CHAPTER I

INTRODUCTION

Introduction to the First (1992) Edition

This book is in three sections. First, we describe in detail an algorithm based on modular symbols for computing modular elliptic curves: that is, one-dimensional factors of the Jacobian of the modular curve $X_0(N)$, which are attached to certain cusp forms for the congruence subgroup $\Gamma_0(N)$. In the second section, various algorithms for studying the arithmetic of elliptic curves (defined over the rationals) are described. These are for the most part not new, but they have not all appeared in book form, and it seemed appropriate to include them here. Lastly, we report on the results obtained when the modular symbols algorithm was carried out for all $N \leq 1000$. In a comprehensive set of tables we give details of the curves found, together with all isogenous curves (5113 curves in all, in 2463 isogeny classes[1]). Specifically, we give for each curve the rank and generators for the points of infinite order, the number of torsion points, the regulator, the traces of Frobenius for primes less than 100, and the leading coefficient of the L-series at $s = 1$; we also give the reduction data (Kodaira symbols, and local constants) for all primes of bad reduction, and information about isogenies.

For $N \leq 200$ these curves can be found in the well-known tables usually referred to as "Antwerp IV" [2], as computed by Tingley [67], who in turn extended earlier tables of curves found by systematic search; our calculations agree with that list in all 281 cases. For values of N in the range $200 < N \leq 320$ Tingley computed the modular curves attached to newforms for $\Gamma_0(N)$ only when there was no known curve of conductor N corresponding to the newform: these appear in his thesis [67] but are unpublished. As in [2], the curves E we list for each N have the following properties.

(1) They have conductor N, as determined by Tate's algorithm [65].
(2) The coefficients given are those of a global minimal model for E, and these coefficients (or, more precisely, the c_4 and c_6 invariants) agree with the numerical values obtained from the modular calculation to several decimal places: in most cases, depending on the accuracy obtained—see below—differing by no more than 10^{-30}.
(3) Their traces of Frobenius agree with those of the modular curves for all primes $p < 1000$.

We have also investigated, for each curve, certain numbers related to the Birch–Swinnerton-Dyer conjecture. Let $f(z)$ be a newform for $\Gamma_0(N)$ with rational Fourier coefficients, and E the elliptic curve defined over $\mathbb{Q}$ attached to f. The value of $L(f, 1)$ is a rational multiple of a period of f, and may be computed easily using modular symbols (see [37] and Section 2.8 below). We have computed this rational number in each case, and find that it is always consistent with the Birch–Swinnerton-Dyer conjecture for E. More specifically, let $\Omega_0(f)$ be the least positive real period of F and $\Omega(f) = 2\Omega_0(f)$ or $\Omega_0(f)$ according as the period lattice of f is or is not rectangular. Then we find that $L(f, 1)/\Omega(f) = 0$ if (and only if) the Mordell-Weil group $E(\mathbb{Q})$

[1] In the first edition, these numbers were given as 5089 and 2447 respectively, as the curves of conductor 702 were inadvertently omitted.

has positive rank, and when $E(\mathbb{Q})$ is finite we find in each case that

$$L(f,1)/\Omega(f) = \prod_{p|N} c_p \cdot |E(\mathbb{Q})|^{-2} \cdot S$$

with $S \in \mathbb{N}$ (in fact $S = 1$ in all but four cases: $S = 4$ in three cases and $S = 9$ in one case). This is consistent with the Birch–Swinnerton-Dyer conjecture if the Tate–Shafarevich group Ш is finite of order S. (Here c_p is the local index $[E(\mathbb{Q}_p) : E_0(\mathbb{Q}_p)]$; see [58, p.362].) When $L(f,1) = 0$, we compute the sign w of the functional equation for $L(f,s)$, and verify that $w = +1$ if and only if the curve has even rank. More precisely, we also compute the value of $L^{(r)}(f,1)$, where r is the rank, the regulator R, and the quotient

$$S = \frac{L^{(r)}(f,1)}{r!\,\Omega(f)} \Bigg/ \frac{(\prod c_p)\,R}{|E(\mathbb{Q})_{\mathrm{tors}}|^2}\ .$$

In all but the four cases mentioned above we find that $S = 1$ to within the accuracy of the computation.

Our algorithm uses modular symbols to compute the 1-homology of $\Gamma_0(N)\backslash\mathcal{H}^*$ where $\mathcal{H}^*$ is the extended upper half-plane $\{z \in \mathbb{C} : \mathrm{Im}(z) > 0\} \cup \{\infty\} \cup \mathbb{Q}$. While similar in some respects to Tingley's original algorithm described in [67], it also uses ideas from [37] together with some new ideas which will be described in detail below. One important advantage of our method, compared with Tingley's, is that we do not need to consider explicitly the exact geometric shape of a fundamental region for the action of $\Gamma_0(N)$ on $\mathcal{H}^*$: this means that highly composite N can be dealt with in exactly the same way as, say, prime N. Of course, for prime N there are other methods, such as that of Mestre [43], which are probably faster in that case, though not apparently yielding the values of the "Birch–Swinnerton-Dyer numbers" $L(f,1)/\Omega(f)$. There is also a strong similarity between the algorithms described here and those developed by the author in his investigation of cusp forms of weight two over imaginary quadratic fields [12], [13], [15]. A variant of this algorithm has also been used successfully to study modular forms for $\Gamma_0(N)$ with quadratic character, thus answering some questions raised by Pinch (see [48] or [49]) concerning elliptic curves of everywhere good reduction over real quadratic fields. See [14] for details of this, and for a generalization to $\Gamma_1(N)$: one could find cusp forms of weight two with arbitrary character using this extension of the modular symbol method, though at present it has only been implemented for quadratic characters, as described in [14].

It is not our intention in this book to discuss the theory of modular forms in any detail, though we will summarize the facts that we need, and give references to suitable texts. The theoretical construction and properties of the modular elliptic curves will also be excluded, except for a brief summary. Likewise, we will assume that the reader has some knowledge of the theory of elliptic curves, such as can be obtained from one of the growing number of excellent books on the subject. Instead we will be concentrating on computational aspects, and hope thus to complement other, more theoretical, treatments.

In Chapter 2 we describe the various steps in the modular symbol algorithm in detail. At each step we give the theoretical foundations of the method used, with proofs or references to the literature. Included here are some remarks on our implementation of the algorithms, which might be useful to those wishing to write their own programs. At the end of this stage we have equations for the curves, together with certain other data for the associated cusp form: Hecke eigenvalues, sign of the functional equation, and the ratio $L(f,1)/\Omega(f)$.

Following Chapter 2, we give some worked examples to illustrate the various methods.

In Chapter 3 we describe the algorithms we used to study the elliptic curves we found using modular symbols, including the finding of all curves isogenous to those in the original list.

These algorithms are more generally applicable to arbitrary elliptic curves over $\mathbb{Q}$, although we do not consider questions which might arise with curves having bad reduction at very large primes. (For example, we do not consider how to factorize the discriminant in order to find the bad primes, as in all cases in the tables this is trivially achieved by trial division). Here we compute minimal equations, local reduction types, rank and torsion, generators for the Mordell–Weil group, the regulator, and traces of Frobenius. This includes all the information published in the earlier Antwerp IV tables. The final calculations, relating to the Birch–Swinnerton-Dyer conjecture, are also described here; these combine values obtained from the cusp forms (specifically, the leading coefficient of the expansion of the L-series at $s = 1$, and the real period) with the regulator and local factors obtained directly from the curves. Thus we can compute in each case the conjectural value S of the order of Ш, the Tate–Shafarevich group.

Finally, in Chapter 4 we discuss the results of the computations for $N \leq 1000$, and introduce the tables which follow.

All the computer programs used were written in Algol68 (amounting to over 10000 lines of code in all) and run on the ICL3980 computer at the South West Universities Regional Computing Centre at Bath, U.K.. The author would like to express his thanks to the staff of SWURCC for their friendly help and cooperation, and also to Richard Pinch for the use of his Algol68 multiple-length arithmetic package. At present, our programs are not easily portable, mainly because of the choice of Algol68 as programming language, which is not very generally available. However we are currently working on a new version of the programs, written in a standard version of the object-oriented language C++, which would be easily portable. The elliptic curve algorithms themselves are currently (1991) available more readily, in a number of computer packages.[2] In particular, the package `apecs`, written in Maple and available free via anonymous file transfer from Ian Connell of McGill University, will compute all the data we have included for each curve. (A slightly limited version of `apecs`, known as `upecs`, runs under UBASIC on MS-DOS machines). There are also elliptic curve functions available for Mathematica (Silverman's Elliptic Curve Calculator) and in the PARI/GP package. These packages are all in the process of rapid development.

An earlier version of Chapter 2 of this book, with the tables, has been fairly widely circulated, and several people have pointed out errors which somehow crept in to the original tables. We have made every effort to eliminate typographical errors in the tables, which were typeset directly from data files produced by the programs which did the calculations. Where possible, the data for each curve has been checked independently using other programs. Amongst those who have spotted earlier errors or have helped with checking, I would like to mention Richard Pinch, Harvey Rose, Ian Connell, Noam Elkies, and Wah Keung Chan; obviously there may still be some incorrect entries, but these remain solely my responsibility.

Introduction to the Second (1996) Edition

Since the first edition of this book appeared in 1992, some significant advances have been made in the algorithms described and in their implementation. The second edition contains an account of these advances, as well as correcting many errors and omissions in the original text and tables. We give here a summary of the more substantial changes to the text and tables.

Of course, the most significant theoretical advance of the last four years is the proof by Wiles, Taylor–Wiles and others of most cases of the Shimura–Taniyama–Weil conjecture, which almost makes the word "modular" in the title of this book redundant. However, the only effect the new results have on this work are to guarantee that every elliptic curve defined over the rationals and of conductor less than 1000 is isomorphic to one of those in our Table 1.

[2] See the end of the introduction for more on obtaining these packages.

Chapter 2. Section 2.1 has been completely rewritten and expanded to give a much more coherent, self-contained, and (we hope) correct account of the theoretical background to the modular symbol method. The text here is based closely on some unpublished lecture notes of the author for a series of lectures he gave in Bordeaux in 1995 at the meeting "État de la Recherche en Algorithmique Arithmétique" organized by the Société Mathématique de France.

In Section 2.4, we give a self-contained treatment of the method of Heilbronn matrices for computing Hecke operators, similar to the treatment by Merel in [42], as this now forms part of our implementation.

In Section 2.10, we give a new method of computing periods of cusp forms, as described in [18], which is as efficient as the "indirect" method; this largely makes the indirect method redundant, but we still include it in Section 2.11. Also in Section 2.11, we include some tricks and shortcuts which we have developed as we pushed the computations to higher levels, which can greatly reduce the computation time needed to find equations for the curves of conductor N, at the expense of not necessarily knowing which is the so-called "strong Weil" curve in its isogeny class.

Section 2.14 has been rewritten to take into account the results of Edixhoven on the Manin constant (see [21]), which imply that the values of c_4 and c_6 which we compute for each curve are known *a priori* to be integral. This means that the values we compute are guaranteed to be correct, and eliminates the uncertainty previously existing as to whether the curves we obtain by rounding the computed values are the modular elliptic curves they are supposed to be.

Section 2.15 is entirely new: we show how to compute the degree of the modular parametrization map $\varphi\colon X_0(N) \to E$ for a modular elliptic curve of conductor N, using our version (see [17]) of a method of Zagier [69]. This method is easy to implement within the modular symbol framework, and we have added it to our programs, so that we now compute the degree automatically for each curve we find.

The appendix to Chapter 2, containing worked examples, now includes the Heilbronn matrix method, and also illustrates some of the tricks mentioned in Section 2.11.

Implementation changes. The implementations of all the algorithms described here have been completely rewritten in C++, to be easily portable. We use the GNU compiler `gcc` for this. For multiprecision arithmetic we use either the GNU package `libg++` or the package `LiDIA`. For solving the systems of linear equations giving the relations between M-symbols, we use sparse matrix routines which not only reduce memory requirements, but also speed up that part of the computations considerably. These routines were written by L. Figueiredo specifically for his work on imaginary quadratic fields (see [24]) which in turn built on the author's work in [12] and [13].

Chapter 3. In Section 3.1 we give simpler formulae for recovering the Weierstrass coefficients of a curve from the invariants c_4 and c_6; this enables us to simplify the Kraus–Laska–Connell algorithm slightly. In Section 3.4 we give a slightly improved formula for the global canonical height, and include this as a separate algorithm. Section 3.5 now contains references to other bounds between the naive and canonical heights, and other methods for the infinite descent step, but without details.

The main changes in this chapter are to Section 3.6 on two-descent algorithms. On the one hand, we give a better explanation of the theoretical basis for these algorithms, making the account more self-contained (though we do not include all proofs). We have also moved the discussion on testing homogeneous spaces for local and global solubility forward, as this is common to the two main algorithms (general two-descent and two-descent via 2-isogeny). On the other hand, several parts of the algorithm have been subject to major improvements over the last few years, thanks to collaboration with P. Serf, S. Siksek and N. P. Smart, and these

are now included. Notable here are the syzygy sieve in the search for quartics, the systematic use of group structure in the 2-isogeny case, and the use of quadratic sieving in searching for rational points on homogeneous spaces. We also simplify the test for equivalence of quartics and the process of recovering rational points on the curve from points on the homogeneous spaces. Many of these improvements are from the author's paper [20], which contains some proofs omitted here.

Implementation changes. As with the modular symbol algorithms, we have rewritten all the elliptic curve algorithms in C++. In the case of the program to find isogenies, which is very sensitive to the precision used, we have written an independent implementation in PARI/GP; using this we have a check on the isogeny computations which gave the isogenous curves listed in Table 1. (The standard precision version of this program, while much faster, does miss several of the isogenies, for reasons given in Section 3.8.)

Versions of our algorithms will shortly become generally available in two forms. First, the package `LiDIA` (a library of C++ classes for computational number theory, developed by the LiDIA group at the Universität des Saarlandes in Germany) will include them in a coming release. Secondly, the package Magma is also in the process of implementing the algorithms.

In addition to these packages and those mentioned in the original Introduction, we should also mention the package Simath, developed by H. G. Zimmer's research group in Saarbrücken, which also has a large collection of very efficient elliptic curve algorithms.

See the end of this Introduction for how to obtain more information on these packages.

Chapter 4 and Tables. The two main changes in the tables are to include all the data for $N = 702$ in Tables 1–4 and include the new Table 5 giving the degree of the modular parametrization for each strong Weil curve. The omission of level 702 in the first edition is hard to explain; in our original implementation and file structure, it was not possible to distinguish between a level which had run successfully, but with no rational newforms found, and a level which had not yet run. The original runs were done as batch jobs on a remote mainframe computer, with manual record-keeping to keep track of which levels had run successfully. Our current implementation is much more robust in this respect. We are grateful to Henri Cohen who first discovered this error on comparing our data with his own tables (of modular forms of varying weight and level, computed by him together with Skoruppa and Zagier). The omission was also noted by Jacques Basmaji of Essen, who recomputed Table 3 independently.

The new implementation finds the newforms at each level in a consistent order. In the original runs, the order in which the newforms were found changed as the program developed. Unfortunately, we did not recompute the earlier levels with the final version of the program before publishing the first edition of the tables, and the identifying letter for each newform given in the tables has now become standard. Hence our current implementation reorders the newforms during output to agree with the order as originally published (this is necessary for 147 levels in all, the largest being 450).

Also concerning the order and naming of the curves: the convention we normally use is that in each isogeny class the first curve is the strong Weil curve whose period lattice is exactly that of the corresponding newform for $\Gamma_0(N)$, such as 11A1 for example. In precisely one case, an error caused the first curve listed in class 990H to be not the strong Weil curve but a curve isogenous to it. The strong Weil curve in this class is in fact 990H3 and not 990H1. In the notation of Section 2.11, the correct values of l^+ and m^+ to obtain the strong Weil curve 990H3 are 13 and 8, but for some reason we had used the value $m^+ = 24$ which leads to the 3-isogenous curve 990H1.

In Table 1, the other corrections are: $N = 160$ has the Antwerp codes corrected, and 916B1 has a spurious indication of a non-existent 3-isogeny removed.

In Table 2, we include the generators for the curves of conductor 702 and positive rank, and

again correct the Antwerp code for curve 160A1. We also give the generator of 427C1 correctly as $(-3,1)$ rather than $(-3,0)$ as previously, and for 990H we give the generator $(-35,97)$ of the strong Weil curve 990H3 rather than a generator of 990H1 as before.

In Table 3, as well as inserting the data for $N = 702$, we correct the eigenvalues for $N = 100$ which had been given incorrectly.

In Table 4, we insert the data for $N = 702$ and also for 600E–600I which had been omitted by mistake. Moreover, for $N = 990$ we give the data for 990H3 instead of 990H1 as before, as this is the strong Weil curve (the only difference being that Ω has been multiplied by 3 and R divided by 3).

Extension of the Tables. Using our new implementation of the algorithms of Chapter 2, we have extended the computations of all modular elliptic curves up to conductor 5077 (chosen as the smallest conductor of a curve of rank 3). We have also computed in each case the other data tabulated here for conductors up to 1000. For reasons of space, we cannot print extended versions of the tables: as there are 17598 newforms (or isogeny classes) and a total of 31586 curves up to 5077, this would have made this book approximately six times as thick as it is at present!

Instead, the data for curves whose conductors lie in the range from 1001 to 5077 (and beyond, as they become available) may be obtained by anonymous file transfer from the author's web site at `http://www.maths.nott.ac.uk/personal/jec/ftp/data/INDEX.html`

Finally, many thanks to those who have told me of misprints and other errors in the First Edition, including J. Basmaji, G. Bailey, B. Brock, F. Calegari, J. W. S. Cassels, T. Kagawa, B. Kaskel, P. Serf, S. Siksek, and N. Smart. Apologies to any whose names have been omitted. Extra thanks are also due to Nigel Smart, who read a draft of Chapter 3 of the Second Edition, and made useful suggestions.

web and ftp sites

More information on the packages mentioned above, and in most cases the packages themselves, can be obtained from the following web and ftp sites. Apart from MAGMA they are all free.

`apecs` (for Maple):	`ftp://math.mcgill.ca/pub/apecs`
Elliptic Curve Calculator (for Mathematica):	`ftp://gauss.math.brown.edu/dist/EllipticCurve`
`LiDIA`:	`http://www-jb.cs.uni-sb.de/LiDIA`
MAGMA:	`http://www.maths.usyd.edu.au:8000/comp/magma`
`mwrank`:	`ftp://euclid.ex.ac.uk/pub/cremona/progs`
PARI/GP:	`ftp://megrez.math.u-bordeaux.fr/pub/pari`
SIMATH:	`http://emmy.math.uni-sb.de/~simath`
`upecs` (for UBASIC):	`ftp://math.mcgill.ca/pub/upecs`

Links to all of these can be found at the following place, which will be updated.

`http://www.maths.nott.ac.uk/personal/jec/packages.html`