Elliptic Curves over Function Fields: A Numerical Investigation of Lower Bounds for Ulmer Curves

Peter Stevens

A thesis submitted to the faculty of graduate studies in partial fulfillment of the requirements for the degree of Master of Arts

Graduate program in Mathematics and Statistics

York University

Toronto, Ontario

September 2025

Abstract

This thesis investigates the ranks of Ulmer curves over the function fields $\mathbb{F}_p(t)$, p a prime, with a focus on computational techniques to estimate their group structure. Using SageMath, we implement point-generation algorithms, discriminant checks, and height-pairing computations to produce numerical evidence supporting predicted ranks. We combine brute-force and probabilistic sampling methods, enabling point generation and verification across a range of parameters. These results illustrate the computational challenges in large rank detection, suggest refinements, and contribute to the broader study of function fields.

Acknowledgements

This project would not have been possible without the diligent and studious supervision of Prof. Patrick Ingram, who continued to support me despite taking a new position and The Fields Institute.

As important to this whole enterprise is the support and mentorship of Prof. Ján Mináč, who has always encouraged me to pursue my goals no matter what was going on in my life.

Thanks are due to Prof. Shahin Kamali, who stepped in very quickly as the arm's-length chair, and to Prof. Paul Szeptycki for his moral support both leading up to and at the defense.

Finally, very special acknowledgements are due to my three referees from undergrad: John Inglis, Michael Nyenhuis, and Lin Hammill. I wouldn't be here without you.

Table of Contents

List of Tables

List of Figures

Chapter 1: History and Context

The study of Diophantine equations—polynomial equations in integer coefficients—is one of the oldest and most venerated disciplines of mathematics. One of the most celebrated examples is at the heart of Fermat's Last Theorem: $x^n + y^n = z^n$. The problem is to show that, for natural number $n > 2$, the only rational solutions are those in which one of x, y, or z is 0. Indeed, this is the true challenge of a Diophantine equation: to find any and all rational solutions.

Finding rational (and thus integer) solutions is easy for equations in one variable. Gauss' lemma teaches that for the polynomial $a_0 + a_1x + a_2x^2 + ... + a_nx^n = 0$, where a_0 to a_n are integers and $a_n \neq 0$, the only rational solutions can be those of the form $\pm\frac{a}{b}$, where $a|a_0$ and $b|a_n$. This does not say the polynomial will have those rational solutions, but gives an exhaustive list of what they could be. This also gives an efficient algorithm for determining which of those potentials *are* solutions, if any: simply substitute every such rational $\frac{a}{b}$ into the polynomial and determine whether it evaluates to 0.

With the introduction of even just one more variable, however, finding rational roots becomes significantly more difficult. In the 1650s, Fermat posed the problem of showing that the Diophantine equation $y^2 - x^3 = -2$ has only two integer solutions, $(3, \pm 5)$, despite it being hypothesized by Bachet that it has arbitrarily many rational solutions. Fermat's challenge went seemingly unanswered until 1908, when Axel Thue proved it as a special case of a more general result. (Silverman and Tate xvi-xvii)

Although solving Diophantine equations is a vexed and redoubtable enterprise, it is not just computational tools that one can avail themself of. With Descartes' introduction of Cartesian coordinates, Diophantine equations in two variables assumed a new dimension as, not just equations, but as curves existing in the $\mathbb{R}^2$ plane. Recalling the previous equation

$y^2 - x^3 = -2$, as a curve, many incredible results are unlocked. The aforementioned Bachet gave a *duplication* formula for a point (x, y) on the curve by giving a geometric argument: take a tangent line to the curve at a known rational point and determine the second point of intersection with the curve. Remarkably, not only can a second point always be found, but this second point will always be rational.

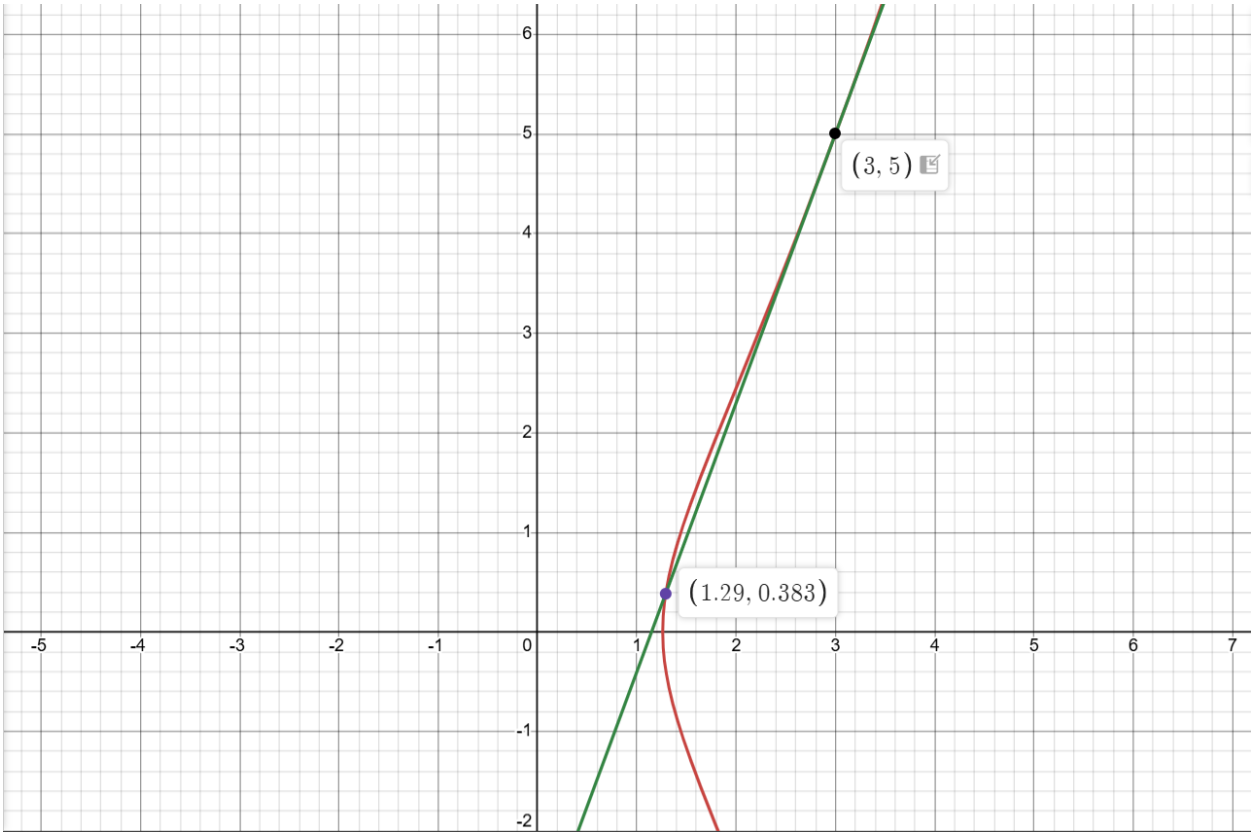

Figure 1.1: A visualization of the duplication formula on the elliptic curve $y^2 - x^3 = -2$. The point $P = (3, 5)$ on the curve is doubled to yield the rational point $2P = (1.29, 0.383)$.

Diophantine equations like this that are second degree in y and third degree in x have come to be known as *elliptic* curves. In addition to providing a trove of problems for the ambitious mathematician, the study of elliptic curves has found itself embroiled in the cloak-and-dagger world of public key cryptography. Data encryption is a long-standing staple of informational

warfare, putting enemies in an arms race to break each other's encryption keys. Encoding and decoding messages is all well and good so long as sender and recipient have access to a well-guarded key. However, this becomes inviable when the parties have no means of secure communication, and must resort to potentially monitored communication channels like email or phone. In this case, the parties cannot risk their secret falling into the wrong hands.

Elliptic curves have found application here. Public key encryption assumes that the enemy has access to the encryption method. The fundamental idea is that, in order to break an encoded message, one must solve a problem of the form $mP = Q$, where P and Q are points on some elliptic curve over a field $\mathbb{F}_q$ and m is some number to be decoded. It is assumed that the curve, Q, and P are not safe from enemy knowledge. However, solving $mP = Q$ is not a trivial task; Neal Koblitz and Victor Miller independently suggested that for P and Q on an elliptic curve over the finite field $\mathbb{F}_q$, it will require a multiple of $\sqrt{q}$ steps to solve $Q = mP$. (Silverman and Tate 155)

Encryption via elliptic curves not only allows encoded messages to be sent publicly, but reduces the information load needed to store the decryption key. In comparison to a standard encryption, an elliptic curve key takes about 10% of the storage space (Silverman and Tate 156). Hence, despite their fabled difficulty, elliptic curves offer an opportunity for safe and effective intelligence operations.

Elliptic curves are a challenging but versatile entity of mathematics, one which has yet to give up its greatest treasures. It is therefore with anticipation and excitement that we begin our sojourn to discover what else these humble yet humbling equations have to offer.

Chapter 2: Elliptic Curves and the Mordell-Weil Group

Elliptic curves, and Diophantine equations in general, can be seen as living at the intersection of number theory and geometry. Finding rational points on an elliptic curve is fundamentally a number-theoretic exercise. Since the elliptic curve is a curve through $\mathbb{R}^2$, it also belongs to the realm of geometry, as demonstrated by Bachet's duplication formula. But the pedigree of the elliptic curve goes still deeper, and in fact Bachet's duplication formula is highly suggestive of an underlying algebra on its rational points.

A natural first question is whether this duplication formula generalizes to arbitrary elliptic curves with integer coefficients. Let $E(x, y) = 0$ be a smooth elliptic curve of genus 1[1] over $\mathbb{Q}$ that admits a rational point, $P = (x_0, y_0)$. The slope of the tangent to $E(x, y)$ is a rational function in x and y, and so at P this is a rational number: let it be m. Then $y = m(x - x_P) + y_P$ is the tangent at P. Finding $2P$ is tantamount to first solving the polynomial equation $E(x, m(x - x_P) + y_P) = 0$ in x. This will be a third-degree equation, hence will have exactly three roots. By hypothesis, x_P is a repeated rational root, so there will be a second, distinct root, x_Q. Since the polynomial has rational coefficients and one root is a repeated rational number, x_Q is rational. Then the associated $y-$coordinate, $y_Q = m(x_Q - x_P) + y_p$, is also rational, so $2P = (x_Q, y_Q) \in E(\mathbb{Q})$. The duplication formula preserves rationality of points, not just on the elliptic curves considered by Bachet, but any with integer coefficients.

This preservation goes further. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are points of $E(x, y) = 0$, the slope of the line through P and Q is $m = \frac{y_Q - y_P}{x_Q - x_P}$; if $x_Q \neq x_P$, the slope is rational. The equation of the line is then $y = m(x - x_P) + y_P$; by similar reasoning, $E(x, m(x - x_P) + y_P) = 0$ is a third-degree polynomial equation in x with rational coefficients and rational roots x_P

[1] 'Genus' is a nonnegative number associated with an elliptic curve, and in general a non-singular elliptic curve (i.e. one that has a well-defined tangent everywhere) is of genus 1. (Silverman and Tate 120)

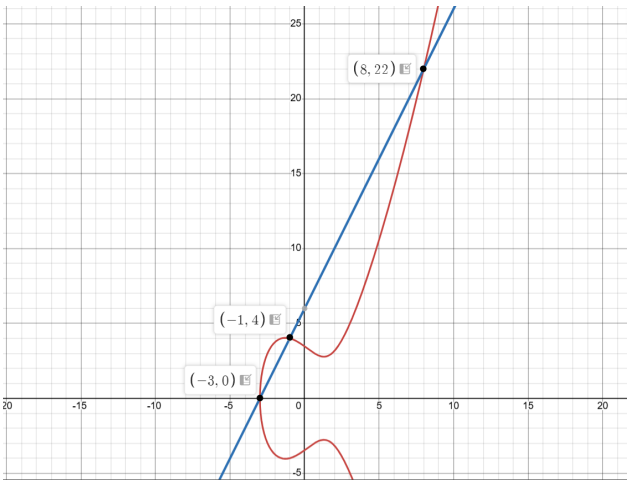

Figure 2.1: The elliptic curve $y^2 = x^3 - 5x + 12$ with rational points $P = (-3, 0)$, $Q = (-1, 4)$, and the rational point $P * Q = (8, 22)$.

and x_Q; the third root must then be rational, and thus the third point of intersection of the line and $E(x, y) = 0$, if any, is also rational. Silverman and Tate refer to this third point as $P * Q$ (9). The geometric relationships between the points is given in the following diagram.

These observations hint at a deeper algebraic structure between rational points on an elliptic curve. The preservation of rationality under $*$ may elicit the conclusion that $E(\mathbb{Q})$ forms a group under $*$. There is no group, though, unless we can legitimately baptize one of its points as the identity, i.e. a point $\mathcal{O} \in E(\mathbb{Q})$ such that for every $P \in E(\mathbb{Q})$, $\mathcal{O} * P = P * \mathcal{O} = P$. However, Silverman and Tate include a recipe whereby a given element of $E(\mathbb{Q})$ can assume the role of the identity, which then allows us to define group addition. Like Bachet's duplication formula, everything is done using tangents and secants of the elliptic curve. Once $\mathcal{O}$ has been designated, we define $P + Q = \mathcal{O} * (P * Q)$ (Silverman and Tate 12), i.e. it is the third point of intersection of the secant line connecting $\mathcal{O}$ and $P * Q$ with the elliptic curve.

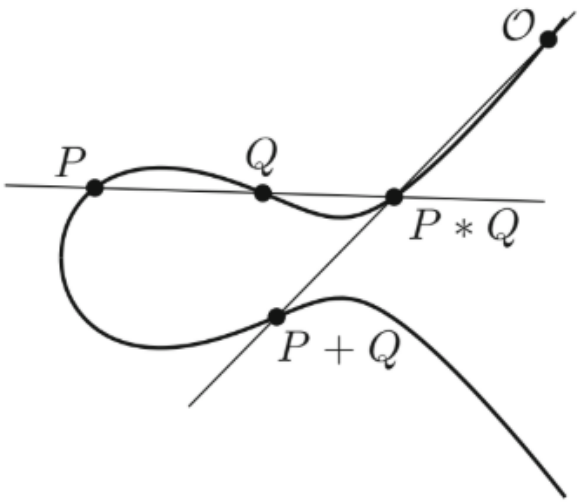

Figure 2.2: Finding the sum $P + Q$ with $\mathcal{O}$ acting as the identity. (Silverman and Tate 12)

The reason for giving $\mathcal{O}$ a role in determining $P + Q$ becomes apparent when we try to impose $P + \mathcal{O} = P$; under our definition, the point $P * \mathcal{O}$ is the third point of intersection of the line through P and $\mathcal{O}$ with the elliptic curve. By defining $P + \mathcal{O} = \mathcal{O} * (P * \mathcal{O})$, the sum is the third common point between the curve and the line joining $\mathcal{O}$ and $P * \mathcal{O}$, which is just P.

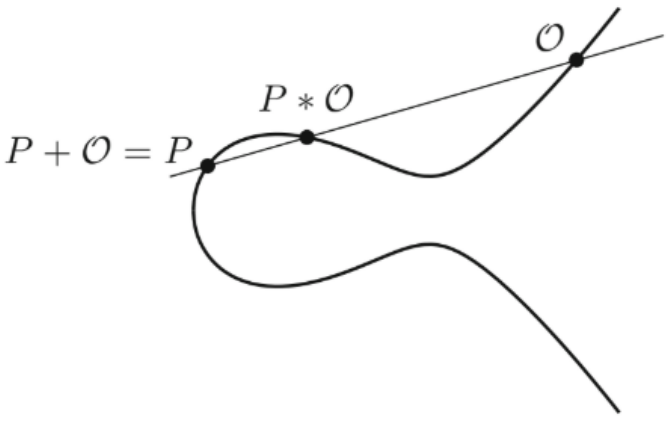

Figure 2.3: The point P with $\mathcal{O}$ acting as the identity. (Silverman and Tate 12)

These observations may give the impression that there are multiple groups associated with

an elliptic curve. However, further insight suggests there is a best choice for the identity. We may treat elliptic curves as curves in the projective plane, $\mathbb{P}^2$, where points are now (X, Y, Z), not all 0, and equivalent when they are non-zero scalar multiples of each other. The point $\mathcal{O}$ is taken to be the point at infinity, i.e. the point in $\mathbb{P}^2$ where $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$ grow arbitrarily large as $Z \to 0$. This is the set of points where $x = z = 0$ and $y \neq 0$ in $\mathbb{P}^2$. Consequently, the set $E(\mathbb{Q})$ is the set of rational points on $E(x, y) = 0$ and $\mathcal{O} \in \mathbb{P}^2$. $-P$ is the point such that $P + -P = \mathcal{O} \star (P \star -P) = \mathcal{O}$; we can therefore define it as the second intersection of $E(x, y) = 0$ with the vertical line $x = x(P)$, $x(P)$ the $x-$coordinate of P, since $\mathcal{O}$ is the point at infinity. (Silverman and Tate 23-24)

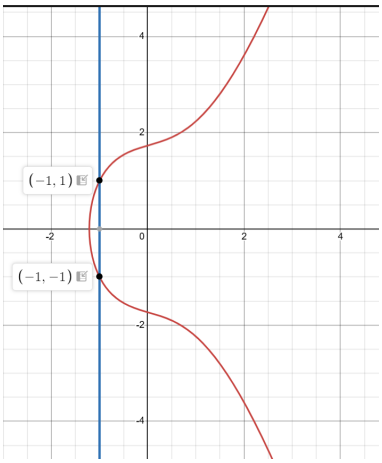

Figure 2.4: The point $(-1, 1)$ on $y^2 = x^3 + x + 3$ and its inverse, $(-1, -1)$.

We have so far confirmed that $E(\mathbb{Q})$ is closed under addition, contains an identity element, and contains inverses of its elements. Addition, though, is further expected to be associative. Silverman and Tate outline a nontrivial proof on pg.14-15. Although closure, identity, and inverses are relatively straightforward to show geometrically, associativity is significantly harder, requiring algebraic geometry machinery. As such, we accept it without proof. $E(\mathbb{Q})$ is thus a group under addition, and is baptized the Mordell-Weil group.

It is much clearer, perhaps trivial, that $+$ is commutative. Indeed, for P, $Q \in E(\mathbb{Q})$, the line

connecting P, Q, and $P * Q$ is unique, so is also the line connecting P, Q, and $Q * P$, which implies $P * Q = Q * P$. Extending this a step further, $\mathcal{O} * (P * Q) = \mathcal{O} * (Q * P)$, implying that $P + Q = Q + P$.

Having identified the pedigree of $E(\mathbb{Q})$, it is natural to inquire as to its structure. The Mordell-Weil group is a well known one, and with just a few conditions on the elliptic curve, it has a very familiar structure:

Theorem 2.0.1 (Mordell)**.** *Let E be a non-singular elliptic curve over $\mathbb{Q}$. Then the group of rational points $E(\mathbb{Q})$ is finitely generated and*

$$E(\mathbb{Q}) \cong \mathbb{Z}^k \oplus T$$

where T is the finite torsion subgroup and k the free group with k generators. (Silverman and Tate 16)

The torsion subgroup is the set of all elements t of a group such that there is integer m whereby $mt = \mathcal{O}$, and is free if no such integer exists. The number k is the rank of the elliptic curve, measuring the number of independent rational points of infinite order. Understanding this rank, and how to bound it for a particular class of elliptic curves, is the topic at the heart of this thesis.

Chapter 3: Elliptic Curves over Function Fields

While confirming a group's general structure is always a mathematical success, the specific group may still be difficult to identify. Indeed, the ranks of elliptic curves over $\mathbb{Q}$ are nearly as storied and mysterious as the giant squid. According to Silverman, while elliptic curves tend to have small rank, it has been conjectured that there exist elliptic curves over $\mathbb{Q}$ of arbitrarily large rank. Such curves are hardly forthcoming; one of rank at least 28 was produced by Elkies, but found to have coefficients of over 50 digits. (Silverman *Arithmetic* 254)

While $\mathbb{Q}$ seems to obfuscate rank, function fields have a richer, more complex structure, which consequently can make them more responsive to tools of algebraic geometry; such elliptic curves are potentially more tractable. Let $\mathcal{C}$ be a smooth, projective, and geometrically connected curve, and $\mathbb{F}_q$ a finite field. $\mathcal{F} = \mathbb{F}_q(\mathcal{C})$ is then the field of rational functions on C with coefficients in $\mathbb{F}_q$. Let E be an elliptic curve over $\mathbb{F}_q(\mathcal{C})$. The Mordell-Weil theorem still holds for E over a function field (Ulmer "Analogies Between Number Fields" 2), hence the rank is still finite.

An additional tool we can avail ourselves of in both number and function fields is L-functions. Consider an elliptic curve E over $\mathbb{Q}$ with integer coefficients. For every prime $p > 2$, reduce the coefficients of E modulo p to obtain an elliptic curve $\overline{E_p}$ over the finite field $\mathbb{F}_p$. Let $\overline{E_p}(\mathbb{F}_p)$ be the group of rational points in $\mathbb{F}_p$ lying on $\overline{E_p}$, complete with $\mathcal{O}$. If $\overline{E_p}$ is non-singular and irreducible of genus g, the Hasse-Weil theorem guarantees that the cardinality of $\overline{E_p}(\mathbb{F}_p)$ is given by $p + 1 - \epsilon_p$, where $|\epsilon_p| \leqslant 2g\sqrt{p}$ (Silverman and Tate 120).

Silverman and Tate write that each ϵ_p may be viewed "as a record that describes the reduction of E modulo (good) primes" (247), where 'good' prime means $\overline{E_p}$ is non-singular. This list then defines the $L-$function of E as the Euler product:

$$L(E,s) = \prod_{p \text{ odd prime}} \left(1 - \frac{\epsilon_p}{p^s} + \frac{1}{p^{2s-1}}\right)^{-1}$$

for $s \in \mathbb{C}$. If we recognize that each factor of the Euler product is an infinite geometric series, we may expand these factors into a Dirichlet series:

$$L(E,s) = \sum_{n=1}^{\infty} \frac{\epsilon_n}{n^s}$$

(248-249)[1] This function is defined by the Dirichlet series at s where the series converges. For all other complex numbers, $L(E,s)$ is locally described by a Taylor series.

The Birch and Swinnerton-Dyer conjecture asserts that for an elliptic curve E, its rank over $\mathbb{Q}$, $E(\mathbb{Q})$, is equivalent to $ord_{s=1}L(E,s)$, i.e. the order of vanishing of the function at $s = 1$, or the smallest integer k such that the kth derivative of $L(E,s)$ is nonzero at $s = 1$. Over function fields, the order of this vanishing is a known upper bound on the rank, with equality holding in some cases. (Ulmer "Analogies Between Number Fields" 3)

With an upper bound on the rank in hand, one may now wonder whether a lower bound can be similarly found. In fact, Ulmer's "Elliptic Curves with Large Rank over Function Fields" contains seminal results for a special family of elliptic curves. Let q be a power of prime p; let E be the elliptic curve $y^2 + xy = x^3 - t^d$, d a divisor of $p^n + 1$ for some $n \in \mathbb{Z}^+$. Rank is found by summing over certain divisors of d, with additional terms based on the divisibility of d and $q - 1$.

Theorem 3.0.1 (Ulmer)**.** *Let q be a power of prime p, E an elliptic curve $y^2 + xy = x^3 - t^d$, d a divisor of $p^n + 1$ for some $n \in \mathbb{Z}^+$. Then over the function field $\mathbb{F}_q(t)$, the Birch and Swinnerton-Dyer conjecture holds, and the rank is given by:*

[1] On page 10 of "The L-Functions of Elliptic Curves and Modular Forms", Tianxiang Liu shows how the $L-$function can be extended to points of bad reduction on E.

$$\textit{rank } E(\mathbb{F}_q(t)) = \sum_{\substack{e|d \\ e\nmid 6}} \frac{\phi(e)}{o_e(q)} + \begin{cases} 0 & \textit{if } 2 \nmid d \textit{ or } 4 \nmid q-1 \\ 1 & \textit{if } 2 \mid d \textit{ and } 4 \mid q-1 \end{cases} + \begin{cases} 0 & \textit{if } 3 \nmid d \\ 1 & \textit{if } 3 \nmid q-1 \\ 2 & \textit{if } 3 \mid q-1 \end{cases}$$

where $\phi(e)$ is the cardinality of e in $G = (\mathbb{Z}/e\mathbb{Z})^\times$ and $o_e(q)$ the order of q in G. (312)

In the event that $d = p^n + 1$, the minimum value of this sum becomes $\frac{p^n-1}{2n}$, giving a lower bound on the rank of $y^2 + xy = x^3 - t^d$ over $\mathbb{F}_p(t)$. (Ulmer "Elliptic Curves with Large Rank" 313)

The central issue explored in this thesis is one of robustness: can this lower bound be sharpened, and if so, when? This bound and the associated family of elliptic curves will remain the primary focus throughout the remainder of this work, with the core objective being to analyze how it behaves under changes to p and d, with an interest for when and how to sharpen it.

Chapter 4: Determining Points on Ulmer Curves over $\mathbb{F}_p(t)$, $p \leqslant 5$

Any elliptic curve of the form $y^2 + xy = x^3 - t^d$, p a prime, n a positive integer, and $d = p^n + 1$, will be referred to as an *Ulmer curve over* $\mathbb{F}_p(t)$. Our goal is to investigate Ulmer's lower bound on its rank. Computations and simulations were performed with SageMath; the complete source code is included in the appendix. The elements of $\mathbb{F}_p(t)$ are rational functions in t over p, i.e. ratios of polynomials with coefficients in $\mathbb{F}_p$. Thus, an Ulmer curve is a collection of pairs of elements from $\mathbb{F}_p(t)$ that satisfy the associated equation. Hence, these curves generalize not just our understandings of *elliptic curves* and *fields*, but even *points*.

The principal goal of this thesis is to see if there are sharper lower bounds on the ranks of Ulmer curves. One way to do this for an elliptic curve, E, is to take a set of points $\mathcal{S} \subseteq E(\mathbb{F}_p(t))$ and determine whether any subset generates the free abelian subgroup. If such a subset can be found, its rank is at least equal to the number of linearly independent points in $\mathcal{S}$.

This raises two important questions: first, where does $\mathcal{S}$ come from? Second, how do we determine if $\mathcal{S}$ belongs to the free abelian group?

We will address the first question in this section, deferring the second to section 6. The first hurdle in determining points on an elliptic curve over $\mathbb{F}_p(t)$ is that a point takes a pair of rational functions in t. A pair of rational functions corresponding to a point on $E(\mathbb{F}_p(t))$ must simultaneously satisfy two independent constraints: the functions must be valid $x-$ and $y-$coordinates on $E(\mathbb{F}_p(t))$, but must also satisfy the defining equation.

Early on, it was recognized that the equation $y^2 + xy = x^3 - t^d$ is quadratic in y over the field $\mathbb{F}_p(t)$. Rearranging gives $y^2 + xy + (-x^3 + t^d) = 0$. If $p \neq 2$, then for $x(t) \in \mathbb{F}_p(t)$, we may solve for y in $y^2 + x(t)y + (-x(t)^3 + t^d) = 0$ by completing the square, according to the

following procedure.

$$y^2 + x(t)y = x(t)^3 - t^d$$
$$y^2 + x(t)y + \frac{x(t)^2}{4} = \frac{x(t)^2}{4} + x(t)^3 - t^d$$
$$(y + \frac{x(t)}{2})^2 = \frac{x(t)^2+4(x(t)^3-t^d)}{4}$$
$$y + \frac{x(t)}{2} = \frac{\pm\sqrt{x(t)^2+4(x(t)^3-t^d)}}{2}$$
$$y = \frac{-x(t)\pm\sqrt{x(t)^2-4(-x(t)^3+t^d)}}{2}$$

For $p \neq 2$, the discriminant associated with $x(t)$ is $x(t)^2 + 4x(t)^3 - 4t^d$. Thus, the rational function $x(t)$ is accepted as a potential $x-$coordinate of some $P \in E(\mathbb{F}_p(t))$ if and only if this discriminant is a perfect square over $\mathbb{F}_p(t)$ when $p \neq 2$.

In the main function of the program, a rational function $x(t) \in \mathbb{F}_p(t)$ is generated and passed to check_disc: if $p \neq 2$, the discriminant of $y^2 + x(t)y - x(t)^3 + t^d$ is calculated. The program checks if this is a square root over $\mathbb{F}_p(t)$, returning **True** or **False**. **True** confirms that $x(t)$ is a valid $x-$coordinate of $E(\mathbb{F}_p(t))$; else, $x(t)$ is ignored for the remainder of the program. If $p = 2$, this part of the program is bypassed, as the completion of the square formula is invalid.

```
def check\_discriminant(a, t, d, p):
    if p ==2:
        return True
    disc = a^2 + 4*a^3 - 4*t^d
    return disc.is_square()
```

Figure 4.1: SageMath code to determine the discriminant of (t). If $p = 2$, the function automatically returns **True** and is essentially skipped.

With a confirmed $x(t)$ in hand, we need its mate, $y(t)$. $x(t)$ is placed in a set, which automatically eliminates duplicates. Prior to this, $x(t)$ is reduced via a reducing function, ensuring no equivalent rational functions are counted as separate solutions. $x(t)$ is then passed to a solving function, which attempts to solve for y in the equation $y^2 + x(t)y =$

$(x(t))^3 - t^d$ over. This reduces to computing roots of a quadratic over $\mathbb{F}_p(t)$, which is handled efficiently by SageMath's algebra routines. $x(t)$ and the corresponding solution $y(t)$ are then united into a single point, P, belonging to $E(\mathbb{F}_p(t))$. Every such point is then put into a set, which constitutes $\mathcal{S}$.

```
def solve(a, p, t, d, PR, y, E, set_of_points, j, FracR):
    try:
        rhs = y^2 + y*a - a^3 + FracR(t)^d
        poly_y = PR(rhs)
        roots_y = poly_y.roots(multiplicities=False)
        if roots_y:
            raw_y = roots_y[0]
            y_value = canonical_fraction(raw_y, FracR)
            P = E(a, y_value)
            values
            if P not in set_of_points:
                set_of_points.add(P)
                print(f"iteration {j}: ({a}, {y_value})\n")
    except (TypeError, ValueError) as e:
         pass
```

Figure 4.2: SageMath code to solve for and confirm y when a valid $x(t)$ is known for the Ulmer curve $y^2 + xy = x^3 - t^d$.

So far, we have discussed what to do with a set of $x-$coordinate functions when it is produced; we have yet to address where the set comes from.

One approach would be to systematically generate and test all rational functions up to a fixed degree. This may seem a natural approach; the field of coefficients is finite, so there will always be finitely many rationals up to a fixed degree. Moreover, access to a computer makes this process far faster than it would otherwise be with human computation. However, computation comes at a cost; what seems natural may be infeasible for a computer. Indeed, even for $p = 2$ and degree up to 6, there are $2^7(2^7 - 1)$ rational functions with non-zero denominator. Eliminating equivalent fractions can reduce the number to test, but the program will still need computational power to perform reductions and checks.

With this in mind, we opt for the second route: randomly generating rational functions up to a specified degree and checking which result in quadratic equations in y with a perfect square discriminant. However, settling one question raises a series of others. How should the rational functions be generated? How can we efficiently check that they are points on $E(\mathbb{F}_p(t))$ without overextending computational power? A simple Monte Carlo algorithm with uniform distribution was initially used. While this showed some promise in generating rational functions in small fields such as $\mathbb{F}_2(t)$, higher values of d in the defining equation led to prohibitively long wait times in generating points.

```
def random_rational(p, d, x, t, FracR):
    seed = floor(gauss(d*x, d*x/3))
    while seed < 0:
        seed = floor(gauss(d*x, d*x/3))
    t_powers = [t**j for j in range(seed + 1)]
    N = [GF(p).random_element() for j in range(seed + 1)]
    D = [GF(p).random_element() for k in range(seed + 1)]
    num = sum([t_powers[j] * N[j] for j in range(seed + 1)])
    den = sum([t_powers[j] * D[j] for j in range(seed + 1)])
    if den != 0:
        return canonical_fraction(num / den, FracR)
    else:
        raise ValueError
```

Figure 4.3: The generating function implementing a Gaussian (or normal) distribution to determine the polynomial's maximum degree. Uniform distributions are used to generate the coefficients.

To improve efficiency, we hypothesized that the exponent d in the Ulmer curve correlates with the typical degree of rational functions corresponding to points in $E(\mathbb{F}_p(t))$. This led us to adopting a randomly generated Gaussian variable to determine the maximum degree of the polynomial. It is centered at a user-defined multiple of d and standard deviation $\frac{1}{3}$ of the mean to reduce the likelihood of sampling non-positive values, which are invalid as degrees of rational functions over $\mathbb{F}_p(t)$.

This has led to some promising results when $p = 2$, 3, and 5, for values of n up to 3.

p value	n value	d value	Points Found	Ulmer's Lower Bound
2	1	3	3	1
2	2	5	2	1
2	3	9	7	2
3	1	4	1	1
3	2	10	1	2
5	1	6	7	2

Table 4.1: Empirical point counts on Ulmer curves compared to Ulmer's lower bound.

The points were generated using a Gaussian distribution, with a mean from $0.333d$ to $1.5d$.

The number of points generated in each case either matches or exceeds Ulmer's lower bound on the rank, meaning if these points form a linearly independent set under addition for elliptic curves, they all belong to the rank, and hence represent a sharpening of this bound. The discerning reader will notice that we have conspicuously omitted any discussion of Ulmer curves over $\mathbb{F}_p(t)$ when $p \geqslant 7$. The simple fact is that, when the algorithm is applied to Ulmer curves over this function field, or to an Ulmer curve where $n > 3$, it has yet to return any points on said curve.

Two immediate possible reasons for this are that, firstly, the rational functions satisfying these curves are very sparse over the region of rational functions the algorithm examines. Ulmer's lower bound may give the impression of overabundance for sufficiently large p and n; e.g. when $p = 11$ and $n = 5$, Ulmer's lower bound computes a rank of at least $\frac{11^5-1}{2*5} = 16,105$. But the pool of rational functions being searched could have millions of potential candidates. Indeed, if we are unlucky enough to pick a mean for the Gaussian distribution in an especially barren range of powers, our search may be fruitless for millions of trials.

A second reason this may be a statistical snipe hunt is the rational functions that *do* satisfy the associated equation may be of very large degree. If the search happens upon a rational function $x(t)$ associated with a point on the elliptic curve, but it is of degree 21, SageMath will have to raise a polynomial of up to 22 terms to the power of 3 in the course of solving the quadratic equation. This could lead to SageMath either throwing an error for the sheer magnitude of the powers involved, or silently failing and moving on to potentially more feasible roots. In either case, the best case scenario is being prompted that no solutions to the Ulmer curve were found.

Chapter 5: Points on General Ulmer Curves

While the Gaussian search has had limited success in larger fields, there are at least some discoverable facts about the points lying on the Ulmer curves over such fields. Three theorems apply to them, presented here in increasing specificity.

Throughout the theorems and their proofs, we assume $y^2 + xy = x^3 - t^d$ defines an Ulmer curve over the function field $\mathbb{F}_p(t)$, where $d = p^n + 1$, p a prime, n a positive integer.

Theorem 5.0.1. *If $P = (x(t), y(t)) \in E(\mathbb{F}_p(t))$, then $-P = (x(t), -x(t) - y(t))$.*

Proof. E can be regarded as a Weierstrass equation of the form $y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5$, where $a_1 = 1$, $a_2 = a_3 = a_4 = 0$, and $a_5 = -t^d$. Silverman gives a formula for the inverse of point $P = (x_0, y_0)$ as $-P = (x_0, -y_0 - a_1x_0 - a_2)$ (53). Applying this to elliptic curves over $\mathbb{F}_p(t)$ gives the result immediately. □

Theorem 5.0.2. *$p \equiv 2 \pmod 3$ and $n \equiv 1 \pmod 2$ if and only if $(t^{\frac{d}{3}}, 0)$ and $(t^{\frac{d}{3}}, -t^{\frac{d}{3}})$ are elements of $E(\mathbb{F}_p(t))$.*

Proof. Suppose first that $p \equiv 2 \pmod 3$ and $n \equiv 1 \pmod 2$. Then $n = 2k + 1$ for some positive integer k. $p^n \equiv 2^{2k+1} \pmod 3$, and $p^n \equiv 2^{2k}2 \pmod 3$. Since $2 \equiv -1 \pmod 3$, $2^{2k} \equiv 1 \pmod 3$, so $p^n \equiv 2 \pmod 3$. Then $p^n + 1 \equiv 0 \pmod 3$. This implies that $d = p^n + 1$ is divisible by 3, and $t^{\frac{d}{3}} \in \mathbb{F}_p(t)$. Let $x = t^{\frac{d}{3}}$; $(t^{\frac{d}{3}})^3 - t^d = 0$. Taking $x = t^{\frac{d}{3}}$, the associated Ulmer equation reduces to $y^2 - t^{\frac{d}{3}}y = 0$, which has solutions $y = 0$ and $y = t^{\frac{d}{3}}$. This establishes the first part of the theorem.

Next, assume that $(t^{\frac{d}{3}}, 0)$ and $(t^{\frac{d}{3}}, -t^{\frac{d}{3}})$ are elements of $E(\mathbb{F}_p(t))$. Then in particular, $t^{\frac{d}{3}} \in \mathbb{F}_p(t)$, and hence $p^n + 1 \equiv 0 \pmod 3$. By adding 2 to both sides, the rules for congruence give $p^n \equiv 2 \pmod 3$. p is congruent to one of 0, 1, or 2 modulo 3; the first two force $p^n + 1 \equiv 1 \pmod 3$ and $p^n + 1 \equiv 2 \pmod 3$ respectively, contradicting the assumption that $d = p^n + 1$

is divisible by 3. p has order 2 in $\mathbb{Z}_3$ by Euler's theorem, and hence $p^{2k} \equiv 1 \pmod 3$ for all positive integers k. Thus, $n \equiv 1 \pmod 2$. This establishes the second part of the claim.
In general, we have $p \equiv 2 \pmod 3$ and $n \equiv 1 \pmod 2$ if and only if $(t^{\frac{d}{3}}, 0)$ and $(t^{\frac{d}{3}}, -t^{\frac{d}{3}})$ are elements of $E(\mathbb{F}_p(t))$. □

Theorem 5.0.3. *$p \equiv 1 \pmod 4$ if and only if $(0, ut^{\frac{d}{2}})$ and $(0, -ut^{\frac{d}{2}})$ are distinct elements of $E(\mathbb{F}_p(t))$, where $u^2 \equiv -1 \pmod p$.*

Proof. Begin by assuming $p \equiv 1 \pmod 4$, and assume that $x(t) = 0$ is being considered as a valid $x-$coordinate function. Since $p \neq 2$, we may use the quadratic formula, giving $y = \frac{\pm\sqrt{-4t^d}}{2}$. It is a fact of number theory that -1 is a square modulo p if and only if $p = 2$ or $p \equiv 1 \pmod 4$. Hence, the solution to y simplifies to $\frac{\pm 2ut^{\frac{d}{2}}}{2} = \pm ut^{\frac{d}{2}}$, where $u^2 \equiv -1 \pmod p$. Since p is odd, d is even, and u is distinct from $-u$. Thus, $ut^{\frac{d}{2}}$ and $-ut^{\frac{d}{2}}$ are distinct elements of $\mathbb{F}_p(t)$ such that $(0, ut^{\frac{d}{2}})$ and $(0, -ut^{\frac{d}{2}})$ belong to $E(\mathbb{F}_p(t))$.
Next, assume that $(0, ut^{\frac{d}{2}})$ and $(0, -ut^{\frac{d}{2}})$ are distinct elements of $E(\mathbb{F}_p(t))$. Direct substitution gives $(ut^{\frac{d}{2}})^2 = -t^d$ and $(-ut^{\frac{d}{2}})^2 = -t^d$. Thus, $u^2 \equiv -1 \pmod p$ and $(-u)^2 \equiv -1 \pmod p$, so either $p = 2$ or $p \equiv 1 \pmod 4$. Since u and $-u$ are distinct by assumption, we must have $p \equiv 1 \pmod 4$.
Thus, $p \equiv 1 \pmod 4$ if and only if $(0, ut^{\frac{d}{2}})$ and $(0, -ut^{\frac{d}{2}})$ are distinct elements of $E\mathbb{F}_p(t)$, where $u^2 \equiv -1 \pmod p$. □

Besides these cases, we turn to the issue of when the random Gaussian generation fails to return a valid x-coordinate. In such a case, it may be possible to learn something via slightly more unwieldy avenues. The virtue of a set of randomly generated points is that it is possible to test rationals of a wide range of degrees in a very short period of time. However, that same algorithm may inadvertently step over regions of $\mathbb{F}_p(t)$ of highly concentrated solution points. The alternative, ensuring none of these points are missed, is the exhaustive search that was eschewed in favour of a random Gaussian search.

To that end, if the random_rational function has failed to return even a single rational function, our plan B is to resort back to an exhaustive search of all the rational functions up to a maximum degree. This is the role of the function, brute_force_algorithm.

```
def brute_force_algorithm(checked_set, p, t, max_degree, FracR):
    rationals_to_check = set()
    polynomials = set()
    for degree in range(0, max_degree + 1):
        for m in range(p ** (degree + 1)):
            current_polynomial = 0
            mm = m
            for k in range(degree + 1):
                coeff = mm % p
                current_polynomial += coeff * (t ** k)
                mm //= p
            polynomials.add(current_polynomial)
    for numerator in polynomials:
        for denominator in polynomials:
            if denominator == 0:
                continue
            rational = canonical_fraction(numerator / denominator, FracR)
            if (rational not in checked_set) and (rational not in rationals_to_check):
                rationals_to_check.add(rational)
    return rationals_to_check
```

Figure 5.1: Algorithm generating all rationals in $\mathbb{F}_p(t)$ up to degree max_degree.

Because these are rational functions with coefficients in a finite field, there are finitely many. Indeed, over $\mathbb{F}_p(t)$, there are at most $p^{j+1}(p^{j+1}-1)$ distinct rational functions of degree j or less. Thus, the algorithm will terminate after generating finitely many rational functions. The first **for** loop iterates from 0 to max_degree, while the nested loop inside generates the terms of each polynomial. The coefficients appearing in the polynomial are each determined by dividing m by p to the power of the index, and then determining this result modulo p. To provide some computational relief, the set of any rational functions generated by the random_rational function is passed tot he brute_force_algorithm. Since no points at all have been found on the elliptic curve in question, any rational functions in this set are failed

$x-$coordinates on the curve, and thus don't need to be checked again.

Consider the table below, which generates all polynomials in $\mathbb{Z}_2[t]$ of degree at most 2.

m value	k values	coefficients	polynomial
0	0, 1, 2	$\lfloor(0\|2^0)\rfloor \equiv 0$, $\lfloor(0\|2^1)\rfloor \equiv 0$, $\lfloor(0\|2^2)\rfloor \equiv 0 \pmod 2$	0
1	0, 1, 2	$\lfloor(1\|2^0)\rfloor \equiv 1$, $\lfloor(1\|2^1)\rfloor \equiv 0$, $\lfloor(1\|2^2)\rfloor \equiv 0 \pmod 2$	1
2	0, 1, 2	$\lfloor(2\|2^0)\rfloor \equiv 0$, $\lfloor(2\|2^1)\rfloor \equiv 1$, $\lfloor(2\|2^2)\rfloor \equiv 0 \pmod 2$	t
3	0, 1, 2	$\lfloor(3\|2^0)\rfloor \equiv 1$, $\lfloor(3\|2^1)\rfloor \equiv 1$, $\lfloor(3\|2^2)\rfloor \equiv 0 \pmod 2$	$1+t$
4	0, 1, 2	$\lfloor(4\|2^0)\rfloor \equiv 0$, $\lfloor(4\|2^1)\rfloor \equiv 0$, $\lfloor(4\|2^2)\rfloor \equiv 1 \pmod 2$	t^2
5	0, 1, 2	$\lfloor(5\|2^0)\rfloor \equiv 1$, $\lfloor(5\|2^1)\rfloor \equiv 0$, $\lfloor(5\|2^2)\rfloor \equiv 1 \pmod 2$	$1+t^2$
6	0, 1, 2	$\lfloor(6\|2^0)\rfloor \equiv 0$, $\lfloor(6\|2^1)\rfloor \equiv 1$, $\lfloor(6\|2^2)\rfloor \equiv 1 \pmod 2$	$t+t^2$
7	0, 1, 2	$\lfloor(7\|2^0)\rfloor \equiv 1$, $\lfloor(7\|2^1)\rfloor \equiv 1$, $\lfloor(7\|2^2)\rfloor \equiv 1 \pmod 2$	$1+t+t^2$

Table 5.1: Illustration of how the brute_force_algorithm generates all polynomials up to specified degree using the exponent modulo powers of p to determine the coefficients of each term.

When all of the polynomials have been generated, the function forms all the possible rational functions using the polynomials as numerators and non-zero polynomials as denom-

inators. Each rational is passed to the fraction_reduction function to be reduced. The brute_force_algorithm then checks whether this function was previously generated by random_rational, and hence whether it is in the previously passed set. If it is not, the function adds this to a set of rationals to be checked by the main function.

The main function passes this set to the solve function, which substitutes each $x(t)$ in the set into the equation $y^2 + xy = x^3 - t^d$, attempting to solve for y. If y can be solved, then we have successfully identified a point that was missed by the Gaussian generator. If none of the $x-$values correspond to $y-$values, it need not imply utter failure; indeed, if we have tested all rational functions up to degree j, we have methodically shown that *no rational function of degree j or lower* can be an $x-$coordinate of any point in $E(\mathbb{F}_p(t))$. Thus, we will have to search for rational functions of degree $j+1$ or higher. Ulmer's lower bound guarantees these points do exist, but this result will indicate we are hunting for rational functions of higher degree.

Thus far, the focus of this discussion has been on finding rational points on an elliptic curve. This was only part of what we set out to accomplish: we now turn to the question of determining which of these points belong to the torsion-free subgroup of the Mordell-Weil group.

Chapter 6: Naive and Canonical Heights

Recall that the elements of $E(\mathbb{F}_p(t))$ form a group under addition defined geometrically in section 2. We denote this the Mordell-Weil group; it is isomorphic to $\mathbb{Z}^k \bigoplus T$, T the torsion subgroup and $\mathbb{Z}^k$ the free abelian group of rank k. As the focus of this thesis is the rank, the hope is that the points generated by the Gaussian distribution or the brute_force function are generators of the group.

To justify the next major tool of use, it is helpful to get slightly into the weeds of linear algebra and vector spaces. In a real, finite-dimensional vector space V over the field $\mathcal{F}$, a finite set of vectors U is linearly independent if and only if the equation $\sum_{\overrightarrow{u} \in U} a_u \overrightarrow{u} = \overrightarrow{0}$ has only the trivial solution where each $a_u = 0$. Furthermore, this set of linearly independent vectors is a subset of a basis for V.[1] The cardinality of the basis is the dimension of V, $dim(V)$, and is analogous to the rank of the Mordell-Weil group. Moreover, we have $dim(V) \geqslant |U|$. This raises an interesting question: can the generating set of a free abelian group be similarly flushed out?

Determining linear independence can be computationally harrowing; however, if a real, finite-dimensional vector space admits an inner product, this can provide an algorithm for accomplishing this. Recall that an inner product is a function $\langle ., . \rangle : V \times V \to \mathbb{R}$ such that, for $\overrightarrow{u}$, $\overrightarrow{v}$, $\overrightarrow{w} \in V$, and $\lambda \in \mathbb{R}$, $\langle ., . \rangle$ is:

positive definite ($\langle \overrightarrow{u}, \overrightarrow{u} \rangle \geqslant 0$)

symmetric ($\langle \overrightarrow{u}, \overrightarrow{v} \rangle = \langle \overrightarrow{v}, \overrightarrow{u} \rangle$)

linear in its first coordinate ($\langle \lambda \overrightarrow{u} + \overrightarrow{w}, \overrightarrow{v} \rangle = \lambda \langle \overrightarrow{u}, \overrightarrow{v} \rangle + \langle \overrightarrow{w}, \overrightarrow{v} \rangle$)

When such a function is available, the linear independence of a finite set $S \subseteq V$ can be determined from the Gram matrix, which is the square symmetric and positive semi-definite

[1] Recall that a basis is a set of vectors such that every $\overrightarrow{v} \in V$ can be expressed as a unique linear combination of the vectors in U^*. The fact that this U^* exists is a consequence of Zorn's lemma. cf. pg. 71-72 of *Topology* by James Munkres.

$n \times n$ matrix G where $n = |S|$ and $G_{ij} = \langle \overrightarrow{u_i}, \overrightarrow{u_j} \rangle$, $u_i, u_j \in S$. Since G is square, we may take its determinant, and since G is positive semi-definite, S is linearly independent if and only if $det(G) > 0$.

While the Mordell-Weil group is not a vector space, if we can equip it with a function similar to the inner product, we may be able to elicit similar results. To have a chance of that, we need to understand why we should be interested in an inner product at all.

An inner product satisfies many interesting properties; perhaps most importantly, it supports a notion of *magnitude*. In an inner product space, we can define a function $|| \cdot || : V \to \mathbb{R}$ as $||\overrightarrow{v}|| = \sqrt{\langle \overrightarrow{v}, \overrightarrow{v} \rangle}$. The inner product is positive definite, so $||\overrightarrow{v}||$ is well-defined. This is the *norm induced by the inner product* of V, and satisfies the properties we expect of a norm: it is positive-definite, absolutely homogeneous, and satisfies the triangle inequality.

To see if elliptic curves admit of a similar function, we need some way to measure the size or complexity of its elements. This role is played by *heights*.

A height is a function $h : E(\mathcal{F}) \to [0, \infty)$ that measures the complexity of the point P from a number theoretic standpoint. Silverman and Tate give a simple example of such a function which sends a reduced rational number $\frac{m}{n}$ to $max\{|m|, |n|\}$. If we consider two rational numbers $\frac{1}{2}$ and $\frac{9999}{20000}$, there is little difference between them as real numbers; one is only 0.00005 more than the other. However, $\frac{9999}{20000}$ is more complex in certain aspects. (65) E.g. as a decimal, it has many more non-zero leading digits than $\frac{1}{2}$ does, and thus is stands to reason that $h(\frac{1}{2}) < h(\frac{9999}{20000})$. Height functions express these intuitions concretely.

For elliptic curves over fields, Silverman and Tate apply a height function by taking the $x-$coordinate of $P \in E(\mathcal{F})$. (66) Such curves are quadratic in y and cubic in x, so a given $x-$value corresponds to at most 2 $y-$values. The $x-$coordinate of P thus plays a highly deterministic role in P's properties. We therefore follow Silverman and Tate in applying the height function to the $x-$coordinates of points on the Ulmer curves.

Since the elements of $E(\mathbb{F}_p(t))$ are pairs of rational functions over $\mathbb{F}_p$, a natural measure of

height is degree. Let $x(t) = \frac{a(t)}{b(t)}$, $a(t)$, $b(t) \in \mathbb{F}_p[t]$, where $b(t) \neq 0$ and GCD$(a(t), b(t)) = 1$. Let $h(x(t)) = max\{\deg(a(t)), \deg(b(t))\}$. This captures the intuition that rational functions of higher degree are more complex than others. We define $h(P) = h(x(P))$, where $x(P)$ is the $x-$coordinate function of P. Call this the *naive* height, h, of P.

The naive height, however, is not the function we want. One property the inner product has is that doubling a vector $\overrightarrow{v} \in V$ quadruples the inner product, i.e. $\langle 2\overrightarrow{v}, 2\overrightarrow{v} \rangle = 4\langle \overrightarrow{v}, \overrightarrow{v} \rangle$. As noted by Silverman and Tate, doubling the height is nearly quadratic: $h(2P) \geqslant 4h(P) - k$, where k is a constant. (68) Moreover, the naive height as it stands does not properly define a *binary* relationship between points, but rather a *unitary* property of points. Hence, we need to take this analysis a step further.

Silverman defines the divisor group of a curve C, $Div(C)$, as a free abelian group on the points of C, given by the formal sum:

$$D = \sum_{P \in C} n_P(P)$$

where $n_P = 0$ for all but finitely many $P \in C$. We extend this to define the divisor associated with the function $f : C \to \overline{K}$ on the curve C as:

$$div(f) = \sum_{P \in C} ord_P(f)(P)$$

(Silverman *Arithmetic* 27). In this context, $ord_P(f)(P) = n$, where $n > 0$ if there are n 0s at P, $n < 0$ if there are n poles at P, and 0 otherwise. If $f : E(\mathbb{F}_p(t)) \to \mathbb{F}_p(t)$ is the map $(x, y) \mapsto x$, the point $P = (x_0, y_0) \in E(\mathcal{F})$ is a root of f if and only if $x_0 = 0$. By Theorem 5..0.3 above, the Ulmer curve E has at most 2 distinct points where $x(t) = 0$. Moreover, the $x-$coordinate of a point P will have poles nowhere except at the identity, $\mathcal{O}$. To investigate its behaviour, we follow Silverman and Tate in rewriting the variable ratios as:

$$u = \frac{x}{y}, s = \frac{1}{y}$$

(49), using u instead of t as the latter already plays a role in the Ulmer curve. Substituting this into $y^2 + xy = x^3 - t^d$ then yields:

$$(\tfrac{1}{s})^2 + (\tfrac{u}{s})(\tfrac{1}{s}) = (\tfrac{u}{s})^3 - t^d$$
$$\rightarrow \tfrac{1}{s^2} + \tfrac{u}{s^2} = \tfrac{u^3}{s^3} - t^d$$
$$\rightarrow s + su = u^3 - s^3 t^d$$

This corresponds to a shift of the point at infinity to the point $(0,0)$, hence as $(x,y) \rightarrow \mathcal{O}$, $(s,u) \rightarrow (0,0)$ in this new space (Silverman and Tate 50). The authors consider the local ring R_c, where $c \in C$ and $R_c = \{\frac{a}{b}\}$, where a and b are functions from $E(\mathcal{F}) \rightarrow \mathcal{F}$ such that $b(c) \neq 0$. This allows us to study a curve's behaviour on a neighbourhood near a point of interest. They show that $(u,s) \in C(c^v)$, v a natural number, if and only if $u \in c^v R_p$ and $s \in p^{3v} R_c$ (51). Hence, when (u,s) is close to $(0,0)$, u vanishes with order 1. u is therefore the local parameter near $(0,0)$. As $(u,s) \rightarrow (0,0)$, the order of vanishing of the right-hand side is dominated by s, the right by u^3. Hence, near $(0,0)$, s vanishes as u^3; we represent this with the notation $s \sim u^3$. Using $x = \frac{u}{s}$ gives $x \sim \frac{u}{s} \sim \frac{u}{u^3} \sim u^{-2}$. Thus, near $\mathcal{O}$, x behaves as u^{-2}, so $ord_{\mathcal{O}} x = -2$. But $ord_{\mathcal{O}} f(\mathcal{O}) = ord_{\mathcal{O}} x = -2$. Thus, $ord_{\mathcal{O}} f(\mathcal{O})$ is well-defined. Hence, $div(f) \in Div(E)$.

To establish our desired results, we must get a bit further into the weeds with height functions and divisors. Let $D_1 = 2\mathcal{O} \in Div(E)$. Let $(x)_0$ be the divisor of zeros of f, with coefficients their multiplicity; let this be D_2, which is also in $Div(E)$. Thus, $div(f) = (x)_0 - 2\mathcal{O} = D_2 - D_1$. f is a rational function, since it is defined everywhere on E but $\mathcal{O}$. Thus, $D_2 - D_1$ is a principal divisor; by Theorem 4.8a in Silverman, D_1 and D_2 are linearly equivalent. By the same theorem, there are heights h_{E,D_1} and h_{E,D_2} associated with each divisor, and we have:

$$h_{E,D_1}(P) = h_{E,D_2}(P) + O_{E,D_1,D_2}(1)$$

("Canonical Heights" 14). Making the obvious substitutions gives:

$$h_{E,(x)_0}(P) = h_{E,2\mathcal{O}}(P) + O_{E,(x)_0,2\mathcal{O}}(1)$$

Define the projective map $\phi : E \to \mathbb{P}^1$, where $\mathbb{P}^1$ is the projective line. In homogeneous coordinates, this is $P \mapsto [1 : x(P)]$. Since x has a double pole at $\mathcal{O}$, define $\phi(\mathcal{O}) = [0 : 1] = \infty$; since $\infty \in \mathbb{P}^1$, ϕ is defined at $\mathcal{O}$, and therefore globally defined. It is thus a morphism, and by Theorem 4.8b:

$$h_{E,2\mathcal{O}}(P) = h_{\mathbb{P}^1,\infty}(\phi(P)) + O_{E,\mathbb{P}^1,\phi}(1)$$

(14). Substituting again, we have:

$$h_{E,(x)_0}(P) = h_{\mathbb{P}^1,\infty}(\phi(P)) + K_1$$

where K_1 does not depend on P.

For the morphism ϕ, if $x(P) = \frac{a(t)}{b(t)}$, $a(t)$ and $b(t)$ coprime, then $\phi(P) = [b(t) : a(t)]$. Let $deg(a(t)) = m$, $deg(b(t)) = n$. n is the number of times $b(t)$ vanishes, which corresponds to the point at infinity on $\mathbb{P}^1$, since $a(t)$ and $b(t)$ have no common roots. If $m > n$, $m - n$ measures by how much the numerator dominates the denominator, which also corresponds to points at infinity.

$h_{\mathbb{P}^1,\infty}$ measures the arithmetic complexity of points as rational functions relative to infinity; the higher m and n, the more complex the interactions with infinity. Thus, $h_{\mathbb{P}^1,\infty}(\phi(P))$ counts the poles of the function and by how much the numerator dominates the denominator (if at all), so this height is just $n + \max\{m - n, 0\} = \max\{m, n\}$. But this is precisely the naive height, h. Hence, $h_{\mathbb{P}^1,\infty}(\phi(P)) = h(P) + K_2$. Thus, by the previously established results;

$$h_{E,(x)_0}(P) = h(P) + K_3$$

K_2 and K_3 do not depend on P.

Finally, it all comes together in the following theorem:

Theorem 6.0.1 (Néron-Tate)**.** *Let E be an elliptic curve over the function field $\mathcal{F}$, and let $D \in Div(E)$. Let $P \in E(\overline{K})$. If D is symmetric, then the limit:*

$$\hat{h}_{E,D}(P) = \lim_{n\to\infty} \frac{h_{E,D}(2^n P)}{4^n}$$

exists.

(Silverman "Canonical Heights" 23). D is symmetric if and only if the pullback of D under the inverse image $[-1]$ is linearly equivalent to D, i.e. $[-1] * D \ D$. (Silverman "Canonical Heights" 23) If $P \in E(\mathcal{F})$ such that $P = (0, b(t))$ with multiplicity n, then $-P = (0, -b(t))$ with multiplicity n by Theorem 5.1. If nP is in the series for $(x)_0$, $n(-P)$ is also in the series. Hence, $[-1] * (x)_0$ maps nP to $n(-P)$, so is a permutation of $(x)_0$, and $[-1] * (x)_0 = (x)_0$. $[-1] * (x)_0 - (x)_0 = div(g)$, where $g(P)$ is a nonzero constant rational function, hence is principal. $(x)_0$ is symmetric, and thus:

$$\hat{h}_{E,(x)_0}(P) = \lim_{n\to\infty} \frac{h_{E,(x)_0}(2^n P)}{4^n}$$

Since $h_{E,(x)_0}$ and h differ by a constant:

$$\hat{h}_{E,(x)_0}(P) = \lim_{n\to\infty} \frac{h(2^n P)}{4^n}$$

We thus define $\hat{h}_{E,(x)_0}$ to be **the** canonical height, and write:

$$\hat{h}(P) = \lim_{n\to\infty} \frac{h(2^n P)}{4^n}$$

The canonical height has many properties similar to an inner product, which are outlined in the following two theorems.

Theorem 6.0.2. *Let E be an elliptic curve over $\mathcal{F}$; for $m \in \mathbb{Z}$ $P \in E(\mathcal{F})$, $\hat{h}(mP) = m^2\hat{h}(P)$.*

Proof. If $m = 0$, $0P = (0 + 0)P$, so $0P = 0P + 0P$. Add the inverse of $0P$ to both sides to obtain $0P = \mathcal{O}$. Now, $\hat{h}(\mathcal{O}) = \lim_{n\to\infty} \frac{h(2^n\mathcal{O})}{4^n}$, but $2^n\mathcal{O} = \mathcal{O}$, so $h(2^n\mathcal{O})$ is constant. Hence, $\lim_{n\to\infty} \frac{h(2^n\mathcal{O})}{4^n} = 0$. Thus, $\hat{h}(\mathcal{O}) = 0$. Since $\hat{h}(P)$ is defined, $0\hat{h}(P) = 0$. Thus, for $m = 0$, $\hat{h}(mP) = m^2\hat{h}(P)$.

If $m < 0$, then since naive height is even, $h(mP) = h(-mP)$, and thus $\hat{h}(mP) = \hat{h}(-mP)$.

Ergo, whatever is true of mP is true of $-mP$.

Finally, let $m > 0$. By corollary 6.4 in *The Arithmetic of Elliptic Curves*, $h(mP) = m^2h(P)+O(1)$, where $O(1)$ depends on E and m (Silverman 238). Hence, for some $k \in \mathbb{R}$, $h(mP) = m^2h(P) + k$. Thus, for 2^nP, $h(m(2^nP)) = m^2h(2^nP) + k$. Dividing by 4^n yields:

$$\frac{h(2^n(mP))}{4^n} = \frac{m^2h(2^nP)}{4^n} + \frac{k}{4^n}$$

and letting n go to infinity yields:

$$\hat{h}(mP) = m^2\hat{h}(P)$$

In general then, $\hat{h}(mP) = m^2\hat{h}(P)$. □

Theorem 6.0.3. *$\hat{h}(P) = 0$ if and only if P belongs to the torsion subgroup of $E(\mathcal{F})$.*

Proof. Suppose firstly that P is in the torsion subgroup. Then $ord(P)$ is finite, so 2^nP will repeat, hence $h(2^nP)$ assumes finitely many values. It is therefore bounded, and there is $k \in \mathbb{R}$ such that $h(2^nP) \leqslant k$ for all n. Hence, $\frac{k}{4^n} \geqslant \frac{h(2^nP)}{4^n}$; letting n go to infinity then gives $\hat{h}(P) = \lim_{n\to\infty} \frac{h(2^nP)}{4^n} \leqslant \lim_{n\to\infty} \frac{k}{4^n} = 0$. Since naive height is nonnegative, $\hat{h}(P) = 0$.

Suppose next that P is *not* in the torsion subgroup; then it is in the free abelian subgroup of $E(\mathcal{F})$. Silverman and Tate show in their text that there is a constant k such that $h(2P) \geqslant 4h(P) - k$ for each $P \in E(\mathcal{F})$ (68), which in turn implies $h(2^jP) \geqslant 4h(2^{j-1}P) - k$. If we apply that inequality to $h(2^{j-1}P)$, this gives:

$$h(2^jP) \geqslant 4h(2^{j-1}P) - k \geqslant 4(4h(2^{j-2}P) - k) - k = 4^2h(2^{j-2}P) - (4 + 1)k$$

Applying this j times and the geometric series formula to the k terms gives:

$$h(2^jP) \geqslant 4^jh(P) - k\sum_{n=0}^{j-1} 4^n = 4^jh(P) - \frac{4^j-1}{3}k$$

Because P is not in the torsion subgroup, we may choose positive integer m such that $h(mP) - \frac{k}{3} > 0$. Then we have:

$$\hat{h}(mP) = \lim_{n\to\infty} \frac{h(2^nmP)}{4^n} \geqslant \lim_{n\to\infty}(\frac{4^nh(mP)}{4^n} - \frac{4^n-1}{4^n3}k) = (h(mP) - \frac{1}{3}k) > 0$$

Let $C = h(mP) - \frac{1}{3}k$; by Theorem 6.2, $\hat{h}(mP) = m^2\hat{h}(P)$. Thus, $\hat{h}(P) > \frac{C}{m^2}$, so $\hat{h}(P) \neq 0$. We therefore have that $\hat{h}(P) = 0$ if and only if P is in the torsion subgroup of $E(\mathcal{F})$. $\square$

The canonical height will help us build an inner product, but an inner product can only work properly on an inner product *space*. Thus, we have yet more to do; we have to provide a real vector space on which it can work.

Chapter 7: The Néron-Tate Pairings

Let $E(\mathcal{F})$ be an elliptic curve over the field $\mathcal{F}$; the Néron-Tate pairing is defined in the following way:

$$\langle \cdot, \cdot \rangle : E(\overline{\mathcal{F}}) \times E(\overline{\mathcal{F}}) \to \mathbb{R}$$

$$\langle P, Q \rangle = \tfrac{1}{2}(\hat{h}(P+Q) - \hat{h}(P) - \hat{h}(Q)) \text{ (Silverman "Canonical Heights" 30)}$$

The remarkable feature of this pairing is that it has the properties of an inner product.

Theorem 7.0.1. *For P, $Q \in E(\mathcal{F})$, $\langle P, Q \rangle$ is bilinear over $\mathbb{Z}$ and symmetric. $\langle P, P \rangle = 0$ if and only if P is in the torsion subgroup.*

Proof. Symmetry follows immediately from the definition of $\langle P, Q \rangle$. Theorem 6.2 gives $\langle P, P \rangle = \hat{h}(P)$, which theorem 6.3 guarantees is 0 if and only if P is in the torsion subgroup. Bilinearity follows from the fact that the canonical height obeys the parallelogram law. Silverman gives a proof that, for P, $Q \in E(\mathcal{F})$, $h(P+Q) + h(P-Q) = 2h(P) + 2h(Q) + k$, k a constant (*Arithmetic* 235-237). Then we have:

$$\tfrac{h(2^n(P+Q))}{4^n} + \tfrac{h(2^n(P-Q))}{4^n} = 2\tfrac{h(2^nP)}{4^n} + 2\tfrac{h(2^nQ)}{4^n} + \tfrac{k}{4^n}$$

Taking the limit as n goes to infinity gives the parallelogram law for canonical heights:

$$\hat{h}(P+Q) + \hat{h}(P-Q) = \hat{h}(P) + \hat{h}(Q)$$

Silverman shows that this law implies bilinearity on pages 249-250 of *Arithmetic*. □

Notice that, as a group isomorphic to $\mathbb{Z}^r \bigoplus T$, T the torsion subgroup, $E(\mathcal{F})$ is a $\mathbb{Z}$-module. This allows us to define a new object: $E(\mathcal{F}) \bigotimes_{\mathbb{Z}} \mathbb{R}$.

Tensoring with $\mathbb{R}$ collapses the torsion subgroup to the 0 element. Let $\mathcal{O} \bigotimes_{\mathbb{Z}} 0$ be the 0 element of $E(\mathcal{F}) \bigotimes_{\mathbb{Z}} \mathbb{R}$. If $t \in T$, t has finite order m. If $m = 0$, $t = \mathcal{O}$, and for any $\lambda \in \mathbb{R}$:

$$\mathcal{O} \bigotimes_{\mathcal{Z}} \lambda$$
$$= 0\mathcal{O} \bigotimes_{\mathbb{Z}} \lambda$$
$$= \mathcal{O} \bigotimes_{\mathbb{Z}} 0\lambda$$
$$= \mathcal{O} \bigotimes_{\mathbb{Z}} 0$$

If $m \neq 0$, then:

$$t \bigotimes_{\mathbb{Z}} \lambda$$
$$= t \bigotimes_{\mathbb{Z}} \frac{m\lambda}{m}$$
$$= mt \bigotimes_{\mathbb{Z}} \frac{\lambda}{m}$$
$$= \mathcal{O} \bigotimes_{\mathbb{Z}} \frac{\lambda}{m}$$
$$= 0\mathcal{O} \bigotimes_{\mathbb{Z}} \frac{\lambda}{m}$$
$$= \mathcal{O} \bigotimes_{\mathbb{Z}} 0$$

Thus, we have $E(\mathcal{F}) \bigotimes_{\mathbb{Z}} \mathbb{R} \cong \mathbb{R}^k$, where k is the rank of $E(\mathcal{F})$. This is therefore a finite-dimensional, real vector space, and the Néron-Tate height pairing function is now a legitimate inner product on this space (Silverman 252). With this in hand, we can perform linear algebra with the elements of $E(\mathbb{F})$, including forming the Gram matrix and taking its determinant. If n is the number of points in the set $S \subseteq E(\mathcal{F})$, there are n^2 of these pairings. We may enumerate the points as $\{P_1, P_2, ..., P_n\} = S$, and define the Néron-Tate matrix, M, as the $n \times n$ matrix whose i, j entry is $\langle P_i, P_j \rangle$. From the definition of $\langle P_i, P_j \rangle$, this is a matrix of inner products, hence it is a Gram matrix, and thus $\det(M) \geqslant 0$.

Silverman defines the *elliptic regulator of* $E(\mathcal{F})$ as the determinant of M, where M is the Néron-Tate matrix for a set of generators of the free abelian subgroup of $E(\mathcal{F})$ (*Arithmetic* 253). And we find the last piece of the puzzle in corollary 9.7 of Silverman's work:

Theorem 7.0.1. *Let E be an elliptic curve over the field $\mathcal{F}$, and let $\{P_1, P_2, ..., P_n\} \subseteq E(\mathbb{F})$ generate the free abelian subgroup of $E(\mathcal{F})$. Then the regulator, $R(E(\mathcal{F}))$, is positive. (Silverman Arithmetic 253)*

If S generates a torsion-free subgroup of $E(\mathcal{F})$, the determinant of the Néron-Tate pairing matrix must be positive. Hence, if the set of points do *not* form a basis for a free abelian group, this determinant will not be positive, but for a Gram matrix, this implies it must be 0. We need only form the Néron-Tate matrix for our set of points and compute its determinant. Thus, what matters is the certainty to which we can measure the determinant.

This last observation may sound like so much skulduggery and chest-beating; the Néron-Tate pairings are defined in terms of canonical heights, and the determinant depends entirely on those pairings. Hence, the only way to compute the determinant is via computation of the canonical heights. It is true that these are needed for the determinant, and finding the canonical height, which is defined as a limit, of a point in a function field is probably too unwieldy for the computation power we currently have. However, if we are not going to calculate the determinant, the next best thing is to *estimate* it. To that end, we will estimate the Néron-Tate pairings using the naive heights and attempt to bound the error.

To bound the difference between canonical and naive heights, recall $\langle P, Q\rangle = \frac{1}{2}(\hat{h}(P+Q) - \hat{h}(P) - \hat{h}(Q))$, and observe that:

$$\begin{aligned} &|\langle P, Q\rangle - \tfrac{1}{2}(h(P+Q) - h(P) - h(Q))| \\ &= \tfrac{1}{2}|\hat{h}(P+Q) - \hat{h}(P) - \hat{h}(Q) - h(P+Q) + h(P) + h(Q)| \\ &\leqslant \tfrac{1}{2}(|\hat{h}(P+Q) - h(P+Q)| + |h(P) - \hat{h}(P)| + |h(Q) - \hat{h}(Q)|) \end{aligned}$$

Hence, if we can place a bound of desired magnitude on $|\hat{h}(P) - h(P)|$, we could potentially use the naive heights to compute the canonicals to any precision.

Let $h(P)$ be the naive height of a point $P \in E(\mathbb{F}_p(t))$. The formula for the $x-$coordinate of

$2P$ is given by:

$$\frac{x^4+x^2+4t^d}{4x^3+x-4t^d}$$

(Ingram, personal correspondence). Let $f(P)$ be the map taking P to the $x-$coordinate of $2 * P$. If we let $x = \frac{a}{b}$, $gcd(a, b) = 1$, then:

$$f(\frac{a}{b}) = \frac{a^4+a^2b^2+ab^4t^d}{4x^3b+xb^3-4b^4t^d}$$

In the numerator, $deg(a^4) \leqslant 4deg(a)$, $deg(a^2b^2) \leqslant 2deg(a) + 2deg(b)$, and $deg(4b^4t^d) \leqslant 4deg(b)+deg(t^d)$. Each of these is bounded by $4\max\{deg(a), deg(b)\}+deg(t^d) = 4h(x)+h(t^d)$. Thus, this is a bound on the degree of the numerator. The denominator is similarly bounded. If the expression reduces, then the degree of the numerator and denominator reduce by equal factors; hence, the overall decrease in degree does not affect this upper bound. Hence:

$$h(f(x)) \leqslant 4h(x) + h(t^d)$$

The second claim we invoke without proof:

$$4h(P) \leqslant h(2P) + 4h(t^d)$$

(Ingram, personal correspondence). Since $h(t^d) = d$, we can write these respectively as:

$$\frac{h(2P)}{4} \leqslant h(P) + \frac{d}{4}$$
$$h(P) \leqslant \frac{h(2P)}{4} + d$$

Because d is positive, these can be rearranged into $-d < \frac{-d}{4} \leqslant h(P) - \frac{1}{4}h(2P) \leqslant d$, which establishes:

Lemma 1. $|h(P) - \frac{1}{4}h(2P)| \leqslant d$

This bound between $h(P)$ and $\frac{1}{4}h(2P)$ allows us to go further.

$$
\begin{aligned}
&|h(P) - \hat{h}(P)| \\
&= |h(P) - \lim_{n\to\infty} \tfrac{h(2^n P)}{4^n}| \\
&= |\sum_{n=0}^{\infty} (\tfrac{1}{4^n} h(2^n P) - \tfrac{1}{4^{n+1}} h(2^{n+1} P))| \\
&= |\sum_{n=0}^{\infty} \tfrac{1}{4^n} (h(2^n P) - \tfrac{1}{4}(h(2^{n+1} P)))| \\
&\leqslant \sum_{n=0}^{\infty} \tfrac{1}{4^n} |h(2^n P) - \tfrac{1}{4} h(2^{n+1} P)| \\
&\leqslant \sum_{n=0}^{\infty} \tfrac{1}{4^n} |d| \\
&= d \sum_{n=0}^{\infty} \tfrac{1}{4^n} \\
&= \tfrac{4d}{3}
\end{aligned}
$$

Apply this to the original subtraction between $\langle P, Q \rangle$ and the naive heights.

$$
\begin{aligned}
&|\langle P, Q \rangle - \tfrac{1}{2}(h(P+Q) - h(P) - h(Q))| \\
&\leqslant \tfrac{1}{2}(|\hat{h}(P+Q) - h(P+Q)| + |h(P) - \hat{h}(P)| + |h(Q) - \hat{h}(Q)|) \\
&\leqslant \tfrac{1}{2}(\tfrac{4d}{3} + \tfrac{4d}{3} + \tfrac{4d}{3}) \\
&= 2d
\end{aligned}
$$

Finally, Theorem 6.2 implies that $\hat{h}(2^j P) = 4^j \hat{h}(P)$, $j \in \mathbb{Z}^+$. Observe:

$$
\begin{aligned}
|\tfrac{1}{2}\hat{h}(2^j P) - \tfrac{1}{2} h(2^j P)| &= \tfrac{1}{2}|\hat{h}(2^j P) - h(2^j P)| \leqslant \tfrac{2d}{3} \\
\tfrac{1}{4^j}\tfrac{1}{2}|\hat{h}(2^j P) - h(2^j P)| &\leqslant \tfrac{2d}{3 * 4^j} \\
|\tfrac{1}{4^j}\tfrac{1}{2} * 4^j \hat{h}(P) - \tfrac{1}{2}\tfrac{1}{4^j} h(2^j P)| &\leqslant \tfrac{2d}{3 * 4^j} \\
|\tfrac{1}{2}\hat{h}(P) - \tfrac{1}{2}\tfrac{1}{4^j} h(2^j P)| &\leqslant \tfrac{2d}{3 * 4^j}
\end{aligned}
$$

This gives us a second lemma:

Lemma 2. $|\hat{h}(P) - \tfrac{1}{4^j} h(2^j P)| = 2 * \tfrac{1}{2}|\hat{h}(P) - \tfrac{1}{4^j} h(2^j P)| \leqslant \tfrac{4d}{3 * 2^j}$

Generalizing this to the Néron-Tate pairing gives:

$$
|\langle P, Q \rangle - \tfrac{1}{2}\tfrac{1}{4^j}(h(2^j P + 2^j Q) - h(2^j P) - h(2^j Q))|
$$

$$\leqslant |\tfrac{1}{2}\hat{h}(P+Q) - \tfrac{1}{2}\tfrac{1}{4^j}h(2^jP+2^jQ)| + |\tfrac{1}{2}\hat{h}(P) - \tfrac{1}{2}\tfrac{1}{4^j}h(2^jP)| + |\tfrac{1}{2}\hat{h}(Q) - \tfrac{1}{2}\tfrac{1}{4^j}h(2^jQ)|$$
$$\leqslant 3(\tfrac{2d}{3*4^j}) = \tfrac{2d}{4^j}$$

Since the real numbers are Archimedean, for any $\epsilon > 0$, there is a sufficiently large positive integer j depending on d such that $\frac{2d}{4^j} < \epsilon$. And thus, for this same j, we have:

$$|\langle P, Q\rangle - \tfrac{1}{2}\tfrac{1}{4^j}(h(2^jP+2^jQ) - h(2^jP) - h(2^jQ))| < \epsilon$$

Ergo, every element of the Néron-Tate pairing matrix can be computed to arbitrary precision with the appropriate choice of j. We can compute the minimum required whole number, j, as:

$$\epsilon > \tfrac{2d}{4^j}$$
$$4^j > \tfrac{2d}{\epsilon}$$
$$\log 4^j > \log \tfrac{2d}{\epsilon}$$
$$j \log 4 > \log 2d - \log \epsilon$$
$$j > \tfrac{\log 2d - \log \epsilon}{\log 4}$$

Thus, j is the smallest nonnegative integer greater than the right-hand quantity.

All of this assumes that we have generated a nonempty set of points, either randomly, brute forcefully, or cleverly by noting p and n satisfy certain conditions. For this set of points, we want to know if they form a linearly independent set or not. Moreover, knowing their canonical height beforehand would safeguard against adding any points of finite order. This leads to the next series of algorithms in the program, beginning with height_estimate.

```
def height_estimate(P, powers):
    m = powers[0]
    scale = powers[1]
    approximation = naive_height(m*P)/scale
    return float(approximation)

def naive_height(P):
    if P.is_zero():
        return 0
    x = P.xy()[0]
    return max(x.numerator().degree(), x.denominator().degree())

def canonical_approximation(P, Q, powers):
    m = powers[0]
    scale = powers[1]
    P_2 = m*P
    height_of_P = naive_height(P_2)
    Q_2 = m*Q
    height_of_Q = naive_height(Q_2)
    height_of_sum = naive_height(P_2 + Q_2)
    approximation = float((height_of_sum - height_of_P - height_of_Q))/(2*scale)
    return approximation
```

Figure 7.1: The different functions using a variation of height.

When a point is generated in the main function, it calls the height_estimate function to estimate the canonical height. It does this by using the powers of 2 and 4, encoded in a set passed to the function, required to minimize the difference between naive and canonical height. It calls the naive_height function with the appropriate multiple and passes $\frac{1}{4^n}h(2^nP)$ back to the main function. If this naive height falls below the user-set error bound, we cannot be appropriately confident that the canonical height is positive, and the point is eliminated as a potential generator. For convenience and as a conservative measure, we take the powers associated with the j computed above.

The matrix_build algorithm takes as inputs a nonempty set of points in $E(\mathbb{F}_p(t))$, the already estimated canonical heights of the points, and the power of 2 and 4 needed to estimate the other entries. The algorithm creates a real-valued $n \times n$ matrix A, n the number of points

in the set in question, and populates it with approximations of the Néron-Tate height pairs. Because the matrix is symmetric, for row i from 1 to n, we need only compute the entries a_{ij} for j from 1 to i, since $a_{ji} = a_{ij}$.

If $i \neq j$, the entry is off the main diagonal, and the program will estimate a_{ij} via the function canonical_approximation, which approximates the pairing as $\frac{1}{2}\frac{1}{4^j}(h(2^jP + 2^jQ) - h(2^jP) - h(2^jQ))$, h the naive height. If $i = j$, the entry is on the main diagonal, and the program estimates $\langle P, P \rangle$ as $\frac{1}{2}\frac{1}{4^j}(h(2^jP + 2^jP) - h(2^jP) - h(2^jP) = \frac{1}{2}\frac{1}{4^j}(4h(2^jP) - 2h(2^jP)) = \frac{1}{4^j}(h(2^jP)$, which is just the height_estimate associated with P.

```
def matrix_build(set_of_points, known_canonical_estimates, powers):
    list_of_points = list(set_of_points) #turn the set of points into a list
    m = len(list_of_points)
    height_pairing_matrix = Matrix(RR, m)
    for i in range(m):
        for j in range(i + 1):
            Pi = list_of_points[i]
            if i != j:
                Pj = list_of_points[j]
                height_pairing_matrix[i, j] = canonical_approximation(Pi, Pj, powers)
                height_pairing_matrix[j, i] = height_pairing_matrix[i, j]
            else:
                height_pairing_matrix[i, i] = known_canonical_estimates.get(Pi,
                    height_estimate(Pi, powers))
    det = height_pairing_matrix.determinant()
    return height_pairing_matrix, det
```

Figure 7.2: The function to build the Néron-Tate matrix to test the set of points as a generator of the Mordell-Weil group's free abelian subgroup.

Once we have the matrix of Néron-Tate pairings, the last step is to take the determinant and see if the set consists of generators of $E(\mathbb{F}_p(t))$. However, because the Néron-Tate pairs were estimated, there is an error associated with them, which will then be compounded by the formula for the determinant. The question now becomes how do we bound the error on that determinant.

Let $A \in M_n(\mathbb{R})$, where each entry a_{ij} has a maximum error of $|\delta(a_{i,j})| \leqslant \epsilon$. We wish to bound the error on $\det(A)$. $\det(A)$ is a function from $\mathbb{R}^{n^2}$ into $\mathbb{R}$. In general, if f is a real-valued function from $D \subseteq \mathbb{R}^m$ into $\mathbb{R}$, and let $\overrightarrow{x} \in D$ with error $(\overrightarrow{\delta(x)})$. Define the line segment $\overrightarrow{x} + t\overrightarrow{\delta(x)}$ for $t \in [0,1]$, and assume it is in D. Let f be smooth on the line segment. The mean value theorem for multivariate functions gives, for some $t^* \in [0,1]$:

$$f(\overrightarrow{x} + \overrightarrow{\delta(x)}) - f(\overrightarrow{x}) = \nabla f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)}) \cdot (\overrightarrow{x} + \overrightarrow{\delta(x)} - \overrightarrow{x})$$

where ∇ is the gradient on f and $\cdot$ is the standard dot product on $\mathbb{R}^m$. 1 and ∞ are conjugate exponents, since $\frac{1}{1} + \frac{1}{\infty} = 1$, $\frac{1}{\infty} = 0$ by convention. Then by Hölder's inequality:

$$\begin{aligned} &= |f(\overrightarrow{x} + \overrightarrow{\delta(x)}) - f(\overrightarrow{x})| \\ &= |\nabla f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)}) \cdot (\overrightarrow{x} + \overrightarrow{\delta(x)} - \overrightarrow{x})| \\ &\leqslant ||\nabla f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)})||_\infty ||\overrightarrow{\delta(x)}||_1 \end{aligned}$$

where $||\nabla f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)})||_\infty = \max\limits_{1 \leqslant i \leqslant n^2} |\frac{\partial f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)})}{\partial x_i}|$ and $||\overrightarrow{\delta(x)}||_1 = \sum\limits_{i=1}^{n^2} |\delta(x_i)|$. As a conservative bound, $||\nabla f(\overrightarrow{x} + t^*\overrightarrow{\delta(x)})||_\infty \leqslant \sup\limits_{t \in [0,1]} ||\nabla f(\overrightarrow{x} + t\overrightarrow{\delta(x)})||$

For an $n \times n$ matrix A, $\det(A) = \sum_{k=1}^{n} a_{ik} C_{ik}$, where i is an arbitrary row number and cofactor $C_{ik} = (-1)^{i+k} \det(M_{ik})$, where M_{ik} is the $(n-1) \times (n-1)$ submatrix of A with the ith row and kth column deleted. To compute $\frac{\partial \det(A)}{\partial a_{ij}}$, compute $\det(A)$ along row i. Notice that every term in the series but $a_{ij}C_{ij}$ is constant with respect to a_{ij}. Taking the partial derivative with respect to a_{ij} eliminates every term in the series except $a_{ij}C_{ij}$, since every other is a different entry from a_{ij} multiplied by a cofactor from which a_{ij} is absent. The partial derivative of $a_{ij}C_{ij}$ with respect to a_{ij} is just C_{ij}. Hence, $\nabla \det(A)$ is the n^2 vector of these cofactors.

Let f be the map sending A to $\det(A)$, and $\delta(A)$ the $n \times n$ matrix of errors on the entries. We assume f is smooth on the line segment between A and $A + \delta(A)$. To bound the error on $||\delta(A)||_1$, let ϵ be the maximum error on the entries of A. We then have:

$$
\begin{aligned}
&||\delta(A)||_1 \\
&= \textstyle\sum_{1 \leqslant i,j \leqslant n} |\delta(a_{ij})| \\
&\leqslant \textstyle\sum_{1 \leqslant i,j \leqslant n} \epsilon \\
&= n^2\epsilon
\end{aligned}
$$

To bound the error on $||\nabla \det(A + t\delta(A))||_\infty$, $t \in [0,1]$, notice that $\nabla \det(A)$ is just the n^2 cofactors of A, so the norm is the maximum of their absolute values. The absolute value of each cofactor is the absolute value of the determinant of an $(n-1) \times (n-1)$ matrix. Using the formal definition of an $(n-1) \times (n-1)$ determinant is:

$$\det(A) = \textstyle\sum_{\sigma \in S_{(n-1)}} sgn(\sigma) \prod_{i=1}^{n-1} a_{i,\sigma(i)}$$

If each entry a_{ij} is bounded by $B \in \mathbb{R}$ on the line segment, then for $t \in [0,1]$, we have: $|a_{ij} + t\delta(a_{ij})| \leqslant |a_{ij}| + t|\delta(a_{ij})| \leqslant B + \epsilon = B^*$. Then $|a_{ij} + t\delta(a_{ij})| \leqslant B^*$, and:

$$
\begin{aligned}
\sup_{t \in [0,1]} ||\nabla \det(A + t\delta(A))||_\infty &\leqslant \textstyle\sum_{\sigma \in S_{n-1}} |sgn(\sigma) \prod_{i=1}^{n-1} B^*| \\
&\leqslant \textstyle\sum_{\sigma \in S_{n-1}} |\prod_{i=1}^{n-1} B^*| \\
&= \textstyle\sum_{\sigma \in S_{n-1}} (B^*)^{n-1} \\
&= (n-1)!(B^*)^{n-1}
\end{aligned}
$$

since there are $(n-1)!$ elements in the group of permutations, S_{n-1}. Thus, we have:

$$||\nabla \det(A + t\delta(A))||_\infty ||\delta(A)||_1 \leqslant ((n-1)!(B^*)^{n-1})(n^2\epsilon) = n\epsilon n!(B^*)^{n-1}$$

We now have all the pieces for a numerical examination of Ulmer's lower bounds. Let us run the simulation, crunch the numbers, and see what we can learn from the results.

Chapter 8: Results and Further Development

Having run both the random_rational and the brute_force algorithm multiple times, some results are tabulated below.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{\frac{t^3+t+1}{t^2+1}, \frac{t^3+1}{t^3}\}$	3.0 ± 0.001	3.0 ± 0.001	1

Table 8.1: Subset of regulator found on $y^2 + xy = x^3 - t^3$ over $\mathbb{F}_2(t)$, $n = 1$, $d = 3$.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{(t^2, t^3)\}$	0.800 ± 0.001	0.800 ± 0.001	1

Table 8.2: Subset of regulator found on $y^2 + xy = x^3 - t^5$ over $\mathbb{F}_2(t)$, $n = 2$, $d = 5$.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{(t^3 + t^2, t^4), (t^3, 0\}$	1.333 ± 0.006	$1.444 \pm 0.001, 1.000 \pm 0.001$	2

Table 8.3: Subset of regulator found on $y^2 + xy = x^3 - t^9$ over $\mathbb{F}_2(t)$, $n = 3$, $d = 9$.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{(t^2, 2t^3 + t^2)\}$	1.00 ± 0.001	1.00 ± 0.001	1

Table 8.4: Subset of regulator found on $y^2 + xy = x^3 - t^4$ over $\mathbb{F}_3(t)$, $n = 1$, $d = 4$.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{(t^4, 2t^6)\}$	1.60 ± 0.001	1.60 ± 0.001	2

Table 8.5: Subset of regulator found on $y^2 + xy = x^3 - t^{10}$ over $\mathbb{F}_3(t)$, $n = 2$, $d = 10$.

Points in Regulator	Néron-Tate Matrix Determinant and Maximum Error	Estimated Canonical Heights and Errors	Ulmer's Lower Bound
$\{(0, 2t^3), (t^2, 0)\}$	3.00 ± 0.013	0.500 ± 0.001, 0.333 ± 0.003	2

Table 8.6: Subset of regulator found on $y^2 + xy = x^3 - t^6$ over $\mathbb{F}_5(t)$, $n = 1$, $d = 6$.

Of particular note here is that none of the sets produced had greater cardinality than the lower bound given in Theorem 3.1. In all instances but one was this bound met, and in the exception the algorithm did not produce the two elements the theorem said were required. However, it is also noteworthy that, for all runs where the approximate canonical heights were displayed, *all of the points had non-zero canonical heights.*

This suggests three things. Firstly, Ulmer's lower bound of $\frac{p^n-1}{2n}$ for the Mordell-Weil group of $y^2 + xy = x^3 - t^d$ over $\mathbb{F}_p(t)$, $d = p^n + 1$, is equality for at least small prime fields, like $\mathbb{F}_2$ and $\mathbb{F}_3$. Secondly, and closely related to this, is that the integer span of a single point $P \in E(\mathbb{F}_p(t))$ of non-zero canonical height will be almost all of $E(\mathbb{F}_p(t))$. Put differently, for a randomly selected element $Q \in E(\mathbb{F}_p(t))$, there will all but certainly be an integer n such that $nP = Q$. Thirdly, and finally, is that the torsion subgroup of $E(\mathbb{F}_p(t))$ is very small, perhaps trivial.

So where does this leave our narrative? We have opened the door a crack, and can already see a myriad of possibilities. Indeed, there are several directions to take the show from here. The first and most obvious is *more*; get *more* points and run the algorithms for *more* time. One obvious way to support this requirement for more is to choose substance over style;

apply *more effective* algorithms. Computing power is a bottleneck that can condemn the programs to die and fail, sometimes without all but the most diligent examination.

Leaving aside computing power, the question of the torsion subgroup's composition still lingers. Mazur's theorem for elliptic curves over $\mathbb{Q}$ puts considerable restrictions on the torsion subgroup (Silverman *Arithmetic* 242), but it is unknown if a similar theorem applies to the Ulmer curves with which we have grown so familiar.

One limitation of this project is that the only elliptic curves examined were those over the field $E(\mathbb{F}_p(t))$, p a prime, where the constant term t was of power $d = p^n + 1$. In fact, Ulmer's result is far more expansive than this, holding for elliptic curves over $E(\mathbb{F}_q(t))$, q a power of p, and d a divisor of $p^n + 1$. Adapting the algorithms used here to this more general result could potentially pay dividends, either by confirming similar results suggested here or generating an elliptic curve and field where Ulmer's lower bound is strictly less than the rank.

The methods developed here demonstrate that, even in function fields, computational experimentation can offer meaningful evidence toward rank predictions for families of elliptic curves. Our results are consistent with existing conjectures, but further work could extend the scope of testing, refine point-generation algorithms, and integrate rank determination techniques. Future investigations may also explore analogous curve families over different base fields, as well as potential connections to the Birch and Swinnerton-Dyer conjecture in the function field context.

Works Cited

Ingram, Patrick M. Written notes. July 18, 2025.

Liu, Tianxiang. “The L-Functions of Elliptic Curves and Modular Forms." The University of Rochester, 2022, www.sas.rochester.edu/mth/undergraduate/honorspaperspdfs/tianxiangliu2022.pdf.

Silverman, Joseph H. *The Arithmetic of Elliptic Curves*. Springer, 2009.

Silverman, Joseph H. “Canonical Heights on Abelian Varieties." Arizona Winter School, 2–6 Mar. 2024. Online PDF.

Silverman, Joseph H., and John T. Tate. *Rational Points on Elliptic Curves*. Springer, 2015.

Ulmer, Douglas. “Elliptic Curves and Analogies Between Number Fields and Function Fields." 22 May 2003, arxiv.org/pdf/math/0305320. *Heegner Points and L-Series*, MSRI Publications, vol. 48.

Ulmer, Douglas. “Elliptic Curves with Large Rank over Function Fields.” *Annals of Mathematics*, vol. 155, 2002, pp. 295–315.

Appendix A: SageMath Code

```
from sage.arith.m\clearpage
\appendix
\addcontentsline{toc}{chapter}{Appendices} % <-- adds the "Appendices" line to TOC
isc import gcd
from sage.all import ZZ
from random import gauss
from sage.functions.log import log
from sage.functions.other import ceil #various packages loaded in to perform necessary
    computations
import sys

def setup_function_field(p, n):
    d = p^n + 1 #determines the exponent of the constant term of the elliptic curve
    R.<t> = PolynomialRing(GF(p)) #defines the ring of polynomials in 't' over the Galois
        field GF(p)
    FracR = R.fraction_field() #defines the fraction field associated with R
    PR.<y> = PolynomialRing(FracR) #defines the ring of polynomials in 'y' over the
        aforementioned fraction field
    return d, R, FracR, PR, t, y

def canonical_fraction(a, FracR):
    # Convert to the *same* fraction field
    a = FracR(a) #coerce 'a' into the appropriate fraction field
    num = a.numerator()
    den = a.denominator()
    if den == 0:
        raise ZeroDivisionError("Denominator is zero")
    R = den.parent() # Work in the polynomial ring of den
    num = R(num); den = R(den)
    # Cancel gcd
    g = gcd(num, den)
    if g != 1:
        num //= g
        den //= g
    # Make denominator monic
    lc = den.leading_coefficient()
    if lc != 1:
        inv = 1 / lc
        num *= inv
        den *= inv
```

```
    return FracR(num) / FracR(den) #ensure both numerator and denominator belong to the same fraction field

def check_discriminant(a, t, d, p):
    if p ==2:
        return True
    disc = a^2 + 4*a^3 - 4*t^d
    return disc.is_square()
    #for a given function to be the x-coordinate of a point on the curve, the y-coordinate must be a rational function in t:
    #the resulting equation E(a, y) must therefore have a determinant that is a perfect square in F_p(t).

def solve(a, p, t, d, PR, y, E, set_of_points, j, FracR):
    try:
        rhs = y^2 + y*a - a^3 + FracR(t)^d #ensure t^d is in FracR
        poly_y = PR(rhs) #ensure the polynomial belongs to the appropiate polynomial ring
        roots_y = poly_y.roots(multiplicities=False) #attempt to solve the associated equation for y, irrespecitve of multiplicity
        if roots_y:
            raw_y = roots_y[0] #extract the first solution
            y_value = canonical_fraction(raw_y, FracR) #reduce y over FracR
            P = E(a, y_value) #find a point on the elliptic curve corresponding to those values
            if P not in set_of_points:
                set_of_points.add(P)
                print(f"iteration {j}: ({a}, {y_value})\n")
    except (TypeError, ValueError) as e:
        # Uncomment this to debug:
        # print(f"Error for a = {a}: {e}")
        pass

def brute_force_algorithm(checked_set, p, t, max_degree, FracR):
    rationals_to_check = set()
    polynomials = set()
    for degree in range(0, max_degree + 1):
        for m in range(p ** (degree + 1)):
            current_polynomial = 0
            mm = m
            for k in range(degree + 1):
                coeff = mm % p
                current_polynomial += coeff * (t ** k)
                mm //= p
            polynomials.add(current_polynomial)
    for numerator in polynomials:
```

```
        for denominator in polynomials:
            if denominator == 0:
                continue
            rational = canonical_fraction(numerator / denominator, FracR)
            if (rational not in checked_set) and (rational not in rationals_to_check):
                rationals_to_check.add(rational)
    return rationals_to_check

def random_rational(p, d, x, t, FracR):
    seed = floor(gauss(d*x, d*x/3))
    while seed < 0:
        seed = floor(gauss(d*x, d*x/3)) #repeat generating a random Gaussian until a
            non-negative value is achieved
    t_powers = [t**j for j in range(seed + 1)] #the generated Gaussian determines the power
        of the polynomials of the rational function
    N = [GF(p).random_element() for j in range(seed + 1)] #creates a list of 'n' randomly
        generated elements of GF(p): the seeded value becomes the
    #maximum degree of the numerator and denominator
    D = [GF(p).random_element() for k in range(seed + 1)] #repeat the process to generate
        another set of elements in GF(p)
    num = sum([t_powers[j] * N[j] for j in range(seed + 1)]) #generates an 'n'-degree
        polynomial in 't' with those coefficients for the numerator
    den = sum([t_powers[j] * D[j] for j in range(seed + 1)]) #creates another 'n'-degree
        polynomial in 't' for the denominator
    if den != 0:
        return canonical_fraction(num / den, FracR) #returns a rational function in 't' as
            the ratio of the two polynomials, provided the denominator is nonzero
    else:
        raise ValueError #throw an error if the denominator is zero

def matrix_build(set_of_points, known_canonical_estimates, powers): #the main heavy-lifting
    function that distributes most of the work
    #print("entered height_approximation")
    list_of_points = list(set_of_points) #turn the set of points into a list
    m = len(list_of_points)
    height_pairing_matrix = Matrix(RR, m) #create a square height-pairing matrix of real
        numbers
    for i in range(m):
        for j in range(i + 1): #since the matrix is symmetric, we need only work with
            elements that are on or below the main diagonal, and then copy
            #any below the diagonal to its transposes
            Pi = list_of_points[i]
            if i != j:
                Pj = list_of_points[j]
```

```
                height_pairing_matrix[i, j] = canonical_approximation(Pi, Pj, powers) #if i
                    and j are different indices, pass them to the canonical
                #height approximator
                height_pairing_matrix[j, i] = height_pairing_matrix[i, j] #set the (j, i)
                    entry to be the same as its transpose, (i, j)
            else:
                height_pairing_matrix[i, i] = known_canonical_estimates.get(Pi,
                    height_estimate(Pi, powers)) #use the estimated canonical height of
                #P to estimate the diagonal entry <P, P>
    det = height_pairing_matrix.determinant() #find the determinant of the height matirx
    return height_pairing_matrix, det #return the matrix and its determinant back to the
        main function

def determinant_error(M, eps):
    m = M.nrows()
    max_bound = max(abs(M[i, j]) for i in range(m) for j in range(0, i+1))
    B_star = max_bound + eps
    return m*eps*factorial(m)*(B_star**(m-1))

def height_estimate(P, powers):
    m = powers[0]
    scale = powers[1]
    approximation = naive_height(m*P)/scale #estimate the canonical height of P with the
        naive height h(2^jP)/4^j
    return float(approximation)

def naive_height(P):
    if P.is_zero():
        return 0
    x = P.xy()[0]
    return max(x.numerator().degree(), x.denominator().degree()) #return 0 or the maximum
        degree

def canonical_approximation(P, Q, powers):
    m = powers[0]
    scale = powers[1]
    P_2 = m*P
    height_of_P = naive_height(P_2)
    Q_2 = m*Q
    height_of_Q = naive_height(Q_2)
    height_of_sum = naive_height(P_2 + Q_2)
    approximation = float((height_of_sum - height_of_P - height_of_Q))/(2*scale)
        #approximate <P, Q> using
    #(h(2^jP + 2^jQ) - h(2^jP) - h(2^jQ))/(2*4^j)
    return approximation
```

```
150
151 def check_subsets(set_of_points, powers, known_canonical_estimates, error):
152     #function is called when a set of points has been generated, but it has failed to be
        linearly independent.
153     #the canonical heights of all points is already known, and all place the respective
        points outside the torsion subgroup
154     linearly_independent_set = set()
155     for P in (set_of_points):
156         set_to_check = linearly_independent_set | {P}
157         temp_height, temp_det = matrix_build(set_to_check, known_canonical_estimates, powers)
158         temp_error = determinant_error(temp_height, error)
159         if abs(temp_det) > temp_error and abs(temp_det) > 0.001:
160             #recursively add points to the set until a maximal set is reached.
161             linearly_independent_set.add(P)
162     if linearly_independent_set:
163         Neron_Tate_matrix, det = matrix_build(linearly_independent_set,
            known_canonical_estimates, powers)
164         max_error = determinant_error(Neron_Tate_matrix, error)
165         print("The set:")
166         for P in (linearly_independent_set):
167             print(f"({P.xy()[0]}, {P.xy()[1]}), ")
168             print()
169         print(f"forms a Neron-Tate matrix with determinant {det} to within an error of
            {max_error}.")
170         print("It likely belongs to the generators of the free abelian group.")
171         #give the determinant of the resulting Neron-Tate matrix.
172     else:
173         print("None of the points form a Neron-Tate matrix with nonzero determinant to
            within the accepted error.")
174         print("It is likely they are all elements of the torsion subgroup.")
175         #alert the user that no such subset was found
176
177 def main(p, n, sample_size, max_error, mu, max_degree, show):
178     checked_rationals = set()
179     known_canonical_estimates = {}
180     initial_points = set()
181     powers = []
182     p = ZZ(p) #ensure Sage reads p as an integer
183     set_of_points = set() #set that will store the generated rational functions that
        qualify as x-coordinates of a point on E
184     d, R, FracR, PR, t, y = setup_function_field(p, n) #set up the function field
185     print(f"Field: GF({p})(t), d = {d}\n")
186     E = EllipticCurve(FracR, [1, 0, 0, 0, -t^d]) #define the elliptic curve
187     print(E)
188     print()
```

```
    if p%3 == 2 and n%2 == 1:
        exponent = d//3
        P = E(t**exponent, 0)
        initial_points.add(P)
        print(f"The point ({P.xy()[0]}, {P.xy()[1]}) is part of the Mordell-Weil group.")
        print()
    if p%4 == 1:
        u = GF(p)(-1).sqrt()
        exponent = d//2
        P = E(0, u*t**exponent)
        initial_points.add(P)
        print(f"The point ({P.xy()[0]}, {P.xy()[1]}) is part of the Mordell-Weil group.")
        print()
        #these 'if' statements generate known points on the elliptic curve
    for P in initial_points:
        checked_rationals.add(P.xy()[0]) #add these x-coordinate functions to the set of
            rationals that do not need to be checked for solutions
    print(f"Points are being randomly generated. The powers form a normal distibution of
        mean {float(d*mu)} and standard deviation {float(d*mu/3)}.")
    print()
    for j in range(sample_size):
        try:
            a = random_rational(p, d, mu, t, FracR) #generate a random rational of degree
                determined by user input
            if a in checked_rationals:
                continue #skip any that are in the set checked_rationals (i.e. that don't
                    need to be/have already been checked)
            checked_rationals.add(a)
            if check_discriminant(a, t, d, p): #check if the discriminant is a perfect
                square in the function field
                solve(a, p, t, d, PR, y, E, set_of_points, j, FracR) #if it is, attempt to
                    solve for the associated y-function
        except ValueError:
            pass
    if not set_of_points:
        print(f"No solutions for the elliptic curve y^2 + x*y = x^3 - t^{d} found via a
            random search.")
        print()
        print(f"Switching to brute-force algorithm check of rationals up to degree
            {max_degree}.")
        to_check_rationals = brute_force_algorithm(checked_rationals, p, t, max_degree,
            FracR)
        to_check_rationals = list(to_check_rationals) #create a list of rationals generated
            by the brute_force function
        for j in range(len(to_check_rationals)):
```

```
            try:
                solve(to_check_rationals[j], p, t, d, PR, y, E, set_of_points, j, FracR)
                    #attempt to find the y-value associated with each x-value
            except ValueError:
                pass
    set_of_points = set_of_points | initial_points #add any known points to the updated
        set_of_points
    if not set_of_points:
        print(f"No solutions: the Mordell-Weil group of y^2 + x*y = x^3 - t^{d} over
            GF({p})(t) has points with x-coordinates of degree >{max_degree}.")
    else:
        C = 2*d/3
        numerator = log(2*d) - log(max_error)
        denominator = log(4)
        steps = numerator/denominator
        if steps <0:
            steps = 0
        elif steps in ZZ:
            steps = steps + 1
        else:
            steps = Integer(ceil(numerator/denominator)) #compute the power needed to bring
                the naive height to within a specified range of its
            #canonical height
        power_of_two = 2**steps
        power_of_four = 4**steps
        powers.insert(0, power_of_two)
        powers.insert(1, power_of_four) #powers of 2 and 4 needed for the canonical height
            approximation
        if show:
            print("The approximate canonical heights are as follows:")
        points_to_keep = []
        for P in set_of_points:
            approximate_canonical_height = height_estimate(P, powers)
            print(f"For point ({P.xy()[0]}, {P.xy()[1]}):")
            if show:
                print(f"the canonical height is approximately
                    {approximate_canonical_height}, to an error of {max_error}.")
            if approximate_canonical_height > max_error:
                print("The estimated canonical height is large enough that the point lies
                    outside the torsion subgroup.")
                points_to_keep.append(P)
                known_canonical_estimates[P] = approximate_canonical_height
            else:
                print("The estimated canonical height is not large enough to conclude the
                    point lies outside the torsion subgroup.")
```

```
            #if the canonical heights are clearly non-zero, keep the points. Else, remove them as a potential element of the generating set
        set_of_points = set(points_to_keep)
        height_matrix, det = matrix_build(set_of_points, known_canonical_estimates, powers) #build the Neron-Tate height-pairing matrix
        error = determinant_error(height_matrix, max_error) #compute the error on the determinant
        print(f"The determinant of the Neron-Tate matrix was estimated to be {det} to within an error of: {error}")
        if abs(det) >0.001 and abs(det)>error: #check if the determinant is to within the desired error and absolute value
            print()
            print("The set of points are likely generators of the free abelian subgroup.")
        else:
            print()
            print("We cannot conclude that the set is linearly independent to within the estimated error.")
            if len(set_of_points) > 1:
                print("Now checking subsets for linear independence:")
                print()
                check_subsets(set_of_points, powers, known_canonical_estimates, max_error)
                #if the determinant meets those conditions, the set is probably linearly independent.
                #if not, attempt to build the maximal linear independent subset.
            else:
                print(f"The point likely belongs to the torsion subgroup.")
```