsection we present a technique for repeatedly solving a sparse system of linear equations as in *Step II.* of Algorithm 4.2. The methods we present in this section are discussed in great detail in a survey by Reid [55].

Before continuing, let us briefly justify why spare matrix methods are appropriate in this case:

Remark 4.3.9. Let τ and $\tau_0, \ldots, \tau_n$ denote the number of non-zero terms of the polynomials P and $\partial_0 P, \ldots, \partial_n P$, respectively. We observe that the auxiliary matrix Δ_k described in the proof of Theorem 4.3.7 contains at most τ_j non-zero entries in column (j, g), since the non-zero entries in this column are coefficients of the polynomial $\partial_j P$ corresponding to monomials in R_k via multiplication by g. In particular, the number of non-zero entries in the matrix is bounded above by $\tau\,|R_k|$.

Notation 4.3.10. In order to simplify the notation to what is relevant in this section, we shall consider the system of linear equations given by

$$Ax = b \tag{4.10}$$

where A is a non-singular $n \times n$ matrix over a field of characteristic zero. Moreover, for later reference we let τ denote the number of non-zero entries in A.

Our first observation is that the reduction of poles procedure from the previous section requires a linear system as in equation (4.10) to be solved multiple times for the same matrix A but with different column vectors b, which suggests using a pre-processing approach such as LUP decomposition.

The LUP decomposition of a matrix A is a way of performing the classical Gaussian elimination with bookkeeping. It consists of a permutation matrix P, a unit lower-triangular matrix L and an upper triangular matrix U such that $PA = LU$. Once these are known, the original system can be solved in a two step process as

$$Ly = Pb, \quad Ux = y \tag{4.11}$$

which consists of two triangular systems. We briefly note that, using a classical dense approach [19], the LUP decomposition can be computed using a number cubic in n of operations in the base field, and that solving the two triangular systems only requires a number quadratic in n of operations in the base field.

However, assuming that the matrix A is sparse, with further pre-processing a much better performance can be realised. As a first step, we permute the rows of the matrix to ensure that the diagonal contains only non-zero entries. We can achieve this here because we assume that the matrix is non-singular. Effectively, this problem is the same as that of computing a perfect matching in the $n \times n$ bipartite graph with vertex sets $\{v_i\}_{i=1}^n$ and $\{w_i\}_{i=1}^n$, where an edge $v_i w_j$ is present if and only the element A_{ij} is non-zero. If we find a perfect matching $\{v_{\pi i} w_i\}_{i=1}^n$ in the form of a permutation $\pi \in S_n$, then, after defining the permutation matrix P_0 via

$$(P_0)_{ij} = \begin{cases} 1 & \text{if } j = \pi i, \\ 0 & \text{otherwise,} \end{cases} \tag{4.12}$$

we conclude that $P_0 A$ has a zero-free diagonal. A detailed discussion of this problem together with an implementation can be found in the work of Duff [22, 21]. We end our discussion here by quoting the worst-case and experimental average-case complexity results as $\mathcal{O}(\tau n)$ and $\mathcal{O}(\tau + n)$, respectively.

The second step is to compute the block-triangularization of the matrix $P_0 A$. We compute a permutation matrix Q_0 such that the matrix $Q_0 P_0 A Q_0^t$ is block-triangular, that is, is of the form

$$Q_0 P_0 A Q_0^t = \begin{pmatrix} A^{(11)} & & & \\ A^{(21)} & A^{(22)} & & \\ \vdots & & \ddots & \\ A^{(N1)} & A^{(N2)} & \dots & A^{(NN)} \end{pmatrix} \tag{4.13}$$

where each diagonal block $A^{(kk)}$ is square and can itself not be symmetrically permuted to block-triangular form. Again an asymptotically optimal algorithm for this problem stems from the realm of graphs: it is Tarjan's linear time algorithm for computing the strongly connected components in a directed graph. In this setting, its asymptotic complexity is given by $\mathcal{O}(n + \tau)$. For further details, we refer to the joint work of Duff and Reid [24, 23].

Finally, this enables us to solve N potentially much smaller systems of linear equations instead of the one we started with. We can rewrite our original system of equations $Ax = b$ as $A'y = c$ where $A' = Q_0 P_0 A Q_0^t$, $y = (Q_0^t)^{-1}x$ and $c = Q_0 P_0 b$ and now use the fact that this system is block-triangular. This allows us to instead solve the sequence of systems of linear equations

$$A^{(kk)} y_k = c_k - \sum_{j=1}^{k-1} A^{(kj)} y_j \tag{4.14}$$

for $k = 1, \ldots, N$, where we implicitly think of the column vectors y and c to be divided into N corresponding blocks $y_1, \ldots, y_N$ and $c_1, \ldots, c_N$. An important consequence of this approach is that after the above two steps of pre-processing, the off-diagonal blocks are not changed further; in particular, the phenomenon of fill-in during Gaussian elimination, which refers to the introduction of new non-zero entries, is limited to the diagonal blocks.

4.4 Description of the algorithm

We now describe the action of the Gauss–Manin connection ∇ on basis elements of $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$. Suppose that we are given a basis element $x^i\Omega/P^k \in \mathcal{B}_k$, where for $1 \leq k \leq n$ the set $\mathcal{B}_k$ is defined as in Definition 4.3.3. Following the description in Section 4.2, the action of the connection is given by exterior differentiation, that is, differentiation with respect to t. We first compute

$$\frac{d}{dt}\left(\frac{x^i\Omega}{P^k}\right) = \frac{-kx^i P_t \Omega}{P^{k+1}},$$

where $P_t = dP/dt$. The second step is to repeatedly apply the reduction of poles procedure in de Rham cohomology in order to express the n-form above as

$$\frac{d}{dt}\left(\frac{X^i\Omega}{P^k}\right) \equiv \frac{\gamma_{k+1}\Omega}{P^{k+1}} + \cdots + \frac{\gamma_1\Omega}{P}$$

where, for $1 \leq i \leq k+1$, γ_i is an element in the $K(t)$-span of B_i.

We formalise this for later reference in Algorithm 4.3.

Algorithm 4.3 Computing the Gauss–Manin connection matrix

Input: Homogeneous polynomial P in $K[t][x_0, \ldots, x_n]$ of degree d, defining a family of smooth projective hypersurfaces containing a diagonal fibre, where K is a field of characteristic zero.

Output: Basis $B_1 \cup \cdots \cup B_n$ for $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$; connection matrix M with respect to this basis.

procedure GMConnection(P)

Step I. Compute the partial derivatives $\partial_0 P, \ldots, \partial_n P$ and the exterior derivative dP/dt.

Step II. Compute the basis sets $B_1, \ldots, B_n$. In the following, we use the convenient way to index their union by (i, f), referring to a polynomial $f \in B_i$.

Step III. Compute the auxiliary matrices Δ_k, for $k = \lfloor n/d \rfloor + 1, \ldots, n+1$, and perform pre-processing on each auxiliary matrix as described in Sections 4.3.2 and 4.3.3.

Step IV. For all $(j, g) \in \bigcup_{k=1}^{n} B_k$, let $Q = -jgP_t$ and set $\gamma_1, \ldots, \gamma_n$ to the output of Reduce($P, B_1, \ldots, B_n, Q$). Then, for each $(i, f) \in \bigcup_{k=1}^{n} B_k$, let $M_{(i,f),(j,g)}$ be the coefficient of f in γ_i.

Step V. **return** $B_1 \cup \cdots \cup B_n$, M

4.5 Examples

In this section, we present a few numerical examples. In each case, we include a comparison of the runtime and memory usage of previously existing Magma routines[2] by Lauder and our new implementation in C. Lauder's routines were executed using Magma version 2.15-12 on a departmental machine comprising eight Dual Core AMD Opteron processors running at 2.2GHz as well as 32GB of memory. The author's new implementation was run on the machine `wolverine.maths.ox.ac.uk` featuring twenty-four Intel Xeon processors running at 2.93GHz. However, all routines involved limit their use to a single processor. We also note that, for the smallest examples, the comparison of the memory requirements is not fair in the sense that, even without executing user specific code, the Magma set-up required about 9.53MB of memory and therefore only the excess of this should be attributed to Lauder's routines.

To begin with, the following Table 4.1 illustrates the numerical size of the problems we need to consider even for relatively small dimensions and degrees.

[2]Specifically, we used the routines `GriffithsRed-v1.4.m` and `ConnMatrix-v1.4.m`.

Table 4.1: Dimensions of the graded parts of $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$

d		n											
		2			3				4				
	k	1	2	3	1	2	3	4	1	2	3	4	5
3	$\lvert\mathcal{B}_k\rvert$	1	1	0	0	6	0	0	0	5	5	0	0
	$\operatorname{rank}\mathcal{F}_k$	1	10	28	0	10	56	165	0	5	70	330	1001
	k	1	2	3	1	2	3	4	1	2	3	4	5
4	$\lvert\mathcal{B}_k\rvert$	3	3	0	1	19	1	0	0	30	30	0	0
	$\operatorname{rank}\mathcal{F}_k$	3	21	55	1	35	165	455	0	35	330	1365	3876
	k	1	2	3	1	2	3	4	1	2	3	4	5
5	$\lvert\mathcal{B}_k\rvert$	6	6	0	4	44	4	0	1	101	101	1	0
	$\operatorname{rank}\mathcal{F}_k$	6	36	91	4	84	364	969	1	126	1001	3876	10626

4.5.1 A toy example

We begin by giving many details of the computation in the case of the toy example given by $P(X,Y,Z) = X^3 + Y^3 + Z^3 + tXYZ$, containing the diagonal and only one varying cross-term, which is symmetric in all variables.

After computing the derivatives, we find that $\ell = 1$ and $u = 2$, and hence compute the basis sets $B_1 = \{1\}$ and $B_2 = \{XYZ\}$.

The auxiliary matrix Δ_2 is computed as

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & t & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & t & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & t & 0 & 0 & 0 & 0 & 0 \\ 0 & t & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t & 3 & 0 & 0 \\ 0 & 0 & t & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

but in this initial form it still has three zero entries on the diagonal. The permutation P_0 such that $P_0 A$ has a zero-free diagonal is given by the cycle (3 5 4). The subsequent permutation to block-triangular form is given by (2 6 4)(3 7)(5 8). Together they

transform Δ_2 to $Q_0 P_0 \Delta_2 Q_0^t$ which is

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & t & 0 & 0 & 0 & 0 & 0 \\ 0 & t & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & t & 0 & 0 \\ 0 & 0 & 0 & 0 & t & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}.$$

The above matrix is block-triangular with block sizes $1, 3, 3, 1, 1$. The following LUP decomposition performed on the two 3×3 submatrices yields the matrix

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & t & 0 & 0 & 0 & 0 & 0 \\ 0 & t/3 & 3 & -t^2/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t/3 & (t^3+27)/9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & t & 0 & 0 \\ 0 & 0 & 0 & 0 & t/3 & 3 & -t^2/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t/3 & (t^3+27)/9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}.$$

We do not include full details for the computation of Δ_3, which is a 28×28 matrix. However, in order to illustrate the usefulness of applying sparse rather then dense matrix techniques, we point out that the corresponding block sizes partition 28 as $19 \times 1 + 3 \times 3$.

In order to compute the Gauss–Manin connection matrix with respect to our choice of basis, we need to reduce the two elements $-XYZ\Omega/P^2$ and $-2X^2Y^2Z^2\Omega/P^3$ corresponding to the basis elements Ω/P and $XYZ\Omega/P^2$, respectively. This gives

$$\begin{aligned} \nabla\Big(\frac{\Omega}{P}\Big) &= \frac{-XYZ\Omega}{P}, \\ \nabla\Big(\frac{XYZ\Omega}{P^2}\Big) &= \frac{-2X^2Y^2Z^2\Omega}{P^3} \equiv \frac{t/(t^3+27)\Omega}{P} + \frac{-3t^2/(t^3+27)\Omega}{P^2}. \end{aligned}$$

Finally, the runtime and memory requirements of the Magma routines and the new C implementation are given by 0.0035s and 0.00017s as well as 7.32MB and 136KB, respectively.

4.5.2 Quartic surfaces

We now consider a sequence of quartic surfaces with an increasing number of cross-terms. This highlights the practical limitations of the previous MAGMA code and demonstrates the usefulness of the approach investigated in this thesis.

From Table 4.1 we see that $|B_1| = 1$, $|B_2| = 19$, $|B_3| = 1$ and $|\mathcal{R}_2| = 16$, $|\mathcal{R}_3| = 164$, $|\mathcal{R}_4| = 455$. We are particularly interested in how these last three cardinalities are partitioned by the block-triangularisation since this provides a measure of the usefulness of the sparse matrix techniques employed here. For a partition of the matrix Δ_k into N_k diagonal blocks of size $n_1, \ldots, n_{N_k}$, we include the two quantities $\alpha_k = |\mathcal{R}_k|^2 / (n_1^2 + \cdots + n_{N_k}^2)$ and $\beta_k = |\mathcal{R}_k|^3 / (n_1^3 + \cdots + n_{N_k}^3)$, rounded to the nearest integer. These two quantities give an indication of the improvement in runtime gained by employing sparse matrix techniques instead of standard dense matrix techniques. The former value pertains to the quadratic time routines, which in particular includes the solving of linear systems as part of the reduction process, whereas the latter is relevant for the LUP decomposition, which in the current implementation requires cubic time.

A quartic surface with one cross-term

We consider the family of surfaces given by

$$W^4 + X^4 + Y^4 + Z^4 + tWXYZ.$$

The partitions we obtain are $16 = 16\times1$, $164 = 104\times1+15\times4$ and $455 = 391\times1+16\times4$, which yield values $\alpha = (\alpha_k)_{k=1}^3 = (16, 78, 319)$ and $\beta = (\beta_k)_{k=1}^3 = (256, 4145, 66570)$.

As expected from the small number of terms and high level of symmetry, both Lauder's MAGMA routine and the new implementation compute this quickly in 0.045s and 0.008s, requiring 7.71MB and 1.35MB of memory, respectively.

We also note that Gerkmann [30, §7.5] mentions a runtime of about 7s in this case.

A quartic surface with three cross-terms

We consider the family of surfaces given by

$$W^4 + X^4 + Y^4 + Z^4 + t(W^3X + WYZ^2 + XZ^3).$$

The block sizes give rise to partitions $16 = 16 \times 1$, $164 = 70 \times 1 + 8 \times 4 + 1 \times 62$ and $455 = 253 \times 1 + 33 \times 4 + 1 \times 70$, from which we find $\alpha = (16, 6, 36)$ and $\beta = (256, 18, 273)$.

The time requirements for the two routines are 27.88s and 0.82s, and the space requirements are 18.84MB and 5.29MB, respectively. In particular, the new routine is about 34 times faster.

A quartic surface with four cross-terms

We consider the family of surfaces given by

$$W^4 + X^4 + Y^4 + Z^4 + t(W^3X + WYZ^2 + XZ^3 + WXYZ).$$

The partitions we obtain are $16 = 16 \times 1$, $164 = 53 \times 1 + 4 \times 4 + 1 \times 95$ and $455 = 231 \times 1 + 29 \times 4 + 1 \times 108$. This gives $\alpha = (16, 3, 17)$ and $\beta = (256, 5, 75)$.

The time and space requirements for the two routines are 4190.610s, 8.01s and 238.38MB, 16.62MB. At this point, the new implementation is already more than 532 times faster.

A quartic surface with six cross-terms

We consider the family of surfaces given by

$$W^4 + X^4 + Y^4 + Z^4 + t(2W^3X + 7W^2XY - 11WX^2Y + 13X^2YZ + 17X^2Z^2 - WXYZ).$$

The block-triangularisations result in the following partitions of the ranks of the auxiliary matrices: $16 = 14 \times 1 + 1 \times 2$, $164 = 38 \times 1 + 4 \times 2 + 4 \times 4 + 1 \times 102$ and $455 = 181 \times 1 + 18 \times 2 + 22 \times 4 + 7 \times 5 + 1 \times 115$. We hence find that $\alpha = (14, 3, 15)$ and $\beta = (186, 4, 62)$.

For this example, Lauder's Magma routine takes 4.52 days and requires 3.57GB of memory. The new implementation completes the computation in 49.47s and uses 66.7MB. That is, the new implementation is more than 7894 times faster.

A quartic surface with seven cross-terms

We consider the family of surfaces given by

$$W^4 + X^4 + Y^4 + Z^4 + t(-3W^3X + 5W^3Y + 7W^2XY \\ - 23WX^2Y - 29X^2YZ + 31Y^2Z^2 - 37WXYZ)$$

In this case, we obtain the partitions $16 = 14 \times 1 + 1 \times 2$, $164 = 32 \times 1 + 1 \times 4 + 1 \times 128$ and $455 = 152 \times 1 + 15 \times 4 + 3 \times 28 + 1 \times 159$ and values $\alpha = (14, 2, 7)$ and $\beta = (186, 2, 23)$.

The previous Magma routine solved this example in 34.04 days using 12.5GB of memory. In contrast to this, the runtime of the new C implementation has only doubled when compared to the previous example and is 117.32s, with a memory requirement of 126.5MB.

4.5.3 A quintic surface

We finally give an example which begins to show limitations of the approach discussed in this work:

$$(1-t)(W^5 + X^5 + Y^5 + Z^5) + t\big((WXZ + Y^3)(W^2 + XY + Z^2)\big).$$

For this example, the new implementation requires about 31.27 minutes and 979.5MB. The author did not obtain the corresponding data from Lauder's Magma routine since it was terminated after running for over 34 days and using nearly 5GB of memory.

4.6 Further remarks

We remark that we have restricted our computations to families X defined over the prime field $\mathbf{F}_p$, with lifts $\mathcal{X}$ defined by a polynomial $P \in \mathbf{Z}[t][x_0, \ldots, x_n]$ over the in-

tegers. Thus, the computation of the connection takes place over the field of rational functions $\mathbf{Q}(t)$. We note that our implementation using exact arithmetic demonstrates a significant improvement over previous work already. However, as part of the deformation method, we only require a p-adic approximation of the connection matrix. Further practical improvements should be possible by developing an implementation that precomputes the Macaulay resultant $r(t) \in \mathbf{Z}[t]$ and then performs arithmetic operations in $\mathbf{Q}_p[t, r(t)^{-1}]$ to finite p-adic precision.

Chapter 5

From differential systems to zeta functions

In this chapter we describe the remaining steps of the deformation method: how to use the differential system governed by the Gauss–Manin connection matrix and the action of Frobenius on the initial fibre $H^n_{\mathrm{rig}}(U_0)$ to determine the action of Frobenius on the fibre $H^n_{\mathrm{rig}}(U_{t_1})$ and hence the zeta function of the hypersurface X_{t_1}. We point out that there are essentially no original contributions from the author; we present the work of Lauder [48] and also follow the description by Gerkmann [30], focussing on the details relevant to explicit computations. The improvements in practical runtimes that we observe in this part of the algorithm are due to improved estimates of certain p-adic valuations by Kedlaya [44] and estimates of pole orders by Kedlaya and Tuitman [46] as well as implementation details.

5.1 p-Adic differential system and local expansion of Frobenius

Recall that we let Φ denote the matrix for the σ-semilinear action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$, where σ is the standard lift of the pth-power Frobenius to $\mathbf{Q}_q$ and $\mathbf{Q}_q(t)$ sending $t \mapsto t^p$, and M the matrix for the Leibniz-linear connection on $H^n_{\mathrm{dR}}(\mathfrak{U}/\mathfrak{S})$. As described for

example by Gerkmann [30, §5], these matrices satisfy the following differential system,

$$\Big(\frac{d}{dt} + M\Big)\Phi = pt^{p-1}\Phi\sigma(M).$$

Our first goal, discussed in this section, is the computation of a power series expansion of Φ around the origin, satisfying the above differential equation and the initial condition that Φ_0 be the matrix for the action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_0)$. We begin by observing that, upon setting

$$\Phi = C\Phi_0\sigma(C)^{-1},$$

the above differential system is equivalent to the homogeneous system

$$\Big(\frac{d}{dt} + M\Big)C = 0 \tag{5.1}$$

with the initial condition that $C(0)$ is the identity matrix. Let us write

$$M(t) = \frac{B(t)}{r(t)}$$

where $B(t) = \sum_{i=0}^{\deg(B)} B_i t^i$ is a matrix over $\mathbf{Q}[t]$ with $\deg(B) = \max_{i,j}\deg(B_{ij})$ and $r(t) \in \mathbf{Z}[t]$ is the *Macaulay resultant* for the polynomial P. We can now obtain a power series solution $C(t) = \sum_{i\geq 0} C_i t^i$ around $t = 0$ for

$$r(t)\frac{dC}{dt} + B(t)C(t) = 0$$

using the recursion

$$C_{i+1} = \frac{-1}{r_0(i+1)}\Bigg(\sum_{j=\max\{0,i-\deg(B)\}}^{i} B_{i-j}C_j + \sum_{j=\max\{0,i-\deg(r)\}+1}^{i} r_{i-j+1}jC_j\Bigg).$$

where we recall from Notation 2.2.1 that by assumption $r_0 \neq 0 \pmod p$ and so, in particular, $r_0 \neq 0$.

From a computational perspective, we are only interested in a power series solution to some p-adic and t-adic precisions N and K_2, respectively, as discussed in the next

two sections. As such we need to understand the loss of p-adic precision involved in the computation of $C_0, C_1, \ldots, C_{K_2-1}$. Specifically, let $D_0, D_1, \ldots$ denote an approximation defined by $D_0 = I$ and

$$D_{i+1} = \frac{-1}{r_0(i+1)} \left(\sum_{j=\max\{0, i-\deg(B)\}}^{i} B_{i-j} D_j + \sum_{j=\max\{0, i-\deg(r)\}+1}^{i} r_{i-j+1} j D_j \right) + p^{\tilde{N}} E_{i+1},$$

where $(E_i)_{i \geq 1}$ is a sequence of p-adically integral matrices.

The relationship between the desired precision N and the working precision $\tilde{N}$ is described by Lauder [49, Theorem 5.1], but this can be improved significantly by including recent bounds on the valuation of $C_0, C_1, \ldots$ by Kedlaya [44].

Lemma 5.1.1. Recall that the matrix Φ for the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$ has entries in $\mathbf{Q}_p\langle t, r(t)^{-1}\rangle^{\dagger}$, and let Φ_t denote the matrix for the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_t)$ with entries in $\mathbf{Q}_p$. Let us set

$$r = \mathrm{ord}_p((n-1)!),$$
$$s = (n+1)\left\lfloor \log_p(n-1) \right\rfloor.$$

and define $\delta = r + s$. The p-adic valuation of the matrices Φ and Φ_t is then at least $-\delta$.

Proof. This follows from Gerkmann [30, Lemma 3.3]. □

Theorem 5.1.2. For all $i \geq 0$, the p-adic valuation of the matrix C_i can be bounded by

$$\mathrm{ord}_p(C_i) \geq -\big(2\delta + (n-1)\big)\left\lceil \log_p i \right\rceil.$$

Proof. Let us recall from Lemma 5.1.1 that $\mathrm{ord}_p(\Phi) \geq -\delta$, and hence $\mathrm{ord}_p(p^{n-1}\Phi^{-1}) \geq -\delta$ by Poincaré duality. Thus, we see that

$$\mathrm{ord}_p(\Phi) + \mathrm{ord}_p(\Phi^{-1}) \geq -2\delta - (n-1)$$

and the bound follows now from Theorem 18.3.3 by Kedlaya [44]. □

Remark 5.1.3. Kedlaya [44, Remark 18.3.4] also includes another bound,

$$\operatorname{ord}_p(C_i) \geq (b-1)\operatorname{ord}_p(M) - \big(2\delta + (n-1)\big)\left\lfloor \log_p i \right\rfloor,$$

which is better than the estimate used in the previous theorem whenever $\operatorname{ord}_p(M)$ is positive.

Theorem 5.1.4. Let the sequences of matrices (C_i) and (D_i) over $\mathbf{Q}$ and $\mathbf{Q}_p$, respectively, be defined as above. Then, for all $i \geq 0$,

$$\operatorname{ord}_p(C_i - D_i) \geq \tilde{N} - \Big(2\big(2\delta + (n-1)\big) + 1\Big)\left\lceil \log_p i \right\rceil.$$

Proof. This follows analogously to the result of Lauder [49, Theorem 5.1]. Indeed, its proof shows more specifically that

$$\operatorname{ord}_p(C_i - D_i) \geq \tilde{N} + \min_{k+\ell=i}\Big(\operatorname{ord}_p(C_k) + \operatorname{ord}_p(\ell^{-1}C_{\ell-1}^{-1})\Big).$$

We use the bound from Theorem 5.1.2 and observe that they also apply to the inverse matrix C^{-1}, as this matrix satisfies the related differential equation

$$\Big(\frac{d}{dt} - M^t\Big)\big(C^{-1}\big)^t = 0. \tag{5.2}$$

The result now follows. □

Remark 5.1.5. In order to determine the power series expansion of the matrix Φ, we also need to compute the matrix $\sigma(C)^{-1}$. As we assume that the family $\mathcal{X}$ of hypersurfaces is defined over $\mathbf{Z}_p$, the connection matrix M is a matrix over $\mathbf{Q}_p[t, r(t)^{-1}]$ and the local solution C has entries in $\mathbf{Q}_p[[t]]$. As σ acts trivially on $\mathbf{Q}_p$, we have that $\sigma(C)^{-1} = C(t^p)^{-1} = C^{-1}(t^p)$.

The matrix C^{-1} could be computed using matrix inversion over the ring $\mathbf{Q}_p[[t]]$. An alternative approach follows from observing that whenever C satisfies equation (5.1), its inverse C^{-1} satisfies the related differential equation (5.2). In practice, solving this equation for C^{-1} is typically favourable compared to inverting the matrix C.

With p-adic approximations to the power series modulo t^{K_2} available for C, Φ_0, and $\sigma(C)^{-1}$, we are in a position to determine the power series for $\Phi = C\Phi_0\sigma(C)^{-1}$. The final issue to address in this section is the p-adic precision that we require for the three matrices C, Φ_0, and C^{-1}. We begin by developing generic bounds on the products of p-adic numbers and matrices.

Proposition 5.1.6. Let $x_1, \ldots, x_\ell \in \mathbf{Q}_p$, where $\ell \geq 2$, be given as $x_i = p^{v_i}u_i$ with $v_i = \mathrm{ord}_p(x_i) \in \mathbf{Z}$ and $u_i \in \mathbf{Z}_p^\times$ whenever $x_i \neq 0$ and $u_i = v_i = 0$ otherwise. Suppose that $N \in \mathbf{Z}$ is given such that $N > \sum_{j=1}^{\ell} v_j$ and, for all i, $N > \sum_{j\neq i} v_j$.

Let $\tilde{x}_1, \ldots, \tilde{x}_\ell$ be p-adic approximations satisfying $\mathrm{ord}_p(x_i - \tilde{x}_i) \geq N - \sum_{j\neq i} v_j$ for all i. Then

$$\mathrm{ord}_p(x_1 \cdots x_\ell - \tilde{x}_1 \cdots \tilde{x}_\ell) \geq N.$$

Proof. First, note that $x_i \neq 0$ implies that $\tilde{x}_i \neq 0$. Otherwise, if $\tilde{x}_i = 0$ then

$$N - \sum_{j\neq i} v_j \leq \mathrm{ord}_p(x_i - \tilde{x}_i) = \mathrm{ord}_p(x_i) = v_i$$

which implies that $N \leq \sum v_j$, a contradiction.

Represent $\tilde{x}_i$ as $p^{\tilde{v}_i}\tilde{u}_i$ with the same convention that $\tilde{u}_i \in \mathbf{Z}_p^\times$ unless $\tilde{x}_i = 0$, in which case $\tilde{u}_i = \tilde{v}_i = 0$. We first show that $v_i = \tilde{v}_i$ whenever $x_i \neq 0$. Indeed, if $v_i \neq \tilde{v}_i$ then

$$\mathrm{ord}_p(x_i - \tilde{x}_i) = \min\{\mathrm{ord}_p(x_i), \mathrm{ord}_p(\tilde{x}_i)\} \leq \mathrm{ord}_p(x_i) = v_i$$

whereas from the assumptions we derive

$$\mathrm{ord}_p(x_i - \tilde{x}_i) \geq N - \sum_{j\neq i} v_j > v_i,$$

a contradiction.

Note that, for all i such that $x_i \neq 0$, the assumption $\mathrm{ord}_p(x_i - \tilde{x}_i) \geq N - \sum_{j\neq i} v_j$ implies that $\mathrm{ord}_p(u_i - \tilde{u}_i) \geq N - \sum_j v_j$. Specifically,

$$\mathrm{ord}_p(u_i - \tilde{u}_i) = \mathrm{ord}_p(x_i - \tilde{x}_i) - v_i \geq N - \sum_{j=1}^{\ell} v_j.$$

Considering the product $x_1 \cdots x_\ell$, we distinguish two cases. First, assume that $x_i \neq 0$ for all i. We find that

$$\operatorname{ord}_p(x_1 \cdots x_\ell - \tilde{x}_1 \cdots \tilde{x}_\ell) = \sum_j v_j + \operatorname{ord}_p(u_1 \cdots u_\ell - \tilde{u}_1 \cdots \tilde{u}_\ell) \geq N.$$

In the other case, suppose that for some $k < \ell$, $x_1, \ldots, x_k$ are non-zero and that $x_{k+1}, \ldots, x_\ell$ are all zero. In particular, $v_{k+1} = \cdots = v_\ell = 0$. Then

$$\begin{aligned}
\operatorname{ord}_p(x_1 \cdots x_\ell - \tilde{x}_1 \cdots \tilde{x}_\ell) &= \operatorname{ord}_p(\tilde{x}_1 \cdots \tilde{x}_\ell) \\
&\geq v_1 + \cdots + v_k + N - \sum_{j \neq k+1} v_j + \cdots + N - \sum_{j \neq \ell} v_j \\
&= (\ell - k)N - (\ell - k - 1)(v_1 + \cdots + v_k) \\
&> N.
\end{aligned}$$

□

Proposition 5.1.7. Let $A_1, \ldots, A_\ell$ be $b \times b$ matrices over $\mathbf{Q}_p$, where $\ell \geq 2$, given as $A_i = p^{v_i} U_i$ with $v_i = \operatorname{ord}_p(A_i) \in \mathbf{Z}$ and at least one entry of U_i a p-adic unit $A_i \neq 0$ and $v_i = 0$ and U_i the zero matrix otherwise. Suppose that $N \in \mathbf{Z}$ is given such that $N > \sum_{j=1}^{\ell} v_j$ and, for all i, $N > \sum_{j \neq i} v_j$.

Let $\tilde{A}_1, \ldots, \tilde{A}_\ell$ be p-adic approximations satisfying $\operatorname{ord}_p(A_i - \tilde{A}_i) \geq N - \sum_{j \neq i} v_j$ for all i. Then

$$\operatorname{ord}_p(A_1 \cdots A_\ell - \tilde{A}_1 \cdots \tilde{A}_\ell) \geq N.$$

Proof. We can follow the proof of Proposition 5.1.6, observing that, for matrices A, B over $\mathbf{Q}_p$, we have $\operatorname{ord}_p(A + B) \geq \min\{\operatorname{ord}_p(A), \operatorname{ord}_p(B)\}$. □

Corollary 5.1.8. Suppose that K_2, N_2 are positive integers such that N_2 is greater than the sum of two or three of $\operatorname{ord}_p(\Phi_0)$, $\operatorname{ord}_p\big(C \bmod t^{K_2}\big)$, and $\operatorname{ord}_p\big(C^{-1} \bmod t^{\lceil K_2/p \rceil}\big)$. In order to compute the power series expansion around the origin of the matrix Φ modulo t^{K_2} to p-adic precision N_2, it suffices to compute the matrices C, C^{-1} and Φ_0 to p-adic precision N_3, N_3' and N_4, respectively, where

$$\begin{aligned}
N_3 &\geq N_2 + \big(2\delta + (n-1)\big)\left\lceil \log_p \lceil K_2/p \rceil \right\rceil + \delta, \\
N_3' &\geq N_2 + \big(2\delta + (n-1)\big)\left\lceil \log_p K_2 \right\rceil + \delta,
\end{aligned}$$

$$N_4 \geq N_2 + \big(2\delta + (n-1)\big)\big(\lceil \log_p K_2 \rceil + \lceil \log_p \lceil K_2/p \rceil \rceil\big).$$

Proof. This follows from Proposition 5.1.7, the bounds on the valuations of C and C^{-1} from Theorem 5.1.2 and the bound $\mathrm{ord}_p(\Phi_0) \geq -\delta$ from Lemma 5.1.1. □

5.2 Analytic continuation and evaluation

In the previous section we described how to compute the matrix Φ for the action of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$ as a local power series expansion around $t = 0$. In order to evaluate it at a second point $t = t_1$, we wish to p-adically approximate it using rational functions with denominators a power of $r(t)$, which is possible as the entries of the matrix Φ are *overconvergent functions* as defined in Definition 2.2.4 and described e.g. by Lauder [49, §3.5].

This step, the analytic continuation of the local expansion, is described by Lauder in [49, §5.2] and the computational details are explained in [47, §8.1]. Combining this step with the subsequent evaluation, our specific problem is the following. Given a desired p-adic precision N_2, we would like to determine integers K_1 and K_2 such that, modulo p^{N_2}, the entries of the matrix $\Psi(t) = r(t)^m\Phi(t)$ truncated modulo t^{K_2} allow us to compute the matrix Φ_{t_1} for the action of $p^{-1}F_p$ on $H^n_{\mathrm{dR}}(\mathfrak{U}_{t_1})$ as $r(\hat{t_1})^{-1}\Psi(\hat{t_1})$ modulo p^{N_2}.

Besides the work of Lauder [47, §8.1], Gerkmann also described suitable results in [30, §6], however the estimates are not sharp in practice. There has been recent progress by Kedlaya and Tuitman [46, Theorem 2.1], which we present in a slightly simplified and weakened form:

Theorem 5.2.1. Let $\mathfrak{Z}$ denote the complement of the open dense subscheme $\mathfrak{S}$ of $\mathbf{P}^1(\mathbf{Q}_q)$ and let z be an unramified geometric point of $\mathfrak{Z}$ such that $\mathfrak{Z}$ does not contain any other points with the same reduction modulo p. For a fixed a basis for $H^n_{\mathrm{rig}}(U/S)$, suppose that the matrix for the connection ∇ has at most a simple pole at z and that the exponents $\lambda_1, \ldots, \lambda_b$, which are defined as the eigenvalues of $(t-z)M|_{t=z}$ and known to be rational numbers, have non-negative p-adic valuation. Then the matrix Φ for the

action of $p^{-1}\,\mathcal{F}_p$ has a pole at z of order at most

$$\max_i \lambda_i - p \min_i \lambda_i + pg(N) \tag{5.3}$$

where $g(N)$ is defined by

$$g(N) = \max\Big\{ i \in \mathbf{N} : i - \delta - \big(2\delta + (n-1)\big)\,\lceil \log_p i \rceil < N \Big\}. \tag{5.4}$$

Proof. See [46, Theorem 2.1]. □

Remark 5.2.2. In practice, it might be convenient to avoid computing the exponents and verifying the hypotheses of the previous theorem. This is particularly relevant since even when the hypotheses are not satisfied, the required bounds are often closer to the result of Kedlaya and Tuitman than to the estimates provided by Gerkmann.

The contribution of the terms $\max_i \lambda_i - p \min_i \lambda_i$ in equation (5.3) typically is small compared to the term $pg(N_2)$. Therefore, in a heuristic implementation, one could choose e.g. $K_1 = 1.1 \times pN_2$ and $K_2 = (\deg(r) + 1)K_1$.

Having obtained an expression for the action of $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_{t_1})$ in terms of rational functions, we conclude this section by describing the evaluation at the fibre $t = t_1$. Modulo p^{N_2}, the matrix Φ_{t_1} representing $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_{t_1})$ is equal to

$$\Phi_{t_1} = r(\hat{t}_1)^{-K_1} \Big(r(t)^{K_1} \Phi(t) \bmod t^{K_2} \Big)\Big|_{t=\hat{t}_1} \pmod{p^{N_2}}. \tag{5.5}$$

Since, by assumption, $r(t_1)$ is a p-adic unit and hence so is $r(\hat{t}_1)^{-K_1}$ we observe that it suffices to compute the Teichmüller lift $\hat{t}_1$ and the matrix $r(t)^{K_1}\Phi(t)$ over $\mathbf{Q}_p[t]$ to p-adic precision N_2.

5.3 Recovering zeta functions

Recall that the zeta function of the hypersurface X_{t_1} is of the form,

$$Z(X_{t_1}, T) = \frac{p(T)^{(-1)^n}}{(1-T)(1-qT)\cdots(1-q^{n-1}T)}$$

where $p(T) = \det\big(1 - Tq^{-1}\,\mathcal{F}_q\,|H^n_{\mathrm{dR}}(\mathfrak{U}_{t_1})\big)$ is a polynomial defined over the integers.

Thus, the remaining two steps in the deformation method are as follows. Firstly, we obtain the matrix of the qth-power Frobenius from the matrix of the pth-power Frobenius. Secondly, we compute its reverse characteristic polynomial to suitable p-adic precision in order to recover the zeta function.

5.3.1 Computing the action of $q^{-1}\,\mathcal{F}_q$ on $H^n_{\mathbf{rig}}(U_{t_1})$

Let Φ_{t_1} and $\Phi^{(q)}_{t_1}$ denote the matrices representing the actions of $p^{-1}\,\mathcal{F}_p$ and $q^{-1}\,\mathcal{F}_q$ on $H^n_{\mathrm{rig}}(U_{t_1})$, respectively. As the action of $p^{-1}\,\mathcal{F}_p$ is σ-semilinear, we have that

$$\Phi^{(q)}_{t_1} = \Phi_{t_1}\sigma(\Phi_{t_1})\cdots\sigma^{a-1}(\Phi_{t_1}),$$

where $a = \log_p q$. Note that the lift of Frobenius σ is valuation preserving and hence the valuations of the matrices $\Phi_{t_1}, \Phi^{\sigma}_{t_1}, \ldots, \Phi^{\sigma^{a-1}}_{t_1}$ are at least $-\delta$ by Lemma 5.1.1.

It now follows from Proposition 5.1.7 that it suffices to compute the matrix Φ_{t_1} to p-adic precision N_2 at least $N_1 + (a-1)\delta$ in order to determine $\Phi^{(q)}_{t_1}$ to p-adic precision N_1.

5.3.2 Computing characteristic polynomials

In this section we address the precision loss when computing the reverse characteristic polynomial $\det(1 - tA)$ of a $b \times b$ matrix A given to finite precision over $\mathbf{Q}_p$.

In general, from the definition of the determinant function and Proposition 5.1.6, it appears that the precision loss could be as great as $(b-1)\,\mathrm{ord}_p(A)$. However, in the case of matrices representing the action of Frobenius on $H^n_{\mathrm{rig}}(U_{t_1})$, much better bounds are available.

Lemma 5.3.1. Define $\delta = \delta(n,p)$ as in Lemma 5.1.1 and suppose that $\tilde{\Phi}^{(q)}_{t_1}$ is an approximation to $\Phi^{(q)}_{t_1}$ with $\mathrm{ord}_p\big(\Phi^{(q)}_{t_1} - \tilde{\Phi}^{(q)}_{t_1}\big) \geq N + \delta$. Then

$$\mathrm{ord}_p\Big(\det\big(1 - t\Phi^{(q)}_{t_1}\big) - \det\big(1 - t\tilde{\Phi}^{(q)}_{t_1}\big)\Big) \geq N.$$

Proof. See Gerkmann [30, Lemma 3.3, Lemma 3.4]. □

The previous lemma allows us to take $N_1 = N_0 + \delta$ in our description of the deformation algorithm.

5.3.3 Computing Weil polynomials

In the final step of the deformation method, we compute the reverse characteristic polynomial $p(T)$ of the matrix $\Phi_{t_1}^{(q)}$, which represents $q^{-1}\,\mathcal{F}_q$ on $H^n_{\mathrm{rig}}(U_{t_1})$, to some finite p-adic precision. Which precision N_0 is necessary in order to correctly recover the polynomial $p(T)$ over the integers?

Theorem 5.3.2. In order to recover $p(T)$ over $\mathbf{Z}$ it suffices to compute an approximation modulo p^N where

$$p^N > 2 \max_{0 \le i \le b} \binom{b}{i} q^{i(n-1)/2}.$$

Moreover, this can be improved to

$$p^N > 2 \binom{b}{\lfloor b/2 \rfloor} q^{\lfloor b/2 \rfloor (n-1)/2}$$

provided that the sign $\epsilon = \mathrm{sgn}(\det(F_q))$ of the functional equation for the zeta function is known.

Proof. This is a slight reformulation of Theorem 3.2 in [30], which follows readily from its proof. □

Remark 5.3.3. We observe that the sign $\epsilon = \mathrm{sgn}(\det(q^{-1}\,\mathcal{F}_q)) = 1$ whenever n is even. This can be seen from the statement of the Weil conjectures in Theorem 1.1.2.

Remark 5.3.4. In the case that n is odd, the sign ϵ of the functional equation is unknown a priori. However, in practice it is often still possible to avoid having to use the greater of the two precisions in Theorem 5.3.2 by just computing one additional coefficient exactly. Writing $p(T) = \sum_{i=0}^{b} a_i T^i$, we can recover $a_0, \ldots, a_{\lfloor b/2 \rfloor + 1}$ exactly provided

$$p^N > 2 \max \left\{ \binom{b}{\lfloor b/2 \rfloor} q^{\lfloor b/2 \rfloor (n-1)/2}, \binom{b}{\lfloor b/2 \rfloor + 1} q^{(\lfloor b/2 \rfloor + 1)(n-1)/2} \right\}.$$

This allows us to determine the sign ϵ from the two coefficients $a_{\lceil b/2 \rceil - 1}$ and $a_{\lfloor b/2 \rfloor + 1}$, provided that they are non-zero, and we can then recover the remaining coefficients using the functional equation.

Remark 5.3.5. In the case of smooth projective surfaces, when $p > 2$ and subject to certain technical conditions, we can often exploit the growing divisibility of the coefficients of $p(T)$ ensured by the Hodge polygon. For such a surface of degree d, we know that the Hodge numbers $h_{0,2}$, $h_{1,1}$ and $h_{2,0}$ satisfy $h_{0,2} = h_{2,0} = \binom{d-1}{3}$ and $2h_{0,2} + h_{1,1} = b$. It is now easier to determine the integer coefficients of the polynomial $q^{h_{0,2}} p(T/q)$ as the roots of the polynomial $p(T/q)$ lie on the unit circle. This allows us to take

$$N_0 = ah_{0,2} + \left\lfloor \log_p \left(2 \binom{b}{\lfloor b/2 \rfloor} \right) \right\rfloor + 1,$$
$$N_1 = N_0 + a,$$

where N_0 here refers to the required precision for $q^{h_{0,2}} p(T/q)$ and N_1 as before refers to the precision required for the matrix representing $q^{-1} \mathcal{F}_q$ on $H^n_{\mathrm{rig}}(U_{t_1})$. For further details, we refer the reader to Lauder [49, §9.3.2, Proposition 9.6].

Chapter 6

Examples

In this chapter we present a small selection of computational examples, demonstrating the usefulness of the deformation method for point counting in practice.

We point out that the zeta functions that we compute in our examples are correct. However, the computations are not *a priori* provably correct as we are using the heuristic t-adic bounds from Remark 5.2.2. In some cases, we explicitly verify the computation by considering the exponents of the connection matrix occurring in Theorem 5.2.1. In other cases, we can rerun the computation with sufficiently increased precisions. Note that the former verification, applicable when the hypotheses of Theorem 5.2.1 are satisfied, is not time-consuming when compared to the remaining parts of the deformation method. The omission is a consequence of our implementation[1] in the C language on top of FLINT [33], lacking the required functionality of root finding algorithms and number field arithmetic that is necessary for the computation of the exponents.

For practical purposes, this does not present a problem at all: we know that the correct output $p(T)$ is the reverse of a Weil polynomial, that is, it is a polynomial with constant term one and all its roots have the same absolute value. This property allows us to at least heuristically verify the output, even when computed with insufficient precision. Further work has been carried out by Kedlaya [43] on verifying the correctness of a computation that has been carried out with intermediate p-adic precisions too low to be provably correct. The methods investigated and implemented are crucially based

[1] The source code is available in a git repository at `https://github.com/SPancratz/deformation`.

on the fact that the polynomial $\det\bigl(1 - Tq^{-1}\,\mathcal{F}_q\,|H^n_{\mathrm{dR}}(\mathfrak{U}_{t_1})\bigr)$ is the reverse of a Weil polynomial, and in particular that its roots all have absolute value $q^{(1-n)/2}$ and that the bottom coefficients can be obtained correctly even with low p-adic precision, imposing further restrictions on the remaining coefficients.

The timings that we present in this chapter were obtained on the departmental machine `wolverine.maths.ox.ac.uk`, featuring twenty-four Intel Xeon processors running at 2.93GHz, although none of the computations utilise more than one processor. In the following sections, we describe some runtimes as *near instantaneous*, by which we mean that the resolution provided by the C language command `clock()` is insufficient.

6.1 A genus six curve

We consider the family of genus six curves given by

$$x_0^5 + x_1^5 + x_2^5 + tx_0x_1x_2^3 \in \mathbf{F}_3[t][x_0, x_1, x_2]$$

which Gerkmann considers in [30, §7.4] with $a = [\mathbf{F}_q : \mathbf{F}_p] = 40$ and t_1 a generator for $\mathbf{F}_{3^{40}}^{\times}$. As the dimension $n-1$ of the hypersurface X_{t_1} is odd, we only need to determine the bottom half of the coefficients of the polynomial $p(T)$ directly. In our computation we use the following choice of p-adic precisions $N_0 = N_1 = N_2 = 127$, $N_3 = 134$, $N_3' = 135$, $N_4 = 142$ and note that in the analytic continuation step we take $K_1 = 419$ and $K_2 = 2514$.

We now compare the performance of our implementation with the timings reported in [30, §7.4] where possible, providing Gerkmann's timings in parentheses. The computation of the Gauss–Manin connection is near instantaneous with our implementation. The computation of the Frobenius matrix on the diagonal fibre takes 0.04 seconds (5.86 minutes). The local solution $C(t)$, its inverse $C(t)^{-1}$ and the local expansion for $p^{-1}\,\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$ are computed in 0.38, 0.11 and 1.28 seconds, respectively, adding up to 1.77 seconds (83 seconds). The analytic continuation and subsequent evaluation at a Teichmüller lift of t_1 take 0.05 and 0.2 seconds, and the computation of the qth-power Frobenius requires another 7.33 seconds (101.4 minutes).

During the computation of the connection matrix over $\mathbf{Q}(t)$, we find that it has denominator $r(t) = 135t^5 + 15625$. We determine the exponents at each of the poles as -1 (with multiplicity 5) and 0 (with multiplicity 7), which allows us to justify the choice of precisions using Theorem 5.2.1.

6.2 A quartic surface

We consider another example that allows for comparison with Gerkmann [30, §7.5],

$$x_0^4 + x_1^4 + x_2^4 + x_3^4 + tx_0x_1x_2x_3 \in \mathbf{F}_3[t][x_0, x_1, x_2, x_3].$$

Here, we choose $a = 20$ and $t_1 = \alpha^{2345}$ where α is a generator for $\mathbf{F}_{3^{20}}^{\times}$. As the dimension $n - 1 = 2$ is even, a priori the sign of the functional equation is unknown. However, we can use Remark 5.3.5 and determine the polynomial $3^{20}p\big(3^{-20}T\big) \in \mathbf{Z}[T]$ instead of $p(T)$. Thus, we take $N_0 = 33$, $N_1 = N_2 = 53$, $N_3 = 65$, $N_3' = 67$, $N_4 = 79$ as well as $K_1 = 174$ and $K_2 = 870$.

The computation of the connection matrix is near instantaneous again, although in this case this is largely due to the sparseness of multivariate polynomial defining the family. The Frobenius matrix for the fibre at $t = 0$ is determined in 0.02 seconds (45.26 minutes). The local solutions $C(t)$ and $C(t)^{-1}$ require 0.27 and 0.08 seconds, which together with the computation local expansion of Frobenius in 0.15 seconds yields 0.5 seconds (19.9 minutes). The analytic continuation, evaluation at the Teichmüller lift of t_1 and construction of the qth-power Frobenius take 0.02, 0.04 and 0.62 seconds, respectively, adding up to 0.68 seconds (18.66 minutes). The root-unitary polynomial that we find is given by

$$\begin{aligned}
3^{20}p\big(3^{-20}T\big) = {} & -3486784401T^{21} - 39675197243T^{20} - 1915066148 66T^{19} \\
& - 482588946510T^{18} - 552821487569T^{17} + 243001138765T^{16} \\
& + 1641410078472T^{15} + 1793016627512T^{14} - 410199003010T^{13} \\
& - 2617001208822T^{12} - 1586643774924T^{11} + 1586643774924T^{10} \\
& + 2617001208822T^{9} + 410199003010T^{8} - 1793016627512T^{7}
\end{aligned}$$

$$- 1641410078472T^6 - 243001138765T^5 + 552821487569T^4$$
$$+ 482589946510T^3 + 191506614866T^2 + 39675197243T + 3486784401.$$

As before, we prove that our heuristic choice of t-adic precisions was sufficient by computing the exponents of the connection. We find that the connection matrix has denominator $r(t) = 2t^4 - 512$, which has distinct roots modulo 3, and at each of the four poles we compute the exponents as $-3/2$ (with multiplicity 1), $-1/2$ (with multiplicity 15), and 0 (with multiplicity 5). Now Theorem 5.2.1 establishes that the chosen parameters were sufficient.

6.3 A quintic surface

The final example from Gerkmann [30, §7.6], containing a diagonal fibre and hence being suitable for comparison, is given by

$$x_0^5 + x_1^5 + x_2^5 + x_3^5 + tx_0^2x_1x_2x_3 \in \mathbf{F}_2[t][x_0, x_1, x_2, x_3],$$

with the point t_1 chosen as a generator of $\mathbf{F}_{2^{10}}^\times$. Note that here we do *not* have $p \geq n$ so the Frobenius matrix is not automatically integral; specifically, we find that $\delta = 5$. Thus, we can not employ Remark 5.3.5. Moreover, the dimension $n-1 = 2$ of the hypersurface X_{t_1} is even and hence the sign of the functional equation is a priori unknown. We choose the p-adic precisions $N_0 = 522$, $N_1 = 527$, $N_2 = 572$, $N_3 = 721$, $N_3' = 890$, and $N_4 = 872$ and let $K_1 = 1159$, $K_2 = 6954$.

The computation of the connection matrix, which has density $98/52^2$ or approximately 0.036, requires 0.07 seconds (40 seconds). The Frobenius matrix on the diagonal fibre requires 3.4 seconds (64.13 minutes). The computations of the matrices $C(t)$ and $C(t)^{-1}$ are computed in 33.05 and 17.07 seconds, and the matrix product $C(t)F(0)C^{-1}(t^p)$ is requires another 40.08 seconds, leading to an overall time of 90.2 seconds (31.03 minutes) for the local expansion of Frobenius. Finally, the analytic continuation, evaluation at $\hat{t}_1$ and computation of the qth-power Frobenius take 3.09, 0.66, and 1.16 seconds, yielding a total of 4.91 seconds (4.9 hours) for this part of the computation.

We note that Gerkmann observes that the family in this example does not contain any singular fibres, which implies that a much lower value of K_1 would be sufficient. This observation is important here as we cannot apply Theorem 5.2.1 in this case. We find the denominator of the connection matrix $r(t) = 20t^5 + 15625$ and compute the exponents $-3/2$ (with multiplicity 4), $-1/2$ (with multiplicity 21), and 0 (with multiplicity 27), only to observe that some of them have negative 2-adic valuation.

6.4 A cyclic cubic threefold

We now consider the family of threefolds given by

$$-x_0^3 + x_1^3 + x_2^3 + x_3^3 + x_4^3 + t\big((x_1 + x_2)(x_1 + 2x_3)(x_1 + 3x_4) + 3x_2x_3(x_1 + x_2 + x_4)\big)$$

defined over $\mathbf{F}_7$. With the final fibre defined by $t_1 = 1$, this is an example that is extensively discussed by Kedlaya [45, Example 1.6.1].

Our work demonstrates that examples like this one can be computed in an automated fashion in short time. As the dimension $n - 1 = 3$ is odd, our choice of p-adic precisions is $N_0 = N_1 = N_2 = 12$, $N_3 = 24$, $N_3' = 27$, $N_4 = 39$ and we have heuristic t-adic parameters $K_1 = 92$ and $K_2 = 5796$.

The Gauss–Manin connection matrix is a 10×10 matrix with 50 non-zero entries. We find that the denominator $r(t)$ is a degree 62 polynomial and has $r(1) \equiv 2 \pmod 7$. This part of the computation requires 0.89 seconds. The Frobenius matrix for the diagonal fibre is obtained in 0.01 seconds. The matrices $C(t)$ and $C(t)^{-1}$ are computed in 9.99 and 1.34 seconds, respectively, followed by a matrix product to compute the local expansion of Frobenius at $t = 0$ in 3.67 seconds. The analytic continuation step now requires 0.21 seconds, but the remaining part of the computation is near instantaneous.

We also verify that our output polynomial

$$\begin{aligned} p(T) = {} & 4747561509943T^{10} + 96889010407T^9 + 17795940687T^8 + 807072147T^7 \\ & - 25529833T^6 - 756315T^5 - 74431T^4 + 686T^3 + 441T^2 + 7T + 1 \end{aligned}$$

agrees with Kedlaya's work [45, p. 20].

6.5 A generic quintic curve

A family of quintic curves is given by a homogeneous polynomial in three variables of degree 5, which generically has $\binom{3+5-1}{5} = 21$ non-zero coefficients. We choose

$$\begin{aligned} x_0^5 + x_1^5 + x_2^5 + t\big(&3x_0^4x_1 - x_0^4x_2 + 2x_0x_1^4 + x_1^4x_2 + 4x_0x_2^4 + 5x_1x_2^4 + x_0^3x_1^2 + x_0^3x_2^2 \\ &+ x_0^2x_1^3 + x_1^3x_2^2 + x_0^2x_2^3 + x_1^2x_2^3 + x_0^3x_1x_2 + x_0x_1^3x_2 + x_0x_1x_2^3 \\ &+ x_0^2x_1^2x_2 + x_0^2x_1x_2^2 + x_0x_1^2x_2^2\big) \end{aligned}$$

over $\mathbf{F}_{11}$ and $t_1 = 1$.

We note that $n - 1 = 1$ is odd, that the sign of the functional equation is positive and hence we only need to determine the bottom half of the coefficients directly. This allows us to choose the p-adic precisions as $N_0 = N_1 = N_2 = 7$, $N_3 = 10$, $N_3' = 11$, and $N_4 = 14$ and let $K_1 = 84$, $K_2 = 6636$.

The computation of the connection matrix requires 6.55 seconds, yielding a dense matrix with denominator $r(t)$ a degree 78 polynomial, and we verify that $r(1) \equiv 3 \pmod{11}$. We obtain the Frobenius matrix for the diagonal fibre in 0.02 seconds. The local solutions $C(t)$ and $C(t)^{-1}$ are computed in 36.64 and 2.75 seconds, respectively, which then allows us to compute the local expansion of the Frobenius matrix around $t = 0$ in an additional 14.45 seconds. Finally, the analytic continuation and evaluation steps require 0.54 and 0.02 seconds.

The output of the computation is the polynomial

$$\begin{aligned} p(T) = 1771561T^{12} &- 322102T^{11} + 14641T^{10} + 17303T^9 - 1936T^8 \\ &+ 2310T^7 - 250T^6 + 210T^5 - 16T^4 + 13T^3 + T^2 - 2T + 1. \end{aligned}$$

We heuristically confirm the correctness of this result by numerically computing its complex roots, finding that they have absolute value $11^{-1/2}$ as required.

6.6 A generic sextic curve

A family of sextic curves is given by a homogeneous polynomial in three variables of degree 6, which generically has $\binom{3+6-1}{6} = 28$ non-zero coefficients. We choose

$$\begin{aligned} x_0^6 + x_1^6 + x_2^6 + t\big(& - x_0^5x_1 + 7x_0^5x_2 + 2x_0x_1^5 + x_1^5x_2 + 2x_0x_2^5 + x_1x_2^5 + 2x_0^4x_1^2 + 2x_0^4x_2^2 \\ & + 3x_0^2x_1^4 + x_1^4x_2^2 + 3x_0^2x_2^4 + x_1^2x_0 + 2^4 + 3x_0^4x_1x_2 + 3x_0x_1^4x_2 + x_0x_1x_2^4 \\ & - x_0^3x_1^3 - 2x_0^3x_2^3 + 4x_1^3x_2^3 + 2x_0^3x_1^2x_2 + x_0^3x_1x_2^2 - x_0^2x_1^3x_2 + x_0x_1^3x_2^2 \\ & + 2x_0^2x_1x_2^3 + x_0x_1^2x_2^3 + x_0^2x_1^2x_2^2\big) \end{aligned}$$

over $\mathbf{F}_5$ and $t_1 = 2$.

In this example, we note that $n - 1 = 1$ is odd and hence we only need to determine the bottom half of the coefficients of $p(T)$ directly. The p-adic precisions we are led to use are $N_0 = N_1 = N_2 = 13$, $N_3 = 18$, $N_3' = 19$, and $N_4 = 24$, and we take heuristic t-adic parameters $K_1 = 71, K_2 = 8591$.

The Gauss–Manin connection matrix, which is dense and has a degree 120 denominator $r(t)$ in this example, is computed in 60.08 seconds. We also calculate $r(t_1) \equiv 1 \pmod 5$. The computation of the Frobenius matrix for the diagonal fibre takes 0.01 seconds. The matrices $C(t)$ and $C(t)^{-1}$ require 530.08 and 99.71 seconds, respectively, and we then compute the local expansion of the Frobenius matrix in 94.87 seconds. The remaining two steps, the analytic continuation and evaluation require 2.71 and 0.11 seconds, respectively.

The final result of the computation is the polynomial

$$\begin{aligned} p(T) = {} & 9765625T^{20} - 7812500T^{19} + 1562500T^{18} + 781250T^{17} - 531250T^{16} \\ & - 56250T^{15} + 108125T^{14} + 5375T^{13} - 21475T^{12} + 4945T^{11} - 503T^{10} \\ & + 989T^9 - 859T^8 + 43T^7 + 173T^6 - 18T^5 - 34T^4 + 10T^3 + 4T^2 - 4T + 1 \end{aligned}$$

We verify that the reverse of $p(T)$ is a Weil polynomial and numerically find that the absolute value of its complex roots is $5^{-1/2}$.

6.7 A generic quartic surface

We consider a generic quartic surface, defined by a homogeneous equation with 35 non-zero coefficients,

$$\begin{aligned} x_0^4 + x_1^4 + x_2^4 + x_3^4 + t\big(& - 3x_0^3x_1 + 2x_0^3x_2 - 2x_0x_1x_2x_3 + x_0^3x_3 - x_0x_1^3 - 3x_1^3x_2 + x_2^3x_3 \\ & + 2x_1^3x_3 + x_0x_2^3 - 2x_1x_2^3 - x_0x_3^3 + x_1x_3^3 + 3x_2x_3^3 + x_0^2x_1^2 \\ & + 3x_0^2x_2^2 + x_0^2x_3^2 + 2x_1^2x_2^2 - 2x_1^2x_3^2 + x_2^2x_3^2 + 2x_0^2x_1x_2 \\ & + x_0^2x_1x_3 + 3x_0^3x_2x_3 - x_0x_1^2x_2 + 2x_0x_1^2x_3 + 3x_1^2x_2x_3 - x_0x_1x_2^2 \\ & + 3x_0x_2^2x_3 + x_1x_2^2x_3 + 2x_0x_1x_3^2 + 2x_0x_2x_3^2 + 2x_1x_2x_3^2\big) \end{aligned}$$

defined over $\mathbf{F}_5$ with $t_1 = 1$.

This example is a surface and we have $p \geq n$, hence the matrix of Frobenius is automatically integral. We aim to compute $5p(T/5)$ instead of $p(T)$, following Remark 5.3.5. This allows us to take $N_0 = 10$, $N_1 = N_2 = 11$, $N_3 = 21$, $N_3' = 23$, and $N_4 = 33$, together with $K_1 = 60$ and $K_2 = 13740$.

The Gauss–Manin connection matrix is computed in 323.55 seconds and found to have denominator $r(t)$ of degree 228, which explains the large value of the t-adic parameter K_2. The matrix for $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U_0)$ is computed in 0.01 seconds. The local solutions C and C^{-1} form the most time-consuming part in this example, requiring 1909.3 and 342.22 seconds. The matrix product to determine the local expansion of $p^{-1}\mathcal{F}_p$ on $H^n_{\mathrm{rig}}(U/S)$ takes 262.6 seconds. Finally, the analytic continuation and evaluation steps require 4.53 and 0.18 seconds, respectively.

6.8 Another quartic surface

We consider family of quartic surfaces inspired by an example of Abbott, Kedlaya, and Roe [1, Example 4.2.1],

$$x_0^4 + x_1^4 + x_2^4 + x_3^4 + t\big(-x_1x_2^3 + x_0x_1x_2^2 + x_0x_1x_2x_3 + x_0^2x_1x_2 - x_0^2x_1x_3 + x_0x_2^3 - x_0x_2^2x_3\big)$$

defined over $\mathbf{F}_3$. They consider the smooth surface X_1 and report a runtime of $40,144$ seconds, or 11.15 hours. Unfortunately, with our current implementation we cannot tackle this exact example as, with our fixed choice of basis, the Gauss–Manin connection matrix has an apparent singularity at $t = 1$ (and also at $t = 2$). Instead, we let t_1 be a generator of $\mathbf{F}_{3^2}^{\times}$.

Once again, we can employ Remark 5.3.5, which here allows us to take p-adic precisions $N_0 = 15$, $N_1 = N_2 = 17$, $N_3 = 33$, $N_3' = 35$, $N_5 1$, and t-adic parameters $K_1 = 56$ and $K_2 = 9632$.

The computation of the Gauss–Manin connection matrix requires 36.6 seconds, which also shows that the denominator $r(t)$ is a degree 168 polynomial. The action of $p^{-1}\,\mathcal{F}_p$ for the diagonal fibre is determined in 0.01 seconds. The local solutions $C(t)$ and $C(t)^{-1}$ are found in 953.59 and 311 seconds, which then allow us to compute the local expansion of Frobenius in 153.44 seconds. The analytic expansion and evaluation require 3.77 and 0.24 seconds, respectively, and the final construction of the qth-power Frobenius is near instantaneous. This yields a total runtime of about $1,458$ seconds.

Part II

The fibration method and fast radix conversion for polynomials

Chapter 7

Introduction

In this part of the thesis, we describe various strategies to solve the problem of polynomial radix conversion. Two approaches, the naive repeated division by the radix and the divide and conquer strategies, are well known, see e.g. [63, §9.2]. We present a practical improvement to the latter approach based on the use of precomputed power series inverses. A number theoretical application is the fibration method for point counting, in current implementations of which the runtime in practice is dominated by radix conversions.

In its general form, the problem of polynomial radix conversion can be stated as follows:

Problem 7.0.1. Let R be a ring[1] and suppose that a polynomial $f \in R[X]$ is presented in the usual monomial basis as $f(X) = \sum_{i=0}^{n} a_i X^i$. Suppose that $r \in R[X]$ is another polynomial, called the *radix*, such that $m = \deg(r) \geq 1$ and the leading coefficient of r is invertible in R. Compute polynomials $b_0, \ldots, b_\ell \in R[X]$ in terms of the monomial basis such that

$$f = b_0 + b_1 r + \cdots b_\ell r^\ell$$

where necessarily $\ell = \lfloor n/m \rfloor$.

The existence and uniqueness of the sequence of polynomials $b_0, \ldots, b_\ell$ follows easily from the properties of Euclidean division; an explicit algorithm is presented in Sec-

[1] The term *ring* here means commutative ring with multiplicative identity.

tion 8.1.

To motivate the work in this context, we point out that Lauder's fibration algorithm for counting points on smooth projective varieties in practice heavily benefits from a fast implementation for this problem of radix conversions, see Lauder [49, §6.5.2] or Walker [64, §3.2.2]. In both these cases, the authors utilise the divide and conquer approach in their computations. In the fibration algorithm, one has to convert the power series of each entry in a local expansion of a Frobenius matrix. That is to say, the number of instances with the same radix polynomial is quadratic in the dimension of the middle-dimensional rigid cohomology space, which motivates our approach using precomputed power series inverses.

In this context, we consider our ring R to be the field $\mathbf{Q}_p$, although in practice computations will only be carried out to finite p-adic precision.

Problem 7.0.2. Given two polynomials $f, r \in \mathbf{Q}_p[X]$ such that $r \neq 0$ and the valuation of the leading coefficient of r is equal to $\mathrm{ord}_p(r)$, find a representation of f in the form

$$f = b_0 + b_1 r + \cdots + b_\ell r^\ell.$$

In the remaining part of this section, we describe how to reduce Problem 7.0.2 to the original Problem 7.0.1 and carry out the p-adic precision analysis.

Write $f = p^{\mathrm{ord}_p(f)} g$ and $r = p^{\mathrm{ord}_p(r)} s$ with two polynomials g and s in $\mathbf{Z}_p[X]$ each containing at least one coefficient which is a unit in $\mathbf{Z}_p$. The above equation then becomes

$$g = p^{-\mathrm{ord}_p(f)} b_0 + p^{-\mathrm{ord}_p(f)+\mathrm{ord}_p(r)} b_1 s + \cdots + p^{-\mathrm{ord}_p(f)+\ell\,\mathrm{ord}_p(r)} b_\ell s^\ell$$

and we also note that the leading coefficient of s is a unit in $\mathbf{Z}_p$. In order to solve Problem 7.0.2, we first solve the above for the polynomials g and s over $\mathbf{Z}_p$, obtaining a sequence of polynomials $c_0, \ldots, c_\ell$ in $\mathbf{Z}_p[X]$ such that

$$g = c_0 + c_1 s + \cdots + c_\ell s^\ell.$$

We can then recover the sequence $b_0, \ldots, b_\ell$ via

$$b_i = p^{\operatorname{ord}_p(f) - i \operatorname{ord}_p(r)} c_i.$$

We observe that, in order to determine the polynomial b_i modulo p^N we need to determine the polynomial c_i modulo $p^{N+i \operatorname{ord}_p(r) - \operatorname{ord}_p(f)}$. Maximising this over the whole sequence of polynomials, we see that we need to determine the polynomials $c_0, \ldots, c_\ell$ modulo $p^{N+\eta}$ where

$$\eta = -\operatorname{ord}_p(f) + \max\{\ell \operatorname{ord}_p(r), 0\}.$$

Thus, we require that the polynomial $f \in \mathbf{Q}_p[X]$ is provided to precision

$$N + \eta + \operatorname{ord}_p(f) = N + \max\{\ell \operatorname{ord}_p(r), 0\}.$$

and that $r \in \mathbf{Q}_p[X]$ is provided to precision

$$N + \eta + \operatorname{ord}_p(r) = N - \operatorname{ord}_p(f) + \operatorname{ord}_p(r) + \max\{\ell \operatorname{ord}_p(r), 0\}.$$

Chapter 8

Algorithms

8.1 Euclidean division and complexity results

In this section we collect a few well known results for later reference.

Proposition 8.1.1. Let R be a ring and suppose that $f, r \in R[X]$ with the leading coefficient of r a unit in R. Then there exist unique polynomials $q, s \in R[X]$ such that $f = qr + s$ with $s = 0$ or $\deg(s) < \deg(r)$.

This result is well known in the context of polynomial rings over fields. The usual proof of this result in $\mathbf{Q}[X]$ applies upon observing that only the inverse of the leading coefficient is required in the division process.

Proposition 8.1.2. There exists an algorithm returning the degree $2n$ product of two polynomials of degree n in $\mathcal{O}(n \log n \log\log n)$ ring operations.

Such an algorithm can be constructed based on Fast Fourier Transform techniques, see for example [63, §Theorem 8.23] or [6, §4].

Proposition 8.1.3. There exists an algorithm returning the degree n quotient in a Euclidean division with a dividend of degree $2n$ and a divisor of degree n in $\mathcal{O}(n \log n \log\log n)$ ring operations.

This result can be derived from the previous one by exhibiting an algorithm for Euclidean division with the same asymptotic time requirements as multiplication, see for example [6, §17].

Corollary 8.1.4. In the unbalanced case for a dividend of degree n and a divisor of degree m, where $m \leq n$, there exists an algorithm that returns the degree $n - m$ quotient in $\mathcal{O}(n \log m \log\log m)$ ring operations.

This can be achieved by splitting the dividend into parts of degree $2m$ and applying the previous algorithm in these balanced divisions, carefully combining the results.

8.2 Repeated division by the radix

The naive solution to the radix conversion problem consists of obtaining the sequence of polynomials $b_0, \ldots, b_\ell$ by performing ℓ Euclidean divisions with remainder. Letting DivRem denote a routine that performs Euclidean division, this approach can be formalised as follows:

Algorithm 8.1 Repeated division by the radix

```
procedure RadixConversion(b, f, r)
    f_0 ← f
    for i ← 0, ..., ℓ − 1 do
        f_{i+1}, b_i ← DivRem(f_i, r)
    b_ℓ ← f_ℓ
```

We can repeatedly apply the previous result on the complexity of unbalanced divisions, Corollary 8.1.4, with fixed divisor r. The conclusion is that the complexity of this algorithm is $\mathcal{O}\big(n^2 m^{-1} \log m \log\log m\big)$.

8.3 Divide and conquer

In this section we present another algorithm, which achieves a better time complexity. The driving idea is that we divide the polynomial f into two halves and then deal with each half separately. This recursive idea is formalised in Algorithm 8.2.

Algorithm 8.2 Recursive divide and conquer

```
procedure RCRecursive(b, f, r, k)
    N ← ⌊deg(f)/ deg(r)⌋
    if N = 0 then
        b_k ← f
    else
        e ← ⌈N/2⌉
        Q, S ← Divrem(f, r^e)
        RCRecursive(b, q, r, k + e)
        RCRecursive(b, s, r, k)
procedure RadixConversion(b, f, r)
    RCRecursive(b, f, r, 0)
```

Before we proceed with the analysis of this algorithm, we make the following assumption, which will allow for a cleaner presentation, analysis and implementation of the algorithm:

Notation 8.3.1. Assume that $n = 2^k m - 1$ for some $k \geq 1$.

Remark 8.3.2. Since all Euclidean divisions then involve a dividend of length $2^i m$ and a divisor of length $2^{i-1}m + 1$, where $i = k, \ldots, 1$, we can more efficiently re-use precomputed data such as powers and Newton inverses of the radix r.

Moreover, this assumption can easily be incorporated in an implementation by choosing the least integer k such that $n \leq 2^k m - 1$ and then zero-padding the top coefficients of f.

Remark 8.3.3. The runtime of an implementation which enforces the above assumption by zero-padding as a function of n is not smooth, featuring jumps as n changes from $2^k m - 1$ to $2^k m$. Although this does not affect the asymptotic behaviour, it is undesirable in practice. However, it is possible to combine the two approaches, for example, by following Algorithm 8.2 at the first level and only pad the lengths to powers of 2 as necessary at the subsequent levels.

In order to determine the complexity of Algorithm 8.2 we count the number of multiplications as well as the number of Euclidean divisions of each size.

First, we note that we can precompute r^{2^i} for $i = 1, \ldots, k - 1$ before proceeding with the Euclidean divisions, by computing $k - 1$ squares involving input polynomials

of length $2^i m + 1$ for $i = 0, \ldots, k-2$. This part of the algorithm has complexity

$$\mathcal{O}\Big(\sum_{i=0}^{k-2} \big(2^i m\big) \log\big(2^i m\big) \log\log\big(2^i m\big)\Big)$$

which can be simplified to $\mathcal{O}(n \log n \log\log n)$. By inspection, there are 2^i divisions with dividend of length $2^{k-i}m$ and divisor of length $2^{k-1-i}m + 1$ for $i = 0, \ldots, k-1$. Thus, the complexity is

$$\mathcal{O}\Big(\sum_{i=0}^{k-1} \big(2^{k-1-i} m + 1\big) \log\big(2^{k-1-i} m + 1\big) \log\log\big(2^{k-1-i} m + 1\big)\Big)$$

which can be simplified to $\mathcal{O}\big(n \log n \log\log n\big)$.

Thus, the overall complexity of Algorithm 8.2 is $\mathcal{O}(n \log n \log\log n)$.

8.4 Euclidean division with precomputed inverses

Since the first part of the divide and conquer algorithm, the precomputation of the powers of the radix, already forces a time complexity of at least $\mathcal{O}(n \log n \log\log n)$ ring operations we cannot hope for an asymptotic improvement.

However, by precomputing further data the cost of the repeated Euclidean divisions can be reduced significantly. This is particularly valuable in the context of the fibration algorithm where the operation of radix conversion is called repeatedly for varying polynomials f of similar degree but with an invariant radix r. The approach presented in this section precomputes power series inverses, allowing us to reduce the cost of a Euclidean division to two short products, that is, routines that only require the lower half of the coefficients of a degree $2n$ product of two degree n factors.

We now temporarily abandon the earlier notation to explain a procedure for Euclidean division with precomputed inverses. The algorithm, which is presented as Algorithm 8.3, assumes that we have two subroutines at our disposal. Firstly, a routine $\textsc{Rev}(f, n)$ that returns $X^{\deg(f)} f(X^{-1})$ modulo X^n, that is, a routine that simply reverses the top n coefficients of f, possibly including zero padding if f has degree less than $n - 1$. Secondly, a routine $\textsc{Inv}(f, n)$ that returns the power series inverse of f

modulo X^n.

Algorithm 8.3 Newton division

Input: Polynomials $a, b \in R[X]$ with $\deg(a) \geq \deg(b) > 0$ and the leading coefficient of b invertible.

Output: Polynomials $q, r \in R[X]$ such that $a = qb + r$ with $r = 0$ or $\deg(r) < \deg(b)$.

procedure DivRem(q, r, a, b)
  $n \leftarrow \deg(a) - \deg(b) + 1$
  $a_r \leftarrow \text{Rev}(a, n)$
  $b_r \leftarrow \text{Rev}(b, n)$
  ${b_r}^{-1} \leftarrow \text{Inv}(b_r, n)$
  $q_r \leftarrow a_r {b_r}^{-1} \bmod X^n$
  $q \leftarrow \text{Rev}(q_r, n)$
  $r \leftarrow a - qb$
  return q, r

Remark 8.4.1. The properties of Euclidean division imply that $\deg(r) < \deg(b)$, whenever r is non-zero, which in turn implies that the top $\deg(a) - \deg(b) + 1$ coefficients of a and qb coincide. Thus, in order to compute r we do *not* need to compute the full product qb. It suffices to compute the product modulo $X^{\deg(b)}$.

Theorem 8.4.2. Algorithm 8.3 correctly determines the quotient and remainder of the Euclidean division of a upon b.

Proof. Let v_0 and v_∞ denote the valuations on $R(X)$ at 0 and ∞, respectively. Note that, for $f \in R[X]$,

$$v_\infty(f) = -\deg(f).$$

We observe that $q, r \in R[X]$ such that $a = qb + r$ are the output of the Euclidean division if and only if

$$v_\infty(a - qb) > v_\infty(b).$$

Let $\tilde{b}^{-1}$ denote an approximation to b^{-1} for v_∞ and set $q = a\tilde{b}^{-1}$. Then

$$a - qb = ab\big(b^{-1} - \tilde{b}^{-1}\big),$$

and it suffices to have an approximation such that $v_\infty\big(b^{-1} - \tilde{b}^{-1}\big) > -v_\infty(a)$.

We now relate this to the reverse polynomials. Under the map $X \mapsto X^{-1}$ we have that v_∞ maps to v_0, and b maps to $X^{v_\infty(b)} b_r$, where b_r denotes the reverse polynomial

to b. Thus, in order to compute $\tilde{b}^{-1}$ such that $v_\infty\big(b^{-1} - \tilde{b}^{-1}\big) > -v_\infty(a)$ we have to compute an approximation $\tilde{b}_r^{-1}$ to b_r^{-1} for v_0 such that

$$v_0\big(X^{-v_\infty(b)}\big(b_r^{-1} - \tilde{b}_r^{-1}\big)\big) > -v_\infty(a)$$

or, equivalently,

$$v_0\big(b_r^{-1} - \tilde{b}_r^{-1}\big) > v_\infty(b) - v_\infty(a) = \deg(a) - \deg(b),$$

as required. □

Remark 8.4.3. In the notation of Algorithm 8.3, we observe that if the power series inverse b_r^{-1} is part of the input, the procedure reduces the Euclidean division to two short products, one subtraction, and one reversal.

Remark 8.4.4. In Algorithm 8.2, all short products involve either a power r^{2^i} of r, where $i = 0, \ldots, k-1$ or the power series inverse of such a power. Thus, a further practical improvement might be possible if the underlying arithmetic software packages of an implementation allow for re-using data relevant to only one of the factors in a polynomial multiplication.

Chapter 9

Examples

We have implemented a routine for radix conversion as described in the previous chapter in the C language as part of the open-source library FLINT [33]. Specifically, the routine is contained in the module for polynomial arithmetic over $\mathbf{Z}/n\mathbf{Z}$ as the function `fmpz_mod_poly_radix()`. The timings that we present for our routines were obtained on the departmental machine `wolverine.maths.ox.ac.uk`, featuring Intel Xeon processors running at 2.93GHz.

We consider an example due to Lauder [49, Example 9.1]. Specifically, we consider the toric compactification of the smooth surface given by

$$y^2 = x^3 + (4t^4 + 5t^3)x + (t^{13} + 6t^{12} + 5t^{10} + 8t^9 + 8t^8 + 5t^5 + t^4 + 5t^3 + t^2 + 1)$$

over $\mathbf{F}_{17}$. Lauder reports an overall runtime of 23 hours and 13 minutes, pointing out that over half the time was spent in the radix conversion routine.

In the notation used by Lauder, we reestablish the parameters used in the computation as $g = 1$, $N_3 = 18$, $x_{fin} = 476$ and $N_{2,fin} = 26$. Moreover, writing the equation defining the surface as $y^2 = x^3 + A(t)x + B(t)$ we compute the resultant as $4A(t)^3 + 27B(t)^2$, which is a polynomial of degree 26. As the $r(t)$-adic precision is given by x_{fin}, we find that the t-adic precision is $12,376$.

In order to establish a comparison of Lauder's computation with the routine we implemented, we perform a radix reduction of a random, dense polynomial of degree $12,376$

with coefficients in $\mathbf{Z}/17^{26}\mathbf{Z}$. The precomputation step of our routine, which computes powers of $r(t)$ and their power series inverses, requires 0.06 seconds. The actual conversion step for a single polynomial requires 0.14 seconds. As the relative Frobenius matrix in Lauder's example is a 2×2 matrix of power series, the relevant figure amounts to $0.06 + 4 \times 0.14 = 0.62$ seconds. This is an improvement over the timings reported by Lauder by a factor of at least $67,403$.

Part III

Implementing the underlying arithmetic

Chapter 10

Unramified p-adic arithmetic

We present an overview of arithmetic operations in $\mathbf{Q}_p$ and unramified extensions. Much of the material is well-known in either the context of p-adic arithmetic itself, see e.g. Vercauteren [17, §12] or integer arithmetic, see e.g. the survey by Bernstein [6]. The original contribution consists of the adaptation of an algorithm for computing the exponential function by Brent [10] to the case of p-adic arithmetic, and the derivation of a related algorithm for computing the logarithm function. Moreover, this work is accompanied by an implementation in the open-source computational software FLINT [33].

The sections on the p-adic exponential and logarithm functions are joint work with Fredrik Johansson, RISC, Austria.

10.1 Data format

We represent a non-zero p-adic number $x \in \mathbf{Q}_p$ in the form

$$x = p^v u$$

where $u, v \in \mathbf{Z}$ and $p \nmid u$. When working to precision N, it is sometimes useful to assume that $u \in [0, p^{N-v})$.

When working in the unique unramified extension of $\mathbf{Q}_p$ of degree d, which we shall

refer to as $\mathbf{Q}_q$ where $q = p^d$ with ring of integers $\mathbf{Z}_q$, we choose to represent this as

$$\mathbf{Q}_q \cong \mathbf{Q}_p[X]/(f(X))$$

where $f(x) \bmod p$ is an irreducible and separable polynomial. We store $f(X)$ as a sparse polynomial and assume that its coefficients are reduced modulo p. If $f(X)$ has at most a constant number of non-zero terms, the reduction of a degree n polynomial modulo $f(X)$ can be carried out in $\mathcal{O}(nd)$ additions and multiplications in $\mathbf{Q}_p$. With this set-up, a non-zero element g in $\mathbf{Q}_q$ can be represented as

$$g(X) = p^v h(X)$$

where $v \in \mathbf{Z}$ and $h(X) \in \mathbf{Z}[X]$ is such that at least one coefficient of $h(X)$ is a p-adic unit and $\deg(h) < d$. As before, it can be useful to assume that all coefficients of $h(X)$ are reduced modulo p^{N-v}.

Remark 10.1.1. We could aim for a genuine base p representation, explicitly showing the p-adic digits in the form

$$x = p^v(a_0 + a_1 p + a_2 p^2 + \cdots)$$

with $a_i \in [0, p)$ for all i, or perhaps

$$x = p^v(a_0 + a_1 p^k + a_2 p^{2k} + \cdots)$$

with $a_i \in [0, p^k)$ where k is least such that p^k fits into a machine word. This data format would be beneficial to many algorithms. But in practice, given the high quality and architecture specific details of implementations for arbitrary precision integer arithmetic in base 2, this approach does not seem favourable.

10.2 Review of complexity results and related definitions

We let $\mathcal{O}(M(N))$ denote the complexity of multiplying two N-bit integers. For example, the algorithm of Schönhage–Strassen [57] allows us to take $M(N) = N \log N \log\log N$. The same complexity can also be achieved for the division with remainder of a $2N$-bit integer by an N-bit integer. As elements of $\mathbf{Z}/(p^N)$ have bit size $N \log p$, multiplication of two numbers in $[0, p^N)$, with or without a subsequent reduction modulo p^N, has complexity $\mathcal{O}(M(N \log p))$.

Moreover, we let $\mathcal{O}(\mu(p, d, N))$ denote the complexity of multiplication in $\mathbf{Z}_q/(p^N)$. This operation involves one multiplication of two degree $d-1$ polynomials in $\mathbf{Z}/(p^N)$ and one reduction of a degree $2d-2$ polynomial in $\big(\mathbf{Z}/(p^N)[X]\big)/(f(X))$. A fast Fourier transform based multiplication routine as described in [6, §4] requires $\mathcal{O}(d \log d \log\log d)$ operations in $\mathbf{Z}/(p^N)$ and hence has complexity $\mathcal{O}\big((d \log d \log\log d) M(N \log p)\big)$. The reduction modulo $f(X)$ also requires $\mathcal{O}(d \log d \log\log d)$ ring operations in $\mathbf{Z}/(p^N)$ in general but only $\mathcal{O}(d)$ when $f(X)$ is *sparse*, by which we mean that the number of non-zero coefficients of $f(X)$ is bounded independently of the parameters p, d, and N. In either case, for later reference we can set $\mu = \mu(p, d, N) = (d \log d \log\log d) M(N \log p)$.

In the remaining part of this chapter we make the assumption that the complexity of multiplying two N-bit integers satisfies $2M(N) \leq M(2N)$. In particular, this allows us to show that for $k+1$ multiplications of bit sizes $1, 2, \ldots, 2^k$,

$$M(1) + M(2) + \cdots + M(2^k) \leq \big(2^{-k} + 2^{-k+1} + \cdots + 1\big) M(2^k) \leq 2M(2^k).$$

This calculation shows that the complexity of Hensel lifting routines that we consider in the following sections is dominated by their last step at the greatest p-adic precision.

We also define ω to be the exponent of matrix multiplication, that is, such that the multiplication of two $n \times n$ matrices over a ring requires $\mathcal{O}(n^\omega)$ ring operations. The classical approach realises $\omega = 3$, the practical algorithm of Strassen [60] achieves $\omega < 2.81$, and the still essentially best theoretical result by Coppersmith–Winograd [18] shows that $\omega < 2.38$.

10.3 Hensel lifting

Theorem 10.3.1. Let $g \in \mathbf{Z}_q[X]$ be a polynomial whose leading coefficient is a unit and suppose that $x_0 \in \mathbf{Z}_q$ is such that

$$\mathrm{ord}_p(g(x_0)) \geq m+n, \quad \mathrm{ord}_p(g'(x_0)) \leq m$$

for some $0 \leq m < n$. Then there exists a unique $x \in \mathbf{Z}_q$ such that $g(x) = 0$ and $x \equiv x_0 \pmod{p^n}$.

Proof. We construct a sequence (x_k) such that

$$\begin{aligned} \mathrm{ord}_p(g(x_k)) &\geq 2^k(n-m)+2m, \\ \mathrm{ord}_p(g'(x_k)) &\leq m, \\ \mathrm{ord}_p(x_{k+1}-x_k) &\geq 2^k(n-m)+m, \end{aligned}$$

where the choice of x_{k+1} is unique given x_k, for $k \geq 0$. As the sequence (x_k) is Cauchy, the result then follows taking $x = \lim_{k\to\infty} x_k$ and using the continuity of g to establish $g(x) = 0$.

In order to satisfy the last condition, begin by writing $x_{k+1} = x_k + p^{2^k(n-m)+m}T$ and expand $g(x_{k+1})$ as a Taylor series about x_k,

$$\begin{aligned} g(x_{k+1}) &= \sum_{j=0}^{\deg(g)} \frac{g^{(j)}(x_k)}{j!} p^{(2^k(n-m)+m)j} T^j \\ &\equiv g(x_k) + g'(x_k)p^{2^k(n-m)+m}T \pmod{p^{2^{k+1}(n-m)+2m}} \end{aligned}$$

where the last line follows upon observing that $g^{(j)}(x_k)/j!$ is p-adically integral for all j. This forces the unique choice $x_{k+1} = x_k - g(x_k)/g'(x_k)$ modulo $p^{2^{k+1}(n-m)+m}$. □

Remark 10.3.2. In order to approximate the root of a polynomial to a desired precision N, it is preferable to choose the sequence of precisions

$$e_k = N, e_{k-1} = \lceil (e_k+m)/2 \rceil, \ldots, e_0 = \lceil (e_1+m)/2 \rceil \leq n$$

as this choice minimises the computational cost in every step and in the dominating last step, in particular.

Theorem 10.3.3. Let $g \in \mathbf{Z}_q[X]$ be a polynomial whose leading coefficient is a unit and suppose that $x_0 \in \mathbf{Z}_q$ is such that

$$\operatorname{ord}_p(g(x_0)) \geq m+n, \quad \operatorname{ord}_p(g'(x_0)) \leq m$$

for some $0 \leq m < n$. Moreover, let x be the unique root of g lifting x_0 and define the sequences

$$\begin{aligned} y_0 &= \big(g'(x_0)\big)^{-1}, \\ x_{k+1} &= x_k - g(x_k)y_k, \\ y_{k+1} &= y_k\big(2 - y_k g'(x_{k+1})\big), \end{aligned}$$

where x_k, y_k are computed to p-adic precision $2^k(n-m)+m$. Then x_k agrees with x modulo $p^{2^k(n-m)+m}$.

Remark 10.3.4. The above theorem leads to an algorithm running two Hensel lifting procedures in parallel, which is favourable to the approach suggested by the expression $x_{k+1} = x_k - g(x_k)/g'(x_k)$ in Theorem 10.3.1, which leads to a nested lifting routine to compute the inverse of $g'(x_k)$ from scratch at each step.

10.4 Addition, subtraction, negation

In order to add $x_0, x_1 \in \mathbf{Q}_q$ to precision N, we compute

$$x_0 + x_1 = p^{v_0}(u_0 + p^{v_1 - v_0}u_1) \tag{10.1}$$

where we assume $v_0 \leq v_1$. Note that the second factor may only be divisible by p if $v_0 = v_1$. Moreover, the second factor will have to be reduced modulo p^{N-v_0} using a division routine. We can improve on this if we add the assumption that the input is reduced modulo p^N, in which case we observe that there the reduction can be facilitated

by subtracting at most one multiple of p^{N-v_0}.

10.5 Multiplication

Given $x_0, x_1 \in \mathbf{Q}_q$, we can compute their product to precision N via

$$x_0 x_1 = p^{v_0+v_1} u_0 u_1, \tag{10.2}$$

reducing the product of units $u_0 u_1$ modulo $p^{N-v_0-v_1}$.

10.6 Inversion

Suppose that we want to compute the inverse of $x = p^v u \in \mathbf{Q}_q$ to precision N. Note that the inverse has valuation $-v$ and hence whenever $-v \geq N$ we return zero. Otherwise, our aim is to compute the inverse of $u \in \mathbf{Z}_q^\times$ to precision $N + v$.

In order to compute the inverse of $u \in \mathbf{Z}_q^\times$, we apply Hensel lifting on the polynomial $g(X) = 1 - uX$, leading to the iteration

$$\begin{aligned} x_{k+1} &= x_k - x_k(x_k u - 1), \\ &= x_k(2 - x_k u), \\ &= 2x_k - u x_k^2 \end{aligned} \tag{10.3}$$

starting from an approximation modulo p.

Note that we provide various expressions for x_{k+1} in equation (10.3). Roughly speaking, and ignoring the logarithmic dependence on p, the second expression consists of one 2^k-by-2^{k+1} bit product and one 2^k-by-(3×2^k) bit product whereas the third expression consists of one 2^k-by-2^k bit product and one 2^{k+1}-by-2^{k+1} bit product.

Inverses over finite fields can be computed efficiently by extended greatest common divisor algorithms, for example, using Euclid's algorithm or the asymptotically fast Half-GCD algorithm [61], which requires $\mathcal{O}\big(n(\log n)^2\big)$ operations in $\mathbf{F}_p$ for two input polynomials in $\mathbf{F}_p[X]$ of degree $\mathcal{O}(n)$. A constant factor improvement to the general method can be made by computing only one cofactor, that is, when computing s, t such

that $1 = \gcd(a, b) = sa + tb$ we in fact only require one of the two cofactors s, t.

Following this procedure, the complexity of inverting an element of $\mathbf{Z}_q^\times$ to precision N is given by $\mathcal{O}\big(d(\log d)^2 M(\log p) + \mu\big)$.

10.7 Inverse square root

The method of computing p-adic inverse square roots turns out to be interesting because it will be used as a precursor in computing p-adic square roots.

Given a non-zero $x = p^v u \in \mathbf{Q}_q$, the task is to compute one of its inverse square roots $p^{-v/2}u^{-1/2}$. Necessarily, we require that v is even and u a square in $\mathbf{Z}_q^\times$.

The computation then reduces to computing p-adic inverse square roots of $u \in \mathbf{Z}_q^\times$, which can be carried out using a Hensel lifting approach on $g(X) = 1 - uX^2$. This leads to the almost division-free iteration

$$\begin{aligned} x_{k+1} &= x_k + (1 - ux_k^2)/(2ux_k) \\ &= x_k + x_k(1 - ux_k^2)/2 \end{aligned} \tag{10.4}$$

In the notation of Theorem 10.3.1, this means that when $p > 2$ is an odd prime we can take $m = 0$, $n = 1$, but when $p = 2$ the division by 2 in the above formula forces us to account for the precision loss, taking $m = 1$, $n = 2$.

When $p = 2$, we obtain an initial approximation as follows. Let $s = (u \bmod 2)^{-1/2}$ modulo 2. If there exists a solution t to $t^2 + st \equiv (1/u - s^2)/4 \pmod 2$ then $x_0 = s + 2t$ is an approximate inverse square root of u to precision 2, otherwise u is not a square modulo 8 and hence not a square in $\mathbf{Z}_q^\times$. When $p > 2$ is an odd prime, we can take the inverse square root modulo p as our initial approximation. However, the topic of computing square roots in finite fields is outside the scope of this discussion. We refer the reader to standard results in finite field arithmetic, e.g. as presented in Bach–Shallit [4, §7].

We observe that this iteration does indeed not require computationally intensive p-adic inversions. When p is odd, either the numerator is divisible by 2 as an integer, or we can add an appropriate (odd) power of p to this, yielding an even integer representative

of the same value. The division by 2 is then simply a bitshift. When $p = 2$, the result on Hensel lifting in Theorem 10.3.1 guarantees that the numerator is always divisible by 2 and the loss of precision is accounted for in the sequence of precisions.

We now discuss the complexity of this operation. The initial step modulo p consists of an inversion and a square root in $\mathbf{F}_q$. The first of these can be achieved using the Half-GCD algorithm in $\mathcal{O}\big(d(\log d)^2\big)$ operations in $\mathbf{F}_p$. The second problem is easy in practice but much harder in theory. According to Bach–Shallit [4, §7, Exercise 34], it can be solved in $\mathcal{O}\big((d + \log p)(\log p)^2\big)$ bit operations using a probabilistic algorithm. The theoretically difficult part is the determination of a non-square element in $\mathbf{F}_q^\times$ when q is odd, but the algorithm can be made deterministic assuming the Extended Riemann Hypothesis. In terms of the dependency on N, the main contribution comes from the deterministic Hensel lifting procedure, which as in the case of inversion has complexity $\mathcal{O}(\mu)$. The overall complexity is given by the sum of the above three estimates since none dominates another as the parameters p, d, and N vary independently.

10.8 Square root

We observe that a non-zero $x = p^v u \in \mathbf{Q}_q$ has a square root if and only if v is even and $u \in \mathbf{Z}_q^\times$ is a square. Thus, for the remaining part of the discussion we may assume that $u \in \mathbf{Z}_q^\times$ is a square.

In order to compute $\sqrt{u}$ modulo p^N, we can use the previous method for computing the inverse square root to precision N and then multiply the result by u. This approach is favourable in pracice as the Hensel lifting iteration is more efficient for inverse square roots than for square roots. As noted by Karp and Markstein [39], this algorithm can be improved by replacing the final iteration in the Hensel lifting procedure for the inverse square root by

$$t = ux_k \pmod{p^{N'}} \tag{10.5}$$

$$x_{k+1} = t + x_k(u - t^2)/2 \pmod{p^N} \tag{10.6}$$

where N' is the precision in the penultimate step, and omitting the subsequent multi-

plication by u mentioned above.

The approach described above clearly has the same complexity as the algorithm for the inverse square root problem.

10.9 Teichmüller lift

Recall that $\mathbf{Q}_q$ contains exactly $(q-1)$ elements that are $(q-1)$th roots of unity, one above each non-zero element of the residue field $\mathbf{F}_q$. The Teichmüller lift of an element $u \in \mathbf{F}_q^\times$ is defined to be the unique lift to a $(q-1)$th root of unity in $\mathbf{Q}_q$.

In order to compute the Teichmüller lift of $u \in \mathbf{F}_q^\times$, we apply Hensel lifting on $g(X) = X^q - X$, starting with $x_0 = u$ and noting that $\operatorname{ord}_p(g'(x_0)) = 0$.

As a noticeable practical improvement, we observe that the denominators $g'(x_k) = qx_k^{q-1} - 1$ converge to $q-1$, which is defined over $\mathbf{Z}_p$ already, in a way such that $\operatorname{ord}_p\big(g'(x_k)-(q-1)\big)$ is non-decreasing. Thus, we can replace the Hensel lifting iteration by

$$x_{k+1} = x_k - (q-1)^{-1}(x_k^q - x_k), \tag{10.7}$$

also noting that $(q-1)^{-1}$ only has to be computed to the precision of the previous step.

The complexity of this approach is dominated by the evaluations of $g(x_k) = x_k^q - x_k$ to the precisions $2, \ldots, \lceil N/2 \rceil, N$ during the Hensel lifting procedure. Using binary exponentiation, computing the power x_k^q requires $\mathcal{O}(\log q)$ operations in $\mathbf{Z}_q$ to the appropriate p-adic precision at each step. Thus, the complexity is given by $\mathcal{O}\big((\log q)\mu\big)$.

10.10 Frobenius

Let $\Sigma \in \operatorname{Gal}(\mathbf{Q}_q/\mathbf{Q}_p) \cong \operatorname{Gal}(\mathbf{F}_q/\mathbf{F}_p)$ be the lift of $\sigma\colon \mathbf{F}_q \mapsto \mathbf{F}_q, x \mapsto x^p$. For any $x \in \mathbf{Q}_q$ and $k \in \mathbf{Z}$, we aim to compute $\Sigma^k x$ modulo p^N, noting that Σ has order d. As such, we may assume that k is reduced modulo d, and in fact that $0 < k < d$ as Σ^0 is the identity map. Moreover, as Σ is a $\mathbf{Q}_p$-linear map, we may assume that $x = \sum_{i=0}^{d-1} a_i X^i$ is a unit in $\mathbf{Z}_q$.

First, we can compute $\Sigma^k X$ using Hensel lifting on $f(X)$, starting from $x_0 = X^{p^k}$ in $\mathbf{F}_p[X]/(f(X))$. We note that there is no precision loss during the Hensel lifting

procedure; in the notation of Theorem 10.3.1, we can take $m = 0$, $n = 1$. This is because $f \bmod p$ is separable and hence $f'(x_0) \bmod p$ is non-zero. At each step, the Hensel lifting routine involves the computations of $f(x_k)$ and $f'(x_k)$, which are polynomial compositions modulo $f(X)$ and a power of p. We discuss the general case of modular compositions further below, but note here that when $f(X)$ is sparse the naive approach to composition requires only $\mathcal{O}\big((\log d)\mu\big)$.

Finally, we can compute $\Sigma^k x$ via

$$\Sigma^k x = \Sigma^k\Big(\sum_{i=0}^{d-1} a_i X^i\Big) = \sum_{i=0}^{d-1} a_i (\Sigma^k X)^i, \tag{10.8}$$

which entails a polynomial composition modulo $f(X)$ and p^N.

In a first approach, we might use Horner's method to carry out the composition, which requires about d multiplications in $\mathbf{Z}_q$, yielding a complexity of $\mathcal{O}(d\mu)$.

However, we can use the rectangular splitting approach of Paterson–Stockmeyer [53] instead, based on the expression

$$\Sigma^k x = \sum_{i=0}^{\lfloor d/B \rfloor - 1} \Big(\sum_{j=0}^{B-1} a_{Bi+j} (\Sigma^k X)^j\Big) (\Sigma^k X)^{Bi}, \tag{10.9}$$

where $B = \lfloor \sqrt{d} \rfloor$ and we define $a_{Bi+j} = 0$ whenever $Bi + j \geq d$, note that in the last iteration the inner sum generally has fewer than B terms. From an algorithmic point of view, it is important that this allows us to precompute $\Sigma^k(X)^i$ for $i = 0, \ldots, B$. Thus, this approach only requires about $2\sqrt{d}$ multiplications in $\mathbf{Z}_q$ but additional space for about $d^{3/2}$ elements of $\mathbf{Z}/(p^N)$. As there are another d scalar multiplications of elements of $\mathbf{Z}_q$ by elements of $\mathbf{Z}_p$, the complexity of this polynomial composition is given by $\mathcal{O}\big(\sqrt{d}\mu + d^2 M(N \log p)\big)$, which is $\mathcal{O}\big(d^2 M(N \log p)\big)$. Thus, the asymptotic improvement in d when compared to Horner's method only consist of removed logarithmic factors, however, in practive the smaller number of $\mathbf{Z}_q$-multiplications is important.

This can be improved asymptotically by employing the modular composition algorithm of Brent–Kung [11], which requires only $\mathcal{O}(n^{(\omega+1)/2})$ ring operations when all three input polynomials have degrees $\mathcal{O}(n)$, where ω is the exponent of matrix multipli-

cation. Here, the ring in question in $\mathbf{Z}/(p^N)$ and the polynomials have degree at most d, hence we obtain the bit complexity estimate $\mathcal{O}\big(d^{(\omega+1)/2}M(N\log p)\big)$.

We now discuss the overall complexity of the computation of $\Sigma^k x$. Using binary exponentiation, computing $x_0 = X^{p^k}$ requires about $k\log p$ products in $\mathbf{F}_p[X]/(f(X))$ and hence has complexity $\mathcal{O}\big((d\log p)\mu(p,d,1)\big)$. The Hensel lifting routine is dominated by its last step, which involves evaluating $f(x_k)$ and $f'(x_k)$ in $\mathbf{Z}_q$ to precision N. This in turn consists of two modular compositions. As described above, the final step is another modular composition. Thus, the overall complexity is given by $\mathcal{O}\big((d\log p)\mu(p,d,1) + d^{(\omega+1)/2}M(N\log p)\big)$.

10.11 Trace

The image of $x \in \mathbf{Q}_q$ under the trace function $\mathrm{Tr}_{\mathbf{Q}_q/\mathbf{Q}_p}\colon \mathbf{Q}_q \to \mathbf{Q}_p$ is defined as the sum of the Galois conjugates of x. Equivalently, it is defined as the trace of the $\mathbf{Q}_p$-linear map on $\mathbf{Q}_q$ given by multiplication by x. As the trace map itself is $\mathbf{Q}_p$-linear, we may now assume that $x \in \mathbf{Z}_q^\times$.

There are two efficient ways to compute the trace:

First, we can compute this from the definition of $\mathrm{Tr}(x)$ as the trace of the multiplication by x map by reducing xX^i modulo $f(X)$ and forming the sum of the X^i-coordinates, for $i = 0, \ldots, d-1$.

Recalling that the reduction of a degree $2d-2$ polynomial modulo $f(X)$ in $\mathbf{Z}/(p^N)$ requires time $\mathcal{O}(\mu)$, it is clear that this approach takes time $\mathcal{O}(d\mu)$. When we assume that $f(X)$ is sparse, this decreases to $\mathcal{O}\big(d^2M(N\log p)\big)$.

Alternatively, we can observe that on writing $x = \sum_{i=0}^{d-1} a_i X^i$ we have $\mathrm{Tr}(x) = \sum_{i=0}^{d-1} a_i \mathrm{Tr}(X^i)$ and then use the Newton–Girard formulae,

$$\mathrm{Tr}(X^i) + \sum_{j=1}^{i-1} \mathrm{Tr}(X^{i-j}) f_{d-j} + i f_{d-i} = 0 \pmod{p^N}, \tag{10.10}$$

for $i = 1, \ldots, d-1$, where $f(X) = \sum_{i=0}^{d-1} f_i X^i$. We also note that $\mathrm{Tr}(X^0) = \mathrm{Tr}(I) = d$. This approach visibly requires time $\mathcal{O}\big(d^2M(N\log p)\big)$ in general. When $f(X)$ is sparse, this further decreases to $\mathcal{O}(dM(N\log p))$ as the sum in equation (10.10) has a bounded

number of non-zero terms.

10.12 Norm

The image of $x \in \mathbf{Q}_q$ under the norm function $\mathrm{N}_{\mathbf{Q}_q/\mathbf{Q}_p} \colon \mathbf{Q}_q \to \mathbf{Q}_p$ is defined as the product of the Galois conjugates of x. Thus, for $x \in \mathbf{Q}_q$ and $\lambda \in \mathbf{Q}_p$, we have that $\mathrm{N}(\lambda x) = \lambda^d \mathrm{N}(x)$ and we may hence assume that $x \in \mathbf{Z}_q^\times$. Since $\mathrm{Gal}(\mathbf{Q}_q/\mathbf{Q}_p) = \langle \Sigma \rangle$ is cyclic of order d, we find that, letting $a(X) \in \mathbf{Z}_p[X]$ denote the same polynomial as $x = \sum_{i=0}^{d-1} a_i X^i$,

$$\begin{aligned} \mathrm{N}_{\mathbf{Q}_q/\mathbf{Q}_p}(x) &= \prod_{i=0}^{d-1} \Sigma^i(x) \\ &= \prod_{i=0}^{d-1} a(\Sigma^i(X)) \\ &= \ell(f)^{-\deg(a)} \operatorname{Res}(f(X), a(X)). \end{aligned} \tag{10.11}$$

where $\ell(f)$ denotes the leading coefficient of $f(X)$.

The resultant of $f(X)$ and $a(X)$ can be defined as the determinant of the Sylvester matrix [16, Lemma 3.3.4], which is a square matrix over $\mathbf{Z}/(p^N)$ with $\deg(f) + \deg(a) \leq 2d-1$ columns. In order to avoid precision loss, this can be computed with the division-free determinant algorithm of Kaltofen [38], which requires $\mathcal{O}(d^{\omega/2+2} \log d \log\log d)$ ring operations in $\mathbf{Z}/(p^N)$ where ω is the exponent for matrix multiplication. Thus, the bit complexity of this computation can be given by $\mathcal{O}\big((d^{\omega/2+2} \log d \log\log d) M(N \log p)\big)$.

We note, however, that we have only implemented the simpler division-free determinant algorithm of Seifullin [59, Algorithm 4.2], which requires $\mathcal{O}(n^4)$ ring operations to compute the determinant of an $n \times n$ matrix. This leads to a bit complexity estimate for the norm computation of $\mathcal{O}\big(d^4 M(N \log p)\big)$.

Whenever $x \in \mathbf{Z}_q^\times$ satisfies $\mathrm{ord}_p(x-1) > (p-1)^{-1}$ we can typically improve on this significantly, computing the norm via

$$\mathrm{N}_{\mathbf{Q}_q/\mathbf{Q}_p}(x) = \exp\big(\mathrm{Tr}_{\mathbf{Q}_q/\mathbf{Q}_p}(\log(x))\big). \tag{10.12}$$

We omit a detailed analysis of the complexity of this approach, but note that together with the asymptotically fast algorithm for the logarithm function it behaves quasi-linearly in d and N when $f(X)$ is sparse.

10.13 Exponential

10.13.1 Definition

For $x \in \mathbf{Z}_q$ with $\mathrm{ord}_p(x) \geq 2$ or $\mathrm{ord}_p(x) \geq 1$ as $p = 2$ or $p > 2$, respectively, the p-adic exponential functions is defined via

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}. \tag{10.13}$$

In order to compute $\exp(x)$ modulo p^N, using that for positive integers z, we have

$$\mathrm{ord}_p(z!) = \frac{z - s_p(z)}{p-1} \leq \frac{z}{p-1} \tag{10.14}$$

where $s_p(-)$ denote the sum of p-adic digits, we are led to compute the truncated series

$$\exp(x) = \sum_{i=0}^{n-1} \frac{x^i}{i!} \tag{10.15}$$

where $n = \lceil \big((p-1)N - 1\big) / \big((p-1)v - 1\big) \rceil$ with $v = \mathrm{ord}_p(x)$.

Note that we require $\mathcal{O}(N)$ terms of the sum, and that when computing these iteratively for $i = 0, \ldots, n-1$ each update step of the summand amounts to multiplication by x/i, which consists of an inversion in $\mathbf{Z}_p$, a scalar multiplication in $\mathbf{Z}_q$, and a product in $\mathbf{Z}_q$. Altogether, this yields the complexity estimate $\mathcal{O}(N\mu)$, which is quasi-quadratic in N.

10.13.2 Rectangular splitting

The rectangular splitting algorithm due to Paterson and Stockmeyer [53] applied to the case of the p-adic exponential evaluates the truncated series (10.15) using the expression

$$\begin{aligned}\exp(x) &= \sum_{j=0}^{\lceil n/B\rceil-1}\left(\sum_{i=0}^{B-1}\frac{x^i}{(Bj+i)!}\right)x^{Bj}\\ &= \sum_{j=0}^{\lceil n/B\rceil-1}\frac{1}{(B(j+1)-1)!}\left(\sum_{i=0}^{B-1}\frac{(B(j+1)-1)!}{(Bj+i)!}x^i\right)x^{Bj}\\ &= \frac{1}{(n-1)!}\sum_{j=0}^{\lceil n/B\rceil-1}\frac{(n-1)!}{(B(j+1)-1)!}\left(\sum_{i=0}^{B-1}\frac{(B(j+1)-1)!}{(Bj+i)!}x^i\right)x^{Bj}\end{aligned} \tag{10.16}$$

where $B = \lfloor\sqrt{n}\rfloor$ and we omit the detail that in the last iteration in inner sum generally has fewer than B terms.

There are three noticeable advantages to this approach. Firstly, it reduces the number of p-adic inversions as now the division in the inner sum is an exact integer division. Secondly, the scalar factors in the inner sum $(Bj+i+1)\cdots(Bj+B-1)$ are only of size $\mathcal{O}(\sqrt{n}\log n)$. Thirdly, in the case when $q > p$, this approach reduces the number of multiplication in $\mathbf{Q}_q$.

We present the details in Algorithm 10.1. In order to conclude that this approach does not lead to an asymptotic improvement in N, we observe that the inner loop contains a scalar product where the scalar has bit size $\mathcal{O}(\sqrt{N}\log N)$ and the vector has d coefficients each of bit size $\mathcal{O}(N\log p)$. As this statement is executed about N times, the time complexity is still quasi-quadratic in N.

10.13.3 Binary splitting

In this section, we describe Brent's binary splitting, or bit-burst algorithm [10] for computing the exponential function, adapted to the p-adic case. In order to simplify the exposition, we restrict ourselves to the case of $\mathbf{Q}_p$ but note that it is possible to obtain the same asymptotic results in terms of their dependency on N in the case of $\mathbf{Q}_q$.

Algorithm 10.1 Computing the exponential via rectangular splitting

Input: Integer $x \in [1, p^N)$ with $v = \mathrm{ord}_p(x) > (p-1)^{-1}$,
Output: $\exp(x) \bmod p^N$.

```
procedure ExpRectangular(x, p, N)
    n ← ⌈((p − 1)N − 1)/((p − 1)v − 1)⌉
    if n ≤ 3 then return Σ_{i=0}^{n−1} x^i/i! mod p^N
    else
        k ← ⌊(n − 2)/(p − 1)⌋
        B ← ⌊√n⌋
        for i = 0 to B do
            x^(i) ← x^i mod p^{N+k}
        S ← 0
        f ← 1
        for i = ⌈n/B⌉ − 1 to 0 by −1 do
            ℓ ← iB
            s ← 0
            c ← 1
            for h = min{n − 1, ℓ + B − 1} to ℓ by −1 do
                s ← s + c x^(h−ℓ)
                if h ≠ 0 then
                    c ← hc
            S ← sf + S x^(B) mod p^{N+k}
            f ← cf
        return f^{−1} S mod p^N
```

Exponential, exact rational

Algorithm 10.2 computes the sum

$$(a-1)!x^{1-a}\sum_{i=a}^{b-1}\frac{x^i}{i!} \tag{10.17}$$

as an exact rational number. This expression is chosen in such a way as to allow for a recursive computation of the exponential series. Visibly, we can invoke this routine with $a = 1$, $b = n$ in order to compute $\sum_{i=1}^{n-1} x^i/i!$.

Algorithm 10.2 Computing the exponential as an exact rational

Input: Integer x, integers $1 \leq a < b$.
Output: $P = x^{b-a}$, $Q = (b-1)!/(a-1)!$, $T = (b-1)!x^{1-a}\sum_{i=a}^{b-1} x^i/i!$.

```
procedure ExpBSplit(P, Q, T, x, a, b)
    if b − a = 1 then
        (P, Q, T) ← (x, a, x)
    else if b − a = 2 then
        (P, Q, T) ← (x^2, a(a + 1), x(a + 1) + x^2)
    else
        m ← ⌊(a + b)/2⌋
        ExpBSplit(P_0, Q_0, T_0, x, a, m)
        ExpBSplit(P_1, Q_1, T_1, x, m, b)
        P ← P_0 P_1
        Q ← Q_0 Q_1
        T ← T_0 Q_1 + P_0 T_1
```

We now provide a sketch of the complexity analysis, assuming for simplicity that $a = 1$ and $b = n = 2^k + 1$. Let us label the various levels in the recursive procedure $0, 1, \ldots, k$ with 0 and k denoting the base cases and the top level, respectively. We begin by estimating the size of the return values P, Q, T at each level i. Initially, when $i = 0$, P, and T are of the same size as x and Q is at most n. At level i, P is equal to x^{2^i} and hence of size $2^i \log x$ bits. Q is the product of 2^i numbers at most n and hence of size at most $2^i \log n$. Finally, from the expression $T \leftarrow T_0Q_1 + P_0T_1$, we see that at level i the size of T is at most $2^i \max\{\log n, \log x\} + i$. We now count the number of operations at each level. At level 0 we only set P, Q, and T, and this is carried out 2^k times. At levels $i = 1, \ldots, k$, we carry out the following operations 2^{k-i} times: two products with factors of size at most $2^{i-1} \max\{\log n, \log x\} + (i-1)$ bits, one product with factors of size $2^{i-1} \log x$, and one product with factors of size at most $2^{i-1} \log n$ bits. Here, we

ignore the computation $m \leftarrow \lfloor (a+b)/2 \rfloor$ and the sum in $T \leftarrow T_0 Q_1 + P_0 T_1$. Thus, we can form the estimate,

$$\begin{aligned}\sum_{i=1}^{k} 2^{k-i} M\big(2^{i-1} \max\{\log n, \log x\} + (i-1)\big) &\leq \sum_{i=1}^{k} 2^{k-i} M\big(2^{i} \max\{\log n, \log x\}\big) \\ &\leq \sum_{i=1}^{k} 2^{k-i} 2^{-k+i} M\big(2^{k} \max\{\log n, \log x\}\big) \\ &\leq (\log n) M\big(n \max\{\log n, \log x\}\big)\end{aligned}$$

and conclude that the complexity is given by $\mathcal{O}\big((\log n) M(n \max\{\log n, \log x\})\big)$.

Main part

We now describe how to evaluate the truncated series $\sum_{i=0}^{n-1} x^i/i! \bmod p^N$ where we may assume that $0 < x < p^N$. First, write

$$x = \sum_{i=1}^{\lceil \log_p N \rceil} x_i \tag{10.18}$$

with $0 \leq x_i < p^{2^i}$ and $\operatorname{ord}_p(x_i) \geq 2^{i-1}$. Thus,

$$\exp(x) = \prod_{i=1}^{\lceil \log_p N \rceil} \exp(x_i) \tag{10.19}$$

and we observe that the computation of $\exp(x_i) \bmod p^N$ requires us to evaluate a sum of

$$n_i \leq \frac{(p-1)N - 1}{(p-1)2^{i-1} - 1} \leq \frac{N}{2^{i-2}} \tag{10.20}$$

terms. We carry out this computation using the routine from the previous section.

When using the complexity result from the previous section, we have to consider two cases because of the $\max\{-,-\}$ function present in the expression. In order to obtain an overall estimate for the $\lceil \log_p N \rceil$ computations of smaller exponentials we consider

the two cases of the $\max\{-,-\}$ function separately. Firstly, we have that

$$\begin{aligned}\sum_{i=1}^{\lceil \log_p N\rceil} \big(\log(2^{-i+2}N)\big)M\big(2^{-i+2}N\log(2^{-i+2}N)\big) &\leq \sum_{i=1}^{\lceil \log_p N\rceil} (\log N)M\big(2^{-i+2}N\log N\big)\\ &\leq \sum_{i=1}^{\lceil \log_p N\rceil} (\log N)2^{-i}M(4N\log N)\\ &\leq (\log N)M(4N\log N).\end{aligned}$$

Secondly,

$$\sum_{i=1}^{\lceil \log_p N\rceil} \big(\log(2^{-i+2}N)\big)M\big(2^{-i+2}N\log p^{2^i}\big) \leq \sum_{i=1}^{\lceil \log_p N\rceil} (\log N)M(4N\log p).$$

We also note that we can ignore the cost of forming the final product of the $\lceil \log_p N\rceil$ factors to precision N, which is $\mathcal{O}((\log_p N)M(N\log p)$. Therefore, we find that the complexity of the computation of $\exp(x)$ modulo p^N using the binary splitting routine is

$$\mathcal{O}\big((\log N)M(N\log N) + (\log N)(\log_p N)M(N\log p)\big). \tag{10.21}$$

10.14 Logarithm

10.14.1 Definition

For $x \in \mathbf{Z}_q$ with $\mathrm{ord}_p(1-x)$ at least 2 or 1 whenever $p = 2$ or $p > 2$, respectively, the p-adic logarithm function is defined as

$$\log(x) = -\sum_{i=1}^{\infty} \frac{(1-x)^i}{i}. \tag{10.22}$$

In order to compute $\log(x) \bmod p^N$, we need to consider all indices i such that $iv - \mathrm{ord}_p(i) < N$ where $v = \mathrm{ord}_p(1-x)$. With $c = N - \lfloor \log_p v\rfloor$ and setting $n = c + \lceil \log_p c\rceil + 1$, we have that $iv - \mathrm{ord}_p(i) \geq N$ for all $i \geq n$. Thus,

$$\log(x) = -\sum_{i=1}^{n-1} \frac{(1-x)^i}{i} \pmod{p^N}.$$

As in the case of the exponential, since the number of terms is $\mathcal{O}(N)$, this truncated series can be evaluated in time $\mathcal{O}(N\mu)$ using Horner's method.

10.14.2 Rectangular splitting

A significant constant factor improvement can be made by using a rectangular splitting algorithm, rewriting the sum as

$$\begin{aligned}\sum_{i=1}^{n-1}\frac{y^i}{i} &= \sum_{j=0}^{\lceil (n-1)/B\rceil -1}\Big(\sum_{i=1}^{B}\frac{y^i}{Bj+i}\Big)y^{Bj}\\ &= \sum_{j=0}^{\lceil (n-1)/B\rceil -1}(Bj+1)_B^{-1}\Big(\sum_{i=1}^{B}\frac{(Bj+1)_B}{Bj+i}y^i\Big)y^{Bj}\end{aligned} \tag{10.23}$$

where $B=\lfloor\sqrt{n}\rfloor$ and for any integer z we let $z_B=\prod_{j=0}^{B-1}(z+j)$. As the inner sum now features an exact integer division, this approach reduces the number of p-adic inversions to about $\sqrt{n}$.

We include a detailed version of this as Algorithm 10.3.

Algorithm 10.3 Computing the logarithm via rectangular splitting

Input: Integer $y\in[1,p^N)$,
Output: $\sum_{i=1}^{n} y^i/i \bmod p^N$.

```
procedure LogRectangular(y, n, p, N)
    if n ≤ 3 then return Σ_{i=1}^{n} y^i/i mod p^N
    else
        k ← ⌊log_p n⌋
        B ← ⌊√n⌋
        for i = 0 to B do
            y^(i) ← y^i mod p^(N+k)
        z ← 0
        for i = ⌈n/B⌉ − 1 to 0 by −1 do
            h ← min{B, n − iB}
            f ← Π_{j=1}^{h} (iB + j)
            c ← 0
            for j = 1 to h do
                c ← c + (f/(iB + j)) y^(j)
            c ← p^k (f^{-1} mod p^N) c
            z ← c + z y^(B) mod p^(N+k)
        return p^{-k} z
```

10.14.3 Binary splitting

Logarithm, exact rational

Algorithm 10.4 computes the sum

$$(a-1)!y^{1-a}\sum_{i=a}^{b-1}\frac{y^i}{i} \tag{10.24}$$

as the rational number, which allows us to recursively compute the logarithm series when invoking this routine with $a = 1$, $b = n$. We provide a precise algorithmic formulation of this idea in Algorithm 10.4 and observe that ananalysis analogous to the case of the p-adic exponential shows that the complexity of this routine is $\mathcal{O}\big((\log n)M(n\max\{\log n, \log y\})\big)$.

Algorithm 10.4 Computing the logarithm as an exact rational

Input: Integer y, integers $1 \le a < b$.
Output: $P = y^{b-a}$, $Q = (b-1)!/(a-1)!$, $T = (b-1)!y^{1-a}\sum_{i=a}^{b-1} y^i/i$.

```
procedure LogBSplit(P, Q, T, y, a, b)
    if b − a = 1 then
        (P, Q, T) ← (y, a, y)
    else if b − a = 2 then
        (P, Q, T) ← (y^2, a(a + 1), y(a + 1) + y^2 a)
    else
        m ← ⌊(a + b)/2⌋
        LogBSplit(P_0, Q_0, T_0, y, a, m)
        LogBSplit(P_1, Q_1, T_1, y, m, b)
        P ← P_0 P_1
        Q ← Q_0 Q_1
        T ← T_0 Q_1 + T_1 P_0 Q_0
```

Main part

We can employ the same idea as in the case of the exponential function, relying on the expression

$$\log(1+x+y) = \log(1+x) + \log\Big(1+\frac{y}{1+x}\Big). \tag{10.25}$$

More precisely, let us define the sequences $(r_i)_{i\ge1}$ and $(t_i)_{i\ge0}$ by $t_0 = x$ and

$$r_i = t_{i-1} \bmod p^{2^i}, \tag{10.26}$$

$$t_i = \frac{t_{i-1} - r_i}{1 + r_i} \bmod p^N. \tag{10.27}$$

Then

$$\log(1+x) = \sum_{i=1}^{\lceil \log_p N \rceil} \log(1 + r_i) \pmod{p^N}. \tag{10.28}$$

The complexity analysis for this approach is analogous to that for the binary splitting approach for the exponential.

10.15 Benchmarks

In this section, we compare the practical performance of our implementation of various p-adic routines in FLINT [33] to those available in MAGMA [9]. Specifically, we build FLINT with MPIR 2.4.0 and use MAGMA 2.18-6. We use the machine `wolverine.maths.ox.ac.uk`, a multi-core 64-bit machine featuring 24 Intel Xeon X5670 CPUs at 2.93 GHz, although we only use a single core for each computation.

For our computations over the base field $\mathbf{Q}_p$, we choose the prime $p = 17$ and the precisions $N = 2^i$ for $i = 0, \ldots, 19$. In order to avoid problems associated with different random number sequences on the test systems, we choose explicit elements in $\mathbf{Z}/(p^N)$ as input to the various arithmetic functions.

In the case of $\mathbf{Q}_q$, we consider the same prime and range of precisions where possible, although some data points are missing because of excessive time or memory requirements. We focus on the dependency on the precision N, fixing the extension degree 97. The only exception is the computation of the norm, which we present for both $d = 5$ and $d = 97$ in order to highlight the usefulness of our two separate algorithms.

The timings are collected in Figures 10.1, 10.2 and 10.3. To summarise the results, it appears that our routines are competitive with MAGMA, and often they are a constant factor faster. In the case of $\mathbf{Q}_p$, the two noticeable exceptions are the exponential and logarithm functions, where we can observe that the binary splitting versions are asymptotically faster. In the case of $\mathbf{Q}_q$, our routines are competitive again. Considering the square root function, the graph shows that for values of N up to about 10^3, the runtime is completely dominated by the initial square root computation over $\mathbf{F}_q$. The

norm computation for $d = 5$ shows that the resultant approach matches the MAGMA routine, both being a factor of about ten faster than the analytic approach. However, when $d = 97$ the contribution of d^4 to the complexity of the resultant approach is apparent. We observe that our implementation of the analytic approach is fastest, and that both of our routines display a quasi-linear dependence on N while the MAGMA routine appears to be quasi-quadratic. Regarding the exponential function, while our implementation of the binary splitting algorithm is a constant factor faster than the MAGMA routine, both display a quasi-linear dependence on N. Finally, our binary splitting implementation for the logarithm function displays a quasi-linear dependence on N while both the rectangular splitting version and the MAGMA routine show a worse asymptotic runtime.

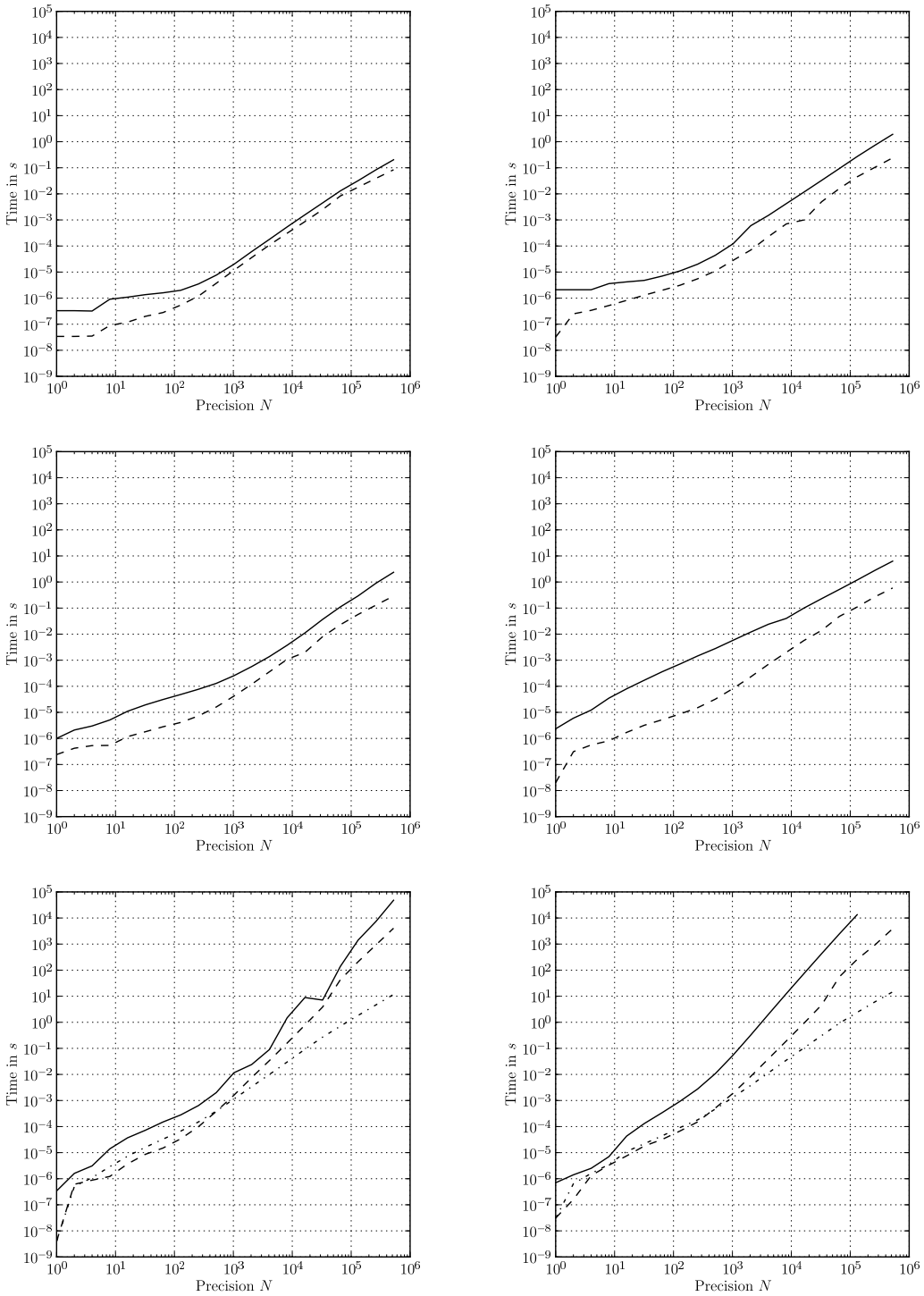


Figure 10.1: From left to right, top to bottom, we compute $3^{3N} \times 5^{3N}$, $\left(3^{3N}\right)^{-1}$, $\sqrt{3^{6N}}$, the Teichmüller lift of 3, $\exp\left(17 \times 3^{3N}\right)$, and $\log\left(1 + 17 \times 3^{3N}\right)$ modulo 17^N. The solid lines represent the routines in Magma, the dashed and dotted lines the routines in FLINT.

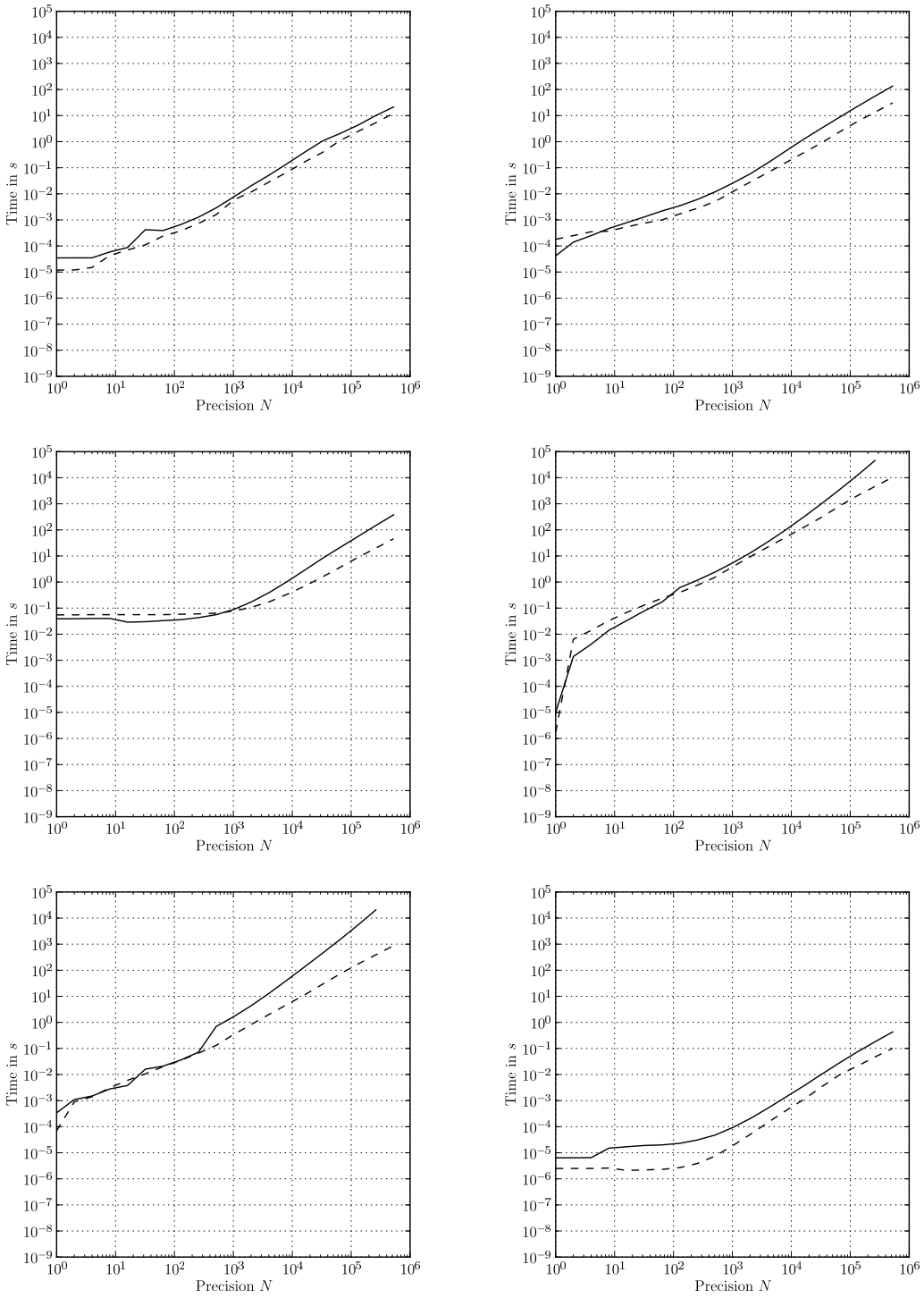


Figure 10.2: From left to right, top to bottom, we compute $a \times b$, a^{-1}, $\sqrt{a^2}$, the Teichmüller lift of c, the Frobenius image of a, and the trace of a in $\mathbf{Z}_{17^{97}}$ modulo 17^N, where $a = \sum(3+i)^{3N}X^i$, $b = \sum(5+2i)^{3N}X^i$, and $c = \sum(3+i)X^i \bmod p$. The solid lines represent the routines in Magma, the dashed and dotted lines the routines in FLINT.

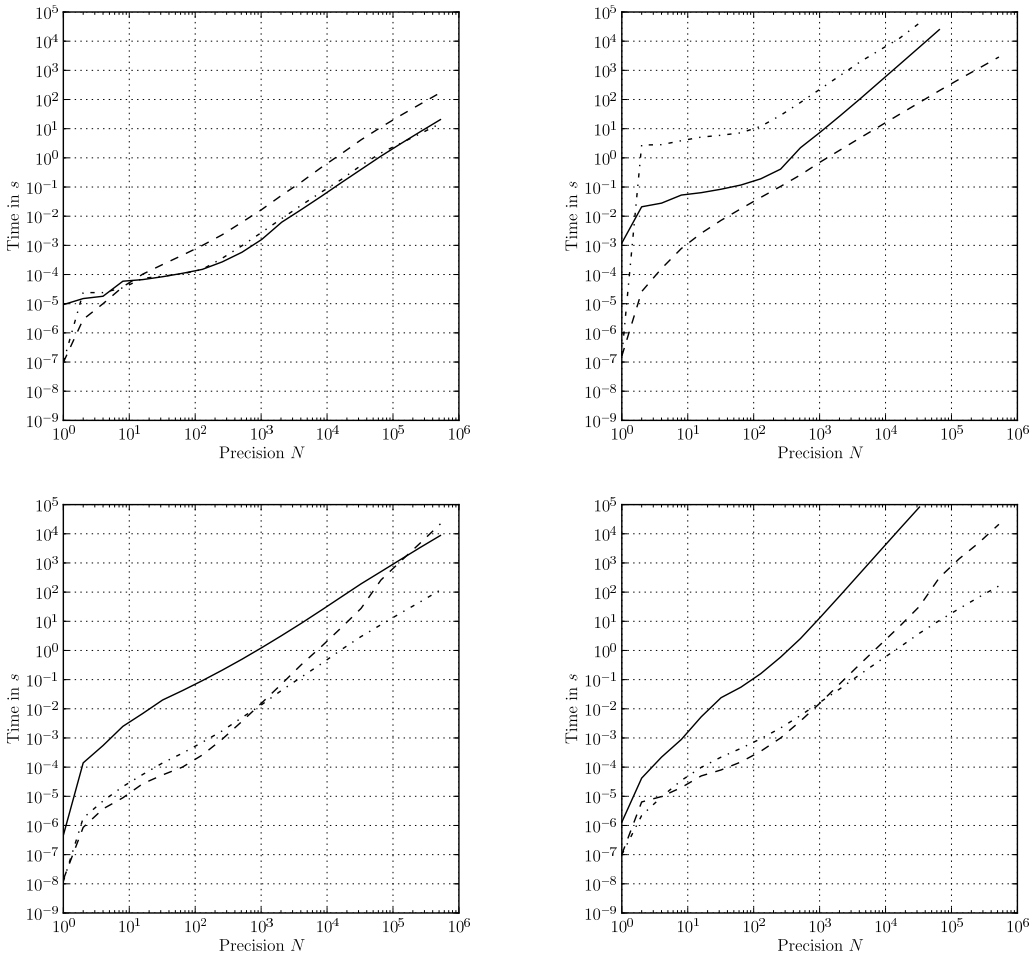


Figure 10.3: From left to right, top to bottom, we compute the norm of a in $\mathbf{Q}_{17^5}$ and $\mathbf{Q}_{17^{97}}$, and the exponential of $17a$ and the logarithm of $1 + 17a$ in $\mathbf{Q}_{17^{97}}$, where $a = \sum(3+i)^{3N}X^i$. The solid lines represent the routines in Magma, the dashed and dotted lines the routines in FLINT.

Bibliography

[1] T. Abott, K. S. Kedlaya, and D. Roe, *Bounding Picard numbers of surfaces using p-adic cohomology*, Arithmetic, Geometry, and Coding Theory (AGCT-10), 2006.

[2] A. Atkin, *The number of points on an elliptic curve modulo a prime, draft*, 1988.

[3] ———, *The number of points on an elliptic curve modulo a prime (ii). draft*, 1992.

[4] E. Bach and J. Shallit, *Algorithmic number theory, vol. 1: Efficient algorithms (Foundations of Computing)*, MIT Press, 1996.

[5] F. Baldassarri and B. Chiarellotto, *Algebraic versus rigid cohomology with logarithmic coefficients*, Barsotti Symposium in Algebraic Geometry, Perspectives in Mathematics, vol. 15, 1995, pp. 11–50.

[6] D. J. Bernstein, *Fast multiplication and its applications*, Algorithmic number theory: lattices, number fields, curves and cryptography, Math. Sci. Res. Inst. Publ., vol. 44, Cambridge Univ. Press, Cambridge, 2008, pp. 325–384.

[7] P. Berthelot, *Géométrie rigide et cohomlogie des variétés algébriques de caractéristique p*, Mémoires de la S. M. F. **2** (1986), no. 23, 7–32.

[8] E. Bombieri, *On exponential sums in finite field II*, Invent. Math. **47** (1978), 29–39.

[9] W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993).

[10] R. P. Brent, *The complexity of multiple-precision arithmetic*, The Complexity of

Computational Problem Solving (R. Anderssen and R. Brent, eds.), University of Queensland Press, 1976, pp. 126–165.

[11] R. P. Brent and H. Kung, *Fast algorithms for manipulating formal power series*, J. ACM **25** (1978), no. 4, 581–595.

[12] W. Castryck, J. Denef, and F. Vercauteren, *Computing zeta functions of nondegenerate curves*, IMRP Int. Math. Res. Pap. (2006), Art. ID 72017, 57.

[13] W. Castryck, H. Hubrechts, and F. Vercauteren, *Computing zeta functions in families of $C_{a,b}$ curves using deformation*, Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 5011, Springer, Berlin, 2008, pp. 296–311.

[14] A. Chambert-Loir, *Compter (rapidement) le nombre de solutions d'équations dans les corps finis*, Astérisque (2008), no. 317, Exp. No. 968, vii, 39–90, Séminaire Bourbaki. Vol. 2006/2007.

[15] D. Clark, *A note on the p-adic convergence of solutions of linear differential equations*, Proc. Amer. Math. Soc. **17** (1966), 262–269.

[16] H. Cohen, *A course in computational algebraic number theory*, Springer, 1993.

[17] H. Cohen and G. Frey (eds.), *Handbook of elliptic and hyperelliptic curve cryptography*, CRC Press, 2005.

[18] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symb. Comput. **9** (1990), no. 3, 251–280.

[19] T. H. Cormen, C. H. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, Massachusetts, 1990.

[20] J. Denef and F. Vercauteren, *An extension of Kedlaya's algorithm to hyperelliptic curves in characteristic 2*, J. Cryptology **19** (2006), no. 1, 1–25.

[21] I. Duff, *Algorithm 575: Permutations for a zero-free diagonal*, ACM Trans. Math. Softw. **7** (1981), no. 3, 387–390.

[22] ———, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Softw. **7** (1981), no. 3, 315–330.

[23] I. Duff and J. Reid, *Algorithm 529: Permutations to block triangular form [F1]*, ACM Trans. Math. Softw. **4** (1978), no. 2, 189–192.

[24] ———, *An implementation of Tarjan's algorithm for the block triangularization of a matrix*, ACM Trans. Math. Softw. **4** (1978), no. 2, 137–147.

[25] B. Dwork, *On the rationality of the zeta function of an algebraic variety*, Amer. J. Math. **82** (1960), 631–648.

[26] ———, *A deformation theory for the zeta function of a hypersurface*, Proceedings of the International Congress of Mathematicians, Stockholm, 1962, pp. 247–259.

[27] ———, *On the zeta function of a hypersurface I*, Publication Mathématiques de l'IHÉS **12** (1962), 5–68.

[28] ———, *On the zeta function of a hypersurface II*, The Annals of Mathematics, Second Series **80** (1964), no. 2, 227–299.

[29] N. D. Elkies, *Elliptic and modular curves over finite fields and related computational issues*, Computational perspectives on number theory (Chicago, IL, 1995), AMS/IP Stud. Adv. Math., vol. 7, Amer. Math. Soc., Providence, RI, 1998, pp. 21–76.

[30] R. Gerkmann, *Relative rigid cohomology and deformation of hypersurfaces*, Int. Math. Res. Pap. IMRP (2007), no. 1, 67.

[31] P. Griffiths and J. Harris, *Principles of algebraic geometry*, John Wiley & Sons, 1978.

[32] P. A. Griffiths, *On the periods of certain rational integrals. I, (resp. II)*, Ann. of Math. (2) **90** (1969), no. 3, 460–495 (resp. 496–541).

[33] W. Hart, S. Pancratz, A. Novocin, F. Johansson, and D. Harvey, *FLINT: Fast Library for Number Theory – Version 2.2*, June 2011, `http://www.libflint.org`.

[34] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics, vol. 52, Springer, 1977.

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

[36] H. Hubrechts, *Point counting in families of hyperelliptic curves in characteristic 2*, LMS J. Comput. Math. **10** (2007), 207–234.

[37] ———, *Point counting in families of hyperelliptic curves*, Found. Comput. Math. **8** (2008), no. 1, 137–169.

[38] E. Kaltofen, *On computing determinants of matrices without divisions*, Proc. ISSAC ´1992 (P. Wang, ed.), ACM Press, 1992, pp. 342–349.

[39] A. H. Karp and P. Markstein, *High-precision division and square root*, ACM Trans. Math. Software **23** (1997), no. 4, 561–589.

[40] N. Katz and T. Oda, *On the differentiation of de Rham cohomology classes with respect to parameters*, J. Math. Kyoto Univ. **8** (1968), no. 2, 199–213.

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

[42] ———, *Computing zeta functions via p-adic cohomology*, Algorithmic number theory, Lecture Notes in Comput. Sci., vol. 3076, Springer, Berlin, 2004, pp. 1–17.

[43] ———, *p-adic cohomology: from theory to practice*, p-adic Geometry: Lectures from the 2007 Arizona Winter School (D. Savitt and D. Thakur, eds.), University Lecture Series, vol. 45, American Mathematical Society, 2008, pp. 175–201.

[44] ———, *p-adic differential equations*, Cambridge Univ. Press, 2010.

[45] ———, *Effective p-adic cohomology for cyclic cubic threefolds*, Computational Algebraic and Analytic Geometry of Low-dimensional Varities, American Mathematical Society, 2011.

[46] K. S. Kedlaya and J. Tuitman, *Effective bounds for Frobenius structures*, Rendiconti del Seminario Matematico della Universitá di Padova (2012), 9.

[47] A. G. B. Lauder, *Counting solutions to equations in many variables over finite fields*, Foundations of Computational Mathematics **4** (2004), no. 3, 221–267.

[48] ———, *Deformation theory and the computation of zeta functions*, Proc. London Math. Soc **3** (2004), 565–602.

[49] ———, *A recursive method for computing zeta functions of varieties*, LMS J. Comp. Math. **9** (2006), 222–269.

[50] ———, *Degenerations and limit Frobenius structures in rigid cohomology*, LSM J. Comp. Math. **14** (2011), 1–33.

[51] F. Macaulay, *The algebraic theory of modular systems*, Cambridge Mathematical Library, Cambridge Univ. Press, 1994.

[52] B. Osserman, *The Weil conjectures*, The Princeton Companion to Mathematics (W. Gowers, ed.), Princeton Univ. Press, 2008, pp. 729–732.

[53] S. Paterson and L. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials*, SIAM J. Comput. **2** (1973), no. 1, 7.

[54] J. Pila, *Frobenius maps of abelian varieties and finding roots of unity in finite fields*, Math. Comp. **55** (1990), no. 192, 745–763.

[55] J. Reid, *Solution of linear systems of equations: Direct methods*, Sparse Matrix Techniques (Copenhagen) (V. Barker, ed.), vol. 572, Springer, 1976.

[56] T. Satoh, *The canonical lift of an ordinary elliptic curve over a finite field and its point counting*, J. Ramanujan Math. Soc. **15** (2000), no. 4, 247–270.

[57] A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292.

[58] R. Schoof, *Elliptic curves over finite fields and the computation of square roots mod p*, Math. Comp. **44** (1985), no. 170, 483–494.

[59] T. R. Seĭfullin, *Computation of the determinant, adjoint matrix, and characteristic polynomial without division*, Cybernet. Systems Anal. **38** (2002), no. 5, 650–672.

[60] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. **13** (1969), 354–356.

[61] K. Thull and C. K. Yap, *A unified approach to HGCD algorithms for polynomials and integers*, 1990.

[62] J. Tuitman, *Counting points in families of nondegenerate curves*, Ph.D. thesis, Katholieke Universiteit Leuven, 2011.

[63] J. von zur Gathen and J. Gerhard, *Modern computer algebra (2. ed.)*, Cambridge Univ. Press, 2003.

[64] G. Walker, *Computing zeta functions of varieties via fibration*, Ph.D. thesis, University of Oxford, 2009.

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