HEIGHTS ON ELLIPTIC CURVES OVER NUMBER FIELDS, PERIOD LATTICES, AND COMPLEX ELLIPTIC LOGARITHMS

By
Thotsaphon Thongjunthug

A thesis submitted for the degree of
Doctor of Philosophy

Mathematics Institute
The University of Warwick
Coventry, England

February 2011
Contents

Acknowledgements vii
Declaration viii
Summary ix

1 Introduction 1
  1.1 An Overview of Elliptic Curves . . . . . . . . . . . . . . . . . . . 1
  1.2 Elliptic Curves over Number Fields . . . . . . . . . . . . . . . . . 4
    1.2.1 Heights . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    1.2.2 Mordell–Weil Bases . . . . . . . . . . . . . . . . . . . . . . . 8
  1.3 Elliptic Curves over \( \mathbb{C} \) . . . . . . . . . . . . . . . . . . . 9

2 Height Bound I 13
  2.1 Points of Good Reduction . . . . . . . . . . . . . . . . . . . . . . . 13
  2.2 Estimation of Local Heights . . . . . . . . . . . . . . . . . . . . . . 15
    2.2.1 Non-Archimedean Cases . . . . . . . . . . . . . . . . . . . . . 15
    2.2.2 Archimedean Cases . . . . . . . . . . . . . . . . . . . . . . . . 18
  2.3 Multiplication by \( n \) . . . . . . . . . . . . . . . . . . . . . . . . . . 18
  2.4 A Bound for Multiples of Points of Good Reduction . . . . . . . . . 20
  2.5 Solving Inequalities I: Real Embeddings . . . . . . . . . . . . . . . 23
    2.5.1 Periods and Elliptic Logarithms . . . . . . . . . . . . . . . . . 23
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3  Height Bound II: Complex Embeddings</strong></td>
<td></td>
</tr>
<tr>
<td>3.1 Corresponding Regions I: An Overview</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Fundamental Parallelograms</td>
<td>29</td>
</tr>
<tr>
<td>3.1.2 Visualising the Region</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Corresponding Regions II: Estimation</td>
<td></td>
</tr>
<tr>
<td>3.2.1 The Weierstrass ( \wp )-function</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Interval Arithmetic</td>
<td>32</td>
</tr>
<tr>
<td>3.2.3 Approximate Corresponding Regions</td>
<td>34</td>
</tr>
<tr>
<td>3.3 Solving Inequalities II: Complex Embeddings</td>
<td>38</td>
</tr>
<tr>
<td>3.4 An Algorithm for Height Bound</td>
<td>41</td>
</tr>
<tr>
<td>3.5 Remarks</td>
<td>43</td>
</tr>
<tr>
<td><strong>4  Period Lattices and Complex Elliptic Logarithms</strong></td>
<td>46</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>46</td>
</tr>
<tr>
<td>4.2 AGM Sequences</td>
<td>48</td>
</tr>
<tr>
<td>4.3 Chains of Lattices</td>
<td>51</td>
</tr>
<tr>
<td>4.3.1 Optimal Chains and Rectangular Lattices</td>
<td>54</td>
</tr>
<tr>
<td>4.4 Chains of 2-Isogenies</td>
<td>57</td>
</tr>
<tr>
<td>4.5 Period Lattices of Elliptic Curves</td>
<td>61</td>
</tr>
<tr>
<td>4.5.1 General Case</td>
<td>62</td>
</tr>
<tr>
<td>4.5.2 Special Case I: Rectangular Lattices</td>
<td>68</td>
</tr>
<tr>
<td>4.5.3 Special Case II</td>
<td>69</td>
</tr>
<tr>
<td>4.5.4 A Relationship amongst the Periods</td>
<td>70</td>
</tr>
<tr>
<td>4.6 Complex Elliptic Logarithms</td>
<td>75</td>
</tr>
<tr>
<td>4.7 Examples</td>
<td>84</td>
</tr>
<tr>
<td><strong>5  Applications</strong></td>
<td>95</td>
</tr>
<tr>
<td>5.1 Height Bound III: Examples</td>
<td>95</td>
</tr>
<tr>
<td>5.1.1 Case I: Totally Real Number Fields</td>
<td>96</td>
</tr>
</tbody>
</table>
## Contents

5.1.2 Case II: Number Fields with Complex Embeddings  ... 101
5.2 Mordell–Weil Bases .................................. 106
  5.2.1 Sieving Procedure ................................. 108
  5.2.2 Examples Revisited ............................... 109
  5.2.3 Comparison with a Searching Points Method ...... 115
5.3 Integral Points on Elliptic Curves .................. 116
  5.3.1 Introduction .................................... 116
  5.3.2 Initial Bounds .................................. 117
  5.3.3 Bound Reduction ................................. 120
  5.3.4 Examples ...................................... 122
5.4 Elliptic Curves with Everywhere Good Reduction ... 128
  5.4.1 Cremona–Lingham’s Method: An Overview ....... 128
  5.4.2 Examples I: Real Quadratic Fields .............. 132
  5.4.3 Examples II: Imaginary Quadratic Fields ...... 149

A MAGMA Source Code ................................. 159
  A.1 Period Lattices and Complex Elliptic Logarithms ... 159
  A.2 Integral Points on Elliptic Curves .................. 166
  A.3 Height Bound I: Main Functions .................... 178
  A.4 Height Bound II: Computing $\alpha_v$ ............... 185
  A.5 Height Bound III: Intersection of Intervals ...... 192
  A.6 Height Bound IV: Intersection of Regions ......... 196
    A.6.1 intersect_complex.m .................. 196
    A.6.2 wp.m .................................. 202
    A.6.3 interval_wp.m ............................ 205
  A.7 Height Bound V: Interval Arithmetic ............... 212

Bibliography ........................................ 216
List of Figures

1.1 Elliptic curves over \( \mathbb{R} \) ................................................................. 2
1.2 Addition on elliptic curves ................................................................. 3

3.1 The boundary on \( \mathcal{H}_\tau \) associated to different \( U_\xi \) .................. 31
3.2 Loops on the torus \( \mathbb{C}/\Lambda_\tau \) when the boundary varies ...................... 32
3.3 Four quarters of \( C_j \) ................................................................. 39
3.4 An illustration of how to obtain \( S^{(v)} \) ................................................. 41
3.5 Division on \( F_\tau \) by 3 ................................................................. 42

4.1 A chain of 2-isogenies linked with a chain of lattices .......................... 63
List of Tables

3.1 Maximum values for $\epsilon(k)$ ........................................ 34

5.1 Illustration of the algorithm for Example 5.1.1 ...................... 98
5.2 Illustration of the algorithm for Example 5.1.2 ...................... 99
5.3 Illustration of the algorithm for Example 5.1.3 ...................... 101
5.4 Illustration of the algorithm for Example 5.1.4 ...................... 103
5.5 Illustration of the algorithm for Example 5.1.6 ...................... 106
5.6 Sieving procedure for the elliptic curve $E_2$ ......................... 111
5.7 Sieving procedure for the elliptic curve $E_4$ ......................... 113
5.8 LLL reduction used in Example 5.3.5 ................................. 124
5.9 Integral points on $y^2 = x^3 - 16x + 16$ over $\mathbb{Q}(\sqrt{-2})$ . 125
5.10 LLL reduction used in Example 5.3.6 ............................... 127
5.11 Integral points on the elliptic curve $E_6$ ............................ 127
5.12 Elliptic curves over $\mathbb{Q}(\sqrt{38})$ with everywhere good reduction . 135
5.13 Elliptic curves over $\mathbb{Q}(\sqrt{41})$ with everywhere good reduction . 136
5.14 Elliptic curves over $\mathbb{Q}(\sqrt{43})$ with everywhere good reduction . 137
5.15 Elliptic curves over $\mathbb{Q}(\sqrt{46})$ with everywhere good reduction . 138
5.16 Elliptic curves over $\mathbb{Q}(\sqrt{51})$ with everywhere good reduction . 139
5.17 Elliptic curves over $\mathbb{Q}(\sqrt{55})$ with everywhere good reduction . 140
5.18 Elliptic curves over $\mathbb{Q}(\sqrt{59})$ with everywhere good reduction . 141
5.19 Elliptic curves over $\mathbb{Q}(\sqrt{62})$ with everywhere good reduction . 142
5.20 Elliptic curves over $\mathbb{Q}(\sqrt{65})$ with everywhere good reduction . 143
5.21 Elliptic curves over $\mathbb{Q}(\sqrt{67})$ with everywhere good reduction . . . . 145
5.22 Elliptic curves over $\mathbb{Q}(\sqrt{78})$ with everywhere good reduction . . . . 146
5.23 Elliptic curves over $\mathbb{Q}(\sqrt{87})$ with everywhere good reduction . . . . 147
5.24 Elliptic curves over $\mathbb{Q}(\sqrt{95})$ with everywhere good reduction . . . . 148
5.25 Elliptic curves over $\mathbb{Q}(\sqrt{-26})$ with everywhere good reduction . . . . 150
5.26 Elliptic curves over $\mathbb{Q}(\sqrt{-29})$ with everywhere good reduction . . . . 151
5.27 Elliptic curves over $\mathbb{Q}(\sqrt{-31})$ with everywhere good reduction . . . . 152
5.28 Elliptic curves over $\mathbb{Q}(\sqrt{-38})$ with everywhere good reduction . . . . 153
5.29 Elliptic curves over $\mathbb{Q}(\sqrt{-53})$ with everywhere good reduction . . . . 154
5.30 Elliptic curves over $\mathbb{Q}(\sqrt{-59})$ with everywhere good reduction . . . . 155
5.31 Elliptic curves over $\mathbb{Q}(\sqrt{-61})$ with everywhere good reduction . . . . 156
5.32 Elliptic curves over $\mathbb{Q}(\sqrt{-83})$ with everywhere good reduction . . . . 157
5.33 Elliptic curves over $\mathbb{Q}(\sqrt{-87})$ with everywhere good reduction . . . . 158
Acknowledgements

First of all, I am most indebted to my supervisors Professor John E. Cremona and Professor Samir Siksek for all their valuable guidance and support I have constantly received during my time at the University of Warwick. Without their time and devotion, this thesis certainly could not come this far.

Secondly, I wish to thank my family for their moral support during my hard time overseas. Although they may not have much idea of my research, I believe that they will be proud of what I have achieved.

In addition, thanks also go to my fellow postgraduate students at the Mathematics Institute, particularly all members of Number Theory Group, for making my time at Warwick enjoyable and unforgettable.

Last but not least, I am grateful to the Development and Promotion of Science and Technology Talents Project (DPST), Thailand, for a scholarship to my postgraduate study.

*Coventry, England*              Thotsaphon Thongjunthug
*February 2011*
Declaration

I hereby declare that this thesis is my own work and to the best of my knowledge it contains no materials previously published or written by another person, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at the University of Warwick or elsewhere, is explicitly acknowledged in the thesis.

In this thesis, an alternative version of Chapter 2 and 3 has been published in [Tho08] and [Tho10] respectively. Chapter 4 is based on my joint work with Professor John E. Cremona at the University of Warwick; another version of this chapter has been submitted for publication as a joint paper [CT].

Each algorithm based on this thesis has been implemented in MAGMA, a computer package produced and distributed by the School of Mathematics and Statistics of the University of Sydney, Australia.

Finally, I declare that this thesis has not been submitted for a degree at any other institution.

Thotsaphon Thongjunthug
Summary

This thesis presents some major improvements in the following computations: a lower bound for the canonical height, period lattices, and elliptic logarithms.

On computing a lower bound for the canonical height, we have successfully generalised the existing algorithm of Cremona and Siksek [CS06] to elliptic curves over totally real number fields, and then to elliptic curves over number fields in general. Both results, which are also published in [Tho08] and [Tho10] respectively, will be fully explained in Chapter 2 and 3.

In Chapter 4, we give a complete method on computing period lattices of elliptic curves over $\mathbb{C}$, whereas this was only possible for elliptic curves over $\mathbb{R}$ in the past. Our method is based on the concept of arithmetic-geometric mean (AGM). In addition, we extend our method further to find elliptic logarithms of complex points. This work is done in collaboration with Professor John E. Cremona; another version of this chapter has been submitted for publication [CT].

In Chapter 5, we finally illustrate the applications of our main results towards certain computations which did not work well in the past due to lack of some information on elliptic curves. This includes determining a Mordell–Weil basis, finding integral points on elliptic curves over number fields [SS97], and finding elliptic curves with everywhere good reduction [CL07].

A number of computer programs have been implemented for the purpose of illustration and verification. Their source code (written in MAGMA) can be found in Appendix A.
Dedicated to my beloved aunt Pissawong Thongjunthug

“You always live on in my heart”
Chapter 1

Introduction

We will first introduce all underlying concepts which are necessary for later chapters, together with an overview of this thesis. In this chapter, we will start by a brief definition of elliptic curves, before move on to describe more specific concepts related to elliptic curves over number fields, and finally, elliptic curves over $\mathbb{C}$. A synopsis of each chapter will be also mentioned where appropriate.

1.1 An Overview of Elliptic Curves

In this section, we will briefly describe the definition an elliptic curve over a general field, and how to construct an operation defining the group law on it.

**Definition.** Let $K$ be a field. An elliptic curve $E$ defined over $K$ (denoted by $E/K$) is a non-singular projective plane curve of degree 3 over $K$, with a specified point of inflection $O$ which is also defined over $K$.

We can assume (see [Mil06, Proposition 1.2]) that $E$ is given by a homogeneous Weierstrass equation

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

where all $a_j \in K$ are constants. If $Z \neq 0$, then we can divide every term above by
Z^3 to obtain an affine Weierstrass equation

\[ E : \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{1.1} \]

via \( x = X/Z, \ y = Y/Z \). The line \( Z = 0 \) intersects \( E \) at \( (0 : 1 : 0) \) with multiplicity 3, so we may take \( O = (0 : 1 : 0) \); this is called the point at infinity of \( E \). It is also easy to prove (see [Was03, p. 20]) that every vertical line intersects \( E \) at \( O \).

From now on, we shall always assume that an elliptic curve \( E \) is given by an affine Weierstrass equation (1.1), unless otherwise stated. As in [Sil86, p. 46], we define the following quantities associated to a Weierstrass equation:

\[
\begin{align*}
  b_2 &= a_1^2 + 4a_2, \quad b_4 = 2a_4 + a_1a_3, \quad b_6 = a_3^2 + 4a_6, \\
  b_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\
  \Delta &= -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6.
\end{align*}
\]

The quantity \( \Delta \) is known as the discriminant of \( E \). An example of elliptic curves defined over \( \mathbb{K} = \mathbb{R} \) is illustrated in Figure 1.1.

\[ \text{(a) } \Delta < 0 \quad \text{(b) } \Delta > 0 \]

Figure 1.1: Elliptic curves over \( \mathbb{R} \)

In this thesis, our field \( K \) will be either \( \mathbb{R}, \mathbb{C} \), or a number field, hence \( \text{char}(K) = 0 \). For now we note that, since \( \text{char}(K) \neq 2 \), we can rewrite the Weierstrass equation of \( E \) as

\[ E : \quad (2y + a_1x + a_3)^2 = f(x) = 4x^3 + b_2x^2 + 2b_4x + b_6. \]
1.1. An Overview of Elliptic Curves

It can be shown (see [Sil86, Proposition III.1.4]) that $\Delta \neq 0$ if and only if $f(x)$ has three distinct roots, which is equivalent to the non-singularity of $E$.

**Definition.** Let $E$ be an elliptic curve defined over a field $K$, and let $L \supseteq K$ be a field. The set of all $L$-points of $E$, denoted by $E(L)$, is given by

$$E(L) = \{(x, y) \in L^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}.$$ 

For any two points $P_1, P_2 \in E(L)$, we can construct an operation so-called *addition* (denoted by $+$) geometrically as follows. First, let $L_1$ be the straight line through $P_1, P_2$ (or if $P_1 = P_2$, take $L_1$ to be the tangent line to $E$ at that point). Then $L_1$ will intersect $E$ at another point, say, $P'_3$. Let $L_2$ be the vertical line through $P'_3$. Then $L_2$ will intersect $E$ at another point, say $P_3$. Finally, we define $P_1 + P_2 = P_3$. An example of this process for $L = K = \mathbb{R}$ is shown in Figure 1.2.

![Figure 1.2: Addition on elliptic curves](image)

It is readily shown (see, e.g., [Was03, Section 2.2]) that $E(L)$ becomes an abelian group with $O$ as the identity once being equipped with this addition. We say that a point $P \in E(L)$ is a *torsion point* if $P$ has finite order in $E(L)$; otherwise, $P$ is said to be *non-torsion*. 
1.2 Elliptic Curves over Number Fields

In this section, we shall first explain the definition of heights on elliptic curves over number fields, and then briefly describe the importance of a lower bound for the canonical height towards computing a Mordell–Weil basis. Throughout this section, our elliptic curve $E$ will be defined over a number field $K$.

1.2.1 Heights

Roughly speaking, the height function is a way to measure how “complicated” the $x$-coordinate of a point $P \in E(K)$ is. In this thesis, we will be using the canonical height, which can be expressed as a sum of all contributions from local heights. It should be noted that normalisation of heights varies in literature. In our case, the local and canonical heights are defined with respect to the divisor $2(O)$. This leads to the same normalisation as the one used in the computer package MAGMA, and gives double the values compared with Silverman’s paper [Sil88] where heights are defined with respect to $(O)$.

The Canonical Height

Denote the sets of real and complex archimedean places of $K$ by $M^r_K$ and $M^c_K$ respectively, and let $M_K$ be the set of all places of $K$. For $v \in M_K$, let $n_v = [K_v : \mathbb{Q}_v]$, and let $\sigma_v$ be the associated embedding of $K$ into the completion $K_v$.

**Definition.** For $x \in K$, the absolute value of $x$ at a place $v \in M_K$ is given by

$$|x|_v = \begin{cases} 
|\sigma_v(x)| & \text{if } v \in M^r_K \cup M^c_K, \\
N(p)^{-\text{ord}_p(x)/np} & \text{if } v = p,
\end{cases}$$

(1.2)

where $p$ is the prime ideal associated to a non-archimedean place $v$, and $N$ denotes the norm of an integral ideal of $K$. 

It is a standard fact (see, e.g., [Coh07, Proposition 4.1.14]) that this definition satisfies all axioms of valuation theory and the product formula $\prod_{v \in M_K} |x|_v^{n_v} = 1.$

**Definition.** For $P \in E(K)$, the *naive height* of $P$ (relative to $K$) is defined by

$$H_K(P) = \begin{cases} 
1 & \text{if } P = O, \\
\prod_{v \in M_K} \max\{1, |x(P)|_v\}^{n_v} & \text{if } P \neq O.
\end{cases}$$

(1.3)

**Definition.** For $P \in E(K)$, the *logarithmic height* of $P$ is defined by

$$h(P) = \frac{1}{[K: \mathbb{Q}]} \log H_K(P).$$

(1.4)

From this, the *canonical height* of $P$ is given by

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

(1.5)

Observe that $h(P) \geq 0$ for all $P \in E(K)$, thus we also have $\hat{h}(P) \geq 0$ for all $P \in E(K)$. Moreover, we have $\hat{h}(mP) = m^2 \hat{h}(P)$ for all $P \in E(K)$ and $m \in \mathbb{Z}$ (see [Sil86, p. 230] for the proof). In particular, if $P$ is a torsion point of order $m$, then we have

$$0 = \hat{h}(O) = \hat{h}(mP) = m^2 \hat{h}(P)$$

(the fact that $\hat{h}(O) = 0$ follows easily from (1.6)), i.e., $\hat{h}(P) = 0$. In fact, the canonical height $\hat{h} : E(K) \to [0, \infty)$ is a positive definite quadratic form on $E(K)/E_{\text{tors}}(K)$, which gives it the structure of a lattice. Hence there exists a positive lower bound for $\hat{h}(P)$ among all non-torsion $P \in E(K)$.

Computing such a lower bound has a number of applications in the arithmetic of elliptic curves. In particular, it is a crucial step in determining a Mordell–Weil basis for $E(K)$; see Section 1.2.2 for more details. In the past, a number of explicit lower bounds for the canonical height on $E(K)$ have been proposed. Some of
them, including [HS88, Theorem 0.3], aim to prove Lang’s conjecture (see [Sil86, Conjecture VIII.9.9]), which states that there exists a constant $c_K$, depending only on $K$, such that

$$\hat{h}(P) \geq c_K \log \mathcal{N}(D_{E/K})$$

for all non-torsion $P \in E(K)$, where $D_{E/K}$ is the minimal discriminant of $E/K$. As we will see later on, however, the lower bound obtained by that result is too small for practical use.

In this thesis, we will develop an alternative method for determining a larger positive lower bound for the canonical height on elliptic curves over number fields. The underlying methodology is mainly inspired by the algorithm of Cremona and Siksek [CS06], which allows one to compute such a lower bound for elliptic curves defined over $\mathbb{Q}$ only. Our work on this is divided into two parts, namely, determining certain contributions from all real embeddings, and then from all complex embeddings. Both parts will be described in Chapter 2 and 3 respectively.

**Local Height Functions**

Finally, we give the definition of local heights. Suppose $P \in E(K)$ with $2P \neq O$. Then one can observe that $x(2P) = g(P)/f(P)$, where

$$f(P) = 4x(P)^3 + b_2 x(P)^2 + 2b_4 x(P) + b_6, \quad g(P) = x(P)^4 - b_4 x(P)^2 - 2b_6 x(P) - b_8.$$  

Hence by (1.3), we have

$$H_K(2P) = \prod_{v \in M_K} \max \left\{ 1, \left| x(2P) \right|_v \right\}^{n_v}$$

$$= \prod_{v \in M_K} \max \left\{ 1, \left| g(P) \right|_v / \left| f(P) \right|_v \right\}^{n_v}$$

$$= \prod_{v \in M_K} \left| f(P) \right|_v^{n_v} \cdot \prod_{v \in M_K} \max \left\{ 1, \left| g(P) \right|_v / \left| f(P) \right|_v \right\}^{n_v}$$

$$= \prod_{v \in M_K} \max \left\{ \left| f(P) \right|_v, \left| g(P) \right|_v \right\}^{n_v}$$
(note that $\prod_{v \in M_K} |f(P)|_v^{n_v} = 1$ by the product formula), and so

$$h(2P) = \frac{1}{[K : \mathbb{Q}]} \sum_{v \in M_K} n_v \log \max\{|f(P)|_v, |g(P)|_v\}$$

by (1.4). Together with (1.4) again, this easily yields

$$h(2P) - 4h(P) = \frac{1}{[K : \mathbb{Q}]} \sum_{v \in M_K} n_v \log \Phi_v(P), \quad (1.6)$$

where

$$\Phi_v(P) = \begin{cases} 1 & \text{if } P = O, \\ \max\{|f(P)|_v, |g(P)|_v\} & \text{if } P \neq O. \end{cases} \quad (1.7)$$

**Definition.** For $v \in M_K$, let $K_v$ be the completion of $K$ at $v$. The function $\lambda_v : E(K_v) \to \mathbb{R}$ defined by

$$\lambda_v(P) = \log \max\{1, |x(P)|_v\} + \sum_{j=0}^{\infty} \frac{\log \Phi_v(2^j P)}{4^{j+1}}. \quad (1.8)$$

is called the **local height function** at $v$.

To see the relationship between the canonical height and local heights, we use (1.5) and the telescoping sum to obtain

$$\hat{h}(P) = h(P) + \left[ \frac{h(2P)}{4} - h(P) \right] + \left[ \frac{h(2^2P)}{4^2} - \frac{h(2P)}{4} \right] + \cdots$$

$$= \frac{1}{[K : \mathbb{Q}]} \sum_{v \in M_K} n_v \left( \log \max\{1, |x(P)|_v\} + \frac{\log \Phi_v(P)}{4} \right) + \frac{\log \Phi_v(2P)}{4^2} + \cdots$$

$$= \frac{1}{[K : \mathbb{Q}]} \sum_{v \in M_K} n_v \lambda_v(P). \quad (1.9)$$

(the second equality follows directly from (1.4) and (1.8)). This therefore allows us to obtain $\hat{h}(P)$ by combining the contribution of $\lambda_v$ on each local model $E(K_v)$, noting that $\lambda_v(P) = 0$ for almost all $v$. 
1.2.2 Mordell–Weil Bases

Recall that $E(K)$ is an abelian group under addition. By the Mordell–Weil theorem (see, e.g., [Sil86, Chapter VIII] for more details), it is also well known that $E(K)$ is finitely generated. It then follows that

$$E(K) \cong E_{\text{tors}}(K) \times \mathbb{Z}^r,$$

where $E_{\text{tors}}(K)$ is the torsion subgroup of $E(K)$ (i.e., the set of all torsion points in $E(K)$), and the rank $r \geq 0$ of $E(K)$ is the cardinality of a Mordell–Weil basis for $E(K)$ (i.e., the set of all non-torsion points in $E(K)$ whose images in $E(K)/E_{\text{tors}}(K)$ form a $\mathbb{Z}$-basis for it).

In general, it turns out that the torsion subgroup of $E(K)$ can be determined more easily than a Mordell–Weil basis for $E(K)$. According to [Sik95], the task of explicit computation of such a basis consists of the following steps:

1. Determine $P_1, \ldots, P_r$ whose images in $E(K)/E_{\text{tors}}(K)$ generate a subgroup of finite index of $E(K)/E_{\text{tors}}(K)$. Usually, these are obtained by performing an $m$-descent for some $m \geq 2$.

2. A lower bound $\lambda > 0$ for the canonical height $\hat{h}(P)$ is determined, which in turn yields an upper bound on the index $n = [E(K)/E_{\text{tors}}(K) : \langle P_1, \ldots, P_r \rangle]$.

3. A sieving procedure [Sik95, Section 4] is then used to deduce a Mordell–Weil basis for $E(K)$.

In step (2), we certainly wish to have an upper bound for $n$ as small as possible. In particular, $P_1, \ldots, P_r$ will certainly be a Mordell–Weil basis of $E(K)$ if $n = 1$. It follows from the following theorem that, in order to have a smaller upper bound for $n$, one must obtain a larger lower bound for the canonical height.
1.3 Elliptic Curves over \( \mathbb{C} \)

**Theorem 1.2.1** (The Geometry of Numbers). If \( E(K) \) contains no points \( P \) of infinite order with \( \hat{h}(P) \leq \lambda \) for some \( \lambda > 0 \), then the index \( n \) satisfies

\[
n \leq R(P_1, \ldots, P_r)^{1/2} (\gamma_r/\lambda)^{r/2},
\]

where \( R(P_1, \ldots, P_r) = \det(\langle P_i, P_j \rangle)_{1 \leq i,j \leq r} \) and

\[
\langle P_i, P_j \rangle = \frac{1}{2} \left( \hat{h}(P_i + P_j) - \hat{h}(P_i) - \hat{h}(P_j) \right).
\]

Moreover, the values \( \gamma_r \) may be taken to be

\[
\begin{align*}
\gamma_1 &= 1, & \gamma_2 &= 4/3, & \gamma_3 &= 2, & \gamma_4 &= 4, \\
\gamma_5 &= 8, & \gamma_6 &= 64/3, & \gamma_7 &= 64, & \gamma_8 &= 2^8,
\end{align*}
\]

and \( \gamma_r = (4/\pi)\Gamma(r/2 + 1)^{2/r} \) for \( r \geq 9 \).

**Proof.** See [Sik95, Theorem 3.1]. \( \square \)

As mentioned earlier, we will fully explain a new method for computing \( \lambda \) in Chapter 2 and 3. Some examples on how to determine a Mordell–Weil basis using \( \lambda \) and the process above will be also shown in Chapter 5.

1.3 Elliptic Curves over \( \mathbb{C} \)

We now move on to elliptic curves defined over \( \mathbb{C} \), where we will give a brief introduction on period lattices of elliptic curves and elliptic logarithms of points, which will be the subject of Chapter 4.

**Definition.** A lattice \( \Lambda \) is a free \( \mathbb{Z} \)-module of rank 2 embedded as a discrete subgroup of \( \mathbb{C} \), i.e.,

\[
\Lambda = \{ n_1 w_1 + n_2 w_2 : n_1, n_2 \in \mathbb{Z} \}
\]
for some $w_1, w_2 \in \mathbb{C}$ with $w_1/w_2 \notin \mathbb{R}$.

For a lattice $\Lambda$, we can also identify $\mathbb{C}/\Lambda$ with the set

$$F_{w_1,w_2} = \{\lambda_1 w_1 + \lambda_2 w_2 : 0 \leq \lambda_1, \lambda_2 < 1\}$$

called the (open) fundamental parallelogram for $\Lambda$ (or if we allow both $\lambda_j = 1$, we say that it is closed). In the topological point of view, this is a torus. Clearly, choosing a different $\mathbb{Z}$-basis for $\Lambda$ yields a different fundamental parallelogram.

Let $E$ be an elliptic curve defined over $\mathbb{C}$. With some change of variables, we can assume that the Weierstrass equation of $E$ is of the form

$$E : \quad Y^2 = 4(X-e_1)(X-e_2)(X-e_3),$$

where all $e_j$ are distinct and $\sum_{j=1}^{3} e_j = 0$. It is well known (see, e.g., [Was03, Chapter 9]) that $E(\mathbb{C}) \cong \mathbb{C}/\Lambda$ for some lattice $\Lambda$ via the map

$$P = (\wp_\Lambda(z), \wp_\Lambda'(z)) \longmapsto z \pmod{\Lambda},$$

$$O \longmapsto 0 \pmod{\Lambda}.$$  

We say that $\Lambda$ is the period lattice of $E$, and $z$ is an elliptic logarithm of $P$. The values of $\wp_\Lambda(z)$ and $\wp_\Lambda'(z)$ can be computed using the power series expansion as shown in the following proposition.

**Proposition 1.3.1** ([Coh93, Proposition 7.4.4]). Let $\{w_1, w_2\}$ be a $\mathbb{Z}$-basis for $\Lambda$ chosen so that $\Im(w_2/w_1) > 0$. Set

$$\tau = w_2/w_1, \quad q = \exp(2i\pi \tau), \quad u = \exp(2i\pi z/w_1)$$
1.3. Elliptic Curves over \( C \) (here, \( i = \sqrt{-1} \)). Then

\[
\wp_{\Lambda}(z) = \left( \frac{2i\pi}{w_1} \right)^2 \left( \frac{1}{12} + \frac{u}{(1-u)^2} + \sum_{j=1}^{\infty} \left[ \frac{q^j u}{(1-q^j u)^2} + \frac{q^j u^{-1}}{(1-q^j u^{-1})^2} - \frac{2q^j}{(1-q^j)^2} \right] \right)
\]

and

\[
\wp'_{\Lambda}(z) = \left( \frac{2i\pi}{w_1} \right)^3 u \left( \frac{1+u}{(1-u)^3} + \sum_{j=1}^{\infty} q^j \left[ \frac{1+q^j u}{(1-q^j u)^3} + \frac{q^j + u}{(q^j - u)^3} \right] \right).
\]

To be precise, a \( Z \)-basis for the period lattice of \( E \) is given by any two of the generators \( w_1, w_2, w_3 \), where \( \wp_{\Lambda}(w_j/2) = e_j \) and \( \wp'_{\Lambda}(w_j/2) = 0 \) for all \( j \). Suppose \( \ell_j \) is the straight line on the complex plane starting from 0 to \( w_j/2 \). Then we have

\[
\frac{w_j}{2} = \int_{\ell_j} dz = \int_{\ell_j} \frac{d\wp_{\Lambda}(z)}{\wp'_{\Lambda}(z)} = \int_{C_j} \frac{dX}{Y},
\]

where \( C_j \) is the image of \( \ell_j \) on \( E \) under \((\wp_{\Lambda}, \wp'_{\Lambda})\), i.e.,

\[
C_j = \{ (\wp_{\Lambda}(tw_j/2), \wp'_{\Lambda}(tw_j/2)) : 0 \leq t \leq 1 \}.
\]

More generally, if \( z_P \) is an elliptic logarithm of \( P \in E(\mathbb{C}) \), then

\[
z_P = \int_{C_P} \frac{dX}{Y} \pmod{\Lambda},
\]

where \( C_P = \{ (\wp_{\Lambda}(tz_P), \wp'_{\Lambda}(tz_P)) : 0 \leq t \leq 1 \} \).

If \( E \) is defined over \( \mathbb{R} \), then we obtain one of two special cases for the lattice \( \Lambda \) of \( E \). It can be shown (see, e.g., [Was03, pp. 274–275]) that if \( E \) has positive discriminant (see Figure 1.1b), then \( \Lambda \) is rectangular, i.e., there exists a \( Z \)-basis \( \{w_1, w_2\} \) for \( \Lambda \) where \( w_1 \in \mathbb{R} \) and \( w_2 \in i\mathbb{R} \). In this case, the connected component of the identity (i.e., the one containing \( O \)) is parameterised by the line \( \{tw_1 : 0 \leq t < 1 \} \), while the “loop” component is parameterised by the line \( \{tw_1 + w_2/2 : 0 \leq t \leq 1 \} \).
For $E/\mathbb{R}$ with negative discriminant (see Figure 1.1a), we obtain a \textit{skewed} lattice, i.e., there exists a $\mathbb{Z}$-basis for $\Lambda$ with $w_1 \in \mathbb{R}$ and $\Re(w_2/w_1) = 1/2$. In this case, the whole $E(\mathbb{R})$ is connected and parameterised by the line $\{tw_1 : 0 \leq t < 1\}$.

Finding period lattices and elliptic logarithms is an important computation in its own right, and also has a number of applications towards certain algorithms, including one for determining a lower bound for the canonical height on elliptic curves over number fields, which will be fully explained in Chapter 2 and 3. Although there are some algorithms including \cite[Algorithm 7.4.7 and 7.4.8]{Coh93} readily available for computing both period lattices and elliptic logarithms, those algorithms only work for elliptic curves over $\mathbb{R}$. In Chapter 4, we will show how to develop a complete method for computing period lattices and elliptic logarithms for elliptic curves over $\mathbb{C}$ in general, based on the method of \textit{arithmetic-geometric mean} (AGM). As we will see later on, our algorithm will allow one to compute both values with high degree of precision very rapidly.

In conclusion, we have introduced all necessary concepts to be used later on in this thesis, including an overview of each chapter. The next two chapters will focus on development of our first main result, namely, an algorithm for computing a lower bound for the canonical height on elliptic curves over number fields.
Chapter 2

Height Bound I

We will now focus on our first main result, where we develop an algorithm for computing a lower bound for the canonical height on elliptic curves over number fields. Our algorithm, which is inspired by the one of Cremona and Siksek [CS06], involves estimating local heights and solving a system of certain inequalities on both real and complex embeddings.

In this chapter, we will first show how to derive an estimate for local heights, and then show how to solve the system of inequalities mentioned above on real embeddings. This in turn will suffice for computing a lower bound for the canonical height on elliptic curves over number fields with at least one real embedding. A more sophisticated method for solving such inequalities on complex embeddings will be explained in Chapter 3.

Another version of this chapter, which is more specific to elliptic curves over totally real number fields, has been published in [Tho08].

2.1 Points of Good Reduction

Let $E$ be an elliptic curve defined over a number field $K$, given by a Weierstrass equation

$$E : \quad y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$

13
with all \( a_j \in \mathcal{O}_K \), where \( \mathcal{O}_K \) is the ring of integers of \( K \). Let \( \Delta \) be the discriminant of \( E \). As in Chapter 1, we denote the sets of real and complex archimedean places by \( M'_K \) and \( M'_K \) respectively, and let \( M_K \) be the set of all places of \( K \).

For all non-archimedean places \( v \), let \( E^{(v)} \) be a local minimal model for \( E \) over the completion \( K_v \), while we simply set \( E^{(v)} = E \) for all archimedean places \( v \). We define the map
\[
\phi : E(K) \to \prod_{v \in S} E^{(v)}(K_v),
\]
where \( S = M'_K \cup M'_K \cup \{ p : p \mid \Delta \} \), in such a way that \( P \) is mapped into its corresponding point on:

- \( E^{(v)}(\mathbb{R}) \), for each \( v \in M'_K \), and
- \( E^{(v)}(\mathbb{C}) \), for each \( v \in M'_K \), and
- \( E^{(v)}(K_v) \), for each non-archimedean place \( v \mid \Delta \).

Note that if \( K \) has class number greater than 1, then \( E^{(v)} \) may differ for different non-archimedean places \( v \), i.e., \( E \) may not have a globally minimal model. As we will see, our formulae can be simplified if \( E \) is given by a globally minimal model.

We wish to compute a positive lower bound \( \lambda \) for the canonical height \( \hat{h} \) on \( E(K) \). Instead of working on \( E(K) \) directly, we determine a positive lower bound \( \mu \) for the canonical height on the subgroup
\[
E_{gr}(K) = \phi^{-1} \left( \prod_{v \in S} E_0^{(v)}(K_v) \right),
\]
where \( E_0^{(v)}(K_v) \) is the connected component of the identity (for non-archimedean \( v \), this is the set of points of good reduction). The next lemma shows that we can obtain \( \lambda \) very easily once \( \mu \) is known.

**Lemma 2.1.1.** Let \( \mu \) be a positive lower bound for the canonical height on \( E_{gr}(K) \). Set
\[
\lambda = \mu / c^2,
\]
where \( c \) is the least common multiple of the Tamagawa indices

\[
c_v = [E^{(v)}(K_v) : E_0^{(v)}(K_v)]
\]

for all \( v \in M_K \). Then \( \lambda \) is a positive lower bound for the canonical height on \( E(K) \).

**Proof.** Note that \( c \) is well-defined since \( c_v = 1 \) for all \( v \notin S \). For all non-torsion point \( P \in E(K) \), it is clear that \( cP \in E_{gr}(K) \). Then by quadraticity of \( \hat{h} \), we have

\[
\mu \leq \hat{h}(cP) = c^2 \hat{h}(P),
\]

and so \( \hat{h}(P) \geq \mu/c^2 \). Hence we can take \( \lambda = \mu/c^2 \).

In this chapter, we will first derive an explicit formula for computing \( \mu \). The value of \( \mu \) obtained by this formula, in practice, will not be as good as the one obtained by the algorithm to be derived later on in Chapter 3. Using a number of criteria, our algorithm will check whether a given \( \mu > 0 \) is a lower bound on \( E_{gr}(K) \). The value of \( \mu \) then can be refined further by repeating the algorithm.

## 2.2 Estimation of Local Heights

Recall the definition of local and canonical heights in Section 1.2.1. From (1.9), we have seen that the canonical height can be written as a sum of local heights given by (1.8). This therefore allows us to estimate \( \hat{h}(P) \) by approximating each local height \( \lambda_v \) for \( v \in M_K \).

### 2.2.1 Non-Archimedean Cases

For \( P \in E(K) \), let \( P^{(p)} \) be its corresponding point of \( P \) (via the map \( \phi \)) on the minimal model \( E^{(p)} \). Let \( \lambda_p \) and \( \lambda_p^{(p)} \) be the local heights associated to \( E \) and \( E^{(p)} \) respectively. Assume that \( E \) is integral and \( E^{(p)} \) has all coefficients in \( \mathcal{O}_p = \)
\(\{x \in K : \text{ord}_p(x) \geq 0\}\), we denote \(\Delta\) and \(\Delta^{(p)}\) the discriminants of \(E\) and \(E^{(p)}\) respectively. These values are related by \(\Delta = (u^{(p)})^{12}\Delta^{(p)}\), for some \(u^{(p)} \in \mathcal{O}_p\). If \(E\) is given by a globally minimal model, then we may take \(E^{(p)} = E\) for all \(p\).

The following lemma illustrates the relation between \(\lambda_p\) and \(\lambda_p^{(p)}\).

**Lemma 2.2.1.**

\[
\lambda_p(P) = \lambda_p^{(p)}(P^{(p)}) + \frac{1}{6} \log |\Delta/\Delta^{(p)}|_p.
\]

**Proof.** This follows from the use of two different normalisations of local heights which differ by \(\log |\cdot|_p/6\), and the fact that one of them is independent of the choice of Weierstrass model. For full details, see [CPS06, Section 4]. \(\square\)

Now for \(P \in E_{gr}(K)\), it follows that \(P^{(p)} \in E^{(p)}_0(K_p)\) at every prime ideal \(p\). In this case, we can easily compute \(\lambda_p^{(p)}(P^{(p)})\) with the following lemma.

**Lemma 2.2.2.** Let \(p\) be a prime ideal and \(P^{(p)} \in E^{(p)}_0(K_p) \setminus \{O\}\) (i.e., \(P\) is a point of good reduction). Then

\[
\lambda_p^{(p)}(P^{(p)}) = \log \max\{1, |x(P^{(p)})|_p\}.
\]

**Proof.** This is a standard result; see, e.g., [Sil88, Section 5]. Note that the definition that we use of local height of a point with good reduction does not include a multiple of \(-\log |\Delta^{(p)}|_p\) (cf. [Sil88, p. 351]). \(\square\)

**Definition.** Let \(x \in K\). The denominator ideal of \(x\), denoted by \(\text{denom}(x)\), is the integral ideal \(B\) such that \(\langle x \rangle = AB^{-1}\), where \(A, B\) are coprime integral ideals.

The next lemma yields a simplified formula for computing the sum of all non-archimedean local heights on \(E_{gr}(K)\).

**Lemma 2.2.3.** Suppose \(P \in E_{gr}(K) \setminus \{O\}\). Then

\[
\sum_p n_p \lambda_p(P) = L(P) - \frac{1}{6} \log \mathcal{N}(M_E),
\]
2.2. Estimation of Local Heights

where

\[ L(P) = \log \mathcal{N} \left( \prod_{p \mid \text{denom}(x(P^{(p)})))} p^{-\text{ord}_p(x(P^{(p)})))} \right), \quad M_E = \prod_{p} p^{\text{ord}_p(\Delta/\Delta^{(p)})}. \]

Note that \( \mathcal{N}(M_E) = 1 \) if \( E \) is given by a globally minimal model.

**Proof.** Since \( P \in E_{gr}(K) \) by assumption, we have \( P^{(p)} \in E_0^{(p)}(K_p) \) for all \( p \). It then follows from Lemma 2.2.1 and Lemma 2.2.2 that

\[
\sum_{p} n_p \lambda_p(P) = \sum_{p} n_p \lambda_p^{(p)}(P^{(p)}) + \frac{1}{6} \sum_{p} n_p \log |\Delta/\Delta^{(p)}|_p
\]

\[
= \sum_{p} n_p \log \max\{1, |x(P^{(p)}|_p\} + \frac{1}{6} \sum_{p} n_p \log |\Delta/\Delta^{(p)}|_p. \quad (2.1)
\]

Clearly, the term \( \log \max\{1, |x(P^{(p)}|_p\} \) will vanish if \( |x(P^{(p)}|_p \leq 1 \). Hence the first sum in (2.1) is obtained by all those \( p \) satisfying \( |x(P^{(p)}|_p > 1 \). Recall from (1.2) that

\[ |x(P^{(p)}|_p = \mathcal{N}(p)^{-\text{ord}_p(x(P^{(p)}))/n_p}. \]

Observe that \( |x(P^{(p)}|_p > 1 \) if and only if \( p \mid \text{denom}(x(P^{(p)})). \) Therefore, the first sum in (2.1) becomes

\[
\sum_{p} n_p \log \max\{1, |x(P^{(p)}|_p\} = \log \mathcal{N} \left( \prod_{p \mid \text{denom}(x(P^{(p)})))} p^{-\text{ord}_p(x(P^{(p)})))} \right) = L(P).
\]

Secondly, it follows from (1.2) that the second sum in (2.1) is

\[
\frac{1}{6} \sum_{p} n_p \log |\Delta/\Delta^{(p)}|_p = -\frac{1}{6} \log \mathcal{N} \left( \prod_{p} p^{\text{ord}_p(\Delta/\Delta^{(p)})} \right) = -\frac{1}{6} \log \mathcal{N}(M_E).
\]

Finally, if \( E \) is given by a globally minimal model, then \( \Delta^{(p)} = \Delta \) for all \( p \), so \( \mathcal{N}(M_E) = 1 \). \( \square \)
2.2.2 Archimedean Cases

For \( v \in M_K \cup M_K^c \), we define \( \alpha_v \) by

\[
\alpha_v^{-3} = \inf_{P \in E_0^{(v)}(K_v)} \Phi_v(P)
\]

(see (1.7) for the definition of \( \Phi_v \)). The exponent \(-3\) is introduced in order to simplify expressions appearing later on. These \( \alpha_v \) can be computed very rapidly using the method in [CPS06, Section 7 and 9], according as \( v \in M_K \) and \( v \in M_K^c \).

The following lemma gives us an estimate for the archimedean local heights.

**Lemma 2.2.4.** If \( P \in E_0^{(v)}(K_v) \setminus \{0\} \), then

\[
\log \max\{1, |x(P)|_v\} - \lambda_v(P) \leq \log \alpha_v.
\]

**Proof.** Rearrange (1.8) and use the fact that

\[
\sum_{j=0}^{\infty} \frac{\log \Phi_v(2^j P)}{4^j+1} \geq \sum_{j=0}^{\infty} \frac{\log(\alpha_v^{-3})}{4^j+1} = -\log \alpha_v.
\]

\[\Box\]

2.3 Multiplication by \( n \)

In this section, we will derive a lower estimate for the contribution that multiplication by \( n \) makes towards \( L(nP) \), where \( L \) is defined as in Lemma 2.2.3.

Let \( k_p \) be the residue class field of \( p \), and let \( e_p \) be the exponent of the group \( E_{ns}^{(p)}(k_p) \cong E_0^{(p)}(K_p)/E_1^{(p)}(K_p) \). Define

\[
D_E(n) = \sum_{\substack{p \text{ prime} \in \N \nmid n}} 2(1 + \text{ord}_{c(p)}(n/e_p)) \log \mathcal{N}(p), \tag{2.2}
\]

where \( c(p) \) is the characteristic of \( k_p \). Note that \( k_p \) is a finite field, so \( c(p) \) is always
2.3. Multiplication by $n$

a prime number. In particular, $\mathcal{N}(p) = |k_p| \leq c(p)[K:Q]$.

**Proposition 2.3.1.** If $e_p \mid n$, then we have the following:

1. $\mathcal{N}(p) \leq (n + 1)^{\max\{2,[K:Q]\}}$.

2. $D_E(n)$ is finite. Moreover, if $P \in E_{\text{gr}}(K)$ is non-torsion and $n \geq 1$, then

$$L(nP) \geq D_E(n).$$

**Proof.** Suppose $e_p \mid n$. If $E^{(p)}$ has bad reduction at $p$, then $e_p$ is $c(p)$, $\mathcal{N}(p) + 1$, or $\mathcal{N}(p) - 1$ depending on whether $E^{(p)}$ has additive, non-split multiplicative, or split multiplicative reduction at $p$. In any case, this implies

$$n \geq e_p \geq \mathcal{N}(p)^{1/[K:Q]} - 1,$$

and thus $\mathcal{N}(p) \leq (n + 1)^{[K:Q]}$. Now for $p$ at which $E^{(p)}$ has good reduction, we have

$$E_{\text{ns}}^{(p)}(k_p) = E^{(p)}(k_p) \cong \mathbb{Z}/d_1 \mathbb{Z} \times \mathbb{Z}/d_2 \mathbb{Z},$$

where $d_1 \mid d_2$ and $d_2 = e_p$. It then follows from Hasse’s theorem (see, e.g., [Sil86, Theorem V.1.1]) that

$$(\sqrt{\mathcal{N}(p)} - 1)^2 \leq |E_{\text{ns}}^{(p)}(k_p)| = d_1 d_2 \leq e_p^2 \leq n^2.$$

Thus $\mathcal{N}(p) \leq (n + 1)^2$. Combining this with above result, this yields $\mathcal{N}(p) \leq (n + 1)^{\max\{2,[K:Q]\}}$, which proves (1). It is then immediate that $D_E(n)$ is finite.

To prove the rest of (2), first note that $P \in E_{\text{gr}}(K)$ implies $P^{(p)} \in E_0^{(p)}(K_p)$ for every $p$. Define

$$E_n^{(p)}(K_p) = \{P \in E_0^{(p)}(K_p) : \text{ord}_p(x(P)) \leq -2n\}.$$
Then it is known (see [Coh07, Lemma 7.3.28]) that for all \( n \geq 1, \)

\[
E_n^{(p)}(K_p)/E_{n+1}(K_p) \cong k_p^+ \cong (\mathbb{Z}/e(p)\mathbb{Z})^t,
\]

for some integer \( t > 0. \) Let \( e(p) = \text{ord}_c(n/e_p). \) Then \( nP^{(p)} \in E_{e(p)+1}(K_p), \) i.e.,

\[
\text{ord}_p(\text{denom}(x(nP^{(p)}))) \geq 2(e(p) + 1).
\]

This implies that \( e_p | n \) is equivalent to \( p | \text{denom}(x(nP^{(p)})). \) Hence

\[
\prod_{p | \text{denom}(x(nP^{(p)}))} \mathcal{N}(p)^{-\text{ord}_p(x(nP^{(p)}))} \geq \prod_{p \text{ prime}} \mathcal{N}(p)^{2(e(p)+1)}.
\]

The result then follows after taking logarithms on both sides. \( \square \)

## 2.4 A Bound for Multiples of Points of Good Reduction

In this section, we will first derive a bound for the \( x \)-coordinates of \( nP, \) where \( P \in E_{\text{gr}}(K) \) is non-torsion. This in turn yields an explicit lower bound for the canonical height on \( E_{\text{gr}}(K). \)

For \( \mu > 0 \) and \( n \in \mathbb{Z}_{>0}, \) define \( B_n(\mu) \) by

\[
\log B_n(\mu) = [K : \mathbb{Q}]n^2\mu - D_E(n) + \frac{1}{6} \log \mathcal{N}(M_E) + \sum_{v \in M_K^*} \log \alpha_v + 2 \sum_{v \in M_K^*} \log \alpha_v.
\]

**Proposition 2.4.1.** If \( B_n(\mu) \leq 1 \) then \( \hat{h}(P) \geq \mu \) for all non-torsion \( P \in E_{\text{gr}}(K). \)

If \( B_n(\mu) \geq 1, \) then for all non-torsion \( P \in E_{\text{gr}}(K) \) with \( \hat{h}(P) \leq \mu, \) we have

\[
|x(nP)|_v \leq \begin{cases} B_n(\mu) & \text{if } v \in M_K^*, \\ \sqrt{B_n(\mu)} & \text{if } v \in M_K^*.
\end{cases}
\]
2.4. A Bound for Multiples of Points of Good Reduction

Proof. Suppose \( P \in E_{gr}(K) \) is a non-torsion point with \( \hat{h}(P) \leq \mu \). By Lemma 2.2.4, we have

\[
\log \max\{1, |x(nP)|_v \} - \lambda_v(nP) \leq \log \alpha_v
\]

for all \( v \in M_K \). This implies that

\[
\sum_{v \in M_K^r} \log \max\{1, |x(nP)|_v \} + 2 \sum_{v \in M_K^c} \log \max\{1, |x(nP)|_v \} \\
\leq \sum_{v \in M_K^r} \lambda_v(nP) + 2 \sum_{v \in M_K^c} \lambda_v(nP) + \sum_{v \in M_K^r} \log \alpha_v + 2 \sum_{v \in M_K^c} \log \alpha_v. \tag{2.3}
\]

Note that \( n_v = 1 \) for all \( v \in M_K^r \) and \( n_v = 2 \) for all \( v \in M_K^c \). By writing \( \hat{h}(nP) \) as a sum of local heights (see (1.9)), we have

\[
\sum_{v \in M_K^r} \lambda_v(nP) + 2 \sum_{v \in M_K^c} \lambda_v(nP) = [K : \mathbb{Q}] \hat{h}(nP) - \sum_p n_p \lambda_p(nP) \\
= [K : \mathbb{Q}] \hat{h}(nP) - L(nP) + \frac{1}{6} \log \mathcal{N}(M_E) \quad \text{by Lemma 2.2.3} \\
\leq [K : \mathbb{Q}] \hat{h}(nP) - D_E(n) + \frac{1}{6} \log \mathcal{N}(M_E) \quad \text{by Proposition 2.3.1(2)} \\
\leq [K : \mathbb{Q}] n^2 \mu - D_E(n) + \frac{1}{6} \log \mathcal{N}(M_E) \quad \text{since } \hat{h}(P) \leq \mu.
\]

Combining this with (2.3) and taking the exponential, we obtain

\[
\left( \prod_{v \in M_K^r} \max\{1, |x(nP)|_v \} \right) \left( \prod_{v \in M_K^c} \max\{1, |x(nP)|_v \}^2 \right) \leq B_n(\mu).
\]

But the left-hand side is at least 1. Thus, if \( B_n(\mu) < 1 \), then we have a contradiction, i.e., \( \hat{h}(P) > \mu \) for all non-torsion \( P \in E_{gr}(K) \). On the other hand, it can be seen that \( |x(nP)|_v \leq B_n(\mu) \) for all \( v \in M_K^r \), and \( |x(nP)|_v^2 \leq B_n(\mu) \) for all \( v \in M_K^c \). \( \square \)

We are now ready to state an explicit formula for a lower bound on \( E_{gr}(K) \).
Theorem 2.4.2. Let \( p \) be a prime ideal such that

\[
\mathcal{N}(p) > \left( \prod_{v \in M_K^r} \sqrt{\alpha_v} \right) \left( \prod_{v \in M_K^c} \alpha_v \right) \mathcal{N}(M_E)^{1/12}.
\]

(2.4)

Set \( n = e_p \) and

\[
\mu_0 = \frac{1}{[K : \mathbb{Q}] n^2} \left( D_E(n) - \sum_{v \in M_K^r} \log \alpha_v - 2 \sum_{v \in M_K^c} \log \alpha_v - \frac{1}{6} \log \mathcal{N}(M_E) \right).
\]

Then \( \mu_0 > 0 \), and \( \hat{h}(P) \geq \mu_0 \) for all non-torsion \( P \in E_{gr}(K) \).

Proof. Suppose \( p \) is a prime ideal satisfying (2.4). By definition of \( D_E(n) \) (see (2.2)), we have

\[
D_E(n) \geq 2 \log \mathcal{N}(p) > \sum_{v \in M_K^r} \log \alpha_v + 2 \sum_{v \in M_K^c} \log \alpha_v + \frac{1}{6} \log \mathcal{N}(M_E),
\]

which implies that \( \mu_0 > 0 \). Then for any \( \mu < \mu_0 \), we have

\[
[K : \mathbb{Q}] n^2 \mu - D_E(n) + \sum_{v \in M_K^r} \log \alpha_v + 2 \sum_{v \in M_K^c} \log \alpha_v + \frac{1}{6} \log \mathcal{N}(M_E)
\]

\[
< [K : \mathbb{Q}] n^2 \mu_0 - D_E(n) + \sum_{v \in M_K^r} \log \alpha_v + 2 \sum_{v \in M_K^c} \log \alpha_v + \frac{1}{6} \log \mathcal{N}(M_E) = 0.
\]

Thus \( B_n(\mu) < 1 \), and so \( \hat{h}(P) > \mu \) for all non-torsion \( P \in E_{gr}(K) \) by Proposition 2.4.1. Since this is true for all \( \mu < \mu_0 \), then \( \hat{h}(P) \geq \mu_0 \) as claimed.

Although it is possible to obtain a lower bound for the canonical height on \( E_{gr}(K) \) simply from this theorem, our practical experience shows that this bound is not as good as the one obtained by collecting more information on \( x(nP) \). This claim will be illustrated in Example 5.1.1.
2.5 Solving Inequalities I: Real Embeddings

In order to obtain a larger positive lower bound on $E_{gr}(K)$ than the one obtained by Theorem 2.4.2, we finally concentrate on how to derive an alternative criterion for deciding whether a given $\mu > 0$ is a lower bound. This new criterion, which requires more information on $x(nP)$, will involve solving a system of certain inequalities on each embedding $E^{(v)}$, for every $v \in M^r_K \cup M^c_K$.

Given $\mu > 0$, we wish to check whether $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{gr}(K)$. If $B_n(\mu) < 1$ for some $n > 0$, then it follows easily from Proposition 2.4.1 that $\mu$ is a lower bound. On the other hand, if no such $n$ exists, then Proposition 2.4.1 states that all non-torsion $P \in E_{gr}(K)$ with $\hat{h}(P) \leq \mu$ must satisfy

$$|x(nP)|_v \leq \begin{cases} B_n(\mu) & \text{if } v \in M^r_K, \\ \sqrt{B_n(\mu)} & \text{if } v \in M^c_K, \end{cases}$$

for all $n > 0$. This can be regarded as a system of inequalities on each embedding $E^{(v)}$. In particular, if such a system has no solution, then this contradicts our assumption that $\hat{h}(P) \leq \mu$ for some non-torsion $P \in E_{gr}(K)$, so $\mu$ must be a lower bound on $E_{gr}(K)$.

In this section, we will explain how to solve this system of inequalities on each real embedding $E^{(v)}$ (i.e., where $v \in M^r_K$). A similar computation on each complex embedding, however, is more sophisticated, and hence will be explained later in Chapter 3. To prove that $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{gr}(K)$, we attempt to derive a contradiction from these inequalities using an application of period lattices and elliptic logarithms, which will be fully described in Chapter 4.

2.5.1 Periods and Elliptic Logarithms

We will now introduce a simplified definition of periods and elliptic logarithms on elliptic curves over $\mathbb{R}$, and use it to obtain a contradiction from the system of
inequalities mentioned earlier on each real embedding $E^{(v)}$.

For $v \in M'_K$, recall that $E^{(v)}$ is of the form

$$E^{(v)} : y^2 + \sigma_v(a_1)xy + \sigma_v(a_3)y = x^3 + \sigma_v(a_2)x^2 + \sigma_v(a_4)x + \sigma_v(a_6),$$

where $\sigma_v$ is the associated embedding from $K$ to $\mathbb{R}$. With the change of variables

$$x = X - \frac{\sigma_v(b_2)}{12}, \quad y = \frac{Y - \sigma_v(a_1)x - \sigma_v(a_3)}{2},$$

we can rewrite $E^{(v)}$ as

$$Y^2 = 4(X - e_1)(X - e_2)(X - e_3)$$

for some $e_1, e_2, e_3$ with $\sum_{j=1}^{3} e_j = 0$. Since $E^{(v)}$ is defined over $\mathbb{R}$, then either all $e_j \in \mathbb{R}$, or there is only one $e_j \in \mathbb{R}$. Without loss of generality, we can assume that $e_3$ is the largest real root.

Recall from Section 1.3 that the connected component $E^{(v)}_0$ can be parameterised by the real line $\{\Omega_v : 0 \leq t < 1\}$, where $\Omega_v \in \mathbb{R}$ is one of the periods generating the period lattice of $E^{(v)}$. We will see in Chapter 4 that $\Omega_v$ is uniquely determined up to sign, but for now we shall take $\Omega_v > 0$. It then follows from (1.10) that

$$\Omega_v = 2 \int_{(e_3,0)}^{O} \frac{dX}{Y} = 2 \int_{\beta_v}^{\infty} \frac{dx}{\sqrt{f_v(x)}}$$

(we rearrange $O$ and $(e_3,0)$ so that $\Omega_v > 0$), where

$$f_v(x) = 4x^3 + \sigma_v(b_2)x^2 + 2\sigma_v(b_4)x + \sigma_v(b_6),$$

and $\beta_v = e_3 - \frac{\sigma_v(b_2)}{12}$ is the largest real root of $f_v$. If $\xi$ is a real number satisfying $\xi \geq \beta_v$, then there exists $\eta$ such that $2\eta + \sigma_v(a_1)\xi + \sigma_v(a_3) \geq 0$ and $P = (\xi, \eta) \in$
An elliptic logarithm of $P$ is then obtained in a similar way, i.e.,

$$z_{P,v} = \int_{\mathcal{O}}^{P} dX \frac{dX}{\mathcal{Y}} = \int_{-\infty}^{\xi} \frac{dx}{\sqrt{f_{v}(x)}} \quad (\text{mod } \Omega_{v}).$$

Note that $z_{P,v} \in [\Omega_{v}/2, \Omega_{v}]$. Moreover, we may take $z_{-P,v} = -z_{P,v} \pmod{\Omega_{v}}$ (so that $z_{-P,v} \in [0, \Omega_{v}/2]$). At this point, one may use [Coh93, Algorithm 7.4.7 and 7.4.8] to compute $\Omega_{v}$ and $z_{P,v}$ respectively. We will explain a complete method for computing period lattices and elliptic logarithms on elliptic curves over $\mathbb{C}$ in Chapter 4.

For convenience, we shall define $\varphi_{v} : E_{0}^{(v)}(\mathbb{R}) \to [0, 1)$, the normalised elliptic logarithm, by

$$\varphi_{v}(P) = \varphi_{v}((\xi, \eta)) = \begin{cases} \frac{z_{P,v}}{\Omega_{v}} & \text{if } 2\eta + \sigma_{v}(a_1)\xi + \sigma_{v}(a_3) \geq 0, \\ 1 - \varphi_{v}(-P) & \text{otherwise}. \end{cases}$$

For $\xi \geq \beta_{v}$, we also define

$$\psi_{v}(\xi) = \varphi_{v}((\xi, \eta)) \in [1/2, 1),$$

where $(\xi, \eta) \in E_{0}^{(v)}(\mathbb{R})$ with $2\eta + \sigma_{v}(a_1)\xi + \sigma_{v}(a_3) \geq 0$. In other words, $\psi_{v}(\xi)$ is the normalised elliptic logarithm of the “higher” of the two points on $E_{0}^{(v)}$ with $x$-coordinate $\xi$.

For real $\xi_1, \xi_2$ with $\xi_1 < \xi_2$, we define the subset $S^{(v)} \subset [0, 1)$ as follows:

$$S^{(v)}(\xi_1, \xi_2) = \begin{cases} \emptyset & \text{if } \xi_2 < \beta_{v}, \\ [1 - \psi_{v}(\xi_2), \psi_{v}(\xi_2)] & \text{if } \xi_1 < \beta_{v} \leq \xi_2, \\ [1 - \psi_{v}(\xi_2), 1 - \psi_{v}(\xi_1)] \cup [\psi_{v}(\xi_1), \psi_{v}(\xi_2)] & \text{if } \xi_1 \geq \beta_{v}. \end{cases}$$

The following lemma is clear.
Lemma 2.5.1. Suppose $\xi_1 < \xi_2$ are real numbers. Then $P \in E_0^{(v)}(\mathbb{R})$ satisfies $\xi_1 \leq x(P) \leq \xi_2$ if and only if $\varphi_v(P) \in S^{(v)}(\xi_1, \xi_2)$.

If $\bigcup_j [a_j, b_j]$ is a disjoint union of intervals and $\alpha \in \mathbb{R}$, we define

$$\alpha + \bigcup_j [a_j, b_j] = \bigcup_j [a_j + \alpha, b_j + \alpha],$$

$$\alpha \bigcup_j [a_j, b_j] = \bigcup_j [\alpha a_j, \alpha b_j] \quad (\text{for } \alpha > 0).$$

Lemma 2.5.2. Suppose $\xi_1 < \xi_2$, and $n \in \mathbb{Z}_{>0}$. Let

$$S_n^{(v)}(\xi_1, \xi_2) = \bigcup_{\alpha=0}^{n-1} \left( \frac{\alpha}{n} + \frac{1}{n} S^{(v)}(\xi_1, \xi_2) \right).$$

Then $P \in E_0^{(v)}(\mathbb{R})$ satisfies $\xi_1 \leq x(nP) \leq \xi_2$ if and only if $\varphi_v(P) \in S_n^{(v)}(\xi_1, \xi_2)$.

Proof. By Lemma 2.5.1, $P \in E_0^{(v)}(\mathbb{R})$ satisfies $\xi_1 \leq x(P) \leq \xi_2$ if and only if $\varphi_v(P) \in S^{(v)}(\xi_1, \xi_2)$. Let $\nu_n$ be the multiplication-by-$n$ map on $\mathbb{R}/\mathbb{Z}$. If $\delta \in [0, 1)$, then

$$\nu_n^{-1}(\delta) = \left\{ \frac{\alpha}{n} + \frac{\delta}{n} : \alpha = 0, 1, 2, \ldots, n-1 \right\}.$$

But since $\varphi_v$ is an isomorphism, we have $\varphi_v(nP) = n\varphi_v(P) \pmod{1}$. Hence

$$\varphi_v(nP) \in S^{(v)}(\xi_1, \xi_2) \iff \varphi_v(P) \in \nu_n^{-1}(S^{(v)}(\xi_1, \xi_2)) = S_n^{(v)}(\xi_1, \xi_2).$$

□

This together with Proposition 2.4.1 leads to the following proposition.

Proposition 2.5.3. If $B_n(\mu) < 1$ for some integer $n > 0$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{gr}(K)$. If $B_n(\mu) \geq 1$ for all $n = 1, \ldots, n_{\text{max}}$, then every non-torsion point $P \in E_{gr}(K)$ with $h(P) \leq \mu$ satisfies

$$\varphi_v(\sigma_v(P)) \in \bigcap_{n=1}^{n_{\text{max}}} S_n^{(v)}(-B_n(\mu), B_n(\mu))$$
for every $v \in M_K^r$. Here, $\sigma_v : K \to \mathbb{R}$ is the real embedding of $K$ associated to $v$.

In particular, if the intersection is empty for some $v \in M_K^r$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{gr}(K)$.

Finally, we remark that if $K$ is also a totally real number field (i.e., $M_K^c = \emptyset$), then Proposition 2.5.3 alone will suffice for computing a lower bound for the canonical height on $E/K$. We shall not discuss such computation in detail here, since this will be a special case of our algorithm to be developed in Section 3.4. Some examples illustrating the applications of this algorithm will be shown in Chapter 5.

To summarise, we have developed all necessary formulas for estimating local heights, which leads to a criterion for deciding if a given $\mu > 0$ is a lower bound for the canonical height. Such criterion requires solving a system of certain inequalities on each embedding of $E$. In this chapter, we have managed to do this for real embeddings, which turns out to be sufficient for computing a lower bound for the canonical height on elliptic curves over number fields with at least one real embedding. The next chapter will focus on our remaining task, i.e., solving inequalities on complex embeddings.
Chapter 3

Height Bound II: Complex Embeddings

In this chapter, we will continue our work on computing a lower bound for the canonical height from Chapter 2 by introducing a new method for solving a system of certain inequalities on complex embeddings. This together with our work we have done so far will allow us to compute such a lower bound on elliptic curves over number fields in general, which will complete our work on height bound.

Let $E$ be an elliptic curve defined over a number field $K$. Recall the definition of $B_n(\mu)$ in Section 2.4. If $B_n(\mu) \geq 1$, then Proposition 2.4.1 implies that all non-torsion $P \in E_{gr}(K)$ with $\hat{h}(P) \leq \mu$ satisfy $|x(nP)|_v \leq \sqrt{B_n(\mu)}$ for every $v \in M_K^c$.

By computing $B_n(\mu)$ for several $n \in \mathbb{Z}_{>0}$, this yields a system of certain inequalities on each complex embedding $E^{(v)}$. We will see later that each of these inequalities corresponds to a region in the fundamental parallelogram for the period lattice of $E^{(v)}$, and solving the system of these inequalities is equivalent to finding the intersection of all such regions.

A combined version of Chapter 2 and this chapter, which explains a complete algorithm for computing a lower bound for the canonical height on elliptic curves over number fields, has been published in [Tho10].
3.1 Corresponding Regions I: An Overview

In this section, we will describe how to visualise an inequality on the $x$-coordinate of points in $E^{(v)}(\mathbb{C})$ obtained by Proposition 2.4.1 as a corresponding region on the fundamental parallelogram for the period lattice of $E^{(v)}$.

3.1.1 Fundamental Parallelograms

For $v \in M^*_K$, let $E^{(v)}$ be the complex embedding of $E$ associated to $v$. As mentioned in Section 1.3, it is well known that there exists a complex analytic group isomorphism $\varphi_v : E^{(v)}(\mathbb{C}) \to \mathbb{C}/\Lambda$, for some lattice $\Lambda$ (for more details on computing this isomorphism, see Chapter 4).

**Definition.** Let $\Lambda$ be a lattice with $\mathbb{Z}$-basis $\{w_1, w_2\}$. The (closed) fundamental parallelogram for $\Lambda$ is the set

$$F_{w_1, w_2} = \{\lambda_1 w_1 + \lambda_2 w_2 : 0 \leq \lambda_1, \lambda_2 \leq 1\}.$$

Note that every element of $\mathbb{C}/\Lambda$ has a representative in $F_{w_1, w_2}$ which is unique except for points on the boundary of $F_{w_1, w_2}$. After choosing a lift in $F_{w_1, w_2}$ for each $P \in E^{(v)}(\mathbb{C})$, we may view $\varphi_v$ as a map $E^{(v)}(\mathbb{C}) \to F_{w_1, w_2} \subset \mathbb{C}$.

Without loss of generality, we can choose a $\mathbb{Z}$-basis for $\Lambda$ so that the quantity $\tau = w_2/w_1$ satisfies the following:

$$|\tau| \geq 1, \quad |\Re(\tau)| \leq 1/2, \quad \Im(\tau) \geq \sqrt{3}/2. \quad (3.1)$$

Let $\Lambda_\tau$ be the lattice generated by $1, \tau$. Then it is clear that the map $\delta : \mathbb{C} \to \mathbb{C}$ given by $z \mapsto z/w_1$ induces a bijection $\Lambda \to \Lambda_\tau$. To ease notation, we shall denote $F_{1, \tau}$ by $F_\tau$, and let $H_\tau$ be the “lower half” of $F_\tau$, i.e.,

$$H_\tau = \{\lambda_1 + \lambda_2 \tau : 0 \leq \lambda_1 \leq 1, \quad 0 \leq \lambda_2 \leq 1/2\}.$$
Let $\psi' = \delta \circ \varphi_v$ (viewed as a map $E^{(v)}(C) \to F_\tau$). Clearly, $\psi'$ maps each $P \in E^{(v)}(C)$ to a point $z \in F_\tau$, and maps either $P$ or $-P$ to a point in $\mathcal{H}_\tau$. Hence we can let

$$
\psi_v(P) = \begin{cases} 
\psi'_v(P) & \text{if } \psi'_v(P) \in \mathcal{H}_\tau, \\
\psi'_v(-P) & \text{if } \psi'_v(P) \notin \mathcal{H}_\tau,
\end{cases}
$$

(3.2)

so that $\psi_v(P) \in \mathcal{H}_\tau$ in all cases.

### 3.1.2 Visualising the Region

From now on, we shall always assume that our $\mathbb{Z}$-basis $\{w_1, w_2\}$ for $\Lambda$ is chosen so that $\tau$ satisfies (3.1). To see what the region corresponding to an inequality given by Proposition 2.4.1 looks like, we first recall that the Weierstrass parameterisation $\mathbb{C}/\Lambda_\tau \to E_W(\mathbb{C})$, where $E_W$ is the elliptic curve of the form $Y^2 = 4X^3 - g_2(\Lambda_\tau)X - g_3(\Lambda_\tau)$ (for the definition of $g_j$, see, e.g., [Was03, Section 9.2]), is given by

$$
z \mapsto (\wp_{\Lambda_\tau}(z), \wp'_{\Lambda_\tau}(z)).
$$

(3.3)

Suppose $E^{(v)}$ is given by a Weierstrass equation

$$
E^{(v)}: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6
$$

for some $a_j \in \mathbb{C}$. Then we have an isomorphism $E_W(\mathbb{C}) \to E^{(v)}(\mathbb{C})$, given by

$$
(X, Y) \mapsto (x, y) = \left( w_1^{-2}X - \frac{b_2}{12}, w_1^{-3}Y - a_4x - a_3 \right).
$$

Hence for any $\xi \geq 0$, it is immediate from the triangle inequality and (3.3) that $|x| \leq \xi$ if and only if $|\wp_{\Lambda_\tau}(z)| \leq U_\xi$, where

$$
U_\xi = |w_1|^2 \left( \xi + \frac{|b_2|}{12} \right).
$$
We can now consider the set \( \ell = \{ z \in H_\tau : |\wp_{\Lambda_\tau}(z)| = U_\xi \} \) as a curve\(^1\) on \( H_\tau \) (see Figure 3.1). This is the boundary of the region

\[
\mathcal{R}^{(v)}(\xi) = \{ z \in H_\tau : |\wp_{\Lambda_\tau}(z)| \leq U_\xi \}.
\]

Since the Weierstrass \( \wp \)-function becomes a one-to-one continuous map once its domain is restricted to \( H_\tau \), the equation \( |\wp_{\Lambda_\tau}(z)| = U_\xi \) yields only one curve on \( H_\tau \). By symmetry (about the mid-point of \( F_\tau \)), we also have another identical boundary on the upper half of \( F_\tau \). Depending on \( U_\xi \), the boundaries on both halves topologically form either one or two identical loops on the torus \( \mathbb{C}/\Lambda_\tau \), as shown in Figure 3.2.

![Figure 3.1: The boundary on \( H_\tau \) associated to different \( U_\xi \). Each curve is labelled by the relevant value of \( \xi \).](image)

### 3.2 Corresponding Regions II: Estimation

In practice, however, it is very difficult to determine the region \( \mathcal{R}^{(v)} \) exactly. For example, it is impossible to store an infinitesimal amount of its information on a computer. To circumvent this problem, we approximate \( \mathcal{R}^{(v)} \) by a finite number of parallelograms whose union covers \( \mathcal{R}^{(v)} \). Denote by \( S^{(v)} \) the finite set of these

\(^1\)This may have either one or two connected components on \( H_\tau \).
parallellograms. A finer approximation to $R^{(v)}$ then can be obtained by decreasing the size of parallelograms in $S^{(v)}$.

This section, to which most of our work on height bound is devoted, will focus on a number of approximation techniques which eventually allow us to construct $S^{(v)}$. For now we mention that $S^{(v)}(\xi)$ has the following properties:

1. $\bigcup_{C \in S^{(v)}(\xi)} C \supseteq R^{(v)}(\xi)$, i.e., the union of all parallelograms in $S^{(v)}(\xi)$ contains the actual region $R^{(v)}(\xi)$.

2. Every $C \in S^{(v)}(\xi)$ contains $z$ such that $|\wp_{\Lambda, \tau}(z)| \leq U_\xi$, hence $C \cap R^{(v)}(\xi) \neq \emptyset$ for all $C \in S^{(v)}(\xi)$.

### 3.2.1 The Weierstrass $\wp$-function

Let $q = \exp(2\pi i \tau)$ and let $u = \exp(2\pi i z)$ (where $i = \sqrt{-1}$). For $k \in \mathbb{Z}$, we define

$$f_k(z, \tau) = (2\pi i)^2 \left[ \frac{u}{(1-u)^2} + \frac{1}{12} + \sum_{j=1}^{k-1} \left( \frac{q^j u}{(1-q^j u)^2} + \frac{q^j u^{-1}}{(1-q^j u^{-1})^2} - \frac{2q^j}{(1-q^j)^2} \right) \right].$$

It can be seen from Proposition 1.3.1 that $\wp_{\Lambda, \tau}(z) = \lim_{k \to \infty} f_k(z, \tau)$ for all non-lattice points $z$. By choosing a suitable $k$, we can bound the error which occurs when $|f_k(z, \tau)|$ is used as the approximation to $|\wp_{\Lambda, \tau}(z)|$, as shown in the next lemma.
Lemma 3.2.1. For $z \in H_\tau$ with $z \neq 0, 1$, let $\alpha = \Im(z)/\Im(\tau)$. Define 

$$
\epsilon(k) = \frac{4\pi^2}{1 - |q|} \left( \frac{|q|^{k+\alpha}}{(1 - |q|^{k+\alpha})^2} + \frac{|q|^{k-\alpha}}{(1 - |q|^{k-\alpha})^2} + \frac{2|q|^k}{(1 - |q|^k)^2} \right).
$$

Then $||\varphi_{\Lambda,z}(z)| - |f_k(z, \tau)|| \leq \epsilon(k)$.

Proof. By Proposition 1.3.1, we have 

$$
\varphi_{\Lambda,z}(z) - f_k(z, \tau) = (2\pi i)^2 \sum_{j=k}^{\infty} \left[ \frac{q^j u}{(1 - q^j u)^2} + \frac{q^{j-1} u}{(1 - q^{j-1} u)^2} - \frac{2q^j}{(1 - q^j)^2} \right].
$$

Observe that $|u| = |q|^\alpha$. By the triangle inequality, we obtain 

$$
||\varphi_{\Lambda,z}(z)| - |f_k(z, \tau)|| \leq 4\pi^2 \sum_{j=k}^{\infty} \left[ \frac{|q|^{j+\alpha}}{(1 - |q|^{j+\alpha})^2} + \frac{|q|^{j-\alpha}}{(1 - |q|^{j-\alpha})^2} + \frac{2|q|^j}{(1 - |q|^j)^2} \right]. \quad (3.4)
$$

Since we work on $H_\tau$, we have $|q| < 1$ and $0 \leq \alpha \leq 1/2$, which implies that $|q|^{j+\alpha} < 1$ for all $j \geq 1$. Thus we have the estimate 

$$
\sum_{j=k}^{\infty} \frac{|q|^{j+\alpha}}{(1 - |q|^{j+\alpha})^2} \leq \frac{1}{(1 - |q|^{k+\alpha})^2} \sum_{j=k}^{\infty} |q|^{j+\alpha} \leq \frac{|q|^{k+\alpha}}{(1 - |q|^{k+\alpha})^2(1 - |q|)},
$$

and similarly, 

$$
\sum_{j=k}^{\infty} \frac{2|q|^j}{(1 - |q|^j)^2} \leq \frac{2|q|^k}{(1 - |q|^k)^2(1 - |q|)}.
$$

This together with (3.4) and the triangle inequality yields the result. \qed

One can easily verify that, in the range $0 \leq \alpha \leq 1/2$, the absolute error $\epsilon(k)$ given by Lemma 3.2.1 attains its maximum at $\alpha = 1/2$, and becomes smaller as $k$ increases. Moreover, it can be seen that $\epsilon(k)$ decreases as $\Im(\tau)$ increases. Some examples of maximum values for $\epsilon(k)$ are listed in Table 3.1 (based on $\alpha = 1/2$ and $\Im(\tau) = \sqrt{3}/2$).

Recall that every parallelogram $C$ in $S^{(v)}(\xi)$ satisfies $|\varphi_{\Lambda,z}(z)| \leq U_\xi$ for some $z \in C$. In practice, we can compute $|f_k(z, \tau)|$ and add it with the error given by
Table 3.1: Maximum values for $\epsilon(k)$

<table>
<thead>
<tr>
<th>$k$</th>
<th>Maximum error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.349</td>
</tr>
<tr>
<td>2</td>
<td>0.013</td>
</tr>
<tr>
<td>3</td>
<td>$5.568 \times 10^{-5}$</td>
</tr>
<tr>
<td>4</td>
<td>$2.413 \times 10^{-7}$</td>
</tr>
<tr>
<td>5</td>
<td>$1.046 \times 10^{-9}$</td>
</tr>
<tr>
<td>10</td>
<td>$1.598 \times 10^{-21}$</td>
</tr>
<tr>
<td>20</td>
<td>$3.731 \times 10^{-45}$</td>
</tr>
<tr>
<td>23</td>
<td>$3.036 \times 10^{-52}$</td>
</tr>
</tbody>
</table>

Lemma 3.2.1 to obtain a (small) interval which contains $|\varphi_{\Lambda_x}(z)|$. On each of the four line segments comprising the boundary of $C$, we can parameterise $|f_k(z, \tau)|$ by a real-valued function $f_k(x, \tau)$ or $f_k(y, \tau)$, where $x = \Re(z)$ and $y = \Im(z)$. We wish to find the range of $f_k$ when $x$ or $y$ varies along the line. For this computation, we find some techniques from *interval arithmetic* (see [Moo66]) to be very useful.

### 3.2.2 Interval Arithmetic

Before we proceed to its application, we shall first explain briefly what interval arithmetic is.

**Definition.** Let $I = [a, b]$ and $J = [c, d]$ (with $a \leq b$ and $c \leq d$) be two intervals of real numbers. An *arithmetic operation* on intervals $I, J$ is defined by

$$I * J = \{x \cdot y : a \leq x \leq b, c \leq y \leq d\},$$

where $*$ is an operation on real numbers.

A number of usual arithmetic operations on real numbers can be extended to the ones on intervals. For example,

$$I + J = [a + c, b + d], \quad I - J = [a - d, b - c],$$

$$I \cdot J = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}],$$

$$I/J = [a, b] \cdot [1/d, 1/c] \quad \text{(provided that } 0 \notin J).$$
3.2. Corresponding Regions II: Estimation

It can be seen easily that interval addition and interval multiplication are both associative and commutative. Distributivity, however, does not always hold for interval arithmetic. For example,

\[
[1, 3] \cdot ([1, 3] - [1, 3]) = [1, 3] \cdot [-2, 2] = [-6, 6], \text{ whereas }
\]

\[
[1, 3] \cdot [1, 3] - [1, 3] \cdot [1, 3] = [1, 9] - [1, 9] = [-8, 8].
\]

Instead, we always have subdistributivity, i.e., \( I \cdot (J + K) \subset I \cdot J + I \cdot K \) for all intervals \( I, J, K \).

One important property of interval arithmetic is that it is inclusion monotonic, i.e., if \( I \subset K \) and \( J \subset L \) are intervals, then

\[
I + J \subset K + L, \quad I - J \subset K - L, \quad I \cdot J \subset K \cdot L, \\
I/J \subset K/L \quad \text{(provided that } 0 \notin L). 
\]

This leads to the following theorem.

**Theorem 3.2.2** ([Moo66, Theorem 3.1]). *Let \( f(X_1, \ldots, X_n) \) be a rational expression with real coefficients in the interval variables \( X_1, \ldots, X_n \), i.e., a finite combination of \( X_1, \ldots, X_n \) and a finite set of constant intervals with interval arithmetic operations. Then

\[
X'_1 \subset X_1, \ldots, X'_n \subset X_n \quad \implies \quad f(X'_1, \ldots, X'_n) \subset f(X_1, \ldots, X_n)
\]

for every set of intervals \( X_1, \ldots, X_n \) for which the interval arithmetic operations in \( f \) are defined.*

Suppose \( f(x_1, \ldots, x_n) \) is a real rational expression, i.e., \( f \) is a quotient of real polynomials in terms of \( x_1, \ldots, x_n \). Then by Theorem 3.2.2, the resulting interval \( F = f(X_1, \ldots, X_n) \) will always contain the actual range of \( f(x_1, \ldots, x_n) \) for \( x_j \in X_j \). In particular, \( F \) will be the actual range of \( f(x_1, \ldots, x_n) \) for \( x_j \in X_j \) if each variable
$x_j$ occurs only once in $f$ (note that $x_j^2 = x_j \cdot x_j$ is taken as two occurrences). With some techniques, for example, using subdistributivity to group common terms in $f$, the resulting interval $F$ can be made smaller. For more information on this subject, see [Moo66, Chapter 3 and 6].

Recall the function $f_k(z, \tau)$ in Section 3.2.1. Suppose $z = x + iy \in \mathbb{C}$ is on a fixed line segment $L$. Depending on $L$, we can regard $z$ as a function of either $x$ or $y$ (for example, if $z$ is on a vertical line, then $x$ is fixed but $y$ varies). Thus, provided that $L$ is fixed and $z \in L$, we can consider the function $g(z) = |f_k(z, \tau)|^2$ as a real function of one real variable, i.e., either $g(z) = g(z(x))$ or $g(z) = g(z(y))$, depending on $L$. To ease notation, we shall write

$$f(*) = g(z(*)),$$

where $*$ is either $x$ or $y$, depending on how $z$ is parameterised along $L$.

The next proposition shows that we can apply interval arithmetic to $f(*)$.

**Proposition 3.2.3.** Define $f(*)$ as above. Then $f$ can be extended to a real rational expression of at most three interval variables, depending on the line segment $L$.

**Proof.** First, we note that

$$f(*) = |f_k(z, \tau)|^2 = \Re(f_k(z, \tau))^2 + \Im(f_k(z, \tau))^2.$$

We will show how to obtain the real part of $f_k(z, \tau)$; the imaginary part of $f_k(z, \tau)$ can be deduced in a similar way.

The real part of $f_k(z, \tau)$ consists of the real parts of the terms

$$\frac{u}{(1-u)^2}, \quad \frac{1}{12}, \quad \frac{q^j u}{(1-q^j u)^2}, \quad \frac{q^j u^{-1}}{(1-q^j u^{-1})^2}, \quad \frac{q^j}{(1-q^j)^2},$$

(3.5)
where \( u = \exp(2\pi iz) \) and \( q = \exp(2\pi i\tau) \). Write \( z = x + iy \). Let

\[
x_1 = \exp(-2\pi y), \quad x_2 = \cos(2\pi x), \quad x_3 = \sin(2\pi x).
\]

Consider the following two cases:

1. **If \( L \) is a non-vertical line** (i.e., \( y = \alpha x + \beta \) for some finite \( \alpha \) and \( \beta \)), then

\[
\Re \left( \frac{u}{(1-u)^2} \right) = \frac{x_1 x_2 (1 + x_1^2) - 2x_1^2}{(1 - 2x_1 x_2 + x_1^2)^2}
\]

Similarly, it can be shown that the real parts of the other terms in (3.5) can be written as rational expressions in terms of \( x_1, x_2, x_3 \).

2. **If \( L \) is a vertical line** (i.e., \( x \) is fixed), then we have \( \Re(u/(1-u)^2) \) as above. Since \( x_2 \) and \( x_3 \) are now constant, we have \( \Re(u/(1-u)^2) \) as a rational expression in terms of \( x_1 \) only. This is also the case for the real parts of the other terms in (3.5).

Thus we have \( f(\ast) \) as a real rational expression in terms of \( x_1, x_2, x_3 \). Suppose that \( a \leq x \leq b \) and \( c \leq y \leq d \) on \( L \) (note that \( c, d \geq 0 \) since we work on \( H_\tau \)). Let

\[
X_1 = \exp(-2\pi[c, d]) = [\exp(-2\pi d), \exp(-2\pi c)],
\]

\[
X_2 = \cos(2\pi[a, b]) = [\min_{a \leq x \leq b} \cos(2\pi x), \max_{a \leq x \leq b} \cos(2\pi x)],
\]

\[
X_3 = \sin(2\pi[a, b]) = [\min_{a \leq x \leq b} \sin(2\pi x), \max_{a \leq x \leq b} \sin(2\pi x)].
\]

(3.6)

After replacing \( x_1, x_2, x_3 \) in \( f \) with \( X_1, X_2, X_3 \) respectively, we finally obtain the interval version of \( f \).

Since \( f(X_1, X_2, X_3) \) is a real rational expression of interval variables, then Theorem 3.2.2 applies. Together with the error term in Lemma 3.2.1, the following proposition is immediate.
Proposition 3.2.4. Define $X_1, X_2, X_3$ to be the intervals depending on a line segment $L$ as in (3.6). For a fixed $k \in \mathbb{Z}_{>0}$, let $\epsilon = \epsilon(k)$ be the maximum absolute error given by Lemma 3.2.1. Then for all $z \in L$, we have

$$\sqrt{u_1} - \epsilon \leq |\varphi_{\Lambda_+}(z)| \leq \sqrt{u_2} + \epsilon,$$

where $[u_1, u_2] = f(X_1, X_2, X_3)$ (with $u_2 \geq u_1 \geq 0$).

### 3.2.3 Approximate Corresponding Regions

We are now ready to construct $S^{(v)}$, which in turn yields an approximation to the corresponding region $R^{(v)}$.

Let $L$ be a line segment in the complex plane. By Proposition 3.2.4, the interval

$$I(L) = [\sqrt{u_1} - \epsilon, \sqrt{u_2} + \epsilon]$$

contains the actual range of $|\varphi_{\Lambda_+}(z)|$ for $z \in L$. We can then extend this notion to any parallelogram $C$ by letting

$$I(C) = \bigcup_{L \in \partial C} I(L),$$

where $\partial C$ is the boundary of $C$. Note that the four intervals $I(L)$ for $L \in \partial C$ will overlap, so $I(C)$ is an interval.

For $v \in M^c_K$ and $\xi \geq 0$, we define $S^{(v)}(\xi)$ recursively as follows. First we let

$$S^{(v,0)}(\xi) = \{H_\tau\}.$$

Next, for $r \geq 0$, suppose $S^{(v,r)}(\xi) = \{C_1, \ldots, C_m\}$, where $m = 4^r$. Let

$$S^{(v,r+1)} = \{C_{11}, \ldots, C_{14}, \ldots, C_{m1}, \ldots, C_{m4} : C_j = \bigcup_{k=1}^{4} C_{jk}\},$$
3.2. Corresponding Regions II: Estimation

![Figure 3.3: Four quarters of $C_j$](image)

i.e., $C_{j1}, \ldots, C_{j4}$ are the four quarters of $C_j$, as shown in Figure 3.3.

Suppose $E^{(v)}$ is of the form $Y^2 = 4X^3 + AX + B$ for some $A, B \in \mathbb{C}$. Let $P \in E^{(v)}(\mathbb{C})$ be a point with $X(P) = 0$. Let $C_0 \in S^{(v,r+1)}$ be the parallelogram containing $\psi_v(P)$ (see (3.2) for its definition). Note that we may have $I(C_0) \cap [0, U_\xi] = \emptyset$. Then we define

$$S^{(v,r+1)}(\xi) = \{C_0\} \cup \{C \in S^{(v,r+1)}: I(C) \cap [0, U_\xi] \neq \emptyset\}.$$ 

Finally, we let $S^{(v)}(\xi) = S^{(v,r)}(\xi)$ for some $r > 0$.

For a set $S$ of parallelograms in $\mathbb{C}$, we denote $\bigcup_{C \in S} C$ simply by $\bigcup S$. It is then obvious from the construction above that

$$\bigcup S^{(v,0)}(\xi) \supset \bigcup S^{(v,1)}(\xi) \supset \cdots \supset \bigcup S^{(v,r)}(\xi) \supset \cdots \supset \mathcal{R}^{(v)}(\xi).$$

In other words, our approximation to $\mathcal{R}^{(v)}$ becomes finer as $r$ increases.

In general, computing $S^{(v,r+1)}$ with above definition can be very time-consuming. Fortunately, we can usually speed up this process using a combination of the following techniques.

**Lemma 3.2.5 (Four-Corner Test).** Suppose $C \in S^{(v,r+1)}(\xi)$. Let $z_1, \ldots, z_4$ be the corners of $C$, and let $\epsilon = \epsilon(k)$ be the maximum absolute error given by Lemma 3.2.1 for some fixed $k \in \mathbb{Z}_{>0}$. Define

$$I(z) = \left[ |f_k(z, \tau)| - \epsilon, |f_k(z, \tau)| + \epsilon \right].$$
If $I(z_j) \subset [0, U_\xi]$ for some $j = 1, \ldots, 4$, then $C \in S^{(v,r+1)}(\xi)$.

**Proof.** If such condition holds, then we simply have $|\wp_{\Lambda_r}(z)| \leq U_\xi$ for some $z \in C$, namely, $z = z_j$. Hence $C \in S^{(v,r+1)}(\xi)$. \hfill \Box

In practice, checking whether $C$ is in $S^{(v,r+1)}(\xi)$ by this test is considerably faster than the usual criterion $I(C) \cap [0, U_\xi]$. The next lemma provides a quick way to exclude all parallelograms which are not in $S^{(v,r+1)}(\xi)$.

**Lemma 3.2.6.** For $r \geq 0$, let $S_{r+1}$ be the set of all parallelograms in $S^{(v,r+1)}(\xi)$ which satisfy the condition in Lemma 3.2.5. Let

$$\partial S_{r+1} = \{C \in S^{(v,r+1)}(\xi) \setminus S_{r+1} : C \text{ is adjacent to } \bigcup S_{r+1}\}.$$ 

If $I(C) \cap [0, U_\xi] = \emptyset$ for all $C \in \partial S_{r+1}$, then $S^{(v,r+1)}(\xi) = S_{r+1}$.

**Proof.** If all parallelograms in $\partial S_{r+1}$ are excluded from $S^{(v,r+1)}(\xi)$, then this means that there is no part of the boundary $\ell$ of the actual region $R^{(v)}(\xi)$ passing through $\bigcup \partial S_{r+1}$. Thus the one-to-one and continuity properties of the Weierstrass $\wp$-function on $H_\tau$ imply that the boundary $\ell$ of $R^{(v)}(\xi)$ lies entirely in $\bigcup S_{r+1}$, and so all parallelograms in $S^{(v,r+1)}(\xi) \setminus S_{r+1}$ can be discarded. \hfill \Box

An illustration of using these lemmas to construct $S^{(v)}$ is shown\(^2\) in Figure 3.4. In this figure, the process of determining $S^{(v)}$ consists of the following steps:

1. Starting with $S^{(v,r+1)}(\xi)$ for some $r$, we use Lemma 3.2.5 to identify a number of parallelograms $C \in S^{(v,r+1)}(\xi)$ which are also in $S^{(v,r+1)}(\xi)$ (these are marked by "*"). Let $S_{r+1}$ be the set of all such parallelograms $C$.

2. Identify all parallelograms in $\partial S_{r+1}$ (these are marked by "?").

3. For each $C \in \partial S_{r+1}$, check if $I(C) \cap [0, U_\xi] = \emptyset$. If so, then $C \notin S^{(v,r+1)}(\xi)$ and thus can be discarded (this is marked by ".").

\(^2\)Here $S^{(v)} = S^{(v,3)}(0.4)$ for the elliptic curve $y^2 = x^3 + x + (1 + 4i)$ defined over $\mathbb{Q}(i)$. 

\[\text{\tt |}\text{\tt \ )\}}\text{\tt |}\text{\tt 3. Height Bound II: Complex Embeddings}\]
Figure 3.4: An illustration of how to obtain $S^{(v)}$. The top-left entry represents the parallelogram containing $z = 0$.

4. If it turns out that the set $\partial S_{r+1}$ is entirely discarded, then by Lemma 3.2.6, we have $S^{(v,r+1)}(\xi) = S_{r+1}$. In other words, every parallelogram in $S^{(v,r+1)}(\xi) \setminus S_{r+1}$ is discarded. Finally, we let $S^{(v)}(\xi) = S^{(v,r+1)}(\xi)$.

3.3 Solving Inequalities II: Complex Embeddings

In this section, we finally explain how to solve a system of inequalities given by Proposition 2.4.1 on complex embeddings, which is analogous to our previous result in Section 2.5.

As we have already seen, the inequality $|x(P)|_v \leq \xi$ yields the cover $\bigcup S^{(v)}(\xi)$ which approximates the corresponding region $R^{(v)}$ in $\mathcal{H}_r$. Since the Weierstrass $\wp$-function is even, we also have another identical region in the upper half of $F_r$. Let $T^{(v)}(\xi)$ be the union of both regions. Then clearly $T^{(v)}(\xi)$ contains the set

$$\{ z \in F_r : |\wp_{\Lambda_{\tau}}(z)| \leq U_{\xi} \}.$$

Recall the isomorphism $\psi'_v : E^{(v)}(\mathbb{C}) \rightarrow F_r$ from Section 3.1.1. Given a point $P \in E^{(v)}(\mathbb{C})$, we wish to consider all points $Q \in E^{(v)}(\mathbb{C})$ such that $P = nQ$. Let $z = \psi'_v(P)$ and $z' = \psi'_v(Q)$. Then we have

$$z = nz' \pmod{\Lambda_{\tau}}.$$
In fact, if \( z = \alpha + \beta \tau \) for some \( 0 \leq \alpha, \beta \leq 1 \), then

\[
z' \in \left\{ \frac{\alpha + s}{n} + \frac{(\beta + t) \tau}{n} : 0 \leq s, t \leq n - 1 \right\}.
\]

This therefore allows us to “divide” \( T^{(v)}(\xi) \) by \( n \) (see Figure 3.5 for an illustration) to obtain a new region

\[
T^{(v)}_n(\xi) = \{ z' \in F_\tau : nz' \pmod{\Lambda_\tau} \in C \text{ for some } C \in T^{(v)}(\xi) \}.
\]

Due to the symmetry of \( T^{(v)}_n \), we can let

\[
T^{(v)}_n(\xi) = T^{(v)}_n(\xi) \cap H_\tau.
\]

The following lemma is analogous to Lemma 2.5.2.

**Lemma 3.3.1.** If \( P \in E^{(v)}(\mathbb{C}) \) satisfies \( |x(nP)| \leq \xi \), then \( \psi_v(P) \in T^{(v)}_n(\xi) \).

**Proof.** If \( |x(nP)| \leq \xi \), then we have \( \psi_v(nP) \in C \) for some \( C \in S^{(v)}(\xi) \subset T^{(v)}(\xi) \).

Since \( n\psi_v(P) \) is either \( \psi_v(nP) \) or \( -\psi_v(nP) \pmod{\Lambda_\tau} \), in any case we have \( \psi_v(P) \in T^{(v)}_n(\xi) \cap H_\tau = T^{(v)}_n(\xi) \).

The next proposition, which is analogous to Proposition 2.5.3, follows easily from the previous lemma together with Proposition 2.4.1.

**Proposition 3.3.2.** If \( B_n(\mu) \geq 1 \) for all \( n = 1, \ldots, n_{\text{max}} \), then every non-torsion point \( P \in E_{gr}(K) \) with \( \hat{h}(P) \leq \mu \) satisfies

\[
\psi_v(\sigma_v(P)) \in \bigcap_{n=1}^{n_{\text{max}}} T^{(v)}_n\left(\sqrt{B_n(\mu)}\right)
\]
for all $v \in M_k^r$. Here, $\sigma_v : K \rightarrow \mathbb{C}$ is the complex embedding of $K$ associated to $v$.

In particular, if the intersection is empty for some $v \in M_k^c$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{\text{gr}}(K)$.

## 3.4 An Algorithm for Height Bound

Combining Proposition 2.5.3 and Proposition 3.3.2, we are now ready to state our main theorem.

**Theorem 3.4.1.** Let $\mu > 0$. If $B_n(\mu) < 1$ for some $n \in \mathbb{Z}_{>0}$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{\text{gr}}(K)$. Otherwise, if $B_n(\mu) \geq 1$ for all $n = 1, \ldots, n_{\text{max}}$, then every non-torsion point $P \in E_{\text{gr}}(K)$ with $\hat{h}(P) \leq \mu$ satisfies

$$\varphi_v(\sigma_v(P)) \in \bigcap_{n=1}^{n_{\text{max}}} S_n^{(v)}(-B_n(\mu), B_n(\mu))$$

for every $v \in M_k^r$, and moreover,

$$\psi_v(\sigma_v(P)) \in \bigcap_{n=1}^{n_{\text{max}}} T_n^{(v)}(\sqrt{B_n(\mu)})$$

for every $v \in M_k^c$.

In particular, if one of the intersections is empty for some $v \in M_k^r \cup M_k^c$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{\text{gr}}(K)$.

Theorem 3.4.1 in turn yields an algorithm for computing a lower bound for the canonical height on $E_{\text{gr}}(K)$, which consists of the following steps:

1. Given an initial value $\mu > 0$ and the number of steps $n_{\text{max}}$, we start by computing $B_n(\mu)$ for $n = 1, \ldots, n_{\text{max}}$. If $B_n(\mu) < 1$ for some $n$, then we can conclude immediately that $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{\text{gr}}(K)$.

2. Otherwise, we proceed to compute $\bigcap_{n=1}^{n_{\text{max}}} S_n^{(v)}(-B_n(\mu), B_n(\mu))$ for every $v \in M_k^r$. If the intersection is empty for some $v$, then again $\hat{h}(P) > \mu$ for all
non-torsion $P \in E_{\text{gr}}(K)$.

3. If not, then we compute $\cap_{n=1}^{n_{\text{max}}} T_n^{(v)}(\sqrt{B_n(\mu)})$ for every $v \in M_K^c$. Again, if the intersection is empty for some $v$, then $\hat{h}(P) > \mu$ for all non-torsion $P \in E_{\text{gr}}(K)$. Otherwise, we fail to show that $\mu$ is a lower bound on $E_{\text{gr}}(K)$.

4. We can refine $\mu$ further in the following way: if $\mu$ is shown to be a lower bound, then we increase $\mu$ and repeat the process to see if it is still a lower bound. However, if the algorithm fails to show that $\mu$ is a lower bound, then we decrease $\mu$ (or increase $n_{\text{max}}$) and repeat the process.

5. Return the largest value of $\mu$ which is known to be a lower bound for $E_{\text{gr}}(K)$.

Once $\mu$ is determined, we can simply use Lemma 2.1.1 to obtain a positive lower bound for the canonical height on $E(K)$. Some examples on how to compute such a lower bound using this algorithm (see Appendix A.3 for its MAGMA code) will be shown in Chapter 5.

3.5 Remarks

Finally, it should be noted that the lower bound we obtain is not model-independent, unlike the one of Hindry and Silverman [HS88, Theorem 0.3]. For example, the values $\alpha_v$ in Section 2.2.2 depend on the coefficients of the Weierstrass equation of $E$. At present, we have not systematically investigated how the bound obtained by our algorithm is affected by a change of model. As mentioned in Chapter 2, however, our formulas can be simplified if $E$ is given by a globally minimal model.

Regarding the computational complexity, it can be seen that computing $B_n(\mu)$ is less time-consuming than computing $S_n^{(v)}$, which in turn is less time-consuming than computing $T_n^{(v)}$. Therefore it is plausible to use $B_n(\mu)$ as the first criterion, followed by the intersection of $S_n^{(v)}$ and $T_n^{(v)}$ respectively, as we do in our algorithm.
3.5. Remarks

Let $c$ be the least common multiple of all Tamagawa indices as in Section 2.1. As pointed out by an anonymous referee of [Tho10], it may be possible to obtain a larger lower bound by making use of the explicit formulas for the local heights at non-archimedean places of bad reduction (see, e.g., [Sil88, Theorem 5.2]), provided that $c$ is large. This approach, however, is different to ours which uses the subgroup of points of good reduction. In particular, our lower bound on $E(K)$ will be small if $c$ is large. Nevertheless, it might be an interesting area for further study.

In conclusion, we have completed our work on height bound by introducing a method for solving a system of certain inequalities on complex embeddings. Our method involves a number of approximation techniques which eventually yield an approximate region corresponding to each inequality, where finding a solution to the system of these inequalities is equivalent to finding the intersection of all such regions. Together with our results from Chapter 2, we finally obtain an algorithm for computing a lower bound for the canonical height on elliptic curves over number fields in general.

Finally, in order to solve a system of inequalities using the methods in Section 2.5 and 3.3, we need to compute period lattices of certain real and complex embeddings, as well as elliptic logarithms of certain real and complex points. Nevertheless, algorithms for determining both quantities are currently available only for elliptic curves over $\mathbb{R}$ (see Section 1.3 for more discussion). Motivated by this problem, the next chapter will aim to develop a complete method for computing period lattices of elliptic curves over $\mathbb{C}$, and elliptic logarithms of complex points.
Chapter 4

Period Lattices and Complex Elliptic Logarithms

We will now move on to the second main result of this thesis, where we present a complete method for computing period lattices of elliptic curves over \( \mathbb{C} \), and then generalise it to compute elliptic logarithms of complex points. Based on the complex arithmetic-geometric mean (AGM) first studied by Gauss, our method will allow one to compute both quantities to a high degree of precision very quickly. For more background on this chapter, see Section 1.3.

The work in this chapter is done in collaboration with Professor John E. Cremona at the University of Warwick. Another version of this chapter has been submitted for publication as a joint paper [CT].

4.1 Introduction

In this chapter, we will assume that an elliptic curve \( E \) is defined over \( \mathbb{C} \), and is given by a Weierstrass equation of the form

\[
E : Y^2 = 4(X - e_1)(X - e_2)(X - e_3),
\]
where all the roots $e_j \in \mathbb{C}$ are distinct and $\sum_j e_j = 0$. As mentioned in Section 1.3, it is well known that there exists an isomorphism (of complex analytic Lie groups) $\mathbb{C}/\Lambda \rightarrow E(\mathbb{C})$ for some lattice $\Lambda$, given by the map

$$
\begin{align*}
z \pmod{\Lambda} & \mapsto P = (\wp_{\Lambda}(z), \wp'_{\Lambda}(z)) \\
0 \pmod{\Lambda} & \mapsto O.
\end{align*}
$$

(4.1)

**Definition.** Let $E$ be an elliptic curve defined over $\mathbb{C}$, the *period lattice* of $E$ is the lattice $\Lambda$ for which $E(\mathbb{C}) \cong \mathbb{C}/\Lambda$ via (4.1).

To be precise, we take $\Lambda$ to be the lattice of periods of the invariant differential $dX/Y$ on $E$. It is a discrete subgroup of $\mathbb{C}$ spanned by a $\mathbb{Z}$-basis $\{w_1, w_2\}$ with $w_2/w_1 \notin \mathbb{R}$.

**Definition.** The inverse map of (4.1) is called the *elliptic logarithm*. For $P \in E(\mathbb{C})$, we say that a value $z$ such that

$$
P \mapsto z \pmod{\Lambda}
$$

via this inverse is an *elliptic logarithm* of $P$ (note that $z$ is determined modulo $\Lambda$).

From this, two natural questions are:

1. Given a Weierstrass equation of $E$, how can we compute a $\mathbb{Z}$-basis for its period lattice $\Lambda$?

2. Given a point $P \in E(\mathbb{C})$, how can we compute its elliptic logarithm $z$?

For elliptic curves over $\mathbb{R}$, these questions have been answered satisfactorily, since algorithms for computing period lattices of elliptic curves over $\mathbb{R}$ and elliptic logarithms of real points are well-known and available in the literature (see, e.g., [Coh93, Algorithm 7.4.7 and 7.4.8] or [Cre97, §3.7]). The theory behind these algorithms, which heavily relies on the AGM of positive real numbers, is explained.
succinctly by Bost and Mestre [BM88]. The situation for elliptic curves over $\mathbb{C}$, however, is less satisfactory.

In this chapter, we therefore aim to develop a complete method for computing period lattices and elliptic logarithms for elliptic curves over $\mathbb{C}$. Our approach will closely follow that of [BM88] in the real case, and will also illustrate the connection between the following three classes of objects:

- AGM sequences over $\mathbb{C}$, which were first studied by Gauss and have been explored in depth by Cox [Cox84];
- Chains of lattices in $\mathbb{C}$;
- Chains of 2-isogenies between elliptic curves defined over $\mathbb{C}$.

This connection will allow us to derive an explicit formula for computing the period lattice of $E$, which yields the first algorithm of this chapter. We then continue further by generalising it to an algorithm for computing elliptic logarithms of points in $E(\mathbb{C})$. Finally, we illustrate the efficiency of both algorithms via some examples.

For computational purposes, we have implemented both algorithms in MAGMA (see Appendix A.1 for the source code); these have been also implemented independently in Sage (available from version 4.4) by Professor John E. Cremona.

4.2 AGM Sequences

In this section, we will give a brief overview of arithmetic-geometric mean of complex numbers. For more in-depth survey on this subject, see [Cox84].

**Definition.** Let $(a, b) \in \mathbb{C}^2$ be a pair of complex numbers satisfying

$$a \neq 0, \quad b \neq 0, \quad a \neq \pm b.$$  \hfill (4.2)
We say that \((a, b)\) is \textit{good} if \(\Re(a/b) \geq 0\), or equivalently

\[
|a - b| \leq |a + b|; \tag{4.3}
\]

otherwise the pair is said to be \textit{bad}.

Clearly, only one of the pair \((a, b)\), \((a, -b)\) is good, unless \(\Re(a/b) = \Re(b/a) = 0\) (or equivalently, \(|a - b| = |a + b|\)), in which case both pairs are good.

**Definition.** An \textit{AGM sequence} is a sequence \(((a_n, b_n))_{n=0}^\infty\) whose pairs satisfy the relation

\[
2a_{n+1} = a_n + b_n, \quad b_{n+1}^2 = a_nb_n \tag{4.4}
\]

for all \(n \geq 0\).

It is easy to see that if any one pair \((a_n, b_n)\) in the sequence satisfies (4.2) then all do, and we will make this restriction henceforth.

Given a starting pair \((a_0, b_0)\), one can obtain uncountably many AGM sequences by iterating the procedure of replacing \((a_n, b_n)\) by their \textit{arithmetic mean} \(a_{n+1} = (a_n + b_n)/2\) and their \textit{geometric mean} \(b_{n+1} = \sqrt{a_nb_n}\), with a choice of the square root for \(b_{n+1}\) at each step. However, we usually prefer to consider the entire sequence as a whole.

**Definition.** We say that an AGM sequence is \textit{good} if the pairs \((a_n, b_n)\) are good for all but finitely many \(n\). A good AGM sequence in which \((a_n, b_n)\) are good for all \(n > 0\) is said to be \textit{optimal}; and \textit{strongly optimal} if in addition \((a_0, b_0)\) is good.

If an AGM sequence is not good, then we say that it is \textit{bad}.

For an optimal AGM sequence \(((a_n, b_n))_{n=0}^\infty\) with a given starting pair \((a_0, b_0)\), at first it might seem that there could be many such sequences, since there could be several \(n \geq 0\) for which both pairs \((a_n, \pm b_n)\) are good. Fortunately, the following lemma shows that this is not the case.
Lemma 4.2.1. For every starting pair \((a_0, b_0)\), there is exactly one optimal AGM sequence \((a_n, b_n)_{n=0}^{\infty}\), unless \(a_0/b_0\) is real and negative, in which case there are two optimal AGM sequences with different signs of \(b_1\).

Proof. For \(n \geq 0\), let \(r_n = a_n/b_n\). Using (4.4), we can rewrite \(r_{n+1}\) as

\[
r_{n+1} = \pm \frac{1}{2} \left( \sqrt{r_n} + \frac{1}{\sqrt{r_n}} \right).
\]

One can then verify the following very easily:

- \(r_n\) is real and positive if and only if \(r_{n+1}\) is real;
- \(r_n\) is real and negative if and only if \(r_{n+1}\) is purely imaginary.

If both pairs \((a_{n+1}, \pm b_{n+1})\) are good, then (4.3) implies that \(r_{n+1}\) is purely imaginary, and so all preceding ratios \(r_n\) are real. Thus equality can hold in (4.3) at most once in any AGM sequence; and only for \(n = 0\) or \(n = 1\) in an optimal sequence (since \(\Re(r_n) \geq 0\) for all \(n \geq 1\) in an optimal sequence). In particular, this only holds for \(n = 1\) (i.e., both \((a_1, \pm b_1)\) are good) if and only if \(r_0\) is real and negative. \(\square\)

The following proposition is due to Cox; see [Cox84] for its proof. Note that Cox defines the notion of “good” more strictly than above (when \(\Re(a/b) = 0\) he requires \(\Im(a/b) > 0\), so that exactly one of \((a, \pm b)\) is good in every case), but in view of the preceding remarks this does not affect the following result.

Proposition 4.2.2. Given a pair \((a_0, b_0)\) \(\in \mathbb{C}^2\) satisfying (4.2), every AGM sequence \((a_n, b_n)_{n=0}^{\infty}\) starting at \((a_0, b_0)\) satisfies the following:

1. \(\lim_{n \to \infty} a_n\) and \(\lim_{n \to \infty} b_n\) exist and are equal;

2. The common limit, say \(M\), is non-zero if and only if the sequence is good;

3. \(|M|\) attains its maximum (among all AGM sequences starting at \((a_0, b_0)\)) if and only if the sequence is optimal.
4.3 Chains of Lattices

For an AGM sequence $((a_n, b_n))_{n=0}^\infty$ starting at $(a_0, b_0)$, we will denote the common limit $\lim_{n \to \infty} a_n = \lim_{n \to \infty} b_n$ by $M_S(a_0, b_0)$, where $S \subseteq \mathbb{Z}_{>0}$ is the set of all indices $n$ for which the pair $(a_n, b_n)$ is bad. For example, $M_\emptyset(a_0, b_0)$ denotes the common limit for the optimal AGM sequence. Note that the sequence is good if and only if $S$ is a finite set. To ease notation, we shall write $M_\emptyset(a_0, b_0)$ as $M(a_0, b_0)$.

4.3 Chains of Lattices

We now move on to consider the second class of objects, namely, *chains of lattices of index 2*. In this section, we will give the definition of a chain and describe its properties, which later will be seen to be analogous to those of an AGM sequence.

Throughout this chapter, a *lattice* will always be a free $\mathbb{Z}$-module of rank 2, embedded as a discrete subgroup of $\mathbb{C}$. Elements of lattices will often be called *periods*, since in our application lattices will arise as period lattices of elliptic curves defined over $\mathbb{C}$.

**Definition.** A *chain of lattices (of index 2)* is a sequence of lattices $(\Lambda_n)_{n=0}^\infty$ which satisfies the following conditions:

1. $\Lambda_n \supset \Lambda_{n+1}$ for all $n \geq 0$;
2. $[\Lambda_n : \Lambda_{n+1}] = 2$ for all $n \geq 0$;
3. $\Lambda_0/\Lambda_n$ is cyclic for all $n \geq 1$; equivalently, $\Lambda_{n+1} \neq 2\Lambda_{n-1}$ for all $n \geq 1$.

Thus for each $n \geq 1$, we have

$$\Lambda_{n+1} = \langle w \rangle + 2\Lambda_n$$  \hspace{1cm} (4.5)

for some $w \in \Lambda_n \setminus 2\Lambda_{n-1}$.

Given an initial lattice $\Lambda_0$, there are three choices for $\Lambda_1$. When $n \geq 1$, one of the three choices for $\Lambda_{n+1}$ is excluded since it is contained in $2\Lambda_{n-1}$ (which contradicts
the last condition in the definition of a chain), and so there are only two choices for \( \Lambda_{n+1} \). The number of such chains starting with \( \Lambda_0 \) is therefore uncountable; we will distinguish a countable subset of these as follows. Let

\[
\Lambda_\infty = \bigcap_{n=0}^{\infty} \Lambda_n.
\]

Then \( \Lambda_\infty \) is free of rank at most 1; the rank cannot be 2 since for all \( n \),

\[
[\Lambda_0 : \Lambda_\infty] \geq [\Lambda_0 : \Lambda_n] = 2^n,
\]

so \([\Lambda_0 : \Lambda_\infty]\) is infinite.

**Definition.** A chain of lattices \((\Lambda_n)_{n=0}^{\infty}\) is said to be *good* if \( \Lambda_\infty \) has rank 1; in this case a generator for \( \Lambda_\infty \) will be called a *limiting period* of the chain. If a chain is not good, then we say that it is *bad*.

We will first show that the limiting period is *primitive*, i.e., not in \( m\Lambda_0 \) for any \( m \geq 2 \).

**Lemma 4.3.1.** Let \((\Lambda_n)_{n=0}^{\infty}\) be a good chain with \( \Lambda_\infty = \langle w_\infty \rangle \). Then we have the following:

1. \( w_\infty \) is primitive; equivalently, \( \Lambda_0/\Lambda_\infty \) is free of rank 1;
2. \( \Lambda_n = \langle w_\infty \rangle + 2^n\Lambda_0 \) for all \( n \geq 0 \).

**Proof.** Suppose \( w_\infty = mw \) for some \( m \geq 1 \), and \( w \in \Lambda_n \subseteq \Lambda_0 \) for some \( n \geq 0 \). If \( m \) is odd, then \((m - 1)w \in \Lambda_{n+1} \). Since \( mw = w_\infty \in \Lambda_{n+1} \), we have \( w \in \Lambda_{n+1} \). Hence \( w \in \Lambda_\infty \) by induction. Thus \( w_\infty = mw = m(m'w_\infty) \) for some \( m' \in \mathbb{Z} \), and so \( m = 1 \).

Next, suppose that \( w_\infty = 2w \) for some \( w \in \Lambda_0 \). By definition of \( w_\infty \), we then have \( w \notin \Lambda_\infty \), and hence there exists \( n > 0 \) such that \( w \notin \Lambda_n \). This implies that
$w_\infty \in \Lambda_n \setminus 2\Lambda_{n-1}$ (recall that $\Lambda_n \supset 2\Lambda_{n-1}$). But since $w_\infty \in \Lambda_{n+1}$, we have

$$\Lambda_{n+1} = \langle w_\infty \rangle + 2\Lambda_n = \langle 2w \rangle + 2\Lambda_n \subseteq 2\Lambda_0,$$

which contradicts the definition of a chain. Thus $w_\infty$ is primitive, which proves (1).

Since $w_\infty$ is primitive, then $\Lambda_n/2^n\Lambda_0$ is cyclic of order $2^n$, and is generated by $w_\infty$ modulo $2^n\Lambda_0$. Hence (2) follows.

So far, our notion of a good chain has been defined as a property of the chain as a whole, and only used the abstract structure of lattices as free $\mathbb{Z}$-modules. Using the next definition, we will see that this property can be also seen in terms of individual steps $\Lambda_n \supset \Lambda_{n+1}$, when all lattices $\Lambda_n$ are embedded in $\mathbb{C}$. In view of (4.5), the choice of $\Lambda_{n+1}$ is determined by the class of $w$ modulo $2\Lambda_n$.

**Definition.** For $n \geq 1$, we say that $\Lambda_{n+1} \subset \Lambda_n$ is a right choice of sublattice of $\Lambda_n$ if $\Lambda_{n+1} = \langle w \rangle + 2\Lambda_n$, where $w$ is a minimal element in $\Lambda_n \setminus 2\Lambda_{n-1}$ (with respect to the usual complex absolute value).

In general, there will be only one right choice at each step; for more details on the exceptional case, see Section 4.3.1.

**Lemma 4.3.2.** Let $(\Lambda_n)_{n=0}^\infty$ be a good chain with $\Lambda_\infty = \langle w_\infty \rangle$. Then $w_\infty$ is minimal in $\Lambda_n$ for all but finitely many $n \geq 0$.

**Proof.** Let $w_1 = w_\infty$. By Lemma 4.3.1, $w_1$ is primitive and there exists $w_2 \in \Lambda_0$ such that $\Lambda_n = \langle w_1, 2^nw_2 \rangle$ for all $n \geq 0$. For a non-zero $w \in \Lambda_n$, we write $w = mw_1 + k2^nw_2$ with $m, k \in \mathbb{Z}$. If $k = 0$, then clearly $|w| = |m||w_1| \geq |w_1|$. On the other hand, if $|k| \geq 1$, then

$$|w/w_1| = |m + k2^nw_2/w_1| \geq 2^n|\Re(w_2/w_1)| \geq 1,$$

for all $n > -\log_2|\Re(w_2/w_1)|$. This proves the lemma. $\square$
Chapter 4. Period Lattices and Complex Elliptic Logarithms

The following proposition yields an alternative definition of a good chain. For now we remark that this is analogous to the definition of a good AGM sequence in Section 4.2; more of its analogues will be seen in later sections.

**Proposition 4.3.3.** A chain of lattices $(\Lambda_n)_{n=0}^\infty$ is good if and only if $\Lambda_{n+1} \subset \Lambda_n$ is a right choice for all but finitely many $n \geq 1$.

**Proof.** Let $(\Lambda_n)_{n=0}^\infty$ be a good chain with $\Lambda_\infty = \langle w_\infty \rangle$. Then by Lemma 4.3.2, there exists an integer $n_0$ such that $w_\infty$ is minimal in $\Lambda_n$ for all $n \geq n_0$. Since $\Lambda_{n+1} = \langle w_\infty \rangle + 2\Lambda_n$ for all $n$, then by definition, $\Lambda_{n+1} \subset \Lambda_n$ is a right choice for all $n \geq n_0$.

Conversely, suppose that $\Lambda_{n+1} \subset \Lambda_n$ is a right choice for all $n \geq n_0$ (where $n_0 \geq 1$). Without loss of generality, we may suppose that $n_0 = 1$. Let $w_1$ be a minimal element of $\Lambda_1$. Then $w_1$ is certainly primitive (as an element of $\Lambda_1$, though not necessarily in $\Lambda_0$). We claim that $w_1 \in \Lambda_n$ for all $n \geq 1$, so that the chain is good with limiting period $w_1$.

To prove the claim, suppose that $w_1 \in \Lambda_j$ for all $j \leq n$. Then $\Lambda_n = \langle w_1 \rangle + 2^{n-1}\Lambda_1$, since the latter is contained in the former and both have index $2^{n-1}$ in $\Lambda_1$. Hence $\Lambda_n = \langle w_1, 2^{n-1}w_2 \rangle$, where $w_2 \in \Lambda_1$ is such that $\Lambda_1 = \langle w_1, w_2 \rangle$. By minimality of $w_1$, the right sublattice $\Lambda_{n+1}$ of $\Lambda_n$ is clearly $\langle w_1 \rangle + \Lambda_n$ (note that $w_1$ is a candidate since $w_1 \in \Lambda_n \setminus 2\Lambda_{n-1}$); in particular, $w_1 \in \Lambda_{n+1}$ as required.

In the next subsection, we will introduce a special type of a lattice chain, whose properties will be analogous to those of an optimal AGM sequence. This type of lattice chain will play an important role in Section 4.5, where we develop an algorithm for computing period lattices of elliptic curves over $\mathbb{C}$.

### 4.3.1 Optimal Chains and Rectangular Lattices

**Definition.** A lattice chain $(\Lambda_n)_{n=0}^\infty$ is said to be *optimal* if $\Lambda_{n+1} \subset \Lambda_n$ is a right choice for all $n \geq 1$. 
In general, there will be only one optimal chain for each of the three choices of \( \Lambda_1 \subset \Lambda_0 \). In order to describe this statement more precisely, however, some preparation is necessary.

We say that a lattice \( \Lambda \subset \mathbb{C} \) is **rectangular** if it has an orthogonal \( \mathbb{Z} \)-basis \( \{w_1, w_2\} \), i.e., one which satisfies \( \Re(w_2/w_1) = 0 \). For example, the period lattice of an elliptic curve defined over \( \mathbb{R} \) with positive discriminant is rectangular, where an orthogonal basis is given by the least real period and the least imaginary period (see Section 1.3 for more details). In fact, rectangular lattices are homothetic to the period lattices of this family of elliptic curves.

If \( \{w_1, w_2\} \) is any \( \mathbb{Z} \)-basis for a lattice \( \Lambda \), the three non-trivial cosets of \( 2\Lambda \) in \( \Lambda \) are given by \( C_j = w_j + 2\Lambda \) for \( j = 1, 2, 3 \), where \( w_3 = w_1 + w_2 \). By a **minimal coset representative**, we mean a minimal element of one of these cosets. The next three lemmas explain some of its properties.

**Lemma 4.3.4.** Minimal coset representatives are primitive.

**Proof.** Let \( w \) be a minimal coset representative. Then \( w \not\in 2\Lambda \), since by definition \( w \) does not represent the trivial coset \( 2\Lambda \). Moreover, if \( w = mw' \) for some odd \( m \geq 3 \), then \( |w| = m|w'| > |w'| \). But since both \( w, w' \) belong to the same coset modulo \( 2\Lambda \), this contradicts the minimality of \( w \). Hence \( m = 1 \), i.e., \( w \) is primitive. \( \square \)

**Lemma 4.3.5.** In each coset \( C_j \), the minimal coset representative is unique up to sign, unless \( \Lambda \) is a rectangular lattice with orthogonal \( \mathbb{Z} \)-basis \( \{w_1, w_2\} \), in which case the coset \( C_3 \) has four minimal vectors \( \pm(w_1 \pm w_2) \).

**Proof.** For a rectangular lattice \( \Lambda \) with orthogonal \( \mathbb{Z} \)-basis \( \{w_1, w_2\} \), it is easy to see that the minimal coset representatives are as stated. Conversely, if a lattice \( \Lambda \) has a coset \( C \) with at least two pairs of minimal elements \( \pm w, \pm w' \), then \( w_1, w_2 = (w \pm w')/2 \in \Lambda \) are easily seen to be orthogonal.

Next, we will show that \( w_1, w_2 \), and \( w = w_1 + w_2 \) are non-trivial coset representatives modulo \( 2\Lambda \). If \( w_1 \equiv 0 \pmod{2\Lambda} \), then \( w_2 \equiv w \pmod{2\Lambda} \). But then
\[ |w_2| < |w_1 + w_2| = |w|, \] which contradicts the minimality of \( w \) in its coset. Hence \( w_1 \not\equiv 0 \pmod{2\Lambda} \). Similarly, \( w_2 \not\equiv 0 \pmod{2\Lambda} \). Moreover, \( w_1 \not\equiv w_2 \pmod{2\Lambda} \) since \( w = w_1 + w_2 \not\equiv 0 \pmod{2\Lambda} \). Therefore, \( w_1, w_2, w \) do represent the three non-trivial cosets modulo \( 2\Lambda \).

Finally, it remains to show that \( \{w_1, w_2\} \) is a \( \mathbb{Z} \)-basis for \( \Lambda \). Suppose the contrary that this is not the case. Then there would exist a non-zero period \( w_0 = \alpha w_1 + \beta w_2 \) with \( 0 \leq \alpha, \beta < 1 \). Clearly, one of \( w_0, w_0 - w_1, w_0 - w_2, w_0 - w \) is in the same coset as \( w \), but since all these periods are smaller than \( w \), this yields a contradiction.

As we will see later on, our algorithm for computing period lattices of elliptic curves will actually compute these minimal coset representatives. To ensure that we thereby obtain a \( \mathbb{Z} \)-basis for the lattice, the following lemma is required.

**Lemma 4.3.6.** For \( j = 1, 2, 3 \), let \( w_j \) be minimal coset representatives for a non-rectangular lattice \( \Lambda \). Then any two of these \( w_j \) form a \( \mathbb{Z} \)-basis for \( \Lambda \), and \( w_3 = \pm(w_1 \pm w_2) \).

**Proof.** We may assume that \( |w_1| \leq |w_2| \leq |w_3| \). Then \( w_1 \) is minimal in \( \Lambda \) and \( w_2 \) is minimal in \( \Lambda \setminus \langle w_1 \rangle \). Hence (by negating \( w_2 \) if necessary), \( \tau = w_2/w_1 \) is in the standard fundamental region for \( SL_2(\mathbb{Z}) \) acting on the upper half-plane, \( \{w_1, w_2\} \) is a \( \mathbb{Z} \)-basis for \( \Lambda \), and \( w_3 = w_1 \pm w_2 \); the sign depends on that of \( \Re(\tau) \).

The following proposition shows that the limiting period of an optimal chain is closely related to minimal coset representatives.

**Proposition 4.3.7.** A good chain of lattices \( (\Lambda_n)_{n=0}^{\infty} \) with \( \Lambda_\infty = \langle w_\infty \rangle \) is optimal if and only if \( w_\infty \) is a minimal coset representative of \( 2\Lambda_0 \) in \( \Lambda_0 \).

**Proof.** Suppose that \( w_\infty \) is a minimal coset representative. Then it is clear that \( \Lambda_{n+1} = \langle w_\infty \rangle + 2\Lambda_n \subset \Lambda_n \) is a right sublattice for all \( n \geq 1 \), since \( w_\infty \) is certainly minimal in \( \Lambda_n \setminus 2\Lambda_{n-1} \). Thus the chain \( (\Lambda_n)_{n=0}^{\infty} \) is optimal with limiting period \( w_\infty \).

Conversely, suppose that a chain \( (\Lambda_n)_{n=0}^{\infty} \) is optimal with limiting period \( w_\infty \). Let \( w \in \Lambda_1 \) be a minimal element of \( \Lambda_1 \setminus 2\Lambda_0 \), so that \( w \) is a minimal coset representative
for the unique non-trivial coset modulo $2\Lambda_0$ contained in $\Lambda_1$. Note that $w$ is unique up to sign, unless $\Lambda_0$ is rectangular, in which case (for one of the cosets) there will be two possibilities for $w$ up to sign. By optimality, the sublattice $\Lambda_2 \subset \Lambda_1$ is the right choice. In particular, if $\Lambda_0$ is not rectangular, then we must therefore have $\Lambda_2 = \langle w \rangle + 2\Lambda_1$. This, however, may not hold in the rectangular case, but it will hold if we replace $w$ by the other choice of minimal coset representative.

Now we claim that $\Lambda_n = \langle w \rangle + 2\Lambda_{n-1}$ for all $n \geq 2$. We already know this for $n = 2$. If the claim is true for $n$, then certainly $w$ is also minimal in $\Lambda_n \setminus 2\Lambda_{n-1}$, so the (unique) right choice of sublattice of $\Lambda_n$ is $\langle w \rangle + 2\Lambda_n$. By optimality, this is $\Lambda_{n+1}$, and so the claim holds for $n + 1$. Thus $w \in \bigcap_{n=0}^{\infty} \Lambda_n = \langle w_\infty \rangle$, and indeed, $w = \pm w_\infty$, since $w$ is primitive by Lemma 4.3.4.

This together with Lemma 4.3.5 gives the following conclusion.

**Corollary 4.3.8.** Every non-rectangular lattice $\Lambda$ has precisely three optimal lattice chains $(\Lambda_n)_{n=0}^{\infty}$ (with $\Lambda_0 = \Lambda$), whose limiting periods are the minimal coset representatives in each of the three non-zero cosets of $2\Lambda$ in $\Lambda$. Every rectangular lattice $\Lambda$ has precisely four optimal lattice chains associated to it.

### 4.4 Chains of 2-Isogenies

We finally consider the last class of objects, where we construct a chain of elliptic curves defined over $\mathbb{C}$ using $2$-isogenies. Since each elliptic curve uniquely has an associated period lattice, we will see that this chain will be analogous to a chain of lattices defined in Section 4.3. Most of the formulas we use in this section are due to Bost and Mestre [BM88].

Let $E_0$ be an elliptic curve over $\mathbb{C}$ given by a Weierstrass equation

$$E_0 : \quad Y_0^2 = 4(X_0 - e_1^{(0)})(X_0 - e_2^{(0)})(X_0 - e_3^{(0)}) \quad (4.6)$$
where all roots $e_j^{(0)}$ are distinct and $\sum_{j=1}^{3} e_j^{(0)} = 0$. Assume that the ordering of $e_j^{(0)}$ is fixed. Similar to [BM88], we define $a_n, b_n$ for $n \geq 0$ by

$$
a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}}, \quad b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}},
$$

$$
a_{n+1} = \frac{a_n + b_n}{2}, \quad b_{n+1} = a_n b_n \quad \text{for } n \geq 0.
$$

Note that $a_0, b_0$ are so far defined only up to sign, and also satisfy (4.2) since all $e_j^{(0)}$ are distinct. Given $a_n, b_n$, it is easy to see that we can compute $a_{n+1}$ unambiguously, whereas $b_{n+1}$ is obtained up to sign. Starting from a given pair $(a_0, b_0)$, this then determines an AGM sequence $((a_n, b_n))_{n=0}^{\infty}$, where we obtain a different sequence by choosing the sign of $b_{n+1}$ differently at each step. Associated to this AGM sequence, for $n \geq 1$ we let

$$
e_1^{(n)} = \frac{a_n^2 + b_n^2}{3}, \quad e_2^{(n)} = \frac{a_n^2 - 2b_n^2}{3}, \quad e_3^{(n)} = \frac{b_n^2 - 2a_n^2}{3}.
$$

(4.7)

Observe that this identity also holds for $n = 0$, and all $e_j^{(n)}$ are distinct and satisfy $\sum_{j=1}^{3} e_j^{(n)} = 0$ for all $n \geq 0$. Hence we can construct a sequence $(E_n)_{n=0}^{\infty}$ of elliptic curves over $\mathbb{C}$, where $E_n$ is given by the Weierstrass equation

$$
E_n : \quad Y_n^2 = 4(X_n - e_1^{(n)})(X_n - e_2^{(n)})(X_n - e_3^{(n)}).
$$

For $n \geq 1$, we define $\varphi_n : E_n \to E_{n-1}$ to be the morphism which sends $(X_n, Y_n) \mapsto (X_{n-1}, Y_{n-1})$, where

$$
X_{n-1} = X_n + \frac{(e_3^{(n)} - e_1^{(n)})(e_3^{(n)} - e_2^{(n)})}{X_n - e_3^{(n)}},
$$

$$
Y_{n-1} = Y_n \left(1 - \frac{(e_3^{(n)} - e_1^{(n)})(e_3^{(n)} - e_2^{(n)})}{(X_n - e_3^{(n)})^2}\right).
$$

(4.8)

Observe that $\ker(\varphi_n) = \{(e_3^{(n)}, 0)\}$. Thus $\varphi_n$ is a 2-isogeny (note that it is a 2-to-1 map). It is well known (see, e.g., [Sil86, Theorem III.6.1]) that there exists a dual
isogeny \( \hat{\varphi}_n : E_{n-1} \to E_n \) such that \( \hat{\varphi}_n \circ \varphi_n \) is the multiplication-by-2 map on \( E_n \). Note that \( \ker(\hat{\varphi}_n) = \langle (e_1^{(n-1)}, 0) \rangle \). However, \( \varphi_n \circ \varphi_{n+1} \) is not the multiplication-by-2 map on \( E_{n+1} \), since, for example,

\[
\varphi_n \left( \varphi_{n+1} \left( (e_1^{(n+1)}, 0) \right) \right) = \varphi_n((e_1^{(n)}, 0)) = (e_1^{(n-1)}, 0) \neq O.
\]

This therefore allows us to construct a chain of 2-isogenies, as depicted below.

\[
\cdots \xleftarrow{1} E_n \xrightarrow{3} \cdots \xleftarrow{1} E_{n-1} \xrightarrow{2} \cdots \xleftarrow{1} E_1 \xrightarrow{3} \cdots \xleftarrow{1} E_0
\]

The number \( j \) next to each arrow originating from \( E_n \) denotes the point \((e_j^{(n)}, 0)\).

To see the effect of choosing a different sign of \( b_n \), first note that we can rewrite \( e_j^{(n+1)} \) given by (4.7) as

\[
e_1^{(n+1)} = \frac{e_1^{(n)} + 2a_nb_n}{4}, \quad e_2^{(n+1)} = \frac{e_1^{(n)} - 2a_nb_n}{4}, \quad e_3^{(n+1)} = \frac{-e_1^{(n)}}{2}.
\]  (4.9)

For \( n \geq 1 \), if we replace \((a_n, b_n)\) by \((a_n, -b_n)\), then this interchanges \( e_1^{(n+1)} \) and \( e_2^{(n+1)} \) but leaves \( e_3^{(n+1)} \) unchanged. This relabelling of the roots \( e_j^{(n+1)} \) therefore has no effect on the curve \( E_{n+1} \), but in turn yields a different curve \( E_{n+2} \). For \( n = 0 \), recall that \( a_0, b_0 \) are determined up to sign. It is easy to see that if only one of \( a_0, b_0 \) changes the sign, then this interchanges \( e_1^{(1)} \) and \( e_2^{(1)} \) but fixes \( e_3^{(1)} \); all \( e_j^{(1)} \) remain unchanged if both signs of \( a_0, b_0 \) are changed.

Given a pair \((a_0, b_0) \in \mathbb{C}^2\) satisfying (4.2), knowing \((a_0, b_0)\) not only tells us which curve \( E_0 \) we started with and which 2-torsion point \((e_1^{(0)}, 0)\) we used to construct \( E_1 \), it also determines the labelling of all the roots of \( E_1 \) (see (4.9)), and hence determines the curve \( E_2 \). Thus we have a bijection between:

- The set of all AGM sequences starting at \((a_0, b_0)\), and
• The set of all chains of 2-isogenies starting at $E_0$ with the subsequence

$$E_2 \rightarrow E_1 \rightarrow E_0.$$ 

We now consider what happens to a chain of 2-isogenies $(E_n)_{n=0}^\infty$ as $n \rightarrow \infty$. Given an elliptic curve $E_0 = E$ over $\mathbb{C}$, we can construct all elliptic curves $E_n$ recursively as above. This construction in turn yields an AGM sequence $((a_n, b_n))_{n=0}^\infty$ associated to the isogeny chain. Let $S \subseteq \mathbb{Z}_{>0}$ be the set of all indices for which the pair $(a_n, b_n)$ is bad. It then follows from (4.7) that

$$\lim_{n \rightarrow \infty} e_1^{(n)} = \frac{2M_S(a_0, b_0)^2}{3}, \quad \lim_{n \rightarrow \infty} e_2^{(n)} = \lim_{n \rightarrow \infty} e_3^{(n)} = \frac{-M_S(a_0, b_0)^2}{3},$$

where $M_S(a_0, b_0)$ is the common limit of the AGM sequence. The limiting curve $E_\infty$ for the isogeny chain is thus given by

$$E_\infty : \quad y^2 = 4 \left( x_\infty - \frac{2M_S(a_0, b_0)^2}{3} \right) \left( x_\infty + \frac{M_S(a_0, b_0)^2}{3} \right)^2.$$  

(4.11)

Observe that $E_\infty$ is a singular curve. We say that the isogeny chain is good if the singular point of $E_\infty$ is a node; otherwise it is said to be bad.

In Section 4.3, we have seen that the notion of a good chain of lattices, which has been defined as a property of the chain as a whole, can be also considered in terms of individual steps. We will finally show that this is also the case for our notion of a good isogeny chain defined above.

**Definition.** Let $(E_n)_{n=0}^\infty$ be a chain of 2-isogenies. For $n \geq 2$, we say that $E_n$ is the right choice for the isogeny chain if its roots satisfy

$$|e_3^{(n)} - e_2^{(n)}| \leq |e_3^{(n)} - e_1^{(n)}|;$$

otherwise we say that $E_n$ is the bad choice.
4.5 Period Lattices of Elliptic Curves

In other words, $E_n$ is the right choice if $e_{3}^{(n)}$ is closer to $e_{2}^{(n)}$ than it is to $e_{1}^{(n)}$. The following lemma is immediate.

**Lemma 4.4.1.** Let $(E_n)_{n=0}^{\infty}$ be an isogeny chain, and let $((a_n, b_n))_{n=0}^{\infty}$ be its associated AGM sequence. Then for all $n \geq 1$, the pair $(a_n, b_n)$ is good if and only if $E_{n+1}$ is the right choice for the isogeny chain.

**Proof.** From (4.7), we have

$$4|e_{3}^{(n+1)} - e_{2}^{(n+1)}| = 4|b_{n+1}^2 - a_{n+1}^2| = |b_n - a_n|^2,$$

and

$$4|e_{3}^{(n+1)} - e_{1}^{(n+1)}| = 4|a_{n+1}|^2 = |b_n + a_n|^2.$$

The lemma now follows directly from the definition of a right choice given above. □

The following proposition, which is analogous to Proposition 4.3.3, follows easily from the properties of complex AGM.

**Proposition 4.4.2.** The following are equivalent:

1. A chain of 2-isogenies $(E_n)_{n=0}^{\infty}$ is good;

2. Its associated AGM sequence is good;

3. $E_n$ is chosen to be the right choice for all but finitely many $n \geq 2$.

**Proof.** Equivalence of (1) and (2) follows easily from Proposition 4.2.2, while equivalence of (2) and (3) is immediate from Lemma 4.4.1 and the definition of a good AGM sequence. □

4.5 Period Lattices of Elliptic Curves

In this section, we will combine all three classes of objects we have introduced so far into an algorithm for computing period lattices of elliptic curves over $\mathbb{C}$. 
Our approach will follow closely with the classical definition of periods and elliptic logarithms given in Section 1.3. For an alternative approach, see [CT, Section 6].

### 4.5.1 General Case

Let $E_0$ be an elliptic curve over $\mathbb{C}$ of the form (4.6) as before, where all roots $e_j^{(0)}$ are given in some fixed order. As mentioned in Section 1.3, it is well known that $E_0 \cong \mathbb{C}/\Lambda_0$ for some lattice $\Lambda_0$ via the map

$$P = (\wp_{\Lambda_0}(z), \wp'_{\Lambda_0}(z)) \mapsto z \pmod{\Lambda_0}$$

$$O \mapsto 0 \pmod{\Lambda_0}.$$

Let $w_1 \in \Lambda_0$ and let $z_1 = w_1/2$. Then by above isomorphism, we can see that $(\wp_{\Lambda_0}(z_1), \wp'_{\Lambda_0}(z_1))$ is a 2-torsion point in $E_0(\mathbb{C})$. Hence we can assume that

$$\wp_{\Lambda_0}(z_1) = e_1^{(0)}, \quad \wp'_{\Lambda_0}(z_1) = 0$$

(note that there are three ways to choose $e_1^{(0)}$). Define $\ell_1$ to be the straight line on the complex plane starting from 0 to $z_1$. Then we have

$$\frac{w_1}{2} = z_1 = \int_{\ell_1} dz = \int_{C_1^{(0)}} \frac{dX_0}{Y_0},$$

where $C_1^{(0)}$ is the path on the elliptic curve $E_0$ defined by

$$C_1^{(0)} = \{(\wp_{\Lambda_0}(t z_1), \wp'_{\Lambda_0}(t z_1)) : 0 \leq t \leq 1\}.$$

Given $E_0$, we can construct a chain of 2-isogenies $(E_n)_{n=0}^{\infty}$ and its associated AGM sequence $((a_n, b_n))_{n=0}^{\infty}$ using the method described in Section 4.4. As we will see below, this will also yield the corresponding chain of lattices $(\Lambda_n)_{n=0}^{\infty}$. Note that one can obtain a different isogeny chain with the same starting curve $E_0$ depending...
on how the sign of $b_n$ is chosen at each step. We will now combine what we have so far in order to determine $z_1$ (and hence $w_1$).

For each $n \geq 1$, since $E_n \cong \mathbb{C}/\Lambda_n$ for some lattice $\Lambda_n$, we have the connection between an isogeny chain and a chain of lattices as shown in Figure 4.1.

\[
\cdots \to \mathbb{C} \xrightarrow{id} \mathbb{C} \xrightarrow{\text{id}} \cdots \\
\cdots \to \mathbb{C}/\Lambda_n \xrightarrow{\phi_n} \mathbb{C}/\Lambda_{n-1} \xrightarrow{\phi_{n-1}} \cdots \\
\cdots \to E_n \xrightarrow{\phi_n} E_{n-1} \xrightarrow{\phi_{n-1}} \cdots
\]

Figure 4.1: A chain of 2-isogenies linked with a chain of lattices

By definition of $\phi_n$ (see (4.8)), it can be verified that

\[
\phi_n^* \left( \frac{dX_{n-1}}{Y_{n-1}} \right) = \frac{dX_n}{Y_n}
\] (4.12)

for all $n \geq 1$, where $\phi_n^*$ is the pullback of the differential on $E_{n-1}$ by $\phi_n$. This therefore induces the identity map $\text{id} : \mathbb{C} \to \mathbb{C}$, which in turn induces the map $\mathbb{C}/\Lambda_n \to \mathbb{C}/\Lambda_{n-1}$ via $z \pmod{\Lambda_n} \mapsto z \pmod{\Lambda_{n-1}}$. Since $\phi_n$ is a 2-to-1 map, it then follows that $\Lambda_{n-1} \supset \Lambda_n$ with $[\Lambda_{n-1} : \Lambda_n] = 2$. Moreover, since $\phi_n \circ \phi_{n+1}$ is not the multiplication-by-2 map on $E_{n+1}$ for all $n \geq 1$, we have $\Lambda_{n+1} \neq 2\Lambda_{n-1}$. Hence this gives us the relationship between a chain of lattices and a chain of 2-isogenies.

Next, let $C_1^{(n)}$ be the path on the elliptic curve $E_n$ defined by

\[
C_1^{(n)} = \{ (\phi_n(tz_1), \phi_n'(tz_1)) : 0 \leq t \leq 1 \}.
\]

Then we can regard $C_1^{(n)}$ as a map $[0, 1] \to E_n$, as shown in the diagram below.

\[
\begin{array}{ccc}
[0, 1] & \xrightarrow{C_1^{(n+1)}} & E_{n+1} \\
\downarrow \text{id} & & \downarrow \phi_{n+1} \\
[0, 1] & \xrightarrow{C_1^{(n)}} & E_n
\end{array}
\]
By commutativity of this diagram, we have \( \varphi_{n+1} \circ C^{(n+1)}_1 = C^{(n)}_1 \). Together with (4.12), this implies that

\[
\int_{C^{(n+1)}_1} \frac{dX_{n+1}}{Y_{n+1}} = \int_{C^{(n)}_1} \frac{dX_n}{Y_n},
\]

and so

\[
\frac{w_1}{2} = z_1 = \int_{\mathcal{E}_1} dz = \int_{C^{(0)}_1} \frac{dX_0}{Y_0} = \int_{C^{(1)}_1} \frac{dX_1}{Y_1} = \cdots = \int_{C^{(n)}_1} \frac{dX_n}{Y_n} = \cdots
\]

Recall the limiting curve \( E_\infty \) of the isogeny chain \( (E_n)_{n=0}^\infty \) from (4.11). As \( n \to \infty \), the path \( C^{(n)}_1 \) on \( E_n \) approaches to some path on \( E_\infty \), say, \( C^{(\infty)}_1 \), whose starting point is \( O \). We now describe another end-point of \( C^{(\infty)}_1 \) as follows. For all \( n \geq 1 \), one can easily check from (4.8) that

\[
\varphi_n(O) = \varphi_n \left( (e^{(n)}_3, 0) \right) = O
\]

\[
\varphi_n \left( (e^{(n)}_1, 0) \right) = \varphi_n \left( (e^{(n)}_2, 0) \right) = (e^{(n-1)}_1, 0).
\]

Moreover, we can rewrite (4.8) as

\[
X_n = \frac{(X_{n-1} + e^{(n)}_3) + s_n}{2}, \quad Y_n = \frac{Y_{n-1}}{1 - \frac{(e^{(n)}_3 - e^{(n)}_1)(e^{(n)}_3 - e^{(n)}_2)}{(X_n - e^{(n)}_3)^2}}
\]

(4.13)

where

\[
s_n = \sqrt{(X_{n-1} - e^{(n)}_3)^2 - 4(e^{(n)}_3 - e^{(n)}_1)(e^{(n)}_3 - e^{(n)}_2)}
\]

is defined up to sign. By definition of \( C^{(n)}_1 \), it is clear that its starting point is always \( O \in E_n(\mathbb{C}) \). In addition, we already know that \( \varphi_n \circ C^{(n)}_1 = C^{(n-1)}_1 \). Hence for all \( n \geq 1 \), the sign of \( s_n \) must be chosen so that \( O \in E_{n-1} \mapsto O \in E_n \). This can be
achieved provided that $s_n$ satisfies

$$|(X_{n-1} - e_3^{(n)}) - s_n| \leq |(X_{n-1} - e_3^{(n)}) + s_n|. \quad (4.14)$$

By continuity, this criterion for $s_n$ then holds along the path $c_1^{(n-1)}$. But since another end-point of $c_1^{(0)}$ is $(e_1^{(0)}, 0)$ by definition, (4.14) eventually implies that $(e_1^{(n)}, 0) \mapsto (e_1^{(1)}, 0)$. Hence by induction, the two end-points of $c_1^{(n)}$ are $O$ and $(e_1^{(n)}, 0)$ for all $n \geq 0$, and so the two end-points of $c_1^{(\infty)}$ are

$$P_1 = O, \quad P_2 = \left( \lim_{n \to \infty} e_1^{(n)}, 0 \right) = \left( \frac{2MS(a_0, b_0)^2}{3}, 0 \right)$$

(the last equality is from (4.10)), where $((a_n, b_n))_{n=0}^{\infty}$ is the AGM sequence associated to the isogeny chain, and $S$ is the set of all indices for which $(a_n, b_n)$ is bad.

Similar to [BM88], by writing

$$X_\infty = t^2 + \frac{2MS(a_0, b_0)^2}{3}, \quad Y_\infty = 2t(t^2 + MS(a_0, b_0)^2),$$

and letting $\tan \theta = t/MS(a_0, b_0)$, we have

$$dX_\infty \over Y_\infty = \frac{d\theta}{MS(a_0, b_0)}$$

and also

$$t = \frac{Y_\infty}{2\left(X_\infty + \frac{MS(a_0, b_0)^2}{3}\right)}.$$

Then it is easy to see that

$$P = P_1 \iff t = \infty \iff \cos \theta = 0 \iff \theta = \frac{(2k_1 - 1)\pi}{2}$$

$$P = P_2 \iff t = 0 \iff \sin \theta = 0 \iff \theta = k_2\pi$$

for some $k_1, k_2 \in \mathbb{Z}$. If we choose $k_1 = k_2 = k$ for some $k \in \mathbb{Z}$ (i.e., we choose the
values for arctan from the same branch), we finally have

\[
w_1 = \cdots = 2 \int \frac{dX_\infty}{Y_\infty} = \frac{2}{M_S(a_0, b_0)} \int_{k\pi - \frac{\pi}{2}}^{k\pi} d\theta = \frac{\pi}{M_S(a_0, b_0)}.
\]

Note that \( w_1 \) we just obtained is up to sign. If we had chosen \( k_1, k_2 \) differently then we would have obtained some odd multiple of \( w_1 \), which would not change \( w_1 \) modulo \( 2\Lambda_0 \).

Given \((a_0, b_0)\), we already know that the value of \( M_S(a_0, b_0) \) depends on the set \( S \). If we choose \( S = \emptyset \), then \( |M_S(a_0, b_0)| = |M(a_0, b_0)| \) will attain its maximum among all AGM sequences starting at \((a_0, b_0)\) by Proposition 4.2.2. Thus the period \( w_1 \in \Lambda_0 \) obtained by making the optimal choice for the AGM sequence with a fixed starting pair \((a_0, b_0)\) will be the minimal one (hence primitive), and may also be a minimal coset representative modulo \( 2\Lambda_0 \). If we can determine the other two minimal coset representatives \( w_2, w_3 \) (by choosing \( e_1^{(0)} \) differently and computing \( w_j \) in a similar way), then by Lemma 4.3.6, any two of these \( w_j \) form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \).

Recall that

\[
a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}}, \quad b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}},
\]

i.e., both \( a_0, b_0 \) are determined up to sign. Then we have the problem of deciding which one of \( \pi/M(a_0, \pm b_0) \) is actually a minimal coset representative of \( 2\Lambda_0 \) in \( \Lambda_0 \), since both are periods of \( \Lambda_0 \) and belong to the same coset modulo \( 2\Lambda_0 \). To avoid this ambiguity, we will regard the pairs \((a_0, \pm b_0)\) as the two results of computing AGM “one step backwards”. To be precise, we wish to find \( a_{-1}, b_{-1} \) such that

\[
2a_0 = a_{-1} + b_{-1}, \quad b_0^2 = a_{-1}b_{-1}.
\]

It is then easy to show that

\[
a_{-1} = a_0 \pm c_0, \quad b_{-1} = a_0 \mp c_0,
\]
where \( c_0^2 = a_0^2 - b_0^2 \), so both pairs \((a_0, \pm b_0)\) come from \((a_0 + c_0, a_0 - c_0)\).

Let \( w = \pi / M(a_0 + c_0, a_0 - c_0) \). One can easily observe that changing the sign of \( c_0 \) has no effect on \( w \), whereas changing the sign of \( a_0 \) simply negates \( w \). Thus \( w \) is uniquely determined up to sign regardless of the signs of \( a_0, c_0 \). Moreover, the optimality of \( M(a_0 + c_0, a_0 - c_0) \) implies that

\[
\pi = w = \frac{\pi}{M(a_0 + c_0, a_0 - c_0)} = \frac{\pi}{M(a_0, b_0)},
\]

provided that \((a_0, b_0)\) is good, i.e., \(|a_0 - b_0| \leq |a_0 + b_0|\). Hence if \((a_0, b_0)\) is chosen to be good, then \( w_1 = \pi / M(a_0, b_0) = w \) is smaller than \( \pi / M(a_0, -b_0) \), and is thus a minimal coset representative modulo \( 2\Lambda_0 \). Finally, it follows from Proposition 4.3.7 that the corresponding lattice chain \((\Lambda_n)_{n=0}^{\infty}\) is optimal with limiting period \( w_1 \), and there exists \( w_2 \in \Lambda_0 \) such that \( \Lambda_n = \langle w_1, 2^n w_2 \rangle \) for all \( n \geq 0 \).

Thus we have proved our first result on period lattices of elliptic curves over \( \mathbb{C} \).

**Theorem 4.5.1** (Period Lattices of Elliptic Curves over \( \mathbb{C} \), first version). Let \( E_0 \) be an elliptic curve over \( \mathbb{C} \) of the form (4.6), with period lattice \( \Lambda_0 \). Set

\[
a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}}, \quad b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}},
\]

where the sign of \( b_0 \) is chosen so that \((a_0, b_0)\) is good (i.e., \(|a_0 - b_0| \leq |a_0 + b_0|\)). Then

\[
w_1 = \frac{\pi}{M(a_0, b_0)}
\]

is a primitive period of \( \Lambda_0 \), and is a minimal coset representative modulo \( 2\Lambda_0 \).

Define the other two minimal coset representatives \( w_2, w_3 \) in a similar way (by permuting \( e_j^{(0)} \)). Then any two of these \( w_j \) form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \).

As we can see, this theorem computes each minimal coset representative \( w_j \) by choosing \( e_1^{(0)} \) differently at a time. Nevertheless, it turns out that we may obtain all \( w_j \) by using only one fixed ordering of the roots \( e_j^{(0)} \) and three different AGM
computations. This alternative method, which also exhibits a certain relationship among all $w_j$, will be explained in Section 4.5.4.

### 4.5.2 Special Case I: Rectangular Lattices

For the rest of this chapter, we set $i = \sqrt{-1}$. Recall that if $|a_0 - b_0| = |a_0 + b_0|$, then both $(a_0, \pm b_0)$ are good and $\Re(b_0/a_0) = 0$. This implies that $b_0 = ika_0$ for some $k \in \mathbb{R} \setminus \{0\}$. Let $r = k^2$. Then we have

$$b_0 = ika_0 \iff \frac{e_1^{(0)} - e_2^{(0)}}{e_1^{(0)} - e_3^{(0)}} = -r < 0.$$

Using the fact that $\sum_{j=1}^{3} e_j^{(0)} = 0$, we can rewrite $e_3^{(0)}$ in terms of $e_1^{(0)}, e_2^{(0)}$ and obtain

$$e_2^{(0)} = \left(\frac{1 + 2r}{1 - r}\right) e_1^{(0)}, \quad e_3^{(0)} = \left(\frac{r + 2}{r - 1}\right) e_1^{(0)},$$

provided that $r \neq 1$. Clearly, the sign of $(1 + 2r)/(1 - r)$ is always opposite to the sign of $(r + 2)/(r - 1)$ for all $r > 0$ (apart from 1). Geometrically, this means that all $e_j^{(0)}$ are collinear on the complex plane with $e_1^{(0)}$ in the middle. If $r = 1$, then one can easily check that this is still the case, with $e_1^{(0)} = 0$ and $e_2^{(0)} = -e_3^{(0)}$.

To see what the associated period lattice looks like, first we let $w = \pi/M(a_0, b_0)$ and $w' = \pi/M(a_0, -b_0)$. Then both $w, w'$ (up to sign) are the minimal elements in the same coset modulo $2\Lambda_0$. Hence by Lemma 4.3.5, the periods $w_1, w_2 = (w \pm w')/2$ form an orthogonal $\mathbb{Z}$-basis for $\Lambda_0$, so the period lattice is rectangular. Alternatively, we could obtain a $\mathbb{Z}$-basis for $\Lambda_0$ by computing two minimal coset representatives (see Theorem 4.5.1) using the two other roots of $E_0$ which are not “in the middle” in the role of $e_1^{(0)}$.

Finally, we note that if all $e_j^{(0)}$ are collinear, then we can always “rotate” them by a suitable constant in $\mathbb{C} \setminus \mathbb{R}$ so that the scaled roots $e'_j$ are all real. Then one could use an algorithm for computing period lattices of elliptic curves over $\mathbb{R}$ (e.g., [Coh93, Algorithm 7.4.7]) to compute the period lattice of the elliptic curve...
E': $(Y')^2 = 4(X' - e_1')(X' - e_2')(X' - e_3')$. The period lattice of our original elliptic curve $E_0$ is then obtained after some suitable scaling.

In particular, one can arrange all roots $e_j'$ so that $e_1' > e_2' > e_3'$ and obtain an orthogonal $\mathbb{Z}$-basis for the period lattice of $E'$ by setting

$$w_1 = \frac{\pi}{M\left(\sqrt{e_1' - e_3'}, \sqrt{e_1' - e_2'}\right)}, \quad w_2 = \frac{i\pi}{M\left(\sqrt{e_1' - e_3'}, \sqrt{e_2' - e_3'}\right)}$$

with all positive square roots. In fact, these formulas are familiar from the literature; see, e.g., [Coh93, Algorithm 7.4.7] or [Cre97, (3.7.1)].

4.5.3 Special Case II

If the roots of $E_0$ are such that

$$\left|\frac{e_1^{(0)} - e_2^{(0)}}{e_1^{(0)} - e_3^{(0)}}\right| = 1 \quad \text{with} \quad e_1^{(0)} - e_2^{(0)} \neq \pm (e_1^{(0)} - e_3^{(0)}),$$

then geometrically all $e_j^{(0)}$ lie on an isosceles triangle having $e_1^{(0)}$ as the vertex where the sides of equal length intersect. As before, one can rotate this triangle by a suitable constant in $\mathbb{C} \setminus \mathbb{R}$ so that $e_1^{(0)} \in \mathbb{R}$, and $e_2^{(0)}, e_3^{(0)}$ are complex conjugates. This yields a new elliptic curve $E'$ over $\mathbb{R}$, whose Weierstrass equation has only one real root. In other words, $E'/\mathbb{R}$ has negative discriminant.

Again, one can use an algorithm for computing period lattices of elliptic curves over $\mathbb{R}$ (e.g., [Coh93, Algorithm 7.4.7]) to compute the period lattice of $E'$. It is well known that the period lattice of $E'$ is of the form $\Lambda' = \langle w_1', w_2' \rangle$, for some $w_1', w_2'$ satisfying

$$w_1' \in \mathbb{R}, \quad \Re(w_2') = \frac{w_1'}{2}.$$

The period lattice $\Lambda_0 = \langle w_1, w_2 \rangle$ of $E_0$ can then be obtained by a suitable scaling on $w_1', w_2'$. Note that this time we have $\Re(w_2/w_1) = 1/2$. This will be illustrated in Example 4.7.4.
### 4.5.4 A Relationship amongst the Periods

In this subsection, we will present an alternative method for computing period lattices of elliptic curves over \( \mathbb{C} \). Unlike Theorem 4.5.1, this time all three minimal coset representatives \( w_1, w_2, w_3 \) will be determined by using only one fixed ordering of the roots and three different strongly optimal AGM sequences. Finally, we will also show that all \( w_j \) obtained by this new method satisfy a certain linear relation.

Let \( E_0 \) be an elliptic curve over \( \mathbb{C} \) of the form (4.6) as before. Assume that its roots are arranged in some fixed order, say, \( (e^{(0)}_1, e^{(0)}_2, e^{(0)}_3) \). Then we can compute \( a_0, b_0 \) (uniquely up to sign) as before. Let \( c_0 = \sqrt{a_0^2 - b_0^2} \), which is again up to sign.

We claim that one can always choose the signs of \( a_0, b_0, c_0 \) in such a way that the following conditions hold:

\[
|a_0 - b_0| \leq |a_0 + b_0|, \quad |c_0 - ib_0| \leq |c_0 + ib_0|, \quad |a_0 - c_0| \leq |a_0 + c_0|.
\]  

(4.15)

To prove this claim, we first consider the case when equality occurs in one of the conditions in (4.15). It might seem possible at first that there could be at least two equalities occurring in (4.15) simultaneously. However, it is very easy to verify that this is not the case. If there exists exactly one equality in (4.15), then one can always choose the sign of the variables appearing in the equality in such a way that all the conditions in (4.15) are satisfied. For example, if

\[
a_0 = 1, \quad b_0 = i, \quad c_0 = \sqrt{2}
\]

(note that \( c_0^2 = a_0^2 - b_0^2 \)), then we have

\[
|a_0 - b_0| = |a_0 + b_0|, \quad |c_0 - ib_0| > |c_0 + ib_0|, \quad |a_0 - c_0| < |a_0 + c_0|.
\]

However, if \( b_0 = -i \), then \( a_0, b_0, c_0 \) now satisfy all the conditions in (4.15).

Now we consider the case when all the conditions in (4.15) are strictly inequal-
4.5. Period Lattices of Elliptic Curves

In this case, we start by choosing the sign of $a_0$ arbitrarily. Then we can always choose the signs of $b_0, c_0$ so that the first and the third conditions in (4.15) hold. The second condition, however, requires some extra work. First, we note that this condition is equivalent to $\Im(c_0/b_0) \geq 0$. Hence, if our chosen $b_0, c_0$ are such that $\Im(c_0/b_0) \leq 0$, then at first one might consider interchanging $b_0$ and $c_0$; this would make the second condition satisfied, whereas other conditions remained unaffected.

Unfortunately, such attempt will affect our curve $E_0$. To be precise, suppose we interchange $b_0$ and $c_0$. Then by (4.7), we have

$$
equiv_1(0) = \frac{a_0^2 + b_0^2}{3} \quad \iff \quad \frac{a_0^2 + c_0^2}{3} = -\equiv_3(0)$$
$$
equiv_2(0) = \frac{a_0^2 - 2b_0^2}{3} \quad \iff \quad \frac{a_0^2 - 2c_0^2}{3} = -\equiv_2(0)$$
$$
equiv_3(0) = \frac{b_0^2 - 2a_0^2}{3} \quad \iff \quad \frac{c_0^2 - 2a_0^2}{3} = -\equiv_1(0),$$

i.e., $E_0$ is mapped to another elliptic curve isomorphic to it. In this case, we should therefore use a new ordering for the roots of $E_0$. Let

$$a' = i a_0, \quad b' = i c_0, \quad c' = i b_0.$$  

Then one can easily check that $(a')^2 - (b')^2 = (c')^2$. Moreover, we have

$$|a' \pm b'| = |a_0 \pm c_0|, \quad |c' \pm ib'| = |ib_0 \mp c_0|, \quad |a' \pm c'| = |a_0 \mp b_0|.$$

Since $a_0, b_0, c_0$ satisfy all but the second conditions in (4.15), this leads to

$$|a' - b'| < |a' + b'|, \quad |c' - ib'| < |c' + ib'|, \quad |a' - c'| < |a' + c'|,$$

and so $a', b', c'$ satisfy all conditions in (4.15). Defining $\equiv_1', \equiv_2', \equiv_3'$ in a similar way as
in (4.7), we finally obtain

\[ e'_1 = \frac{(a')^2 + (b')^2}{3} = \frac{b_0^2 - 2a_0^2}{3} = e_3^{(0)} \]
\[ e'_2 = \frac{(a')^2 - 2(b')^2}{3} = \frac{a_0^2 - 2b_0^2}{3} = e_2^{(0)} \]
\[ e'_3 = \frac{(b')^2 - 2(a')^2}{3} = \frac{a_0^2 + b_0^2}{3} = e_1^{(0)} \]

This can be summarised into the following proposition.

**Proposition 4.5.2.** Let \((e_1^{(0)}, e_2^{(0)}, e_3^{(0)})\) be an ordering of the roots of \(E_0\), and define \(a_0, b_0, c_0\) as before. Then we have one of the following cases:

1. \(a_0, b_0, c_0\) readily satisfy all the conditions in (4.15);

2. \(a_0, b_0, c_0\) yield an equality in (4.15). In this case, we can choose the sign of the variables appearing in the equality so that all the conditions in (4.15) are satisfied;

3. Otherwise, suppose the signs of \(a_0, b_0, c_0\) are chosen so that

\[ |a_0 - b_0| < |a_0 + b_0| \text{ and } |a_0 - c_0| < |a_0 + c_0|. \]

Then if \(|c_0 - ib_0| > |c_0 + ib_0|\), the new ordering \((e_3^{(0)}, e_2^{(0)}, e_1^{(0)})\) will give a new set \(\{a_0, b_0, c_0\}\) (where \(a_0, b_0, c_0\) is replaced by \(ia_0, ic_0, ib_0\) respectively), whose signs can be chosen so that all conditions in (4.15) hold.

We are now ready to state an alternative version of Theorem 4.5.1.

**Theorem 4.5.3** (Period Lattices of Elliptic Curves over \(\mathbb{C}\), second version). Let \(E_0\) be an elliptic curve over \(\mathbb{C}\) of the form (4.6), with period lattice \(\Lambda_0\) as before. Order the roots \((e_1^{(0)}, e_2^{(0)}, e_3^{(0)})\) of \(E_0\) so that the signs of

\[ a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}}, \quad b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}}, \quad c_0 = \sqrt{e_2^{(0)} - e_3^{(0)}} \]
can be chosen to satisfy all the conditions in (4.15). Define

\[ w_1 = \frac{\pi}{M(a_0, b_0)}, \quad w_2 = \frac{\pi}{M(c_0, ib_0)}, \quad w_3 = \frac{i\pi}{M(a_0, c_0)}. \]

Then all \( w_j \) are primitive periods of \( \Lambda_0 \), and are minimal coset representatives modulo \( 2\Lambda_0 \); any two of these form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \).

Proof. By Proposition 4.5.2, it is always possible to order \((e_1^{(0)}, e_2^{(0)}, e_3^{(0)})\) so that \((a_0, b_0, c_0)\) satisfies all the conditions in (4.15). We will show that this new definition of \( w_j \) still agrees with Theorem 4.5.1 where each root \( e_j^{(0)} \) plays the role of \( e_1^{(0)} \).

To show this, first note that \( w_1 = \pm \pi/M(a_0, b_0) \) as before, since \((a_0, b_0)\) is good.

Letting \((e_1', e_2', e_3') = (e_2^{(0)}, e_1^{(0)}, e_3^{(0)})\), we find that \((a', b') = (c_0, ib_0)\) is good, so

\[ \frac{\pi}{M(a', b')} = \pm \frac{\pi}{M(c_0, ib_0)} = w_2. \]

Similarly, by letting \((e_1', e_2', e_3') = (e_3^{(0)}, e_1^{(0)}, e_2^{(0)})\), we find that \((a', b') = (ic_0, ia_0)\) is good, and so

\[ \frac{\pi}{M(a', b')} = \pm \frac{i\pi}{M(a_0, c_0)} = w_3. \]

Note that these \( w_j \) are minimal coset representatives of \( 2\Lambda_0 \) in \( \Lambda_0 \) by Theorem 4.5.1, hence any two of them form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \) by Lemma 4.3.6.

Since any two of \( w_1, w_2, w_3 \) given by Theorem 4.5.3 form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \), we have \( \pm w_1 \pm w_2 \pm w_3 = 0 \) for some suitable signs. We now aim to determine these signs unambiguously.

Suppose that \((e_1^{(0)}, e_2^{(0)}, e_3^{(0)})\) is an ordering of the roots of \( E_0 \) which yields \((a_0, b_0, c_0)\) satisfying all conditions in (4.15). Let \((e_1'^{(0)}, e_2'^{(0)}, e_3'^{(0)})\) be another ordering of the roots of \( E_0 \) which also yields \((a'_0, b'_0, c'_0)\) satisfying all conditions in (4.15). If \( \{w_1, w_2, w_3\} \) and \( \{w'_1, w'_2, w'_3\} \) are the periods obtained by Theorem 4.5.3
using \((a_0, b_0, c_0)\) and \((a'_0, b'_0, c'_0)\) respectively, then for some suitable signs, we have

\[ \pm w_1 \pm w_2 \pm w_3 = 0 \quad \text{and} \quad \pm w'_1 \pm w'_2 \pm w'_3 = 0. \]

But since all \(w_j, w'_j\) are of minimal length, we must have \(w'_j = \pm w_k\) for some \(j, k\).

This leads to a system of two linear equations in terms of \(w_j\). It turns out that all signs in the first equation must be either identical or opposite to those in the second equation; since otherwise we will have either \(w_j = 0\) for some \(j\), or \(w_j = \pm w_k\) for some \(j \neq k\), which contradicts the fact that any two of \(w_1, w_2, w_3\) form a \(\mathbb{Z}\)-basis.

Finally, to obtain the right signs of \(w_1, w_2, w_3\), we explore all possible cases of rearranging the roots of \(E_0\). For each ordering \((e'_1(0), e'_2(0), e'_3(0))\), we compute \((a'_0, b'_0, c'_0)\) which satisfies all conditions in (4.15) as before (using Proposition 4.5.2 to rearrange the roots if necessary), and rewrite it in terms of \(a_0, b_0, c_0\). Next, note that there are eight possible triples \((\pm w_1, \pm w_2, \pm w_3)\). By applying \((a'_0, b'_0, c'_0)\) to Theorem 4.5.3 and rewriting \(w'_1, w'_2, w'_3\) in terms of \(w_1, w_2, w_3\), we “map” each triple \((\pm w_1, \pm w_2, \pm w_3)\) (which can be regarded as a linear equation) to another triple. By above argument, the right signs of \(w_1, w_2, w_3\) are therefore the ones which remain fixed for any ordering of the roots of \(E_0\). By exhaustive trials and errors, we will eventually see that

\[ w_1 - w_2 - w_3 = 0. \]

From this, we can state a more general result in view of complex AGM.

**Proposition 4.5.4.** Let \(a, b, c \in \mathbb{C} \setminus \{0\}\) satisfying \(c^2 = a^2 - b^2\) and the following:

\[ |a - b| \leq |a + b|, \quad |a - c| \leq |a + c|, \quad |c - ib| \leq |c + ib|. \]

Then

\[ \frac{1}{M(a, b)} - \frac{1}{M(c, ib)} + \frac{1}{M(ia, ic)} = 0. \]

A more symmetric version of this identity may be obtained by replacing \(a\) by
and imposing the relation $a^2 + b^2 + c^2 = 0$. We have not done so above, since it seemed more natural to state the theorem above as giving the set of all values of $M(a, b)$ rather than the set of all values of $M(ia, b)$.

### 4.6 Complex Elliptic Logarithms

In this section, we will finally generalise our method for computing period lattices of elliptic curves over $\mathbb{C}$ to compute elliptic logarithms of complex points.

As before, we let $E_0$ be an elliptic curve over $\mathbb{C}$ given by a Weierstrass equation (4.6). Recall that an elliptic logarithm of $P \in E_0(\mathbb{C})$ is a value $z_P \in \mathbb{C}$ such that $P = (\wp_{\Lambda_0}(z_P), \wp'_{\Lambda_0}(z_P))$. Note that $z_P$ is unique modulo $\Lambda_0$, where $\Lambda_0$ is the lattice of periods of the differential $dX_0/Y_0$, so that $E_0(\mathbb{C}) \cong \mathbb{C}/\Lambda_0$. We will show that $z_P$ can be determined by a similar method to that shown in Section 4.5.

Let $\ell_P$ be the straight line from 0 to $z_P$ (which is to be found) on the complex plane. Then

$$z_P = \int_{\ell_P} dz = \int_{C^{(0)}_P} \frac{dX_0}{Y_0},$$

where $C^{(0)}_P$ is the path on the elliptic curve $E_0$ defined by

$$C^{(0)}_P = \{(\wp_{\Lambda_0}(tz_P), \wp'_{\Lambda_0}(tz_P)) : 0 \leq t \leq 1\}.$$ 

Given $E_0$, we construct a chain of 2-isogenies $(E_n)_{n=0}^{\infty}$ using the method described in Section 4.4, with a good AGM sequence (and preferably, a strongly optimal one, as we will see later). Consider the diagram shown in Figure 4.1. For all $n \geq 1$, it follows that

$$\varphi_n \circ C^{(n)}_P = C^{(n-1)}_P,$$

where $C^{(n)}_P$ is the path on the elliptic curve $E_n$ defined by

$$C^{(n)}_P = \{(\wp_{\Lambda_n}(tz_P), \wp'_{\Lambda_n}(tz_P)) : 0 \leq t \leq 1\}.$$
Together with (4.12), we again have

\[
\int_{C_P^{(n)}} \frac{dX_n}{Y_n} = \int_{C_P^{(n-1)}} \frac{dX_{n-1}}{Y_{n-1}},
\]

and so

\[
z_P = \int_{C_P^{(0)}} \frac{dX_0}{Y_0} = \int_{C_P^{(1)}} \frac{dX_1}{Y_1} = \cdots = \int_{C_P^{(n)}} \frac{dX_n}{Y_n} = \cdots
\]

Recall the definition of \( \varphi_n : E_n \to E_{n-1} \) from (4.8). In Section 4.5.1, we have seen that (4.8) can be rewritten as (4.13), i.e.,

\[
X_n = \frac{(X_{n-1} + e_3^{(n)}) + s_n}{2}, \quad Y_n = \frac{Y_{n-1}}{1 - \frac{(e_3^{(n)} - e_1^{(n)})(e_3^{(n)} - e_2^{(n)})}{(X_n - e_3^{(n)})^2}}
\]

where

\[
s_n = \sqrt{(X_{n-1} - e_3^{(n)})^2 - 4(e_3^{(n)} - e_1^{(n)})(e_3^{(n)} - e_2^{(n)})}.
\]

Since the starting point of \( C_P^{(n)} \) is \( O \) for all \( n \geq 0 \), we must again choose the sign of \( s_n \) (for \( n \geq 1 \)) so that (4.14) holds, i.e.,

\[
|(X_{n-1} - e_3^{(n+1)}) - s_n| \leq |(X_{n-1} - e_3^{(n+1)}) + s_n|,
\]

and by continuity, this will be also satisfied along the path \( C_P^{(n)} \).

Note that if \( P \) is the 2-torsion point \((e_1^{(0)}, 0) \in E_0(\mathbb{C})\), then the above process is simply what we have seen in Section 4.5.1. In particular, if we construct our isogeny chain using a strongly optimal AGM sequence, then by Theorem 4.5.1, we will have \( 2z_P \) as a minimal coset representative modulo \( 2\Lambda_0 \), where two of which also form a \( \mathbb{Z} \)-basis for \( \Lambda_0 \). Hence from now on, we will always construct our isogeny chain using a strongly minimal AGM sequence.

For \( P = (x_0, y_0) \in E_0(\mathbb{C}) \), we can therefore construct the subsequent points \((x_n, y_n) \in E_n \) for all \( n \geq 1 \) using (4.13) and (4.14). This then gives us a limiting point \((x_\infty, y_\infty) \) on the limiting curve \( E_\infty \) (see (4.11) for the equation of \( E_\infty \)). As discussed
earlier, in this case we compute $M_S(a_0, b_0)$ with a strongly minimal sequence. It then follows that

$$z_P = \int_{C_P^{(0)}} \frac{dX_0}{Y_0} = \cdots = \int_{C_P^{(\infty)}} \frac{dX_\infty}{Y_\infty},$$

where $C_P^{(\infty)}$ is the path on $E_\infty$ given by

$$C_P^{(\infty)} = \left\{ \lim_{n \to \infty} \wp_{\Lambda_n}(tz_P), \lim_{n \to \infty} \wp'_{\Lambda_n}(tz_P) : 0 \leq t \leq 1 \right\}$$

(for the formulas of $\lim_{n \to \infty} \wp_{\Lambda_n}(z)$ and $\lim_{n \to \infty} \wp'_{\Lambda_n}(z)$, see Proposition 4.6.1), starting from $O$ to $(x_\infty, y_\infty)$. Note that if we choose $e_1^{(0)}$ differently, then this will result in a different sequence $((x_n, y_n))_{n=0}^{\infty}$, and hence a different limiting point $(x_\infty, y_\infty)$.

The next proposition confirms that $(x_\infty, y_\infty)$ does exist.

**Proposition 4.6.1.** Suppose $\{w_1, w_2\}$ is a $\mathbb{Z}$-basis for $\Lambda_0$ with $\Im(w_2/w_1) > 0$. Let $\Lambda_n = \langle w_1, 2^nw_2 \rangle$ and let $u = \exp(2\pi i z/w_1)$. Define

$$X_\infty(z) = \left( \frac{2\pi i}{w_1} \right)^2 \left( \frac{u}{(1-u)^2} + \frac{1}{12} \right)$$

$$Y_\infty(z) = \left( \frac{2\pi i}{w_1} \right)^3 \frac{u(1+u)}{(1-u)^3}.$$

Then as $n \to \infty$, $\wp_{\Lambda_n}(z)$ converges uniformly to $X_\infty(z)$, and $\wp'_{\Lambda_n}(z)$ converges uniformly to $Y_\infty(z)$. In consequence, $(x_\infty, y_\infty)$ exists and is not equal to $O$.

**Proof.** We will first prove the uniform convergence for $\wp_{\Lambda_n}(z)$; such convergence for $\wp'_{\Lambda_n}(z)$ can be proved in a similar way.

Let $X_n(z) = \wp_{\Lambda_n}(z)$, and let

$$\tau_n = \frac{2^nw_2}{w_1}, \quad u = \exp\left( \frac{2\pi iz}{w_1} \right), \quad q_n = \exp(2\pi i \tau_n).$$
Then by Proposition 1.3.1, we have

\[ X_n(z) = \left(\frac{2\pi i}{w_1}\right)^2 \left(\frac{u}{1-u^2} + \frac{1}{12} + \sum_{j=1}^\infty \left[ \frac{q_n^j u}{(1-q_n^j u)^2} + \frac{q_n^j u^{-1}}{(1-q_n^j u^{-1})^2} - \frac{2q_n^j}{(1-q_n^j)^2} \right] \right). \]

Since \( \Im(w_2/w_1) > 0 \), we have \( |q_0| < 1 \), and so \( |q_n| = |q_0|^2^n < 1 \). By writing \( z = \alpha w_1 + \beta w_2 \) with \( 0 \leq \alpha, \beta < 1 \), we also have \( |u| = \exp(-2\pi \beta \Im(\tau_0)) = |q_0|^\beta < 1 \).

Hence for all \( n > m \), we have

\[
\frac{|w_1|^2|X_n(z) - X_\infty(z)|}{4\pi^2} = \sum_{j=1}^\infty \left[ \frac{|q_n^j u|}{(1-|q_n^j u|)^2} + \frac{|q_n^j u^{-1}|}{(1-|q_n^j u^{-1}|)^2} + \frac{2|q_n^j|}{(1-|q_n^j|)^2} \right] \leq \sum_{j=1}^\infty \left[ \frac{|q_n^j u|}{(1-|q_n^j u|)^2} + \frac{|q_n^j u^{-1}|}{(1-|q_n^j u^{-1}|)^2} + \frac{2|q_n^j|}{(1-|q_n^j|)^2} \right].
\]

It can be seen that

\[
\sum_{j=1}^\infty \frac{|q_n^j u|}{(1-|q_n^j u|)^2} \leq \frac{1}{(1-|q_m u|)^2} \sum_{j=1}^\infty |q_m|^j \leq \frac{|q_m|}{(1-|q_m|)^3} = \frac{|q_0|^{2^m}}{(1-|q_0|^{2^m})^3}.
\]

Similarly,

\[
\sum_{j=1}^\infty \frac{2|q_n^j|}{(1-|q_n^j|^2)^2} \leq \frac{2|q_m|}{(1-|q_m|)^3} = \frac{2|q_0|^{2^m}}{(1-|q_0|^{2^m})^3},
\]

and

\[
\sum_{j=1}^\infty \frac{|q_n^j u^{-1}|}{(1-|q_n^j u^{-1}|)^2} \leq \sum_{j=1}^\infty \frac{|q_0|^{j 2^{m-1}}}{(1-|q_0|^{j 2^{m-1}})^2} \leq \frac{|q_0|^{2^m}}{|q_0|((1-|q_0|^{2^{m-1}})(1-|q_0|^{2^m}))}.
\]

Putting everything together, we finally have

\[
|X_n(z) - X_\infty(z)| \leq \frac{4\pi^2}{|w_1|^2} \left( \frac{3|q_0|^{2^m}}{(1-|q_0|^{2^m})^3} + \frac{|q_0|^{2^m}}{|q_0|((1-|q_0|^{2^{m-1}})(1-|q_0|^{2^m}))} \right).
\]
for all \( n > m \). Observe that the right-hand side is strictly decreasing as \( m \) increases. Hence for any given \( \epsilon > 0 \), we can always find \( m = m(\epsilon) \) (not depending on \( z \)) such that \( |X_n(z) - X_\infty(z)| < \epsilon \) for all \( n > m \). Thus \( X_n(z) \) converges uniformly to \( X_\infty(z) \), which proves the first part.

To prove the second part, we first note that our isogeny chain \((E_n)_{n=0}^\infty\), where each elliptic curve \( E_n \) is of the form

\[
E_n : \quad Y_n^2 = 4(X_n - e_1^{(n)})(X_n - e_2^{(n)})(X_n - e_3^{(n)}),
\]

is now constructed by a strongly minimal AGM sequence. This in turn yields a corresponding chain of lattices \((\Lambda_n)_{n=0}^\infty\). Let \( w_1 \in \mathbb{C} \) be the one obtained by Theorem 4.5.1. Then we have already seen in Section 4.5.1 that \( w_1 \) is a minimal coset representative modulo \( 2\Lambda_0 \), and the lattice chain is indeed optimal with limiting period \( w_1 \). Hence there exists \( w_2 \in \Lambda_0 \) such that \( \Lambda_n = \langle w_1, 2^n w_2 \rangle \) for all \( n \geq 0 \). Without loss of generality, we can also assume that \( \Im(w_2/w_1) > 0 \).

By definition of elliptic logarithm, we have

\[
X_n = \wp_{\Lambda_n}(z), \quad Y_n = \wp'_\Lambda(z),
\]

so we can regard \( X_n, Y_n \) as functions of \( z \). The first part then implies that \( X_n(z) \) converges uniformly to \( X_\infty(z) \), and \( Y_n(z) \) converges uniformly to \( Y_\infty(z) \), for any \( z \).

By letting \( z = z_P \), we finally have \( x_\infty = X_\infty(z_P) \) and \( y_\infty = Y_\infty(z_P) \). In addition, if \( z_P \) is not a lattice point in \( \Lambda_0 \), then \( u(z_P) \neq 1 \), and so \( (x_\infty, y_\infty) \neq O \). \( \square \)

Although one can compute the limiting point \((x_\infty, y_\infty)\) as above, we find that it is more convenient to obtain \((x_\infty, y_\infty)\) by making some change of variables. Below we will present one possible way to do this; an alternative version can be seen in [CT, Section 8.3].
Given an elliptic curve $E_0$ of the form (4.6) as before, let

$$a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}}, \quad b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}}.$$  

Recall that we wish to construct an isogeny chain $(E_n)_{n=0}^\infty$ using a strongly minimal AGM sequence. Hence we compute the AGM sequence $((a_n, b_n))_{n=0}^\infty$ using the method in Section 4.2, in such a way that $|a_n - b_n| \leq |a_n + b_n|$ for all $n \geq 0$. For $P = (x_0, y_0) \in E_0(\mathbb{C}) \setminus \{O\}$, we define

$$u_0 = \sqrt{x_0 - e_3^{(0)}}, \quad v_0 = \sqrt{x_0 - e_2^{(0)}}.$$  

The sign of $u_0$ can be chosen arbitrarily. To choose the sign of $v_0$, we first recall that for all $n \geq 1$ we map $(x_{n-1}, y_{n-1}) \in E_{n-1} \mapsto (x_n, y_n) \in E_n$ via (4.13) in such a way that $s_n$ satisfies the criterion (4.14). It can be verified that such criterion is equivalent to

$$|u_{n-1} - v_{n-1}| \leq |u_{n-1} + v_{n-1}| \quad (4.16)$$

(the situation when this becomes an equality will be explained later). Hence we choose the sign of $v_0$ so that $|u_0 - v_0| \leq |u_0 + v_0|$. Next, we define

$$t_0 = \begin{cases} \sqrt{x_0 - e_1^{(0)}} & \text{if } x_0 = e_j^{(0)} \text{ for some } j = 1, 2, 3, \\ \frac{y_0}{2u_0v_0} & \text{otherwise.} \end{cases}$$

Note that if $P$ is a 2-torsion point in $E_0(\mathbb{C})$, then $t_0$ is determined up to sign. This will have no effect on our result, since we will obtain half a primitive period of $\Lambda_0$ in this case. For a non-2-torsion point $P$, one can easily observe that if we had chosen the other sign for $u_0$, then $v_0$ would also have been negated, but $t_0$ remains unchanged. In fact, $t_0$ is where we embed the information on the sign of $z_P$. 
For \( n \geq 1 \), we define \( u_n, v_n, t_n \) in a similar way, i.e.,

\[
\begin{align*}
\sqrt{x_n - e_3^{(n)}} & = u_n, \\
\sqrt{x_n - e_2^{(n)}} & = v_n, \\
\sqrt{x_n - e_1^{(n)}} & = t_n = \frac{y_n}{2u_nv_n}.
\end{align*}
\]

We will show that these quantities can be determined by \( u_{n-1}, v_{n-1}, t_{n-1} \) obtained earlier. In (4.13), one can rewrite \( x_n \) in terms of \( u_n \) to obtain (after some algebra)

\[
u_n = \frac{u_{n-1} + v_{n-1}}{2}.
\]

Note that the sign of \( u_n \) is determined unambiguously. For \( v_n \), it is also easy to check that

\[
v_n = \sqrt{x_n - e_2^{(n)}} = \sqrt{u_n^2 - e_n^2},
\]

where \( e_n^2 = a_n^2 - b_n^2 \). Again, we choose the sign of \( v_n \) so that (4.16) holds. Similarly, one can show using (4.8) that

\[
t_n = \frac{u_n t_{n-1}}{v_n}.
\]

Recall that if we had chosen the other sign for \( u_0 \), then by (4.16), \( v_0 \) would be also negated, while \( t_0 \) remains unchanged. From these new formulas, it then follows that this will also negate both \( u_n, v_n \), while again \( t_n \) will remain unchanged.

By definition, we have \( u_n^2 = x_n - e_3^{(n)} \), \( v_n^2 = x_n - e_2^{(n)} \), and \( t_n^2 = x_n - e_1^{(n)} \). Since \( x_\infty = \lim_{n \to \infty} x_n \) exists by Proposition 4.6.1, and \( \lim_{n \to \infty} e_j^{(n)} \) exists for all \( j = 1, 2, 3 \) (see (4.10)), then each \( u_n, v_n, t_n \) has a limit as \( n \to \infty \). Let

\[
U = \lim_{n \to \infty} u_n = \lim_{n \to \infty} v_n, \quad T = \lim_{n \to \infty} t_n, \\
M = M(a_0, b_0) \quad (\text{where } |a_0 - b_0| \leq |a_0 + b_0|).
\]
We finally have

\[
\begin{align*}
x_\infty &= \left( \lim_{n \to \infty} u_n \right)^2 + \lim_{n \to \infty} e_3^{(n)} = U^2 - \frac{M^2}{3} \\
y_\infty &= 2 \left( \lim_{n \to \infty} t_n \right) \left( \lim_{n \to \infty} u_n \right) (\lim_{n \to \infty} v_n) = 2TU^2.
\end{align*}
\]  

(4.17)

It is easy to see that if \( P = (e_1^{(0)}, 0) \), then \( t_n = 0 \) for all \( n \geq 0 \), so \( T = 0 \).

Consider the limiting curve \( E_\infty \). Similar to [BM88], we again define \( t, \theta \) by

\[
\begin{align*}
X_\infty &= t^2 + \frac{2M^2}{3}, \\
Y_\infty &= 2t(t^2 + M^2), \\
\tan \theta &= \frac{t}{M}.
\end{align*}
\]

As before, this gives us

\[
\frac{dX_\infty}{Y_\infty} = \frac{d\theta}{M}
\]

and

\[
t = \frac{Y_\infty}{2 \left( X_\infty + \frac{M^2}{3} \right)}.
\]

Hence we have

\[
P = O \iff t = \infty \iff \cos \theta = 0 \iff \theta = \frac{(2k + 1)\pi}{2} \quad \text{for some} \quad k \in \mathbb{Z}.
\]

Let \( \tan \theta^* = T/M \). Then we have

\[
z_P = \int_{c_P(\infty)} \frac{dX_\infty}{Y_\infty} = \frac{1}{M} \int_{\frac{(2k+1)\pi}{2}}^{\theta^*} d\theta = \frac{\theta^* - \frac{(2k+1)\pi}{2}}{M}.
\]

If we choose \( k \) differently, then \( z_P \) will be changed by adding a multiple of \( w_1 = \pi/M \), which is a primitive period of \( \Lambda_0 \) by Theorem 4.5.1. Thus \( z_P \) we just obtained is unique modulo \( \Lambda_0 \). By letting \( k = 0 \) and using the fact that \( \tan(\theta - \pi/2) = -1/\tan(\theta) \), we finally have

\[
z_P = \frac{\theta^* - \frac{\pi}{2}}{M} = \frac{-1}{M} \arctan \left( \frac{M}{T} \right).
\]  

(4.18)
If \( T = 0 \) (i.e., \( x_0 = e_1^{(0)} \)), then one can use the fact that \( \arctan(x) \to \pi/2 \) as \( x \to \infty \) to obtain \( z_P = -\pi/(2M) \), which (up to sign) is half of the primitive period \( w_1 \) obtain by Theorem 4.5.1.

To summarise, we have the following algorithm.

**Algorithm 4.6.2** (Elliptic Logarithms of Complex Points). Given an elliptic curve \( E_0 \) defined over \( \mathbb{C} \) and a point \( P \in E_0(\mathbb{C}) \), return an elliptic logarithm of \( P \).

**Input:** An elliptic curve \( E_0 \) of the form (4.6), and \( P = (x_0, y_0) \in E_0(\mathbb{C}) \).

1. If \( P = O \), return \( z_P = 0 \).
2. Let \( a_0 = \sqrt{e_1^{(0)} - e_3^{(0)}} \) and \( b_0 = \sqrt{e_1^{(0)} - e_2^{(0)}} \). Choose the sign of \( b_0 \) so that \( |a_0 - b_0| \leq |a_0 + b_0| \).
3. Let \( u_0 = \sqrt{x_0 - e_3^{(0)}} \) and \( v_0 = \sqrt{x_0 - e_2^{(0)}} \). Choose the sign of \( v_0 \) so that \( |u_0 - v_0| \leq |u_0 + v_0| \).
4. Let \( t_0 = \begin{cases} \sqrt{x_0 - e_1^{(0)}} & \text{if } x_0 = e_j^{(0)} \text{ for some } j = 1, 2, 3, \\ \frac{y_0}{2u_0v_0} & \text{otherwise.} \end{cases} \)

   If \( t_0 = 0 \), return \( z_P = \pi/(2M(a_0, b_0)) \).
5. Set \( n = 1 \). Repeat the following:
   
   (a) Let \( a_n = \frac{a_{n-1} + b_{n-1}}{2}, \quad b_n = \sqrt{a_{n-1}b_{n-1}}, \quad c_n^2 = a_n^2 - b_n^2 \).

      Choose the sign of \( b_n \) so that \( |a_n - b_n| \leq |a_n + b_n| \).

   (b) Let \( u_n = (u_{n-1} + v_{n-1})/2 \) and \( v_n = \sqrt{u_n^2 - c_n^2} \). Choose the sign of \( v_n \) so that \( |u_n - v_n| \leq |u_n + v_n| \).

   (c) Let \( t_n = u_n t_{n-1}/v_n \).

   (d) \( n \leftarrow n + 1 \).
until $|a_n - b_n|$ and $|u_n - v_n|$ are sufficiently small. Let $M$ and $T$ be the limiting values of $a_n$ and $t_n$ respectively.

Output:

$$z_P = -\frac{1}{M} \arctan \left( \frac{M}{T} \right).$$

Remark. If the criterion (4.16) becomes an equality, then we have $\Re(u_{n-1}/v_{n-1}) = 0$, or equivalently, $x_{n-1}$ lies on the straight line joining $e_2^{(n-1)}$ and $e_3^{(n-1)}$. To avoid the ambiguity of the sign of $v_{n-1}$, one can recover all $e_j^{(n-1)}$, $x_{n-1}$, and rearrange the roots of $E_{n-1}$ so that the new $u'_{n-1}, v'_{n-1}$ satisfy a strict inequality. Nevertheless, we will see in Example 4.7.5 that both $\pm v_{n-1}$ are equally good for computing $z_P$.

Moreover, the requirement for sufficiently small $|u_n - v_n|$ as another stopping criterion in Algorithm 4.6.2 may be omitted in practice, since our experience shows that both AGM sequences $((a_n, b_n))_{n=0}^\infty$ and $((u_n, v_n))_{n=0}^\infty$ seem to converge roughly at the same rate.

Finally, note that our formulas shown in Algorithm 4.6.2 are somewhat similar to the ones in Cohen’s algorithm [Coh93, Algorithm 7.4.8] for computing elliptic logarithms of real points (where our $u_n$ is called $c_n$ in his algorithm). Using the fact that $U^2 = T^2 + M^2$, we can rewrite $z_P$ as

$$z_P = -\frac{1}{M} \arcsin \left( \frac{M}{U} \right),$$

which is similar (up to sign) to the output of Cohen’s algorithm. By writing $z_P$ this way, however, we have an ambiguity of the sign of $z_P$, since this information is embedded in $T$. Our formulas remove this ambiguity.

4.7 Examples

Finally, we will illustrate our method for computing period lattices of elliptic curves over $\mathbb{C}$ and elliptic logarithms of complex points via some examples.
For computational purposes, we have implemented our algorithms in MAGMA (see Appendix A.1 for the code), including our own function for computing an optimal AGM sequence (since the existing function in MAGMA does not always return an optimal one). Note that all complex numbers in our examples are computed correctly up to 100 decimal places, but only the first 20 decimal places are shown.

**Example 4.7.1.** Let $E$ be the elliptic curve over $\mathbb{C}$ given by the Weierstrass equation $Y^2 = 4(X - e_1)(X - e_2)(X - e_3)$, where

$$e_1 = 3 - 2i, \quad e_2 = 1 + i, \quad e_3 = -4 + i.$$ 

Observe that $\sum_{j=1}^{3} e_j = 0$. We will compute the period lattice of $E$ using the method described in Section 4.5.4. To do this, first we let $E_0 = E$ and calculate

$$a_0 = \sqrt{e_1 - e_3}, \quad b_0 = \sqrt{e_1 - e_2}, \quad c_0 = \sqrt{a_0^2 - b_0^2},$$

in such a way that the signs of $a_0, b_0, c_0$ satisfy all conditions in (4.15), i.e.,

$$|a_0 - b_0| \leq |a_0 + b_0|, \quad |a_0 - c_0| \leq |a_0 + c_0|, \quad |c_0 - ib_0| \leq |c_0 + ib_0|.$$ 

In this example, one can verify that such $a_0, b_0, c_0$ are

$$a_0 = 2.703310295347530788867\ldots - i0.55487525889334275023\ldots$$

$$b_0 = 1.67414922803554004044\ldots - i0.89597747612983812471\ldots$$

$$c_0 = 2.23606797749978969640\ldots .$$

In fact, all conditions in (4.15) are strictly inequalities in this case, so the period
lattice of $E$ is non-rectangular. By Theorem 4.5.3, we have

$$w_1 = 1.29215151748713051904 \ldots + i0.44759218107818896608 \ldots$$

$$w_2 = 1.42661373451784507587 \ldots - i0.80963848056301882107 \ldots$$

$$w_3 = -0.13446221703071455682 \ldots + i1.25723066164120778715 \ldots$$

and any two of $w_j$ form a $\mathbb{Z}$-basis for the period lattice $\Lambda$ of $E$. In our computation, we also have $|w_1 - w_2 - w_3| \approx 10^{-100}$, which agrees with the result given by Proposition 4.5.4. Note that these $w_j$ are minimal coset representatives of $2\Lambda$ in $\Lambda$.

Next, we wish to compute an elliptic logarithm of the point $P = (2-i, 8+4i) \in E(\mathbb{C})$ (which has infinite order), i.e., a value $z_P$ such that $P = (\wp_\Lambda(z_P), \wp'_\Lambda(z_P))$. Using $a_0, b_0$ as above, Algorithm 4.6.2 shows that

$$x_\infty = 1.67097624471645689380 \ldots - i1.23329436157704253331 \ldots$$

$$y_\infty = 7.78679958972849436041 \ldots + i4.93520281519385276354 \ldots$$

and then

$$z_P = -0.72212997914002299126 \ldots + i0.01717122412650902249 \ldots$$

Let $u_P = \exp(2\pi i z_P / w_1)$. One can verify that

$$\left| x_\infty - \left( \frac{2\pi i}{w_1} \right)^2 \left( \frac{u_P}{(1 - u_P)^2} + \frac{1}{12} \right) \right| \approx 10^{-100}$$

$$\left| y_\infty - \left( \frac{2\pi i}{w_1} \right)^3 \frac{u(1+u)}{(1-u)^3} \right| \approx 10^{-100}.$$ 

which agrees with our result in Proposition 4.6.1.

Note that $z_P$ is unique modulo $\Lambda$. Depending on a $\mathbb{Z}$-basis $\{w_1, w_2\}$ for $\Lambda$, it can be seen that $z_P$ we just obtained may not necessarily lie in the fundamental
4.7. Examples

parallelogram

\[ F_{w_1, w_2} = \{ \lambda_1 w_1 + \lambda_2 w_2 : 0 \leq \lambda_1, \lambda_2 < 1 \} \]

In this example, one can check that

\[ z_P = (-0.33249952362000772434 \ldots) w_1 - (0.20502411273191295799 \ldots) w_2 \]

\[ \equiv (0.66750047637999227565 \ldots) w_1 + (0.79497588726808704200 \ldots) w_2, \]

and so \( z_P \) is not in the fundamental parallelogram spanned by \( \{ w_1, w_2 \} \). Finally, we verify that

\[ |\wp(\Lambda)(z_P) - x(P)| \approx 10^{-99}, \quad |\wp'(\Lambda)(z_P) - y(P)| \approx 10^{-100}. \]

Moreover, we have

\[ |\wp\left(\frac{w_1}{2}\right) - e_1| \approx 10^{-99}, \quad |\wp\left(\frac{w_2}{2}\right) - e_2| \approx 10^{-100}, \quad |\wp\left(\frac{w_3}{2}\right) - e_3| \approx 10^{-100}, \]

and \( \wp'(w_j/2) \approx 0 \) for all \( j = 1, 2, 3 \).

Finally, one can verify the above results by the following MAGMA instructions together with our code given in Appendix A.1 (elog.m) and Appendix A.6.3 (wp.m):

```
> Attach("elog.m"); // main program for computing AGM and periods
> Attach("wp.m"); // for computing Weierstrass \wp-function and its derivative
> C<i> := ComplexField(100);
> e1 := 3 - 2*i;
> e2 := 1 + i;
> e3 := -e1-e2;
> // SetVerbose("Elog", 1); // enable this line to see more details
> // Find the periods of E
> w1, w2, w3 := Explode(PeriodLattice([e1,e2] : Prec := 95));
> // Verify the linear relation given by Proposition 4.5.4
> // x-coordinates
> Abs(w1-w2-w3);
> 1.1781644357515005406272652499344840363658756213424388601691323101044030022\1613252867545093671261E-100
> // Verify if w1, w2, w3 are correct
> Abs(WeierstrassP([w2, w1], w1/2, 50) - e1);
> 2.074753133004389699654294381275847188126730617040279215444803133824417103268\1614311548981370662044E-99
> // Abs(\wp(\Lambda)(\wp[P], w2/w1, w1/2, 50) - e1);
> 3.0146407176779044548666662279222953749205619468450345921692677749128000\15892434123830156519330E-100
> Abs(\wp(\Lambda)(\wp[P], w2/w1, w3/2, 50) - e3);
> 3.186902681996780381373548295364491416844348967066576261499869952898413491169\145991879778156020261E-100
```
Example 4.7.2 (Rectangular Lattice). Let $E$ be the elliptic curve over $\mathbb{C}$ given by the Weierstrass equation $Y^2 = 4(X - e_1)(X - e_2)(X - e_3)$, where

$$e_1 = 1 + 3i, \quad e_2 = -4 - 12i, \quad e_3 = 3 + 9i.$$ 

Observe that $\sum_{j=1}^{3} e_j = 0$ and all $e_j$ are collinear. By letting $E_0 = E$ and computing $a_0, b_0, c_0$ as before, we have

$$a_0 = 1.47046851723128684330 \ldots - i2.04016608641756892919 \ldots$$

$$b_0 = -3.22578581905571472955 \ldots - i2.32501487101070997214 \ldots$$

$$c_0 = 2.75099469475848546460 \ldots - i3.81680125374499001591 \ldots.$$ 

This time, however, we have $|a_0 - b_0| = |a_0 + b_0|$, while the other two conditions in (4.15) are strictly inequalities. Let $\Lambda$ be the period lattice of $E$. Then we have two minimal elements (up to sign) in one coset of $2\Lambda$ in $\Lambda$, so $\Lambda$ is rectangular.

To obtain an orthogonal $\mathbb{Z}$-basis for $\Lambda$, first we let $w, w' = \pi/M(a_0, \pm b_0)$. In this example, we have

$$w = -0.29920293143872535713 \ldots + i1.10940038117892953702 \ldots$$

$$w' = 1.14708588706988127437 \ldots + i0.06697438037476960963 \ldots.$$
One can check that $|w| = |w'|$. Let $w_1, w_2 = (w \pm w')/2$. Then $w_1, w_2$ form an orthogonal $\mathbb{Z}$-basis for $\Lambda$, as proved in Lemma 4.3.5. Here, we have

$$w_1 = 0.42394147781557795862 \ldots + i0.58818738077684957333 \ldots$$
$$w_2 = -0.72314440925430331575 \ldots + i0.5212130040207996369 \ldots$$

Note that $\Re(w_2/w_1) = 0$.

Let $z_P$ be an elliptic logarithm of the point $P = (3 + 2i, 28 - 14i) \in E(\mathbb{C})$ (which has infinite order). Algorithm 4.6.2 shows that

$$z_P = -0.42599662534207481578 \ldots - i0.02491254923738153924 \ldots$$
$$\equiv (0.6285822453897766753 \ldots)w_1 + (0.37134662195976180031 \ldots)w_2.$$

Finally, we verify that

$$|\wp_\Lambda(z_P) - x(P)| \approx 10^{-98}, \quad |\wp'_\Lambda(z_P) - y(P)| \approx 10^{-97},$$

and moreover,

$$|\wp_\Lambda(w_1/2) - e_2| \approx 10^{-99}, \quad |\wp_\Lambda(w_2/2) - e_3| \approx 0, \quad |\wp_\Lambda(w/2) - e_1| \approx 10^{-99},$$
$$|\wp'_\Lambda(w_1/2)| \approx 10^{-99}, \quad |\wp'_\Lambda(w_2/2)| \approx 10^{-99}, \quad |\wp'_\Lambda(w/2)| \approx 10^{-100}.$$
Example 4.7.3. Let \( K = \mathbb{Q}(\theta) \) where \( \theta \) is a root of the polynomial \( x^3 - 2 \). Let \( E \) be the elliptic curve defined over \( K \) given by the Weierstrass equation

\[
E : \quad Y^2 = 4(X - \theta)(X - 1)(X + 1 + \theta).
\]

Note that \( K \) has one real embedding and one conjugate pair of complex embeddings.
Let $E_1, E_2$ be the real and complex embedding of $E$ respectively, i.e.,

\[
E_1 : \quad Y^2 = 4(X - \sqrt[3]{2})(X - 1)(X + 1 + \sqrt[3]{2})
\]
\[
E_2 : \quad Y^2 = 4(X - \omega \sqrt[3]{2})(X - 1)(X + 1 + \omega \sqrt[3]{2})
\]

where $\omega = \exp(2\pi i/3)$ is a cube root of unity. Since $E_1$ has three real roots, then the period lattice of $E_1$ is rectangular. In fact, by letting $e_1^{(0)} = \sqrt[3]{2}, e_2^{(0)} = 1, e_3^{(0)} = -1 - \sqrt[3]{2}$, we can compute $a_0, b_0, c_0$ satisfying all the conditions in (4.15) as follows:

\[
a_0 = 1.87612422291002530767 \ldots
\]
\[
b_0 = 0.50982452853395859808 \ldots
\]
\[
c_0 = 1.80552514518487755254 \ldots
\]

One can then verify that $|c_0 - ib_0| = |c_0 + ib_0|$. As before, we compute

\[
w = \frac{\pi}{M(c_0, ib_0)} = 2.90130425944817643666 \ldots - i1.70677932803214980295 \ldots
\]
\[
w' = \frac{\pi}{M(c_0, -ib_0)} = \bar{w},
\]

and let $w_1, w_2 = (w \pm w')/2$. Then $w_1, w_2$ form an orthogonal $\mathbb{Z}$-basis for the period lattice of $E_1$. In this example, we have $w_1 = \Re(w)$ and $w_2 = i\Im(w)$.

Nevertheless, the period lattice of $E_2$ is non-rectangular, since all roots of $E_2$ are not collinear. In fact, by letting $e_1^{(0)} = -1 - \omega \sqrt[3]{2}, e_2^{(0)} = 1, e_3^{(0)} = \omega \sqrt[3]{2}$ (note that we use Proposition 4.5.2 here to ensure that $a_0, b_0, c_0$ satisfy all the conditions in (4.15)), we have

\[
a_0 = 1.10851094368231305521 \ldots - i0.98431471713501219051 \ldots
\]
\[
b_0 = 0.43669517024285334726 \ldots - i1.24929666083200513980 \ldots
\]
\[
c_0 = 1.34004098848655674756 \ldots - i0.40712323180652750769 \ldots
\]
In fact, one can check that all the conditions in (4.15) are strictly inequalities, hence this also confirms that the period lattice of $E_2$ is non-rectangular. By Theorem 4.5.3, we finally obtain

$$w_1 = 1.28194824894788708942 \ldots + i1.88277404359595361782 \ldots$$

$$w_2 = 2.36557653380849535471 \ldots - i0.03808700290170419307 \ldots$$

$$w_3 = -1.08362828486060826529 \ldots + i1.92086104649765781090 \ldots$$

with $|w_1 - w_2 - w_3| \approx 10^{-100}$, as claimed by Proposition 4.5.4.

**Example 4.7.4.** Let $E$ be the elliptic curve over $\mathbb{C}$ given by the Weierstrass equation $Y^2 = 4(X - e_1)(X - e_2)(X - e_3)$, where

$$e_1 = -1 - 3i, \quad e_2 = 3 + i, \quad e_3 = -2 + 2i.$$ 

Observe that $\sum_{j=1}^{3} e_j = 0$ and $|e_1 - e_3| = |e_2 - e_3|$. Thus $e_1, e_2, e_3$ form an isosceles triangle, as explained in Section 4.5.3. By letting $E_0 = E$ and computing $a_0, b_0, c_0$ satisfying all the conditions in (4.15) as before, we have

$$a_0 = 1.74628455779589152702 \ldots - i1.43161089573822132705 \ldots$$

$$b_0 = 0.91017972112445468260 \ldots - i2.19736822693561993207 \ldots$$

$$c_0 = 2.24711142509587014360 \ldots - i0.22250788030178260411 \ldots$$

Hence by Theorem 4.5.3, we obtain

$$w_1 = 0.81646689790312614904 \ldots + i1.1077333340066743861 \ldots$$

$$w_2 = 1.36061503191563570645 \ldots - i0.20595647167234558716 \ldots$$

$$w_3 = -0.54414813401250955741 \ldots + i1.31368980507301302578 \ldots$$

with $|w_1 - w_2 - w_3| \approx 10^{-100}$, as claimed by Proposition 4.5.4. In addition, one can
check that $\Re(w_1/w_3) = 1/2$ as claimed in Section 4.5.3. Let $\Lambda$ be the period lattice of $E$. We finally verify that $|\wp_\Lambda(w_j/2) - e_j| \approx 10^{-100}$ for all $j = 1, 2, 3$, and

$$|\wp'_\Lambda(w_1/2)| \approx 10^{-99}, \quad |\wp'_\Lambda(w_2/2)| \approx 10^{-100}, \quad |\wp'_\Lambda(w_3/2)| \approx 10^{-99}.$$  

**Example 4.7.5.** Let $E$ be the elliptic curve over $\mathbb{C}$ given by the Weierstrass equation $Y^2 = 4(X - e_1)(X - e_2)(X - e_3)$, where

$$e_1 = -2 - 2i, \quad e_2 = -1 + 6i, \quad e_3 = 3 - 4i.$$  

By Theorem 4.5.3, the period lattice $\Lambda$ of $E$ has the following minimal coset representatives; two of which form a $\mathbb{Z}$-basis for $\Lambda$:

$$w_1 = 1.04665075729832942736\ldots + i0.45525281255263173893\ldots$$

$$w_2 = 0.67791651620742852409\ldots - i0.7797238161544820221\ldots$$

$$w_3 = 0.36873424109000900326\ldots + i1.23322519416807994115\ldots$$

with $|w_1 - w_2 - w_3| \approx 10^{-101}$, which again agrees with Proposition 4.5.4.

Now we wish to find an elliptic logarithm of the point

$$P = (1 + i, \sqrt{12 + 492i}) = (1 + i, 15.8768\ldots + i15.4942\ldots) \in E(\mathbb{C}).$$

Letting $E_0 = E$ and computing $u_0, v_0$, we have

$$u_0^2 = x(P) - e_3^{(0)} = -2 + 5i, \quad v_0^2 = x(P) - e_2^{(0)} = 2 - 5i = -u_0^2.$$  

Thus $|u_0 - v_0| = |u_0 + v_0|$. If we choose

$$v_0 = \sqrt{2 - 5i} = 1.9216\ldots - i1.3009\ldots,$$
then Algorithm 4.6.2 gives

\[ z_\ell^{(1)} = -0.5201357344339582317 \ldots + i0.13628275717388366013 \ldots. \]

However, if we choose \(-v_0\), then we obtain

\[ z_\ell^{(2)} = 0.15778078177346870092 \ldots - i0.64168962444156454207 \ldots. \]

But then \(z_\ell^{(2)} - z_\ell^{(1)} = w_2\). Hence both choices for \(v_0\) are equally good for computing elliptic logarithms. Finally, we verify that

\[ \left| x(P) - \wp_{\Lambda_0}(z_\ell^{(1)}) \right| \approx 10^{-98}, \quad \left| y(P) - \wp_0'(z_\ell^{(1)}) \right| \approx 10^{-98}. \]

In conclusion, we have presented a complete method, based on complex AGM, for computing period lattices of elliptic curves defined over \(\mathbb{C}\), and generalised it into an algorithm for computing elliptic logarithms of complex points. As we can see from the above illustrative examples, this work, which is done in collaboration with Professor John E. Cremona, finally allows one to compute both quantities on any elliptic curves over \(\mathbb{C}\), while such computations in the past were possible only for elliptic curves over \(\mathbb{R}\). For more information on precise running time of complex AGM, see Dupont’s thesis [Dup06] or his paper [Dup].

In the next chapter, we will bring all the main results we have obtained so far to illustrate their applications in assisting some essential computations in the arithmetic of elliptic curves over number fields.
Chapter 5

Applications

In this final chapter, we will illustrate the applications of all the main results we have obtained so far towards some computations in the arithmetic of elliptic curves over number fields, whose existing methods experienced some difficulties in the past due to lack of certain information on elliptic curves.

In this chapter, we will start by showing how to compute a lower bound for the canonical height (see Chapter 2 and 3) and use it to determine Mordell–Weil bases for elliptic curves over number fields. Then we will move on to demonstrate an algorithm of Smart and Stephens [SS97] for computing integral points on elliptic curves over number fields, which involves determining complex elliptic logarithms (see Chapter 4) of all generators of Mordell–Weil bases. Finally, we will conclude this chapter by illustrating some examples of finding all elliptic curves with everywhere good reduction based on the method of Cremona and Lingham [CL07], which requires integral points on elliptic curves of a certain type over number fields.

5.1 Height Bound III: Examples

In this section, we will show several illustrative examples on how to use Theorem 3.4.1 to determine a positive lower bound for the canonical height on elliptic curves over number fields. Note that our computation, which also involves real and complex
elliptic logarithms (see Chapter 4), will be more sophisticated if the base number fields are not totally real. We have implemented our algorithm for computing the following examples in MAGMA; its source code can be found in Appendix A.3. For a brief demonstration of how to use our program, see Example 5.1.4.

5.1.1 Case I: Totally Real Number Fields

We will first concentrate on the case when our elliptic curves are defined over totally real number fields. As we will see, this will require periods of elliptic curves over $\mathbb{R}$ and elliptic logarithms of real points, which can be obtained by Algorithm 4.6.2 or Cohen’s algorithms [Coh93, Algorithm 7.4.7 and 7.4.8]. For the relevant notations, the reader should refer to Chapter 2.

Example 5.1.1. Let $E = E_1$, where $E_1$ is the elliptic curve defined over $K = \mathbb{Q}(\sqrt{2})$ given by the Weierstrass equation

$$E_1 : \quad y^2 = x^3 + x + (1 + 2\sqrt{2}).$$

The discriminant $\Delta$ of $E$ is $-3952 - 1728\sqrt{2}$. Moreover, we have $\langle \Delta \rangle = p_1^8 p_2^2 p_3$, where

$$p_1 = \langle \sqrt{2} \rangle, \quad p_2 = \langle 7, 3 + \sqrt{2} \rangle, \quad p_3 = \langle 769, 636 + \sqrt{2} \rangle,$$

are prime ideals. Since $\text{ord}_{p_j}(\Delta) < 12$ for all $j$, then $E$ is given by a globally minimal model, and so $M_E = 1$.

As explained in Section 3.4, our algorithm, based on Theorem 3.4.1, will start by checking whether a given $\mu > 0$ is a lower bound for the canonical height on $E_{\text{gt}}(K)$ by computing $B_n(\mu)$ for $n = 1, \ldots, n_{\text{max}}$. If $B_n(\mu) < 1$ for some $n$, then $\mu$ is indeed a lower bound. Otherwise, we proceed to compute $\bigcap_{n=1}^{n_{\text{max}}} S_n(v)(-B_n(\mu), B_n(\mu))$ for every real archimedean place $v \in M_K$ (here, we do not have to compute any $T_n^{(v)}$, since $K$ is totally real). If the intersection is empty for some $v$, then $\mu$ is a lower bound. Note that we obtain no conclusion if none of the intersections is empty.
5.1. Height Bound III: Examples

In this example, we define $v_+, v_-$ to be the real archimedean place of $K$ whose associated real embedding sends $\sqrt{2} \mapsto \pm 1.414\ldots$ respectively. By letting $\mu = 1$ and $n_{\text{max}} = 5$, we have

$$B_1(\mu) = 8.117389, \quad B_2(\mu) = 8.186971 \times 10^2, \quad B_3(\mu) = 7.213201 \times 10^7,$$

$$B_4(\mu) = 5.421641 \times 10^{12}, \quad B_5(\mu) = 5.685757 \times 10^{21}.$$  

Since none of these is less than 1, we have to compute $\bigcap_{n=1}^{n_{\text{max}}} S_{n}^{(v)}(-B_n(\mu), B_n(\mu))$ for every $v \in M^r_K$. Recall from Section 2.5 that $S_n^{(v)}(\xi_1, \xi_2)$ is defined in terms of $\psi_v(\xi_1), \psi_v(\xi_2)$, where $\psi_v : E_0^{(v)}(\mathbb{R}) \to [1/2, 1)$ is the normalised elliptic logarithm of the “higher” of the two points on $E_0^{(v)}$ with the same $x$-coordinate. For $v = v_+$, one can check that the corresponding real embedding $E^{(v)}$ has only one real root at $\beta_v = -1.352786$. Using Algorithm 4.6.2, we have (after normalisation)

$$\psi_v(B_1(\mu)) = 0.891227,$$

which yields 1 $S_1^{(v)}(-B_1(\mu), B_1(\mu)) = [0.108773, 0.891227]$.

Computing $S_n^{(v)}(-B_n(\mu), B_n(\mu))$ for all $n = 2, \ldots, n_{\text{max}}$ in a similar way, we will eventually see that $\bigcap_{n=1}^{n_{\text{max}}} S_n^{(v)}(-B_n(\mu), B_n(\mu)) \neq \emptyset$. A similar procedure also shows that another intersection associated to $v = v_-$ is non-empty. Hence we fail to show that $\mu = 1$ is a lower bound on $E_{gr}^r(K)$, in which case we shall repeat the above computation with a smaller $\mu$ (and/or a larger $n_{\text{max}}$). On the other hand, if $\mu$ is known to be a lower bound, then we can repeat such process with a larger $\mu$ to see if it is still a lower bound. This refinement can be done repeatedly as required.

After a number of refinements as shown in Table 5.1, our algorithm finally shows that

$$\hat{h}(P) > \mu = 0.2415$$

1 Only $\psi_v(B_1(\mu))$ is required in this case, since $-B_1(\mu) < \beta_v < B_1(\mu)$. 


Table 5.1: Illustration of the algorithm for Example 5.1.1

<table>
<thead>
<tr>
<th>Initial $\mu$</th>
<th>Initial $n_{\max}$</th>
<th>Is any $B_n(\mu) &lt; 1$?</th>
<th>Is any intersection empty?</th>
<th>Is $\mu$ a lower bound?</th>
<th>Next $\mu$</th>
<th>Next $n_{\max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>5</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.5000</td>
<td>6</td>
</tr>
<tr>
<td>0.5000</td>
<td>6</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2500</td>
<td>7</td>
</tr>
<tr>
<td>0.2500</td>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1250</td>
<td>8</td>
</tr>
<tr>
<td>0.1250</td>
<td>8</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.1875</td>
<td>8</td>
</tr>
<tr>
<td>0.1875</td>
<td>8</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2187</td>
<td>8</td>
</tr>
<tr>
<td>0.2187</td>
<td>8</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2343</td>
<td>8</td>
</tr>
<tr>
<td>0.2343</td>
<td>8</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2421</td>
<td>8</td>
</tr>
<tr>
<td>0.2421</td>
<td>8</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2382</td>
<td>9</td>
</tr>
<tr>
<td>0.2382</td>
<td>9</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2402</td>
<td>9</td>
</tr>
<tr>
<td>0.2402</td>
<td>9</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2412</td>
<td>9</td>
</tr>
<tr>
<td>0.2412</td>
<td>9</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2416</td>
<td>9</td>
</tr>
<tr>
<td>0.2416</td>
<td>9</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2414</td>
<td>10</td>
</tr>
<tr>
<td>0.2414</td>
<td>10</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2415</td>
<td>10</td>
</tr>
<tr>
<td>0.2415</td>
<td>10</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2415</td>
<td>11</td>
</tr>
<tr>
<td>0.2415</td>
<td>11</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.2416</td>
<td>9</td>
</tr>
</tbody>
</table>

for all non-torsion points $P \in E_{gr}(K)$. Nevertheless, the lower bound for $E_{gr}(K)$ derived from Theorem 2.4.2 is not as good as this one. In this example, we have

$$\alpha_{v_+} = 1.096562, \quad \alpha_{v_-} = 1.001830,$$

and so $\alpha_{v_+} \alpha_{v_-} = 1.098569$. We now choose a prime ideal $p$ whose norm is greater than $\sqrt{\alpha_{v_+} \alpha_{v_-}}$, and set $n = e_p$. To minimise $n$, we choose $p = \langle \sqrt{2} \rangle$ to obtain $n = e_p = 2$. Then we have $D_E(2) = 1.386294$, which finally yields the lower bound

$$\mu_0 = \frac{D_E(n) - \log(\alpha_{v_+} \alpha_{v_-})}{[K : \mathbb{Q}]n^2} = \frac{1.386294 - \log(1.098569)}{8} = 0.1615.$$

In order to obtain a lower bound for the canonical height on $E(K)$, we first note that the Tamagawa indices $c_v$ at $v = p_1, p_2, p_3$ are 4, 2, and 1 respectively. Moreover, one can easily see that both real embeddings of $E$ have only real root, so $c_{v_+} = c_{v_-} = 1$. Hence $c = \text{lcm}\{4, 2, 1\} = 4$. By Lemma 2.1.1, we finally have

$$\hat{h}(P) > \mu/c^2 = 0.2415/4^2 = 0.0150$$

for all non-torsion points $P \in E(K)$. 
Example 5.1.2. Let $E = E_2$, where $E_2$ is the elliptic curve defined over $K = \mathbb{Q}(\sqrt{7})$ given by the Weierstrass equation

$$E_2 : y^2 + (3 + 3\sqrt{7})xy + y = x^3 + (26 + 4\sqrt{7})x^2 + x.$$ 

The discriminant $\Delta$ of $E$ is $-937513 - 299394\sqrt{7}$. Moreover, $\langle \Delta \rangle$ can be factorised into a product of prime ideals as $p_1 p_2 p_3$, where

$$p_1 = \langle 4219, 1083 + \sqrt{7} \rangle, \quad p_2 = \langle 4657, 3544 + \sqrt{7} \rangle, \quad p_3 = \langle 12799, 5358 + \sqrt{7} \rangle.$$ 

Again, since $\text{ord}_{p_j}(\Delta) < 12$ for all $j$, $E$ is already given by a globally minimal model, and thus $M_E = 1$. Our algorithm shows that

$$\hat{h}(P) > 0.1415$$

for all non-torsion points $P \in E_{gr}(K)$. This is obtained after a number of refinements as shown in Table 5.2.

Finally, we note that the Tamagawa indices $c_v$ at $v = p_1, p_2, p_3$ are all 1. In addition, since both real embeddings of $E$ have three real roots, we have $c_v = 2$ for

Table 5.2: Illustration of the algorithm for Example 5.1.2

<table>
<thead>
<tr>
<th>Initial $\mu$</th>
<th>Initial $n_{\text{max}}$</th>
<th>Is any $B_n(\mu) &lt; 1$?</th>
<th>Is any intersection empty?</th>
<th>Is $\mu$ a lower bound?</th>
<th>Next $\mu$</th>
<th>Next $n_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>5</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.5000</td>
<td>6</td>
</tr>
<tr>
<td>0.5000</td>
<td>6</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2500</td>
<td>7</td>
</tr>
<tr>
<td>0.2500</td>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1250</td>
<td>8</td>
</tr>
<tr>
<td>0.1250</td>
<td>8</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1875</td>
<td>8</td>
</tr>
<tr>
<td>0.1875</td>
<td>8</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1562</td>
<td>9</td>
</tr>
<tr>
<td>0.1562</td>
<td>9</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1406</td>
<td>10</td>
</tr>
<tr>
<td>0.1406</td>
<td>10</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1484</td>
<td>10</td>
</tr>
<tr>
<td>0.1484</td>
<td>10</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1445</td>
<td>11</td>
</tr>
<tr>
<td>0.1445</td>
<td>11</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1425</td>
<td>12</td>
</tr>
<tr>
<td>0.1425</td>
<td>12</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1416</td>
<td>13</td>
</tr>
<tr>
<td>0.1416</td>
<td>13</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.1411</td>
<td>14</td>
</tr>
<tr>
<td>0.1411</td>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1413</td>
<td>14</td>
</tr>
<tr>
<td>0.1413</td>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1414</td>
<td>14</td>
</tr>
<tr>
<td>0.1414</td>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1415</td>
<td>14</td>
</tr>
<tr>
<td>0.1415</td>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>0.1413</td>
<td>14</td>
</tr>
</tbody>
</table>
both \( v \in M'_K \), and so \( c = 2 \). Hence by Lemma 2.1.1, we have

\[
\hat{h}(P) > 0.1415/2^2 = 0.0353
\]

for all non-torsion points \( P \in E(K) \).

**Example 5.1.3.** Let \( E = E_3 \), where \( E_3 \) is the elliptic curve defined over \( K = \mathbb{Q}(\sqrt{10}) \) given by a Weierstrass equation

\[
E_3 : \quad y^2 = x^3 + 125.
\]

Note that \( K \) has class number 2. Decomposing the discriminant \( \Delta \) of \( E \) into prime ideals, it can be seen that \( \langle \Delta \rangle = \langle -2^43^56 \rangle = p_1^{12}p_2^3p_3^3p_4^8 \), where

\[
\begin{align*}
p_1 &= \langle 5, \sqrt{10} \rangle, & p_2 &= \langle 3, 4 + \sqrt{10} \rangle, & p_3 &= \langle 3, 2 + \sqrt{10} \rangle, & p_4 &= \langle 2, \sqrt{10} \rangle.
\end{align*}
\]

Observe that the model of \( E \) is now minimal everywhere except at \( p_1 \). With the substitutions

\[
x = (\sqrt{10})^2x', \quad y = (\sqrt{10})^3y',
\]

we have a new elliptic curve \( E' : y'^2 = x'^3 + 1/8 \). This time, however, the model of \( E' \) is minimal everywhere except at all prime ideals dividing 2. Thus we let \( E'(p_1) = E' \) and \( E^{(p)} = E \) for any \( p \neq p_1 \) in our computation. Our algorithm then shows that

\[
\hat{h}(P) > 0.2859
\]

for all non-torsion points \( P \in E_{\text{gr}}(K) \). This is based on a number of refinements as shown in Table 5.3.

To derive a lower bound on \( E(K) \), we first note that the Tamagawa indices \( c_v \) at \( v = p_1, p_2, p_3, p_4 \) are 1, 2, 2, and 1 respectively. Moreover, we have \( c_v = 1 \) for both \( v \in M'_K \), since both real embeddings of \( E \) have only one real root. Hence \( c = 2, \)
5.1. Height Bound III: Examples

Table 5.3: Illustration of the algorithm for Example 5.1.3

<table>
<thead>
<tr>
<th>Initial $\mu$</th>
<th>Initial $n_{\text{max}}$</th>
<th>$B_n(\mu) &lt; 1?$</th>
<th>Is any intersection empty?</th>
<th>Is $\mu$ a lower bound?</th>
<th>Next $\mu$</th>
<th>Next $n_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5000</td>
<td>6</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2500</td>
<td>7</td>
</tr>
<tr>
<td>0.2500</td>
<td>7</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.3750</td>
<td>7</td>
</tr>
<tr>
<td>0.3750</td>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.3125</td>
<td>8</td>
</tr>
<tr>
<td>0.3125</td>
<td>8</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2812</td>
<td>9</td>
</tr>
<tr>
<td>0.2812</td>
<td>9</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.2968</td>
<td>9</td>
</tr>
<tr>
<td>0.2968</td>
<td>9</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2890</td>
<td>10</td>
</tr>
<tr>
<td>0.2890</td>
<td>10</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2851</td>
<td>11</td>
</tr>
<tr>
<td>0.2851</td>
<td>11</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.2871</td>
<td>11</td>
</tr>
<tr>
<td>0.2871</td>
<td>11</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2861</td>
<td>12</td>
</tr>
<tr>
<td>0.2861</td>
<td>12</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2856</td>
<td>13</td>
</tr>
<tr>
<td>0.2856</td>
<td>13</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.2858</td>
<td>13</td>
</tr>
<tr>
<td>0.2858</td>
<td>13</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td>0.2860</td>
<td>13</td>
</tr>
<tr>
<td>0.2860</td>
<td>13</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
<td>0.2859</td>
<td>14</td>
</tr>
<tr>
<td>0.2859</td>
<td>14</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and thus by Lemma 2.1.1,

$$\hat{h}(P) > \frac{0.2859}{2^2} = 0.0714$$

for all non-torsion points $P \in E(K)$.

5.1.2 Case II: Number Fields with Complex Embeddings

Next, we will consider the case when our elliptic curves are defined over non-totally real number fields. The reader should refer to Chapter 3 for the relevant notations.

Let $K$ be a non-totally real number field (i.e., one having non-real complex embeddings), and let $E$ be an elliptic curve defined over $K$. It can be seen from Theorem 3.4.1 that, in order to determine a positive lower bound for the canonical height on $E_{gr}(K)$, we may have to compute $\bigcap_{n=1}^{n_{\text{max}}} T_{n}(v)(\sqrt{B_n(\mu)})$ for every complex archimedean place $v \in M_K$, in addition to $\bigcap_{n=1}^{n_{\text{max}}} S_{n}(v)(-B_n(\mu), B_n(\mu))$ for every $v \in M_K$. To obtain $T_{n}(v)(\xi)$, we first need to construct the approximate corresponding region $S_{n}(v)(\xi)$. Assume that for each $v \in M_K$, the associated complex embedding
$E^{(v)}$ of $E$ is of the form

$$E^{(v)} : Y^2 = 4X^3 + AX + B$$

for some $A, B \in \mathbb{C}$. Then it can be seen from Section 3.1.2 that the definition of $S^{(v)}(\xi)$ requires the quantity

$$U_\xi = |w_1|^2 \left( \xi + \frac{b_2}{12} \right),$$

where $b_2$ is an invariant as defined in Chapter 1 for $E^{(v)}$, and $w_1 \in \mathbb{C}$ is one of the two vectors forming a $\mathbb{Z}$-basis for the period lattice $\Lambda$ of $E^{(v)}$, in such a way that $\Lambda = \langle w_1, w_2 \rangle$ and $\tau = w_2/w_1$ satisfies (3.1), i.e.,

$$|\tau| \geq 1, \quad |\Re(\tau)| \leq 1/2, \quad \Im(\tau) \geq \sqrt{3}/2.$$

One can then use Theorem 4.5.3 (together with some linear transformation if necessary) to obtain $\Lambda = \langle w_1, w_2 \rangle$ whose $\tau$ satisfies (3.1).

Furthermore, one can see from Section 3.2.3 that construction of $S^{(v)}(\xi)$ requires a parallelogram $C_0$ containing an elliptic logarithm of a point $P \in E^{(v)}(\mathbb{C})$ with $X(P) = 0$. Although one can use Algorithm 4.6.2 to compute an elliptic logarithm of $P$, it should be noted that this is rarely required in practice, since $C_0$ is normally obtained as one of the parallelograms $C$ satisfying $I(C) \cap [0, U_\xi] \neq \emptyset$.

We will now illustrate our algorithm for elliptic curves defined over non-totally real number fields with the following examples. For the rest of this chapter, we shall let $i = \sqrt{-1}$.

**Example 5.1.4.** Let $E = E_4$, where $E_4$ is the elliptic curve defined over $K = \mathbb{Q}(i)$ given by the Weierstrass equation

$$E_4 : \quad y^2 = x^3 + (91 - 26i)x - (144 + 323i).$$
5.1. Height Bound III: Examples

Table 5.4: Illustration of the algorithm for Example 5.1.4

<table>
<thead>
<tr>
<th>µ</th>
<th>n_{max}</th>
<th>Is any $B_n(\mu) &lt; 1$?</th>
<th>Is any $\bigcap T_n^{(v)}$ empty?</th>
<th>Is $\mu$ a lower bound?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>4</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
</tr>
<tr>
<td>0.10</td>
<td>4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0.15</td>
<td>4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>0.18</td>
<td>4</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The discriminant of $E$ can be factorised into a product of prime ideals as $p_1 p_2 p_3$, where

$$p_1 = \langle 799 + 1124i \rangle, \quad p_2 = \langle 7 - 12i \rangle, \quad p_3 = \langle 1 + i \rangle.$$

Hence the model of $E$ is globally minimal, and so $M_E = 1$. Based on a number of refinements as shown in Table 5.4, our algorithm shows that

$$\hat{h}(P) > 0.18$$

for all non-torsion $P \in E_{\text{gr}}(K)$. Note that in this example we only have to compute $T_n^{(v)}(\sqrt{B_n(\mu)})$ but not $S_n^{(v)}(-B_n(\mu), B_n(\mu))$, since $K$ has no real embedding. In addition, we choose $S^{(v)} = S^{(v,4)}$ for $v \in M^c_K$.

It can be checked that the Tamagawa indices $c_v$ of $E$ at $v = p_1, p_2, p_3$ are all 1. Moreover, we have $c_v = 1$ where $v$ is the only complex archimedean place of $K$. Hence $c = 1$, and by Lemma 2.1.1,

$$\hat{h}(P) > 0.18$$

for all non-torsion $P \in E(K)$. One can verify the above results using our MAGMA code (note that we require $\text{elog.m}$ and every file mentioned in Appendix A.3 onwards) together with the following instructions:

```magma
> // Note that all of these files are required
> Attach("elog.m");
> Attach("alphas.m");
> Attach("heightbound.m");
> Attach("intersect_real.m");
> Attach("intersect_complex.m");
> Attach("interval_arith.m");
> Attach("interval_wp.m");
```
On the other hand, the lower bound on $E_{gr}(K)$ (and also $E(K)$ in this case) obtained by Theorem 2.4.2 is not as good as this one. In this example, we have

$$\alpha_v = 4.715889.$$  

Choose a prime ideal $p$ with $\mathcal{N}(p) > \alpha_v$, say, $p = (5, 2 + i)$, and set $n = e_p = 5$. Then we have $D_E(5) = 3.218876$, which yields the lower bound

$$\mu_0 = \frac{3.218876 - 2 \log(4.715889)}{2 \cdot 5^2} = 2.34 \times 10^{-3}.$$  

Finally, one can verify that the lower bound obtained by Hindry and Silverman [HS88, Theorem 0.3] is

$$\hat{h}(P) \geq 3.0624 \times 10^{-25}$$

for all non-torsion $P \in E(K)$. We leave it to the reader to compare the results.

**Example 5.1.5.** The following elliptic curve is from Cremona’s paper [Cre94, Example 2]. Let $E = E_5$, where $E_5$ is the elliptic curve defined over $K = \mathbb{Q}(i)$ given by the Weierstrass equation

$$E_5 : \quad y^2 + iy = x^3 + (1 - i)x^2 - ix.$$
One can easily observe that \( P_0 = (0,0) \in E(K) \). Let \( \Delta \) be the discriminant of \( E \). Then we have \( \langle \Delta \rangle = p \), where \( p = \langle 13 + 8i \rangle \) is prime. Moreover, we have the Tamagawa index \( c_p = 1 \), and also \( c_v = 1 \) where \( v \) is the only complex archimedean place of \( K \). Hence \( c = 1 \). Using the fact that \( \hat{h}(P_0) = 0.0230 \), we set our initial guess \( \mu \) to be smaller than 0.0230, say, \( \mu = 0.01 \). Our algorithm shows that

\[
B_5(\mu) = 0.7772 < 1.
\]

Thus by Proposition 2.4.1, \( \hat{h}(P) > 0.01 \) for all non-torsion \( P \in E_{gr}(K) \). Since \( c = 1 \), we also have \( \hat{h}(P) > 0.01 \) for all non-torsion \( P \in E(K) \) by Lemma 2.1.1.

**Example 5.1.6.** Let \( K = \mathbb{Q}(\theta) \) where \( \theta \) is a root of the polynomial \( x^3 - 2 \). Let \( E = E_6 \), where \( E_6 \) is the elliptic curve defined over \( K \) given by the Weierstrass equation

\[
E_6 : \quad y^2 = x^3 - (\theta^2 + 3\theta)x + \theta^2.
\]

Let \( \Delta \) be the discriminant of \( E \). The prime ideal factorisation of \( \langle \Delta \rangle \) is \( p_1^{16}p_2 \), where

\[
p_1 = \langle 2, \theta \rangle, \quad p_2 = \langle 390433, 218056 + \theta \rangle.
\]

It can be verified that the model of \( E \) is globally minimal, and so \( M_E = 1 \). Our algorithm shows that

\[
\hat{h}(P) > 0.25
\]

for all non-torsion \( P \in E_{gr}(K) \), which is obtained after a number of refinements as shown in Table 5.5. Recall that if \( \bigcap S_n^{(v)} = \emptyset \) for some \( v \in M_K^r \), then \( \mu \) is a lower bound and so there is no need to compute \( \bigcap T_n^{(v)} \) for any \( v \in M_K^r \).

Finally, we note that the Tamagawa indices \( c_v \) at \( v = p_1, p_2 \) are 2 and 1 respectively. Moreover, since \( E \) has only one real embedding, say, \( E^{(v_1)} \) with three real roots, we have \( c_{v_1} = 2 \). Denote the only complex archimedean place of \( K \) by \( v_2 \).
Table 5.5: Illustration of the algorithm for Example 5.1.6

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>$n_{\max}$</th>
<th>Is any $B_n(\mu) &lt; 1$?</th>
<th>Is any $\bigcap S_n(v)$ empty?</th>
<th>Is any $\bigcap T_n(v)$ empty?</th>
<th>Is $\mu$ a lower bound?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>3</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Fail</td>
</tr>
<tr>
<td>0.20</td>
<td>3</td>
<td>No</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
</tr>
<tr>
<td>0.30</td>
<td>3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Fail</td>
</tr>
<tr>
<td>0.25</td>
<td>3</td>
<td>No</td>
<td>Yes</td>
<td>Skipped</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Then again $c_{p_2} = 1$, and so $c = \text{lcm}\{1, 2\} = 2$. Thus by Lemma 2.1.1, we have

$$\hat{h}(P) > 0.25/2^2 = 0.0625$$

for all non-torsion $P \in E(K)$. Note that we have obtained no additional information from the complex place in this specific example; however, there is no reason to suppose that this would be the case in general.

In the next section, we will explain how to use a lower bound for the canonical height to derive Mordell–Weil bases for elliptic curves defined over number fields. This method will be illustrated when we revisit all the examples recently shown.

## 5.2 Mordell–Weil Bases

Computing Mordell–Weil bases for elliptic curves over number fields is one of the most difficult computations in the arithmetic of elliptic curves, and so far there is no known procedure which can determine such a basis in general. In this section, we will illustrate an application of a lower bound for the canonical height in assisting such computation. For more background on this section, see Section 1.2.2 or [Cre97, Section 3.5].

Let $E$ be an elliptic curve defined over a number field $K$. Recall from Section 1.2.2 that, given some non-torsion points $P_1, \ldots, P_r \in E(K)$ whose images in $E(K)/E_{\text{tors}}(K)$ generate a subgroup of finite index of $E(K)/E_{\text{tors}}(K)$, it is possible to “saturate” these points (which are normally obtained by performing an
$m$-descent for some $m \geq 2$) to obtain a full Mordell–Weil basis for $E(K)$. The saturation process consists of the following steps:

1. Determine an upper bound $\ell$ for the index $n = [E(K)/E_{\text{tors}}(K) : \langle P_1, \ldots, P_r \rangle]$ using the geometry of numbers (Theorem 1.2.1), which then requires a positive lower bound for the canonical height on $E(K)$ obtained by Theorem 3.4.1.

2. For each prime $p \leq \ell$, determine whether $p \mid n$, or equivalently, whether there exists $a_1, \ldots, a_r \in \mathbb{Z}$, not all divisible by $p$, such that

$$\sum_{j=1}^r a_j P_j = pQ$$

(5.1)

for some $Q \in E(K)$. Without loss of generality, we can assume that $|a_j| \leq p/2$.

3. If there exists a solution to (5.1), let $a_i$ be the minimal non-zero coefficient (in absolute value). If $a_i = \pm 1$, then we can simply replace $P_i$ by $Q$; otherwise, we find a coefficient $a_j$ not divisible by $a_i$. Write $a_j = a_i q + b$ with $0 < b < |a_i|$. Observe that

$$a_i P_i + a_j P_j = a_i (P_i + qP_j) + bP_j.$$ 

This then allows us to replace the generator $P_i$ by $P_i + qP_j$, replace $a_j$ by $b$, and replace $i$ by $j$. This time, the index of the sublattice generated by the new $P_1, \ldots, P_r$ in $E(K)$ will be at most $\ell |a_i|/p$.

4. Repeat the above steps until the index $n$ is not divisible by any primes. The final set $\{P_1, \ldots, P_r\}$ will be a Mordell–Weil basis for $E(K)$ modulo torsion.

Nevertheless, the upper bound $\ell$ obtained by Theorem 1.2.1 can be very large even though the points $P_1, \ldots, P_r$ may already form a Mordell–Weil basis, and so there can be too many primes $p$ to consider. Fortunately, it is possible to quickly eliminate some of $p$ from our consideration before we actually have to solve (5.1).
5.2.1 Sieving Procedure

This procedure was initially explained in [Sik95, Section 4], and has been described in full details later by Prickett in his thesis [Pri04]. For convenience, we shall give a summary here.

For a given prime \( p \leq \ell \), let \( P_r + 1, \ldots, P_r + s \) be a basis for \( E_{\text{tors}}(K)/pE_{\text{tors}}(K) \). Our aim now is to determine the set

\[
V_p = \{ a \in \mathbb{F}_p^{r+s} : \sum_{j=1}^{r+s} a_j P_j \in pE(K) \}.
\]

It can be seen easily that the index \( n \) is divisible by \( p \) if and only if \( V_p \neq \{0\} \). We choose a prime ideal \( p \) such that \( E \) is minimal at \( p \), and satisfies the following:

1. \( E \) has good reduction at \( p \);
2. \( #E(k_p) \) is divisible by \( p \), but not \( p^2 \) (Here, \( k_p \) is the residue class field).

Write \( #E(k_p) = lp \). Clearly, \( p \nmid l \) due to the choice of \( p \).

Let \( \pi \) be a uniformiser at \( p \). Now for each \( P_j \), we compute \( P'_j \equiv lP_j \pmod{\pi} \) for all \( j = 1, \ldots, r+s \). If \( P'_j \equiv O \pmod{\pi} \) for every \( j \), then this yields no information, and so we should start with a new prime \( p \) satisfying the above conditions. Otherwise, there exists a point, say, \( P'_1 \equiv O \pmod{\pi} \). The condition (2) then ensures that \( lE(k_p) \) is a cyclic group of order \( p \). Thus for all \( j = 1, \ldots, r+s \), we have

\[
P'_j \equiv m_j P'_1 \pmod{\pi}
\]

for some \( m_j \in \mathbb{Z} \). It then follows that if \( a \in V_p \), then \( a \) satisfies the relation

\[
\sum_{j=1}^{r+s} m_j a_j \equiv 0 \pmod{p}.
\]

By solving all of these \( r+s \) relations over \( \mathbb{F}_p \), we eventually reduce \( V_p \) into a smaller set. In particular, if such \( r+s \) relations are independent, then \( V_p = \{0\} \), and so
the index \( n \) is not divisible by \( p \).

It should be noted, however, that this method may sometimes fail to prove that a point is \( p \)-saturated even though it actually is. For more details on modification of this method, see [Pri04].

### 5.2.2 Examples Revisited

We will now revisit all the elliptic curves shown in Section 5.1 and illustrate how to obtain their Mordell–Weil bases using a lower bound for the canonical height together with the sieving procedure.

**Example 5.2.1.** Let \( K = \mathbb{Q}(\sqrt{2}) \) and let \( E = E_1 \) as defined earlier. In Example 5.1.1, we have obtained from our algorithm that \( \hat{h}(P) > 0.0150 \) for all non-torsion \( P \in E(K) \). In fact, one can check that the torsion subgroup of \( E(K) \) is trivial. We now wish to determine whether \( E(K) = \langle P_1 \rangle \), where

\[
P_1 = (1, 1 + \sqrt{2}) \in E(K).
\]

Using MAGMA, we know that \( \hat{h}(P_1) = 0.5033 \), and the rank of \( E(K) \) is at most 1. Hence \( E(K) \) has rank 1. By Theorem 1.2.1, we have

\[
n = [E(K) : \langle P_1 \rangle] \leq \sqrt{\frac{0.5033}{0.0150}} = 5.7927.
\]

It therefore remains to check whether the index \( n \) is divisible by any primes \( p \leq 5 \). Note that this upper bound can be computed using our MAGMA function UpperBound4Index() in the file heightbound.m (see Appendix A.3) as follows:

```plaintext
> Attach("heightbound.m");
> // Define elliptic curve E
> <x> := PolynomialRing(Integers());
> K<a> := NumberField(x^2-2);
> E := EllipticCurve([1, 1+2*a]);
> P1 := E! [1,1+a];
> // Use 0.0150 as a lower bound for the canonical height on E(K)
> UpperBound4Index([P1], 0.0150);
5.79270969603967816405459250
```
In this example, one can easily check using division polynomial that $P_1 \notin pE(K)$ for all $p = 3, 5$, so $n$ must be 1, 2, or 4. A simple observation also shows that $P_1 = 2P$ where

$$P = (1 - \sqrt{2}, 1 - 2\sqrt{2}) \in E(K),$$

and $P \neq 2Q$ for any $Q \in E(K)$. Hence we have $[E(K) : \langle P \rangle] = 1$, i.e., $P$ generates $E(K)$.

**Example 5.2.2.** Let $K = \mathbb{Q}(\sqrt{7})$ and let $E = E_2$ as defined earlier. In Example 5.1.2, we have shown that $\hat{h}(P) > 0.0353$ for all non-torsion $P \in E(K)$. Again, one can check that the torsion subgroup of $E(K)$ is trivial, and the points

$$P_1 = (0, 0), \quad P_2 = (1, \sqrt{7})$$

are in $E(K)$. We wish to show that $E(K) = \langle P_1, P_2 \rangle$.

Using **MAGMA**, one can see that $\hat{h}(P_1) = 0.8051, \quad \hat{h}(P_2) = 1.4957$.

By computing the height pairing matrix, we have

$$R(P_1, P_2) = \det \begin{pmatrix} \langle P_1, P_1 \rangle & \langle P_1, P_2 \rangle \\ \langle P_2, P_1 \rangle & \langle P_2, P_2 \rangle \end{pmatrix} = \begin{vmatrix} 0.8051 & -0.1941 \\ -0.1941 & 1.4957 \end{vmatrix} = 1.1665 \neq 0.$$

Hence $P_1$ and $P_2$ are independent. From **MAGMA**, we also know that the rank of $E(K)$ is at most 2. Hence $E(K)$ has rank 2. By Theorem 1.2.1, we finally have

$$n = [E(K) : \langle P_1, P_2 \rangle] \leq \frac{2\sqrt{1.1665}}{0.0353 \cdot \sqrt{3}} = 35.2450.$$

Thus we shall apply the sieving procedure for all primes $p \leq 31$.

Using a similar argument as in the previous example, we deduce from sieving
As suggested in [Sik95, Section 4.2], it is perhaps more practical to solve (5.1) by
inefficient due to the difficulty in finding all roots of a polynomial of degree
show that

<table>
<thead>
<tr>
<th>p</th>
<th>p</th>
<th>#E(k_p)</th>
<th>t</th>
<th>(m_1, m_2)</th>
<th>(a_1, a_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(17)</td>
<td>318</td>
<td>159</td>
<td>(1, 2)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>3</td>
<td>(3, 1 + \sqrt{7}) (5)</td>
<td>6</td>
<td>2</td>
<td>(1, 2)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>5</td>
<td>(2, 1 + \sqrt{7}) (11)</td>
<td>5</td>
<td>1</td>
<td>(1, 3)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>7</td>
<td>(19, 8 + \sqrt{7}) (29, 6 + \sqrt{7})</td>
<td>21</td>
<td>3</td>
<td>(1, 7)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>11</td>
<td>(47, 30 + \sqrt{7}) (113, 32 + \sqrt{7})</td>
<td>44</td>
<td>4</td>
<td>(1, 5)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>13</td>
<td>(\sqrt{7}) (103, 78 + \sqrt{7})</td>
<td>13</td>
<td>1</td>
<td>(1, 6)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>17</td>
<td>(29, 23 + \sqrt{7}) (31, 21 + \sqrt{7}) (37, 9 + \sqrt{7})</td>
<td>34</td>
<td>2</td>
<td>(1, 2)</td>
<td>(0, 0), (1, 8), (2, -1), (3, 7), (4, -2), (5, 6), (6, -3), (7, 5), (8, -4)</td>
</tr>
<tr>
<td>19</td>
<td>(31, 21 + \sqrt{7}) (37, 9 + \sqrt{7})</td>
<td>38</td>
<td>2</td>
<td>(1, 11)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>23</td>
<td>(11) (337, 119 + \sqrt{7})</td>
<td>115</td>
<td>5</td>
<td>(1, 4)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>29</td>
<td>(103, 25 + \sqrt{7}) (149, 56 + \sqrt{7})</td>
<td>116</td>
<td>4</td>
<td>(1, 8)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>31</td>
<td>(137, 12 + \sqrt{7}) (139, 110 + \sqrt{7})</td>
<td>155</td>
<td>5</td>
<td>(1, 25)</td>
<td>(0, 0)</td>
</tr>
</tbody>
</table>

that V_p = \{0\} for every p \leq 31 except for p = 2, 17. For each p \leq 31, the choice of p and their corresponding (m_1, m_2) is shown in Table 5.6. For p = 2, the sieving method yields a possible set for V_2. To be precise, we have to check whether (0, 1) \in V_2, i.e. if there exists a point Q \in E(K) such that

P_2 = (1, \sqrt{7}) = 2Q.

Using 2-division polynomial, it turns out that there is no such x(Q) \in K which satisfies the polynomial. Hence Q does not exist, and so V_2 = \{0\}.

It still remains to find V_{17}. In this case, it suffices to check only one pair of (a_1, a_2), say, (a_1, a_2) = (1, 8). Again, by division polynomial, one can eventually show that P_1 + 8P_2 \neq 17Q for any Q \in E(K). Thus n = 1, i.e., E(K) = (P_1, P_2).

For large p, note that the p-division polynomial technique may become very inefficient due to the difficulty in finding all roots of a polynomial of degree p^2. As suggested in [Sik95, Section 4.2], it is perhaps more practical to solve (5.1) by
computing elliptic logarithms of all \( P(v) \in E(v) \), where \( E(v) \) is the embedding of \( E \) associated to an archimedean place \( v \in M_K \), and \( P(v) \) is the image of \( P \) on \( E(v) \).

**Example 5.2.3.** Let \( K = \mathbb{Q} (\sqrt{10}) \) and let \( E = E_3 \) as defined earlier. Note that \( E_{\text{tors}}(K) \) is a cyclic group of order 2 generated by the point \( T = (-5,0) \in E(K) \).

We have shown in Example 5.1.3 that \( \hat{h}(P) > 0.0714 \) for all non-torsion \( P \in E(K) \).

Let \( P_1 = (5, 5\sqrt{10}) \in E(K) \). From MAGMA, we know that \( \hat{h}(P_1) = 0.6532 \), and the rank of \( E(K) \) is at most 1. Hence \( E(K) \) has rank 1. By Theorem 1.2.1, we have

\[
n = [E(K)/E_{\text{tors}}(K) : \langle P_1 \rangle] \leq \sqrt{0.6532/0.0714} = 3.0229.
\]

In fact, we verify that \( P_1 \notin pE(K) \) for \( p = 2, 3 \). Hence \( n = 1 \), and so

\[
E(K) = \langle T \rangle \times \langle P_1 \rangle \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}.
\]

**Example 5.2.4.** Let \( K = \mathbb{Q}(i) \) and let \( E = E_4 \) as defined earlier. One can check that the torsion subgroup of \( E(K) \) is trivial. Let

\[
P_1 = (1 + 5i, 2 - i), \quad P_2 = \left( \frac{-32 - 53i}{2}, \frac{-663 + 49i}{4} \right).
\]

Then we have \( P_1, P_2 \in E(K) \). We will show that \( E(K) = \langle P_1, P_2 \rangle \).

In Example 5.1.4, we have obtained from our algorithm that \( \hat{h}(P) > 0.18 \) for all non-torsion \( P \in E(K) \). Moreover, one can check using MAGMA that

\[
\hat{h}(P_1) = 1.2326, \quad \hat{h}(P_2) = 4.2894,
\]

and the rank of \( E(K) \) is at most 2. Computing the height pairing matrix, one can see that

\[
R(P_1, P_2) = \det((P_j, P_k))_{1 \leq j, k \leq 2} = 3.6050 \neq 0,
\]

i.e., \( P_1 \) and \( P_2 \) are independent. Thus \( E(K) \) has rank 2. It then follows from
5.2. Mordell–Weil Bases

Table 5.7: Sieving procedure for the elliptic curve $E_4$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\langle \alpha \rangle$</th>
<th>$#E(kp)$</th>
<th>$i$</th>
<th>$(m_1, m_2)$</th>
<th>$(a_1, a_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\langle 3 \rangle$</td>
<td>10</td>
<td>5</td>
<td>(1, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td></td>
<td>$\langle 53, 23+i \rangle$</td>
<td>54</td>
<td>27</td>
<td>(1, 1)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\langle 13, 8+i \rangle$</td>
<td>21</td>
<td>7</td>
<td>(1, 2)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td></td>
<td>$\langle 17, 4+i \rangle$</td>
<td>24</td>
<td>8</td>
<td>(1, 1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\langle 3 \rangle$</td>
<td>10</td>
<td>2</td>
<td>(1, 2)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td></td>
<td>$\langle 5, 2+i \rangle$</td>
<td>5</td>
<td>1</td>
<td>(1, 3)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$\langle 13, 8+i \rangle$</td>
<td>21</td>
<td>3</td>
<td>(0, 1)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td></td>
<td>$\langle 73, 46+i \rangle$</td>
<td>63</td>
<td>9</td>
<td>(1, 4)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$\langle 7 \rangle$</td>
<td>44</td>
<td>4</td>
<td>(1, 2)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td></td>
<td>$\langle 109, 33+i \rangle$</td>
<td>110</td>
<td>10</td>
<td>(1, 7)</td>
<td></td>
</tr>
</tbody>
</table>

Theorem 1.2.1 that

$$n = [E(K) : \langle P_1, P_2 \rangle] \leq \frac{2 \sqrt{3.6050}}{\sqrt{3} \cdot 0.18} = 12.1801.$$  

The sieving procedure then shows that $n$ is not divisible by any primes $p \leq 11$; see Table 5.7. Therefore $n = 1$, and so $E(K) = \langle P_1, P_2 \rangle$. In fact, it can be verified that $P_1$ has the smallest canonical height among non-torsion $P \in E(K)$, with $\hat{h}(P_1) = 1.2326$. Compare this with our lower bound $\hat{h}(P) > 0.18$.

On the other hand, if we had used the lower bound obtained by Theorem 2.4.2 (i.e., $\hat{h}(P) > 2.34 \times 10^{-3}$ for all non-torsion $P \in E(K)$), then it would follow from Theorem 1.2.1 that $n \leq 936$. Finally, we note that the lower bound obtained by Hindry and Silverman [HS88, Theorem 0.3] (i.e., $\hat{h}(P) > 3.0624 \times 10^{-25}$ for all non-torsion $P \in E(K)$) would lead to $n \leq 7.1591 \times 10^{24}$, which would make it considerably harder to verify that $n = 1$.

**Example 5.2.5.** Let $K = \mathbb{Q}(i)$ and let $E = E_5$ as defined earlier. We have already shown in Example 5.1.5 that $\hat{h}(P) > 0.01$ for all non-torsion $P \in E(K)$. One can check that $E$ has trivial torsion subgroup and the point $P_0 = (0, 0) \in E(K)$. In Cremona’s paper [Cre94, Example 2], it has been asked whether $E(K) = \langle P_0 \rangle$. We will show that this is the case.

Using MAGMA, one can check that the rank of $E(K)$ is at most 1. Since $P_0$ is non-torsion, the rank of $E(K)$ is also at least 1. Hence $E(K)$ has rank 1. Theorem
1.2.1 then implies that

\[ n = [E(K) : \langle P_0 \rangle] \leq \sqrt{0.0230/0.01} = 1.5173 < 2, \]

i.e., \( n = 1 \). Hence \( E(K) = \langle P_0 \rangle \).

Example 5.2.6. Let \( K = \mathbb{Q}(\theta) \) where \( \theta \) is a root of the polynomial \( x^3 - 2 \), and let \( E = E_6 \) as defined earlier. One can verify that the torsion subgroup of \( E(K) \) is trivial, and

\[ P_1 = (0, \theta), \quad P_2 = (1 + \theta, 1), \quad P_3 = (3 - 9\theta + 7\theta^2, 31 + 23\theta - 36\theta^2), \]

are in \( E(K) \). We wish to confirm that \( E(K) = \langle P_1, P_2, P_3 \rangle \).

In Example 5.1.6, we have shown that \( \hat{h}(P) > 0.0625 \) for all non-torsion \( P \in E(K) \). In addition, one can check using MAGMA that

\[ \hat{h}(P_1) = 0.6303, \quad \hat{h}(P_2) = 0.8045, \quad \hat{h}(P_3) = 2.4430, \]

and the rank of \( E(K) \) is at most 3. Computing the height pairing matrix, we have

\[ R(P_1, P_2, P_3) = \det(\langle P_j, P_k \rangle)_{1 \leq j, k \leq 3} = 0.6263 \neq 0, \]

i.e., \( P_1, P_2, P_3 \) are independent. Thus \( E(K) \) does have rank 3. Then by the geometry of numbers (Theorem 1.2.1), we obtain

\[ n = [E(K) : \langle P_1, P_2, P_3 \rangle] \leq \sqrt{2(0.6263)/(\sqrt{0.0625})^3} = 71.6300. \]

Using the sieving procedure, we can eventually show (details omitted) that \( n \) is not divisible by any primes \( p \leq 71 \). Therefore \( n = 1 \), and so

\[ E(K) = \langle P_1, P_2, P_3 \rangle. \]
It can be verified that $P_1$ has the smallest canonical height among non-torsion $P \in E(K)$, with $\hat{h}(P_1) = 0.6303$. Compare this with our lower bound $\hat{h}(P) > 0.0625$.

### 5.2.3 Comparison with a Searching Points Method

Let $E$ be an elliptic curve over a number field $K$. As suggested by a referee of [Tho10], we shall briefly describe an alternative way, as illustrated in [Sil90], to derive a Mordell–Weil basis for $E(K)$, and finally compare it with our method.

Suppose we can find a set of points $\{P_1, \ldots, P_r\} \subset E(K)$ which bijects to a basis for the group $E(K)/mE(K)$ for some $m \geq 2$. Let

$$C_1 = \max\{\hat{h}(Q) : Q = n_1P_1 + \cdots + n_rP_r, \text{ with } 0 \leq n_1, \ldots, n_r < m\}.$$

Then [Sil90, Proposition 7.2] says that the set $S = \{R \in E(K) : \hat{h}(R) \leq C_1\}$ generates $E(K)$. Using a result of [CPS06] or [Sil90], one can compute a constant $C_2$ satisfying $h(P) - \hat{h}(P) \leq C_2$ for all $P \in E(K)$, where $h(P)$ denotes the Weil height of the $x$-coordinate of $P$. It then follows that

$$h(R) \leq C_1 + C_2$$

for all $R \in S$. This, in principle, will allow one to search for $R$. If there exists $R \in S$ which is not a linear combination of $P_1, \ldots, P_r$, then we can replace some $P_j$ by the linear combination of $R$. Repeating this process until no such $R$ exists, the final set of $P_1, \ldots, P_r$ will eventually be a Mordell–Weil basis for $E(K)$.

The difficulty of this method lies in searching for points. Even though the $x$-coordinates have bounded height, this can be a non-trivial task especially if $[K : \mathbb{Q}]$ is large. In contrast, our method completely circumvents this problem. If $P_1, \ldots, P_r$ do not yet form a Mordell–Weil basis, we can use the sieving procedure to derive a new set of candidates. This process, which can be done more quickly than searching for points, however requires an upper bound for the index
[E(K)/E_{tors}(K) : \langle P_1, \ldots, P_r \rangle], which in turn requires a lower bound for the canonical height on E(K).

5.3 Integral Points on Elliptic Curves

In this section, we will explain how Mordell–Weil bases, periods of elliptic curves, and complex elliptic logarithms can assist in finding integral points on elliptic curves over number fields. The method to be described here is a summary of a paper by Smart and Stephens [SS97] with some modifications. Some illustrative examples, which are computed by the algorithm based on this method (see Appendix A.2 for its MAGMA source code), will be also given at the end of this section.

5.3.1 Introduction

**Definition.** Let E be an elliptic curve defined over a number field K. We say that a point \( P = (x, y) \in E(K) \) is an integral point if both \( x, y \in \mathcal{O}_K \), where \( \mathcal{O}_K \) is the ring of integers of K.

If the rank of \( E(K) \) is non-zero, then we have already seen that there are infinitely many points in \( E(K) \). However, this does not imply that the set of all integral points is also infinite. In fact, it has been proved by Siegel [Sie26] that there are only finitely many integral points in \( E(K) \).

Suppose \( \{P_1, \ldots, P_r\} \) is a Mordell–Weil basis for \( E(K) \). Then every point \( P \in E(K) \) can be written as

\[
P = q_1P_1 + \cdots + q_rP_r + T, \tag{5.2}
\]

for some \( T \in E_{\text{tors}}(K) \) and \( q_1, \ldots, q_r \in \mathbb{Z} \). If \( P \) is an integral point, then Siegel’s Theorem implies that there exists an upper bound on each coefficient \( |q_j| \). Let

\[
Q = \max_{1 \leq j \leq r} \{|q_j|\}.
\]
Provided that an upper bound for $Q$ is known, then we can, in principle, obtain all integral points in $E(K)$ simply by brute-force search.

Given a Mordell–Weil basis for $E(K)$, Smart and Stephens [SS97] have proposed a method for computing an upper bound for $Q$. This method can be roughly described as follows. For each archimedean place $v \in M^r_K \cup M^c_K$, we let $E^{(v)}$ be the associated (real or complex) embedding of $E$. Then on each $E^{(v)}$, the method will initially compute a rather large bound $Q_v$, and then repeatedly apply LLL basis reduction [LLL82] to reduce $Q_v$ as much as possible. Finally, we take the maximum among all $Q_v$ to be an upper bound for $Q$.

### 5.3.2 Initial Bounds

For each $v \in M^r_K \cup M^c_K$, let $E^{(v)}$ be the associated (real or complex) embedding of $E$. Without loss of generality, we can assume that $E^{(v)}$ is of the form

$$E^{(v)} : \quad Y^2_v = 4X^3_v + A_v X_v + B_v$$

for some $A_v, B_v \in \mathbb{C}$, depending on $E$ and $v$. Recall from Section 1.3 that there exists an isomorphism (of complex analytic Lie groups) $\mathbb{C}/\Lambda_v \rightarrow E^{(v)}(\mathbb{C})$ for some lattice $\Lambda_v$, given by the map

$$z \pmod{\Lambda_v} \mapsto (\wp_{\Lambda_v}(z), \wp'_{\Lambda_v}(z))$$

$$0 \pmod{\Lambda_v} \mapsto O.$$  

We will denote the inverse of this map, the elliptic logarithm on $E^{(v)}$, by $\varphi_v$. For an integral point $P \in E(K)$, let $P^{(v)}$ be its associated image on $E^{(v)}$. Our aim is to estimate both lower and upper bound for $|\varphi_v(P^{(v)})|$. Combining both bounds then yields an initial upper bound for $Q$.

The following lemma gives an upper bound for $|\varphi_v(P^{(v)})|$; see [SS97] for the detailed proof.
Lemma 5.3.1. If $Q \geq Q_0$, then $|\varphi_v(P_v)| \leq c_9 \exp(-c_{10}Q^2)$, for some explicitly computable constants $^2Q_0, c_9$ and $c_{10}$ depending only on $E$ and $v$.

Note that the constants $Q_0, c_9, c_{10}$ are defined by a number of intermediate constants, which are well explained in [SS97] and are defined accordingly in our algorithm (see Appendix A.2 for the code). For now we mention that the following information is required in order to define these constants.

1. **A Mordell–Weil basis for $E(K)$.** This is essential for computing height pairing matrix, whose least eigenvalue is required for defining $Q_0$ and $c_{10}$.

2. **The period lattice $\Lambda_v$ of $E^{(v)}$.** This can be obtained by Theorem 4.5.3, and is required for the constant $c_9$. Without loss of generality, we can assume that $\Lambda_v = \langle w_1, w_2 \rangle$ with $\tau = w_2/w_1$ satisfying (3.1), i.e.,

\[
|\tau| \geq 1, \quad |\Re(\tau)| \leq 1/2, \quad \Im(\tau) \geq \sqrt{3}/2.
\]

3. **Difference between the Weil (logarithmic) height and the canonical height on $E(K)$.** This result can be found in [Sil90] and more recently in [CPS06], which is required for computing $Q_0$ and $c_9$.

Since $\varphi_v : E^{(v)}(\mathbb{C}) \to \Lambda_v$ is a group isomorphism, it then follows from (5.2) that

\[
\varphi_v(P^{(v)}) \equiv \varphi_v(T^{(v)}) + \sum_{j=1}^r q_j \varphi_v(P_j^{(v)}) \pmod{\Lambda_v}
\]

\[
= \varphi_v(T^{(v)}) + \sum_{j=1}^r q_j \varphi_v(P_j^{(v)}) + m_1 w_1 + m_2 w_2 \quad (5.3)
\]

for some $m_1, m_2 \in \mathbb{Z}$. Let $t = \text{ord}(T)$. Then we have $\varphi_v(T^{(v)}) = (n_1 w_1 + n_2 w_2)/t$, for some integers $0 \leq n_1, n_2 < t$. Together with (5.3), this leads to

\[
t\varphi_v(P^{(v)}) = \sum_{j=1}^r q_j t \varphi_v(P_j^{(v)}) + (m_1 t + n_1) w_1 + (m_2 t + n_2) w_2.
\]

\(^2\text{All constants } c_j \text{ are indexed so that they match the ones defined in [SS97].}\)
In our notations, the following theorem yields a lower bound for this linear form in elliptic logarithms.

**Theorem 5.3.2 ([Dav95, Théorème 2.1]).** Let \( Q' = \max\{Qt, |m_1 t + n_1|, |m_2 t + n_2|\} \).

There exist explicitly computable constants\(^3 d_8, d_9, d_{10} \) and \( h_E \), such that, if \( Q' > \exp(d_8) \), then

\[
\log |t\varphi_v(P^{(v)})| > -d_{10}(\log Q' + \log([K : \mathbb{Q}]d_9))(\log \log Q' + h_E + \log([K : \mathbb{Q}]d_9))^{r+3}.
\]

For now we note that determining \( d_8, d_9, d_{10} \) and \( h_E \) requires elliptic logarithms \( \varphi_v(P^{(v)}_j) \) for all \( 1 \leq j \leq r \), which can be computed using Algorithm 4.6.2. For more details on how to compute these constants, see [Sma98, Appendix A]; these are also defined accordingly in our algorithm shown in Appendix A.2.

In order to make \( t\varphi_v(P^{(v)}) \) lie in the fundamental parallelogram spanned by \( w_1, w_2 \), we require that

\[
|m_j t + n_j| < t \sum_{k=1}^r |q_k| = rQt, \quad \text{for } j = 1, 2.
\]

Thus \( Q' < rQt \leq rQe_{\text{tors}} \), where \( e_{\text{tors}} \) is the exponent of \( E_{\text{tors}}(K) \). Observe that \( t | e_{\text{tors}} \). Combining Lemma 5.3.1 and Theorem 5.3.2, the following proposition is immediate.

**Proposition 5.3.3 (Principal Inequality).** If \( Q > \max\{Q_0, \exp(d_8)\} \), then

\[
c_{10}Q^2 < d_{10}(\log(rQe_{\text{tors}})+\log([K : \mathbb{Q}]d_9))(\log \log(rQe_{\text{tors}})+h_E+\log([K : \mathbb{Q}]d_9))^{r+3}
\]

\[
+ \log(e_{\text{tors}}c_9).
\]

This proposition therefore gives us an initial upper bound for \( Q \), that is, either the one obtained from the above inequality or \( \max\{Q_0, \exp(d_8)\} \), whichever is greater. Denote this initial upper bound by \( Q_v \).

\(^3\)The notations are as defined in [Sma98, Appendix A], with \( c_j \) being replaced by \( d_j \).
5.3.3 Bound Reduction

In general, the initial upper bound $Q_v$ we just obtained is considerably too large for a practical use. The next step in Smart and Stephens’ method is therefore to reduce $Q_v$ as much as possible. This can be achieved using an application of LLL basis reduction [LLL82].

To use an LLL basis reduction, we first choose a constant $C \approx Q_v^{r+2}$.

Consider the $r+2$-dimensional lattice generated by the columns of the matrix

$$L = \begin{pmatrix}
1 & \ldots & 0 & 0 & 0 \\
0 & \ldots & 0 & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \vdots \\
0 & \ldots & 1 & 0 & 0 \\
[C\Re(\varphi_v(P_1^{(v)}))] & \ldots & [C\Re(\varphi_v(P_r^{(v)}))] & [C\Re(w_1)] & [C\Re(w_2)] \\
[C\Im(\varphi_v(P_1^{(v)}))] & \ldots & [C\Im(\varphi_v(P_r^{(v)}))] & [C\Im(w_1)] & [C\Im(w_2)]
\end{pmatrix}$$

(see [Sma98, p. 84]), where $\lfloor x \rfloor$ is the rounding towards 0, i.e.,

$$\lfloor x \rfloor = \begin{cases} 
\lfloor x \rfloor & \text{if } x \geq 0, \\
\lceil x \rceil & \text{if } x < 0.
\end{cases}$$

In general, $C$ can be very large, hence one needs to compute the periods $w_1, w_2$ and complex elliptic logarithms of the points $P_1^{(v)}, \ldots, P_r^{(v)}$ to a very high degree of precision in order to ensure that their integer parts are correct.

Next, we let

$$\ell = L \begin{pmatrix} t_{q_1} \\ \vdots \\ t_{q_r} \\ m_1 t + n_1 \\ m_2 t + n_2 \end{pmatrix} = \begin{pmatrix} t_{q_1} \\ \vdots \\ t_{q_r} \\ \lambda_1 \\ \lambda_2 \end{pmatrix},$$
where

\[
\lambda_1 = (m_1 t + n_1)[C\Re(w_1)] + (m_2 t + n_2)[C\Re(w_2)] + \sum_{j=1}^{r} t q_j [C\Re(\varphi_v(P_j^{(v)})]) ,
\]

\[
\lambda_2 = (m_1 t + n_1)[C\Im(w_1)] + (m_2 t + n_2)[C\Im(w_2)] + \sum_{j=1}^{r} t q_j [C\Im(\varphi_v(P_j^{(v)})]) .
\]

After applying LLL algorithm [LLL82, Proposition (1.11)], one shall obtain another basis \( \{b_1, \ldots, b_{r+2}\} \) which spans the same lattice as \( \mathcal{L} \) does. Since \( \ell \neq 0 \), the LLL algorithm assures that

\[
\|\ell\|^2 \geq 2^{-r-1}\|b_1\|^2 .
\]

Recall that \( |q_j| \leq Q_v \) for all \( 1 \leq j \leq r \), and \( |m_j t + n_j| < r Q_v t \leq r Q_v e_{\text{tors}} \) for \( j = 1, 2 \). This leads to the following proposition.

**Proposition 5.3.4.** If \( \sqrt{2^{-r-1}\|b_1\|^2 - r Q_v^2 e_{\text{tors}}^2} > 3 r Q_v e_{\text{tors}} / \sqrt{2} \), then

\[
Q_v^2 \leq \frac{1}{c_{10}} \left( \log(C c_9 e_{\text{tors}}) - \log \left( \sqrt{2^{-r-1}\|b_1\|^2 - r Q_v^2 e_{\text{tors}}^2} - \frac{3}{\sqrt{2}} r Q_v e_{\text{tors}} \right) \right) . \tag{5.4}
\]

**Proof.** By definition of \([\cdot]\), we have

\[
|\lambda_1 + i \lambda_2 - C t \varphi_v(P^{(v)})| \leq \sqrt{2} \left( \sum_{j=1}^{r} t |q_j|/2 + \frac{|m_1 t + n_1|}{2} + \frac{|m_2 t + n_2|}{2} \right) \leq \frac{3}{\sqrt{2}} r Q_v e_{\text{tors}} .
\]

Thus

\[
|\lambda_1 + i \lambda_2| \leq \frac{3}{\sqrt{2}} r Q_v e_{\text{tors}} + C |t \varphi_v(P^{(v)})| \leq \frac{3}{\sqrt{2}} r Q_v e_{\text{tors}} + C c_9 \exp(-c_{10} Q_v^2) .
\]
Now we have
\[
2^{-r-1} \|b_1\|^2 \leq \|\ell\|^2 = t^2 \sum_{j=1}^{r} q_j^2 + |\lambda_1 + i\lambda_2|^2 \\
\leq rQ_v^2 e_{\text{tors}}^2 + \left( \frac{3}{\sqrt{2}} rQ_v e_{\text{tors}} + Cc_9 \exp(-c_{10} Q_v^2) \right)^2.
\]
Hence
\[
\sqrt{2^{-r-1}} \|b_1\|^2 - rQ_v^2 e_{\text{tors}}^2 - \frac{3}{\sqrt{2}} rQ_v e_{\text{tors}} \leq Cc_9 \exp(-c_{10} Q_v^2),
\]
and (5.4) then follows after taking logarithms on both sides, which will be well-defined provided that
\[
\sqrt{2^{-r-1}} \|b_1\|^2 - rQ_v^2 e_{\text{tors}}^2 > 3rQ_v e_{\text{tors}} / \sqrt{2}.
\]

Note that if $L$ does not satisfy the condition in Proposition 5.3.4, we can redefine $L$ with a larger $C$ until the condition is satisfied. Moreover, once we obtain a smaller $Q_v$, we can repeat the above process with this new $Q_v$ until no further reduction is possible. Finally, we take the maximum among all reduced $Q_v$ to be an upper bound for $Q$. Searching for all integral points then becomes an easy task if this bound is feasible.

### 5.3.4 Examples

We have implemented Smart and Stephens’ method into an algorithm and use it to compute the following examples; its MAGMA code can be found in Appendix A.2. A demonstration of how to use this code will be shown in Example 5.3.6.

**Example 5.3.5.** We will first verify the result from [SS97, Example 2]. Let $E$ be
5.3. Integral Points on Elliptic Curves

the elliptic curve defined over $K = \mathbb{Q}(\sqrt{-2})$ given by the Weierstrass equation

$$E : \quad y^2 = x^3 - 16x + 16.$$ 

It can be checked that the torsion subgroup of $E(K)$ is trivial. In [SS97], it is claimed that $E(K) = \langle P_1, P_2 \rangle$, where

$$P_1 = (0, 4), \quad P_2 = (2, -2\sqrt{-2}).$$

We will first confirm that this is indeed the case. Using our algorithm for computing a lower bound for the canonical height, we obtain

$$\hat{h}(P) > 0.012$$

for all $P \in E(K)$. Then one can see from the height pairing matrix that

$$R(P_1, P_2) = \det(\langle P_j, P_k \rangle)_{1 \leq j, k \leq 2} = 0.0330 \neq 0,$$

i.e. $P_1$ and $P_2$ are independent. In addition, one can check using MAGMA that the rank of $E(K)$ is at most 2. Hence $E(K)$ has rank 2. By the geometry of numbers (Theorem 1.2.1), we have

$$n = [E(K) : \langle P_1, P_2 \rangle] \leq 17.4808.$$ 

In fact, one can verify that $n = 1$ after applying the sieving procedure for all primes $p \leq 17$. Hence $E(K) = \langle P_1, P_2 \rangle$.

Next, we wish to compute all integral points $P \in E(K)$. As discussed earlier, this is equivalent to finding an upper bound for $Q = \max\{|q_1|, |q_2|\}$, where $P = q_1P_1 + q_2P_2$. In this example, $E$ has only one complex embedding $E^{(e)}$. Using
Table 5.8: LLL reduction used in Example 5.3.5

<table>
<thead>
<tr>
<th>Previous ( Q )</th>
<th>( C ) being chosen</th>
<th>New ( Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^{24} )</td>
<td>( 1.0000 \times 10^{24} )</td>
<td>106</td>
</tr>
<tr>
<td>106</td>
<td>( 1.2625 \times 10^8 )</td>
<td>27</td>
</tr>
<tr>
<td>27</td>
<td>( 5.3144 \times 10^5 )</td>
<td>25</td>
</tr>
<tr>
<td>25</td>
<td>( 3.9062 \times 10^5 )</td>
<td>24</td>
</tr>
</tbody>
</table>

Theorem 4.5.3 to compute the period lattice \( \Lambda_v \) of \( E(v) \), we have \( \Lambda_v = \langle w_1, w_2 \rangle \) with

\[
w_1 = -i1.225694\ldots, \quad w_2 = 1.496729\ldots.
\]

Note that \( w_1, w_2 \) are chosen so that \( \tau = w_2/w_1 \) satisfies (3.1) as required. Moreover, one can compute both lower and upper bounds for \( h(P) - \hat{h}(P) \) for all \( P \in E(K) \) using, for example, [Sil90, Theorem 1.1], and obtain

\[
-5.461894 \leq h(P) - \hat{h}(P) \leq 6.211695.
\]

Using the above quantities, our algorithm shows that

\[
Q_0 = 12.2286, \quad c_9 = 2106.0087, \quad c_{10} = 0.0256.
\]

In addition, we obtain the following quantities for David’s constants:

\[
d_8 = 31.5690, \quad d_9 = 4.7156, \quad d_{10} = 1.9249 \times 10^{110}, \quad h_E = 11.6136.
\]

This finally yields \( Q_v \leq 10^{61} \) as an initial upper bound for \( Q \). After applying LLL basis reduction repeatedly until no further reduction is possible (see Table 5.8), we are finally able to reduce an upper bound for \( Q \) to 24. A quick search within this range then reveals all integral points in \( E(K) \), as listed (up to inverse) in Table 5.9.

Note that the quantities \( Q_0, c_9, c_{10} \) we obtained from our algorithm are slightly

\[\text{We use Silverman’s bounds in this example so that our constants can be compared with the ones shown in [SS97, Example 2] directly. In our algorithm (see Appendix A.2), we will use [CPS06, Theorem 1] to compute these bounds.}\]
Table 5.9: Integral points on $y^2 = x^3 - 16x + 16$ over $\mathbb{Q} (\sqrt{-2})$

<table>
<thead>
<tr>
<th>$(q_1, q_2)$</th>
<th>$P = q_1 P_1 + q_2 P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 0)</td>
<td>(0, 4)</td>
</tr>
<tr>
<td>(2, 0)</td>
<td>(4, 4)</td>
</tr>
<tr>
<td>(3, 0)</td>
<td>(-4, -4)</td>
</tr>
<tr>
<td>(4, 0)</td>
<td>(8, -20)</td>
</tr>
<tr>
<td>(5, 0)</td>
<td>(1, -1)</td>
</tr>
<tr>
<td>(6, 0)</td>
<td>(24, 116)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(2, -2\sqrt{-2})</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>(4\sqrt{-2}, -12 + 8\sqrt{-2})</td>
</tr>
<tr>
<td>(1, -1)</td>
<td>(-4\sqrt{-2}, -12 - 8\sqrt{-2})</td>
</tr>
<tr>
<td>(2, 1)</td>
<td>(-4 + 4\sqrt{-2}, 20)</td>
</tr>
<tr>
<td>(2, -1)</td>
<td>(-4 - 4\sqrt{-2}, 20)</td>
</tr>
<tr>
<td>(5, 1)</td>
<td>(-10 - 4\sqrt{-2}, 28 - 18\sqrt{-2})</td>
</tr>
<tr>
<td>(5, -1)</td>
<td>(-10 + 4\sqrt{-2}, 28 + 18\sqrt{-2})</td>
</tr>
<tr>
<td>(3, 2)</td>
<td>(60 - 40\sqrt{-2}, 316 - 480\sqrt{-2})</td>
</tr>
<tr>
<td>(3, -2)</td>
<td>(60 + 40\sqrt{-2}, 316 + 480\sqrt{-2})</td>
</tr>
</tbody>
</table>

different from the ones shown in [SS97, Example 2] due to some modifications in the formulas; this, however, has no effect on the final result.

**Example 5.3.6.** Let $K = \mathbb{Q}(\theta)$ where $\theta$ is a root of the polynomial $x^3 - 2$. In Example 5.2.6, we have readily verified that the elliptic curve $E_6 / K$ given by

$$E_6 : \quad y^2 = x^3 - (\theta^2 + 3\theta)x + \theta^2$$

has $\{P_1, P_2, P_3\}$ as a $\mathbb{Z}$-basis for $E(K)$, where

$$P_1 = (0, \theta), \quad P_2 = (1 + \theta, 1), \quad P_3 = (3 - 9\theta + 7\theta^2, 31 + 23\theta - 36\theta^2).$$

Thus any integral point $P$ can be expressed as $P = q_1 P_1 + q_2 P_2 + q_3 P_3$ for some $q_1, q_2, q_3 \in \mathbb{Z}$. We now wish to find all integral points in $E_6(K)$, that is, to find an upper bound for $Q = \max\{|q_1|, |q_2|, |q_3|\}$. To ease notation, we shall write $E = E_6$ and let $v_r, v_c$ be the real and complex archimedean place of $K$ respectively.

In order to determine such an upper bound, we will start by computing some certain constants associated to $E / K$. By [CPS06, Theorem 1], we first obtain

$$-1.196864 \leq h(P) - \hat{h}(P) \leq 0.174492$$
for all $P \in \mathcal{E}(K)$. Moreover, our algorithm shows that $h_\mathcal{E} = 11.9773$. The next step is to compute other constants associated to each real and complex embedding of $\mathcal{E}$ respectively.

Consider the case when $v = v_r$. Using Theorem 4.5.3 and some linear transformation, one can see that a $\mathbb{Z}$-basis $\{w_1, w_2\}$ for the period lattice of the real embedding $\mathcal{E}^{(v)}$ is given by

$$w_1 = i1.658105\ldots, \quad w_2 = -1.815187\ldots.$$  

Observe that $\tau = w_2/w_1$ satisfies (3.1). Combining all information on $\mathcal{E}/K$ and $\mathcal{E}^{(v)}$ we have obtained so far, our algorithm shows that

$$c_9 = 19.6306, \quad c_{10} = 0.2017, \quad Q_0 = 2.8967,$$

and also

$$d_8 = 32.5576, \quad d_9 = 5.5533, \quad d_{10} = 7.9894 \times 10^{161}.$$  

Hence by Proposition 5.3.3, we obtain $Q_v = 10^{87}$ as an initial upper bound for $Q$. After applying LLL basis reduction repeatedly as shown in Table 5.10, one can finally reduce $Q_v$ to 11.

For $v = v_c$, we also obtain from Theorem 4.5.3 that the period lattice of the complex embedding $\mathcal{E}^{(v)}$ is given by $\Lambda_v = \langle w_1, w_2 \rangle$, where

$$w_1 = 1.106543\ldots + i1.444101\ldots, \quad w_2 = -1.838531\ldots + i1.133717\ldots.$$  

Again, $w_1, w_2$ are chosen so that $\tau = w_2/w_1$ satisfies (3.1). A similar computation as before also shows that

$$c_9 = 214.9545, \quad c_{10} = 0.1009, \quad Q_0 = 3.3679,$$

$$d_8 = 32.5576, \quad d_9 = 5.7684, d_{10} = 6.2775 \times 10^{161}.$$
5.3. Integral Points on Elliptic Curves

Table 5.10: LLL reduction used in Example 5.3.6

<table>
<thead>
<tr>
<th>$v = v_r$</th>
<th>$v = v_u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous $Q_v$</td>
<td>New $Q_v$</td>
</tr>
<tr>
<td>Previous $Q_v$</td>
<td>New $Q_v$</td>
</tr>
<tr>
<td>$10^{57}$</td>
<td>$1.0000 \times 10^{522}$</td>
</tr>
<tr>
<td>$67$</td>
<td>$2.7207 \times 10^{16}$</td>
</tr>
<tr>
<td>$12$</td>
<td>$5.1598 \times 10^{9}$</td>
</tr>
</tbody>
</table>

Table 5.11: Integral points on the elliptic curve $E_6$

<table>
<thead>
<tr>
<th>$(q_1, q_2, q_3)$</th>
<th>$P = q_1 P_1 + q_2 P_2 + q_3 P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1, 0, 0)$</td>
<td>$(0, \theta)$</td>
</tr>
<tr>
<td>$(0, 1, 0)$</td>
<td>$(1 + \theta, 1)$</td>
</tr>
<tr>
<td>$(1, -1, 0)$</td>
<td>$(-\theta, -2\theta)$</td>
</tr>
<tr>
<td>$(2, -1, 0)$</td>
<td>$(9 + \theta, -27 - 4\theta)$</td>
</tr>
<tr>
<td>$(0, 0, 1)$</td>
<td>$(3 - 9\theta + 7\theta^2, 31 + 23\theta - 36\theta^2)$</td>
</tr>
<tr>
<td>$(2, -2, -1)$</td>
<td>$(139 + 111\theta + 87\theta^2, 2837 + 2253\theta + 1788\theta^2)$</td>
</tr>
</tbody>
</table>

In consequence, Proposition 5.3.3 yields $Q_v = 10^{57}$ as an initial bound, which is eventually reduced to 13 after successive LLL reductions as shown in Table 5.10.

Hence we have $Q \leq \max\{11, 13\} = 13$. The complete list of all integral points (up to inverse) in $E(K)$ is shown in Table 5.11; this is computed using our MAGMA code in Appendix A.2 (intpts.m) together with the following instructions:

```magma
> Attach("nfhtbound.m"); // from Cremona - for computing CPS bound
> Attach("intpts.m");  // main program for computing integral points
> Attach("elog.m");    // for computing periods and elliptic logarithms
> SetVerbose("Intpts", 1); // minimal printing
> // Define elliptic curve E
> _<x> := PolynomialRing(Integers());
> K<a> := NumberField(x^3-2);
> E := EllipticCurve([-a^2-3*a, a^2]);
> // Generators for E(K)
> P1 := E![0,a];
> P2 := E![1+a,1];
> P3 := E![3-9*a+7*a^2, 31+23*a-36*a^2];
> L, _ := IntegralPoints(E, [P1,P2,P3]);
> Maximum absolute bound on coefficients = 13
> Exact arithmetic
> [ 1, 0, 0 ] => (0 : a : 1)
> [ 0, 1, 0 ] => (a + 1 : 1 : 1)
> [ 1, -1, 0 ] => (-a : -2*a : 1)
> [ 2, -1, 0 ] => (a + 9 : -4*a - 27 : 1)
> [ 0, 0, 1 ] => (7*a^2 - 9*a + 3 : -36*a^2 - 23*a + 31 : 1)
> [ 2, -2, -1 ] => (87*a^2 + 111*a + 139 : 1788*a^2 + 2253*a + 2837 : 1)

In conclusion, it should be noted that although Smart and Stephens’ method [SS97], in principle, allows one to find all integral points on any elliptic curves over...
number fields, it requires a number of certain results on elliptic curves which may not be obtained easily, especially in the past. For example, lack of an algorithm for computing period lattices of arbitrary elliptic curves over \( \mathbb{C} \) would prevent one to apply the method to most elliptic curves other than the ones having real coefficients. Our main results on height bound (see Chapter 3), period lattices and complex elliptic logarithms (see Chapter 4) therefore enhance Smart and Stephens’ method by minimising its limitations.

5.4 Elliptic Curves with Everywhere Good Reduction

We finally come to the last section of this thesis, where we will illustrate an application of integral points, whose computation requires all the main results of this thesis, on finding elliptic curves with everywhere good reduction over some quadratic number fields. The method for finding this family of elliptic curves is due to Cremona and Lingham [CL07], which will be explained very briefly in this section.

5.4.1 Cremona–Lingham’s Method: An Overview

Definition. Let \( K \) be a number field with ring of integers \( \mathcal{O}_K \), and let \( S \) be a finite set of prime ideals of \( \mathcal{O}_K \). We say that \( x \in K \) is an \( S \)-integer if \( \text{ord}_p(x) \geq 0 \) for all prime ideals \( p \notin S \).

It is easy to verify that that the set of all \( S \)-integers is a ring, which will be denoted by \( \mathcal{O}_{K,S} \) from now on.

For a finite set \( S \) of prime ideals of \( \mathcal{O}_K \) and \( m \in \mathbb{Z}_{>0} \), we define

\[
K(S,m) = \{ x \in K^*/K^{*m} : \text{ord}_p(x) \equiv 0 \pmod{m} \text{ for all } p \notin S \}.
\]

Here, \( K^* = K \setminus \{0\} \). For convenience, we will also abuse the notation and say that
an element \( x \in K^* \) is in \( K(S, m) \) if \( xK^{*m} \in K(S, m) \). The following proposition can be proved very easily.

**Proposition 5.4.1** ([CL07, Proposition 2.1]). Let \( m, n \) be coprime. Then

\[
K(S, mn) \cong K(S, m) \times K(S, n)
\]

via the map \( w \mapsto (w, w) \), with inverse map \((u, v) \mapsto v^{am}u^{bn}, \) where \( am + bn = 1 \).

For this application, we will see later that we will need to consider \( K(S, m) \) for \( m = 4, 6, 12 \), and also the set \( K(S, 6)_{12} \) which is the image of the natural map \( K(S, 12) \rightarrow K(S, 6) \). By Proposition 5.4.1, it then suffices to compute \( K(S, m) \) only for \( m = 2, 3, 4 \). For \( m = 2, 3 \), this can be computed easily using the MAGMA function \texttt{pSelmerGroup()}\). A set of MAGMA functions for determining \( K(S, 4) \) has been implemented by Professor John Cremona who kindly supplied me with them.

Let \( E \) be an elliptic curve defined over \( K \), given by a Weierstrass equation (1.1) as before, i.e.,

\[
E: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.
\]

**Definition.** We say that \( E \) has *good reduction at \( p \) if it has a \( p \)-integral model (i.e., \( \text{ord}_p(a_j) \geq 0 \) for all \( j \)) whose discriminant \( \Delta \) is a \( p \)-unit (i.e., \( \text{ord}_p(\Delta) = 0 \)).

By Shafarevich’s Theorem (see [Sil86, pp. 263–264] for the proof), it is well known that there are finitely many isomorphism classes of \( E/K \) having good reduction outside a finite set of primes \( S \). The following proposition shows the connection between \( K(S, 6)_{12} \) and the set of all elliptic curves over \( K \) with good reduction outside \( S \).

**Proposition 5.4.2.** Let \( E \) be an elliptic curve defined over \( K \) with \( j \)-invariant \( j(E) = j \neq 0, 1728 \) and good reduction at all primes \( p \notin S \). Set \( w = j^2(j - 1728)^3 \).
Then
\[ \Delta \in K(S, 12), \quad j \in \mathcal{O}_{K,S}, \quad w \in K(S, 6)_{12}. \]

Conversely, if \( j \in \mathcal{O}_{K,S} \) with \( w = j^2(j - 1728)^3 \in K(S, 6)_{12} \), then the elliptic curve
\[
E : \quad y^2 = x^3 - 3u^2j(j - 1728)x - 2u^3j(j - 1728)^2
\]
with \( u \in K^* \) satisfying \((3u)^6w \in K(S, 12)\), has \( j(E) = j \) and good reduction at all primes outside the extended set \( S^{(6)} = S \cup \{ p : \text{ord}_p(6) > 0 \} \).

Proof. See [CL07, Proposition 3.2]. \( \square \)

To obtain elliptic curves \( E \) with \( j(E) \neq 0, 1728 \) and good reduction outside \( S \), the strategy of Cremona and Lingham’s method is to consider each class \( w \in K(S, 6)_{12} \) in turn, and determine all possible \( j \in \mathcal{O}_{K,S} \) satisfying \( w \equiv j^2(j - 1728)^3 \pmod{K^*6} \). For each of such \( j \), one then obtain an elliptic curve \( E \) with good reduction outside \( S^{(6)} \) using the converse of Proposition 5.4.2. If such \( E \) also has good reduction at all primes \( p | 6 \), then \( E \) has good reduction outside \( S \), and we discard \( E \) otherwise. By [CL07, Proposition 3.4], the complete set of all curves with \( j \)-invariant \( j \) and good reduction outside \( S \) is obtained by twisting \( E \) with \( u \in K(S, 2) \).

Definition. Let \( E \) be an elliptic curve defined over a number field \( K \), and let \( S \) be a finite set of prime ideals of \( \mathcal{O}_K \). A point \( P = (x, y) \in E(K) \) is said to be an \( S \)-integral point if both \( x, y \in \mathcal{O}_{K,S} \). In addition, if \( S = \emptyset \) (i.e., \( \mathcal{O}_{K,S} = \mathcal{O}_K \)), then we simply say that \( P \) is an integral point.

The next proposition shows that all possible \( j \) can arise from \( S \)-integral points on certain elliptic curves over \( K \).

Proposition 5.4.3. Let \( w \in K(S, 6) \). Each \( j \in \mathcal{O}_{K,S} \setminus \{ 0, 1728 \} \) with \( j^2(j - 1728)^3 \equiv w \pmod{K^*6} \) has the form \( j = x^3/w = 1728 + y^2/w \), where \( P = (x, y) \)
(with \(xy \neq 0\)) is an \(S\)-integral point on the elliptic curve

\[ E_w : \quad y^2 = x^3 - 1728w. \]

**Proof.** See [CL07, Proposition 3.3].

It should be noted, however, that not all the values \(j\) obtained by Proposition 5.4.3 are \(S\)-integral. Furthermore, not every \(S\)-integral \(j\) arising from an \(S\)-integral points on some \(E_w\) will necessarily be the \(j\)-invariant of a suitable elliptic curve, unless \(j\) is derived from \(w \in K(S, 6)_{12}\).

To summarise, in order to find elliptic curves with *everywhere* good reduction, we set \(S = \emptyset\) and apply Cremona and Lingham’s method [CL07]. For those curves with \(j\)-invariant neither 0 nor 1728, the computation proceeds as follows:

1. Compute \(K(\emptyset, 6)\) from \(K(\emptyset, 2)\) and \(K(\emptyset, 3)\), and determine a (finite) representative set \(W\) of \(w \in K(\emptyset, 6)_{12}\).
2. For each \(w \in W\), find all integral points on the elliptic curve \(E_w/K\) such that \(j = x^3/w \in \mathcal{O}_K\).
3. If such \(j\) satisfies \(j^2(j - 1728)^3 \in K(\emptyset, 6)_{12}\), then we determine \(u_0 \in K^*\) such that \((3u_0)^6j^2(j - 1728)^3 \in K(\emptyset, 12)\). Let \(E\) be the elliptic curve

\[ E : \quad y^2 = x^3 - 3u_0^2j(j - 1728)x - 2u_0^3j(j - 1728)^2. \]

Check if \(E\) has good reduction at all primes \(p\) dividing 6; discard \(E\) if not.
4. Repeat step (3) for each quadratic twist \(E^{(u)}\), where \(u \in K(\emptyset, 2)\).

Since \(S = \emptyset\), it is immediate from [CL07, Proposition 4.1] that there is no elliptic curve \(E/K\) with \(j(E) = 0\) and everywhere good reduction. For \(j = 1728\), finding elliptic curves with \(j\)-invariant \(j\) and everywhere good reduction does not involve searching for integral points at all; see [CL07, Proposition 4.2] for more details.
5.4.2 Examples I: Real Quadratic Fields

As we have seen in Section 5.3, the applications of all the main results of this thesis finally allow one to use Smart and Stephens’ method [SS97] to find all integral points on elliptic curves over number fields with less restriction. This in turn benefits to the determination of elliptic curves with everywhere good reduction using Cremona and Lingham’s algorithm [CL07]. In particular, we are able to settle some inconclusive cases appearing in Cremona’s compiled list\footnote{Available at http://www.warwick.ac.uk/~masgaj/ecegr/ecegrqf.html (last checked on November 22, 2010).} on elliptic curves over $K = \mathbb{Q}(\sqrt{d})$ with everywhere good reduction for $2 \leq d \leq 100$. For more information on imaginary quadratic fields $\mathbb{Q}(\sqrt{-d})$ (with $2 \leq d \leq 100$), see Section 5.4.3.

In this subsection, we will illustrate these new results in full details. Note, however, that on some real quadratic fields $K$ we may not fully confirm non-existence of elliptic curves over $K$ with everywhere good reduction, nor that the list of such elliptic curves is complete, owing to the difficulty in searching for non-torsion points on certain elliptic curves. Furthermore, we carry out our computation for all $d \leq 100$ which are inconclusive from Cremona’s table, apart from $d = 71, 79, 91$ in which there are too many elliptic curves $E_w/K$ whose Mordell–Weil bases are unknown.

Based on the tables shown later in this subsection (see next page for the description), we obtain the following conclusion in addition to what we already know from Cremona’s table.

**Proposition 5.4.4.** Let $2 \leq d \leq 100$. Then we have the following:

1. For $d = 55, 78, 95$, there is no elliptic curve over $\mathbb{Q}(\sqrt{d})$ with everywhere good reduction.

2. For $d = 38, 41, 65$, we have the complete list of all elliptic curves over $\mathbb{Q}(\sqrt{d})$ with everywhere good reduction.

**Proof.** For (1), see Table 5.17, 5.22, and 5.24. For (2), see Table 5.12, 5.13, and 5.20. \qed
5.4. Elliptic Curves with Everywhere Good Reduction

Description of Tables

In the following pages, we will illustrate in detail how to find all elliptic curves over a quadratic number field $K$ with everywhere good reduction using Cremona and Lingham’s method (see Section 5.4.1). For each $K$, we give a table whose columns represent the following information:

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
</tr>
</thead>
<tbody>
<tr>
<td># Index of each $w$.</td>
<td>$w$ Each $w \in W$, where $W$ is the set of representatives for $K(\emptyset, 6)_{12}$; note that this is unique modulo $K^{*6}$. If a fundamental unit $\varepsilon$ of $O_K$ exists, then $w$ will be expressed in terms of $\varepsilon$.</td>
<td>Torsion All generators of the torsion subgroup of $E_w(K)$, where $E_w$ is the elliptic curve $y^2 = x^3 - 1728w$. Each generator is denoted by $\langle T, t \rangle$, where $T \in E_w(K)$ is a generator, and $t = \text{ord}(T)$. If the torsion subgroup is trivial, we simply write “$O$”.</td>
<td>Rank The rank of $E_w(K)$.</td>
<td>Mordell–Weil basis A Mordell–Weil basis for $E_w(K)$. If the rank is 0, we simply write “-”.</td>
</tr>
</tbody>
</table>

If some information is currently unknown, then we put “?” in that entry.

Recall that not all $j$ shown in the table may yield an elliptic curve over $K$ with everywhere good reduction. However, if there exists $j$ which gives rise to such curves, then that $j$ and its associated integral point will be shown in **bold**.
Moreover, the details of all curves arising from that \( j \) will be shown in the second table, whose columns are as follows:

\( j \) The \( j \)-invariant of elliptic curves

\# Index of each elliptic curve \( E/K \) with \( j \)-invariant \( j \), having everywhere good reduction.

\( a_1, a_2, a_3, a_4, a_6 \) The \( a \)-invariants of the Weierstrass equation of \( E \).

\( \Delta \) The discriminant of \( E \). In case \( \Delta \) cannot be expressed exactly (for example, when there is no globally minimal model for the curve), then the ideal \( \langle \Delta \rangle \) will be shown instead.

**Torsion** The torsion subgroup of \( E(K) \), represented by the same notation as above.

**Rank** The rank of \( E(K) \).

At the end, a summary line will be given. This can be either a **conclusion** (that our list of elliptic curves over \( K \) with everywhere good reduction is complete, or that there is no such curve), or a **conjecture** (especially when there exists \( w \) which currently cannot be completely settled).
Table 5.12: Finding elliptic curves over $\mathbb{Q}(\sqrt{38})$ with everywhere good reduction

$(a = \sqrt{38}, \varepsilon = 37 + 6a, \mathcal{N}(\varepsilon) = 1)$

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0,1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$((12, 0), 2)$</td>
<td>1</td>
<td>$\left(\frac{111+5a}{31}, \frac{557+192a}{31}\right)$</td>
<td>$(12, 0)$</td>
<td>--</td>
</tr>
<tr>
<td>2</td>
<td>$-1$</td>
<td>$((-12, 0), 2)$</td>
<td>2</td>
<td>$\left(\frac{-10}{9}, \frac{152a}{27}\right), (20, 16a)$</td>
<td>$\left(20, 16a\right)$, $\left(-12, 0\right)$, $\left(1014, 2538a\right)$</td>
<td>$-8000$, $-$, $-1042590744$</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon^7$</td>
<td>$((444 + 72a, 0), 2)$</td>
<td>2</td>
<td>$\left(\frac{91125 + 19158a}{2025}, \frac{3552 + 1364a}{2025}\right)$, $\left(\frac{91125}{2025}, \frac{91125 + 19158a}{2025}\right)$</td>
<td>$\left(740 + 120a, -25200 - 4088a\right)$, $\left(444 + 72a, 0\right)$, $\left(2442 + 396a, 170100 + 27594a\right)$</td>
<td>$8000$, $-$, $287496$</td>
</tr>
<tr>
<td>4</td>
<td>$-\varepsilon^7$</td>
<td>$((-444 - 72a, 0), 2)$</td>
<td>1</td>
<td>$\left(-296 - 48a, 11096 + 1800a\right)$</td>
<td>$\left(-296 - 48a, 11096 + 1800a\right)$, $\left(-444 - 72a, 0\right)$, $\left(3552 + 576a, -299592 - 48600a\right)$</td>
<td>$512$, $-$, $-884736$</td>
</tr>
<tr>
<td>5</td>
<td>$\bar{\varepsilon}^2$</td>
<td>$O$</td>
<td>0</td>
<td>$-$</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>6</td>
<td>$-\bar{\varepsilon}^2$</td>
<td>$O$</td>
<td>1</td>
<td>$\left(-24 + 4a, 1528 - 248a\right)$</td>
<td>$\left(-24 + 4a, 1528 - 248a\right)$</td>
<td>$-768 - 128a$</td>
</tr>
<tr>
<td>7</td>
<td>$\varepsilon$</td>
<td>$O$</td>
<td>1</td>
<td>$\left(\frac{1210 + 344a}{25}, \frac{218232 - 113112a}{25}\right)$</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>8</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>$-$</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>9</td>
<td>Conjugate to #5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Conjugate to #6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Conjugate to #7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Conjugate to #8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

List of curves$^a$

<table>
<thead>
<tr>
<th>$j$</th>
<th>#</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_6$</th>
<th>$\Delta$</th>
<th>Torsion</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>1</td>
<td>$a$</td>
<td>$1 + a$</td>
<td>$1 + a$</td>
<td>$15 + 4a$</td>
<td>$21 + 4a$</td>
<td>$\varepsilon^3$</td>
<td>$\left(\frac{-\varepsilon}{2}, \frac{-\varepsilon}{2}\right), 2$</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>$a$</td>
<td>$1 - a$</td>
<td>$1 + a$</td>
<td>$15 - 5a$</td>
<td>$21 - 5a$</td>
<td>$\varepsilon^3$</td>
<td>$\left(\frac{-\varepsilon}{2}, \frac{-\varepsilon}{2}\right), 2$</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Conclusion: All elliptic curves over $\mathbb{Q}(\sqrt{38})$ with everywhere good reduction have been found.

$^a$These curves were initially found by Cremona; we confirm that no other curves can arise from the remaining cases. Note also that, although their $j$-invariant is rational, these curves are not isomorphic to any elliptic curves over $\mathbb{Q}$. 

5.4. Elliptic Curves with Everywhere Good Reduction
The first two curves were previously found by Commando (Conno). Theorem 3.2; we complete the list by finding other curves.

**Conjecture:** All elliptic curves over $\mathbb{Q}(\sqrt{3})$ with everywhere good reduction have been found.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Conjecture to #7</th>
<th>Conjecture to #8</th>
<th>Conjecture to #6</th>
<th>Conjecture to #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Curve</th>
<th>Conjecture to #9</th>
<th>Conjecture to #10</th>
<th>Conjecture to #11</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.13:** Finding elliptic curves over $\mathbb{Q}(\sqrt{3})$ with everywhere good reduction.
Table 5.14: Finding elliptic curves over $\mathbb{Q}(\sqrt{43})$ with everywhere good reduction
($a = \sqrt{43}$, $\varepsilon = 3482 + 531a$, $N(\varepsilon) = 1$)

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0, 1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$((12, 0), 2)$</td>
<td>1</td>
<td>(99285900, 71388478020)</td>
<td>(12, 0)</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>$-1$</td>
<td>$((-12, 0), 2)$</td>
<td>1</td>
<td>(960, 4536a)</td>
<td>(960, 4536a), (12, 0)</td>
<td>$-884736000, -$</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon^2$</td>
<td>$((-141784 - 6372a, 0), 2)$</td>
<td>2</td>
<td>$(6940 - 10620a, 23012360 - 3509352a)$</td>
<td>(69640 - 10620a, 23012360 - 3509352a), (41784 - 6372a, 0), (229812 - 35046a, 23688126a)</td>
<td>$8000, -$, 287496</td>
</tr>
<tr>
<td>4</td>
<td>$-\varepsilon^2$</td>
<td>$((-141784 + 6372a, 0), 2)$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>5*</td>
<td>$\varepsilon^2$</td>
<td>$O$</td>
<td>$\leq 1$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>$-\varepsilon^2$</td>
<td>$O$</td>
<td>1</td>
<td>$(3200 - 488a, -294088 + 44848a)$</td>
<td>$(3200 - 488a, -294088 + 44848a)$, (3200 - 488a, -294088 + 44848a)</td>
<td>$-2048 + 512a$, $-483328 - 73728a$</td>
</tr>
<tr>
<td>7</td>
<td>$\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #5</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #6</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #7</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #8</td>
</tr>
</tbody>
</table>

**Conjecture:** No elliptic curve over $\mathbb{Q}(\sqrt{43})$ with everywhere good reduction exists.

*We cannot find a Mordell–Weil basis for $E_w(\mathbb{Q}(\sqrt{43}))$ due to the difficulty in searching for a point on the curve. Any elliptic curve with everywhere good reduction, if exists, will arise only from this case.
Table 5.15: Finding elliptic curves over $\mathbb{Q}(\sqrt{46})$ with everywhere good reduction.

<table>
<thead>
<tr>
<th>#</th>
<th>Weil to $\mathbb{Q}$</th>
<th>Weil to $\mathbb{F}_5$</th>
<th>$a$</th>
<th>$\varepsilon$</th>
<th>Integral points</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$(12, 0)$</td>
<td>1</td>
<td>$(0, 21)$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$(12, 0)$</td>
<td>1</td>
<td>$(0, 21)$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$(12, 0)$</td>
<td>1</td>
<td>$(0, 21)$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$\sqrt{46}$</td>
<td>$24335 - 3588$</td>
<td>$(12, 0)$</td>
<td>1</td>
<td>$(0, 21)$</td>
<td></td>
</tr>
</tbody>
</table>

We cannot find a Mordell–Weil basis for $E_{\mathbb{Q}(\sqrt{46})}$ due to the difficulty in searching for a point on the curve. Any elliptic curve with everywhere good reduction exists.

Conjecture: No elliptic curve over $\mathbb{Q}(\sqrt{46})$ with everywhere good reduction exists.
Table 5.16: Finding elliptic curves over \( \mathbb{Q}(\sqrt{51}) \) with everywhere good reduction
\( (a = \sqrt{51}, \varepsilon = 50 - 7a, N(\varepsilon) = 1) \)

<table>
<thead>
<tr>
<th>#</th>
<th>( w )</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>( j \neq 0, 1728 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>((12, 0), 2)</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>( \varepsilon^* )</td>
<td>((600 - 84a, 0), 2)</td>
<td>1</td>
<td>((1000 - 140a, 39592 - 5544a))</td>
<td>((1000 - 140a, 39592 - 5544a), (600 - 84a, 0), (3300 - 462a, -267246 + 37422a))</td>
<td>8000</td>
</tr>
<tr>
<td>3</td>
<td>(-1)</td>
<td>((-12, 0), 2)</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>(-\varepsilon^*)</td>
<td>((-600 + 84a, 0), 2)</td>
<td>1</td>
<td>(\left( \frac{13500000044835409}{17}, \frac{289247112 - 40502616a}{269} \right))</td>
<td>((-600 + 84a, 0))</td>
<td>–</td>
</tr>
<tr>
<td>5</td>
<td>( \varepsilon^* )</td>
<td>(O)</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>( \varepsilon )</td>
<td>(O)</td>
<td>1</td>
<td>((1000 - 84a, 28000 - 5040a))</td>
<td>((1000 - 84a, 28000 - 5040a))</td>
<td>204736761728 - 28668371200a</td>
</tr>
<tr>
<td>7</td>
<td>(-\varepsilon^*)</td>
<td>(O)</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8a</td>
<td>(-\bar{\varepsilon})</td>
<td>(O)</td>
<td>(\leq 1)</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #5</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #6</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #7</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #8</td>
</tr>
</tbody>
</table>

**Conjecture:** No elliptic curve over \( \mathbb{Q}(\sqrt{51}) \) with everywhere good reduction exists.

---

*We cannot find a Mordell–Weil basis for \( E_w(\mathbb{Q}(\sqrt{51})) \) due to the difficulty in searching for a point on the curve. Any elliptic curve with everywhere good reduction, if exists, will arise only from this case.*
Table 5.17: Finding elliptic curves over $\mathbb{Q}(\sqrt{55})$ with everywhere good reduction exists:

<table>
<thead>
<tr>
<th>Conclusion: No elliptic curve over $\mathbb{Q}(\sqrt{55})$ with everywhere good reduction exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>$j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
<tr>
<td>$-j \neq 0$, $1728$</td>
</tr>
</tbody>
</table>

Table 5.17: Finding elliptic curves over $\mathbb{Q}(\sqrt{55})$ with everywhere good reduction exists.
### Table 5.18: Finding elliptic curves over $\mathbb{Q}(\sqrt{59})$ with everywhere good reduction

$(a = \sqrt{59}, \varepsilon = 530 + 69a, \mathcal{N}(\varepsilon) = 1)$

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$((12, 0), 2)$</td>
<td>1</td>
<td>$(160000 + 283020a, 160000 + 283020a)$</td>
<td>$(12, 0)$</td>
</tr>
<tr>
<td>2</td>
<td>−1</td>
<td>$((-12, 0), 2)$</td>
<td>1</td>
<td>$(160000 - 283020a, 160000 - 283020a)$</td>
<td>$(-12, 0)$</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon^3$</td>
<td>$(6360 - 828a, 0), 2$</td>
<td>2</td>
<td>$(10600 - 1380a, 1366568 - 177912a)$, $(6360 - 828a, 0)$, $(297697154818529013885106789419132130737289661391272)$</td>
<td>$(10600 - 1380a, 1366568 - 177912a)$</td>
</tr>
<tr>
<td>4</td>
<td>$-\varepsilon^3$</td>
<td>$(6360 + 828a, 0), 2$</td>
<td>2</td>
<td>$(50000200 - 650940a, 3840591605112 - 50002437772a)$</td>
<td>$(-6360 + 828a, 0)$</td>
</tr>
<tr>
<td>5*</td>
<td>$\varepsilon^2$</td>
<td>$O$, $\leq 2$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>$-\varepsilon^2$</td>
<td>$O$, 2</td>
<td>2</td>
<td>$(552 + 72a, 12168 - 1584a)$, $(8304 - 1080a, 1069560 - 139248a)$</td>
<td>$(552 + 72a, 12168 - 1584a)$, $(8304 - 1080a, 1069560 - 139248a)$</td>
</tr>
<tr>
<td>7*</td>
<td>$\varepsilon$</td>
<td>$O$, $\leq 1$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>8</td>
<td>$-\varepsilon$</td>
<td>$O$, 1</td>
<td>1</td>
<td>$(3385171742, 310792710262)$</td>
<td>$(3385171742, 310792710262)$</td>
</tr>
</tbody>
</table>

Conjugate to #6

Conjugate to #7

Conjugate to #8

Conjuction: No elliptic curve over $\mathbb{Q}(\sqrt{59})$ with everywhere good reduction exists.

---

*We cannot find Mordell–Weil bases for both $E_w(\mathbb{Q}(\sqrt{59}))$ due to the difficulty in searching for a point on both curves. Any elliptic curves with everywhere good reduction, if exist, will arise only from these two cases.*
Table 5.19: Finding elliptic curves over $\mathbb{Q}^\sqrt[6]{(62)}$ with everywhere good reduction.

<table>
<thead>
<tr>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Z}$</td>
<td>1</td>
<td>$(31126430, 1034289)$</td>
<td>$(12, 0)$, $2$</td>
</tr>
<tr>
<td>$\mathbb{Z}^2$</td>
<td>2</td>
<td>$(21343982842a, 1051871913)$</td>
<td>$(12, 0)$, $-2$</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>3</td>
<td>$(30492, -3872a, 25, 8377936)$</td>
<td>$(756 - 96a, 0)$, $2$</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>4</td>
<td>$(1260 - 160a, -56000 + 7112a)$, $(1260 - 160a, -56000 + 7112a)$</td>
<td>$(4158 - 528a, 378000 - 48006a)$, $2$</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>5</td>
<td>$(0, -121, 0)$</td>
<td>$(0, -121, 0)$, $2$</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>6</td>
<td>$(0)$</td>
<td>$(0)$, $2$</td>
</tr>
</tbody>
</table>

Conjecture: No elliptic curve over $\mathbb{Q}^\sqrt[6]{(62)}$ with everywhere good reduction exists.

We cannot find Mordell–Weil bases for both $\mathbb{F}_p\mathbb{G}(\sqrt[6]{(62)})$ due to the difficulty in searching for a point on both curves. Any elliptic curves with everywhere good reduction will arise only from these two cases.

Conjecture: No elliptic curve over $\mathbb{Q}^\sqrt[6]{(62)}$ with everywhere good reduction exists.
Table 5.20: Finding elliptic curves over $\mathbb{Q}(\sqrt{65})$ with everywhere good reduction

\[ (a = \frac{1 + \sqrt{65}}{2}, \varepsilon = 9 - 2a, N(\varepsilon) = -1) \]

<table>
<thead>
<tr>
<th>#</th>
<th>w</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0, 1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$((12, 0), 2)$</td>
<td>2</td>
<td>$(257, 511 - 1022a)$, $(17, -7 + 14a)$</td>
<td>$(257, 511 - 1022a)$, $(17, -7 + 14a), (12, 0)$</td>
<td>$16974593$</td>
</tr>
<tr>
<td>2</td>
<td>$\varepsilon^3$</td>
<td>$((-108 + 24a, 0), 2)$</td>
<td>2</td>
<td>$(217 - 48a, -3997 + 882a)$, $(3353 - 714a, 240667 - 51312a)$</td>
<td>$(217 - 48a, -3997 + 882a)$, $(596225 - 131584a, 614101215 - 135529408a)$, $(108 - 24a, 0)$</td>
<td>$8049 - 666a$</td>
</tr>
<tr>
<td>3</td>
<td>$-1$</td>
<td>$((-12, 0), 2)$</td>
<td>0</td>
<td>–</td>
<td>$(12, 0)$</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>$-\varepsilon^3$</td>
<td>$((-108 + 24a, 0), 2)$</td>
<td>2</td>
<td>$(1937 + 480a, 112889 + 33278a)$, $(-71 + 16a, -1267 + 280a)$</td>
<td>$(1937 + 480a, 112889 + 33278a)$, $(-71 + 16a, -1267 + 280a)$</td>
<td>$126187864952935 + 35735839572482a$</td>
</tr>
<tr>
<td>5</td>
<td>$\varepsilon^2$</td>
<td>$O$</td>
<td>1</td>
<td>$(22669663389, 298457253093)$</td>
<td>$(12, 0)$</td>
<td>–</td>
</tr>
<tr>
<td>6</td>
<td>$\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>$-\varepsilon^2$</td>
<td>$O$</td>
<td>1</td>
<td>$(-12 - 3a, -288 - 81a)$, $(1182 + 24a, -288 - 81a)$</td>
<td>$(1182 + 24a, -288 - 81a)$</td>
<td>$432 - 81a$</td>
</tr>
<tr>
<td>8</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>2</td>
<td>$(448 + 24a, -1504 - 576a)$, $(20 - 3a, -128 + 31a)$</td>
<td>$(448 + 24a, -1504 - 576a)$, $(20 - 3a, -128 + 31a)$</td>
<td>$119055040 + 33720329a$, $848 + 745a$</td>
</tr>
</tbody>
</table>

Continue...
Conclusion: All elliptic curves over \( \mathbb{Q}(\sqrt{65}) \) with everywhere good reduction have been found.

<table>
<thead>
<tr>
<th>Curve #</th>
<th>( \mathbf{a} )</th>
<th>( \mathbf{b} )</th>
<th>Rank</th>
<th>( \mathbf{c} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2 )</td>
<td>( 1 )</td>
<td>( 0 )</td>
<td>( \Delta )</td>
<td>( \mathbf{a} )</td>
</tr>
<tr>
<td>( 2 )</td>
<td>( 1 )</td>
<td>( 0 )</td>
<td>( \Delta )</td>
<td>( \mathbf{a} )</td>
</tr>
</tbody>
</table>

\( \mathbf{a} = 4 \) \( \mathbf{b} = 4 \) \( \mathbf{c} = 4 \)

\( \langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = 4 \) \( \mathbf{d} \)

\( \langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = 4 \) \( \mathbf{d} \)

\( \langle \mathbf{a} + \mathbf{b}, \mathbf{c} \rangle = 4 \) \( \mathbf{d} \)

List of curves (continued)
Table 5.21: Finding elliptic curves over $\mathbb{Q}(\sqrt{67})$ with everywhere good reduction

$(a = \sqrt{67}, \varepsilon = 48842 + 5967a, \mathcal{N}(\varepsilon) = 1)$

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0, 1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$((12, 0), 2)$</td>
<td>1</td>
<td>$(\frac{10436}{111}, -\frac{92988}{111})$</td>
<td>$(12, 0)$</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>−1</td>
<td>$((-12, 0), 2)$</td>
<td>1</td>
<td>$(\frac{-1591}{44}, \frac{-248}{44})$</td>
<td>$(5280, 46872a), (-12, 0)$</td>
<td>−147197952000, –</td>
</tr>
<tr>
<td>3</td>
<td>$\varepsilon$</td>
<td>$((586104 + 71604a, 0), 2)$</td>
<td>$\leq 2$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>$-\varepsilon$</td>
<td>$((-586104 - 71604a, 0), 2)$</td>
<td>$\leq 2$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>$\varepsilon$</td>
<td>$O$</td>
<td>$\leq 1$</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>6</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>1</td>
<td>$(-5304 - 648a, 1955448 + 238896a)$</td>
<td>$(-5304 - 648a, 1955448 + 238896a)$</td>
<td>$-6110208 + 746496a$</td>
</tr>
<tr>
<td>7</td>
<td>$\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #5</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #6</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #7</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #8</td>
</tr>
</tbody>
</table>

**Conjecture:** No elliptic curve over $\mathbb{Q}(\sqrt{67})$ with everywhere good reduction exists.

*We cannot find Mordell–Weil bases for $E_w(\mathbb{Q}(\sqrt{67}))$ due to the difficulty in searching for a point on these curves. Any elliptic curves with everywhere good reduction, if exist, will arise only from these three cases.*
### Conclusion

No elliptic curve over $\mathbb{Q}(\sqrt{78})$ with everywhere good reduction exists.

<table>
<thead>
<tr>
<th>Conjugate to #8</th>
<th>Conjugate to #7</th>
<th>Conjugate to #6</th>
<th>Conjugate to #5</th>
<th>Conjugate to #4</th>
<th>Conjugate to #3</th>
<th>Conjugate to #2</th>
<th>Conjugate to #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
<td>$216 + 24\epsilon$</td>
</tr>
</tbody>
</table>

#### Table 5.22: Finding elliptic curves over $\mathbb{Q}(\sqrt{78})$ with everywhere good reduction

\[ (I = (3)N, V^4 = 35 + 64\epsilon, 78 = \epsilon) \]
Table 5.23: Finding elliptic curves over $\mathbb{Q}(\sqrt{87})$ with everywhere good reduction

$(a = \sqrt{87}, \varepsilon = 28 + 3a, N(\varepsilon) = 1)$

<table>
<thead>
<tr>
<th>#</th>
<th>$\varepsilon$</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0,1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$(12, 0), 2)$</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>2</td>
<td>$-\varepsilon^3$</td>
<td>$((-336 - 36a, 0), 2)$</td>
<td>1</td>
<td>$(168 + 18a, 9234 + 990a)$</td>
<td>$(168 + 18a, 9234 + 990a)$, $(-336 - 36a, 0)$, $(336 + 36a, -12312 - 1320a)$, $(7728 + 828a, 960336 + 102960a)$</td>
<td>$-216, -1728$</td>
<td>$-$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$(12, 0), 2)$</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
<td>$(12, 0)$</td>
<td>$-$</td>
</tr>
<tr>
<td>4</td>
<td>$\varepsilon^4$</td>
<td>$(336 + 36a, 0), 2$</td>
<td>1</td>
<td>$(13388024, -1024)$, $(1504928, 155747313)$</td>
<td>$(336 + 36a, 0)$</td>
<td>$-$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\varepsilon^2$</td>
<td>$O$</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>6</td>
<td>$-\varepsilon$</td>
<td>$O$</td>
<td>1</td>
<td>$(217776 - 21924a, 6241, 158760128 - 168828036)$</td>
<td>$-$</td>
<td>$-$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$-\varepsilon^2$</td>
<td>$O$</td>
<td>0</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>8a</td>
<td>$\bar{\varepsilon}$</td>
<td>$O$</td>
<td>$\leq 1$</td>
<td>$?$</td>
<td>$?$</td>
<td>$?$</td>
<td>$?$</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Conjecture:** No elliptic curve over $\mathbb{Q}(\sqrt{87})$ with everywhere good reduction exists.

*a*We cannot find a Mordell–Weil basis for $E_w(\mathbb{Q}(\sqrt{87}))$ due to the difficulty in searching for a point on the curve. Any elliptic curve with everywhere good reduction, if exists, will arise only from this case.
Table 5.24: Finding elliptic curves over $\mathbb{Q}(\sqrt{95})$ with everywhere good reduction exists.

<table>
<thead>
<tr>
<th>Conclusion: No elliptic curve over $\mathbb{Q}(\sqrt{95})$ with everywhere good reduction exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccc}
\text{Integral points} & \text{Conclusion to} & \text{Conclusion to} & \text{Conclusion to} \\
\text{Rank} & \text{#} & \text{#} & \text{#} \\
\text{Unravelled points} & \text{MSC0101.01} & \text{MSC0101.02} & \text{MSC0101.03} \\
\text{Mordell-Weil basis} & \text{MSC0101.04} & \text{MSC0101.05} & \text{MSC0101.06} \\
\text{Conclusion} & \text{MSC0101.07} & \text{MSC0101.08} & \text{MSC0101.09} \\
\end{array}
\]

\[
\begin{align*}
& (3) \mathcal{N} \mathcal{M} + 394 \mathcal{M}^2 = 49, \\
& (95)^\mathcal{N} \mathcal{M} = 0
\end{align*}
\]
5.4.3 Examples II: Imaginary Quadratic Fields

In this subsection, we will illustrate a similar computation on finding elliptic curves with everywhere good reduction over $K = \mathbb{Q}(\sqrt{-d})$ for $2 \leq d \leq 100$. Most cases, as mentioned in another Cremona’s table\(^6\), have been already proved by various mathematicians, including:

- **Kida** [Kid01, Theorem 1] has proved non-existence of such curve for $d = 35, 37, 51, \text{ and } 91$.

- **Setzer** has proved non-existence for several $d$ up to 161; see [Set78, Theorem 4(a)] for the complete list. He has also showed existence of elliptic curves with everywhere good reduction for $d = 65$ [Set78, Theorem 4(b)].

- **Cremona** has used his joint method [CL07] to confirm non-existence of such curve for $d = 23$.

In general, we also have the following theorem due to Setzer.

**Theorem 5.4.5** ([Set78, Theorem 5]). If the class number of $K = \mathbb{Q}(\sqrt{-d})$ $(d > 0)$ is prime to 6, then there is no elliptic curve over $K$ with everywhere good reduction.

Using all the main results we have so far to assist in computing Mordell–Weil bases and integral points, we are eventually able to use Cremona and Lingham’s method to show non-existence of elliptic curves with everywhere good reduction over more imaginary quadratic fields in addition to the above results\(^7\). To be precise, we obtain the following conclusion from our tables to be shown in the following pages.

**Proposition 5.4.6.** For $d = 26, 29, 31, 59, 83, 87$, there is no elliptic curve defined over $\mathbb{Q}(\sqrt{-d})$ with everywhere good reduction.

**Proof.** See Table 5.25, 5.26, 5.27, 5.30, 5.32, 5.33; note that we cannot use Theorem 5.4.5 since these imaginary quadratic fields have class number not prime to 6. \(\square\)

---

\(^{6}\)Available at http://www.warwick.ac.uk/~masgaj/ecgr/ecgr_imag.txt (last checked on November 30, 2010).

\(^{7}\)Except at $d = 89$, due to the difficulty in finding Mordell–Weil bases for most $E_w(K)$. 
Table 5.25: Finding elliptic curves over $\mathbb{Q}(\sqrt{-26})$ with everywhere good reduction

\[
\begin{array}{|c|c|c|c|c|}
\hline
\# & \mathcal{O} & \text{Torsion} & \text{Rank} & \text{Mordell–Weil basis} \\
\hline
1 & (12,0) & 1 & 1 & \langle (12,0) \rangle \\
2 & (4,8a) & 2 & \langle (8a) \rangle & \langle (4,8a) \rangle, (12,0) \\
3 & (1 + 8a) & 1 & 0 & \langle (1 + 8a) \rangle \\
4 & (12,0) & 0 & 0 & \langle (12,0) \rangle \\
5 & (1 + 8a) & 0 & 0 & \langle (1 + 8a) \rangle \\
6 & (1 + 8a) & 0 & 0 & \langle (1 + 8a) \rangle \\
\hline
\end{array}
\]

Conclusion: No elliptic curve over $\mathbb{Q}(\sqrt{-26})$ with everywhere good reduction exists.

**Note:** $a = \sqrt{-26}$.
Table 5.26: Finding elliptic curves over $\mathbb{Q}(\sqrt{-29})$ with everywhere good reduction $(a = \sqrt{-29})$

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0,1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$(12,0), 2$</td>
<td>0</td>
<td>–</td>
<td>$(12,0)$</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>$-1$</td>
<td>$(-12,0), 2$</td>
<td>0</td>
<td>–</td>
<td>$(-12,0)$</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>$-721 + 20a$</td>
<td>$O$</td>
<td>1</td>
<td>$(356 + 80a, 3280 + 2368a)$</td>
<td>$(356 + 80a, 3280 + 2368a)$</td>
<td>224704 $-$ 15360a</td>
</tr>
<tr>
<td>4</td>
<td>$721 - 20a$</td>
<td>$O$</td>
<td>1</td>
<td>$(188 - 16a, -2192 + 352a)$</td>
<td>$(188 - 16a, -2192 + 352a)$</td>
<td>5056 $-$ 2048a</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>Conjugate to #4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Conclusion:** No elliptic curve over $\mathbb{Q}(\sqrt{-29})$ with everywhere good reduction exists.
### Table 5.27: Finding elliptic curves over $\mathbb{Q}(\sqrt{-31})$ with everywhere good reduction

<table>
<thead>
<tr>
<th>$#_w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell-Weil basis</th>
<th>Integral points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\langle (12,0), (-484 - 25, -2128 + 4256) \rangle$</td>
<td>$\langle (12,0), -\frac{30032476}{680625}, -\frac{29261304464 + 58522608928}{561515625} \rangle$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>$\langle (-12,0), (-30032476 - 680625, -29261304464 + 58522608928) \rangle$</td>
<td>$\langle (-12,0), -\frac{30032476}{680625}, -\frac{29261304464 + 58522608928}{561515625} \rangle$</td>
</tr>
</tbody>
</table>

**Conclusion:** No elliptic curve over $\mathbb{Q}(\sqrt{-31})$ with everywhere good reduction exists.
Table 5.28: Finding elliptic curves over $\mathbb{Q}(\sqrt{-38})$ with everywhere good reduction $(a = \sqrt{-38})$

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0, 1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\langle(12, 0), 2\rangle$</td>
<td>2</td>
<td>$(-20, 16a), (10, -\frac{18a}{2})$</td>
<td>$(-20, 16a), (12, 0)$, $(-1014, 5238a)$</td>
<td>$-8000$, $-1042590744$</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>$\langle(-12, 0), 2\rangle$</td>
<td>1</td>
<td>$(-\frac{103109}{291}, -\frac{86729046}{52721})$</td>
<td>$(-12, 0)$</td>
<td></td>
</tr>
<tr>
<td>3$^a$</td>
<td>$(8 - 2a)^2$</td>
<td>$O$</td>
<td>$\leq 1$</td>
<td>$?$</td>
<td>$?$</td>
<td>$?$</td>
</tr>
<tr>
<td>4</td>
<td>$-(8 - 2a)^2$</td>
<td>$O$</td>
<td>0</td>
<td>$?$</td>
<td>$?$</td>
<td>$?$</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Conjecture:** No elliptic curve over $\mathbb{Q}(\sqrt{-38})$ with everywhere good reduction exists.

$^a$We cannot find a Mordell–Weil basis for $E_w(\mathbb{Q}(\sqrt{-38}))$ due to the difficulty in searching for a point on the curve. Any elliptic curve with everywhere good reduction, if exists, will arise only from this case.
<table>
<thead>
<tr>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$\langle (12, 0) \rangle$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$\langle (12, 0), (12, 0) \rangle$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>$\langle (0, 12) \rangle$</td>
</tr>
</tbody>
</table>

Conjecture: No elliptic curve over $\mathbb{Q}(\sqrt{-53})$ with everywhere good reduction exists.

One can check using MAGMA that the rank of each $E$ is at most 3. By applying 2-descent twice (using MAGMA code implemented by Nils Bruin), currently we can show that 3 out of 7 non-trivial elements of the 2-Selmer group have homogeneous spaces with no rational points, hence $r \leq 2$.

Since we can find a non-torsion point, then $1 \leq r \leq 2$. Hence the Parity Conjecture is true, then $r = 1$, and each $E$ will be generated by $\mathcal{P}_1$ and $\mathcal{P}_2$.

Table 5.29: Finding elliptic curves over $\mathbb{Q}(\sqrt{-53})$ with everywhere good reduction
5.4. Elliptic Curves with Everywhere Good Reduction

Table 5.30: Finding elliptic curves over \( \mathbb{Q}(\sqrt{-59}) \) with everywhere good reduction

\[ (a = \frac{1 + \sqrt{-59}}{2}) \]

<table>
<thead>
<tr>
<th>#</th>
<th>( w )</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>( j \neq 0, 1728 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>((12, 0), 2)</td>
<td>1</td>
<td>((24 + 4a, 96 + 32a))</td>
<td>((24 + 4a, 96 + 32a), (12, 0), (28 - 4a, -128 + 32a))</td>
<td>(-4416 + 7168a), (-)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>((-12, 0), 2)</td>
<td>1</td>
<td>((24 + 4a, 96 + 32a))</td>
<td>((24 + 4a, 96 + 32a), (12, 0), (28 - 4a, -128 + 32a))</td>
<td>(-4416 + 7168a), (-)</td>
</tr>
<tr>
<td>3</td>
<td>(-6 + 7a)</td>
<td>(O)</td>
<td>2</td>
<td>((21 + 13a, -123 + 112a), (-24 - 8a, 312 - 40a))</td>
<td>((21 + 13a, -123 + 112a), (-24 - 8a, 312 - 40a), (48 + 16a, -120 - 184a), (12 - 8a, -96 + 40a))</td>
<td>(-671 + 1785a, -1536 - 512a, 12288 + 4096a, 832 + 192a)</td>
</tr>
<tr>
<td>4a</td>
<td>(-6 - 7a)</td>
<td>(O)</td>
<td>0</td>
<td>(-)</td>
<td>(-)</td>
<td>(-)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Conclusion:** No elliptic curve over \( \mathbb{Q}(\sqrt{-59}) \) with everywhere good reduction exists.

\( ^a \)One can check using MAGMA that the rank \( r \) of \( E_w(K) \) is at most 2. By applying 2-descent twice, we can show that only one non-trivial case exists, hence \( r = 0 \).
We cannot find Mordell–Weil bases for both $E$ due to the difficulty in searching for a point on both curves. Any elliptic curves with everywhere good reduction, if exist, will arise only from these two cases.

**Table 5.31: Finding elliptic curves over $\mathbb{Q}(\sqrt{-61})$ with everywhere good reduction**

<table>
<thead>
<tr>
<th>Torsion Rank</th>
<th>Mordell–Weil Basis</th>
<th>Integral Points</th>
<th>$a$</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(12, 0)</td>
<td>$(-12, 0)$</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Conjecture: No elliptic curve over $\mathbb{Q}(\sqrt{-61})$ with everywhere good reduction exists.
Table 5.32: Finding elliptic curves over $\mathbb{Q}(\sqrt{-83})$ with everywhere good reduction

\[(a = \frac{1 + \sqrt{-83}}{2})\]

<table>
<thead>
<tr>
<th>#</th>
<th>$w$</th>
<th>Torsion</th>
<th>Rank</th>
<th>Mordell–Weil basis</th>
<th>Integral points</th>
<th>$j \neq 0,1728$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>$\langle (12,0), 2 \rangle$</td>
<td>1</td>
<td>$\langle 4a, -48 + 16a \rangle$</td>
<td>$(4a, -48 + 16a), (12,0), \ (4 - 4a, 32 + 16a)$</td>
<td>$-1344 - 1280a, -2624 + 1280a$</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>$\langle (-12,0), 2 \rangle$</td>
<td>1</td>
<td>$\langle \frac{9535415778601181}{2}, \frac{989480472999844a}{2}, \frac{989480472999844a}{2} \rangle$</td>
<td>$(-12,0)$</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>$-12 - 5a$</td>
<td>$O$</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
| 4 | $12 + 5a$ | $O$ | 2 | $\langle -1740 + 76a, 13872 + 15176a \rangle$, \ 
\( (96 + 4a, 912 + 56a) \rangle$ | $\langle -1740 + 76a, 13872 + 15176a \rangle$, \ 
\( (96 + 4a, 912 + 56a) \rangle$ | $-14480448 + 42574400a$ |

**Conclusion:** No elliptic curve over $\mathbb{Q}(\sqrt{-83})$ with everywhere good reduction exists.
Table 5.33: Finding elliptic curves over $\mathbb{Q}(\sqrt{-87})$ with everywhere good reduction

<table>
<thead>
<tr>
<th>#</th>
<th>Conjugate to $\mathbb{Q}(\sqrt{-87})$</th>
<th>$\mathbb{Q}(\sqrt{-87})$</th>
<th>$\mathbb{Q}(\sqrt{-87})$</th>
<th>$\mathbb{Q}(\sqrt{-87})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(12,0)$</td>
<td>$-66 - 3a - 138$</td>
<td>$-93 - 21a$</td>
<td>$-648 - 6a$</td>
</tr>
<tr>
<td>2</td>
<td>$(12,0)$</td>
<td>$-66 + 3a + 138$</td>
<td>$-93 + 21a$</td>
<td>$-648 + 6a$</td>
</tr>
<tr>
<td>3</td>
<td>$(12,0)$</td>
<td>$-66 + 3a + 138$</td>
<td>$-93 + 21a$</td>
<td>$-648 + 6a$</td>
</tr>
<tr>
<td>4</td>
<td>$(12,0)$</td>
<td>$-66 - 3a - 138$</td>
<td>$-93 - 21a$</td>
<td>$-648 - 6a$</td>
</tr>
</tbody>
</table>

Conclusion: No elliptic curve over $\mathbb{Q}(\sqrt{-87})$ with everywhere good reduction exists.

One can check using MAGMA that the rank $r$ of $E_w(K)$ is at most 2. By applying 2-descent twice, we can show that only one non-trivial case exists.

Note that the sieving procedure fails at $p = 29$ in which case we prove that $r = 0$. Hence $r = 0$.

One can check using MAGMA that the rank $r$ of $E_w(K)$ is at most 2. By applying 2-descent twice, we can show that only one non-trivial case exists.

One can check using MAGMA that the rank $r$ of $E_w(K)$ is at most 2. By applying 2-descent twice, we can show that only one non-trivial case exists.
Appendix A

MAGMA Source Code

As mentioned earlier, we have implemented our algorithms based on the main results of this thesis in MAGMA. For convenience, we shall split our source code into several files according to their applications.

A.1 Period Lattices and Complex Elliptic Logarithms

We have implemented all necessary functions for computing complex arithmetic-geometric mean (AGM), period lattices of elliptic curves over \( \mathbb{C} \), and elliptic logarithms of complex points (see Chapter 4). In the following file, some important functions include:

- \texttt{AGM()} This function computes an AGM of two complex numbers based on a specified set of all indices for which the pair in the AGM sequence is bad.

- \texttt{PeriodLattice()} Given an elliptic curve \( E/\mathbb{C} \), this function will compute all three minimal coset representatives of \( \Lambda \) modulo \( 2\Lambda \), where \( \Lambda \) is the period lattice of \( E \). Any two of these minimal coset representatives form a \( \mathbb{Z} \)-basis for \( \Lambda \).
EllipticLog() Given an elliptic curve $E/\mathbb{C}$ and a point $P \in E(\mathbb{C})$, this function computes an elliptic logarithm of $P$.

For more details on these functions, see the documentation inside the code.
A.1. Period Lattices and Complex Elliptic Logarithms

return a;
end intrinsic;

/**
* Given a, b, compute an optimal AGM(a, b) (i.e. no bad pair allowed in the
* AGM sequence for all n>0)
* Input:
* a, b = two complex numbers
* Output:
* An optimal AGM of a, b, subject to |a_n - b_n| < 10^-Prec
* Parameter:
* Prec = precision used as the stopping criterion
**/
intrinsic AGM(a::FldComElt, b::FldComElt : Prec := 25) -> FldComElt
{Compute an optimal AGM of a, b.}
return AGM(a, b, [], : Prec := Prec);
end intrinsic;

/**
* Given an elliptic curve E: Y^2 = 4*(X-e1)*(X-e2)*(X-e3) with e_j distinct
* and e1+e2+e3=0, compute the three minimal coset representatives of
* \Lambda modulo 2*\Lambda, where \Lambda is the period lattice of E.
* Any two of them form a \Z-basis for \Lambda.
* Note: Users may need to do an extra work to obtain an orthogonal basis
* in case of a rectangular lattice.
* Input:
* E = [e1, e2] = two roots of E (note that e3 = -e1-e2)
* Output:
* [w1, w2, w3] = the three minimal coset representatives
* Parameter:
* Prec = precision used as the stopping criterion when computing AGM
**/
intrinsic PeriodLattice(E::SeqEnum : Prec := 25) -> SeqEnum
{Given E = [e1,e2], compute all three minimal coset representatives; two of which
form a \Z-basis for the period lattice of the elliptic curve Y^2=4*(X-e1)*(X-e2)*(X-e3),
with e1+e2+e3=0.}
require #E eq 2: "E must contain exactly two complex numbers";
require (Prec ge 0) and (Prec in Integers()):
"Precision must be a non-negative integer";
e1, e2 := Explode(E);
require ei ne e2: "All roots of E must be distinct";
require Precision(e1) eq Precision(e2): "e1,e2 must have the same precision";
e3 := -e1-e2;
C := Parent(e1); i := C!Sqrt(-1);
a := Sqrt(e1-e3);
b := Sqrt(e1-e2);
c := Sqrt(e2-e3);
// Rearrange e1, e2, e3 (and thus redefine a, b, c) if necessary to make
// w1, w2, w3 satisfy the 3-term relation |w1-w2-w3| = 0.
// First, check if there is an equality among a,b,c (can happen at most once)
equality := false;
if Abs(a-b) eq Abs(a+b) then
  equality := true;
  if Abs(c-i*b) gt Abs(c+i*b) then
    b := -b;
    end if;
  if Abs(a-c) gt Abs(a+c) then
    a := -a;
    end if;
else
  if Abs(c-i*b) eq Abs(c+i*b) then
    equality := true;
    if Abs(a-b) gt Abs(a+b) then
      b := -b;
      end if;
    elseif Abs(c-i*b) eq Abs(c+i*b) then
      equality := true;
      if Abs(a-b) gt Abs(a+b) then
        b := -b;
        end if;
      end if;
  end if;
end if;
if Abs(a-c) gt Abs(a+c) then

c := -c;
end if;

elif Abs(a-c) eq Abs(a+c) then
  equality := true;
  if Abs(a-b) gt Abs(a+b) then
    a := -a;
  end if;
  if Abs(c-i*b) gt Abs(c+i*b) then
    c := -c;
  end if;
end if;

// No equality from this point, choose the sign of a arbitrarily
// and choose b, c to satisfy other conditions
if not equality then
  if Abs(a-b) gt Abs(a+b) then
    b := -b;
  end if;
  if Abs(a-c) gt Abs(a+c) then
    c := -c;
  end if;
  if Abs(c-i*b) gt Abs(c+i*b) then
    // Change the order to (e3, e2, e1);
    vprintf Elog: "Changing the order of e1, e2, e3 ...";
    e1 := e3;
    vprintf Elog: "new e1 = ", e1;
    return PeriodLattice([e1, e2] : Prec := Prec);
  end if;
end if;

// Now a, b, c are valid, proceed to compute w1, w2, w3
vprintf Elog: "a = %o
b = %o
c = %o
", a, b, c;
pi := Pi(C);
w1 := pi/AGM(a, b : Prec := Prec);
w2 := pi/AGM(c, i*b : Prec := Prec);
w3 := i*pi/AGM(a, c : Prec := Prec);

vprintf Elog: "w1 = %o
w2 = %o
w3 = %o
", w1, w2, w3;

// Test relationship among w1, w2, w3;
disc := Abs(w1-w2-w3);

vprintf Elog: "|w1 - w2 - w3| = %o: ", disc;
if disc lt 10^(-Prec) then
  vprintf Elog: "OK"
else;
  return [w1, w2, w3];
end intrinsic;

/**
 * Given an elliptic curve E by a Weierstrass model over complex numbers,
 * compute the three minimal coset representatives of \Lambda modulo
 * 2*\Lambda, where \Lambda is the period lattice of E. Any two of them form
 * a \Z-basis for \Lambda.
 * Input:
 * \ E = an elliptic curve over complex numbers
 * Output:
 * \ [w1, w2, w3] = the three minimal coset representatives
 * Parameter:
 * \ Prec = precision used as the stopping criterion when computing AGM
 */
intrinsic PeriodLattice(E::CrvEll : Prec := 25) -> SeqEnum
(Given an elliptic curve E by a Weierstrass model defined over complex numbers,
compute all three minimal coset representatives; two of which form a \Z-basis
for the period lattice of E.)

  require Type(BaseRing(E)) eq FldCom: "E must be defined over complex numbers";
  // Transform E into the form Y^2 = 4*(X-e1)*(X-e2)*(X-e3) with e1+e2+e3=0
  e1, e2 := Explode(TransformModel(E));
  return PeriodLattice([e1, e2] : Prec := Prec);
end intrinsic

/**
A.1. Period Lattices and Complex Elliptic Logarithms

• Compute an elliptic logarithm of complex points on elliptic curves of the form \( Y^2 = 4*(X-e1)*(X-e3)*(X-e3) \) with \( e1+e2+e3=0 \).

• Input:
  * \( E = [e1, e2] \)
  * \( P = [x,y] = \) a point on \( E \)

• Output:
  * an elliptic logarithm of \( P \)

• Parameter:
  * \( \text{Prec} = \) precision used as the stopping criterion when computing AGM

/**
 intrinsic EllipticLog(E::SeqEnum, P::SeqEnum : Prec := 25) -> FldComElt
(Compute an elliptic logarithm of a point \( P = [x,y] \) on an elliptic curve of the form \( Y^2 = 4*(X-e1)*(X-e3)*(X-e3) \) with \( e1+e2+e3=0 \).)
// Verify if inputs are valid
require #E eq 2: "E must contain exactly 2 complex numbers";
require #P eq 2: "P must contain exactly 2 complex numbers";
require (Prec ge 0) and (Prec in Integers()): "Precision must be a non-negative integer";
e1, e2 := Explode(E);
require e1 ne e2: "All roots of E must be distinct";
require Precision(e1) eq Precision(e2): "e1,e2 must have the same precision";
require Precision(e1) gt Prec: "Prec must be less than the precision of E";
e3 := -e1-e2;
x, y := Explode(P);
require Precision(x) eq Precision(y): "x, y must have the same precision";
require Precision(x) eq Precision(e1): "Precision of E and its point must be the same";
if Round(Abs(y^2 / (4*(x-e1)*(x-e2)*(x-e3)))) ne 1 then
  error "P is not on E";
end if;
a := Sqrt(e1 - e3); b := Sqrt(e1 - e2);
u := Sqrt(x - e3); v := Sqrt(x - e2);
// Use two strongly optimal sequences - choose both (a,b), (u,v) to be good
// If equality holds, either choice will do
if Abs(a-b) gt Abs(a+b) then
  b := -b;
end if;
if Abs(u-v) gt Abs(u+v) then
  v := -v;
end if;
// Define t
if x in [e1, e2, e3] then
  t := Sqrt(x - e1);
else
  t := y/(2*u*v);
end if;
// Special case: t = 0
if t eq 0 then
  return Pi(Parent(e1))/(2*AGM(a, b: Prec := Prec));
end if;
n := 0;
repeat
  // Compute new a, b
  new_a := (a + b) / 2; new_b := Sqrt(a*b);
a := new_a; b := new_b;
  // Optimal sequence: choose right choice at every step
  if Abs(a-b) gt Abs(a+b) then
    b := -b;
  end if;
  c := Sqrt(a^2 - b^2);
  // Compute new u, v, t
  u := (u + v)/2; v := Sqrt(u^2 - c^2);
  // Optimal sequence: choose right choice at every step
  if Abs(u-v) gt Abs(u+v) then
    v := -v;
  end if;
  t := u*t/v;

n := n + 1;
vprintf Elog: \"|a_{%o}-b_{%o}| = %o\n\", n, n, Abs(a-b);
vprintf Elog: \"|u_{%o}-v_{%o}| = %o\n\", n, n, Abs(u-v);
until Abs(a - b) lt (10^-Prec);
// xinf := t^2 + 2/3*(a^2); yinf := 2*t*(t^2 + a^2);
return -Arctan(a/t)/a;
end intrinsic;

/**
 * Compute an elliptic logarithm of complex points on elliptic curves
 * given by a Weierstrass equation [a1,a2,a3,a4,a6]
 * Input:
 * E = an elliptic curve in standard Weierstrass form
 * P = a point on E
 * Output:
 * an elliptic logarithm of P
 * Parameter:
 * Prec = precision used as the stopping criterion when computing AGM
 **/
intrinsic EllipticLog(E::CrvEll, P::PtEll : Prec := 25) -> FldComElt
{Compute an elliptic logarithm of a point P on E, where E is an elliptic
curve over C given by a Weierstrass equation.}
if P eq Identity(E) then
  return BaseRing(E)!0;
end if;
newE, newP := TransformModel(E, P);
return EllipticLog(newE, newP : Prec := Prec);
end intrinsic;

/**************************************************************************/
* Auxiliary intrinsic functions
**************************************************************************/
/**
 * Transform an elliptic curve given by a Weierstrass model [a1,a2,a3,a4,a6]
 * (defined over C) into the form E': Y^2 = 4*(X-e1)*(X-e2)*(X-e3) with
 * e1+e2+e3=0.
 * Input:
 * E = an elliptic curve of the form [a1, a2, a3, a4, a6]
 * Output:
 * [e1, e2]
 **/
intrinsic TransformModel(E::CrvEll) -> SeqEnum
{Transform an elliptic curve over C given by a Weierstrass model [a1,a2,a3,a4,a6]
into the form E': Y^2 = 4*(X-e1)*(X-e2)*(X-e3) with e1+e2+e3=0.}
C := BaseRing(E);
require Type(C) eq FldCom: "E must be defined over complex numbers";
_<t> := PolynomialRing(C);
c4, c6 := Explode(cInvariants(E));
R := Roots(t^3 - (c4/48)*t - c6/864);
e1 := R[1][1]; e2 := R[2][1];
return [e1, e2];
end intrinsic;

/**
 * Transform an elliptic curve given by a Weierstrass model [a1,a2,a3,a4,a6]
 * (defined over C) into the form E': Y'2 = 4*(X-e1)*(X-e2)*(X-e3) with e1*e2*e3=0.
 * Input:
 * E = an elliptic curve of the form [a1, a2, a3, a4, a6]
 * P = a point on E
 * Output:
 * [e1, e2], [X,Y], where [X,Y] is the image of P on E'
 **/
intrinsic TransformModel(E::CrvEll, P::PtEll) -> SeqEnum, SeqEnum
{Transform an elliptic curve over C given by a Weierstrass model [a1,a2,a3,a4,a6]
into the form E': Y'2 = 4*(X-e1)*(X-e2)*(X-e3) with e1*e2*e3=0, and map a point
P on E to its image on E'.
require \( P \neq \text{Identity}(E) \): "\( P \) must not be the point at infinity";
newE := TransformModel(E);
\( a_1, \ldots, a_3, \ldots \) := Explode(aInvariants(E));
b2 := bInvariants(E)[1];
\( X := P[1] + b2/12; \)
\( Y := 2*P[2] + a1*P[1] + a3; \)
return newE, \([X, Y]\);
end intrinsic;

/**
 * Reduce any given \( z \) into the one inside the fundamental parallelogram
 * spanned by \( L = [w_1, w_2] \), and return \([a, b]\) such that
 * reduced \( z = z' = a*w_1 + b*w_2 \), with \( 0 \leq a, b < 1 \)
 * Input:
 * \( L = [w_1, w_2] \) = fundamental parallelogram
 * \( z \) = the complex number to be reduced
 * Output:
 * \( z' \) = the reduced version of \( z \) inside the parallelogram
 * \([a, b]\) = coordinates of new \( z \) on the parallelogram
 **/
intrinsic Reduce2FP(L::SeqEnum, z::FldComElt) -> FldComElt, SeqEnum
(Reduce \( z \) modulo the lattice \( L \))
require \#L eq 2: "\( L \) must have exactly two numbers";
\( w_1, w_2 := \text{Explode}(L); \)
\( \tau := w_2/w_1; \)
\( \text{IsDenomW1 := true; if Im(\tau) eq 0 then error "w2/w1 must not be real"; elseif Im(\tau) lt 0 then } \)
\( \tau := 1/\tau; \)
\( \text{IsDenomW1 := false; denom := w_2; end if;} \)
\( \text{// Reduce } z \text{ into the fundamental parallelogram } \)
\( z := z/denom; \)
\( \text{if IsDenomW1 then } \)
\( \beta := \text{Im}(z)/\text{Im}(\tau); \)
\( \alpha := \text{Re}(z - \beta*\tau); \) // already real, just put it to avoid error
\( z := z - \text{Floor}(\beta)*\tau - \text{Floor}(\alpha); \)
\( \text{else } \)
\( \alpha := \text{Im}(z)/\text{Im}(\tau); \)
\( \beta := \text{Re}(z - \alpha*\tau); \) // already real, just put it to avoid error
\( z := z - \text{Floor}(\alpha)*\tau - \text{Floor}(\beta); \)
\( \text{end if; } \)
\( \beta := \beta - \text{Floor}(\beta); \)
\( \alpha := \alpha - \text{Floor}(\alpha); \)
\( z := z*\text{denom}; \)
return \( z, [\alpha, \beta] \);
end intrinsic;

/**
 * Given a period lattice \( L = \langle w_1, w_2 \rangle \), apply linear transformation in \( \text{SL}(2, \mathbb{Z}) \)
 * so that we obtain a new basis \( \{w_1', w_2'\} \) with \( \tau = w_2'/w_1' \) in the
 * fundamental domain
 * (Based on Algorithm 7.4.2 in Cohen's "A Course in Computational Algebraic
 * Number Theory")
 * Input:
 * \( L = [w_1, w_2] \)
 * Output:
 * \( \{w_1', w_2'\} \) = a new basis with the above property
 * \( A \) = transformation matrix where \( [w_2' w_1']^T = A*[w_2 w_1]^T \)
 **/
intrinsic TransformLattice(L::SeqEnum) -> SeqEnum, Mtrx
(Apply linear transformation to a basis for a lattice so that the new basis
\( \{w_1', w_2'\} \) has \( \tau = w_2'/w_1' \) in the fundamental domain of \( \text{SL}(2, \mathbb{Z}) \))
\( w_1, w_2 := \text{Explode}(L); \)
\( \text{tau := w2/w1; } \)
A.2 Integral Points on Elliptic Curves

The following file is our implementation (with some modifications) of Smart and Stephens’ algorithm [SS97] for finding integral points on elliptic curves over number fields. The only main function in this file is `IntegralPoints()`; see the documentation inside the code for more details.

```magma
require Im(tau) gt 0: "Im(w2/w1) must be positive";
A := Matrix(2, [1,0,0,1]);
while true do
    n := Round(Re(tau));
    tau -:= n;
    A := Matrix(2, [1,-n,0,1]) * A;
    if Abs(tau) ge 1 then
        break;
    end if;
    tau := -1/tau;
    A := Matrix(2, [0,-1,1,0]) * A;
end while;
new_w2 := A[1,1]*w2 + A[1,2]*w1;
new_w1 := A[2,1]*w2 + A[2,2]*w1;
return [new_w1, new_w2], A;
end intrinsic;
```
A.2. Integral Points on Elliptic Curves

• \( L \) is a sequence of points on elliptic curves over number field
• Output:
  • minimum eigenvalue of the height paring matrix
/**
function c3(L)
  \( M := \text{Eigenvalues} (\text{HeightPairingMatrix}(L)) \);
  \( M := \text{SetToSequence}(M) \);
  \( M := [m[1] : m \in M] \); // ignore multiplicity
  \( M := \text{Minimum}(M) \); // least eigenvalue
  return M;
end function;
/**
• Compute constant \( c_6 \)
  • Input:
    • \( E \) is an elliptic curve over real/complex numbers in standard Weierstrass form
  • Require:
    • \text{elog.m}
/**
function c6(E)
  \( R := \text{TransformModel}(E) \);
  \( R := [\text{Abs}(r) : r \in R] \);
  \( R := 2*\text{Maximum}(R) \);
  return R;
end function;
/**
• Compute constant \( c_8 \), and the periods \( w_1, w_2 \) of the period lattice of a
  • given elliptic curve \( E \) (with \( w_2/w_1 \) in the fundamental domain).
• Input:
  • \( E \) is an elliptic curve over real/complex numbers in standard Weierstrass form
• Output:
  • \( L \) is the constant \( c_8 \)
  • \([w_1, w_2] = \) periods of \( E \)
• Parameter:
  • \( \text{Prec} = \) precision for computing periods
  • Require:
    • \text{elog.m}
/**
function c8(E : Prec := 25)
  w1, w2, _ := \text{Explode}(\text{PeriodLattice}(E : \text{Prec} := \text{Prec})) ;
  if \( \text{Im}(w_2/w_1) \geq 0 \) then
    \( L := [w_1, w_2] \);
  else
    \( L := [w_2, w_1] \);
  end if;
  // Apply transformation by \text{SL(2,Z)} so that \( \tau = w_2/w_1 \) is in the
  // fundamental domain
  \( L, _ := \text{TransformLattice}(L) ;\)
  \( \text{wList} := [L[1], L[2], L[1]+L[2]] ;\)
  \( \text{wList} := [\text{Abs}(w) : w \in \text{wList}] ;\)
  \( \gamma := \text{Maximum} (\text{wList}) ;\)
  return \( \gamma, L \);
end function;
/**
• Compute absolute logarithmic height of an element in \( n \)-projective space
  • over a number field \( K \)
• Input:
  • \( X \) is a sequence of elements in a number field \( K \)
/**
function AbsLogHeight(X)
  // \( X \) cannot be zero vector
  if \( \#X \neq \#[x : x \in X | x \neq 0] \) then
    \text{error} "\( X \) cannot be zero vector";
  end if;
//
// Find all prime ideal that divides some denominators of x_i
I := \{
K := Parent(X[1]); // assume each x_i is in the same field
O := RingOfIntegers(K);
for x in X do
  den := O ! Denominator(x);
  L := Decomposition(den);
  L := \{l[1] : l in L\}; // ignore multiplicity
  I := I join L;
end for;

h := 0;
// Non-archimedean contributions
for p in I do
  M := \{Rationals()\};
  for x in X do
    if x eq 0 then
      Append(~M, 0);
    else
      Append(~M, Norm(p)^(-Valuation(x, p)));
    end if;
  end for;
  M := Maximum(M);
  M := Log(M);
  h += M;
end for;

// Archimedean contributions
s1, s2 := Signature(K);
// M = \{log max(|x_1|_v, ..., |x_n|_v) : v in M_K\}
M := \{
for x in X do
  C := Conjugates(x);
  newC := \{
  // Real embedding contributions
  for i := 1 to s1 do
    Append(~newC, Abs(C[i]));
  end for;
  // Complex embedding contributions
  for i := 1 to s2 do
    Append(~newC, Abs(C[s1+(2*i-1)])^2);
  end for;
  if #M eq 0 then
    M := newC;
  else
    for i := 1 to (s1+s2) do
      if newC[i] gt M[i] then
        M[i] := newC[i];
      end if;
    end for;
    M := [Log(m) : m in M];
  end if;
end for;

// Overall absolute logarithmic height
h += (&*M);
h /= Degree(K);
return h;
end function;

**********************************************************************************
* Auxiliary local functions II:
* Computing constants for David's lower bound on linear form in logarithms.
* Notation used as in Appendix A of Smart's book
* "The Algorithmic Resolution of Diophantine Equations", with c's being
* replaced by d's
**********************************************************************************
* Compute the "height" of elliptic curve
* Input:
  * E = elliptic curve over number field in standard Weierstrass form

```plaintext
/**
 * Compute the "height" of elliptic curve
 * Input:
 * E = elliptic curve over number field in standard Weierstrass form
 **/
function h_E(E)
  j := jInvariant(E);
  C4, C6 := Explode(cInvariants(E));
  g2 := C4/12;
  g3 := C6/216;
  m := Maximum([1, AbsLogHeight([1, g2, g3]), AbsLogHeight([1, j])]);
  return m;
end function;
```

* Compute the list of modified height of a point P in E(K)
* depending on embedding
* Input:
  * E = elliptic curve over a number field K in standard Weierstrass form
  * P = a point in E(K)
  * ElogEmbedP = elliptic logarithm of the image of P in some embedding
  * D7 = constant d7 (depending on embedding)

```plaintext
/**
 * Compute the list of modified height of a point P in E(K)
 * depending on embedding
 * Input:
 * E = elliptic curve over a number field K in standard Weierstrass form
 * P = a point in E(K)
 * ElogEmbedP = elliptic logarithm of the image of P in some embedding
 * D7 = constant d7 (depending on embedding)
 **/
function h_m(E, P, ElogEmbedP, D7)
  K := BaseRing(E);
  L := [Height(P), h_E(E), D7/Degree(K)*Abs(ElogEmbedP)^2];
  L := Maximum(L);
  return L;
end function;
```

* Compute two extra h_m's based on the two periods
  * Similar to h_m, but now ElogEmbedP becomes a period of the fundamental parallelogram of some embedding of E
  * Input:
    * E = elliptic curve over a number field in standard Weierstrass form
    * Periods = [w1, w2] = period lattice of some real/complex embedding of E
    * D7 = constant d7 (depending on embedding)

```plaintext
/**
 * Compute two extra h_m's based on the two periods
 * Similar to h_m, but now ElogEmbedP becomes a period of the fundamental parallelogram of some embedding of E
 * Input:
 * E = elliptic curve over a number field in standard Weierstrass form
 * Periods = [w1, w2] = period lattice of some real/complex embedding of E
 * D7 = constant d7 (depending on embedding)
 **/
function Extra_h_m(E, Periods, D7)
  D := Degree(BaseRing(E));
  h := h_E(E);
  h1 := Maximum([0, h, D7/D*Abs(Periods[1])^2]);
  h2 := Maximum([0, h, D7/D*Abs(Periods[2])^2]);
  return [h1, h2];
end function;
```

* Compute constant d8
* Input:
  * E = elliptic curve over a number field K in standard Weierstrass form
  * L = a sequence of points in E(K) (e.g. Mordell-Weil basis)
  * Elog = a sequence of (pre-computed) elliptic logarithms of points in L on some fixed embedding
  * Periods = [w1, w2] = period lattice of some embedding of E
  * D7 = constant d7 (depending on embedding)

```plaintext
/**
 * Compute constant d8
 * Input:
 * E = elliptic curve over a number field K in standard Weierstrass form
 * L = a sequence of points in E(K) (e.g. Mordell-Weil basis)
 * Elog = a sequence of (pre-computed) elliptic logarithms of points in L on some fixed embedding
 * Periods = [w1, w2] = period lattice of some embedding of E
 * D7 = constant d7 (depending on embedding)
 **/
function d8(E, L, Elog, Periods, D7)
  C := [Exp(1)*h_E(E)];
  D := Degree(BaseRing(E));
  for i := 1 to #L do
    Append(~C, h_m(E, L[i], Elog[i], D7)/D);
  end for;
  C := C cat [t/D : t in Extra_h_m(E, Periods, D7)];
  C := Maximum(C);
  return C;
end function;
```
/**
* Compute constant d9
* Input:
* E = elliptic curve over a number field in standard Weierstrass form
* L = a sequence of points in E(K) (e.g. Mordell-Weil basis)
* Elog = a sequence of (pre-computed) elliptic logarithms of points in L
* on some fixed embedding
* Periods = [w1, w2] = period lattice of some embedding of E
* D7 = constant d7 (depending on embedding)
**/
function d9(E, L, Elog, Periods, D7)
    D := Degree(BaseRing(E));
    C := [1];
    for i := 1 to #L do
        tmp := Exp(1) * Sqrt(D * h_m(E, L[i], Elog[i], D7) / D7);
        tmp /= Abs(Elog[i]);
        C[i] := tmp;
    end for;
    // Take minimum among extra_h_m
    Eh_m := Extra_h_m(E, Periods, D7);
    tmp1 := Exp(1) * Sqrt(D*Eh_m[1]/D7) / Abs(Periods[1]);
    tmp2 := Exp(1) * Sqrt(D*Eh_m[2]/D7) / Abs(Periods[2]);
    C := C cat [tmp1, tmp2];
    C := Minimum(C);
    return C;
end function;

/**
* Compute constant d10
* Input:
* E = elliptic curve over a number field in standard Weierstrass form
* L = a sequence of points in E(K) (e.g. Mordell-Weil basis)
* Elog = a sequence of (pre-computed) elliptic logarithms of points in L
* on some fixed embedding
* Periods = [w1, w2] = period lattice of some embedding of E
* D7 = constant d7 (depending on embedding)
**/
function d10(E, L, Elog, Periods, D7)
    D := Degree(BaseRing(E));
    n := #L+2;
    scalar := 2 * 10^(8 + 7*n) * (2/Exp(1))^(2*n^2);
    scalar *= (n+1)^(4*n^2 + 10*n) * D^(2*n + 2);
    scalar *= (Log(d9(E, L, Elog, Periods, D7)))^(-2*n-1);
    for i := 1 to #L do
        scalar *= h_m(E, L[i], Elog[i], D7);
    end for;
    scalar *= &*(Extra_h_m(E, Periods, D7));
    return scalar;
end function;

/**
* Compute the right-hand side of the Principal Inequality
* Input:
* D = Degree(K), where K = number field
* r = rank(E(K))
* C9 = constant c9
* C10 = constant c10
* D9 = constant d9
* D10 = constant d10
* h = h(E), where E = elliptic curve over K
* Q = initial bound for the coefficients of P_i's, where P_i's are points
* in a Mordell-Weil basis for E(K)
* expTors = exponent of the torsion subgroup of E(K)
**/
function RHS(D, r, C9, C10, D9, D10, h, Q, expTors)
    bound := (Log(Log(Q*r*expTors)) + h + Log(D*D9))^*(r+3);
    bound *= D10*(Log(Q*r*expTors) + Log(D*D9));
    bound *= Log(C9*expTors);
bound /:= C10;
return bound;
end function;

/**
 * Approximate initial bound on Q = max_{1 <= i <= r}(|q_i|)
 * Input:
 * D = Degree(K), where K = number field
 * r = rank of E(K)
 * Q0 = constant Q0 (in S&S paper, this is called K0)
 * C9 = constant c9
 * C10 = constant c10
 * D8 = constant d8 (from function d8())
 * D9 = constant d9 (from function d9())
 * D10 = constant d10 (from function d10())
 * h = h_E(E)
 * expTors = exponent of the torsion subgroup of E(K)
 * 
 * Revised: 05 May 2009
 **/
function InitialQ(D, r, Q0, C9, C10, D8, D9, D10, h, expTors)
minQ := Maximum(Q0, Exp(D8));

// Try to approximate Q such that Q^2 = RHS(Q) (i.e. Q makes both sides
// of the Principal Inequality equal)
// Firstly, set a guess for Q, say minQ + 1 (so that Q > minQ)
// For simplicity, let's round Q up to the nearest power of 10
Q := minQ + 1;
x := Ceiling(Log(10, Q)); // x = log_10(Q)
// Check if Q satisfies the Principal Inequality, i.e. if Q^2 < RHS(Q)
// If so, repeat with the larger Q until we find the first Q that
// violates the Principal Inequality
// N.B. This loop will eventually terminate
exp_OK := 0; // the exponent that satisfies P.I.
exp_fail := 0; // the exponent that fails P.I.
while 10^(2*x) lt RHS(D, r, C9, C10, D9, D10, h, 10^x, expTors) do
  exp_OK := x; // Principal Inequality satisfied
  x *= 2; // double x, and retry
end while;
exp_fail := x; // x that fails the Principal Inequality
// So now x = log_10(Q) must lie between exp_OK and exp_fail
// Refine x further using "binary search"
repeat
  x := Floor((exp_OK + exp_fail)/2);
  if 10^(2*x) ge RHS(D, r, C9, C10, D9, D10, h, 10^x, expTors) then
    exp_fail := x;
  else
    exp_OK := x;
  end if;
until (exp_fail - exp_OK) le 1;
return exp_fail; // over-estimate
end function;

/**
 * Reduce the bound Q by LLL reduction until no further improvement
 * is possible. This function initially requires high precision to
 * proceed, although this should be done automatically by now
 * Input:
 * Pts = sequence of points in E(K)
 * j = j-th embedding (based on the index used in Conjugates())
 * EmbedL = a sequence of points on EmbedE (e.g. points in a Mordell-Weil
 * basis when embedded into EmbedE)
 * Elog = a sequence of (pre-computed) elliptic logarithms of points in EmbedL
 * C9 = constant c9
 * C10 = constant c10
 * Periods = [w1, w2] = period lattice of EmbedE
 * expTors = exponent of the torsion subgroup of E(K), K = number field
**/
/* initQ = initial guess for Q to be reduced by LLL */

function ReducedQ(Pts, j, EmbedL, Elog, C9, C10, Periods, expTors, initQ)
  r := #EmbedL;
  newQ := initQ;
  EmbedE := Curve(EmbedL[1]);

  // Repeat LLL reduction until no further reduction is possible
  repeat
    Q := newQ;
    S := r*(Q^2)*(expTors^2);
    T := 3*r*Q*expTors/Sqrt(2);

    // Create the basis matrix
    C := 1;
    repeat
      C := Q*Ceiling((r+2)/2);
      L := ZeroMatrix(Integers(), r+2, r+2);
      // Elliptic logarithm should have precision "suitable to" C
      // e.g. If C = 10^100, then Re(Elog[i]) should be computed
      // correctly to at least 100 decimal places
      pow10_C := Ceiling(Log(10, C));
      if pow10_C > Precision(Elog[1]) then
        vprint Intpts, 2: "Need higher precision, recompute elliptic logarithm ...";
        // Re-compute periods and elliptic logarithms
        // to the right precision
        // First, re-embed E into higher precision
        E := Curve(Pts[1]); // elliptic curve over number field
        a1, a2, a3, a4, a6 := Explode(aInvariants(E));
        a1 := Conjugate(a1, j : Precision := pow10_C+10);
        a2 := Conjugate(a2, j : Precision := pow10_C+10);
        a3 := Conjugate(a3, j : Precision := pow10_C+10);
        a4 := Conjugate(a4, j : Precision := pow10_C+10);
        a6 := Conjugate(a6, j : Precision := pow10_C+10);
        EmbedE := EllipticCurve([a1, a2, a3, a4, a6]);
        EmbedL := [];
        _, Periods := c8(EmbedE : Prec := pow10_C);
        X := [Conjugates(P[1] : Precision := pow10_C+10) : P in Pts];
        for i := 1 to #Pts do
          P := Points(EmbedE, X[i][j] ; J ; Prec := pow10_C/2) then
            Append("EmbedL, P");
          else
            Append("EmbedL, -P");
          end if;
        end for;
        Elog := [EllipticLog(EmbedE, P : Prec := pow10_C) : P in EmbedL];
        vprint Intpts, 2: "Elliptic logarithm recomputed";
      end if;
      w1, w2 := Explode(Periods);
      // Assign all non-zero entries
      for i := 1 to r do
        L[i,i] := 1;
        L[r+1, i] := Truncate(C*Re(Elog[i]));
        L[r+2, i] := Truncate(C*Im(Elog[i]));
      end for;
      L[r+1, r+1] := Truncate(C*Re(w1));
      L[r+1, r+2] := Truncate(C*Re(w2));
      L[r+2, r+1] := Truncate(C*Im(w1));
      L[r+2, r+2] := Truncate(C*Im(w2));
      L := Transpose(L); // In Magma, basis is spanned by row vector!
    end if;
  end repeat;

  // LLL reduction and constants
  L := LLL(L);
  b1 := L[1]; // 1st row of reduced basis
// Norm(b1) = square of Euclidean norm
lhs := 2^(-r-1)*Norm(b1) - S;
until (lhs ge 0) and (Sqrt(lhs) gt T);
newQ := ( Log(C*C9*expTors) - Log(Sqrt(lhs) - T) ) / C10;
newQ := Floor(Sqrt(newQ));
pow10 := Floor(Log(10, C));
vprintf Intpts, 2: "Choose C = %.4o x 10^%o. ", 1.*C/10^pow10, pow10;
vprintf Intpts, 2: "After reduction, Q <= %o", newQ;
until ((Q - newQ) le 1) or (newQ le 1);
return newQ;
end function;

Main intrinsic functions
**************************************************************************

intrinsic IntegralPoints(E::CrvEll, L::[PtEll], Q::RngIntElt : Prec := 0) ->
SeqEnum, SeqEnum
{Given an elliptic curve E over a number field, its Mordell-Weil basis L, and
a positive integer Q, return the sequence of all integral points modulo [-1]
of the form P = q_1*L[1] + ... + q_r*L[r] + T with some torsion point T and
|q_i| <= Q, followed by a sequence of tuple sequences representing the points
as a linear combination of points. An optional tolerance may be given to speed
up the computation when checking integrality of points.)

check input validity
require IsNumberField(BaseRing(E));
"Elliptic curve must be defined over a number field";
require (Prec ge 0) and (Prec in Integers());
"Precision must be a non-negative integer";
require &and[P in E : P in L]: "All points in L must be in E(K)";

// Find the generators of the torsion subgroup of E(K)
Tors, map := TorsionSubgroup(E);
expTors := Exponent(Tors); 
G := Generators(Tors);
if (#L eq 0) and (#G eq 0) then
  return [], []; // nothing to do
end if;
Tors := [map(g) : g in G]; // each generator of E(K)_tors
 OrdG := [Order(g) : g in G]; // order of each generator

// Create all possible (r+#Tors)-tuples
r := #L; // r = rank of E(K)
C := [0 : i in [1..(r + #Tors)]];
ListC := [];
for i := 0 to Q do
  C[i] := i;
  Append("ListC, C");
end for;

for i := 2 to r do
tmp := [];
for j := 1 to Q do
    for l in ListC do
        tup := l;
        tup[i] := j;
        Append(~tmp, tup);
        // Avoid having its negative in the list
        // Only use when all previous entries in tuple are zero
        for k := 1 to i-1 do
            if tup[k] ne 0 then
                tup[i] := -j;
                Append(~tmp, tup);
                break;
            end if;
        end for;
    end for;
end for;
ListC := ListC cat tmp;
end for;
// Add torsion point, if any
if #Tors ne 0 then
    for i := 1 to #Tors do
        tmp := [];
        for j := 1 to (OrdG[i]-1) do
            for l in ListC do
                tup := l;
                tup[r+i] := j;
                Append(~tmp, tup);
            end for;
        end for;
        ListC := ListC cat tmp;
    end for;
end if;
Remove(~ListC, 1); // remove point at infinity
L := L cat Tors;
vprint Intpts, 2: "Generators = ", L;
PtsList := [];
CoeffList := [];
// Skip the complex arithmetic and only perform exact arithmetic if tol = 0
if Prec eq 0 then
    vprint Intpts : "Exact arithmetic";
    for l in ListC do
        P := &+[l[i]*L[i] : i in [1..#L]];
        if IsIntegral(P[1]) and IsIntegral(P[2]) then
            vprintf Intpts: "%o ---> %o
", l, P;
            Append(~PtsList, P);
            TupList := [ <L[i], l[i]> : i in [1..#L] | l[i] ne 0 ];
            Append(~CoeffList, TupList);
        end if;
    end for;
else
    Suggested by John Cremona
    // Point search. This is done via arithmetic on complex points on each
    // embedding of E. Exact arithmetic will be carried if the resulting
    // complex points are "close" to being integral, subject to some precision
    // Embed each generator of the torsion subgroup
    basePrec := Maximum([30, Prec+10]);
    X := [Conjugates(P[1] : Precision := basePrec) : P in (L cat Tors)];
    Y := [Conjugates(P[2] : Precision := basePrec) : P in (L cat Tors)];
    // Create all embeddings of E
    K := BaseRing(E);
s1, s2 := Signature(K);
\[\begin{align*}
a_1, a_2, a_3, a_4, a_6 &:= \text{Explode}(a\text{Invariants}(E)); \\
a_1 &:= \text{Conjugates}(a_1 : \text{Precision} := \text{basePrec}); \\
a_2 &:= \text{Conjugates}(a_2 : \text{Precision} := \text{basePrec}); \\
a_3 &:= \text{Conjugates}(a_3 : \text{Precision} := \text{basePrec}); \\
a_4 &:= \text{Conjugates}(a_4 : \text{Precision} := \text{basePrec}); \\
a_6 &:= \text{Conjugates}(a_6 : \text{Precision} := \text{basePrec}); \\
\text{EmbedEList} &:= [\text{EllipticCurve}(\{a_1[j], a_2[j], a_3[j], a_4[j], a_6[j]\}) : j \in [1..(s_1+2\times s_2)]]; \\
\end{align*}\]

// Use precision to decide a possibility of "being integral".
// Note that too large precision may lead to missing some integral points,
// while too small precision may slow the computation.
vprint Intpts: "Precision = ", Prec;

// Create the matrix containing all embeddings of the integral basis of K
// as its columns
\text{IntBasis} := \text{IntegralBasis}(K);
\text{C} := \text{ComplexField}(\text{basePrec});
\text{B} := \text{Matrix}((\text{C}, \#\text{IntBasis}, \text{Degree}(K),
                \text{Conjugates}(a : \text{Precision} := \text{basePrec}) : a \in \text{IntBasis}));
// Note that B is always invertible, so we can take its inverse
B := B^{-1};

// Modified on 29-30 May 2009
// For each possible tuple representing the coefficients in a linear
// combination of points in L, compute x(P) on each embedding and store
// then. This is to:
// 1) avoid any possible risks of Magma being unable to recognise complex
//    points when calling them from a nested sequence, as happen in the
//    previous version, and
// 2) maintain the same (or even faster) computational speed.
// Technically, the drawback of this version is that it consumes much more
// memory than the previous one, although this is not really a huge problem
// in practice for most computers these days
\text{x_tuple} := [(\text{C}!0) : i \in [1..\text{Degree}(K)]];
\text{x_coord} := [\text{x_tuple} : i \in [1..\#\text{ListC}]];
for j in [(1..s_1) \cup (s_1+1..s_1+2\times s_2 by 2)] do
  // Create the embedding of each point in L (one embedding at a time)
  \text{EmbedL} := [];
  for i := 1 to (r + \#Tors) do
    \text{P} := \text{Points}(\text{EmbedEList}[j], X[i][j])[1];
    // Choose the right sign for the y-coordinate
    // if \((Y[i][j] \ne 0) \text{ and } (\text{Re}(P[2]/Y[i][j]) \lt 0)\text{ then then}
    // \text{Abs}(P[2] - Y[i][j]) \lt 10^{-(\text{basePrec}/2)}\text{ then}
    // Append("EmbedL, P);
    // else
    // Append("EmbedL, -P);
    end if;
  end for;
  for n in [1..\#\text{ListC}] do
    \text{l} := \text{ListC}[n];
    \text{P} := \&\{l[i]\} \times \text{EmbedL}[i] : i \in [1..\#L]\};
    \text{x_coord[n][j]} := P[1];
    // For each pair of complex embeddings, we only have to
    // compute P on just one embedding in the pair. Another P on
    // another embedding is simply the complex conjugate
    if j \gt s_1 then
      \text{x_coord[n][j+1]} := \text{Conjugate}(P[1]);
    end if;
  end for;
end for;

// Point search
for n in [1..\#\text{ListC}] do
  // Check if the x-coordinate of P is "close to" being integral
  // If so, compute P exactly and check if it is integral; skip P otherwise
  \text{xofP} := \text{Matrix}([\text{x_coord[n]}]);
  // Write x(P) w.r.t. the integral basis of K
  // Due to the floating arithmetic, some entries in LX may have very tiny
  // imaginary part, which can be thought as zero
LX := XofP * B;
LX := [Abs(Re(LX[1,i]) - Round(Re(LX[1,i]))) : i in [1..#IntBasis]];
LX := &and[dx lt 10^-Prec : dx in LX];
if not LX then
   // x-coordinate of P is not integral, skip P
   continue;
end if;
// Now check P by exact arithmetic
// Add P and the list of tuples representing P into the list
// if P is integral
l := ListC[n];
P := &+[l[i]*L[i] : i in [1..#L]];
if IsIntegral(P[1]) and IsIntegral(P[2]) then
   vprintf Intpts: "%o ---> %o
", l, P;
   Append(~PtsList, P);
   TupList := [ <L[i], l[i]> : i in [1..#L] | l[i] ne 0 ];
   Append(~CoeffList, TupList);
end if;
end for;
vprint Intpts: "*"^45;
return PtsList, CoeffList;
end intrinsic;

/**
 * Compute all integral points on elliptic curve over a number field.
 * This function simply computes a suitable bound Q, and return
 * IntegralPoints(E, L, Q : tol := ...).
 * Input:
 * E = elliptic curve over a number field K in standard Weierstrass form
 * L = a sequence of points in the Mordell-Weil basis for E(K)
 * Output:
 * S1 = sequence of all integral points on E(K) modulo [-1]
 * S2 = sequence of tuples representing the points as a linear combination
 * of points in L
 * Parameter:
 * Prec = precision used for checking integrality of points.
 * (Default = 0 - only exact arithmetic will be performed)
 * Require:
 * elog.m - for computing elliptic logarithms
 * nfhtbound.m - for computing Cremona-Prickett-Siksek height bounds
 */
intrinsic IntegralPoints(E::CrvEll, L::[PtEll] : Prec := 0) -> SeqEnum, SeqEnum
{Given an elliptic curve over a number field and its Mordell-Weil basis, return
the sequence of all integral points modulo [-1], followed by a sequence of tuple
sequences representing the points as a linear combination of points. An optional
tolerance may be given to speed up the computation when checking integrality of
points. (This function simply computes a suitable Q and call
IntegralPoints(E, L, Q: tol := ...))

   // Check input validity
   require IsNumberField(BaseRing(E));
   "Elliptic curve must be defined over a number field";
   require (Prec ge 0) and (Prec in Integers());
   "Precision must be a non-negative integer";
   require &and[P in E : P in L]: "All points in L must be in E(K)";
   if #L eq 0 then
      return IntegralPoints(E, [], 0 : Prec := Prec);
   end if;
   K := BaseRing(E);
   s1, s2 := Signature(K);
a1, a2, a3, a4, a6 := Explode(aInvariants(E));
   // Set initial precision for computing embeddings, periods, elliptic logs
   // N.B. We only require high-precision elliptic logs during LLL process,
   // so if our initial precision is already high enough, then we do not need
   // to re-compute them again.
   initPrec := 300; // this can be changed arbitrarily
// Embed $E$ into various (real/complex) embeddings.
a1 := Conjugates(a1 : Precision := initPrec);
a2 := Conjugates(a2 : Precision := initPrec);
a3 := Conjugates(a3 : Precision := initPrec);
a4 := Conjugates(a4 : Precision := initPrec);
a6 := Conjugates(a6 : Precision := initPrec);
b2 := Conjugates(bInvariants(E)[1] : Precision := initPrec);
p1 := P1(RealField(initPrec));

// Embed generators in the Mordell-Weil basis
X := [Conjugates(P[1] : Precision := initPrec) : P in L];
Y := [Conjugates(P[2] : Precision := initPrec) : P in L];

// Find the generators of the torsion subgroup of $E(K)$
Tors, map := TorsionSubgroup(E);
expTors := Exponent(Tors);
G := Generators(Tors);
Tors := [map(g) : g in G]; // generators of torsion subgroup
OrdG := [Order(g) : g in G]; // their orders

// Global constants (i.e. do not depend on any embedding of $E$)
C2 := -CPSLowerHeightBound(E);
C3 := c3(L);
h := h_E(E);
vprintf Intpts, 2: "Global constants";
vprintf Intpts, 2: "c2 = %.4o
", C2;
vprintf Intpts, 2: "c3 = %.4o
", C3;
vprintf Intpts, 2: "h_E = %.4o
", h;

Q := [];
// Find the most reduced bound on each embedding of $E$
// But first let's adjust the index
for i := 1 to (s1+s2) do
  if i le s1 then
    j := i;
    nv := 1;
  else
    j := s1 + (2*(i-s1)-1);
    nv := 2;
  end if;
  // Create complex embedding of $E$
ee := EllipticCurve([a1[j], a2[j], a3[j], a4[j], a6[j]]);
  // Local constants (depending on embedding)
  // C9, C10 are used for the upper bound on the linear form in logarithm
  C4 := C3 * Degree(K) / (nv*(s1+s2));
  vprintf Intpts, 2: "c4 = %.4o
", C4;
  C5 := C2 * Degree(K) / (nv*(s1+s2));
  vprintf Intpts, 2: "c5 = %.4o
", C5;
  C6 := c6(ee);
  vprintf Intpts, 2: "c6 = %.4o
", C6;
  delta := 1 + (nv-1)*pi;
  vprintf Intpts, 2: "c8 = %.4o
", C8;
  vprintf Intpts, 2: "c9 = %.4o
", C9;
  vprintf Intpts, 2: "c10 = %.4o
", C10;
  vprintf Intpts, 2: "Q = %.4o
", Q;
  vprintf Intpts, 2: "Periods = %o
", Periods;
  vprintf Intpts, 2: "c10 = %.4o
", C10;
  vprintf Intpts, 2: "Q0 = %.4o
", Q0;
  // Constants for David's lower bound on the linear form in logarithm
  vprintf Intpts, 2: "v1, v2 := Explode(Periods);
  Embedl. := [];

// Embed $E$ into various (real/complex) embeddings.
// Find images of all points in a Mordell–Weil basis for E(K) on
// the embedding ee, and then compute complex elliptic logarithms.
for k := 1 to #L do
  P := Points(ee, X[k][j][1]);
  if Abs(P[2] - Y[k][j]) lt 10^(initPrec/2) then
    Append('EmbedL, P);
  else
    Append('EmbedL, -P);
  end if;
end for;
//EmbedL := [[X[i][j], Y[i][j]] : i in [1..#L]];
vprintf Intpts, 2: "Computing elliptic logarithms...";
Elog := [EllipticLog(ee, P: Prec := initPrec-10) : P in EmbedL];
vprintf Intpts, 2: " : Done";
D7 := 3*pi / ((Abs(w2)^2) * Im(w2/w1));
vprintf Intpts, 2: "d7 = %.4o
", D7;
D8 := d8(E, L, Elog, Periods, D7);
vprintf Intpts, 2: "d8 = %.4o
", D8;
D9 := d9(E, L, Elog, Periods, D7);
vprintf Intpts, 2: "d9 = %.4o
", D9;
D10 := d10(E, L, Elog, Periods, D7);
vprintf Intpts, 2: "d10 = %.4o
", D10;
// Find the reduced bound for the coefficients in the linear
// logarithmic form
initQ := InitialQ(Degree(K), #L, Q0, C9, C10, D8, D9, D10, h, expTors);
  vprintf Intpts, 2: "Initial Q <= 10^%o
", initQ;
new_Q := ReducedQ(L, j, EmbedL, Elog, C9, C10, Periods, expTors,
  initQ);
  Append('Q, new_Q);
vprintf Intpts, 2: "="^45;
end for;
Q := Maximum(Q);
return IntegralPoints(E, L, Q : Prec := Prec);
end intrinsic;

A.3 Height Bound I: Main Functions

For the rest of this appendix, we will give our own implementation of an algorithm
for determining a positive lower bound for the canonical height on elliptic curves
over number fields, based on Theorem 3.4.1. Although all of the following files
are required for this algorithm, the only function that should be called by users is
IsLowerBound(); see the file below for more details.

*******************************************************************************
* heightbound.m
* Computing a positive lower bound for the canonical height on elliptic
* curves over number fields
* (Based on Thongjunthug's paper Math. Comp. 79 (2010), pp. 2431-2449)
* By Thotsaphon Thongjunthug
* Last updated: 07 December 2010
* Any errors should be reported to <nookaussie@yahoo.com>
* Required packages:
* 1) elog.m - for computing periods and elliptic logarithms
A.3. Height Bound I: Main Functions

2) alphas.m - for computing alpha_v for all archimedean place v

declare verbose Bound, 1;

function LCMTamagawa(E)
I := LocalInformation(E);
S := \{t[4] : t in I\}; // Tamagawa index at each bad place
n, _ := Signature(BaseRing(E)); // no. of real embedding
A := aInvariants(E);
A := \{Conjugates(a) : a in A\};
for i := 1 to n do
<x> := PolynomialRing(Parent(A[1][i])); // parent = some real field
// Set y = 0 in y^2 + a1*x*y + a3*y ==> 0 = x^3 + a2*x^2 + a4*x + a6
// and check the number of real roots.
f := x^3 + A[2][i]*x^2 + A[4][i]*x + A[5][i];
if #(Roots(f)) eq 1 then
  Append(~S, 1);
else
  Append(~S, 2);
end if;
end for;
print Bound: "Tamagawa indices = ", S;
return LeastCommonMultiple(S);
end function;

function ExtraConstant(E, P, L)
discE := Discriminant(E);
n := #P; // should be equal to #L
extraConst := 1;
for i := 1 to n do
discEp := Discriminant(L[i]);
extraConst *:= Norm(P[i])^Valuation(discE/discEp, P[i]);
end for;
extraConst := Log(extraConst)/6;
return extraConst;
end function;

function LCMTamagawa(E)
I := LocalInformation(E);
S := \{t[4] : t in I\}; // Tamagawa index at each bad place
n, _ := Signature(BaseRing(E)); // no. of real embedding
A := aInvariants(E);
A := \{Conjugates(a) : a in A\};
for i := 1 to n do
<x> := PolynomialRing(Parent(A[1][i])); // parent = some real field
// Set y = 0 in y^2 + a1*x*y + a3*y ==> 0 = x^3 + a2*x^2 + a4*x + a6
// and check the number of real roots.
f := x^3 + A[2][i]*x^2 + A[4][i]*x + A[5][i];
if #(Roots(f)) eq 1 then
  Append(~S, 1);
else
  Append(~S, 2);
end if;
end for;
print Bound: "Tamagawa indices = ", S;
return LeastCommonMultiple(S);
end function;

function ExtraConstant(E, P, L)
discE := Discriminant(E);
n := #P; // should be equal to #L
extraConst := 1;
for i := 1 to n do
discEp := Discriminant(L[i]);
extraConst *:= Norm(P[i])^Valuation(discE/discEp, P[i]);
end for;
extraConst := Log(extraConst)/6;
return extraConst;
end function;
function AllPrimeIdeals(R, n)
I := []; S := [t : t in [1 .. n] | IsPrime(t)];
for x in S do
tmp := Decomposition(R, x);
// Decomposition always return a list (seq) of tuples
// The first element in each tuple is the prime ideal
tmp := [ t[1] : t in tmp ];
// note that there is always at least 1 elt in tmp
// (really, up to Degree(K) elements)
// check norm of, say, first, prime ideal in tmp
if Norm(tmp[1]) le n then
I := I cat tmp;
end if;
end for;
return I;
end function;

function LocalMinimalModel(E)
L := []; P := [];
// Decompose the discriminant of E
O := RingOfIntegers(BaseRing(E));
disc := O ! Discriminant(E);
c4, c6 := Explode(cInvariants(E));
c4 := O!c4; c6 := O!c6;
tmp := Decomposition(disc);
// check minimality of each bad place
for D in tmp do
// E is minimal at p if ord_p(disc) < 12
// otherwise, minimal at p if ord_p(c4) < 4 or ord_p(c6) < 6
if (D[2] lt 12) or (Valuation(c4, D[1]) lt 4) or (Valuation(c6, D[1]) lt 6) then
Append(~L, E); // E is already minimal at p
else
// otherwise, find an integral minimal model at p
minModelp := MinimalModel(E, D[1]);
Append(~L, minModelp);
end if;
end for;
return P, L;
end function;

function FindEp(E, Plcs, L, p)
norm_p := Norm(p);
F, phi := ResidueClassField(p);
// make sure we're working on a model that is minimal at p
if p in Plcs then
    // p is a bad place (i.e. p | disc(E))
    // Get the corresponding minimal model at p
    E := L[Index(Plcs, p)];
end if;

pl := Place(p);

// Case 1: Good reduction - E_ns is ell. curve over finite field
if (Valuation(Discriminant(E), pl) eq 0) then
    A := phi(Coefficients(E));
    E := EllipticCurve(A);
    // Need group structures = prod. of (up to) 2 cyclic group
    // of finite order
    G := Generators(E);
    G := [Order(g) : g in G];
    // N.B: e_p = exponent = max(d1,d2)
    G, := Max(G);
    return G;
end if;

// Otherwise, E has bad reduction at p
// calculate c4, c6
b2, b4, b6, := Explode(bInvariants(E));
c4, c6 := Explode(cInvariants(E));

// Case 2.1: Additive reduction
if Valuation(c4, pl) gt 0 then
    // E_ns = k_p+
    // if |k_p| is prime, E_ns is cyclic and so e_p = |E_ns| = N(p)
    // otherwise, e_p = char(k_p)
    if IsPrime(#F) then
        return norm_p;
    else
        return Characteristic(F);
    end if;
end if;

// case 1: char(k_p) = 2
if IsZero(norm_p mod 2) then
    a1, a2, a3, _ := Explode(Coefficients(E));
a1 := phi(a1);
a2 := phi(a2);
a3 := phi(a3);
    f := func<x | x^2 + a1*x + (a3/a1 + a2)>;
    for r in F do
        if f(r) eq 0 then
            // a root exist => split mult.
            return (norm_p - 1);
        end if;
    end for;
    // otherwise, non-split
    return (norm_p + 1);
end if;

// case 2: char(k_p) = 3
if IsZero(norm_p mod 3) then
    //print "b2 = ", b2;
b2 := phi(b2);
    if IsSquare(b2) then
        return (norm_p - 1); // split
    else
        return (norm_p + 1); // non-split
    end if;
end if;

// case 3: char(k_p) != 2,3
//c6 := -b2^3 + 36*b2*b4 - 216*b6;
c4 := phi(c4);
c6 := phi(c6);
if IsSquare(c4 * c6) then
    return (norm_p - 1); // split
else
    return (norm_p + 1); // non-split
end if;
end function;

/**
 * Calculate D_E(n)
 * Fixed on 8 Jan 2008
 * Input:
 * E = elliptic curve over number field in standard Weierstrass form
 * n = a positive integer
 **/
function D_E(E, n)
    K := BaseRing(E);
    O := RingOfIntegers(K);
    r, _ := Max([2, Degree(K)]);
    P := AllPrimeIdeals(O, (n+1)^r);
    S := 0;
    Plcs, L := LocalMinimalModel(E);
    // Choose p such that e_p divides n
    for p in P do
        e_p := FindEp(E, Plcs, L, p);
        if ((n mod e_p) ne 0) then
            continue;
        end if;
        cp, _ := ResidueClassField(p);
        cp := Characteristic(cp); // this must be prime number
        S +:= 2 * (1 + Valuation(n/e_p, cp)) * Log(Norm(p));
    end for;
    return S;
end function;

//load "intersect_complex.m";

*****************************************************************************
* Main intrinsic functions
*****************************************************************************

/**
 * Decide if a given number \lambda is a positive lower bound for the canonical
 * height on an elliptic curve E defined over a number field K.
 * Input:
 * E = elliptic curve over K
 * lambda = initial guess for a lower bound on E(K)
 * Output:
 * Return true if \lambda is a lower bound.
 * If the algorithm FAILS to show that \lambda is a lower bound, return false
 * Parameter:
 * n_max = maximum number for computing B_n(\lambda) (i.e. for 1 <= n <= n_max)
 * initRes = initial resolution for region intersection (required only when E
 * has complex embeddings.
 * Note that if initRes = n, then the grid has 2^n-by-2^n dimension
 **/
intrinsic IsLowerBound(E::CrvEll, lambda::FldReElt :
    n_max := 5, initRes := 4) -> BoolElt
(Check if \lambda > 0 is a lower bound for the canonical height on an elliptic
curve E defined over number field. If so, return true. If the algorithm fails
to confirm this, false is returned.)

vprintf Bound: "Computing alphas...";
prodAlphas := &*Alphas(E);
\n\n\nvprintf Bound: " : Done"
\nP, L := LocalMinimalModel(E);
extraConst := ExtraConstant(E, P, L);
\n// Now we work on E_gr(K)
c := LCMTamagawa(E); mu := lambda*c^-2;
\nvprintf Bound: "Check if mu = %o is a lower bound on E_gr(K)\n", mu;
// Step 1: Compute all B_n(mu) for 1 <= n <= n_max
// If some of them is less than 1, then mu is a lower bound on E_gr,
// and thus we return true.
Bns := [];
for n := 1 to n_max do
    Bn := Exp(Degree(K)*(n^2)*mu - D_E(E, n) + extraConst) * prodAlphas;
    \nvprintf Bound: "B_%o(mu) = %o\n", n, Bn;
    if Bn lt 1 then
        \nvprintf Bound: "\n*** B_%o(mu) < 1, we have a lower bound! \n***\n", n;
        return true;
    end if;
    Append(~Bns, Bn);
end for;
\n// Create s real embeddings and t complex embeddings of E
Es := []; Et := [];
s, t := Signature(K);
A := aInvariants(E);
A := [Conjugates(a) : a in A];
for i := 1 to s do
    RR := RealField(Precision(A[1][i]));
    to make sure that curve will be defined over R rather than C
    Append(~Es, EllipticCurve([RR| a[i] : a in A ]));
end for;
for i := 1 to t do
    Append(~Et, EllipticCurve([ a[s+(2*i-1)] : a in A ]));
end for;
\n// Step 2: Real Embeddings
// Find intersection of subintervals of [0, 1]
\nj := 1;
for E in Es do
    vprintf Bound: "Real embedding #\n";
    D := FindSn(E, -Bns[1], Bns[1], 1);
    vprintf Bound: "S_1 is \n";
    if \nD eq 0 then
        vprintf Bound: "\n*** Empty intersection of intervals ***\n";
        return true;
    end if;
end for;
\n// Step 3: Complex Embeddings
// Find intersection of regions on each fundamental paralellogram
\nj := 1;
flag := GetVerbose("Bound");
flag := flag eq 1; // convert to true/false
for E in Et do
vprintf Bound: "Complex embedding \#%o\n", j;
D := ZRegion(E, Sqrt(Bns[1]), initRes : ShowPlot := flag);
oldLevel := 1;
for n := 2 to n_max do
    vprintf Bound: "n = ", n;
    tmp := ZRegion(E, Sqrt(Bns[n]), initRes : ShowPlot := flag);
    tmp := GridEntryTransform(tmp);
    // If the region is the whole lowe-half fundamental parallelogram,
    // then we don't have to do intersection and manification
    if not (and(tmp)) then
        tmp := DivByN(tmp, n); // find T^{(v)}_n(\sqrt{B_n(\mu)})
        vprintf Bound: "After division by \#%o, the size is \#%o\n",
            n, #tmp;
        // Apply region's scaling if necessary before checking
        // the intersection.
        refineLevel := LCM(n, oldLevel);
        D := Magnify(D, Integers() ! (refineLevel/oldLevel));
        tmp := Magnify(tmp, Integers() ! (refineLevel/n));
        D := IntersectTwoGrids(D, tmp);
        oldLevel := refineLevel;
    end if;
    // If the intersection is empty, we have a lower bound
    if not(or(D)) then
        vprintf Bound: "*** Empty intersection of regions ***";
        return true;
    end if;
end for;
//Otherwise, fail to show that \mu is a lower bound on E_gr(K)
return false;
end intrinsic;

/**
* Compute an upper bound for the index n = [E(K)/E_tors(K) : <Pts>], where
* Pts is the set of generators in a Mordell-Weil basis of E(K), using the
* geometry of numbers
* (see Siksek's "Infinite descent on elliptic curves", Theorem 3.1)
* Input:
* Pts = sequence of points in a Mordell-Weil basis
* lambda = a positive lower bound for the canonical height on E(K)
**/
intrinsic UpperBound4Index(Pts::SeqEnum, lambda::FldReElt) -> FldReElt
(Compute an upper bound for the index [E(K)/E_tors(K) : <P_1,...,P_r>] using
the geometry of numbers.)
E := Curve(Pts[1]);
require &and[P in E : P in Pts]: "All points must be on the same curve"
lambda := DetB := Determinant(HeightPairingMatrix(Pts));
r := #Pts;
// Here, gamma := [gamma_r]^r
if r lt 9 then
    gamma := [1, 4/3, 2, 4, 8, 64/3, 64, 2^8];
else
    gamma := ((Gamma(1 + r/2))^2) * (4 * Pi(RealField()))^r;
end if;
// calculate the upper bound of n
n := Sqrt(detR * gamma / (lambda^r));
return n;
end intrinsic;
A.4 Height Bound II: Computing $\alpha_v$

This file computes the quantity $\alpha_v$ (see Section 2.2.2 for its definition) for every archimedean place $v$, using the method mentioned in [CPS06, Section 7 and 9].

```plaintext
 forward RefineAlphaBound; // require for this recursive function

function MaxAbsPQ(P, Q, z)
    tmp := [Abs(Evaluate(P, z))];
    Append(~tmp, Abs(Evaluate(Q, z)));
    tmp, _ := Max(tmp);
    return tmp;
end function;

function JustRoots(f, a, b)
    if a gt b then
        error "a must be < or = b";
    end if;
    R := Roots(f);
    R := [r[1] : r in R];
    R := [r : r in R | (r ge a) and (r le b)];
    Sort(~R);
    return R;
end function;

function IsInInterval(I, t)
    a := I[1]; b := I[2];
```
return ((t ge a) and (t le b));
end function;

/**
 * Find d = inf{|f(x)|, |g(x)|} where x is real and satisfies
 * (this is called e in CPS' paper!)
 * |x| <= 1 and f(x) >= 0.
 * Input:
 * E = elliptic curve over reals
 * f, g = real polynomials in x
 * Output:
 * A value of d > 0. If such d does not exist (due to the region is empty),
 * then -1 is returned.
 **/ function Find_d(E, f, g)
 local leftPt, rightPt;
 // Find the starting point of x in E_0(R)
 // i.e. beta = max real root of RHS
 _, a2, _, a4, a6 := Explode(aInvariants(E));
_<x> := PolynomialRing(Parent(a2));
rhs := x^3 + a2*x^2 + a4*x + a6;
tmp := Roots(rhs); // real roots
beta, _ := Max(tmp);
// Define the domain D = {x:|x|<=1 and f(x)>=0}
if beta gt 1 then
   return -1; // D empty
end if;
a, _ := Max([-1., beta]);
b := 1.;
tmp := JustRoots(f, a, b);
R := {a@};
for i in tmp do
   R := R join {i@};
end for;
R := R join {b@};
r := #R - 1;
D := [ ];
includeStatus := false;
for i := 1 to r do
   midX := (R[i] + R[i+1])/2;
   if Evaluate(f, midX) gt 0 then
      // in case the root has multiplicity 2
      if (#D eq 0) then
         Append(~D, [ R[i], R[i+i] ]); //
      elseif (rightPt eq R[i]) then
         Prune(~D);
         Append(~D, [leftPt, R[i+1]]); //
      else
         Append(~D, [ R[i], R[i+i] ]); //
         leftPt := R[i];
      end if;
      rightPt := R[i+1];
      includeStatus := true; // next pivot point is included
      continue;
   end if;
   if (Evaluate(f, R[i]) eq 0) and not includeStatus then
      Append(~D, [ R[i], R[i] ]); //
      leftPt := R[i];
      end if;
      rightPt := R[i+1];
      includeStatus := false; //
   end for;
if (Evaluate(f, R[r+1]) eq 0) and not includeStatus then
   Append(~D, [ R[r+1], R[r+1] ]); //
end if;
return D;
end function;
A.4. Height Bound II: Computing $\alpha_v$

end if;

if (#D eq 0) then
  //print "FindD: Valid region of x is empty";
  return -1;
end if;

// list all roots of f+g, f-g, f', g'
R := JustRoots(f + g, a, b);
R := R cat JustRoots(f - g, a, b);
R := R cat JustRoots(Derivative(f), a, b);
R := R cat JustRoots(Derivative(g), a, b);
//R := \[r : r in R | (r ge -1) and (r le 1)\];
//Sort(R);

// here, i is the interval
Vals := [];
for i in D do
  Vals := Vals cat [ MaxAbsPQ(f,g,i[1]), MaxAbsPQ(f,g,i[2]) ];
  //print "Vals init = ", Vals;
  for j in R do
    if IsInInterval(i,j) then
      Append(~Vals, MaxAbsPQ(f,g,j));
      //Exclude(~R,j);
    end if;
  end for;
  //print Vals;
end for;

//if IsZero(#Vals) then
//  print "ERROR: Vals is empty";
//end if;

d, _ := Min(Vals);
//print "The infimum is ", d;
return d;
end function;

/**
* Find d' = inf{|F(x)|, |G(x)|} where x is real and satisfies
* |x|<=1 and F(x)>=0 (this is called e' in CPS' paper!)
* Input:
* E = elliptic curve over reals
* F, G = real polynomials in x
* Output:
* a value of d'. If such d does not exist (due to the region
* is empty), then -1 is returned.
**/
function Find_ddash(E, F, G)
  local leftPt, rightPt;
  // Find the starting point of x in E_0(R)
  // i.e. beta = max real root of RHS
  _, a2, a4, a6 := Explode(Coefficients(E));
P<x> := PolynomialRing(RealField());
a2 := P ! a2;
a4 := P ! a4;
a6 := P ! a6;
rhs := x^3 + a2*x^2 + a4*x + a6;
tmp := Roots(rhs);
tmp := \[t[i] : t in tmp\];
beta, _ := Max(tmp);

  // init the domain D'
  if beta le -1 then
    DDashInit := [ [-1., 1./beta], [0., 1.] ];
  elif beta le 1 then
    DDashInit := [ [0., 1.] ];
  else
    DDashInit := [ [0, 1./beta] ];
  end if;
DDash := [ ];
for I in DDashInit do
  a := I[1];
  b := I[2];
  tmp := JustRoots(F, a, b);
  R := {a@};
  for i in tmp do
    R := R join {i@};
  end for;
  R := R join {b@};
  r := #R - 1;
  includeStatus := false;
  for i := 1 to r do
      midX := (R[i] + R[i+1])/2;
      if Evaluate(F, midX) gt 0 then
        // In case the root has multiplicity 2
        if (#DDash eq 0) then
          Append(~DDash, [ R[i], R[i+1] ]); 
          leftPt := R[i];
        else
          Append(~DDash, [ R[i], R[i+1] ]); 
      end if;
      rightPt := R[i+1];
      includeStatus := true; // next pivot point is included
      continue;
      end if;
      if (Evaluate(F, R[i]) eq 0) and not includeStatus then
        Append(~DDash, [ R[i], R[i] ]); 
        leftPt := R[i];
      end if;
      rightPt := R[i+1];
      includeStatus := false;
    end for;
    if (Evaluate(F, R[r+1]) eq 0) and not includeStatus then
      Append(~DDash, [ R[r+1], R[r+1] ]); 
    end if;
  end for;
  if (#DDash eq 0) then
    //print "FindDDash: Valid region of x is empty";
    return -1;
  end if;
  // list all roots of F+G, F-G, F', G'
  R := JustRoots(F + G, -1, 1);
  R := R cat JustRoots(F - G, -1, 1);
  R := R cat JustRoots(Derivative(F), -1, 1);
  R := R cat JustRoots(Derivative(G), -1, 1);
  //R := [ r : r in R | (r ge -1) and (r le 1) ];
  //Sort(~R);
  // here, I is the interval
  Vals := [ ];
  for I in DDash do
    Vals := Vals cat [ MaxAbsPQ(F,G,I[1]), MaxAbsPQ(F,G,I[2]) ]; // end points
    //print "Vals init = ", Vals;
    for j in R do
      if IsInInterval(I,j) then
        Append(~Vals, MaxAbsPQ(F,G,j)); 
        Exclude(~R,j);
      end if;
    end for;
    //print Vals;
  end for;
A.4. Height Bound II: Computing $\alpha_v$

//if IsZero(#Vals) then
// print "ERROR: Vals is empty";
//end if;
d, _ := Min(Vals);
//print "The infimum is ", d;
return d;
end function;

/**
 * Calculate the value of alpha of a given elliptic curve over real numbers
 * Input:
 * E = elliptic curve over reals
 **/
function AlphaReal(E)
b2, b4, b6, b8 := Explode(bInvariants(E));
_<x> := PolynomialRing(Parent(b2)); // assume all b have the same precision
f := 4*x^3 + b2*x^2 + 2*b4*x + b6;
g := x^4 - b4*x^2 - 2*b6*x - b8;
F := 4*x + b2*x^2 + 2*b4*x^3 + b6*x^4; // = f(x)/(x^4) and let x := 1/x
G := 1 - b4*x^2 - 2*b6*x^3 - b8*x^4; // similarly
d := Find_d(E, f, g);
d_dash := Find_ddash(E, F, G);
// take care of cases when region of valid x is empty
if (d lt 0) then
    if (d_dash ge 0) then
        return d_dash^(-1/3);
    else
        error "Both regions of X are empty (should not happen), please report!";
    end if;
elif (d_dash ge 0) then
    alpha, _ := Min([d, d_dash]);
    return alpha^(-1/3);
else
    return d^(-1/3);
end if;
end function;

/******************************************************************************
 * Auxiliary local functions II:
 * Calculate alpha_v for complex archimedean places v in K
 ******************************************************************************/
/**
 * Calculate min H = min{h((m+ni)/10: m^2 + n^2 <= 100}, m, n integer
 * where h = max{|P(z)|, |Q(z)|}.
 * Input:
 * P, Q = polynomials defined over C
 **/
function AlphaInitialGuess(P, Q)
i := BaseRing(P)!Sqrt(-1);
initMin, _ := Max(Abs(Evaluate(P, -1)), Abs(Evaluate(Q, -1)));
for m := -10 to 10 do
    boundN := Floor(Sqrt(100 - m^2));
    for n := -boundN to boundN do
        h := MaxAbsPQ(P, Q, (m + n*i)/10);
        if h lt initMin then
            initMin := h;
            end if;
    end for;
end for;
return initMin;
end function;

/**
 * Compute BigEpsilon(z, eta)
function BigEpsilon(P, Q, z, eta)
    d1 := Degree(P);
    d2 := Degree(Q);
    sum1 := 0;
    sum2 := 0;
    for n := 1 to d1 do
        P := Derivative(P); // P^(n)
        sum1 +:= (eta^n) * Abs(Evaluate(P, z)) / Factorial(n);
    end for;
    for n := 1 to d2 do
        Q := Derivative(Q); // Q^(n)
        sum2 +:= (eta^n) * Abs(Evaluate(Q, z)) / Factorial(n);
    end for;
    eps, _ := Max([sum1, sum2]);
    return eps;
end function;

function RefineAlphaBound(P, Q, mu, S, alpha, level)
    // Step 1: check if the square S intersects unit circle
    // if not, return alpha
    a, b := Explode(S[1]);
    r := S[2];
    // Modified: 25 Mar 2010 (bug found by Robert Bradshaw)
    // If [a, b] = [-1, -1], the below trick won't work.
    // But of course the intersection won't be empty.
    if level eq 0 then
        level := 1;
        S1 := <[a, b], r/2>;
        S2 := <[a, b+r/2], r/2>;
        S3 := <[a+r/2, b], r/2>;
        S4 := <[a+r/2, b+r/2], r/2>;
        alpha := RefineAlphaBound(P, Q, mu, S1, alpha, level);
        alpha := RefineAlphaBound(P, Q, mu, S2, alpha, level);
        alpha := RefineAlphaBound(P, Q, mu, S3, alpha, level);
        alpha := RefineAlphaBound(P, Q, mu, S4, alpha, level);
    end if;
    C := [ S[1], [a, b+r], [a+r, b], [a+r, b+r] ];
    C := [ c[1]^2 + c[2]^2 : c in C ]; // square of modulus of each corner
    C := [ (c le 1) : c in C ]; // check if each corner is in the circle
    if not (true in C) then
        return alpha;
    end if;
    // Step 2:
    if (a+r/2)^2 + (b+r/2)^2 le 1 then
        u := [a+r/2, b+r/2]; // u = mid-point
        eta := r/Sqrt(2);
    else
        // u = any corner that in unit circle
        ind := Index(C, true); // position of the corner in D
        if ind eq 1 then
            u := [a, b];
        elsif ind eq 2 then
            u := [a, b+r];
        end if;
elif ind eq 3 then
    u := \([a+r, b]\);
else
    u := \([a+r, b+r]\);
end if;
eta := r * Sqrt(2);
end if;

// Step 3 and 4: check condition for minimality
i := BaseRing(P)!Sqrt(-1);
u := u\[1\] + u\[2\]*i;
h := MaxAbsPQ(P, Q, u);
epsilon := BigEpsilon(P, Q, u, eta);
if (h - epsilon) gt (alpha * Exp(-mu)) then
    return alpha;
else
    alpha, _ := Min(\([alpha, h]\));
    end if;

// Step 5 and so on: split S into 4 quadrants and recursively apply
// this function to each S_i
level := level + 1;
S1 := \([a, b], r/2]\);
S2 := \([a, b+r/2], r/2]\);
S3 := \([a+r/2, b], r/2]\);
S4 := \([a+r/2, b+r/2], r/2]\);
alpha := RefineAlphaBound(P, Q, mu, S1, alpha, level);
alpha := RefineAlphaBound(P, Q, mu, S2, alpha, level);
alpha := RefineAlphaBound(P, Q, mu, S3, alpha, level);
alpha := RefineAlphaBound(P, Q, mu, S4, alpha, level);
return alpha;
end function;

/**
* Find alphas for a given elliptic curve over complex number
* Use repeated quadrisection method from Section 9 of CPS' paper.
* Input:
* E = elliptic curve over C
* mu = accuracy level (need Exp(-mu) close to 1)
**/ function AlphaComplex(E : mu := 0.005)
b2, b4, b6, b8 := Explode(bInvariants(E));
_<x> := PolynomialRing(Parent(b2));
f := 4*x^3 + b2*x^2 + 2*b4*x + b6;
g := x^4 - b4*x^2 - 2*b6*x - b8;
F := 4*x + b2*x^2 + 2*b4*x^3 + b6*x^4; // = f(x)/(x^4) and let x := 1/x
G := 1 - b4*x^2 - 2*b6*x^3 - b8*x^4; // similarly
S := \([-1,1] \times [-1,1]\), represents as \([a,b], h, v]\) where
// [a, b] = lower left corner, r = length (same for all sides)
// S := \([-1, -1/2], 2/3]\;
// Step 1: find inf max(|f(z)|, |g(z)|), where z on closed unit disc
// Initial guess: alpha_fg = \(min H = min(h((m+ni)/10: m^2 + n^2 <= 100)\)
// where h = max(|f(z)|, |g(z)|)
alpha_fg := AlphaInitialGuess(f, g);
alpha_fg := RefineAlphaBound(f, g, mu, S, alpha_fg, 0);
// Step 2: find inf max(|F(z)|, |G(z)|), where z on closed unit disc
// Initial guess: alpha_FG = \(min H = min(h((m+ni)/10: m^2 + n^2 <= 100)\)
// where h = max(|F(z)|, |G(z)|)
alpha_FG := AlphaInitialGuess(F, G);
alpha_FG := RefineAlphaBound(F, G, mu, S, alpha_FG, 0);
// Then alpha^(-3) = Min(alpha_fg, alpha_FG)
// Note that we need alpha
alpha := (alpha*Exp(-mu))^{-1/3};
return alpha;
end function;

/*****************************************************************************/
/* Main intrinsic function */
/*****************************************************************************/

/**
 * Return the list of alpha_v for all archimedean place v.
 * For those alphas from complex embeddings, the returned values are squared
 * Input:
 * E = elliptic curve over number field K
 * Output:
 * A sequence of all values of alpha_v
 **/
intrinsic Alphas(E::CrvEll) -> SeqEnum
(Given an elliptic curve E over a number field K, compute all alpha_v associated to
E for all archimedean places v of K. If v is a complex place, then the returned
value will be alpha_v^2.)
A:= mInvariants(E);
tmp := [Conjugates(a) : a in A];
s, t := Signature(BaseRing(E));
alphas := [];
// Real embeddings
for j := 1 to s do
    Erj := EllipticCurve([RealField() ! a[j] : a in tmp]);
    Append(~alphas, AlphaReal(Erj));
end for;
// Complex embeddings
// N.B.: we pick only one conjugate from its conjugacy pair
for j := 1 to t do
    Ecj := EllipticCurve([b[s + 2*j - 1] : b in tmp]);
    Append(~alphas, (AlphaComplex(Ecj))^2);
end for;
return alphas;
end intrinsic;

A.5 Height Bound III: Intersection of Intervals

This file consists of all necessary functions for compute $S_n^{(v)}(\xi_1, \xi_2)$ (see Section 2.5
for its definition) for each real archimedean place v. Note that $S_n^{(v)}$ is a disjoint
union of subintervals of $[0,1]$. As we have seen in Section 2.5, an algorithm for
computing periods and elliptic logarithms of real points is also required for this
computation; see Appendix A.1 for its implementation.

/*****************************************************************************/
/* intersect_real.m */
/* Computing $S'(v)_n(\xi_1, \xi_2)$ for all real archimedean place v */
/* */
/* By Thotsaphon Thongjunthug */
/* Last updated: 08 December 2010 */
/* Any errors should be reported to <nookaussie@yahoo.com> */
/* */
/* Required package: */
/* elog.m - for computing normalised elliptic logarithms of real points */
/*****************************************************************************/
A.5. Height Bound III: Intersection of Intervals

/**************************************************************************
 * Auxiliary local functions
  **************************************************************************/

/**
 * Return the list of real roots of \(f(x) = 4x^3 + b_2x^2 + 2b_4x + b_6\)
 * in decreasing order
 * Input:
 * E = elliptic curve defined over real numbers
 * Output:
 * a sequence of real roots
 **/
function RootsF(E)
  b2, b4, b6, b8 := Explode(bInvariants(E));
  \(<t> := PolynomialRing(Parent(b2));
  p := 4*t^3 + b2*t^2 + 2*b4*t + b6;
  R := Roots(p); // real roots only
  R := \[e[1] : e in R\]; // get rid of multiplicity
  Sort(~R);
  Reverse(~R); // decreasing order
  return R;
end function;

/**
 * Find the intersection of two real intervals \(I_1, I_2\)
 * Input:
 * \(I_1 = [a, b], I_2 = [c, d] : \) real intervals
 **/
function IntervalsMeet(I1,I2)
  if (#I1 eq 0) or (#I2 eq 0) then
    // One interval is empty, nothing to do
    return \[
  end if;
  a := I1[1]; b := I1[2];
  c := I2[1]; d := I2[2];
  if (b lt c) or (d lt a) then
    // Both intervals are disjoint
    return \[
  elif (a le c) and (d le b) then
    return I2; // subset of I1
  elif (c le a) and (b le d) then
    return I1; // subset of I2
  elif (a lt c) and (b lt d) then
    return [c,b];
  elif (c lt a) and (d lt b) then
    return [a,d];
  else
    // Should not happen, just for checking
    error "You miss some case";
  end if;
end function;

/**
 * Compute a normalised elliptic logarithm of the "higher" point of the two
 * having the same x-coordinate.
 * Input:
 * E = elliptic curve over reals in standard Weierstrass form
 * x = a real number for the x-coordinate
 **/
function NormalisedElog(E, x)
  // First, find a positive period of E
  r := Precision(BaseRing(E));
  // Warning: PeriodLattice() needs curves over C
  prec := Precision(BaseRing(E));
  C := ComplexField(prec);
  EC := ChangeRing(E, C);
"w1, w2, _ := Explode(PeriodLattice(EC : Prec := r-10));
if Im(w2/w1) gt 0 then
    L := [w1, w2];
else
    L := [w2, w1];
end if;
L, _ := TransformLattice(L);
if Abs(Im(L[1])) lt 10^(-r/2) then
    w := Re(L[1]);
else
    w := Re(L[2]);
end if;
if w lt 0 then
    w := -w; // need positive real period
end if;
//"w = ", w;

// Find the "higher" point
a1, a2, a3, a4, a6 := Explode(aInvariants(E));
_<y> := PolynomialRing(Parent(a1));
f := t^2 + a1*x*t + a3*t - x^3 - a2*x^2 - a4*x - a6;
tmp := Roots(f); // real roots
tmp := [y[1] : y in tmp]; // Chose y such that 2*y + a1*x + a3 >= 0
if (2*tmp[1] + a1*x + a3) ge 0 then
    y := tmp[1];
else
    y := tmp[2];
end if;
PC := Points(EC, x)[1]; // embed P to E(C)
if Abs(PC[2]-y) gt 10^(-r/2) then
    PC := -PC;
end if;
// Compute elliptic logarithm and scale it to [0,1]
elog := EllipticLog(EC, PC : Prec := r-10);
elog := Re(elog/w); // already real, just to make sure
return elog - Floor(elog); // mod 1, and force it to be real
end function;

/**
 * Find S^{(v)}(e1, e2)
 * Input:
 * e1, e2 = two real number with e1 < e2
 * Output:
 * a nested sequence representing disjoint union of subinterval
 **/
function FindS(E, e1, e2)
    if (e1 ge e2) then
        error "e1 must be less than e2";
    end if;
    // beta = max real root of f(x) = 4x^3 + ...
beta := (RootsF(E))[1];
    if (e2 lt beta) then
        return [];
    elseif (e1 lt beta) then
        e12 := NormalisedElog(E, e2);
        return [ [1 - e12, e12] ];
    else
        e11 := NormalisedElog(E, e1);
        e12 := NormalisedElog(E, e2);
        if (e11 eq 0.5) then
            if (e12 eq 0.5) then
                return [[0.5, 0.5]];
            else
                return [[1 - e12, e12]];
            end if;
        end if;
end function;
else
    return [ [1 - e12, 1 - e11], [e11, e12] ];
end if;
end if;
end function;

//*****************************************************************************/
* Main intrinsic functions
*****************************************************************************/
/**
* Compute S(v)_n(e1, e2)
* Input:
* E = elliptic curve over reals in standard Weierstrass form
* e1, e2 = two real numbers (with e1 < e2)
* n = a positive integer
* Output:
* a nested sequence representing a disjoint union of subintervals of [0,1]
**/
intrinsic FindSn(E::CrvEll, e1::FldReElt, e2::FldReElt, n::RngIntElt) -> SeqEnum
{Compute S(v)_n(e1, e2) for real archimedean place v}
    require e1 lt e2 : "e1 must be less than e2";
    require n gt 0: "n must be a positive integer";
    D := [];
    S := FindS(E, e1, e2);
    //print "n =", n, "S = ", S;
    s := #S;
    if s eq 0 then
        return [];
    end if;
    a := (S[1])[1];
    b := (S[1])[2];
    if s eq 1 then
        if (a eq 0) and (b eq 1) then
            return [[0, 1]];
        end if;
        for t := 0 to (n-1) do
            Append(~D, [(t+a)/n, (t+b)/n]);
        end for;
        return D;
    end if;
    // s = 2
    c := (S[2])[1];
    d := (S[2])[2];
    if (a eq 0) and (d eq 1) then
        if (a eq 0) and (d eq 1) then
            D := [[0, b/n]];
            for t := 0 to (n-2) do
                Append(~D, [(t+c)/n, (t+1+b)/n]);
            end for;
            Append(~D, [(n-1+c)/n, (n-1+d)/n]);
            return D;
        end if;
        // otherwise, ordinary append will do
        for t := 0 to (n-1) do
            Append(~D, [(t+a)/n, (t+b)/n]);
            Append(~D, [(t+c)/n, (t+d)/n]);
        end for;
        return D;
    end if;
    // otherwise, ordinary append will do
    for t := 0 to (n-1) do
        Append(~D, [(t+a)/n, (t+b)/n]);
        Append(~D, [(t+c)/n, (t+d)/n]);
    end for;
    return D;
end intrinsic;

/**
* Find the intersection of two disjoint unions of intervals
* Input:
* I1, I2 = nested sequences representing disjoint unions of intervals
*/
A.6 Height Bound IV: Intersection of Regions

The following files provide functions for computing the approximate corresponding region $T^{(v)}_n$ (see Section 3.3 for the definition) for each complex archimedean place $v$. Again, we use our implementation in Appendix A.1 to compute the period lattice of each complex embedding.

A.6.1 intersect_complex.m

This is the main file which does most of the task of computing $T^{(v)}_n$.

```plaintext
function HFPDiscretise(tau, n)
```

- **Output**: 
  - a nested sequences representing disjoint union of intervals

```
intrinsic Intersection(I1::SeqEnum, I2::SeqEnum) -> SeqEnum
(Find the intersection of two disjoint unions of intervals)
D:= [];
for I in I1 do
  for J in I2 do
    tmp := IntervalsMeet(I, J);
    if (#tmp ne 0) then
      Append(D, tmp);
    end if;
  end for;
end for;
return D;
end intrinsic;
```
A.6. Height Bound IV: Intersection of Regions 197

\[ dw1 := 1.; \]
\[ dw2 := \tau/2; \]
\[ pivot := 0.; \]
\[ U := \text{car}\langle \text{Parent}(\tau), \text{Integers()}\rangle; \]
\[ L := [U]; \]
\[ dw1 := dw1/n; \]
\[ dw2 := dw2/n; \]
\[ \text{for } i := 1 \text{ to } n \text{ do} \]
\[ \quad \text{for } j := 1 \text{ to } n \text{ do} \]
\[ \quad \quad \text{initial status is set to be '2' (uncertain) for all blocks} \]
\[ \quad \quad \text{Append}([L, \langle pivot, 2 \rangle]); \]
\[ \quad \quad \text{pivot} +=: dw2; \]
\[ \quad \text{end for}; \]
\[ \quad \text{pivot} := i*dw1; \]
\[ \text{end for}; \]
\[ \text{return} \langle dw1, dw2, L \rangle; \]
\[ \text{end function}; \]

/**
 * Plot a given brief half fundamental parallelogram into figure
 * (for debug only) - output is rotated-right of the real figure
 * Status - 0 = No (cell excluded), else = Yes (included)
 * Input:
 * L = sequence of digits representing cells in the H.F.P.
 **/
procedure GridPlot(L)
    // Check dimension = must be square (nxn)
    if not IsSquare(#L) then
        error "Dimension of L must be square";
    else
        dim := Integers() ! Sqrt(#L);
        i := 1;
        for c in L do
            if c eq 0 then
                printf "x";
            elif c eq 1 then
                printf "y";
            elif c eq 2 then
                printf "f";
            else
                printf "u";
            end if;
            if (i mod dim) eq 0 then
                printf "\n";
                i := 1;
            else
                i +=: 1;
            end if;
        end for;
    end if;
end procedure;

/****************************************************************************
 * Main intrinsic functions
 *************************************************************/

/**
 * Intersect two (brief version of) regions on half fundamental
 * parallelograms (HFP)
 * Input:
 * L1, L2 = sequence of true/false representing the region (true = in region)
 * Output:
 * A sequence of true/false representing the intersection
 **/
intrinsic IntersectTwoGrids(L1::SeqEnum[BoolElt], L2::SeqEnum[BoolElt])
    -> SeqEnum[BoolElt]
(Intersect two regions of the same size)
    // Both L1 and L2 must be of identical size.
// All entries in both L1, L2 are booleans.
require #L1 eq #L2: "Both HFPs must be of identical dimension";

n := #L1; // = #L2 as well
L := [];
for i := 1 to n do
  Append(~L, L1[i] and L2[i]);
end for;
return L;
end intrinsic;

/**
 * Divide a given region L by n. If L has dimension m-by-m, then the new
 * divided region has dimension (m*n)-by-(m*n)
 * Input:
 *  L = a sequence of true/false representing a region
 *  n = a positive integer
 * Output:
 *  a sequence of true/false representing a new region
 ***/
intrinsic DivByN(L::SeqEnum, n::RngIntElt) -> SeqEnum

(Division of region by n)
// Check dimension of L - must be square
require IsSquare(#L): "Dimension of L must be square";
require n ge 1: "n must be a positive integer";
oldDim := Integers()! Sqrt(#L);

// "shrinked" block: dimension = oldDim x oldDim
// split shrink block into oldDim columns
Cols := [];
for i := 1 to oldDim do
  Col := [];
  for j := oldDim*(i-1)+1 to oldDim*i do
    Append(~Col, L[j]);
  end for;
  Append(~Cols, Col);
end for;

// Build up each big column.
// Swap method - top of 1st column concats to top of oldDim-th column.
// Count the total entry - stop when # = oldDim * n
BigCol := [];
for i := 1 to oldDim do
  BigCol[i] := []; // initialise blank big column
end for;
for i := 1 to oldDim do
  ind := 1;
  repeat
    if (ind mod 2) eq 1 then
      // pick entry from i-th column
      BigCol[i] := BigCol[i] cat Cols[i];
    else
      BigCol[i] := BigCol[i] cat Reverse(Cols[oldDim-i+1]);
    end if;
    ind := ind + 1;
  until ind gt n;
end for;

// Concat oldDim big columns together, and keep doing this n times.
// Overall dim = (n*oldDim) x (n*oldDim)
LL := [];
for i := 1 to n do
  for j := 1 to oldDim do
    LL := LL cat BigCol[j];
  end for;
end for;
return LL;
end intrinsic;

/**
Magnify a brief half fundamental parallelogram by \( n \) folds

** Input:
- \( L \) = sequence of true/false representing a region, say, size \( m \times m \)
- \( n \) = a positive integer

** Output:
- a new sequence of true/false of size \( (n\times m) \times (n\times m) \)

```plaintext
intrinsic Magnify(L::SeqEnum, n::RngIntElt) -> SeqEnum
(Magnify a brief half fundamental parallelogram by \( n \) folds)
```

```plaintext
// Check dim of \( L \) - must be square
require IsSquare(#L): "Dimension of \( L \) must be square";
oldDim := Integers() ! Sqrt(#L);
require n ge 1: "\( n \) must be a positive integer";

// initialise BigCol \( n \) of them
BigCol := []
for i := 1 to n do
    BigCol[i] := []
end for;

LL := []
// for each cell, discretise it into nxn subcells
// do this by column
for i := 1 to (oldDim^2) do // no. of org. cells
    for j := 1 to n do // no. of subcol in each cell
        for k := 1 to n do // no. of copies of that org. cell to subcol
            Append(~BigCol[j], L[i]);
        end for;
    end for;
    if (i mod oldDim) eq 0 then // 1 old col done
        // concat those n subcols and put into LL
        for k := 1 to n do
            LL := LL cat BigCol[k];
        end for;
        // clear all n subcols
        BigCol := []
        for k := 1 to n do
            BigCol[k] := []
        end for;
    end if;
end for;
return LL;
end intrinsic;
```

Determine a region \( T^v_n(\xi) \).
This is done by simple methods, but still need more subtle way to confirm
that the pre-excluded cells are indeed excluded.

** Input:
- \( E \) = elliptic curve defined over \( \mathbb{C} \) in standard Weierstrass form
- \( \xi \) = an upper bound for \( |X| \), where \( Y^2 = 4X^3 + A*X + B \)
- initRes = initial resolution (dimension) for the region

** Output:
- ?

** Parameter:
- ShowPlot = if true, print the region

```plaintext
intrinsic ZRegion(E::CrvEll, xi::FldReElt, initRes::RngIntElt : ShowPlot := false) -> SeqEnum
(Find the region \( T^v(\xi) \))
```

```plaintext
prec := Precision(BaseRing(E));
L, _ := Explode(PeriodLattice(E: Prec := prec-10));
if Im(w2/w1) gt 0 then
    L := [w1, w2];
else
    L := [w2, w1];
end if;
L, _ := TransformLattice(L);
w1, w2 := Explode(L); tau := w2/w1;
```
b2 := bInvariants(E)[1];
u_xi := (x + Abs(b2)/12) * Abs(w1)^2; // U_{\xi}

// Create a discretisation of half fundamental parallelogram,
// both full version L and brief version C.
n := 2^initRes; // dimension needs to be a power of 2
// Format: L = <dw1, dw2, [<pivot, status>, ...]>
dw1, dw2, cells := HFPDiscretise(tau, n);
// initial C
C := [c[2] : c in cells]; // status of each cell (2 = uncertain)

// Stage 1: Four-Corner Test
for ind := 1 to (n^2) do
  pivot, _ := (cells[ind])[1];
  // Error term:
  // Using 23 terms yields accuracy of computing \wp(z)
  // within 50 decimal places
  err := EstimateWPMaxError(23, 23);
  // Temporary excluded the corner containing 0 (mod \Lambda)
  if (ind eq 1) or (ind eq (n*(n-1)+1)) then
    continue;
  end if;
  // find p(z,tau) where z are the four corners of the cell c
  if Abs(WeierstrassP([1, tau], pivot, 23)+err le u_xi then
    C[ind] := 1;
  elif Abs(WeierstrassP([1, tau], pivot + dw1, 23)+err le u_xi then
    C[ind] := 1;
  elif Abs(WeierstrassP([1, tau], pivot + dw2, 23)+err le u_xi then
    C[ind] := 1;
  elif Abs(WeierstrassP([1, tau], pivot + dw1 + dw2, 23)+err le u_xi then
    C[ind] := 1;
  end if;
if C[ind] eq 1 then
  // Stage 2: Identify "possibly false" boundary
  // set it to '0' first in order to distinguish the region
  // but is still subject to verification
  // left cell
  if ind gt n then
    if C[ind - n] ne 1 then
      C[ind - n] := 0;
    end if;
  end if;
  // right cell
  if ind le n*(n-1) then
    if C[ind + n] ne 1 then
      C[ind + n] := 0;
    end if;
  end if;
  // top cell
  if (ind mod n) ne 0 then
    if C[ind + 1] ne 1 then
      C[ind + 1] := 0;
    end if;
  end if;
  // bottom cell
  if (ind mod n) ne 1 then
    if C[ind - 1] ne 1 then
      C[ind - 1] := 0;
    end if;
  end if;
end if;

// Stage 3: Confirm "possibly false" boundary that it is indeed excluded
boundaryConfirmed := true;
for ind := 1 to (n^2) do
  if C[ind] eq 0 then
    // lower horizontal
A.6. Height Bound IV: Intersection of Regions

\[ m := 0.; \quad k := \text{Im}(\text{pivot}); \]
\[ X := [\text{Re}(\text{pivot}), \text{Re}(\text{pivot})+dw1]; \]
if HasBoundary(X, m, k, u_xi, tau) then
  \( C[\text{ind}] := 1; \)
  //printf "C[\text{ind}] lower boundary not confirmed\n", ind;
  boundaryConfirmed := false;
  //printf "=*=*=*=*=*";
  continue;
end if;

// upper horizontal
\[ m := 0.; \quad k := \text{Im}(\text{pivot} + dw2); \]
\[ X := [\text{Re}(\text{pivot}+dw2), \text{Re}(\text{pivot}+dw1+dw2)]; \]
if HasBoundary(X, m, k, u_xi, tau) then
  \( C[\text{ind}] := 1; \)
  //printf "C[\text{ind}] upper boundary not confirmed\n", ind;
  boundaryConfirmed := false;
  //printf "=*=*=*=*=*";
  continue;
end if;

// ADDED: 18 Nov 2010
// left/right vertical (i.e. Re(tau)=0)
if Re(tau) eq 0 then
  if HasBoundaryVertical(\text{Re}(\text{pivot}), \text{Im}(\text{pivot}), \text{Im}(\text{pivot}+dw2), u_xi, tau) then
    \( C[\text{ind}] := 1; \)
    //printf "C[\text{ind}] left vertical boundary not confirmed\n", ind;
    boundaryConfirmed := false;
    //printf "=*=*=*=*=*";
    continue;
  end if;
end if;

if HasBoundaryVertical(\text{Re}(\text{pivot}+dw1), \text{Im}(\text{pivot}+dw1), \text{Im}(\text{pivot}+dw1+dw2), u_xi, tau) then
  \( C[\text{ind}] := 1; \)
  //printf "C[\text{ind}] right vertical boundary not confirmed\n", ind;
  boundaryConfirmed := false;
  //printf "=*=*=*=*=*";
  continue;
end if;
else
  // left slant
  \[ n := \text{Im}(\text{tau})/\text{Re}(\text{tau}); \quad k := \text{Im}(\text{pivot}) - n*\text{Re}(\text{pivot}); \]
  if (n lt 0) then
    \[ X := [\text{Re}(\text{pivot}+dw2), \text{Re}(\text{pivot})]; \]
  elif (n gt 0) then
    \[ X := [\text{Re}(\text{pivot}), \text{Re}(\text{pivot}+dw2)]; \]
  end if;
  if HasBoundary(X, m, k, u_xi, tau) then
    \( C[\text{ind}] := 1; \)
    //printf "C[\text{ind}] left boundary not confirmed\n", ind;
    boundaryConfirmed := false;
    //printf "=*=*=*=*=*";
    continue;
  end if;
  // right slant
  \[ n := \text{Im}(\text{tau})/\text{Re}(\text{tau}); \quad k := \text{Im}(\text{pivot}) - n*\text{Re}(\text{pivot}+dw1); \]
  if (m lt 0) then
    \[ X := [\text{Re}(\text{pivot}+dw1+dw2), \text{Re}(\text{pivot}+dw1)]; \]
  elif (m gt 0) then
    \[ X := [\text{Re}(\text{pivot}+dw1), \text{Re}(\text{pivot}+dw1+dw2)]; \]
  end if;
  if HasBoundary(X, m, k, u_xi, tau) then
    \( C[\text{ind}] := 1; \)
    printf "C[\text{ind}] right boundary not confirmed\n", ind;
    boundaryConfirmed := false;
    //printf "=*=*=*=*=*";
  else
    // cell can be excluded
    \( C[\text{ind}] := 0; \)
  if ShowPlot then

### Appendix A. MAGMA Source Code

```magma
printf "C[%o] = %o\n", ind, C[ind];
end if;
//print "----------"
end if;
end if;
end if;
end if;

// Stage 4: shade region correctly
if boundaryConfirmed then
  if ShowPlot then
    print "Excluded boundary confirmed! Shading remaining region ...";
  end if;
  for ind := 1 to (n^2) do
    if C[ind] eq 2 then
      C[ind] := 0; // 0 = exclude from the region
    end if;
  end for;
else
  if ShowPlot then
    print "Excluded boundary not entirely confirmed";
  end if;
end if;

if ShowPlot then
  GridPlot(C);
printf "\n"
print "Size Z = ", #C;
printf "---------------------------"
end if;
return C;
end intrinsic;

/**
 * Convert grid entry from 0 -> false and from 1 -> true.
 * If contains any other number, print error message
 * Input:
 *  C = sequence of binaries
 * Output:
 *  sequence of true/false
 **/
intrinsic GridEntryTransform(C::SeqEnum[RngIntElt]) -> SeqEnum[BoolElt]
(Transform all binaries in a grid into true/false entries)
L := [];
for c in C do
  if c eq 0 then
    Append(L, false);
  elif c eq 1 then
    Append(L, true);
  else
    error "Grid must only contain 0 and 1"
  end if;
end for;
return L;
end intrinsic;
```

#### A.6.2 wp.m

This file consists of functions for computing the approximate Weierstrass $\wp$-function (i.e., using only finite number of terms in the power series expansion), and the maximum error caused by such approximation. For more details, see Section 3.2.1.
### A.6. Height Bound IV: Intersection of Regions

/* wp.m
* Computing Weierstrass \wp-function and the error term of using finite
* number of terms in the power series expansion as the approximate for \wp(z)
* By Thotsaphon Thongjunthug
* Last updated: 08 December 2010
* Any errors should be reported to <nookaussie@yahoo.com>
***************************************************************************/

/**
* Main intrinsic functions
***************************************************************************/

intrinsic EstimateWPMaxError(n1::RngIntElt, n2::RngIntElt : q := Exp(-Sqrt(3)*Pi(RealField())), alpha := 1/2) -> FldReElt
{Estimate the error when using finite number of terms in the power series expansion as the approximate to the Weierstrass p-function}
require (n1 ge 1) and (n2 ge 1): "n1 and n2 must be positive integers";
pi := Pi(RealField());
err := (q^(n1 + alpha))/(1 - q^(n1 + alpha))^2;
err += (q^(n1 - alpha))/(1 - q^(n1 - alpha))^2;
err += 2*(q^n2)/(1 - q^n2)^2;
err *= 4*(pi^2);
err /= (1 - q);
return err;
end intrinsic;

intrinsic WeierstrassP(L::SeqEnum, z::FldComElt, k::RngIntElt) -> FldComElt
{Compute Weierstrass p-function of z using up to k terms in the power series expansion}
require #L eq 2: "L must contain exactly two complex numbers";
p1 := Pi(RealField());
err := (q'^(n1 + alpha))/(1 - q'^(n1 + alpha))^2;
err += (q'^(n1 - alpha))/(1 - q'^(n1 - alpha))^2;
err += 2*(q'n2)/(1 - q'n2)^2;
err *= 4*(pi'^2);
err /= (1 - q');
return err;
end intrinsic;

/**
* Estimate the maximum error of |p(z,tau)-f(u)*(2*pi*i)^2| on the lower-half
* fundamental parallelogram, where u = Exp(2*pi*i*z) and
* f(u) = u/(1-u)^2 + 1/12 + some terms from the right-hand side of the
* expression of p(z,tau) (See Silverman's paper).
* Input:
* n1 = starting index for the infinite sum of terms
* \( (q^n)u/(1-(q^n)u)^2 + (q^n)(1/u)/(1-(q^n)(1/u))^2 \)
* n2 = starting index for the infinite sum of terms
* \( 2*(q^n)/(1-q^n)^2 \)
* Output:
* the maximum absolute error
* Parameter:
* q = Exp(2*pi*i*tau), default = Exp(-Sqrt(3)*pi)
* alpha = Im(z)/Im(tau), default = 1/2
* (The default values of q and alpha are ones of the worst case scenario)
**
(Estimate the error when using finite number of terms in the power series expansion as the approximate to the Weierstrass p-function)
if z eq 0 then
  error "z is a lattice point";
end if;

// Compute precision needed, based on the error term
err := EstimateWPMaxError(k, k);
err := Ceiling(-Log(10, err));
C := Parent(tau);
pi := Pi(C); i := C!Sqrt(-1);
q := Exp(2*pi*i*tau);
u := Exp(2*pi*i*z); // since already let z <- z/w1
f := 1/12 + u/(1-u)^2;
// Reset n up to k-1
// (22 term -> max abs. err. = 3.4291 x 10^-52)
// (50 term -> max abs. err. = 2.3251 x 10^-118)
for n := 1 to (k-1) do
  tmp := u * (1/(1 - (q^n)*u)^2 + 1/((q^n) - u)^2);
  tmp -:= 2/(1 - q^n)^2;
  tmp *:= q^n;
  f += tmp;
end for;

f := (2*pi*i/w1)^2;
return f;
end intrinsic;

/**
 * Calculate the 1st derivative of the Weierstrass \wp-function p(z)
 * for a given z approximately, using Alg 7.4.5, and a finite of terms
 * in Prop 7.4.4 in Cohen's book
 * Input:
 * L = [w1, w2], the periods of L that E(C) = C/
 * z = a complex number that we want to find \wp(z, L)
 * k = number of terms in the infinite sum
 */
intrinsic WeierstrassPDash(L::SeqEnum, z::FldComElt, k::RngIntElt)
-> FldComElt

// Compute the value of the first derivative of Weierstrass \wp-function at z,
// where z is given with respect to the fundamental parallelogram spanned by L.
// This function uses the first k-1 terms in the infinite sum formula)
require #L eq 2: "L must contain exactly two complex numbers";
w1, w2 := Explode(L);
require Im(w2/w1) gt 0: "Im(w2/w1) must be positive";

L := TransformLattice(L);
w1, w2 := Explode(L);
tau := w2/w1;
// reduce z
z := z/w1;
if z eq 0 then
  error "z is a lattice point";
end if;
C := Parent(w1);
i := C!Sqrt(-1);
pi := Pi(C);
q := Exp(2*pi*i*tau);
u := Exp(2*pi*i*z); // since already let z <- z/w1
f := (1 + u)/(1 - u)^3;
for n := 1 to (k-1) do
  tmp := ((q^n)*u)/(1 - (q^n)*u)^3;
  tmp += ((q^n) + u)/((q^n) - u)^3;
  tmp *:= q^n;
  f += tmp;
end for;
A.6 Height Bound IV: Intersection of Regions

\[ f := u \cdot (2\pi i/u1)^3; \]
\[ \text{return } f; \]
end intrinsic;

A.6.3 interval_wp.m

This file involves computing the interval version of the function \( f \) mentioned in Proposition 3.2.3. Note that this also requires some basic arithmetic operations on real intervals, which are implemented as shown in the next subsection.

```plaintext
// required for recursive definition
forward HasBoundary;
forward HasBoundaryVertical;

/*********************************************************************************
* Auxiliary local functions
*********************************************************************************/
/**
* (verified 13 Jul 08)
* Compute the real part of a slant boundary \( y = m \cdot x + c \)
* Input:
* \( L = [x1, x2] = \) range of \( x \)
* \( m = \) gradient of the boundary (\( m = 0 \Rightarrow \) horizontal boundary)
* \( c = \) constant
* \( k = \) number of terms used in the power series expansion
* \( \tau = \) represent the fundamental parallelogram \( \{1, \tau\} \)
* Output:
* interval containing the range of the real part
**/
function realPartSlant(L, m, c, k, tau)
if #L ne 2 then
    error "L must contain exactly two real numbers";
end if;
x1, x2 := Explode(L);
if x1 gt x2 then
    error "x1 must be < or = x2";
end if;
p1 := Pi(Parent(x1));
y1 := -2*pi*(m*x1 + c);
y2 := -2*pi*(m*x2 + c);
if y1 le y2 then
    Y := [y1, y2];
else
    Y := [y2, y1];
end if;
orgY := Y;
Y := invExp(Y);
// Re( u/(1-u)^2 )
// denominator
```
denom := invMul([2., 2.], invCos([2*pi*x1, 2*pi*x2]));
denom := invSub(Y, denom);
denom := invMul(Y, denom);
denom := invAdd([1., 1.], denom);
// print "denom 1 = ", denom;
// In case the interval contains negative numbers, try to adjust it
if denom[1] le 0 then
    // print "problem here (real part 1)"
    // print "Cautio.n at L = ", L;
    // print "Y = ", Y;
    // print "m = ", m;
    // print "denom 1 = ", denom;
    // test for existence of local extremum in L
    diffden := invPow(Y, 2);
    diffden := invSub(invCos([2*pi*x1, 2*pi*x2]), diffden);
    diffden := invMul(diffden, [m, m]);
    diffden := invAdd(invSin([2*pi*x1, 2*pi*x2]), diffden);
    diffden := invMul(diffden, Y);
    if (diffden[1] le 0) and (0 le diffden[2]) then
        // may have local extrem. - should not happen
        error "0 is in denominator interval, please report!"
    else
        // strict behaviour
        denx1 := 1 - 2*Exp(y1)*Cos(2*pi*x1) + Exp(2*y1);
        denx2 := 1 - 2*Exp(y2)*Cos(2*pi*x2) + Exp(2*y2);
        denom := [Min(denx1, denx2), Max(denx1, denx2)];
        // print "Denom 1 modified = "", denom; // successfully modified
        // print "***************************";
        end if;
end if;
// numerator
numer := invAdd([1., 1.], invPow(Y, 2));
numer := invMul(numer, invCos([2*pi*x1, 2*pi*x2]));
numer := invSub(numer, invMul([2., 2.], Y));
numer := invMul(numer, Y);
rp := invDiv(numer, invPow(denom, 2));
for n := 1 to (k-1) do
    aqn := Exp(-2*pi*Im(tau)*n); // = |q^n|
    A := Cos(2*pi*n*Re(tau));
    B := Sin(2*pi*n*Re(tau));
    qnu := invAdd(orgY, [-2*pi*Im(tau)*n, -2*pi*Im(tau)*n]);
    qnu := invExp(qnu);
    // Re (q^n*u / (1 - q^n*u)^2)
    // denominator
    denom := invMul([A, A], invCos([2*pi*x1, 2*pi*x2]));
    denom := invSub(denom, invMul([B, B], invSin([2*pi*x1, 2*pi*x2])));
    denom := invMul(denom, [2., 2.]);
    denom := invSub(qnu, denom);
    denom := invMul(denom, qnu);
    denom := invSub([1., 1.], denom);
end for;
// print "denom 2 = ", denom;
// numerator
numer := invMul([A, A], invCos([2*pi*x1, 2*pi*x2]));
numer := invSub(numer, invMul([B, B], invSin([2*pi*x1, 2*pi*x2])));
numer := invMul(numer, invAdd([1., 1.], invPow(qnu, 2)));
numer := invSub(numer, invMul([2., 2.], qnu));
numer := invMul(numer, qnu);
rp := invAdd(rp, invDiv(numer, invPow(denom, 2)));
A.6. Height Bound IV: Intersection of Regions

//print "denom 3 = ", denom;
// numerator
numer := invMul([A, A], invCos([2*pi*x1, 2*pi*x2]));
numer := invAdd(numer, invMul([B, B], invSin([2*pi*x1, 2*pi*x2])));
numer := invMul(numer, invAdd([aqn^2, aqn^2], invPow(Y, 2)));
numer := invSub(numer, invMul([2., 2.], qnu));
numer := invMul(numer, qnu);
//print "real num 3 = ", numer;
rp := invAdd(rp, invDiv(numer, invPow(denom, 2)));

// Constant real part
// denominator
denom := [1 - 2*A*aqn + aqn^2, 1 - 2*A*aqn + aqn^2];
//print "denom 4 = ", denom;
// numerator
numer := [aqn*A*(1+aqn^2) - 2*aqn^2, aqn*A*(1+aqn^2) - 2*aqn^2];
numer := invMul(numer, [2., 2.]);
rp := invSub(rp, invDiv(numer, invPow(denom, 2)));
end for;
rp := invMul(rp, [-4*pi, -4*pi]);
//print "rp = ", rp;
return rp;
end function;

/**
 * (verified 13 Jul 08)
 * Compute the imaginary part of a slant boundary y = m*x + c
 * Input:
 * L = [x1, x2] = range of x
 * m = gradient of the boundary (m = 0 ==> horizontal boundary)
 * c = constant
 * k = number of terms used in the power series expansion
 * tau = represent the fundamental parallelogram (1, tau)
 * Output:
 * interval containing the range of the imaginary part
 **/
function imPartSlant(L, m, c, k, tau)
if #L ne 2 then
    error "L must have exactly 2 real numbers";
end if;
x1, x2 := Explode(L);
if x1 gt x2 then
    error "x1 must be <= x2";
end if;
p1 := Pi(Parent(x1));
y1 := -2*pi*(m*x1 + c);
y2 := -2*pi*(m*x2 + c);
if y1 le y2 then
    Y := [y1, y2];
else
    Y := [y2, y1];
end if;
orgY := Y;
Y := invExp(Y);
// Im( u/(1-u)^2 )
// denominator
denom := invMul([2., 2.], invCos([2*pi*x1, 2*pi*x2]));
denom := invSub(Y, denom);
denom := invMul(Y, denom);
denom := invAdd([1., 1.], denom);
// In case the interval contains negative numbers, try to adjust it
if denom[1] le 0 then
    error "problem here (Im part 1)";
    //print "Caution at L = ", L;
    //print "Y = ", Y;
    //print "m = ", m;
    //print "denom in 1 = ", denom;
    // test for existence of local extremum in L
diffden := invPow(Y, 2);
diffden := invSub(invCos([2*pi*x1, 2*pi*x2]), diffden);
diffden := invMul(diffden, [m, m]);
diffden := invAdd(invSin([2*pi*x1, 2*pi*x2]), diffden);
diffden := invMul(diffden, Y);
if (diffden[1] le 0) and (0 le diffden[2]) then
  // may have local extremum - should not happen
  error "0 is in denominator interval, please report!";
else
  // strict behaviour
  denx1 := 1 - 2*Exp(y1)*Cos(2*pi*x1) + Exp(2*y1);
denx2 := 1 - 2*Exp(y2)*Cos(2*pi*x2) + Exp(2*y2);
denom := [Min(denx1, denx2), Max(denx1, denx2)];
  //print "Denom 1 modified = ", denom; // successfully modified
  //print "**********************";
end if;
end if;

// numerator
numer := invSub([1., 1.], invPow(Y, 2));
numer := invMul(numer, Y);
numer := invMul(numer, invSin([2*pi*x1, 2*pi*x2]));
ip := invDiv(numer, invPow(denom, 2));
for n := 1 to (k-1) do
  aqn := Exp(-2*pi*Im(tau)*n); // = |q^n|
  A := Cos(2*pi*n*Re(tau));
  B := Sin(2*pi*n*Re(tau));
  qnu := invAdd(orgY, [-2*pi*Im(tau)*n, -2*pi*Im(tau)*n]);
  qnu := invExp(qnu);
  // Im (q^n*u / (1 - q^n*u)^2)
  // denominator
  denom := invMul([A, A], invCos([2*pi*x1, 2*pi*x2]));
denom := invSub(denom, invMul([B, B], invSin([2*pi*x1, 2*pi*x2])));
denom := invMul(denom, [2., 2.]);
denom := invSub(denom, qnu, denom);
denom := invMul(denom, qnu);
denom := invSub([1., 1.], denom);
  //print "denom in 2 = ", denom;
  // numerator
  numer := invMul([B, B], invCos([2*pi*x1, 2*pi*x2]));
numer := invAdd(numer, invMul([A, A], invSin([2*pi*x1, 2*pi*x2])));
numer := invMul(numer, qnu);
numer := invMul(numer, invSub([1., 1.], invPow(qnu, 2)));
  ip := invAdd(ip, invDiv(numer, invPow(denom, 2)));
  // Im( q^n*u / (q^n - u)^2 )
  // denominator
  denom := invMul([A, A], invCos([2*pi*x1, 2*pi*x2]));
denom := invAdd(denom, invMul([B, B], invSin([2*pi*x1, 2*pi*x2])));
denom := invMul(denom, qnu);
denom := invMul(denom, [2., 2.]);
denom := invSub(denom, aqn^2, denom);
denom := invAdd(denom, invPow(Y, 2));
  //print "denom in 3 = ", denom;
  // numerator
  numer := invMul([A, A], invSin([2*pi*x1, 2*pi*x2]));
numer := invAdd(numer, invMul([B, B], invCos([2*pi*x1, 2*pi*x2])));
numer := invMul(numer, qnu);
numer := invMul(numer, invSub([aqn^2, aqn^2], denom));
  //print "num in 3 = ", numer;
ip := invAdd(ip, invDiv(numer, invPow(denom, 2)));
  // Constant part
  // denominator
  denom := [1 - 2*A*aqn + aqn^2, 1 - 2*A*aqn + aqn^2];
  //print "denom in 4 = ", denom;
  // numerator
  numer := [aqn*B*(1-aqn^2), aqn*B*(1-aqn^2)];
numer := invMul(numer, [2., 2.]);
ip := invSub(ip, invDiv(numer, invPow(denom, 2)));
end for;
** Added 18 Nov 2010
* Compute the real part for a vertical boundary (rare case)
* Input:
  * x = fixed x-coordinate
  * c, d = range of y-coordinates (c <= d)
  * k = number of terms used in the power series expansion
  * tau = the fundamental parallelogram \{1, tau\} (here, Re(tau) = 0)
* Output:
  * a real interval containing the range of the real part.
**/ 
function realPartVertical(x, c, d, k, tau)
  if c gt d then
    error "c must be < or = d"
  end if;
  pi := Pi(Parent(x));
  y1 := -2*pi*c; y2 := -2*pi*d;
  Y := [y2, y1];
  orgY := Y;
  Y := invExp(Y);
  //print "Y = ", Y;
  // Re( u/(1-u)^2 )
  // denominator
  denom := invMul([2., 2.], invCos([2*pi*x, 2*pi*x]));
  denom := invSub(Y, denom);
  denom := invMul(Y, denom);
  //print "denom 1 = ", denom;
  // numerator
  numer := invAdd([1., 1.], invPow(Y, 2));
  numer := invMul(numer, invCos([2*pi*x, 2*pi*x]));
  numer := invSub(numer, invMul([2., 2.], Y));
  numer := invMul(numer, Y);
  rp := invDiv(numer, invPow(denom, 2));
  rp := invAdd(rp, [1./12, 1./12]);
  for n := 1 to (k-1) do
    qn := Exp(-2*pi*Im(tau)*n); // = |q^n|
    qnu := invAdd(orgY, [-2*pi*Im(tau)*n, -2*pi*Im(tau)*n]);
    qnu := invExp(qnu);
    // Re (q^n*u / (1 - q^n*u)^2)
    // denominator
    denom := invCos([2*pi*x, 2*pi*x]);
    denom := invSub(denom, [2., 2.]);
    denom := invSub(denom, qnu);
    denom := invSub([1., 1.], denom);
    //print "denom 2 = ", denom;
    // numerator
    numer := invMul([2*pi*x, 2*pi*x]);
    numer := invSub(numer, invAdd([1., 1.], invPow(qnu, 2)));
    numer := invMul(numer, Y);
    numer := invMul(numer, Y);
    rp := invDiv(rp, invDiv(numer, denom));
    rp := invAdd(rp, [1./12, 1./12]);
    for n := 1 to (k-1) do
      qn := Exp(-2*pi*Im(tau)*n); // = |q^n|
      qnu := invAdd(orgY, [-2*pi*Im(tau)*n, -2*pi*Im(tau)*n]);
      qnu := invExp(qnu);
      // Re (q^n*u / (1 - q^n*u)^2)
      // denominator
      denom := invCos([2*pi*x, 2*pi*x]);
      denom := invSub(denom, [2., 2.]);
      denom := invSub(denom, qnu);
      denom := invSub([1., 1.], denom);
      //print "denom 3 = ", denom;
      // numerator
      numer := invMul([2*pi*x, 2*pi*x]);
      numer := invSub(numer, invAdd([1., 1.], invPow(qnu, 2)));
module MAGMA

// numerator
numer := invCos([2*pi*x, 2*pi*x]);
numer := invMul(numer, invAdd([aqn^2, aqn^2], invPow(Y, 2)));
numer := invSub(numer, invMul([2., 2.], qnu));
numer := invMul(numer, qnu);

// print "real num 3 = ", numer;
rp := invAdd(rp, invDiv(numer, invPow(denom, 2)));

// Constant real part

// denominator
denom := [1 - 2*aqn + aqn^2, 1 - 2*aqn + aqn^2];

// print "denom 4 = ", denom;

// numerator
numer := [aqn*(1+aqn^2) - 2*aqn^2, aqn*(1+aqn^2) - 2*aqn^2];
numer := invMul(numer, [2., 2.]);

//print "numerator 3 = ", numer;
rp := invSub(rp, invDiv(numer, invPow(denom, 2)));

end for;
renum := invMul(rp, [-4*pi, -4*pi]);

//print "rp = ", rp;
return rp;
end function;

/**
 * Added 18 Nov 2010
 * Compute the imaginary part for a vertical boundary (rare case)
 * Input:
 * x = fixed x-coordinate
 * c, d = range of y-coordinates (c <= d)
 * k = number of terms used in the power series expansion
 * tau = the fundamental parallelogram {1, tau} (here, Re(tau) = 0)
 * Output:
 * a real interval containing the range of the imaginary part.
 **/ 
function imPartVertical(x, c, d, k, tau)
if c gt d then
    error "c must be <= or = d";
end if;
pi := Pi(Parent(x));
y1 := -2*pi*c; y2 := -2*pi*d;
Y := [y2, y1];
orgY := Y;
Y := invExp(Y);

// Im( u/(1-u)^2 )
// denominator
denom := invMul([2., 2.], invCos([2*pi*x, 2*pi*x]));
denom := invSub(Y, denom);
denom := invMul(Y, denom);
denom := invAdd([1., 1.], denom);

// numerator
numer := invSub([1., 1.], invPow(Y, 2));
numer := invMul(numer, Y);
numer := invMul(numer, invSin([2*pi*x, 2*pi*x]));
ip := invDiv(numer, invPow(denom, 2));

for n := 1 to (k-1) do
    aqn := Exp(-2*pi*Im(tau)*n); // = |q^n|
    qnu := invAdd(orgY, [-2*pi*Im(tau)*n, -2*pi*Im(tau)*n]);
    qnu := invExp(qnu);
    // Im (q^n*u / (1 - q^n*u)^2)
    // denom
denom := invCos([2*pi*x, 2*pi*x]);
denom := invMul(denom, [2., 2.]);
denom := invSub(qnu, denom);
denom := invMul(denom, qnu);
denom := invSub([1., 1.], denom);
    //print "denom in 2 = ", denom;
end for;
numer := invAdd(numer, invSin([2*pi*x, 2*pi*x]));
numer := invMul(numer, qnu);
end function;
A.6. Height Bound IV: Intersection of Regions

numer := invMul(numer, invSub([1., 1.], invPow(qnu, 2)));  
ip := invAdd(ip, invDiv(numer, invPow(denom, 2)));  
// Im( q^n*u / (q^n - u)^2 )  
// denominator  
denom := invCos([2*pi*x, 2*pi*x]);  
denom := invMul(denom, qnu);  
denom := invSub([aqn^2, aqn^2], denom);  
denom := invAdd(denom, invPow(Y, 2));  
//print "denom im 3 = ", denom;  
// numerator  
numer := invSin([2*pi*x, 2*pi*x]);  
numer := invMul(numer, qnu);  
numer := invMul(numer, invSub([aqn^2, aqn^2], invPow(Y, 2)));  
//print "num im 3 = ", numer;  
ip := invAdd(ip, invDiv(numer, invPow(denom, 2)));  
end for;  
ip := invMul(ip, [-4*pi, -4*pi]);  
return ip;  
end function;  

/******************************************************************************  
* Main intrinsic functions  
*******************************************************************************/  
  
/**  
* Check if there is part of the boundary of the region R^v(\xi) on a given  
* (slant or horizontal) boundary y = m*x + c of a parallelogram.  
* Input:  
* L = [x1, x2] = range of x-coordinates  
* m = the gradient of the parallelogram's boundary  
* c = a constant  
* B = bound for the function f(X1, X2, X3) (normally is the U_\xi)  
* tau = representing the fundamental parallelogram (1, tau)  
* Output:  
* true if we (suspect) that it may contain part of R^v(\xi); false otherwise  
/**  
intrinsic HasBoundary(L::SeqEnum, m::FldReElt, c::FldReElt, B::FldReElt,  
tau::FldComElt) -> BoolElt  
(Check if there is part of the boundary of the region R^v(\xi) on a given slant  
(or horizontal) boundary of a parallelogram.)  
require #L eq 2: "L must have exactly two real numbers";  
x1, x2 := Explode(L);  
require x1 le x2: "x1 must be < or = x2";  
require B ge 0: "B must be non-negative";  
require Im(tau) ge 0: "Im(tau) must be non-negative";  
r := realPartSlant(L, m, c, 23, tau);  
if #r eq 1 then  
// print "*** Null denominator ***";  
// return true;  
//end if;  
err := EstimateWPMaxError(23, 23);  
F := invPov(r, 2);  
F := invAdd(F, invPov(imPartSlant(L, m, c, 23, tau), 2));  
F := [Sqrt(F[1]), Sqrt(F[2])];  
F := invSub(F, [B, B]);  
F := [F[1]-err, F[2]+err];  
if not ((F[1] le 0) and (0 le F[2])) then  
return false;  
end if;  
// If still not return false, try to bisect the interval and check  
midPt := (x1 + x2)/2;  
L1 := [x1, midPt];  
L2 := [midPt, x2];  
L1 := HasBoundary(L1, m, c, B, tau);  
L2 := HasBoundary(L2, m, c, B, tau);  
if not(L1) and not(L2) then
\textbf{A.7 Height Bound V: Interval Arithmetic}

This file provides some basic arithmetic operations on real intervals. For more information on the subject of interval arithmetic, see, e.g., \cite{Moo66}.

/*
 * interval_arith.m
 * Functions for basis arithmetic on real intervals
 * By Thotsaphon Thongjunthug
 * Last updated: 08 December 2010
 * Any errors should be reported to <nookaussie@yahoo.com>
 */
/*****************************************************************************
* Unary Operations
*****************************************************************************/
/**
* Cosine function for the interval [a, b]
* Input:
*  L = [a, b] with a <= b
**/
intrinsics invCos(L::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
(Cosine function for a real interval L = [a, b].)
  // Check validity of L = [a, b]
  require #L eq 2: "Interval must contain exactly two real numbers";
  a, b := Explode(L);
  require a le b: "a must be < or = b";
  // Check if L contains any multiples of Pi
  pi := Pi(Parent(a));
  lb := Ceiling(a/pi);
  ub := Floor(b/pi);
  width := ub - lb;
  if ((lb*pi) gt b) or ((ub*pi) lt a) then
    // L contains no extremum for cos function
    return [Min(Cos(a), Cos(b)), Max(Cos(a), Cos(b))];
  elseif width gt 0 then
    // L contains both even and odd multiples of pi, so return [-1, 1]
    return [-1, 1];
  elseif (lb mod 2) eq 0 then
    // lb = ub, and is even, then maximum of cosine = 1
    return [Min(Cos(a), Cos(b)), 1];
  else
    // minimum of cosine = -1
    return [-1, Max(Cos(a), Cos(b))];
  end if;
end intrinsic;

/**
* Sine function for the interval [a, b]
* Input:
*  L = [a, b], with a <= b
**/
intrinsics invSin(L::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
(Sine function for a real interval L = [a, b].)
  // Check validity of L = [a, b]
  require #L eq 2: "Interval must contain exactly two real numbers";
  a, b := Explode(L);
  require a le b: "a must be < or = b";
  // Check if L contains any multiples of Pi/2
  pi := Pi(Parent(a));
  lb := Ceiling(2*a/pi);
  ub := Floor(2*b/pi);
  // only care the odd multiple of pi/2 - N . 8/7/08
  if (lb mod 2) eq 0 then
    lb := lb + 1;
  end if;
  if (ub mod 2) eq 0 then
    ub := ub - 1;
  end if;
  width := (ub - lb)/2;
  // Case 1: when L contains no odd multiple of Pi/2
  if ((lb*pi/2) gt b) or ((ub*pi/2) lt a) then
    return [Min(Sin(a), Sin(b)), Max(Sin(a), Sin(b))];
  end if;
  // Case 2: width >= 1, so L contains two odd multiples of Pi/2
  if width gt 1 then
    return [-1, 1];
  end if;
// Case 2: width = 0 (so lb = ub)
if (lb mod 4) eq 1 then
  // Max of sin = 1
  return [Min(Sin(a), Sin(b)), 1];
else
  // Min of sin = -1
  return [-1, Max(Sin(a), Sin(b))];
end if;
end intrinsic;

/**
 * Exponential function for the interval
 * Input:
 * L = [a, b], with a <= b
 */
intrinsic invExp(L::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
{Exponential function for a real interval L = [a, b].}
// Check validity of L = [a, b]
require #L eq 2: "Interval must contain exactly two real numbers";
  a, b := Explode(L);
  require a le b: "a must be < or = b";
  return [Exp(a), Exp(b)];
end intrinsic;

/**
 * Let L be an interval, compute L^n = L * ... * L, n times
 * Input:
 * L = [a, b], with a <= b
 * n = a non-negative integer
 */
intrinsic invPow(L::SeqEnum[FldReElt], n::RngIntElt) -> SeqEnum[FldReElt]
{Compute n-th power of a real interval L.}
// Check validity of L = [a, b]
require #L eq 2: "Interval must contain exactly two real numbers";
  a, b := Explode(L);
  require a le b: "a must be < or = b";
  require n ge 0: "n must be a non-negative integer";
  if (n mod 2) eq 0 then
    // take care when a and b is of different sign
    if (a le 0) and (b ge 0) then
      return [0, Max(a^n, b^n)];
    else
      return [Min(a^n, b^n), Max(a^n, b^n)];
    end if;
  end if;
end intrinsic;

////////////////////////////////////////////////////////////////////////
* Binary Operations
////////////////////////////////////////////////////////////////////////
/**
 * Add two real intervals
 * Input:
 * L, K = real intervals
 */
intrinsic invAdd(L::SeqEnum[FldReElt], K::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
{Add two real intervals.}
require (#L eq 2) and (#K eq 2):
  "Both L, K must contain exactly two real numbers";
  a, b := Explode(L);
  c, d := Explode(K);
  require (a le b) and (c le d): "Check if a <= b and c <= d";
  return [a+c, b+d];
end intrinsic;
A.7. Height Bound V: Interval Arithmetic

/**
* For real intervals L, K, compute L - K
* Input:
*  L, K = real intervals
**/
intrinsic invSub(L::SeqEnum[FldReElt], K::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
(Subtract two real intervals.)
return invAdd(L, [-K[2], -K[1]]);
end intrinsic;

/**
* For real intervals L, K, compute L*K
* Input:
*  L, K = real intervals
**/
intrinsic invMul(L::SeqEnum[FldReElt], K::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
(Multiply two real intervals.)
require (#L eq 2) and (#K eq 2):
"Both L, K must contain exactly two real numbers";
a, b := Explode(L);
c, d := Explode(K);
require (a le b) and (c le d): "Check if a <= b and c <= d"
lb, _ := Min([a*c, a*d, b*c, b*d]);
ub, _ := Max([a*c, a*d, b*c, b*d]);
return [lb, ub];
end intrinsic;

/**
* For real intervals L, K = [c, d], compute L/K = L*[1/d, 1/c]
* provided that 0 is not in K
* Input:
*  L, K = real intervals
**/
intrinsic invDiv(L::SeqEnum[FldReElt], K::SeqEnum[FldReElt]) -> SeqEnum[FldReElt]
(For real intervals L, K (with 0 not in K), compute L/K.)
require (#L eq 2) and (#K eq 2):
"Both L, K must contain exactly two real numbers";
a, b := Explode(L);
c, d := Explode(K);
require (a le b) and (c le d): "Check if a <= b and c <= d"
if (c le 0) and (0 le d) then
   error "K must not contain 0";
end if;
return invMul(L, [1/d, 1/c]);
end intrinsic;
Bibliography


