Determining the zeta function of curves via p-adic cohomology and deformation theory

Jérôme Grand'Maison

Master of Science

Department of Mathematics and Statistics

McGill University

Montréal, Québec, Canada

August 2007

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements of the above stated degree.

DEDICATION

Je désire dédier ma thèse à ma mère Rosette et à mon père Clermont.

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor Professor Henri Darmon for the continuous support and above all for the great liberty he gave me during this endeavor. That allowed me to spend three weeks in Wyoming and a semester in Toronto to learn about cryptography in mathematics, then to attend the Arizona Winter School where I first got involved with what will be described in Chapter 3, and finally to a Summer School in Germany on higher dimensional geometry over finite field, which is also obviously related to this thesis and to mathematical cryptography. I would also like to thank NSF, NATO, MITACS, etc. to have paid for this. On a more personal note I would like to deeply thank Ralf Gerkmann for his generous help in almost all areas of this thesis, Hugo Chapdeleine for his great improvised lectures, Christian Wuthrich for several enlightening discussions, Wouter Castryck for useful suggestions and the number theory group at McGill University for listening to my presentations. Finally I would like to thank FQRNT, NSERC and McGill University for providing me with money to live with.

ABRÉGÉ

Cette thèse comporte deux parties principales: les deux ont pour objet de calculer la fonction Zêta de certains types de courbes. La première partie contient des considérations théoriques et pratiques pour faire fonctionner l'algorithme de K. Kedlaya et certaines de ses généralisations. On résume les propriétés importantes de la cohomologie de Monsky et Washnitzer, les conjectures de Weil et on rassemble les pièces du casse-tête pour en arriver à un algorithme de P. Gaudry et N. Gürel pour calculer la fonction Zêta d'une courbe superelliptique. Dans la deuxième partie, nous présentons un algorithme pour calculer la fonction Zêta d'une courbe Cab en utilisant la théorie de la déformation. La stratégie générale, attribuée à A. Lauder, est d'étudier la variation de l'action de Frobenius le long d'une certaine famille de courbes. Cet algorithme, presque prêt à utiliser, généralise des travaux similaires de R. Gerkmann et H. Hubrechts.

ABSTRACT

This thesis has two main parts: both of which aim at computing the Zeta function of some curves. The first part contains the theoretical and practical considerations to make K. Kedlaya's algorithm and some of its generalizations work. We review important properties of the Monsky-Washnitzer cohomology, the Weil conjectures and piece the bits together in order to provide an algorithm that would compute the Zeta function of a superelliptic curve, due to P. Gaudry and N. Gürel. In the second part, we present an algorithm to compute the Zeta function of a Cab curve using deformation theory. The general strategy, due to A. Lauder, is to study the variation of the Frobenius action along a certain family. This almost fully practical algorithm generalizes similar work by R. Gerkmann and H. Hubrechts.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

ABRÉGÉ iv

ABSTRACT v

1 Introduction 1

1.1 Introduction to the rational-point counting problem 1

2 Determining the Zeta Function of Certain Types of Curves 4

2.1 Setting 4
2.2 The Monsky–Washnitzer cohomology 8
2.3 The Weil conjectures 13
2.4 Details of an implementation: theoretic 17
2.5 Details of an implementation: computational 19

3 The Zeta Function of Cab Curves via Deformation Theory 31

3.1 Setting 31
3.2 Details of the algorithm 35
3.3 Correctness of chosen bounds 55
3.4 The experiments 59

4 Conclusion 61

4.1 Possible improvements and future investigation 61

References 63

INDEX 65

KEY TO ABBREVIATIONS 66

CHAPTER 1
Introduction

1.1 Introduction to the rational-point counting problem

Counting rational points of algebraic varieties over finite fields is one of the most important question of computational algebraic geometry. For some classes of varieties, this problem has been studied extensively. The most notable case is that of elliptic curves, for which many algorithms exist. This is not surprising since these curves play a central role in many parts of mathematics. They are also used in cryptography and coding theory. Many other types of curves have been considered, for example hyperelliptic curves and so-called non-degenerate curves, as well as higher dimensional varieties. A major open problem is to find an algorithm for curves that would run in polynomial time in both the genus and the logarithm of the field size. Sometimes the requirement of being polynomial in the log of the degree of the curve is also asked for, in order to reflect more properly to size of the input data.

Typically one wants to determine the Zeta function of the variety, which encodes its number of rational points over all finite extensions of its field of definition. Half a century ago, André Weil studied these functions in great depth and was able to formulate his beautiful conjectures. To compute the Zeta function of some curves, an "l-adic method" has been developed a few decades ago by Schoof. Very roughly, these methods compute the numerator of Zeta modulo small primes l not equal to the characteristic p of the field. Such methods are now seen as reinterpretation of

l-adic (étale) cohomological tools, and are very efficient for elliptic curves over a field of large characteristic. This was initially suggested by Schoof, and later improved by several people such as Elkies, Atkin and Couveignes; it now runs in $\tilde{O}((\log q)^4)$, where q is the field size. Pila generalized the algorithm to abelian varieties, but it requires explicit equations for its definition and group law, which is generally a very hard task to find. This was of special interest because Jacobians of curves are of prime importance in public-key cryptography. Gaudry, Harley and Schost worked out the genus 2 case with slow time complexity. Adleman, Huang and Ierardi also improved Pila's result, but the result is still exponential in the genus.

Much more recently (2000) Satoh introduced p-adics methods. Although not explicitly, he used the action of the q-Frobenius in the deRham cohomology of the canonical lift (to characteristic zero) of the elliptic curve. Many authors have worked to improve this algorithm, which now runs with Harley's version at a complexity $\tilde{O}(n^2)$ in both time and space, where $q = p^n$ and p is fixed. It is however exponential in $\log p$, which is an unfortunate characteristic of p-adics methods. Attempts to take advantage of the more explicit nature of p-adic cohomologies for other types of curves led to computing the action of the q-Frobenius endormorphism on their cohomological complexes. The most successful of these attempts was done by Kedlaya, using MW (Monsky-Washnitzer) cohomology (i.e. a "rigid analytical lift"). It applies to hyperelliptic curves, and is the first part of my thesis. It has time complexity $\tilde{O}(g^4 n^3)$ and space complexity $O(g^{3+\epsilon} n^3)$, where g is the genus of the curve. Hence for fixed p it is polynomial in the size of the input data! Furthermore, the relatively small constants make it very efficient for small characteristics.

We also mention that as long as one can efficiently compute these cohomology groups and a lift of the Frobenius endomorphism, there seems to be no theoretical obstruction in generalizing this to more exotic types of curves and even to higher dimensional varieties. The implementation presented in the first part of the thesis is a straightforward generalization to superelliptic curves by Gaudry and Gürel, whereas the analysis in the second part of the thesis will rely heavily on a generalization to Cab curves by Vercauteren and Denef.

Other p-adic approaches include that of Lauder and Wan, who obtained a very general method using the ideas of Dwork (maybe one could argue that all p-adic methods are based on ideas of Dwork, but anyway). That unfortunately did not produce a practical algorithm. On the other hand, Mestre (and later Lercier, Lubicz, Ritzenthaler) has adapted his elliptic curve method to yield a practical algorithm for low genus, but it applies to a much narrower class of curves.

Finally a very promising approach but still less known is that of Lauder using deformation theory. This will be the object of Chapter 3.

CHAPTER 2
Determining the Zeta Function of Certain Types of Curves

In this chapter we present the main ingredients that gave life to Kedlaya's algorithm [Ked01], his following overview article [Ked04] and its generalizations. We also provide theoritical considerations to support the ideas involved. For reasons that will become clear in the next chapter, the focus is going to be on the superelliptic curves implementation by Gaudry and Gürel [GG01]. We believe that this does not hide any of the initial ideas of Kedlaya, and to the contrary shows to some extent how "generalizable" his algorithm his. Whenever there are important differences, we will also present the point of view of the Cab curves algorithm by Vercauteren and Denef [DV06]. This is not the most general as of now, because there is also a version for all non-singular curves by Castryck, Vercauteren and Denef [CDV06]; the reason for spending more time on [DV06] is also to prepare for the next chapter.

2.1 Setting

We start with a smooth affine plane curve $\bar{C}$ defined over the finite field $\mathbb{F}_q$, with $q = p^n$ and $p \geqslant 3$. We fix an algebraic closure $\bar{\mathbb{F}}$ of $\mathbb{F}_q$. We are interested in finding the number of $\mathbb{F}_{q^i}$-rational points of $\bar{C}$: the cardinality of E_i with

$$E_i := \{(a,b) \in \mathbb{F}_{q^i}^2 \quad | \quad (a,b) \in \hat{C} := \bar{C} \times_{\mathbb{F}_q} \bar{\mathbb{F}}\}, \quad i \geq 1.$$

E_i is also sometimes denoted $\bar{C}(\mathbb{F}_{q^i})$ and $\hat{C} = \bar{C} \times_{\mathbb{F}_q} \bar{\mathbb{F}}$ is the corresponding curve over the algebraic closure $\bar{\mathbb{F}}$ of $\mathbb{F}_q$. For this purpose, we define the Zeta function of $\bar{C}$:

Definition 2.1. The Zeta function of $\bar{C}$ is given by

$$Z(\bar{C}; T) := \exp\left(\sum_{i=1}^{\infty} \sharp E_i \frac{T^i}{i}\right) \in \mathbb{Q}[[T]].$$

In a similar way the Zeta function is defined for any scheme of finite type over a finite field. In particular, it is defined for the set of solutions in affine or projective space of a finite number of polynomials, the finiteness being automatic if the ambiant space is finite dimensional. We will discuss the Zeta function in more depth in 2.3. We now define the two classes of curves that will be of prime importance in this chapter. Note that the first class is actually a subclass of the second.

Definition 2.2. A superelliptic curve $\bar{Y}$ is given by an equation

$$\bar{f}(x, y) = y^r - \bar{h}(x) \in \mathbb{F}_q[x, y],$$

such that

i) $p \nmid r$.

ii) $\deg \bar{h}$ is relatively prime to r.

iii) Its affine part $\bar{C}$ is non-singular: this is equivalent to requiring that $\bar{h}$ has no repeated roots in $\bar{\mathbb{F}}$.

Definition 2.3. A Cab curve $\bar{Y}$ is defined by an equation of the form

$$\bar{f}(x, y) = y^a + \sum_{i=1}^{a-1} \bar{f}_i(x) y^i + \bar{f}_0(x) = 0,$$

with

i) $\deg \bar{f}_0 = b$ and $a \deg \bar{f}_i + bi < ab$ for $i = 1, \ldots, a-1$.

ii) a and b are coprime.

iii) Its affine part $\bar{C}$ is non-singular.

Matsumoto showed that a projective Cab curve is absolutely irreducible and has a unique $\mathbb{F}_q$-rational place at infinity. Since this is also true for superelliptic curves as defined above, one sees by a projection from the (projective) curve to $\mathbb{P}^1$ and the use of Riemann-Hurwitz formula that their genus is respectively

$$\frac{(r-1)(\deg \bar{h} - 1)}{2}, \quad \frac{(a-1)(b-1)}{2}.$$

Unlike l-adic methods, we hope to recover exactly the Zeta function of the curve, so the story begins by lifting $\bar{C}$ to characteristic zero. If $\bar{C}$ is defined over $\mathbb{F}_p$ then we will work in the p-adic numbers $\mathbb{Q}_p$ and the coefficients of $\bar{f}$ will be lifted arbitrarily to the ring of p-adic integers $\mathbb{Z}_p$. If the curve is only defined over some degree n extension $\mathbb{F}_q$ of $\mathbb{F}_p$, then we will work over the unique degree n unramified extension of $\mathbb{Q}_p$, denoted $\mathbb{Q}_q$. The p-adic norm on $\mathbb{Q}_p$ induces a norm on $\mathbb{Q}_q$, which is the n-th root of the norm of the product of all Galois conjugates of the element. Its ring of integers will be denoted by $\mathbb{Z}_q$ and its unique maximal ideal by $\boldsymbol{m}$. Note that as a set, $\boldsymbol{m}$ consist of the elements with norm less than one, and that the residue field $\mathbb{Z}_q/\boldsymbol{m} \cong \mathbb{F}_q$. The ring $\mathbb{Z}_q$ is isomorphic to the ring of Witt vectors over $\mathbb{F}_q$. So we will lift a coefficient of $\bar{f}$ to an arbitrary representative in $\mathbb{Z}_q$ under that isomorphism, lifting 0 to 0. If $\mathbb{F}_q \cong \mathbb{F}_p[z]/(\bar{g}(z))$ then $\mathbb{Q}_q \cong \mathbb{Q}_p[\beta] = \mathbb{Q}_p[z]/(g(z))$ for any lift $g(z)$ of $\bar{g}(z)$ and zero β of $g(z)$. Write C for the corresponding curve

over $\mathbb{Q}_q$. The theory will imply that the specific lift chosen will not matter for the purpose of computing the Zeta function, as long as one is consistent in making the same choices throughout the process. As in many other geometric theories over finite fields, the Frobenius endomorphism will be crucial. We will denote the p-Frobenius $\bar{\mathbb{F}} \to \bar{\mathbb{F}}, \quad a \mapsto a^p,$ by $\bar{\mathrm{F}}$ and the q-Frobenius $\bar{\mathbb{F}} \to \bar{\mathbb{F}}, \quad a \mapsto a^q$ by $\bar{\mathbf{F}}$ (so $\bar{\mathbf{F}} = \bar{\mathrm{F}}^n$). We can also look at $\bar{\mathbf{F}}$ on $\hat{C}$ (component-wise), and since $\mathrm{Gal}(\bar{\mathbb{F}}/\mathbb{F}_q) =< \bar{\mathbf{F}} >$, we have

$$E_i = \{(a,b) \in \bar{\mathbb{F}}^2 \,|\, \bar{\mathbf{F}}^i(a,b) = (a,b)\}.$$

In algebraic topology, we have a Lefschetz formula to count the number of fixed points of a morphism. However, as stated above, we wish to work in characteristic zero. The degree of $\bar{\mathbf{F}}$ being p^n, the Riemann-Hurwitz formula implies that if the curve C has genus ≥ 2 then it is impossible to lift $\bar{\mathbf{F}}$ to an endomorphism $\mathbf{F} : C \to C$, or equivalently to an homomorphism $A \otimes_{\mathbb{Z}_q} \mathbb{Q}_q \to A \otimes_{\mathbb{Z}_q} \mathbb{Q}_q$ of the ring of coordinates of C, where A is the $\mathbb{Z}_q$algebra $\mathbb{Z}_q[x,y]/(f)$. The natural thing to look at is the p-adic completion of $\hat{A} := \mathbb{Z}_q\langle x,y\rangle/(f)$ of A (ie completion with respect to pA; it consists of the power series in x, y that converge on the closed disc in $\mathbb{Q}_q^2$ of radius 1 and centered at the origin) , which M. Artin showed to be an Henselian ring [Art68]. Starting with f, we will see later that it is easy to construct a polynomial $G(Z) \in \hat{A}[Z]$ such that $G(0) \equiv 0 \pmod{pG'(0)^2\hat{A}}$, which is sufficient to find a solution $G(Z_0) = 0$ and from which one can easily recover $\mathrm{F}(x)$ and $\mathrm{F}(y)$. However it is well-known that the algebraic deRham cohomology of $\hat{A}$, i.e. (because we are in the affine case) the homology of the complex $0 \longrightarrow \hat{A} \xrightarrow{d_0} \Omega^1_{\hat{A}} \xrightarrow{d_1} \Omega^2_{\hat{A}} \xrightarrow{d_2} \ldots$ (see below), is too large and does not have the properties we need for our computations.

Typically, contrary to archimedean analysis, in non-archimedean analysis one loses convergence on the boundary: a common example is the limit (as $N \to \infty$) of the exact forms $\sum_{i=0}^{N} p^i x^{p^i-1} dx$ is not exact, because $\sum_{i=0}^{\infty} x^{p^i} \notin \hat{A}$ This is unfortunate because $\hat{A} \otimes_{\mathbb{Z}_q} \mathbb{Q}_q$ is a $\mathbb{Q}_q$-affinoid algebra, being the quotient of the Tate algebra in 2 variables over $\mathbb{Q}_q$ by the ideal generated by f, and we know a lot of things about $\mathbb{Q}_q$-affinoids. Still one is tempted to think that working inside $\hat{A}$ should be sufficient, and to fix the cohomological problem we could possibly restrict to some $A \subsetneqq A^\dagger \subsetneqq \hat{A}$.

2.2 The Monsky–Washnitzer cohomology

The Monsky-Washnitzer cohomology is based on the work of Dwork, and was initially developed precisely to find an explicit expression for the Zeta function of an algebraic variety $\bar{X}$ over the finite field $\mathbb{F}_q$. As introduced above, one wishes to find cohomology groups $H^i(\bar{X}; \mathbb{Q}_q)$ (which by construction will be modules over $\mathbb{Q}_q$, and hence $\mathbb{Q}_q$-vector spaces) with an induced Frobenius $\mathbf{F}$ and for which a Lefschetz fixed-point formula holds true.

Assume that $\bar{X}$ is affine. In a similar fashion as done in the introduction, one constructs the finitely-generated, smooth $\mathbb{Z}_q$-algebra

$$B := \mathbb{Z}_q[x_1, \ldots, x_n]/(f_1, \ldots, f_m)$$

such that B/pB is the coordinate ring of $\bar{B}$ of $\bar{X}$. The fact that this is possible is due to Elkik [Elk73]. Now intuitively speaking and in light of the comments above, one would like to extend B to include power series converging on a disc of radius at least greater than 1 (to be in p-adic completion $\hat{B}$ and to prevent the cohomology

from getting too large), but at the same time should not require convergence on a disk of radius much more than 1 hoping to avoid loosing the possibility of lifting the Frobenius. It turns out that requiring convergence on the disc of radius $1+\epsilon$ with $\epsilon > 0$ varying is precisely what is is needed.

Definition 2.4. (cf [VdP86], Definition 2.1) A *weakly complete finitely generated (w.c.f.g.) algebra over* $\mathbb{Z}_q$ is a homomorphic image of the ring $\mathbb{Z}_q\langle t_1, \ldots, t_n\rangle^\dagger$, where $\mathbb{Z}_q\langle t_1, \ldots, t_n\rangle^\dagger$ is

$$\left\{ \sum_{i_1,\ldots,i_n \geq 0} e_{i_1,\ldots,i_n} t_1^{i_1} \cdots t_n^{i_n} \quad \Big| \quad \exists\, c > 0,\ 0 < \rho < 1 \text{ such that } |e_{i_1,\ldots,i_n}|_p \leq c\rho^{(i_1+\cdots+i_n)} \right\}.$$

Note that the condition $\exists c > 0, 0 < \rho < 1$ such that $|e_{i_1,\ldots,i_n}|_p \leq c\rho^{(i_1+\cdots+i_n)}$ is equivalent to $\exists a > 0, b$ such that $v_p(e_{i_1,\ldots,i_n}) > a(i_1 + \cdots + i_n) - b$, which justifies talking about a linear growth requirement on the valuation of the coefficient. For $B^\dagger := \mathbb{Z}_q\langle x_1, \ldots, x_n\rangle^\dagger/(f_1, \ldots, f_m)$, we denote the universal finite module of (Kähler) differentials of $B^\dagger/\mathbb{Z}_q$ by $\Omega^1_{B^\dagger}$. It is the $B^\dagger$-module

$$B^\dagger dx_1 + \cdots + B^\dagger dx_n \quad / \quad \sum_{i=1}^{m} B^\dagger \left(\frac{\partial f_i}{\partial x_1} dx_1 + \cdots + \frac{\partial f_i}{\partial x_n} dx_n \right).$$

Finally we look at the deRham complex

$$0 \longrightarrow B^\dagger \xrightarrow{d_0} \Omega^1_{B^\dagger} \xrightarrow{d_1} \Omega^2_{B^\dagger} \xrightarrow{d_2} \ldots$$

where $\Omega^i_{B^\dagger}$ is the wedge product $\wedge^i \Omega^1_{B^\dagger}$ and d_i is the corresponding exterior derivative. For the remaining part of this thesis we will write $H^i(\bar{X}; \mathbb{Z}_q)$ for the i^{th}

cohomology group of this complex, and $H^i(\bar{X};\mathbb{Q}_q) := H^i(\bar{X};\mathbb{Z}_q)\otimes_{\mathbb{Z}_q}\mathbb{Q}_q$ (or sometimes with $\bar{B}$ instead of $\bar{X}$). The latter is called the i^{th} Monsky-Washnitzer cohomology group of $\bar{X}$. To prove that this notation makes sense, i.e. that $H^i(\bar{X};\mathbb{Q}_q)$ does not depend on the arbitrary choices made along the way (for example how we obtained B from $\bar{B}$), proved to be quite a laborious task. More precisely, for a smooth finitely generated $\mathbb{F}_q$-algebra $\bar{B}$ the map $\bar{B}\mapsto H^{\cdot}(\bar{B};\mathbb{Q}_q) := \bigoplus_{i\geq 0} H^i(\bar{B};\mathbb{Q}_q)$ is well-defined and functorial. This is seen by applying the following theorem to the identity map $\bar{B}\to\bar{B}$:

Theorem 2.5. *(cf [VdP86], Theorem 2.4.4) Let $\bar{A}/\mathbb{F}_q$ be smooth and f.g. Then there exist a lift A of $\bar{A}$, i.e. a w.c.f.g. algebra that is flat over $\mathbb{Z}_q$ and with $A/pA\cong\bar{A}$. Moreover:*

i) Every lift of $\bar{A}$ is $\mathbb{Z}_q$-isomorphic to A.

ii) Let $\bar{D}/\mathbb{F}_q$ be smooth and f.g., let D be a lift of $\bar{D}$ and let $\bar{h}:\bar{A}\to\bar{D}$ be a $\mathbb{F}_q$-homomorphism. Then there exists a $\mathbb{Z}_q$-homomorphism $h:A\to D$ lifting $\bar{h}$.

iii) Let E be a w.c.f.g. algebra and $h_1,h_2:A\to E$ two homomorphism with $h_1\equiv h_2 \pmod{p}$. The induced mappings $h_{1\star},h_{2\star}:\Omega^1_A\otimes_{\mathbb{Z}_q}\mathbb{Q}_q\to\Omega^1_E\otimes_{\mathbb{Z}_q}\mathbb{Q}_q$ are homotopic.

This theorem is obviously relevant for our purpose, not only for providing us a cohomology to work with, but on which we are sure to have a lift $\mathbf{F}$ of the Frobenius $\bar{\mathbf{F}}$. Moreover, whichever lift we use, it will produce the same morphism on the cohomology! This will turn out very useful in the next chapter.

Monsky and Washnitzer could not prove that their cohomology vector spaces are finite dimensional, but they introduced a "trace map" ψ on the differentials (which induces a map on $H^{\cdot}(\bar{B};\mathbb{Q}_q)$ equal to $q^d\mathbf{F}_\star^{-1}$, with $d = \dim \bar{B}$) and were able to find enough of its properties to get a Lefschetz formula for it:

Theorem 2.6. *(cf [VdP86], Theorem 4.1) Let $\bar{A}/\mathbb{F}_q$ be smooth and integral of dimension d. Then the number of $\mathbb{F}_q$-homomorphisms $\bar{A} \to \mathbb{F}_q$ (or in geometric language, $\mathbb{F}_q$-rational points) is equal to*

$$\sum_{i=0}^{d}(-1)^i\, Tr\,(q^d \boldsymbol{F}_\star^{-1}|H^i(\bar{A};\mathbb{Q}_q)).$$

Note that so far we have only worked with smooth affine algebraic sets. The integral requirement simply means that we want our affine set to be irreducible (ie, a (single) affine variety) and reduced (i.e. $\bar{A}$ has no nilpotent elements). In fancier language, we build an affine scheme $\mathrm{Spec}\, B^\dagger$ over $\mathbb{Z}_q$ such that its special fiber is $\mathrm{Spec}\,\bar{B}$ and the deRham cohomology of its generic fiber has the properties that Weil hoped for (see 2.3). $B^\dagger$ is an analytic object, which makes it less easy to compute with; but under certain conditions, could we expect (say via a rigid analytic version of Serre's GAGA and cohomological comparison theorems) to move toward more algebraic objects? Some comparisons already appear in [VdP86], but a more general answer was provided by Berthelot as part of a masterpiece called rigid cohomology. Roughly speaking, rigid cohomology coincides with MW cohomology for smooth affine varieties and with crystalline cohomology for smooth projective varieties. Computationally the latter comparison can help to keep track of the p-adic precision required (crystalline cohomology is integral, i.e. over $\mathbb{Z}_q$ in our

context), whereas as the former tells us that we can work with forms of finite degree. More precisely, we have the following:

Theorem 2.7. *(cf [Ked04], Theorem 1) Let Y be a smooth proper $\mathbb{Z}_q$-scheme, let $Z \subset Y$ be a relative normal crossing divisor, and set $X := Y \backslash Z$. Then there is a canonical isomorphism*

$$H^i_{dR}(X \times_{\mathbb{Z}_q} \mathbb{Q}_q) \xrightarrow{\cong} H^i_{rig}(X \times_{\mathbb{Z}_q} \mathbb{F}_q).$$

This comparison also implies that the MW cohomological vector spaces are finite dimensional, and even better have the same dimension as other cohomological theories. For Cab curves, we will take Z to be the unique point at infinity; for superelliptic curves it is possible (and more convenient) to take the point at infinity together with the Weierstrass points $(y = 0)$.

We finish this section with a theorem on the Artin-Approximation property of w.c.f.g. algebras. This result has been generalized by Bosch [Bos81], however the theorem as it appears in [VdP86] will be sufficient for us.

Theorem 2.8. *Let $h_1, \ldots, h_s \in \mathbb{Z}_q\langle X_1, \ldots, X_n, Y_1, \ldots, Y_m\rangle^\dagger$, let $\epsilon > 0$ and let $\hat{y}_1, \ldots, \hat{y}_m \in \mathbb{Z}_q\langle X_1, \ldots, X_n\rangle$ satisfy $h_i(X_1, \ldots, X_n, \hat{y}_1, \ldots, \hat{y}_m) = 0$ for $i = 1, \ldots, s$. Then there are $y_1, \ldots, y_m \in \mathbb{Z}_q\langle X_1, \ldots, X_n\rangle^\dagger$ with $|y_i - \hat{y}_i| \leq \epsilon$ for $i = 1, \ldots, m$ and $h_i(X_1, \ldots, X_n, y_1, \ldots, y_m) = 0$ for $i = 1, \ldots, s$.*

Combined with [Art68], we will use this result to lift the Frobenius action to $B^\dagger$.

2.3 The Weil conjectures

For a superelliptic or Cab curve $\bar{Y}$ and its affine affine part $\bar{C}$, one can compute that $H^0(\bar{C};\mathbb{Q}_q) = \mathbb{Q}_q$ and $H^i(\bar{C};\mathbb{Q}_q) = 0 \quad \forall\, i \geq 2$. Therefore from Theorem 2.6 applied to $\bar{C}$, one may use Newton's determinant formula to find

$$Z(\bar{C};T) = \frac{\det\left(1 - q\mathbf{F}_\star^{-1}T \,|\, H^1(\bar{C};\mathbb{Q}_q)\right)}{1-qT}. \tag{2.1}$$

Also, since $Z(\bar{Y};T) = Z(\bar{C};T)Z(\{\infty\};T)$, we get

$$Z(\bar{Y};T) = \frac{\det\left(1 - q\mathbf{F}_\star^{-1}T \,|\, H^1(\bar{C};\mathbb{Q}_q)\right)}{(1-T)(1-qT)}.$$

We will need to lift the Frobenius action on $H^1(\bar{C};\mathbb{Q}_q)$, and computationally this can only be done up to some finite p-adic precision. Therefore we need more information on the numerator above. This information will be provided by the Weil conjectures, which will be presented here with an historical motivation inspired by Dieudonné.

E. Artin in 1923 looked at $\bar{D} : y^2 = P(x)$ over $\mathbb{F}_q$, where $P(x)$ has no repeated roots in $\bar{\mathbb{F}}$. He considered $L_0 := \mathbb{F}_q(x)$, $L := L_0[y]/(y^2 - P(x))$ and $\boldsymbol{o}$ the integral closure of $\mathbb{F}_q[x]$ in L. Then $\boldsymbol{o}$ is a Dedekind ring and for every $\boldsymbol{p} \triangleleft \boldsymbol{o}$ we have $[\boldsymbol{o}/\boldsymbol{p} : \mathbb{F}_q] < \infty$. Artin studied Dedekind's generalization of the Zeta function, defining

$$\zeta_L(s) := \sum_{\boldsymbol{a} \triangleleft \boldsymbol{o}} (N\boldsymbol{a})^{-s} = \prod_{\text{prime } \boldsymbol{p} \triangleleft \boldsymbol{o}} (1 - (N\boldsymbol{p})^{-s})^{-1},$$

where $N\boldsymbol{a} := \sharp \boldsymbol{a}/\boldsymbol{o}$. Then Hecke showed that ζ_L can be extended to a meromorphic function and satisfies a functional equation. For $\boldsymbol{a} \triangleleft \boldsymbol{o}$ write

$\deg \boldsymbol{a} := [\boldsymbol{o}/\boldsymbol{p} : \mathbb{F}_q]$ so that $N\boldsymbol{a} = q^{\deg \boldsymbol{a}}$. Define $Z_{\bar{D}}(u) := \prod_{\text{prime } \boldsymbol{p} \lhd \boldsymbol{o}} (1 - u^{-\deg \boldsymbol{p}})^{-1}$ so that $\zeta_L(s) = Z_{\bar{D}}(q^{-s})$. Artin was able to show that $Z_{\bar{D}}(u)$ is rational with coefficient in $\mathbb{Z}$. He also verified for several examples that the zeroes of $Z_{\bar{D}}(u)$ would have $|u| = \sqrt{q}$. Now homomorphisms $\boldsymbol{o} \to \mathbb{F}_q$ are surjective, so they are in bijection with primes $\boldsymbol{p}$ with $\boldsymbol{o}/\boldsymbol{p} \cong \mathbb{F}_q$ (i.e. $N\boldsymbol{p} = q$). These homomorphisms correspond to $\mathbb{F}_q$-rational points of $\bar{D}$ under the map $(x, y) \mapsto (a, b) \in \mathbb{F}_q^2$. So $\log Z_{\bar{D}}(u) = \sharp \bar{D}(\mathbb{F}_q)u + \dots$ Putting all this together (and assuming his Riemann hypothesis type conjecture) he was led to approximate the number of $\mathbb{F}_q$-rational points by $\sharp E_1 = \sharp \bar{D}(\mathbb{F}_q) = q + O(q^{1/2})$.

With the purpose of counting rational points of more exotic varieties in mind, one would like to move away from number theory toward a completely algebraic geometry based approach. How can we express the Zeta function in that language, and what can we say about it? For example, one could start with a nonsingular affine hypersurface $\bar{V}$ defined over $\mathbb{F}_q$, say by $\bar{Q}(x_1, \dots, X_r) = 0$. Let $V := \bar{V} \times_{\mathbb{F}_q} \bar{\mathbb{F}}$. As usual, given $x = (x_1, \dots, x_r) \in V$ one may produce a homomorphism $\mathbb{F}_q[T_1, \dots, T_r] \to \bar{\mathbb{F}}$, $T_i \mapsto x_i$ having kernel m and get that $\mathbb{F}_q(x) := \mathbb{F}_q[T_1, \dots, T_r]/m$ is the smallest field extension of $\mathbb{F}_q$ containing the $x_j's$: define $\deg(m) := [\mathbb{F}_q(x) : \mathbb{F}_q]$. Finally let $Z_{\bar{V}}(u) := \prod_{\bar{Q} \in m} (1 - u^{\deg(m)})^{-1}$. If $\bar{V} = \bar{D}$ then this agrees with Artin's definition. It is not hard to see that for every m such that $\bar{Q} \in m$ there are $\deg(m)$ points $x \in V$ that would give a kernel m in the evaluation map $T_i \mapsto x_i$. Hence $N_w = \sum_{\substack{\bar{Q} \in m \\ \deg(m) | \, w}} \deg(m)$ Using this, a simple

calculation shows that $\frac{uZ'}{Z} = \sum_{i=1}^{\infty} N_i u^i$. Solving the ODE we get

$$Z_{\bar{V}}(u) = e^{\sum_{i=1}^{\infty} N_i \frac{u^i}{i}}.$$

Since everything was done completely algebraically and every step seemed pretty harmless, one could be tempted to believe that this should be generalizable to other types of nonsingular varieties, not necessarily affine. This is what Schmidt did for a nonsingular projective curve $\bar{D}/\mathbb{F}_q$, showing that the Zeta function produced was of the form $\frac{P_{2g}(u)}{(1-u)(1-qu)}$ with the numerator a polynomial of degree twice the genus with integral coefficients. He obtained the functional equation $Z_{\bar{D}}(\frac{1}{qu}) = (qu^2)^{1-g} Z_{\bar{D}}(u)$. Then the conjecture that the zeroes of P_{2g} have $|u| = \sqrt{q}$ is equivalent to

$$|E_i - q^i - 1| \leq 2gq^{i/2} \quad \forall\, i \geq 1.$$

Theorem 2.9. *(The Weil conjectures) Let $\bar{U}/\mathbb{F}_q$ be smooth, projective, of dimension d. Then*

i) Rationality: $Z(\bar{U}; T) = \frac{g(T)}{h(T)}$ *with* $g, h \in \mathbb{Z}[T]$

ii) Functional equation: There exist $E \in \mathbb{Z}$ *such that* $Z\left(\bar{U}; \frac{1}{q^d T}\right) = \pm q^{\frac{dE}{2}} T^E Z(\bar{U}; T)$. *That is* $\zeta(s) = \zeta(1-s)$ *for* $\zeta(s) := Z(\bar{U}; q^{-s})$.

iii) Analogue of Riemann hypothesis: $Z(\bar{U}; T) = \frac{P_1(T) P_3(T) \cdots P_{2d-1}(T)}{P_0(T) P_2(T) \cdots P_{2d}(T)}$ *where*

- $P_0(T) = 1 - T$
- $P_{2d}(T) = 1 - q^d T$
- $P_i(T) = \prod_{j=1}^{B_i} (1 - \alpha_{ij} T) \quad \in \mathbb{Z}[T]$

- *The* α_{ij}*'s are algebraic integers that imbedded into* $\mathbb{C}$ *have* $|\alpha_{ij}| = q^{i/2}$

That is, the zeroes of $\zeta_i(s) = P_i(q^{-s})$ *have real(s)* $= i/2$.

iv) Betti numbers: $E = \sum_{i=0}^{2d} (-1)^i B_i$

Furthermore if $\bar{U}$ *is the reduction modulo* $\boldsymbol{p} \triangleleft \boldsymbol{o}_L$ *of some variety* W *over* $\boldsymbol{o}_L$ *for some number field* L, *one can look at the complex analytic space* W_h *associated with the corresponding scheme* $W \times_{\boldsymbol{o}_L} \mathbb{C}$ *over* $\mathbb{C}$. *Then* $B_i = rank_{\mathbb{Z}} H^i_{singular}(W_h, \mathbb{Z}) = i^{th}$ *Betti number.*

The proof for $\bar{U}$ a curve is due to André Weil himself. The first proof of *i*) and *ii*) for higher dimensional varieties is due to Dwork. In a similar fashion as presented at the beginning of this section, note that if one had a well behaved cohomology theory for $\bar{U}$ with coefficients in a characteristic zero field and with a Lefschetz fixed point formula, then modulo some facts about rational functions over that field one would get *i*). This was accomplished by Grothendieck's étale cohomology, by Lubkin's p-adic cohomology (maybe not very rigourously) and by Grothendieck's crystalline cohomology. *ii*) can be obtained from Poincaré duality on these theories and some facts about pairings, whereas *iv*) can be obtained from comparison theorems. *iii*) proved to be the hardest to establish. Deligne finally succeeded within the framework of étale cohomology, and more recently Kedlaya (with ideas of Crew and Mebkhout) repeated the exploit within the framework of rigid cohomology (thereby completing a fully p-adic proof of the conjectures).

2.4 Details of an implementation: theoretic

In this section we present concretely how one may determine the Zeta function of one of the curves $\bar{C}$ defined in 2.1, the main focus being on the superelliptic curve case. We need to find the numerator $P_{\bar{C}}$ of (2.1). If we assume for now that $H^1(\bar{C};\mathbb{Q}_q)$ has dimension $2g$, following Theorem 2.9 *i)* we may write $P_{\bar{C}}(T) = T^{2g} - a_1T^{2g-1} + \cdots - a_{2g-1}T + a_{2g} \in \mathbb{Z}[T]$. Poincaré duality implies that for some labelling, the α_{1j}'s in Theorem 2.9 *iii)* satisfy $\alpha_{1j}\alpha_{1,g+j} = q$ and this in turn gives

$$a_{2g-i} = q^{g-i}a_i \quad \forall\, 0 \le i \le g. \tag{2.2}$$

So we only need to compute a_i for $0 \le i \le g$, and we observe that we have $|a_i| \le \binom{2g}{i} q^{i/2}$. Therefore one needs to know the a_i's as elements of $\mathbb{Z}_q/p^N\mathbb{Z}_q$ where $p^N \ge 2\binom{2g}{g} q^{g/2}$. Considering the eigenvalues of $q\mathbf{F}_\star^{-1}$ and $\mathbf{F}_\star$ together with $\alpha_{1j}\alpha_{1,g+j} = q$ we get

$$\det\left(1 - q\mathbf{F}_\star^{-1}T \,|\, H^1(\bar{C};\mathbb{Q}_q)\right) = \det\left(1 - \mathbf{F}_\star T \,|\, H^1(\bar{C};\mathbb{Q}_q)\right).$$

Lemma 2.10. *For the superelliptic curve $\bar{C}$ in Definition 2.2, $H^1(\bar{C};\mathbb{Q}_q)$ has $\mathbb{Q}_q$-basis:*

$$\left\{ \frac{x^i dx}{y^j} \quad : \quad 0 \le i \le (deg\ \bar{h} - 2),\ 1 \le j \le (r-1) \right\}.$$

For the Cab curve $\bar{C}$ in Definition 2.3, $H^1(\bar{C};\mathbb{Q}_q)$ has $\mathbb{Q}_q$-basis:

$$\left\{ x^i y^j dx \quad : \quad 0 \le i \le (b-2),\ 1 \le j \le (a-1) \right\}.$$

We are implicitly using the fact that the proposed basis for the superelliptic curve has no pole at $y = 0$, which is true by Equation 3.7. In order to show that the proposed basis spans $H^1(\bar{C}; \mathbb{Q}_q)$ one can use the reduction algorithms that will be presented below. To show that it is linearly independent, one can work directly or use Theorem 2.7 together with the known dimensions of the algebraic DeRham cohomology of these curves. For hyperellptic curves (r=2), a similar result is proved in a great level of details in the proofs of Propositions 4.3.1, 4.3.2 in [Edi03] (Actually he proves that these 1-forms form a basis of $H^1(\bar{C}'; \mathbb{Q}_q)^-$: see below).

Let us now see how we can lift $\bar{\mathbf{F}}$. We will actually lift the p-th power Frobenius $\bar{\mathrm{F}}$ to F, and then take its n-power. We will start with the Cab curves case. F leaves $\mathbb{Q}_p$ fix, and if $\mathbb{Q}_q = \mathbb{Q}_p[\sigma] \cong \mathbb{Q}_p[z]/(g(z))$ one can use Hensel's lemma to find $\mathrm{F}(\sigma)$ such that $g(\mathrm{F}(\sigma)) = 0$ and $\mathrm{F}(\sigma) \equiv \sigma^p \pmod{p}$. To lift x and y the first thing to try is probably to send $x \mapsto x^p$ and see where y would have to map to in order to get $\mathrm{F}(f) = 0$. It is easy to see that in general the image of y will unfortunately not lie in $A^\dagger$. Following [DV06], we try next $x \mapsto x^p + \delta_x V$ and $y \mapsto y^p + \delta_y V$ for some $V \in A^\dagger$, $V \equiv 0 \pmod{p}$. Let $G(V) = f^{\mathrm{F}}(x^p + \delta_x V, y^p + \delta_y V)$. Since $\bar{C}$ is non-singular, the Nullstellensatz implies that there exist $\bar{\alpha}, \bar{\beta}, \bar{\gamma} \in \mathbb{F}_q[x, y]$ such that

$$\bar{\alpha}(x,y)\frac{\partial \bar{f}}{\partial x}(x,y) + \bar{\beta}(x,y)\frac{\partial \bar{f}}{\partial y}(x,y) + \bar{\gamma}(x,y)\bar{f}(x,y) = 1. \tag{2.3}$$

So if δ_x is a lift of $\bar{\alpha}^p$ and δ_y^p is a lift of $\bar{\beta}$, then

$$G'(0) \equiv \delta_x \frac{\partial \bar{f}^{\mathrm{F}}}{\partial x}(x^p, y^p) + \delta_y \frac{\partial \bar{f}^{\mathrm{F}}}{\partial y}(x^p, y^p) \equiv 1 \pmod{p}. \tag{2.4}$$

Thus by Theorem 2.8 we can find a solution $V \in A^\dagger$. Now for the superelliptic case, we could do the same on $A^\dagger = \mathbb{Z}_q\langle x, y\rangle^\dagger/(f)$ to get a lift of Frobenius F_1. However as noted after Theorem 2.7, we will instead remove the points $y = 0$ from $\bar{C}$, and denote the resulting affine curve $\bar{C}'$. Its ring of coordinates is $\mathbb{F}_q[x, y, y^{-1}]/(f)$ so we will use the w.c.f.g. algebra $A'^\dagger = \mathbb{Z}_q\langle x, y, y^{-1}\rangle^\dagger/(f)$. This will allow us to set $\mathrm{F}(x) = x^p$ without running into trouble; the advantage is that the lift created will be much faster to compute. To see what $\mathrm{F}(y)$ is, write $h(x) := y^r - f(x, y)$ and

$$\mathrm{F}(y^r) = \mathrm{F}(h(x)) = h(x)^p \left(1 + \frac{\mathrm{F}(h(x)) - h(x)^p}{h(x)^p}\right) = y^{rp}\left(1 + \frac{pE}{y^{rp}}\right) \tag{2.5}$$

where $E := \frac{\mathrm{F}(h(x))-h(x)^p}{p} \in \mathbb{Z}_q[x]$. By the binomial expansion theorem this give

$$\mathrm{F}(y) = y^p \sum_{k=0}^{\infty} \binom{1/r}{k} \left(\frac{pE}{y^{rp}}\right)^k. \tag{2.6}$$

Clearly, F_1 also induces a lift of Frobenius on $A'^\dagger$, and by Theorem 2.5 both induced map $\mathrm{F}_\star$, $\mathrm{F}_{1\star}$ on $H^1(\bar{C}'; \mathbb{Q}_q)$ are equal. This means that $H^1(\bar{C}; \mathbb{Q}_q)$ as a $\mathbb{Q}_q$-subspace of $H^1(\bar{C}'; \mathbb{Q}_q)$ is stable under the action of $\mathrm{F}_\star$, and that

$$\det\left(1 - \mathbf{F}_{1\star}T \,|\, H^1(\bar{C}; \mathbb{Q}_q)\right) = \det\left(1 - \mathbf{F}_\star T \,|\, H^1(\bar{C}'; \mathbb{Q}_q)^-\right)$$

where $H^1(\bar{C}'; \mathbb{Q}_q)^- := \mathrm{span}_{\mathbb{Q}_q}\left\{\frac{x^i dx}{y^j} \in \Omega^1_{A'^\dagger} \;:\; 0 \le i \le (\deg \bar{h} - 2),\ 1 \le j \le (r-1)\right\}$. By naturality of $F_\star$ we get $\mathrm{F}_\star\left(\frac{x^i dx}{y^j}\right) = px^{p-1}\mathrm{F}\left(\frac{x^i}{y^j}\right)dx$, hence we can compute the endomorphism $\mathrm{F}_\star$ on the 2g-dimensional vector space $H^1(\bar{C}'; \mathbb{Q}_q)^-$.

2.5 Details of an implementation: computational

At this point we are theoretically ready to go; however of course computationally there are still several things to sort out. Among them is how we truncate (2.6) and

with what p-adic precision the calculations must be done. For the reader who is interested in specific implementation details, we believe that including the Maple implementation of [GG01] is much more precise and efficient than trying to write everything up. The major advantage of Maple in this context is that it provides an algorithm just "as one would do it by hand", without using any fancy hidden structures. This comes at the price of not being computationally efficient, but for understanding purpose it is perfect!

Suppose we have a hyperelliptic curve $y^3 = (1+\beta)x^4 + \beta^3 x^3 + 1 + 2\beta$ over $\mathbb{F}_{7^3} = \mathbb{F}_7[\beta] = \mathbb{F}_7[z]/(z^3+z^2+4z+6)$. Then one would define beta and Hc and run FullProgram as follows:

```
alias(beta=RootOf(z^3+z^2+4*z+6));
Hc := [y,3, (1+beta)*x^4+beta^3*x^3+1+2*beta, 7,3,beta,z^3+z^2+4*z+6];
FullProgram;
```

In this thesis' notation $Hc := [y, r, h(x), p, n, \beta, g(z)]$, i.e. we use a naive lift of $\bar{h}(x)$ so that lifted coefficients lie in $\{0, \ldots, p-1\}$. We begin by presenting two small programs that imitate the traditional mod p^{N1} :

```
#Mod polns by p^N1, taking care of possible p in the denominators
precision:=proc(f,N1,p,x,tauu,alpha)
local kkk,ff,ffi,i,j,kk,ffij,fff,k,ffijk;
options system;
fff:=0;
ff:=evala(Simplify(evala(f)),expanded);
kkk:=padic[ordp](denom(ff),p);
```

```
#lprint("precision ", ff);
if kkk=0 then
    fff:=modp(ff,p^N1);
else
 lprint("divided by p to the ",kkk);
  for i from 0 to degree(ff,tauu) do
   ffi:=coeff(ff,tauu,i);
     for j from 0 to degree(ffi,x) do
        ffij:=coeff(ffi,x,j);
          for k from 0 to degree(ffij,alpha) do
            ffijk:=coeff(ffij,alpha,k);
            kk:=padic[ordp](denom(ffijk),p);
            fff:=fff+modp(p^kk*ffijk,p^(N1+kk))*x^j*tauu^i*alpha^k/p^kk;
          od;
     od;
  od;
end if;
end:  #end precision

#Mod matrices by p^N1, taking care of possible p in the denominators
matrixprecision:=proc(M,n,alpha,p,N1)
local i,j,k,Mijk,kk,MMM;
options system;
```

```
MMM:=Matrix(n);
for i from 1 to n do
   for j from 1 to n do
     for k from 0 to degree(M[i,j],alpha) do
       Mijk:=coeff(M[i,j],alpha,k);
       kk:=padic[ordp](denom(Mijk),p);
       MMM[i,j]:=MMM[i,j]+modp(p^kk*Mijk,p^(N1+kk))*alpha^k/p^kk;
     od;
   od;
od;
MMM;
end:  #matrixprecision
```

One sees that we really work mod p^{N1} fixed instead of working in $\mathbb{Q}_q$ with "floating precision N1"; the reason for this is how the analysis (to show that the chosen precisions are sufficient) is done. Next we present 3 little programs that correspond to Step 1 to 3 of the reduction process in [GG01].

```
#Gaudry/Gurel Reduction 1
Red1:=proc (f, tauu, Q, x,N1,p,alpha)
local i,ff,ffi,rr,qq, deg;
options system;
deg := degree(f, tauu);
ff:=f;
for i from deg by -1 to 1 do
```

```
ffi:=coeff(ff, tauu, i);
rr:=rem(ffi,Q,x,'qq');
ff:=precision(ff+(rr-ffi)*tauu^i+qq*tauu^(i-1),N1,p,x,tauu,alpha);
ff:=sort(collect(evala(Simplify(evala(ff)),expanded),tauu),tauu);
end do;
#ff:=sort(collect(ff,tauu),tauu);
end:  #Red1

#Gaudry/Gurel Reduction 2
Red2:=proc (f, tauu, Q, x,r,Qp,u,v,vp,l,N1,p,alpha)
local ff,deg,ffi,U,Vp,i,ffip;
options system;
deg := degree(f, tauu);
ff:=f;
for i from deg by -1 to 1 do
ffi:=coeff(ff, tauu, i);
U:=ffi*u;
ffip:=diff(ffi,x);
Vp:=vp*ffi+v*ffip;
ff:=ff-ffi*tauu^i+(U+(r/(r*(i-1)+l))*Vp)*tauu^(i-1);
ff:=sort(collect(precision(ff,N1,p,x,tauu,alpha),tauu),tauu);
end do;
ff:=sort(collect(ff,x),x);
```

```
end:  #Red2


#Gaudry/Gurel Reduction 3
Red3:=proc (f, Q, tauu, x,r,Qp,l,d,Qd,N1,p,alpha)
local ff,deg,alphaa,i;
options system;
deg := degree(f, x);
ff:=f;
for i from deg by -1 to (d-1) do
alphaa:=coeff(ff, x, i)/(Qd*(i-d+1+(r-l)*d/r));
ff:=ff-alphaa*(i-d+1)*x^(i-d)*Q-alphaa*(r-l)*x^(i-d+1)*Qp/r;
ff:=sort(collect(precision(ff,N1,p,x,tauu,alpha),x),x);
end do;
end:  #Red3
```

Basically, Reduction 1 rewrites the series using the equation of the curve to have degrees in x less than deg $\bar{h}$ (except when that would produce monomials with a positive power of y), then Reduction 2 uses the cohomological equivalence

$$Q_k(x)\tau^k\frac{dx}{y^l} \equiv (U + \frac{r}{r(k-1)+l}V')\tau^{k-1}\frac{dx}{y^l}$$

where $Q_k = Uh + Vh'$, $\tau = \frac{1}{y^r}$ to increase the powers of y appearing. Finally Reduction 3 reduces the degree in x of $Q(x)\frac{dx}{y^l}$ using

$$d(x^{\deg Q - \deg h + 1}y^{r-l}) \equiv 0.$$

Finally, we present the program that make everything run together, with comments at each step.

Note that as it stands, the variable INCREASE has to be adjusted by hand. There is an explicit bound in [GG01] for it, however as observed by Vercauteren and later documented by Edixhoven [Edi03] and then Kedlaya, the matrix of Frobenius does not necessarily have integral entries for small primes, and hence the stated bound is not big enough. One could fix this by taking the non-integrality into consideration in the loss-of-precision analysis, similar to what we will do in Section 3.3, but for our purpose fixing it by hand will be sufficient. If one is interested to carry out this analysis, we would suggest to follow that of [Edi03], noting that a superelliptic generalization of Lemma 2 in [Ked01] gives that the reduction of $Q_k(x)\tau^k \frac{dx}{y^l}$ becomes integral upon multiplication by $p^{\lfloor \log_p(rk+l) \rfloor}$.

```
FullProgram:=proc (hc)
options system;
local Q, p, t, g, T,TT,h, Dh,Dhe,Fa,tt,alphap,Falpha,f,x,r,d,n,N,N1,
i,sqt,yis,Qp,Qd,gg,u,v,ll,f1,f2,f3,M,k,vp,j,MM,alpha,INCREASE,Mn;
# Decoding input
    #y := hc[1];
    Q := evala(Simplify(hc[3]),expanded);
    alpha:=hc[6];
    x := indets(Q)[1];
    p := hc[4];
    r:=hc[2];
```

```
    d:=degree(Q,x);
if (modp(r,p) = 0) then lprint("p divide r"); end if;
# genus and Precision
    g := (degree(Q, x)-1)*(r-1)/2;
    n:=hc[5];
    h:=hc[7];
    if h<>0 then T:=indets(h)[1];  end if;
    N := ceil(log[p](2^(2*g+1)*p^(n*g/2)));
    INCREASE:=3;
    #INCREASE IS TO BE ADJUSTED
    N1:=N+INCREASE;
for i from 1 by 1 while i <= (N1-1/r+log[p]((r+1)*i-1.0)) do
end do;
MM:=i:
lprint("N = ", N);
lprint("N1 = ", N1);
lprint("MM = ", MM);
lprint("g = ", g);

#action of Frobenius on alpha (=beta) using Hensel's Lemma
Falpha[1]:=alpha;
Dh:=diff(h,T);
for j from 1 to (n-1) do
```

```
   alphap:=evala(alpha^(p^j)) mod p;
   Dhe:=evala(eval(Dh,T=alphap)^(-1) mod p) mod p;
   Fa:=alphap;
     for i from 1 to (N1-1) do
        tt:=evala(simplify(expand((-eval(h,T=Fa)/p^i)*Dhe))) mod p;
        Fa:=Fa+tt*p^i;
     od;
   Falpha[j]:=Fa;
od;

#Frobenius of y^-i
lprint("Handling 1/y^sigma");
    t := sort(modp(evala(Simplify(evala((subs(alpha=Falpha[1],
subs(x=x^p, Q))-Q^p)*tauu^p)),expanded),p^N1), x);
    # tauu = 1/y^r
for i from 1 to (r-1) do
 sqt[i] := modp(convert(series((1+TT)^(-i/r), TT, MM+1), polynom),p^N1);
 yis[i] := sort(collect(evala(Simplify(evala(subs(TT=t, sqt[i])*
tauu^iquo(i*p,r))),expanded), tauu), tauu);
od;

# Preparation - Frobenius applied on differential basis
Qp := modp(diff(Q,x),p^N1);
```

```
Qd:=coeff(Q,x,d);
gg:=gcdex(Q, Qp, x, 'u', 'v');
u:=precision(u,N1,p,x,tauu,alpha);
v:=precision(v,N1,p,x,tauu,alpha);
vp:=precision(diff(v,x),N1,p,x,tauu,alpha);
if (gg <> 1) then lprint("GCD Q,Qp not 1"); end if;

# Frob of x^i*y^(-j)*dx
for j from 1 to (r-1) do
  lprint("Handling y=", j, "-th differential");
  ll:=irem(j*p,r);
    for i from 0 to (d-2) do
       lprint("Handling x=", i, "-th differential");
       t := modp(expand(p*x^(p*i+p-1)*yis[j]),p^N1);
       f[i,j] := sort(collect(t, tauu), tauu);
       f1[i,j] := Red1(f[i,j], tauu, Q, x,N1,p,alpha);
       f2[i,j] := Red2(f1[i,j], tauu, Q, x, r,Qp,u,v,vp,ll,N1,p,alpha);
       f3[i,j] := Red3(f2[i,j], Q, tauu, x,r,Qp,ll, d,Qd,N1,p,alpha);
    od;
od;

# Matrix of p-Frob
lprint("Computing matrix");
```

```
M:=Matrix(2*g);
for j from 1 to (r-1) do
 for i from 0 to (d-2) do
  for k from 0 to (d-2) do
    M[(d-1)*(irem(j*p,r)-1)+k+1,(d-1)*(j-1)+i+1] := coeff(f3[i,j], x, k);
  od;
 od;
od;

#Matrix of q-Frob
#To be faster here we could use a "square and multiply" kind of technique,
#using the binary expansion of n, as suggested by Kedlaya
Mn:=M;
for j from 1 to (n-1) do
    Mn:=matrixprecision(evala(Mn.subs(alpha=Falpha[j],M)),2*g,alpha,p,N1);
od;

# Characteristic polynomial using Newton's trace formula
M:=Mn;
CharCoeff:=Vector(2*g+1);
charCoeff[1]:=1;
Traces:=Vector(g);
for i from 1 to g do
```

```
Traces[i]:=evala(linalg[Trace](M));
CharCoeff[i+1]:=evala(-sum(Traces[j]*CharCoeff[i+1-j,j=1..i)/i);
CharCoeff[i+1]:=mods(charCoeff[i+1],p^N);
if i < g then M:=matrixprecision(evala(Mn.M),2*g,alpha,p,N1);; end if;
od;
for i from 1 to g do
CharCoeff[2*g+2-i]:=(p^(n*(g-i+1)))*CharCoeff[i];
od;

#Output Zeta function
t:='t';
ZetaNum:=0;
for i from 1 to (2*g+1) do
ZetaNum:=ZetaNum + CharCoeff[i]*t^(i-1);
od;
lprint("The Zeta function of the curve is:");
ZetaNum/((1-(p^n)*t)*(1-t));

end: # FullProgram
```

CHAPTER 3
The Zeta Function of Cab Curves via Deformation Theory

Another promising technique for computing the Zeta function of some curve is to place it in a family and study how the Frobenius action on the cohomology varies from one curve to another in that family. This has been suggested by Lauder, who was building on ideas of Dwork. It is actually a very general technique and curves are theoretically only a small subset of all varieties on which we can hope to use it. Far less work seems to have been done as of now in this direction, but there are already implementations. Ralf Gerkmann has done this for, among others, elliptic curves in [Ger05]. Also, Hubrechts has done an implementation for hyperelliptic curves [Hub07] which is available in MAGMA and which is really advantageous in certain situations. In this chapter we present a generalization of this algorithm to families of Cab curves. To our knowledge, this is the first time that this has been done. Both the algorithm and the subsequent analysis are inspired by that of [Hub07] and we will rely on [DV06] for convergence estimates.

3.1 Setting

We start with a curve $\bar{C}_t : \{\bar{f}(t,x,y) = 0\}$, $\bar{f} = y^a + \sum_{i=1}^{a-1} \bar{f}_i(x,t)y^i + \bar{f}_0(x,t) \in \mathbb{F}_q[t,x,y]$ such that $\deg_x \bar{f}_0 = b$ and $a \deg_x \bar{f}_i + bi < ab$ for $i = 1, \ldots, a-1$. Assume that $\bar{C}_0 : \{\bar{f}(0,x,y) = 0\}$ defines a Cab curve as in Definition 2.3 and that the coefficient of x^b does not depend on t. Let $\bar{\delta} \in \bar{\mathbb{F}}$ and take m such that $\mathbb{F}_q(\bar{\delta}) = \mathbb{F}_{q^m}$. We further assume that

$\bar{C}_{\bar{\delta}} : \{\bar{f}(\bar{\delta}, x, y) = 0\}$ is also a Cab curve (i.e. is non-singular), and that lifting the Frobenius (as done in Chapter 2) for the curve $\bar{C}_0$ is computationally cheaper than for the curve $\bar{C}_{\bar{\delta}}$. A case of particular interest is when $\bar{C}_0$ is a superelliptic curve, because in that situation we can, as we have seen, take away the points $\{y = 0\}$ and get a computationally much faster Frobenius lift. In that case we will be using Theorem 2.5 *iii)* to compare two a priori different lifts of Frobenius. Another case of interest is when m is sufficiently large, because then working over $\mathbb{Q}_q$ is much faster than over $\mathbb{Q}_{q^m}$.

As in Chapter 2, we start by lifting $\bar{f}$ to $f \in \mathbb{Z}_q[t, x, y]$. We take the resultant $R_1(t, y)$ of f and $\frac{\partial f}{\partial x}$ with respect to x, the resultant $R_2(t, y)$ of f and $\frac{\partial f}{\partial y}$ with respect to x, and finally the resultant $R_3(t)$ of R_1 and R_2 with respect to y. Since $\bar{C}$ is non singular for some values of t, we know that $R_3 \not\equiv 0$. We may factorize it and find the lowest-degree product of its factors elevated to some powers that lies in the ideal $< f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} >$; denote it $r(t)$. That means that we can find $\alpha, \beta, \tau \in \mathbb{Z}_q[t, x, y]$ such that

$$\alpha \frac{\partial f}{\partial x} + \beta \frac{\partial f}{\partial y} + \tau f = r. \tag{3.1}$$

We define $S := \mathbb{Q}_q \left[t, \frac{1}{r(t)}\right]^{\dagger}$, $A := \mathbb{Z}_q \left[t, \frac{1}{r(t)}, x, y\right] / (f)$, $T := A^{\dagger} \otimes_{\mathbb{Z}_q} \mathbb{Q}_q$ and

$$r_1(t, x, y) := \alpha^{\mathrm{F}}(t^p, x^p, y^p),\ r_2(t, x, y) := \beta^{\mathrm{F}}(t^p, x^p, y^p),\ r_3(t, x, y) := \tau^{\mathrm{F}}(t^p, x^p, y^p).$$

Hence from (3.1) we get

$$r_1 \left(\frac{\partial f}{\partial x}\right)^{\mathrm{F}} (t^p, x^p, y^p) + r_2 \left(\frac{\partial f}{\partial y}\right)^{\mathrm{F}} (t^p, x^p, y^p) + r_3 f^{\mathrm{F}}(t^p, x^p, y^p) = r^{\mathrm{F}}(t^p). \tag{3.2}$$

Now let $z_0 := r^{\mathrm{F}}(t^p)$; $\quad z_0 \equiv r(t)^p \pmod p \Longrightarrow z_0^{-1} \in A^\dagger$. Write $z_0 = r^p + pz(t) \Longrightarrow z_0^{-1} = \frac{1}{r^p}\sum_{i=0}^{\infty}(-1)^i\left(\frac{pz}{r^p}\right)^i$. Finally we let $h_i := z_0^{-1}r_i$ for $i = 1, 2, 3$ to have the desired equality

$$h_1\left(\frac{\partial f}{\partial x}\right)^{\mathrm{F}}(t^p, x^p, y^p) + h_2\left(\frac{\partial f}{\partial y}\right)^{\mathrm{F}}(t^p, x^p, y^p) + h_3 f^{\mathrm{F}}(t^p, x^p, y^p) = 1. \tag{3.3}$$

Now we are ready to lift the Frobenius $A/pA \to A/pA$ to $\mathrm{F} : A^\dagger \to A^\dagger$ as we did in Section 2.4. We will try with $t \mapsto t^p$ and assume $x \mapsto x^p + h_1 V$, $y \mapsto y^p + h_2 V$, $\quad V \equiv 0 \pmod p$. Again one lets $G(V) := f^{\mathrm{F}}(t^p, x^p + h_1 V, y^p + h_2 V)$ and easily sees that $G(0) \equiv 0$, $G'(0) \equiv 1 \pmod p$; hence we can find a solution V_0. The significance of this is seen by letting

$$\mathbb{S} := \left\{\omega \in \mathbb{C}_p \mid \omega \text{ is a Teichmuller lift of } \bar{\omega} \text{ and } \overline{r(\omega)} \neq 0\right\}.$$

Then let δ be the Teichmuller lift of $\bar{\delta}$; we have $0, \delta \in \mathbb{S}$ and for any $\omega \in \mathbb{S}$ substituting $\omega \to t$ in A and T gives structures as in Chapter 2 for the Cab curve $\bar{C}_{\bar{\omega}}$. Writing $\Omega^1_{T/S}$ for the differentials relative to S (in which $dt = 0$), we also have a derivative map $D : T \to \Omega^1_{T/S}$ where $D(\theta) = 0$ for any $\theta \in S$, and the S -module $H^1_{MW} := \Omega^1_{T/S}/D(T)$ has basis over S as in Lemma 2.10. Here is a rough outline of the theory; describing it at the level of details of Chapter 2 would be a big chapter by itself. We suggest to see [Dwo63] or Chapter 3 of [Ked] for more details. Reminiscent of Theorem 2.7, H^1_{MW} is canonically isomorphic to the relative algebraic deRham cohomology group $H^1_{dR}(C_t/\mathbb{S})$; hence it is a vector bundle on $\mathbb{S}$. For $\omega \in \mathbb{S}$, the fiber $H^1_{dR}(C_t/\mathbb{S})_\omega$ of this bundle can be identified with $H^1_{dR}(C_\omega)$. Denote by $F(t)$ the $(2g) \times (2g)$ matrix of the action induced on H^1_{MW} by the lift

of Frobenius. Our goal is to evaluate F at δ, which is the action of Frobenius on $H^1_{dR}(C_\delta)$ and hence will lead to the numerator of the Zeta function of $\bar{C_{\bar{\delta}}}$ by looking at the characteristic polynomial of $\hat{F} := \mathrm{F}^{mn-1}(F(\delta))\mathrm{F}^{mn-2}(F(\delta))\cdots\mathrm{F}(F(\delta))F(\delta)$. The big question is how does one find F?

The vector bundle $H^1_{dR}(C_t/\mathbb{S})$ is equipped with the Gauss-Manin connection $\bigtriangledown : H^1_{dR}(C_t/\mathbb{S}) \to H^1_{dR}(C_t/\mathbb{S}) \otimes_S \Omega^1_S$, which can be thought of as differentiating in t. By Berthelot (see [Ked], Theorem 3.6.4), $\bigtriangledown$ commutes with F. If $\{s_1, \ldots, s_{2g}\}$ denote the basis as in Lemma 2.10, we may write the connection matrix $C(t)$ to satisfy

$$\bigtriangledown(s_j) = \left(\sum_{i=1}^{2g} C_{ij}(t)s_i\right) \otimes dt.$$

Then this commutativity condition together with the Leibniz rule for connections are easily seen to imply

$$C(t)F(t) + \frac{d}{dt}F(t) = pt^{p-1}F(t)\mathrm{F}(C(t)). \tag{3.4}$$

As done in [Hub07] we will first compute a solution $J(t)$ to

$$\frac{d}{dt}J(t) = C(t)J(t),\ J(0) = I \tag{3.5}$$

and then (3.4) implies

$$F(t) = J(t)F(0)(\mathrm{F}(J(t)))^{-1}. \tag{3.6}$$

3.2 Details of the algorithm

In this section we present a commented MAGMA implementation. For simplicity, I assume that f is defined over $\mathbb{F}_p$, i.e. $n = 1$. However I do not assume $\bar{\delta} \in \mathbb{F}_p$, so that m might be greater than 1. For the analysis of the bounds involved (in Section 3.3) we will remove the assumption $n = 1$. It should be noted that in Section 3.3 we will work (mod p^N) for some N (that is, the number of digits used to store an element $\lambda \in \mathbb{Q}_q \backslash \mathbb{Z}_q$ is N plus $-v_p(\lambda)$; this was the method used in Section 2.5). However in the presentation of the following algorithm we will work with a mantissa of fixed length (i.e. the number of p-adic digits is fixed regardless of the valuation of the scalar stored). Computationally this is much simpler and seems to be much more used. We have observed in examples that the valuation of the scalars appearing in the computation does not become much less than zero even though there might be many operations of division by p that decrease the accuracy (we don't count additions and multiplications, which can increase the valuation), and hence in these examples the difference between both methods was minimal. However we stress that these are only empirical results which might not be valid for all examples; in which cases one could simply work over $\mathbb{Q}[z]/(g(z))$ "mod p^N " as we did in Section 2.5.

We start by presenting the initial setting of the algorithm.

```
Q:=RationalField();
Ptemp<Py,Px,Pt>:=PolynomialRing(Q,3);
f:=Py^3+Pt*Py^2-Px^4-Px^3-1;  // For example ..
```

```
// Initial constants
// The algorithm outputs the numerator of the zeta function of f at
//t=\bar{delta}, delta is a Teichmuller lift of \bar{delta} mod p^Nb
delta:=1;  // also for example.. p:=5;   // prime p
a:=IntegerRing()!Degree(f,Py); b:=IntegerRing()!Degree(f,Px);
c:=Max(a,b); m:=1;  // \bar{delta} \in F_{p^m}
g:=IntegerRing()!((a-1)*(b-1)/2); kappa:=IntegerRing()!Degree(f,Pt);
N0:=IntegerRing()!Ceiling(Log(p,2*Factorial(2*g)*p^(g*m/2)/Factorial(
g)^2));
theta:= Floor(Log(p,2*g + 2*a + 7*p*b/Log(p)) + 2*(a+b)/7) +
4*(a-1)*Floor(Log(p,2*a-1));
N8:= Floor((g)/(p-1))+(gm-1)*theta;
Nb:= N0+N8;
if not (Coefficient(f,Py,a) eq Ptemp!1) then print("We
assumed f=y^a+..."); end if;

// Find Resultant r(t)
fx:=Derivative(f,Px);
fy:=Derivative(f,Py);
ft:=Derivative(f,Pt);
R1:=Resultant(f,fx,2);
R2:=Resultant(fy,fx,2);
```

```
R3:=Resultant(R1,R2,1);
Fac:=Factorization(R3);
deg:=Fac[1][2]*Degree(Fac[1][1],Pt);
Resultant0:=Fac[1][1];
i:=2;
while deg lt Degree(R3,Pt) do
deg:=deg+Fac[i][2]*Degree(Fac[i][1],Pt);
Resultant0 := Resultant0*Fac[i][1];
i:=i+1;
end while;

// to get an integral resultant (i.e. that clear p in the denom of the
coefficients)
lcm:=1;
for i in [0..Degree(Resultant0,Pt)] do
  lcm:=LeastCommonMultiple(Denominator(Q!Coefficient(Resultant0,Pt,i)),
lcm);
end for;
//lcm:=(lcm/p^Valuation(lcm,p));
Resultant0:=Resultant0*lcm;
if Valuation(lcm,p) gt 0  then print("Resultant not monic");  end if;

Inullst:=IdealWithFixedBasis([f,fx,fy]);
```

```
if not (Resultant0 in Inullst) then print("Multiply Resultant0 by
factors in Fac until Resultant0 lie in Inullst"); end if;
Coord:=Coordinates(Inullst,Resultant0);
// Coord[1]*f+Coord[2]*fx+Coord[3]*fy-Resultant0 eq Ptemp!0

// Computation constants
rho:=Degree(Resultant0,Pt);
phi:=Max(Degree(Coord[2],Pt),Degree(Coord[3],Pt));
tau1:=Floor(Nb*p*(rho+kappa+phi)*(1+(a+b-2)*((phi*c+kappa)/rho + kappa/
(rho+kappa+phi))));
tau2:=2*Nb*p*(1+(a+b-2)*c)-4*p*(a+b-2)*c;
tau3:=7*p*b*a*Nb + 2*p*b*(a+b)+2;
M:=tau2+kappa*tau3;
Nt:=tau1+rho*kappa*tau3+rho*M;
N6:=(2*g*theta+g)*(Ceiling(Log(p,Nt))+Ceiling(Log(p,Nt/p+1)));
Nc:=Nb+N6;
N3:=(4*g*theta+2*g+1)*Ceiling(Log(p,Nt));
Na:=Nc+N3;

F<t>:=FunctionField(Q);   // and put Resultant0 in F
Resultant:=F!0;
for i in [0..Degree(Resultant0,Pt)] do
  Resultant:= Resultant+(Q!Coefficient(Resultant0,Pt,i))*t^i;
```

```
end for;
```

Let's now compute the connection matrix C of $\bigtriangledown : H^1_{dR}(C_t/\mathbb{S}) \to H^1_{dR}(C_t/\mathbb{S}) \otimes_S \Omega^1_S$. We are actually going to compute $N(t) := r(t)C(t)$ to clear denominators in t in the connection matrix, i.e. get entries in $\mathbb{Q}[t]$. This is possible since the initial assumptions on f imply that $H^1_{dR}(C_\omega)$ and hence $H^1_{dR}(C_t/\mathbb{S})_\omega$ has the same generic basis at all non-singular fibers $\omega \in \mathbb{S}$. Then by linearity of the ODE (3.5) we will solve for $E(t)$ in $r(t)\frac{dE}{dt} = NE$. First the setting:

```
Ptemp2<Pyy,Pxx>:=PolynomialRing(F,2);

// Put f in Ptemp2, called f2
f2:=Ptemp2!0;
for i in [0..a] do    for j in [0..Degree(Coefficient(f,Py,i),Px)] do
  for k in [0..Degree(Coefficient(Coefficient(f,Py,i),Px,j),Pt)] do
  f2:=f2+(Q!Coefficient(Coefficient(Coefficient(f,Py,i),Px,j),Pt,k))*
Pyy^i*Pxx^j*t^k;
end for;  end for;   end for;
I:=Ideal([f2]);
P<y,x>,toP:=Ptemp2/I;

// Put Coord[2] in P
```

```
A:=Ptemp2!0;
for i in [0..Degree(Coord[2],Py)] do
for j in [0..Degree(Coefficient(Coord[2],Py,i),Px)] do
for k in [0..Degree(Coefficient(Coefficient(Coord[2],Py,i),Px,j),Pt)]
do
  A:=A+(Q!Coefficient(Coefficient(Coefficient(Coord[2],Py,i),Px,j),
Pt,k))*Pyy^i*Pxx^j*t^k;
end for;  end for;   end for;
A:=toP(A);

// Put Coord[3] in P
B:=Ptemp2!0;
for i in [0..Degree(Coord[3],Py)] do
for j in [0..Degree(Coefficient(Coord[3],Py,i),Px)] do
for k in [0..Degree(Coefficient(Coefficient(Coord[3],Py,i),Px,j),Pt)]
do
  B:=B+(Q!Coefficient(Coefficient(Coefficient(Coord[3],Py,i),Px,j),
Pt,k))*Pyy^i*Pxx^j*t^k;
end for;  end for;   end for;
B:=toP(B);

// Put ft in Ptemp2
ft2:=Ptemp2!0;
```

```
for i in [0..Degree(ft,Py)] do
for j in [0..Degree(Coefficient(ft,Py,i),Px)] do
for k in [0..Degree(Coefficient(Coefficient(ft,Py,i),Px,j),Pt)] do
  ft2:=ft2+(Q!Coefficient(Coefficient(Coefficient(ft,Py,i),Px,j),Pt,
k))*Pyy^i*Pxx^j*t^k;
end for;  end for;   end for;

// toP(fx2)*A+toP(fy2)*B-Resultant) eq P!0
fx2:=Derivative(f2,Pxx);
fy2:=Derivative(f2,Pyy);

// sum function
Summ :=function(w)
   L:=0;
   for i in [1..#w] do
    L:=L+w[i];
   end for;
return L;
end function;

//MaxOrder as defined in Denef/Vercauteren for Cab curves
MaxOrder :=function(w)
M:=-1;
```

```
for k in [1..#Terms(w)] do
if And([Exponents(Terms(w)[k])[2] gt (b-2)],[M lt
a*Exponents(Terms(w)[k])[2] + b*Exponents(Terms(w)[k])[1]])[1] then
    M:=a*Exponents(Terms(w)[k])[2] + b*Exponents(Terms(w)[k])[1];
    n:=k;
end if;
end for;
i := Exponents(Terms(w)[n])[2];
j := Exponents(Terms(w)[n])[1];
return i,j;
end function;

//coefficient of x^i*y^j
Coef := function(w,i,j,FF)
L:=FF!0;
if Degree(w,2) ge i then
if Degree(Coefficients(w,x)[i+1],1) ge j then
L:=FF!Coefficients(Coefficients(w,x)[i+1],y)[j+1];
end if;
end if;
return L;
end function;
```

```
//since g(x)dx congruent to 0 in H^1
killx:=function(w)
w:=w-Coefficients(w,y)[1];
return w;
end function;

// Denef/Vercauteren 's reduction (algorithm 3)
// say f = sum (over 0<=k<=a) f_k * y^k
//put ff[k+1]=f_k ,  df[k+1]=d(f_k)/dx  in their notation (or look
// below to see it defined)
Reduction:=function(w,ff,df,FF)
w:=killx(w);
while Degree(w,2) gt (b-2) do
i,j := MaxOrder(w);
l := i - b + 1;
Delta_x := (Summ([(j/(k+j))*df[k+1]*y^k : k in [0..(a-1)]]));
//not exactly same Deltax as in paper
Delta_y := -l*((a/(a+j))*y^a + Summ([(k/(k+j))*ff[k+1]*y^k : k in
[1..(a-1)]]));
if l gt 0 then Delta := killx(y^j*x^(l-1)*(x*Delta_x + Delta_y));
else Delta:=killx(y^j*(Delta_x)); end if;
gamma := Coef(Delta, i,j,FF);
nu := Coef(w,i,j,FF);
```

```
w:=w-nu*(gamma^-1)*Delta;
end while;
return w;
end function;
```

Now from $0 \equiv d\left(\frac{1}{(j+1)}x^i y^{(j+1)}\right) = x^i y^j dy + \frac{i}{j+1}x^{(i-1)}y^{(j+1)}dx$ we get the function dytodx, which outputs cdx congruent to wdy in H^1_{MW}. Also in $\Omega^1_{T/S}$ we have $0 = df = f_x dx + f_y dy + f_t dt$, hence $r(t)dy \wedge dx \equiv ((-Af_t)dy + Bf_t dx) \wedge dt$ and more generally $r(t)\nabla(x^i y^j dx) \equiv jx^i y^{j-1}((-Af_t)dy + Bf_t dx) \wedge dt$, where A and B are defined in the program and are analogous to respectively α and β in (3.1). From the latter relation we get the function Buildconnection.

```
dytodx :=function(w)
c:=0;
for i in [1..Degree(w,2)] do
 wi:=Coefficients(w,x)[i+1];
  for j in [0..Degree(wi,1)] do
        c:=c+Coefficients(wi,y)[j+1]*(-i/(j+1))*x^(i-1)*y^(j+1);
  end for;
end for;
return c;
end function;

//DDx,DDy,ff,df  defined below
```

```
Buildconnection :=function(DDx,DDy,ff,df,g,FF)
M:=Matrix(FF,2*g,2*g,[]);
for yval in [1..(a-1)] do
   for xval in [0..(b-2)] do
w:=Reduction(dytodx(yval*x^xval*y^(yval-1)*DDy)+yval*x^xval*y^(yval-
1)*DDx,ff,df,FF);
      for yvalp in [1..(a-1)] do
         for xvalp in [0..(b-2)] do
M[(yvalp-1)*a+xvalp+1,(yval-1)*a+xval+1]:=Coef(w,xvalp,yvalp,FF);
         end for;
      end for;
   end for;
end for;
return Matrix(FF,M);
end function;

// Let's now actually compute the connection
DDy:=-A*toP(ft2);
DDx:=B*toP(ft2);
ff:=[toP(Coefficients(f2,Pyy)[k+1]): k in [0..a]];
//That is ff[k+1]=f_k
df:=[toP(Derivative(Coefficients(f2,Pyy)[k+1],Pxx)) : k in [0..a]];
//That is df[k+1]=d(f_k)/dx
```

```
N:=Buildconnection(DDx,DDy,ff,df,g,F);

//Check that there is no denominator in N
for i in [1..(2*g)] do  for j in [1..(2*g)] do
   if not (Denominator(N[i,j]) eq F!1) then print("The connection has
denominators, you will need to multiply the ODEs by their lcm");
end if;
end for;  end for;
```

If f at $t = 0$ is a superelliptic curve, as we mentioned above we want to find $F(0)$ by "removing the points $\{y = 0\}$" and computing as in Section 2.5. This computation is done in the basis $dx/y, xdx/y, x^2dx/y, dx/y^2, \ldots$ and we then have to write $F(0)$ in the suggested Cab basis $ydx, xydx, x^2ydx, y^2dx, \ldots$

For this purpose (in the program's notation and with $y^a = -f0$), we find $BB * (-f0) + CC * (-Diff(f0, x)) = 1$ and look at $0 \equiv d((a/(a - yval)) * CC * x^{xval} * y^{(a-yval)})$: then in H^1_{MW}

$$\frac{x^{xval}dx}{y^{yval}} \equiv (x*BB-(a/(a-yval))*(x*CCp+xval*CC))*x^{(xval-1)}*y^{(a-yval)}dx. \quad (3.7)$$

```
Ptemp2<Pyy,Pxx>:=PolynomialRing(Q,2);

// look at f at t=0 and put f in Ptemp2 (labelled f2)
fat0:=Evaluate(f,Pt,0);
```

```
if not ((fat0 - Py^a - Coefficient(fat0,Py,0)) eq Ptemp!0) then
print("f is not superelliptic at t=0");  end if;
if not (b eq Degree(fat0,Px)) then print("f at t=0 doesn't have
degree in x equal to b, which changes the genus!");  end if;
f2:=Ptemp2!0;
for i in [0..a] do for j in [0..Degree(Coefficient(fat0,Py,i),Px)] do
  f2:= f2+(Q!Coefficient(Coefficient(fat0,Py,i),Px,j))*Pyy^i*Pxx^j;
end for;  end for;

I:=Ideal([f2]);
P<y,x>,toP:=Ptemp2/I;
// Here one has to reload (with new x and y defined) MaxOrder, Coef,
// killx and Reduction

// With y^a=-f0, find BB and CC s. th. BB*(-f0)+CC*(-Diff(f0,x))=1
f0:=f2-Pyy^a;
f0x:=Derivative(f0,Pxx);
Inullst:=IdealWithFixedBasis([f0,f0x]);
if not (1 in Inullst) then print("Curve singular at 0"); end if;
Coord:=Coordinates(Inullst,Ptemp2!-1);
if not ((Coord[1]*f0+Coord[2]*f0x+1) eq Ptemp2!0) then
print("Something wrong with Coord in change of basis");  end if;
BB:=Coord[1];
```

```
CC:=Coord[2];
CCp:=Derivative(CC,Pxx);

Buildchangeofbasis :=function(BB,CC,CCp,ff,df,g,FF)
M:=Matrix([[P!0 : i in [1..(2*g)]]: j in [1..(2*g)]]);
for yval in [1..(a-1)] do
   for xval in [0..(b-2)] do
      if xval gt 0 then
L:=(x*BB-(a/(a-yval))*(x*CCp+xval*CC))*x^(xval-1)*y^(a-yval); else
L:=y^(a-yval)*(BB-(a/(a-yval))*CCp); end if;
      w:=Reduction(L,ff,df,FF);
      for yvalp in [1..(a-1)] do
         for xvalp in [0..(b-2)] do
  M[(yvalp-1)*a+xvalp+1,(yval-1)*a+xval+1]:=Coef(w,xvalp,yvalp,FF);
         end for;
      end for;
   end for;
end for;
return Matrix(FF,M);
end function;

ff:=[toP(Coefficients(f2,Pyy)[k+1]): k in [0..a]];
//That's ff[k+1]=f_k
```

```
df:=[toP(Derivative(Coefficients(f2,Pyy)[k+1],Pxx)) : k in [0..a]];
//That's df[k+1]=d(f_k)/dx
ynegtoypos:=Buildchangeofbasis(toP(BB),toP(CC),toP(CCp),ff,df,g,Q);
ypostoyneg:=ynegtoypos^-1;
```

Although it is not relevant for the current algorithm, it can be useful in practice to change the Connection matrix from a basis to another. We would like to mention in passing how this is done. Say $C = N/r$ is the Connection matrix in the suggested Cab basis and T is a change of basis matrix from some basis to the Cab basis (for example $T =$ ynegtoypos in this program). Then one can compute that $T^{-1}\left(CT - \frac{dT}{dt}\right)$ is the Connection matrix in that other basis. This, for instance, allows to compare the present work with that in [Ger05], [Hub07] and observe that C does not seem to depend on the various choices made in the different algorithms.

Now back to the current algorithm, at this point we need to find the Frobenius matrix Frob0, mod p^{Nc}, of the curve f at $t = 0$. If the curve at the fiber $t = 0$ is not superelliptic, this is done using [DV06]. Note that it is important to be consistent in lifting the coefficient of the curve from elements of $\mathbb{F}_p$ to elements of $\mathbb{Z}_p$ as this does change the matrix $F(0)$. In the present case one would use a naive lift of -f0.

It is now time to solve Equation (3.5) mod t^{Nt}. In order to compute $\mathrm{F}(J)^{-1} = \mathrm{F}(J^{-1})$ we will solve $r(t)\frac{dE}{dt} = -EN \pmod{t^{\lceil Nt/p \rceil}}$, whose solution gives J^{-1}, and then map $t \mapsto t^p$ (since we are working over $\mathbb{Q}_p$). The rest is the same as solving for J, hence we will omit that function.

```
Qp:=pAdicField(p,Na);
Frob0:=Matrix(Qp,Frob0);
Qpt<t>:=PowerSeriesRing(Qp);
h:=Qpt!Resultant;    // h(t) here corresponds to r(t) in Section 3.1
NN:=Matrix(Qpt,N);

//algorithm for finding J, i.e. solving hdE/dt=NE  mod t^Nt
ODEEbasis := function(N,h,g,bound,p,Qp,Qpt)
//h:=Resultant;   in a free precision power series ring in t
// N = ConnectionMatrix*h  in a free precision power series ring in t
Mat := Matrix(Qp,2*g,2*g,[]);
E := [Mat : k in [0..bound]];
E[1] := ScalarMatrix(2*g,Qp!1);
mm:=0;
for i in [1..(2*g)] do   for j in [1..(2*g)] do
      if not (N[i,j] eq 0) then
      mm:=Max(Degree(N[i,j]),mm);   end if;
end for;   end for;
print(mm);
mmm:=Degree(h);
hh:=[Coefficient(h,k):k in [0..mmm]];
NN := [Mat : k in [0..mm]];
```

```
for u in [0..mm] do  for i in [1..(2*g)] do   for j in [1..(2*g)] do
     NN[1+u][i,j]:=Coefficient(N[i,j],u);
end for;   end for;   end for;
for v in [0..(bound-1)] do
L:=ZeroMatrix(Qp,2*g,2*g);
for u in [0..Min(v,mm)] do
L:=L+NN[1+u]*E[1+v-u];
end for;
for s in [1..Min(mmm,v)] do
  L:=L+E[v+2-s]*(hh[1+s]*(s-v-1));
end for;
for i in [1..(2*g)] do  for j in [1..(2*g)] do
E[v+2][i,j]:=Expand((1/((v+1)*hh[1]))*L[i,j]);
end for;   end for;
end for;
EE:=Matrix(Qpt,2*g,2*g,[]);
for i in [1..(2*g)]  do   for j in [1..(2*g)] do
   EE[i,j]:=Qpt![E[u+1][i,j] : u in [0..bound]];
end for; end for;
return EE;
end function;

Qpt<t>:=PowerSeriesRing(Qp,Nt);
```

```
C:=ODEEbasis(NN,h,g,Nt,p,Qp,Qpt);
FrobCinv:=ODEEinverse(NN,h,g,Nt,p,Qp,Qpt);
```

We are now ready to compute $F(t)$ as in Equation (3.6). It can be shown that the entries of $F(t)$ converge on a disc of radius strictly less than 1; hence we cannot evaluate $F(t)$ at $t = \delta$ directly. The way to get around this obstacle is to perform an analytic continuation, which can be thought of as clearing the denominators in r that have appeared during the computation.

```
//finding Frob(t)
Frob0:=Matrix(Qp,Frob0);
Ft:=C*Frob0*FrobCinv;

// Analytic continuation
Qp:=pAdicField(p,Nb+theta);
Qpt<t>:=PowerSeriesRing(Qp,Nt);
Ft:=Matrix(Qpt,Ft);
Qptpoln<Z>:=PolynomialRing(Qp);
hpoln:=Qptpoln!Coefficients(h);
hmpoln:=hpoln^M;
hm:=Qpt!hmpoln;
for i in [1..(2*g)] do   for j in [1..(2*g)] do
Ft[i,j]:=Ft[i,j]*hm;
```

```
end for;   end for;

Qp:=pAdicField(p,Nb);
Qpt<t>:=PowerSeriesRing(Qp,Nt);
Ft:=Matrix(Qpt,Ft);

//Evaluate the Frobenius matrix at t=delta and take its m^{th} power
F1 := Matrix(Qp,2*g,2*g,[]); R1 :=

Qp!(Q!Evaluate(Resultant0,Pt,delta)); R1M := 1/R1^M;
for i in [1..(2*g)] do  for j in [1..(2*g)] do
   F1[i,j] := Evaluate(Ft[i,j],delta) * R1M;
end for; end for;
//Because the Frob on Qp is trivial & delta is a Teichmuller lift:
F1:=F1^m;
```

Finally the last step, we compute the characteristic polynomial of F1 via Newton Trace Formula as in Harrisons' MAGMA implementation of Kedlaya's algorithm. Using Equation (2.2) we get the sequence of the coefficients corresponding to $x^{g+1}, \ldots, x^{2g}$ of the characteristic polynomial of F1 using the trace method. The MAGMA intrinsic function could be used but this is slightly more efficient as only Trace$(F1)$, Trace$(F1^2)$, $\ldots$, Trace$(F1^g)$ need be calculated rather than Trace$(F1)$, $\ldots$, Trace$(F1^{2g})$.

```
function UpperCoeffs(MM,g,ppow,e_val)
    // The sequence [r(v)] is returned where, for a p-adic int v,
    // r(v) is the integer nn s.t.|nn|<ppow/2 and nn=v mod ppow.
    pAd := pAdicField(Parent(MM[1,1]));
    N := MM;
    UCs := [pAd!1];   // coeffs (highest to lowest) of CharPoly(MM)
    TrPows := [pAd|]; // [Trace(MM),Trace(MM^2),...]
    for i in [1..g] do
        Append(~TrPows,Eltseq(Trace(N))[1]);
        Append(~UCs, (- &+[TrPows[j]*UCs[i+1-j] : j in [1..i]])/i);
        if i lt g then N := N*MM; end if;
    end for;
    if Nrows(MM) ne 2*g then // original Q(x) of even degree
        for i in [1..g] do
            UCs[i+1] := UCs[i+1]+e_val*UCs[i];
        end for;
    end if;
    return [((2*uc gt ppow) select uc-ppow else uc) where uc is
              (IntegerRing()!x) mod ppow : x in UCs];
end function;

UCoeffs := UpperCoeffs(F1,g,p^N0,1);
```

```
CharP := PolynomialRing(IntegerRing())!([UCoeffs[i]*p^(g+1-i): i in
[1..g]] cat Reverse(UCoeffs));
f0 := Parent(CharP)!Reverse(Coefficients(CharP));

// Finally the numerator of the Zeta function of f at t=\bar{delta}
print(f0);
```

3.3 Correctness of chosen bounds

The analysis will follow rather closely that of [Hub07]; instead of rewriting everything in our situation, we will generally focus on the differences and omit the similarities. The way we will proceed somewhat follows [DV06] but working over $S := \mathbb{Q}_q[t, \frac{1}{r(t)}]^\dagger$ instead of over $\mathbb{Q}_q$. The following Lemma from [Hub07] will be used repeatedly; it basically allows to reduce our situation to that of [DV06]. Let v_p of a polynomial be the minimum valuation of its coefficients.

Lemma 3.1. *Let* $s(t) = \sum_{k\in\mathbb{Z}} \frac{d_k(t)}{r(t)^k} \in S$ *such that deg* $d_k <$ *deg* r *for all* k. *Suppose we have for infinitely many* $\delta \in S$ *that* $v_p(s(\delta)) \geq \epsilon$ *for some real number* ϵ, *then also for every* $k \in \mathbb{Z}$ *we get* $v_p(d_k) \geq \epsilon$.

With the notation as in Section 3.1, let $\phi := \max\{\deg_t(\alpha), \deg_t(\beta)\}$, $\kappa := \deg_t f$, $c := \max\{a, b\}$ and $\rho := \deg r$. It is easy to see that the highest total $x - y$ degree of the monomials of f is c. One could find upper bounds for ρ using properties of the Resultant and for ϕ using [Kol88]; this would be useful to estimate the complexity of the algorithm but not for the algorithm itself.

Applying the Weil conjectures to $\bar{C}_{\bar{\delta}}$, we need to know the characteristic polynomial of $\hat{F}$ modulo $N_0 := \left\lceil \log_p \left(2\binom{2g}{g} p^{mng/2} \right) \right\rceil$.

Now we need to estimate the p-adic precision loss during the algorithm. Suppose that $A, B \in \mathbb{Q}_q$ (or A, B are matrices with entries in $\mathbb{Q}_q$) known up to p^{Prec_A}, p^{Prec_B} and $v_p(A) \geq Denom_A$, $v_p(B) \geq Denom_B$. Then $v_p(AB) \geq Denom_A + Denom_B$ and AB is known up to $p^{\min\{Prec_A + Denom_B, Prec_B + Denom_A\}}$.

Working backwards, the first goal is to have a lower bound $-\theta$ on the p-adic valuation of the coefficients of entries of the Frobenius matrix $F(\delta)$. Using Corollary 1 in [DV06] we have a lower bound on the valuation of the coefficient in S of the monomials in $x - y$ of the entries of the matrix, and using Lemma 4 in the same article we have an upper bound on the powers of p in the denominator introduced while reducing the differentials. The task is then to find

$$\min_{0 \leq i \leq 7pbN_b + 2pb(a+b)} \left\{ \frac{i}{7pb} - \frac{2(a+b)}{7} - \lfloor \log_p((i+1)a + (a-1)b) \rfloor - 4(a-1)b \lfloor \log_p(2a-1) \rfloor \right\}$$

An easy calculation shows that

$$\theta = \left\lfloor \log_p \left(2g + 2a + \frac{7pb}{\ln p} \right) + \frac{2(a+b)}{7} \right\rfloor + 4(a-1) \lfloor \log_p(2a-1) \rfloor$$

is sufficient.

In particular, while taking $\hat{F} := \mathrm{F}^{mn-1}(F(\delta))\mathrm{F}^{mn-2}(F(\delta)) \cdots \mathrm{F}(F(\delta))F(\delta)$ we may lose at most $(mn-1)\theta$ p-adic digits, and $v_p(\hat{F}) \geq -mn\theta$. Then computing $\det(1 - \hat{F}T)$ using Newton formula we must take $\hat{F}^g$ and divide by at most $(g)!$; hence loosing at most $\left\lfloor \frac{g}{p-1} \right\rfloor + (g-1)mn\theta$ digits. This explains why $N_8 := \left\lfloor \frac{g}{p-1} \right\rfloor + (gmn-1)\theta$. Because $v_p(\delta) = 0$, $v_p(r(\delta)) = 0$, $v_p(r(t)) = 0$ it is sufficient

to know $r(t)^M F(t) \pmod{p^{N_b}}$ where $N_b := N_0 + N_8$ and hence we need $F(t) \pmod{p^{N_b}}$ and $r(t) \pmod{p^{N_b+\theta}}$.

To bound the powers of t and $\frac{1}{r}$ that will occur in $F(t) \pmod{p^{N_b}}$, we will theoretically solve for V using a fixed point iteration with the Taylor expansion of $G(V)$ about 0. We would like to thank Ralf Gerkmann for suggesting this; the trick actually goes back to Monsky and Washnitzer.

$$\sum_{i=1}^{c} b_i V^i = \sigma \quad \text{where } b_i := \frac{G^{(i)}(0)}{i!},\ \sigma := -G(0). \tag{3.8}$$

Since $b_1 \equiv 1 \pmod{p}$, we may rewrite (3.8) as

$$U + \sum_{i=2}^{c} \hat{b_i} U^i = \sigma \quad \text{where } \hat{b_i} := b_1^{-i} b_i. \tag{3.9}$$

Working in the ring $\mathbf{Z}[\hat{b_2}, \ldots, \hat{b_c}]$ with the $\hat{b_i}$ seen as indeterminates, Monsky and Washnitzer [MW68] proved that there is a unique solution $U = \sigma + \sum_{i \geq 2} d_{U,i} \sigma^i \in \mathbf{Z}[\hat{b_2}, \ldots, \hat{b_c}][[\sigma]]$ to (3.9) and that $\deg d_{U,i} \leq i - 1$. Since $\sigma \equiv 0 \pmod{p}$, we know that the degree in $\hat{b_i}'s$ of the formal solution $\hat{U} := U \pmod{p^{N_b}}$ is at most $N_b - 2$.

Next write $h_i = \sum_{\omega} c_{h_i,\omega} v^\omega$ where $v = (x, y, t, \frac{1}{r})$. Looking at the expansion for z_0^{-1} we get that $v_p(c_{h_i,\omega}) \geq \frac{\omega_3}{\rho p} - \frac{\phi}{\rho}$, $v_p(c_{h_i,\omega}) \geq \frac{\omega_4}{p} - 1$. These estimates allow us to find $v_p(c_{b_i,\omega}) \geq \frac{\omega_3}{p\rho} - \frac{\phi i + \kappa}{\rho}$, $v_p(c_{b_i,\omega}) \geq \frac{\omega_4}{p} - i$ and (since $b_1 \equiv 1 \pmod{p}$) $v_p(c_{b_1-1,\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)}$, $v_p(c_{b_1-1,\omega}) \geq \frac{\omega_4}{2p}$. Thus $b_1^{-1} = \sum_{k=0}^{\infty} (1 - b_1)^k$ also satisfies $v_p(c_{b_1^{-1},\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)}$, $v_p(c_{b_1^{-1},\omega}) \geq \frac{\omega_4}{2p}$, which in turn imply $v_p(c_{\hat{b}_i,\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)} -$

$\frac{\phi i+\kappa}{\rho}$, $v_p(c_{\hat{b}_i,\omega}) \geq \frac{\omega_4}{2p} - i$. Finally we can write $v_p(c_{\sigma,\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)} - \frac{\kappa}{\rho+\kappa+\phi}$, $v_p(c_{\sigma,\omega}) \geq \frac{\omega_4}{2p}$.

Using the above-mentioned result of Monsky and Washnitzer, we see that $v_p(c_{d_{U,i},\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)} - (i-1)\frac{\phi c+\kappa}{\rho}$, $v_p(c_{d_{U,i},\omega}) \geq \frac{\omega_4}{2p} - (i-1)c$, and therefore $v_p(c_{\hat{U},\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)} - (N_b-2)\frac{\phi c+\kappa}{\rho} - (N_b-1)\frac{\kappa}{\rho+\kappa+\phi}$, $v_p(c_{\hat{U},\omega}) \geq \frac{\omega_4}{2p} - (N_b-2)c$. Since $\hat{V} = b_1^{-1}\hat{U}$ these estimates yield $v_p(c_{\mathrm{F}(x)^i\mathrm{F}(y)^j d(\mathrm{F}(x)),\omega}) \geq \frac{\omega_3}{p(\rho+\kappa+\phi)} - (N_b-2)(i+j+1)\frac{\phi c+\kappa}{\rho} - (N_b-1)(i+j+1)\frac{\kappa}{\rho+\kappa+\phi}$, $v_p(c_{\mathrm{F}(x)^i\mathrm{F}(y)^j d(\mathrm{F}(x)),\omega}) \geq \frac{\omega_4}{2p} - (i+j+1)(N_b-2)c$; so while lifting the Frobenius action mod N_b, we only expect powers of t less than $\tau_1 := N_b p(\rho+\kappa+\phi)(1+(a+b-2)(\frac{\phi c+\kappa}{\rho} + \frac{\kappa}{\rho+\kappa+\phi}))$ and power of $\frac{1}{r}$ less than $\tau_2 := 2N_b p(1+(a+b-2)c) - 4p(a+b-2)c$.

We then have to take into account the reduction process. Keeping track of the parameter t in Algorithm 3 of [DV06] (working over S instead of over $\mathbb{Z}_q$) and looking at the pole order of γ^{-1} at the zeroes of γ (γ defined at line 3.6 of Algorithm 3), we get that each loop of the algorithm introduces at most $t^{\rho\kappa}$ and $(\frac{1}{r})^\kappa$. Also, from the estimates in [DV06] on the maximum degree in x appearing in the image mod N_b of the chosen basis of differential elements under F and the fact that each loop reduces the pole order at infinity (in $x-y$) of the differentials, we see that there will be at most $\tau_3 := 7pbaN_b + 2pb(a+b) + 2$ loops done. This explains why $M = \tau_2 + \kappa\tau_3$ and $N_t = \tau_1 + \rho\kappa\tau_3 + \rho M$.

We now have results similar to that of Proposition 17 in [Hub07]. We can use Lemma 19 of [Hub07] to get $v_p(F(0)^{-1}) \geq -(2g-1)\theta - g$, and working through Proposition 21 we get that while computing $F(t) = C(t)F(0)(\mathrm{F}(C(t)))^{-1}$ we may lose at most $N_6 = (2g\theta+g)\left(\lceil \log_p(N_t)\rceil + \left\lceil \log_p\left(\frac{N_t}{p}+1\right)\right\rceil\right)$. So we need to know

$F(0)$ at least to precision $N_c = N_b + N_6$. Finally we bound the precision loss while computing C and $\mathrm{F}(C)^{-1}$. From Lemma 22 still in [Hub07] we get that both (independently done) computation will lose at most $N_3 = (4g\theta + 2g + 1)\lceil \log_p(N_t) \rceil$. That explains why we start the algorithm with $N_a = N_c + N_3$.

3.4 The experiments

It is hard to have a meaningful comparison between solving directly for the Zeta function of f_δ using the algorithm in [DV06] and using this algorithm because the predicted bounds are really not tight. For example, with $f := y^3 + ty^2 - x^4 - x^3 - 1,\ p = 5,\ \delta = 1$ we have a working precision of 42 using [DV06] and bounds $Nt = 217820$, $M = 17522$, $Na = 2361$, $Nb = 28$, $Nc = 1153$ for the present algorithm. Even on a good computer both represent several hours of computation, and it is possible that the computer actually runs out of memory before terminating.

By using a method of trial-and-error, we can see that 8 is the minimal working precision for the Cab algorithm [DV06] that will output the correct Zeta function,

$$Z(\bar{C}_1, T) = \frac{125T^6 - 25T^5 + 10T^4 + 117T^3 + 2T^2 - T + 1}{(1 - 5T)(1 - T)}.$$

Running Ralf Gerkmann's MAGMA implementation of [DV06] at that p-adic precision takes 112 seconds and 29MB. We were using a 3.2GHz machine with 3 GB of RAM memory.

As a comparison, trying the present algorithm with $Nt = 200$, $M = 25$, $Na = 15$, $Nb = 6$, $Nc = 6$ gives the same Zeta function in 8 seconds, and does so using

only 5MB! And this is a toy example with $m = 1$, so the advantage of the present algorithm could be very significant.

CHAPTER 4
Conclusion

We looked at two ways of computing the Zeta function via p-adic cohomology of some types of curves. It is in the author's opinion a marvelous example of however abstract and theoretical they can be, theories can sometimes be used to solve fairly concrete problems. The second option discussed, using the Gauss-Manin connection, yielded a new implementation. We conclude with things that are left open about it.

4.1 Possible improvements and future investigation

One obvious generalization of the algorithm presented in Section 3.2 is to include the case where $n > 1$. This would be a fairly easy thing to do: one would work in the unrammified degree n extension as we did in Section 2.5.

A major drawback of the algorithm as it stands now is that the chosen bounds are not practical. This is not so surprising: even though the lift of Frobenius had a closed form (hence was much easier to analyze), already Hubrechts ran into similar problems while implementing [Hub07]. In the present case, the lift of Frobenius is more complicated, and constantly going for the worst case scenario produces bounds that are much too big. For example, we know that the degree in $\hat{b_i}'s$ of the formal solution $\hat{U} := U \pmod{p^{N_b}}$ is at most $N_b - 2$, but we certainly do not expect to have an occurrence of $\hat{b_c}^{N_b-2}$ as c and N_b get large. One could possibly address this issue by looking at probabilistic bounds, but at first sight that does not seem very implementable. Hubrechts in his implementation adresses the

issue in a more conventional way: he makes a loop on the bounds involved starting with some relatively low values and increasing until some criterion is met. We have experimented such a solution in our numerical experiments and it turned out very useful. The hard thing to do is to "guess" ratios that will make the algorithm work, because computing with very high bounds without satisfactory ratios among them does not lead to the correct Zeta function.

Finally, it would be great to have complexity estimates for the algorithm. The methods suggested to bound ρ and ϕ seemed to also give very large bounds compared to what has been observed in our experiments. This, combined with the worst case scenario analysis provided in Section 3.3, would give a complexity analysis (at least the rough size of the constants) that would not give justice to the potential of the algorithm. For this reason, we should probably omit it until better bounds are to be found.

References

[Art68] M. Artin. On the solutions of analytic equations. *Invent. math.*, 5:277–291, 1968.

[Bos81] S. Bosch. A rigid analytic version of M. Artin's theorem on analytic equations. *Math. Ann.*, 255:395–404, 1981.

[CDV06] W. Castryck, J. Denef, and F. Vercauteren. Computing zeta functions of nondegenerate curves. *International Mathematics Research Papers*, 2006.

[DV06] J. Denef and F. Vercauteren. Counting points on Cab curves using Monsky-Washnitzer cohomology. *Finite Fields Appl.*, 12:78–102, 2006.

[Dwo63] B. Dwork. A deformation theory for the zeta function of a hypersurface. *Proc. Internat. Congr. Mathematicians (Stockholm, 1962)*, pages 247–259, 1963.

[Edi03] B. Edixhoven. Point counting after Kedlaya. *notes of a EIDMA-Stieltjes Graduate course given in Leiden*, 2003.

[Elk73] R. Elkik. Solutions d'équations à coefficients dans un anneau henselien. *Ann. Scient. Ec. Norm. Syp. 6*, 4:553–604, 1973.

[Ger05] R. Gerkmann. Relative rigid cohomology and point counting on families of elliptic curves. *preprint*, 2005.

[GG01] P. Gaudry and N. Gürel. An extension of Kedlaya's point-counting algorithm to superelliptic curves. *Advances in cryptology - ASIACRYPT*, 2001.

[Hub07] H. Hubrechts. Point counting in families of hyperelliptic curves. *Foundations of Computational Mathematics*, 2007.

[Ked] K. Kedlaya. p-adic cohomology: from theory to practice. *Lecture notes from the 2007 Arizona Winter School.*

[Ked01] K. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer Cohomology. *J. Ramanujan Math. Soc.*, 16:323–338, 2001.

[Ked04] K. Kedlaya. Computing Zeta Functions via p-adic Cohomology. *Algorithmic Number Theory, Springer Lecture Notes in Computer Science*, 3076:1–17, 2004.

[Kol88] J. Kollar. Sharp effective Nullstellensatz. *Journal of the American Mathematical Society*, 1:963–975, 1988.

[MW68] P. Monsky and G. Washnitzer. Formal cohomology 1. *Annals of Mathematics. Second Series*, 88:181–217, 1968.

[VdP86] M. Van der Put. The cohomology of Monsky and Washnitzer. *Mémoires de la Société Mathématique de France*, 23:33–59, 1986.

INDEX

Artin-Approximation property of w.c.f.g. algebras 12

Connection matrix 44

Gaudry/Gürel's superelliptic curves algorithm 3

Hubrechts' algorithm for families of hyperelliptic curves 31

Kedlaya's algorithm 2

Lefschetz formula for MW cohomology 11

Overconvergent functions, weak completion 9

Statement of the Weil conjectures 15

Vercauteren/Denef's Cab curves algorithm 3

Zeta function 1, 5

KEY TO ABBREVIATIONS

$A^\dagger$: The overconvergent completion of A

Z: The Zeta function

Ω^1_C : The module of Kahler differentials of C

$\hat{A}$: The p-adic completion of A

v_p: The p-adic valuation

dR: deRham cohomology

f.g. : finitely generated

GAGA: Serre's Géométrie algébrique et géométrie analytique

GM connection: The Gauss-Manin connection

MW Cohomology: The Cohomology of Monsky and Washnitzer ,

w.c.f.g. : weakly complete finitely generated