On the Numerical Computation of Modular Forms

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
David Berghaus
aus
Köln

Bonn, November 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Hartmut Monien
2. Gutachter:	Prof. Dr. Albrecht Klemm

Tag der Promotion:	31.10.2023
Erscheinungsjahr:	2023

Acknowledgements

I would like to thank my advisor Prof. Dr. Hartmut Monien for giving me this interesting topic and the opportunity to be part of his research group. I appreciate his explanations of his method for computing genus zero Belyi maps, and really enjoyed our weekly group meetings and his anecdotes, which made me learn a lot about various fields.

I would also like to thank Dr. Danylo Radchenko, who was like my second advisor, for helping me a lot with the involved mathematics and for the fun that I had during our collaboration.

Furthermore, I would like to thank Prof. Dr. Albrecht Klemm for agreeing to be a second examiner for my thesis, and Prof. Dr. Angkana Rüland and Prof. Dr. Simon Stellmer for being part of my commitee. I would also like to thank Prof. Dr. John Voight for useful comments on my work. In addition, I would like to thank the LMFDB staff Dr. Sam Schiavone, Dr. David Roe, and Dr. Edgar Costa for their help in adding my data to the LMFDB database. Additionally, I would like to thank Prof. Dr. Fredrik Strömberg for his help in installing psage and Dr. Robert Stephen Jones for sharing his insights into the GSVD method.

I would also like to thank Dr. Oliver Freyermuth for helping me with all IT related problems and for teaching me a lot about computers, as well as my office mate Philipp Höllmer for the fun that we had and for helping me whenever I struggled with LaTeX, and for proofreading this thesis. Additionally, I would like to thank the BCTP staff for providing a friendly and productive atmosphere.

Last but not least, I would like to thank my parents Britta and Andreas and my sister Jule for all their support and help during challenging times.

List of Publications

This thesis is based on the following papers:

- D.B., Hartmut Monien and Danylo Radchenko,
  On the computation of modular forms on noncongruence subgroups,
  `arXiv:2207.13365 [math.NT]` (electronic preprint)
- D.B., Hartmut Monien and Danylo Radchenko,
  A Database of Modular Forms on Noncongruence Subgroups,
  `arXiv:2301.02135 [math.NT]` (electronic preprint)

Especially, parts of chapters 2, 3, 4, 5 and 7 have been used (up to minor improvements and corrections) in the above mentioned papers. The mentioned parts have been written by the author of this thesis.

Further papers co-authored by the author of the present thesis are:

- D.B., Bogdan Georgiev, Hartmut Monien and Danylo Radchenko,
  On Dirichlet eigenvalues of regular polygons,
  `arXiv:2103.01057 [math.NT]` (electronic preprint)
- D.B., Robert Stephen Jones, Hartmut Monien and Danylo Radchenko,
  Computation of Laplacian eigenvalues of two-dimensional shapes with dihedral symmetry,
  `arXiv:2210.13229 [math.NA]` (electronic preprint)

Contents

CHAPTER 1

Introduction

The deep connection between symmetries and the fundamental laws of our universe has been a subject of fascination for centuries. The topic of this thesis are modular symmetries, which arise in many areas of physics such as conformal field theories [1], string theory [2] and black hole theory [3]. Modular symmetries give rise to a special class of analytic functions called modular forms, which have been studied since the 19th century because of their connection and application in various branches of mathematics. Modular forms appear everywhere, even for the pendulum (see fig. 1.1), which is arguably one of the most classical examples of physics.

To elaborate on this surprising relationship, recall that the mathematical pendulum satisfies conservation of angular momentum

$$ml^2\ddot{\theta} + mlg\sin(\theta) = 0\,, \tag{1.1}$$

where $g \approx 9.81\mathrm{m\,s^{-2}}$ is the gravitational constant. To model the motion of the pendulum, we must therefore solve the second-order differential equation

$$\ddot{\theta} + \Omega^2\sin(\theta) = 0\,, \tag{1.2}$$

where $\Omega^2 = g/l$, with initial conditions $\theta(t_0) = \theta_0$ and $\dot{\theta}(t_0) = 0$. An exact solution for this has been derived by BPMBN [4], so we will briefly sketch its derivation. First, note that $\mathrm{d/dt}\,\dot{\theta}^2 = 2\ddot{\theta}\dot{\theta}$ and

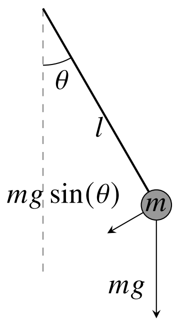

Figure 1.1: The mathematical pendulum of weight m and length l.

$\mathrm{d}/\mathrm{dt}\cos(\theta) = -\dot{\theta}\sin(\theta)$. By multiplying eq. (1.2) with $\dot{\theta}$, we thus obtain

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\dot{\theta}^2 - 2\Omega^2\cos(\theta)\right) = 0\,. \tag{1.3}$$

This quantity is therefore a constant, which we can determine by substituting the initial conditions, from which we obtain

$$\dot{\theta}^2 = 2\Omega^2\left(\cos(\theta) - \cos(\theta_0)\right)\,. \tag{1.4}$$

Using the identity $\cos(\theta) = 1 - 2\sin(\theta/2)^2$ we get

$$\dot{\theta}^2 = 4\Omega^2\left(\sin\left(\frac{\theta_0}{2}\right)^2 - \sin\left(\frac{\theta}{2}\right)^2\right)\,. \tag{1.5}$$

Defining $y := \sin(\theta/2)$ and $k := \sin(\theta_0/2)^2$ we compute

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{\mathrm{d}y}{\mathrm{d}\theta}\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{1}{2}\frac{\mathrm{d}\theta}{\mathrm{d}t}\cos\left(\frac{\theta}{2}\right), \tag{1.6}$$

from which follows that

$$\dot{\theta}^2 = \frac{4}{1-y^2}\dot{y}^2\,. \tag{1.7}$$

Plugging in eq. (1.5), we get

$$\dot{y}^2 = \Omega^2 k(1-y^2)\left(1 - \frac{y^2}{k}\right)\,. \tag{1.8}$$

Introducing $\tau := \Omega t$ and $z := y/\sqrt{k}$ we obtain

$$\left(\frac{\mathrm{d}z}{\mathrm{d}\tau}\right)^2 = (1-z^2)(1-kz^2)\,, \tag{1.9}$$

where $0 < k < 1$, with initial conditions $z(\tau_0) = 1$ and $(\mathrm{d}z/\mathrm{d}\tau)(\tau_0) = 0$. So we get

$$\mathrm{d}\tau = \pm\frac{\mathrm{d}z}{\sqrt{(1-z^2)(1-kz^2)}}\,, \tag{1.10}$$

from which it is clear that the solution must contain elliptic integrals. Indeed, as shown by BPMBN [4], we get that

$$\theta(t) = 2\arcsin\left\{\sin\left(\frac{\theta_0}{2}\right)\mathrm{sn}\left[K\left(\sin\left(\frac{\theta_0}{2}\right)^2\right) - \Omega t, \sin\left(\frac{\theta_0}{2}\right)^2\right]\right\}\,, \tag{1.11}$$

where

$$K(m) = \int_0^1 \frac{\mathrm{d}z}{\sqrt{(1-z^2)(1-mz^2)}}\,, \tag{1.12}$$

is the complete elliptic integral of the first kind. $\mathrm{sn}(u, m)$ is the elliptic sine, which can be written as a quotient of Jacobi theta functions [5, 22.2.E4]

$$\operatorname{sn}(u,m) = \frac{\theta_3(0,q)}{\theta_2(0,q)} \frac{\theta_1(\zeta,q)}{\theta_4(\zeta,q)}, \tag{1.13}$$

where $\zeta = \pi u/(2K(m))$ [5, 22.2.3] and $q = \exp\left(-\pi K'(m)/K(m)\right)$ [5, 22.2.1]. Note that K' is Legendre's complementary complete elliptic integral [5, 19.2.8_1]. The Jacobi theta functions θ_i depend on the theta function [6, 2.3.2]

$$\theta(z,\tau) = \sum_{n=-\infty}^{\infty} \exp\left(\pi i n^2 \tau + 2\pi i n z\right), \tag{1.14}$$

which satisfies $\theta(z,\tau+2) = \theta(z,\tau)$ and $\theta(z,-1/\tau) = \exp\left(\pi i z^2 \tau\right)\sqrt{\tau/i}\,\theta(z\tau,\tau)$ and is therefore modular in τ [6, Prop. 2.3.2]. We have thus shown that modular forms appear in the expressions that model the pendulum. By a similar argument, many other examples in classical mechanics are related to modular forms, such as the top [7, 2.3]. We refer to Brizard [7] for an overview.

The main goal of this thesis is to improve existing algorithms for the numerical computation of modular forms. We focus mainly on noncongruence modular forms, for which, in general, no feasible alternatives to numerical computations are known. The study of noncongruence modular forms was initiated by the work of Atkin and Swinnerton-Dyer [8] and has recently received wide attention after the proof of the unbounded denominator conjecture by Calegari, Dimitrov, and Tang [9]. Noncongruence modular forms also appear in physics, as shown in the work of Magureanu [10].

This thesis is organized as follows: In chapter 2 we introduce the necessary background and notation. In chapter 3 we describe a numerical method due to Hejhal [11] that can be used to compute noncongruence modular forms for arbitrary genera. Building on this, we extend the improvements of [12] in chapter 4 and show how the performance of Hejhal's method can be significantly improved, allowing the computation of examples that were previously inaccessible. Our main idea is based on the observation that the linear system of equations involved in Hejhal's method can be solved using cheap low-precision arithmetic, which can then be used as a preconditioner, making iterative solving methods converge quickly. We also discuss how the action of the matrices involved can be optimized using fast Fourier transforms and optimized dot product algorithms. In chapter 5 we use well-known and highly efficient Newton methods to compute genus zero Belyi maps [12–15], which rely on numerical estimates provided by the method of chapter 4 as initial values. We also discuss how full bases of modular forms can be constructed from the Belyi map, providing a rigorous and efficient alternative to Hejhal's method for the case of genus zero subgroups. In chapter 6 we show how the results of chapters 4 & 5 can be used to compute noncongruence Eisenstein series. We use recently developed algorithms for computing Petersson inner products [16] to construct the orthogonal complement of cusp forms in the space of modular forms, and demonstrate that this provides a very efficient approach to computing Eisenstein spaces. We apply this method to compute the first example (to our knowledge) of a non-trivial noncongruence Eisenstein series. Interestingly however, the vast majority of noncongruence Eisenstein series that we have computed seem to be non-algebraic. In chapter 7 we apply the algorithms developed in this thesis to create a database of modular forms on noncongruence subgroups, which we plan to add to the LMFDB [17] soon. We hope that this large number of examples will help to gain a deeper understanding of noncongruence modular forms. Other numerical computations of modular forms are briefly discussed in chapter 8: In section 8.1

we first focus on the numerical computation of Maass cusp forms on Hecke triangle groups G_n. Our motivation comes from recent numerical evidence that the eigenvalues of Maass cusp forms on Hecke triangle groups can be expanded as a series in $1/n$ [18]. Expansions of this form have previously been studied for Laplace eigenvalues of regular polygons [19–23]. We investigate an application of a method due to Betcke and Trefethen [24] to locate eigenvalues of Maass cusp forms and discuss the potential advantages and challenges of this approach compared to previous methods. In section 8.2 we show that the iterative methods for computing modular forms that have been developed in this work can be used to compute approximations of traces of real singular moduli to very high precision. Our numerical data indicates that these are non-algebraic. We conclude this thesis by highlighting areas for further research in chapter 9.

CHAPTER 2

Background on Modular Forms

This chapter gives an introduction to modular forms and recalls some results and notations that are needed for this work. Parts of this chapter were also used in the paper [25, Section 2].

Many excellent books have been written on the topic of modular forms (see for example [6, 26–28] to name a few). In this chapter, we will mostly follow the most recent one by Cohen and Strömberg [6]. Modular forms are complex analytic functions defined on the upper half plane

$$\mathcal{H} := \{\tau \in \mathbb{C} \,|\, \mathrm{Im}(\tau) > 0\} . \tag{2.1}$$

Let $\mathrm{SL}(2, \mathbb{Z})$ be the group of all integer 2×2 matrices with determinant 1. Let

$$\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}(2, \mathbb{Z}) . \tag{2.2}$$

Then γ acts on $\mathcal{H}$ via Möbius transformations

$$\gamma(\tau) := \frac{a\tau + b}{c\tau + d} . \tag{2.3}$$

Note that

$$\mathrm{Im}(\gamma(\tau)) = \frac{\mathrm{Im}(\tau)}{|c\tau + d|^2} > 0 , \tag{2.4}$$

which means that the elements $\gamma(\tau)$ are also on the upper half plane. It is also immediately obvious that γ and $-\gamma$ lead to the same action. It is therefore often more natural to work with the projective group

$$\mathrm{PSL}(2, \mathbb{Z}) \simeq \mathrm{SL}(2, \mathbb{Z})/\{\pm\mathbb{1}\} . \tag{2.5}$$

In the following we will denote $\mathrm{PSL}(2, \mathbb{Z})$ by Γ and refer to it as the modular group.

Definition 2.0.1 (Modular Form)**.** Let $f(\tau)$ be a holomorphic function from $\mathcal{H}$ to $\mathbb{C}$. Let $G \leqslant \Gamma$ be a finite index subgroup of Γ. Then we say that $f(\tau)$ is a modular form on G if it satisfies the functional equation

$$f(\gamma(\tau)) = (c\tau + d)^k f(\tau) , \tag{2.6}$$

for all $\gamma \in G$.

We refer to $k \in 2\mathbb{N}$ as the weight of $f(\tau)$ and to $(c\tau + d)^k$ as the automorphy factor. (More general definitions of modular forms including odd weights and multiplier systems exist but we will not consider them in this work.) Furthermore, we say that a modular form f is [6, p.5]:

1. *weakly holomorphic* if f is holomorphic in $\mathcal{H}$ but may have poles on the boundary $\partial\mathcal{H} := \mathbb{Q} \cup \{i\infty\}$.
2. *holomorphic* if f is holomorphic in $\overline{\mathcal{H}} := \mathcal{H} \cup \partial\mathcal{H}$.
3. a *cusp form* if f vanishes at $\partial\mathcal{H}$.

In addition, weakly holomorphic modular forms of weight zero are often called *modular functions*. It is also useful to introduce the *slash operator* [6, Definition 5.1.2]

$$(f|_k\gamma)(\tau) := (c\tau + d)^{-k} f(\gamma(\tau))\,, \tag{2.7}$$

which defines a right action of Γ on the space of functions

$$f|_k\gamma_1|_k\gamma_2 = f|_k\gamma_1\gamma_2\,. \tag{2.8}$$

2.1 Fundamental Domains

We define a fundamental domain of a group $G \leqslant \Gamma$ as follows:

Definition 2.1.1 (Fundamental Domain [6, Definition 4.3.1])**.** A closed set $\mathcal{F}(G) \subset \overline{\mathcal{H}}$ is called a *fundamental domain* if

1. For every point $\tau \in \overline{\mathcal{H}}$ there is a $\gamma \in G$ such that $\gamma(\tau) \in \mathcal{F}(G)$.
2. If for any points τ and $\tau' := \gamma(\tau)$ we have $\tau \neq \tau'$ then $\tau, \tau' \in \partial\mathcal{F}(G)$.

Note that Γ can be generated by the elements

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\,. \tag{2.9}$$

The matrix S therefore corresponds to the action $\tau \to -1/\tau$, which can be seen as an inversion, while T corresponds to the action $\tau \to \tau + 1$, a translation. We also have the relations

$$S^2 = \mathbb{1} \quad \text{and} \quad (ST)^3 = \mathbb{1}\,. \tag{2.10}$$

A fundamental domain for the modular group is therefore given by the set

$$\mathcal{F}(\Gamma) = \{\tau \in \overline{\mathcal{H}}\,,\ |\tau| \geq 1 \text{ and } |\mathrm{Re}(\tau)| \leq 1/2\} \cup \{i\infty\}\,. \tag{2.11}$$

The fundamental domain $\mathcal{F}(\Gamma)$ has three points that play a special role:

1. $i\infty$: a *cusp*
2. i: an *elliptic point of order 2* which has a non-trivial stabilizer S with $S^2 = \mathbb{1}$;

3. $\rho = \exp(2\pi i/3)$: an *elliptic point of order 3* which has a non-trivial stabilizer ST with $(ST)^3 = \mathbb{1}$ (alternatively we could also choose the point $-\bar{\rho} = \exp(\pi i/3)$).

Furthermore, since $\gamma(i\infty) = a/c$, we can see that the cusps are located at $\mathbb{P}^1(\mathbb{Q}) = \{i\infty\} \cup \mathbb{Q}$. For a finite index subgroup $G \leqslant \Gamma$ of index μ, a fundamental domain for $G \setminus \mathcal{H}$ is given by

$$\mathcal{F}(G) = \cup_{i=1}^{\mu} \gamma_i \mathcal{F}(\Gamma)\,, \tag{2.12}$$

where γ_i are right coset representatives of $G \setminus \Gamma$. The suitably defined quotient $G \setminus \overline{\mathcal{H}}$ (see, e.g., [6, Theorem 4.4.3]) is a Riemann surface whose genus can be computed by using the formula [6, Proposition 5.6.17]

$$g = 1 + \frac{\mu}{12} - \frac{n(e_2)}{4} - \frac{n(e_3)}{3} - \frac{n(c)}{2}\,, \tag{2.13}$$

where $n(e_2), n(e_3)$ denote the number of inequivalent elliptic points of order two and three, respectively, and $n(c)$ denotes the number of cusp representatives.

Definition 2.1.2 (Signature)**.** We define the *signature* of $G \leqslant \Gamma$ to be the tuple $(\mu, g, n(c), n(e_2), n(e_3))$. (Note that a signature does not uniquely specify G.)

We call the maps $A_j \in \mathrm{PSL}(2,\mathbb{Z})$, which map $i\infty$ to the cusp p_j on the real line

$$A_j(i\infty) = p_j\,, \tag{2.14}$$

and satisfy

$$A_j^{-1} S_j = T^N\,, \tag{2.15}$$

the *cusp normalizers*, where S_j is the generator of the stabilizer of p_j (we use the notation of Strömberg [29], some authors use the reverse notation) and N denotes the cusp width at infinity.

Example 2.1.1 ($\mathbf{\Gamma_0(5)}$). Consider the group $\Gamma_0(5)$ (defined as in eq. (2.21)) with signature $(6,0,2,2,0)$. A set of right coset representatives can be chosen to be

$$\left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -2 & -1 \\ 1 & 0 \end{pmatrix} \right\}, \tag{2.16}$$

which can be expressed as words in S and T by

$$\left\{ \mathbb{1}\,, T\,, T^2\,, T^{-1}\,, T^{-2}\,, T^{-2}S \right\}. \tag{2.17}$$

A fundamental domain can therefore be chosen as shown in fig. 2.1. We can see that this group has two cusps: One of width 5 at $i\infty$ and one of width 1 at -2. The cusp normalizer for the cusp at -2 is given by

$$\begin{pmatrix} -2 & -1 \\ 1 & 0 \end{pmatrix}. \tag{2.18}$$

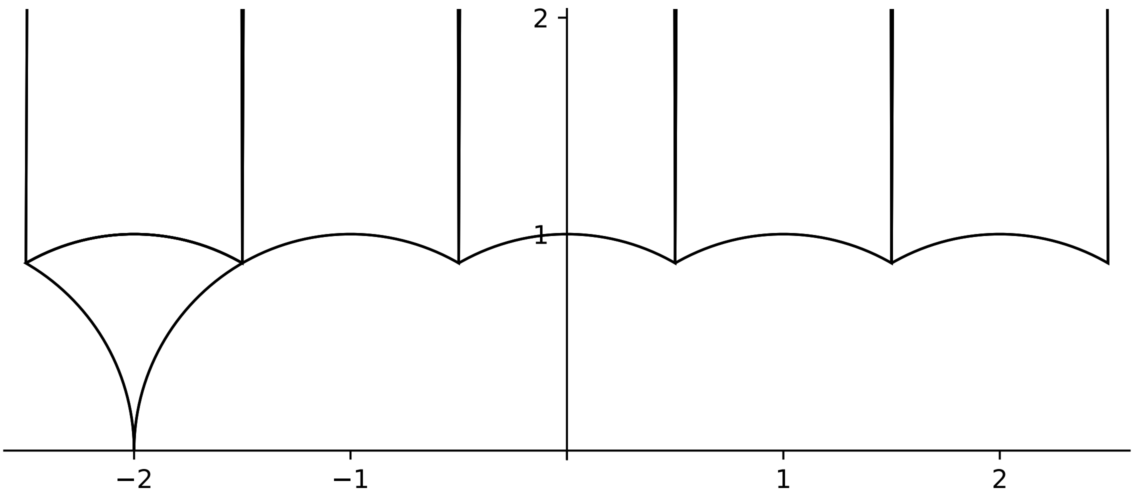

Figure 2.1: A fundamental domain for $\Gamma_0(5)$.

2.2 Subgroups of the Modular Group

2.2.1 Congruence Subgroups

Let N be a positive integer. Then we call

$$\Gamma(N) := \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \pmod{N} \quad \text{and} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma \right\}, \tag{2.19}$$

the *principal congruence subgroup of level N*. The index of $\Gamma(N)$ is given by [6, Corollary 6.2.13]

$$[\Gamma : \Gamma(N)] = \frac{1}{2} N^3 \prod_{p|N} \left(1 - \frac{1}{p^2}\right), \tag{2.20}$$

and is therefore finite.

Definition 2.2.1 (Congruence Subgroup)**.** A subgroup $G \leqslant \Gamma$ is a *congruence subgroup* of level N if and only if it contains $\Gamma(N)$ for some $N \in \mathbb{Z}^+$ (i.e., if $\Gamma(N) \leqslant G$).

Important congruence subgroups are

$$\Gamma_0(N) := \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} * & * \\ 0 & * \end{pmatrix} \pmod{N} \quad \text{and} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma \right\}, \tag{2.21}$$

and

$$\Gamma_1(N) := \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix} \pmod{N} \quad \text{and} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma \right\}, \tag{2.22}$$

that satisfy

$$\Gamma(N) \leqslant \Gamma_1(N) \leqslant \Gamma_0(N) \leqslant \Gamma \,. \tag{2.23}$$

Subgroups that are not congruence are called *noncongruence subgroups*. It has been proved by Stothers [30] that noncongruence subgroups are much more numerous than congruence subgroups (in the sense that the proportion of the latter among all subgroups of index n goes to 0 as $n \to \infty$). An algorithm to test whether a given group G is congruence or not has been given by Hsu [31].

2.2.2 Subgroups and Permutations

A useful tool for studying subgroups $G \leqslant \Gamma$ is the interpretation of the action of G on the cosets of $G \setminus \Gamma$ as an action of the permutation group S_μ. This theory was developed by Millington [32] and its usefulness in performing computations on subgroups of Γ was first demonstrated by Atkin-Swinnerton-Dyer [8].

Definition 2.2.2 (Legitimate Pair)**.** A pair (σ_S, σ_R) with $\sigma_S, \sigma_R \in S_\mu$ is *legitimate* if $\sigma_S^2 = \sigma_R^3 = \mathbb{1}$ and if the group Σ that is generated by σ_S and σ_R is transitive [32].

Definition 2.2.3 (Equivalence Modulo 1)**.** Two legitimate pairs (σ_S, σ_R) and (σ'_S, σ'_R) are said to be *equivalent (modulo 1)* if there exists a $\sigma \in S_\mu$ such that $(\sigma^{-1}\sigma'_S\sigma, \sigma^{-1}\sigma'_R\sigma) = (\sigma_S, \sigma_R)$ and $\sigma(1) = 1$ (i.e., that σ fixes 1) [32].

Theorem 2.2.1 (Millington)**.** There is a one-to-one correspondence between subgroups G of index μ in Γ and equivalence classes modulo 1 of legitimate pairs (σ_S, σ_R). Furthermore, $n(e_2)$ and $n(e_3)$ are given by the number of fixed elements of σ_S and σ_R, respectively, and $n(c)$ corresponds to the number of elements that are fixed by $\sigma_T = \sigma_S\sigma_R$. In addition, the cycle structure of σ_T reflects the cusp widths of G.

Proof. See [32, Theorem 2]. □

Millington's theorem thus gives rise to a map

$$\phi : \Gamma \to S_\mu \,, \tag{2.24}$$

which satisfies

$$\phi(x \cdot y) = \phi(x) \cdot \phi(y) \,, \tag{2.25}$$

and is therefore a homomorphism. Note that the set of coset representatives γ_i, $i = 1, ..., \mu$ of G satisfies $\phi(\gamma_i)(1) = i$ (see [29] for more details).

Millington's theorem also provides a method for listing all subgroups of a given index by filtering legitimate pairs into equivalence classes modulo 1. This algorithm has been used by Strömberg [29] to compute representatives of all subgroups in Γ with $\mu \leq 17$ up to relabeling (or in other words, conjugation in Γ). Strömberg has released this data in [33].

Example 2.2.1 ($\boldsymbol{\Gamma_0(5)}$ using Millington's theorem). Let us reconsider Example 2.1.1, this time by using Millington's theorem. As a legitimate pair for $\Gamma_0(5)$ we can choose $\sigma_S = (1)(2)(3\,4)(5\,6)$ and $\sigma_R = (1\,2\,3)(4\,5\,6)$ (this corresponds to the first subgroup of signature $(6, 0, 2, 2, 0)$ in Strömberg's database [29]). From this we get that $\sigma_T = \sigma_S\sigma_R = (1\,2\,3\,5\,4)(6)$. We can therefore label each coset as highlighted in fig. 2.2. In addition, we can see from the signature and from σ_R that $\Gamma_0(5)$ has no elliptic points of order three. The two elliptic points of order two are located at $\gamma_1(i)$ and $\gamma_2(i)$ where γ_j is the coset representative of label j, since 1 and 2 are fixed by σ_S.

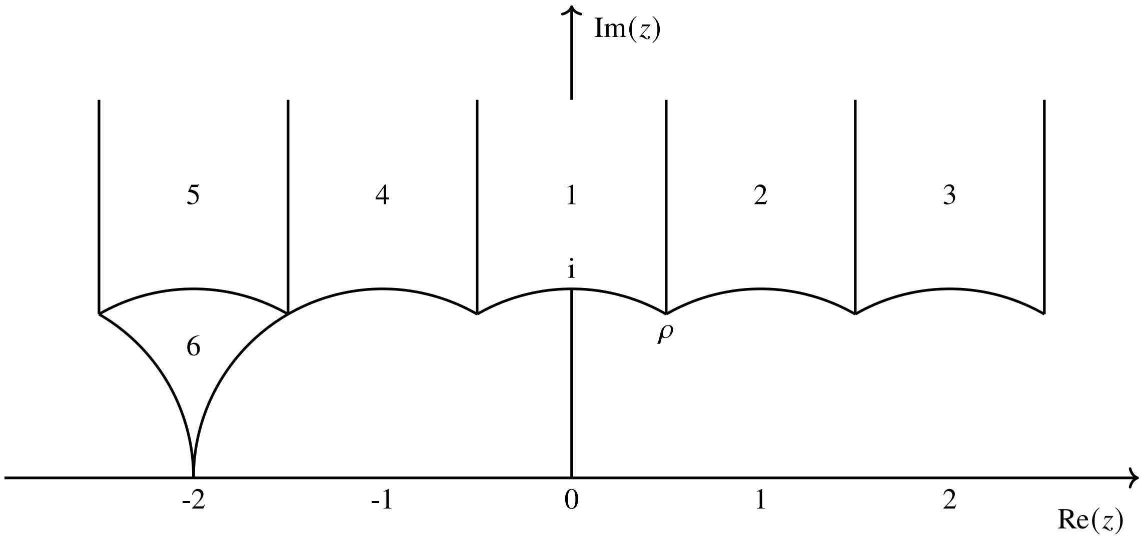

Figure 2.2: A fundamental domain for $\Gamma_0(5)$ corresponding to the legitimate pair $\sigma_S = (1)(2)(3\,4)(5\,6)$ and $\sigma_R = (1\,2\,3)(4\,5\,6)$. This figure has been taken from [25].

2.3 Fourier Expansions of Modular Forms

We have seen in the previous sections that modular forms are functions on the upper half plane that satisfy certain functional equations. We also saw that the cusp widths are always finite and that the modular forms are periodic with respect to these cusp widths. Modular forms can therefore be expanded as Fourier series in the variable

$$q_N := \exp(2\pi i\tau/N) = \exp(2\pi i(x+iy)/N) = \exp(2\pi ix/N)\exp(-2\pi y/N)\,, \tag{2.26}$$

where N is the cusp width and $\tau = x + iy \in \mathcal{H}$ (we will also often use the convention $q := q_1$). It is important to note that q_N decays exponentially as $y \to \infty$. If f is a modular form and the cusp width at $i\infty$ is given by N, then we can write

$$f(\tau) = \sum_{n=-\infty}^{\infty} a_n q_N^n\,, \tag{2.27}$$

with $a_i \in \mathbb{C}$. For congruence subgroups, it is known that there exist bases of modular forms whose Fourier coefficients are defined over $\mathbb{Q}$ or cyclotomic fields.

Example 2.3.1 (Discriminant Modular Form)**.** An example of a Fourier expansion of a modular form on the modular group Γ (which is obviously congruence) is the cusp form of weight 12 that is also called the *discriminant modular form* $\Delta(\tau)$ or *Ramanujan tau function*. Its Fourier expansion is given by [6, Corollary 5.8.2]

$$\Delta(\tau) = q\prod_{n\geq 1}(1-q^n)^{24} = q - 24q^2 + 252q^3 - 1472q^4 + 4830q^5 - 6048q^6 + \dots\,, \tag{2.28}$$

and its coefficients can thus be defined over $\mathbb{Z}$.

Modular forms on congruence subgroups are well known and have been studied extensively. Their Fourier coefficients can be computed using fast and explicit methods, see for example the book by Stein [34], and computer algebra systems such as Sage [35] and PARI [36] offer implementations of these algorithms. We also note that LMDFB [17] hosts large amounts of Fourier coefficients of congruence modular forms.

For noncongruence subgroups the Fourier coefficients are defined over $\bar{\mathbb{Q}}$ and have the form (see for example Atkin-Swinnerton-Dyer [8])

$$a_n = u^m b_n \,, \tag{2.29}$$

where b_n and u^N are defined over a number field K which is generated over $\mathbb{Q}$ by an algebraic number v (i.e., $K = \mathbb{Q}(v)$).

Definition 2.3.1 (Valuation of a modular form)**.** We define the valuation of a modular form as the index of the first Fourier coefficient that is nonzero.

Remark 2.3.1. By using the valuation of a modular form, many properties immediately follow from its q_N expansion. For example, a modular form can only be holomorphic if its Fourier expansion starts at $n \geq 0$, because negative values of n would lead to poles at $i\infty$ due to the decay of q_N. By the same argument, cusp forms must have Fourier expansions starting at $n > 0$.

Theorem 2.3.2 (Coefficient Growth of Cusp forms)**.** Let f be a cusp form of weight k. Then the Fourier coefficients of f grow like $O(n^{k/2})$.

Proof. First proved by Hecke, but see for example Serre [37, Theorem 5]. □

Theorem 2.3.3 (Coefficient Growth of Holomorphic Modular Forms)**.** Let f be a holomorphic modular form of weight k. Then the Fourier coefficients of f grow like $O(n^{k-1})$.

Proof. See Serre [37, p. 94]. □

2.4 Eisenstein Series

Let A_j be the cusp normalizer of cusp p_j. For $k > 2$, we call the series

$$E_{k,p_j}(\tau) := \sum_{\gamma \in G_{p_j} \backslash G} j(A_{p_j}^{-1}\gamma, \tau)^{-k} \,, \tag{2.30}$$

where G_{p_j} is the stabilizer of the cusp p_j and $j(\gamma, \tau) = c\tau + d$, for all c, d that are elements of the bottom row of γ, the (holomorphic) Eisenstein series on G. Note that E_k is a modular form of weight k. Eisenstein series also admit a Fourier expansion. For example for the Eisenstein series on Γ we get the Fourier expansions [6, Proposition 5.2.7]

$$E_k(\tau) = 1 - \frac{2k}{B_k} \sum_{n \geq 1} \sigma_{k-1}(n) q^n \,, \tag{2.31}$$

where B_k denotes the Bernoulli numbers and $\sigma_k(n)$ is the divisor sum function.

2.5 Spaces of Modular Forms

We denote the space of holomorphic modular forms of (even) weight k on G by $M_k(G)$ and similarly define $S_k(G)$ as the space of cusp forms and $E_k(G)$ as the space of Eisenstein series. The dimensions of these spaces can be computed from their signatures [6, Theorem 5.6.18]

$$\dim(M_k(G)) = (k-1)(g-1) + \left\lfloor \frac{k}{4} \right\rfloor n(e_2) + \left\lfloor \frac{k}{3} \right\rfloor n(e_3) + \left\lfloor \frac{k}{2} \right\rfloor n(c)\,, \tag{2.32}$$

$$\dim(S_k(G)) = \dim(M_k(G)) - n(c) + \delta_{k,2}\,, \tag{2.33}$$

$$\dim(E_k(G)) = \dim(M_k(G)) - \dim(S_k(G))\,. \tag{2.34}$$

Definition 2.5.1 (Victor Miller Normalization)**.** Let $d = \dim(M_k(G))$ and let $f_i \in M_k(G)$ for $i = 0, 1, ..., d-1$ form a basis of $M_k(G)$ at infinity. Then we say that f_i are in a *Victor Miller normalization* if $a_n(f_i) = \delta_{n,i}$ where $a_n(f_i)$ denotes the n-th Fourier coefficient of f_i and $n = 0, 1, ..., d-1$. Analogously for cusp forms, if $d = \dim(S_k(G))$ and $f_i \in S_k(G)$ for $i = 0, 1, ..., d-1$ form a basis, then f_i are in a Victor Miller normalization if $a_n(f_i) = \delta_{n,i+1}$ for $n = 1, 2, ..., d$. For reference, see [34, Lemma 2.19] although we extend the definition from coefficients over $\mathbb{Z}$ to general algebraic numbers.

Example 2.5.1. Consider the case $G = \Gamma_0(3)$ with signature $(4, 0, 2, 0, 1)$. Then $\dim(M_{10}(\Gamma_0(3))) = 4$ and

$$\begin{aligned}
f_0 &= 1 + 3960q^4 + 28512q^5 + 11880q^6 + \ldots \\
f_1 &= q - 269q^4 - 4374q^5 - 13122q^6 + \ldots \\
f_2 &= q^2 - 63q^4 - 328q^5 - 1701q^6 + \ldots \\
f_3 &= q^3 + 15q^4 + 108q^5 + 558q^6 + \ldots
\end{aligned}$$

denotes the basis for $M_{10}(\Gamma_0(3))$ in Victor Miller form. Similarly, for the case $S_{10}(\Gamma_0(3))$ which has $\dim(S_{10}(\Gamma_0(3))) = 2$,

$$\begin{aligned}
f_0 &= q + 27q^3 + 136q^4 - 1458q^5 + 1944q^6 + \ldots \\
f_1 &= q^2 + 3q^3 - 18q^4 - 4q^5 - 27q^6 + \ldots
\end{aligned}$$

is the basis for $S_{10}(\Gamma_0(3))$ in Victor Miller form.

2.6 The Petersson Product

The covolume of the fundamental domain of the modular group is given by [6, Proposition 4.5.2]

$$\mathrm{covol}(\mathcal{F}(\Gamma)) = \int_{\Gamma\backslash\mathcal{H}} d\tau = \int\int_{\mathcal{F}(\Gamma)} \frac{dx\,dy}{y^2} = \int_{-1/2}^{1/2}\int_{\sqrt{1-x^2}}^{\infty} \frac{dy}{y^2}dx = \frac{\pi}{3}\,, \tag{2.35}$$

from which follows that for a subgroup G of index $[\Gamma : G]$ we have

$$\mathrm{covol}(\mathcal{F}(G)) = [\Gamma : G] \cdot \frac{\pi}{3} . \tag{2.36}$$

For two modular forms $f, g \in M_k(G)$, we define the *Petersson scalar product* by [6, Definition 8.1.1]

$$\langle f, g \rangle_G := \frac{1}{[\Gamma : G]} \int_{G\backslash\mathcal{H}} f(\tau)\overline{g(\tau)} y^k d\tau . \tag{2.37}$$

It is immediately apparent that the Petersson product only converges if at least one of f and g is a cusp form. Note also that

$$\langle g, f \rangle_G = \overline{\langle f, g \rangle_G} . \tag{2.38}$$

Theorem 2.6.1 (Orthogonal Spaces of Petersson Products)**.** $S_k(G)$ and $E_k(G)$ are orthogonal complements with respect to the Petersson scalar product. Furthermore, the space of holomorphic modular forms can be decomposed by an orthogonal direct sum into

$$M_k(G) = E_k(G) \oplus S_k(G) . \tag{2.39}$$

It follows that a modular form $f \in M_k(G)$ is orthogonal to $S_k(G)$ if and only if $f \in E_k(G)$.

Proof. See [6, Corollary 8.2.6] □

2.7 Products of Modular Forms

Theorem 2.7.1. Let f be a modular form of weight k_f on G and g be a modular form of weight k_g on G. Then the following statements hold:

1. The product $f \cdot g$ is a modular form of weight $k_f + k_g$ on G.
2. The quotient f/g (where $g \neq 0$) is a modular form of weight $k_f - k_g$ on G.
3. If f and g are cusp forms then $f \cdot g$ is a cusp form on G.
4. If f and g are non-cuspidal holomorphic modular forms, then $f \cdot g$ is a non-cuspidal holomorphic modular form on G.
5. If f is a cusp form and g is a non-cuspidal holomorphic modular form, then $f \cdot g$ is a cusp form on G.

Proof. Immediate, see for example Miyake [28, Section 2.1]. □

It follows from this result that modular forms of lower weight can be used to construct modular forms of higher weight.

2.8 Hauptmoduls

Subgroups $G \leqslant \Gamma$ of genus zero have a special type of modular function called the *Hauptmodul* (which we denote by j_G).

Definition 2.8.1 (Hauptmodul)**.** Let G be a subgroup of genus zero. Then a Hauptmodul is any isomorphism

$$j_G : G\backslash\overline{\mathcal{H}} \to P^1(\mathbb{C})\,. \tag{2.40}$$

Since the modular group Γ is of genus zero, it has a Hauptmodul which is called *Klein invariant* or *modular j-invariant*. Its Fourier expansion is given by

$$j(\tau) = \frac{E_4^3}{\Delta(\tau)} = q^{-1} + 744 + 196884q + 21493760q^2 + 864299970q^3 + \dots\,. \tag{2.41}$$

and its values at the elliptic points are

$$j(i) = 1728 \quad \text{and} \quad j(\rho) = 0\,. \tag{2.42}$$

Since j has a negative valuation, we can also see that it has a pole of order 1 at infinity. Note that the Hauptmodul can be chosen uniquely up to a constant term. The choice of 744 for the constant term has historical reasons. For groups $G \neq \Gamma$ we will instead set the constant term to zero and use the normalization

$$j_G(\tau) = q_N^{-1} + 0 + \sum_{n=1} a_n q_N^n\,, \tag{2.43}$$

which uniquely specifies j_G [8].

Theorem 2.8.1. Let f be a meromorphic function on $\mathcal{H}$. The following statements are equivalent:

1. f is a modular function on Γ of weight 0.
2. f is a quotient of two modular forms of equal weight.
3. f is a rational function in j.

Proof. See [6, Theorem 5.7.3] □

Theorem 2.8.2. Every modular function on G that is holomorphic outside $i\infty$ can be written as a polynomial $P(j_G(\tau))$.

Proof. See Cox [38, Lemma 11.10 (ii)] for the case of $G = \Gamma$ (the proof for general G is equivalent). □

2.9 Derivatives of Modular Forms

The derivative of a modular form is not a modular form for $k > 0$ (instead it is a so-called *quasi-modular form* of weight k+2 and depth less than or equal to 1) [6, p. 152] because

$$\frac{\partial}{\partial\tau}(f|_k\gamma)(\tau) = (c\tau+d)^{-k-2}\frac{\partial}{\partial\tau}(f)(\gamma(\tau)) - kc(c\tau+d)^{-k-1}f(\gamma(\tau))\,. \tag{2.44}$$

However, if f is a modular function (i.e., a weakly modular form of weight zero), then $f'(\tau)$ is a weakly modular form of weight 2, where we define

$$f'(\tau) := \frac{1}{2\pi i}\frac{\partial}{\partial\tau} f(\tau) \,. \tag{2.45}$$

(The constant factor $1/(2\pi i)$ is useful because the Fourier coefficients of the derivative remain in the same number field.) This means that, for example for genus zero subgroups, the derivative of the Hauptmodul $j'_G(\tau)$ is a non-holomorphic modular form on G of weight 2.

2.10 Elliptic Curves

Let $\Lambda = \mathbb{Z}w_1 + \mathbb{Z}w_2$, where $w_1, w_2 \in \mathbb{C}$ are $\mathbb{R}$ linearly independent, be a complex lattice. We define the Weierstrass $\wp$-function by [6, Definition 2.1.3]

$$\wp(\tau, \Lambda) := \frac{1}{\tau^2} + \sum_{w\in\Lambda\backslash\{0\}} \left(\frac{1}{(\tau - w)^2} - \frac{1}{w^2} \right) , \tag{2.46}$$

which satisfies $\wp(\tau, \Lambda + w) = \wp(\tau, \Lambda)$ and is therefore an elliptic function. Moreover, $\wp$ satisfies the differential equation [6, Theorem 2.1.7]

$$(\wp')^2 = 4\wp^3 - g_2(\Lambda)\wp - g_3(\Lambda) \,, \tag{2.47}$$

with $g_2(\Lambda) := 60G_4(\Lambda)$ and $g_3(\Lambda) := 140G_6(\Lambda)$, where

$$G_k(\Lambda) := \sum_{w\in\Lambda\backslash\{0\}} w^{-k} \,, \tag{2.48}$$

is the Eisenstein series. Eq. (2.47) thus gives rise to an isomorphism from $\mathbb{C} \setminus \Lambda$ to projective algebraic curves (see [6, Proposition 2.2.1] for a formal proof). We can normalize each lattice to be of the form

$$\Lambda(\tau) = \mathbb{Z} \cdot 1 + \mathbb{Z} \cdot \tau \,. \tag{2.49}$$

Then $\Lambda(\tau) = \Lambda(\tau')$ if and only if $\tau' = \gamma(\tau)$ for some $\gamma \in \Gamma$. We can thus define an isomorphism class between elliptic curves by the j-invariant

$$j(\Lambda \cong E) = 1728 \frac{g_2(\Lambda)^3}{g_2(\Lambda)^3 - 27g_3(\Lambda)^2} \,. \tag{2.50}$$

Thus two elliptic curves E and E' are isomorphic if and only if $j(E) = j(E')$ (see for example [39]). However, the connection between modular forms and elliptic curves goes much deeper: By the modularity theorem, every elliptic curve defined over $\mathbb{Q}$ is related to a modular curve (i.e., a curve associated with a congruence subgroup $\Gamma_0(N)$) (see [26] for a detailed introduction). The smallest parameter N for which such a parameterization arises is called the *conductor* of E.

CHAPTER 3

Hejhal's Method

Parts of this chapter were also used in the paper [25, Section 3].

This chapter explains a method for the numerical computation of modular forms that is due to Hejhal [11] (based on an idea of Stark) and lists various applications and extensions by other authors. Hejhal developed this method for computing Maass cusp forms on Hecke triangle groups. Due to the generality of this method (in principle, the only requirements for this method are a converging expansion basis for the modular form and an automorphy condition), it has since then been adapted by many authors. For example, Selander and Strömbergsson [40] generalized the method for fundamental domains with multiple cusps to compute some examples of genus 2 coverings, and Strömberg used this method to compute Maass cusp forms for $\Gamma_0(N)$ and non-trivial multiplier systems [41] as well as Maass cusp forms for noncongruence subgroups [29]. Applications of Hejhal's method using arbitrary precision arithmetic were made by Booker, Strömbergsson, and Venkatesh [42], who computed the first ten Maass cusp forms of Γ to 1000 digits precision, Bruinier and Strömberg [43], who computed harmonic weak Maass cusp forms, and Voight and Willis [44] (see also the improved method in KMSV [12]) who computed Taylor expansions of modular forms.

3.1 The Basic Idea (Hejhal's Original Method)

Before discussing Hejhal's method in more detail, it is useful to start with its predecessor that has also been developed by Hejhal in [45] (we might want to call this method *Hejhal's original method*). Hejhal's original method usually has inferior conditioning but its concepts are easier to follow, making it a useful introduction. For simplicity, we will first illustrate this method for the modular group Γ, whose fundamental domain is given by eq. (2.11). The point inside $\mathcal{F}(\Gamma)$ with the smallest height (i.e., the smallest imaginary value) is given by ρ, whose height is $Y_0 = \sqrt{3}/2$. Now we choose a set of $2Q$ points τ_m that are equally spaced between $-1/2$ and $1/2$ along a horizontal line with height $Y < Y_0$

$$\tau_m = x_m + iY = \frac{1}{2Q}\left(m - Q + \frac{1}{2}\right) + iY, \quad 0 \leq m \leq 2Q-1, \quad Y < Y_0\,. \tag{3.1}$$

Remark 3.1.1. Throughout this work we have always chosen $Y = 0.8 \cdot Y_0$.

We will refer to this horizontal line as a *horocycle*. Note that since these points are located *below*

$\mathcal{F}(\Gamma)$, they are all *outside* $\mathcal{F}(\Gamma)$. Now for each point τ_m there exists a map $\gamma_m \in \Gamma$ such that

$$\tau_m^* = \gamma_m(\tau_m) \in \mathcal{F}(\Gamma). \tag{3.2}$$

We call the maps γ_m the *pullback* to the fundamental domain. For the case $G = \Gamma$, finding such a pullback map is easy, we just have to form words of the generators $S \to -1/\tau$ and $T \to \tau + 1$ depending on whether $|\tau| < 1$, $\mathrm{Re}(\tau) < -1/2$ or $\mathrm{Re}(\tau) > 1/2$ and form the matrix products. Then we expand the modular form in its basis functions (which in our case are given by powers of q) up to a finite order $M_0 := M(Y_0)$, so that our expansion converges inside $\mathcal{F}(\Gamma)$ up to the machine epsilon $\epsilon_{\text{machine}}$. M_0 can be guessed in advance by using the asymptotic growth conditions of the coefficients (see theorems 2.3.2 and 2.3.3). Although such a choice of M_0 works well in practice, it is non-rigorous and therefore there is no guarantee at this point that the result will be correct. This is one of the reasons why it is difficult to make Hejhal's method rigorous. To be a modular form, the expansion must now satisfy (at least numerically) the automorphism condition

$$f(\tau_m) \approx \sum_{n=0}^{M_0} a_n q(\tau_m)^n \stackrel{!}{=} (c_m \cdot \tau_m + d_m)^{-k} f(\tau_m^*) \approx (c_m \cdot \tau_m + d_m)^{-k} \sum_{n=0}^{M_0} a_n q(\tau_m^*)^n \,, \tag{3.3}$$

where $q(\tau) = \exp(2\pi i \tau)$ and c_m, d_m denote the lower entries of γ_m (we illustrate this method here for the example of holomorphic modular forms, but it can of course be applied analogously to cusp forms or Hauptmoduls). It is preferable to work with

$$F(\tau) = y^{k/2} f(\tau) \,, \tag{3.4}$$

where $y = \mathrm{Im}(\tau)$, because the function F transforms like

$$F(\tau_m) = \frac{|c_m \cdot \tau_m + d_m|^k}{(c_m \cdot \tau_m + d_m)^k} F(\tau_m^*) \,, \tag{3.5}$$

and its automorphy factor hence does not change the order of magnitude, which improves the numerical stability. This results in a linear system of equations

$$\begin{pmatrix} \Delta_{0,0} & \cdots & \Delta_{0,M_0} \\ \vdots & \ddots & \vdots \\ \Delta_{2Q-1,0} & \cdots & \Delta_{2Q-1,M_0} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{M_0} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} , \tag{3.6}$$

where

$$\Delta_{m,n} = q(\tau_m)^n - \frac{|c_m \cdot \tau_m + d_m|^k}{(c_m \cdot \tau_m + d_m)^k} q(\tau_m^*)^n \,. \tag{3.7}$$

Although it is in principle possible to solve this linear system of equations by computing the eigenspace or the singular value decomposition, it usually seems preferable to impose a Victor Miller normalization (see definition 2.5.1) and to subtract the column that is normalized to have a coefficient of value 1 and to place it on the right-hand side. The resulting linear system of equations can then be solved (for example with a least-squares fit) to obtain numerical approximations of the coefficients c_n. While Hejhal's original method is easy to implement, it has a major disadvantage in that it is usually ill-conditioned.

The matrix entries $\Delta_{m,n}$ are dominated by the left-hand side because $|q(\tau_m)| > |q(\tau_m^*)|$ and decay for larger n, while the rows are usually quite uniformly distributed. This makes it difficult to apply this method to larger examples or to compute higher order coefficients.

3.2 The Improved Automorphy Method

To overcome the ill-conditioning of his original method, Hejhal presented in [11] a new method based on an idea of Stark. This method is now usually referred to as *Hejhal's method*. The improved method uses the Fourier integral formula to obtain

$$a_n Y^{\frac{k}{2}} \exp(-2\pi nY) = \int_{-\frac{1}{2}}^{\frac{1}{2}} F(\tau) \exp(-2\pi inx) dx \,, \tag{3.8}$$

where Y is the height of the horocycle. Discretizing this integral to approximate it numerically gives

$$a_n Y^{\frac{k}{2}} \exp(-2\pi nY) \approx \frac{1}{2Q} \sum_{m=0}^{2Q-1} F(\tau_m) \exp\bigl(-2\pi inx_m\bigr) \,, \tag{3.9}$$

where $Q > M(Y)$ and τ_m are again given by eq. (3.1). Hejhal then incorporates the automorphy condition by replacing $F(\tau_m)$ with the corresponding pullback

$$\begin{aligned} a_n Y^{\frac{k}{2}} \exp(-2\pi nY) &\approx \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\frac{|c_m\tau_m + d_m|}{(c_m\tau_m + d_m)} \right)^k F(\tau_m^*) \exp\bigl(-2\pi inx_m\bigr) \,, &(3.10)\\ &= \sum_{l=0}^{M_0} a_l \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\frac{|c_m\tau_m + d_m|}{(c_m\tau_m + d_m)} \right)^k (y_m^*)^{\frac{k}{2}} \exp\bigl(2\pi i(l\tau_m^* - nx_m)\bigr) \,, &(3.11)\\ &:= \sum_{l=0}^{M_0} a_l V_{n,l} \,, &(3.12) \end{aligned}$$

where

$$V_{n,l} := \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\frac{|c_m\tau_m + d_m|}{(c_m\tau_m + d_m)} \right)^k (y_m^*)^{\frac{k}{2}} \exp\bigl(2\pi i(l\tau_m^* - nx_m)\bigr) \,. \tag{3.13}$$

Therefore

$$0 = \sum_{l=0}^{M_0} a_l \tilde{V}_{n,l} \,, \tag{3.14}$$

with

$$\tilde{V}_{n,l} := V_{n,l} - \delta_{n,l} Y^{\frac{k}{2}} \exp(-2\pi nY) \,, \tag{3.15}$$

which we can again solve by imposing a Victor Miller normalization. For example, for a one-dimensional space, this would amount to setting $a_0 = 1$ and solving for

$$\begin{pmatrix} \tilde{V}_{1,1} & \dots & \tilde{V}_{1,M_0} \\ \vdots & \ddots & \vdots \\ \tilde{V}_{M_0,1} & \dots & \tilde{V}_{M_0,M_0} \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ \vdots \\ a_{M_0} \end{pmatrix} = \begin{pmatrix} -\tilde{V}_{1,0} \\ \vdots \\ -\tilde{V}_{M_0,0} \end{pmatrix}. \tag{3.16}$$

(Note that we have also removed the first row of $\tilde{V}$ in order to keep the linear system of equations square.) The advantage of this approach over the previous section is that the largest entries of each column are now located on the diagonal. This can be seen in eq. (3.15): $V_{n,l}$ depends on the pullbacked points, which have a larger imaginary value (and hence smaller q-values) than the horocycle points located at height Y. For this reason

$$|Y^{\frac{k}{2}} \exp(-2\pi n Y)| > |V_{n,l}|\,, \tag{3.17}$$

and the largest entries of each column are therefore located on the diagonal. This means that the linear system of equations resulting from this improved method is much better conditioned. The precision of the coefficients depends on the diagonal term in eq. (3.15). We can therefore expect the l-th coefficient (where $1 \leq l \leq M_0$) to be correct to approximately

$$\mathrm{prec}(a_n) = D - \log_{10}^{+} \left| \frac{1}{Y^{k/2} \exp(-2\pi l Y)} \right|, \tag{3.18}$$

digits precision (this is analogous to Maass cusp forms, see [42]). The *precision loss* of higher order coefficients can thus be controlled by choosing a smaller value of Y (and consequently a larger value of Q). Once the coefficients a_l, $l = 0, \dots, M_0$ have been computed with reasonable accuracy, approximations of higher order coefficients with $l' > M_0$ can be obtained from them without additional linear solving by using [11]

$$a_{l'} = \frac{\sum_{l=0}^{M_0} a_l V_{n,l}}{Y^{\frac{k}{2}} \exp\left(-2\pi l' Y\right)}\,, \tag{3.19}$$

where Y is reduced for larger l' (we however mention this only for completeness and had no need to compute these higher order coefficients in this project).

Remark 3.2.1. To heuristically check the accuracy of the coefficients computed by Hejhal's method, one can repeat the computation with an independent choice of Y. This is especially important for Maass cusp forms, since if it is not clear a priori whether the computed solution corresponds to a *true* eigenvalue.

3.3 Hejhal's Method for Multiple Cusps

So far we have only considered the case $G = \Gamma$. The general case, including groups with multiple cusps, has been worked out by Selander and Strömbergsson [40] (see also Strömberg [29, 41]) and follows the same ideas but the resulting expressions are more tedious and the pullback maps more difficult to obtain. If G has multiple cusps then we must include the Fourier expansions at all cusps in order to obtain convergence in $\mathcal{F}(G)$. Let $j = 1, ..., n(c)$ denote the cusps of $G \leqslant \Gamma$.

Definition 3.3.1. (Width-absorbing cusp normalizer) Let A_j be the cusp normalizer of cusp j as defined in eq. (2.14). Let w_j be the width of cusp j. We define the *width absorbing cusp normalizer* of cusp j to be the map $\mathcal{N}_j \in \mathrm{PSL}(2,\mathbb{R})$ such that

$$\mathcal{N}_j(\tau) = A_j(w_j \cdot \tau)\,, \tag{3.20}$$

and therefore

$$\mathcal{N}_j = A_j \cdot \rho_j = A_j \cdot \begin{pmatrix} \sqrt{w_j} & 0 \\ 0 & 1/\sqrt{w_j} \end{pmatrix}, \tag{3.21}$$

because

$$\rho_j(\tau) = \frac{\sqrt{w_j} \cdot \tau}{1/\sqrt{w_j}} = w_j \cdot \tau\,. \tag{3.22}$$

Using width-absorbing cusp normalizers, the expansion at the j-th cusp is given by

$$(f|_k \mathcal{N}_j)(\tau) = \sum_{n=0}^{\infty} a^{(j)} q^n\,, \tag{3.23}$$

and can therefore always be expanded in $q = q_1$, which is useful and simplifies the expressions.

Definition 3.3.2 (Minimal height of $\mathcal{F}(G)$)**.** We define the *minimal height* of $\mathcal{F}(G)$ to be the quantity

$$Y_0 := \frac{\sqrt{3}}{2N_{\max}}\,, \tag{3.24}$$

where $N_{\max}$ is the largest cusp width of G.

To compute the pullback of $\tau \notin \mathcal{F}(G)$ into $\mathcal{F}(G)$ we make use of Millington's theorem (see section 2.2.2). The procedure can be described as follows:

1. Compute the pullback of τ into $\mathcal{F}(\Gamma)$, which creates a word in S, T, T^{-1}.
2. Insert the corresponding word in S, T, T^{-1} into the $\mathrm{PSL}(2,\mathbb{Z})$ and S_μ representations to obtain a map $\gamma_\tau \in \Gamma$ and its permutation $\sigma_\tau := \phi(\gamma_\tau) \in S_\mu$.
3. Let $\sigma_i := \phi(\gamma_i) \in S_\mu$ denote the permutation representations of the coset representatives. Then the pullback goes into the (unique) coset of label j for which $\sigma_\tau(\sigma_j(1)) = 1$.
4. The pullback into $\mathcal{F}(G)$ is thus given by $\gamma_w = \gamma_j \cdot \gamma_\tau \in \Gamma$.

Once the pullback $w = \gamma_w(\tau)$ into $\mathcal{F}(G)$ has been found, we need to identify the cusp that is *closest* to the pullbacked point (in the sense that its Fourier expansion converges the fastest). This leads to a function (following [29, 40, 41])

$$I : \mathcal{H} \to \{1, ..., n(c)\}\,, \tag{3.25}$$

which returns the cusp label k for which the Fourier expansion converges fastest at point w. The complete pullback is thus given by

$$\tau^* = \left(\mathcal{N}_{I(w)}^{-1} \cdot \gamma_w\right)(\tau)\,. \tag{3.26}$$

These pullback routines were contributed by Strömberg to PSAGE [46] and have been used in this project as well.

Hejhal's method for multiple cusps can be summarized as follows: For each cusp j, we select a fixed number of equally spaced points along a horocycle and compute their pullbacks into $\mathcal{F}(G)$. Then we *match* the expansion with the cusp whose Fourier expansion converges fastest on the pullbacked point. This gives us

$$\tau^*_{m,j} = \left(\mathcal{N}^{-1}_{I(m,j)} \cdot \gamma_w \cdot \mathcal{N}_j\right)(\tau_m) = \begin{pmatrix} a_{m,j} & b_{m,j} \\ c_{m,j} & d_{m,j} \end{pmatrix}(\tau_m)\,, \tag{3.27}$$

where $I(m,j) := I(w)$. In analogy to section 3.2 we thus get (see [40])

$$\begin{aligned} a^{(j)}_n Y^{\frac{k}{2}} \exp(-2\pi nY) &\approx \frac{1}{2Q} \sum_{m=0}^{2Q-1} (F|_k\mathcal{N}_j)(\tau_m) \exp(-2\pi i n x_m)\,, && (3.28)\\ &= \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\frac{|c_{m,j}\tau_m + d_{m,j}|}{(c_{m,j}\tau_m + d_{m,j})}\right)^k (F|_k\mathcal{N}_{I(m,j)})(\tau^*_{m,j}) \exp(-2\pi i n x_m)\,, && (3.29)\\ &= \sum_{l=0}^{M_0} a^{(I(m,j))}_l \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\frac{|c_{m,j}\tau_m + d_{m,j}|}{(c_{m,j}\tau_m + d_{m,j})}\right)^k (y^*_{m,j})^{\frac{k}{2}} \exp\Big(2\pi i(l\tau^*_{m,j} - nx_m)\Big)\,. && (3.30) \end{aligned}$$

For the analog of eq. (3.13) we hence get

$$a^{(j)}_n Y^{\frac{k}{2}} \exp(-2\pi nY) = \sum_{j'=1}^{\kappa} \sum_{l=0}^{M_0} a^{(j')}_l V^{(j,j')}_{n,l}\,, \tag{3.31}$$

with

$$V^{(j,j')}_{n,l} = \frac{1}{2Q} \sum_{I(m,j)=j'} \left(\frac{\left|c_{m,j}z_m + d_{m,j}\right|}{\left(c_{m,j}z_m + d_{m,j}\right)}\right)^k (y^*_{m,j})^{\frac{k}{2}} \exp\Big(2\pi i(lz^*_{m,j} - nx_m)\Big)\,, \tag{3.32}$$

where $\sum_{I(m,j)=j'}$ denotes the sum over all $0 \le m \le 2Q-1$ for which $I(m,j) = j'$. So we get

$$\sum_{j'=1}^{n(c)} \sum_{l=0}^{M_0} a^{(j')} \tilde{V}^{(j,j')}_{n,l} = 0\,, \tag{3.33}$$

where

$$\begin{aligned} \tilde{V}^{(j,j')}_{n,l} &= V^{(j,j')}_{n,l} - \delta_{j,j'}\delta_{n,l} Y^{\frac{k}{2}} \exp(-2\pi nY)\,, && (3.34)\\ &= \frac{1}{2Q} \sum_{I(m,j)=j'} \left(\frac{\left|c_{m,j}z_m + d_{m,j}\right|}{\left(c_{m,j}z_m + d_{m,j}\right)}\right)^k (y^*_{m,j})^{\frac{k}{2}} \exp\Big(2\pi i(lz^*_{m,j} - nx_m)\Big) - \delta_{j,j'}\delta_{n,l} Y^{\frac{k}{2}} \exp(-2\pi nY)\,. && (3.35) \end{aligned}$$

This gives us a linear system of equations, which we can solve again by imposing a Victor Miller normalization on the Fourier expansion for the cusp at infinity.

3.4 A Block-Factored Formulation of Hejhal's Method

The matrix V, whose entries are given by eq. (3.13), can be written as the matrix product of two matrices (see Voight and Willis [44], who used an analogous factorization for a similar problem)

$$V = J \cdot W\,, \tag{3.36}$$

with

$$J_{n,m} = \frac{1}{2Q}\left(\frac{|c_m z_m + d_m|}{(c_m z_m + d_m)}\right)^k \exp(-2\pi i n x_m)\,, \tag{3.37}$$

and

$$W_{m,l} = (y^*_m)^{\frac{k}{2}} \exp(2\pi i l z^*_m)\,. \tag{3.38}$$

Similarly, we can write $\tilde{V}_{n,l}$ whose entries are given by eq. (3.15) as

$$\tilde{V} = J \cdot W - D\,, \tag{3.39}$$

where D is a diagonal matrix whose entries are $Y^{\frac{k}{2}} \exp(-2\pi n Y)$. For subgroups with more than one cusp, V can be factored into a *block-factored* form. For example, for two cusps, we would get a matrix of the form

$$\tilde{V} = \begin{pmatrix} J^{(1,1)} \cdot W^{(1,1)} & J^{(1,2)} \cdot W^{(1,2)} \\ J^{(2,1)} \cdot W^{(2,1)} & J^{(2,2)} \cdot W^{(2,2)} \end{pmatrix} - \begin{pmatrix} D^{(1)} & 0 \\ 0 & D^{(2)} \end{pmatrix}\,. \tag{3.40}$$

Obviously, the same approach works analogously for more than two cusps. Factorizing the matrices involved not only simplifies the expressions, but can also significantly improve performance, as we will discuss in the next chapter.

CHAPTER 4

Numerical Computation of Fourier Coefficients of Modular Forms on Noncongruence Subgroups

Parts of this chapter were used in the paper [25, Section 4].

In this chapter we present a new iterative mixed-precision algorithm that is based on Hejhal's method (see chapter 3) and show that this algorithm runs significantly faster than previous algorithms. We apply it to the numerical computation of Fourier coefficients of noncongruence modular forms. Due to the lack of non-trivial Hecke operators, this has so far been the only feasible tool to compute modular forms on general noncongruence subgroups [47, 48]. For this reason, the theory of noncongruence modular forms is still poorly understood, despite the important work of Atkin and Swinnerton-Dyer [8], Scholl [49], Chen [50], and Calegari, Dimitrov and Tang [9]. Our method makes it possible to compute to modular forms on noncongruence subgroups to *high* precision (typically to more than 1000 digits). These results can then be used to identify the coefficients as algebraic numbers using the LLL algorithm (see section 4.4.1).

4.1 Preliminary Remarks on Software and Implementation

4.1.1 Arbitrary Precision Arithmetic

The majority of programs use single (32-bit) or double (64-bit) precision to perform floating point arithmetic. By the IEEE 754 standard, doubles use 53 bits for the mantissa, 11 bits for the exponent, and 1 bit for the sign [52, Section 3.1]. Since $2^{-53} \approx 1 \cdot 10^{-16}$, this is about 16 digits of decimal precision. The smallest representable double has a size of $2^{-1022} \approx 2 \cdot 10^{-308}$ [52, Section 3.1.2] which means that the exponent range of doubles (in decimal) is given by about ± 308. These ranges and accuracies are sufficient for most applications (especially when dealing with real-world data) which is why the modern hardware *natively* supports them. By *native*, we mean that the arithmetic units are specifically designed to handle (for example) 64-bit floating point operations, and can thus perform them in one cycle. (In fact, advanced CPU instructions such as vectorization typically even allow multiple double operations to be performed in a cycle.) When 64-bit precision is not sufficient, the expressions must be broken into smaller chunks that the CPU can handle. To do this in an optimized way, arbitrary precision libraries such as MPFR [53] and Arb [54] have been developed. Performing arithmetic on a floating point number with p bits precision can be done with $O(p\log(p)\log(\log(p)))$ complexity [52, Section 2.3]. However, when switching from *hardware supported* types such as doubles to arbitrary precision types, one gets a huge performance penalty, which (loosely speaking) comes from the fact that arbitrary precision types are not natively supported by the hardware. To illustrate this, consider the following example: Given a real matrix A of dimension 300×300 and a vector b of length 300, both consisting of random entries between 0 and 1, we want to solve $Ax = b$ by LU decomposition. This operation takes about 1.74 ms using doubles. Performing the same computation with the same precision using arbitrary precision arithmetic instead takes $4.58s$, or 3 orders of magnitude longer. Despite this huge performance penalty, there are some applications where arbitrary precision arithmetic is unavoidable, for example, when the problem is very ill-conditioned or when the solution must be known with high accuracy.

4.1.2 Software used in this Project

Most of the code was implemented in Cython [55], which is a compiled and typed extension of Python [56]. The advantage of Cython is that one can write *easy* Python code and optimize the performance of critical parts to get close to the performance of C [57]. The C compilation of Cython also makes it easy to wrap functions from Arb [54], which is a highly optimized C library for arbitrary precision (and additionally interval) arithmetic. Arb is currently the fastest arbitrary-precision library [58, 59], and we used it mainly for its performance and optimized routines (such as linear algebra, fast Fourier transforms, power series reversion, polynomial arithmetic, elementary functions, etc.). For the computations in section 5.2.4, we also used its interval arithmetic to control rounding errors. We also made extensive use of Sage's [35] various implementations and wrappers. In addition, we used the pullback routines of psage [46]. PARI [36] was used for number field arithmetic and its implementation of the LLL algorithm.

4.1.3 Source Code

The source code used for chapters 4, 5, 6 and 7 is available at [60].

4.2 Krylov Subspace Solvers

Classical direct solving of linear systems of equations typically runs in $O(N^3)$ complexity. For example, the Gaussian elimination process used to compute an LU decomposition uses $\sim \frac{2}{3}N^3$ [61, Eq. 20.8] floating operations. The goal of iterative solvers is to use a small number of *cheap* $O(N^2)$ operations to compute a solution to a linear system of equations by improving the accuracy of the solution during each iteration. The iteration count (i.e., the number of iterations until the solution is computed with sufficient accuracy) of iterative solvers is often difficult to predict and can be very high for some problems (so high, in fact, that using direct solving techniques may be faster). A rule of thumb when solving a linear system of equations iteratively is that the matrix should be ”not too far from normal and its eigenvalues clustered” [61, p. 314] in order to achieve fast convergence rates. Iterative solvers also typically excel for problems where the input matrix has special properties (such as sparseness, symmetry, or Hermitianness).

Definition 4.2.1 (Krylov Subspace)**.** Given a matrix $A \in \mathbb{C}^{N \times N}$ and a vector $b \in \mathbb{C}^N$, we define the *Krylov subspace* of index m to be the space [61, Eq. 33.5]

$$\mathcal{K}_m = < b, Ab, ..., A^{m-1}b >, \tag{4.1}$$

i.e., to be the space that is spanned by powers of A times b.

Since the Krylov subspace $\mathcal{K}_m$ has linearly independent basis vectors for $m < N$, it can be transformed into an orthonormal basis, which we denote by Q_m. Such an orthonormal basis is typically formed by so-called *Arnoldi iterations*. Most iterative solvers belong to the *Krylov subspace solvers*. These solvers form a Krylov subspace of smaller dimension than the original space and try to approximate a solution vector from this subspace.

4.2.1 GMRES

In this project we have implemented a Krylov solver that is based on GMRES (short for *Generalized Minimal Residual Method*) due to Saad and Schultz [62] (see also [61, Sec. 35] for more details). The basic idea of GMRES is that, given a linear system of equations $A \cdot x = b$, x is approximated by a vector $x_m \in \mathcal{K}_m$ that minimizes

$$||Ax_m - b|| = ||A\mathcal{K}_m c - b||, \tag{4.2}$$

for some $c \in \mathbb{C}^N$. Since this procedure is numerically unstable, it is preferable to work with an orthonormal Krylov basis Q_m instead (this also speeds up the linear solving as we will see later). This amounts to minimizing

$$||AQ_m y - b||, \tag{4.3}$$

for some $y \in \mathbb{C}^N$. We can transform this equation to [61, Eq. 35.4]

$$||Q_{m+1}\tilde{H}_m y - b||, \tag{4.4}$$

where $\tilde{H}_m$ is a $(m+1) \times m$ matrix consisting of a Hessenberg matrix with an appended row containing only the last entry. Multiplying by $Q_{m+1}^{-1} = Q_{m+1}^{\intercal}$ does not change this norm and yields

$$||\tilde{H}_m y - Q_{m+1}^{\intercal} b||, \tag{4.5}$$

which we can write as [61, Eq. 35.6]

$$||\tilde{H}_m y - ||b||e_1|| =: r_m, \tag{4.6}$$

where $e_1 = (1, 0, 0, ...)^{\intercal}$. y can be obtained from this by linear solving (note that one can exploit the fact that $\tilde{H}_m$ is a Hessenberg matrix to perform this operation with $O(N^2)$ complexity). Finally, an approximation of our solution vector is given by

$$x_m = Q_m y. \tag{4.7}$$

For this project, we implemented a variant of GMRES that uses the modified Gram-Schmidt method to orthogonalize the Krylov subspace and solve eq. (4.6) in $O(N^2)$ complexity.

4.3 Iterative Computation of Fourier Coefficients

We have seen in section 3.4 that the matrices produced by Hejhal's method can be block-factored into products of matrices. Since (classical) matrix multiplication is of $O(N^3)$ complexity, this means that the construction of V also requires $O(N^3)$ operations and is thus quite expensive. This construction can be avoided by using iterative solving techniques. Such an approach was used by Klug, Musty, Schiavone, and Voight [12] to compute Taylor expansions of modular forms. More specifically, KMSV used a Krylov subspace method called *power method* to iteratively compute eigenvectors from which a basis of modular forms can be obtained.

As discussed in section 4.2, the limitation of such iterative methods is that the number of iterations can become very large (especially for *larger* problems where the resulting matrices are larger). For example, using eq. (3.40) to compute $f_0 \in S_8(\Gamma_0(2))$ to 50 digits using GMRES takes 93 iterations, which means that solving the linear system directly would have been faster. However, we can easily improve the convergence rate by noticing that the largest entries are located on the diagonal (this is the effect of Hejhal's method, see section 3). Following from this, we scale each column by the diagonal term. This gives (recall that right-multiplying a matrix by a diagonal matrix corresponds to scaling its columns by the diagonal entries)

$$\tilde{V}_{\mathrm{sc}} := \tilde{V} \cdot \begin{pmatrix} D^{(1)} & 0 \\ 0 & D^{(2)} \end{pmatrix}^{-1}, \tag{4.8}$$

$$= \left(\begin{pmatrix} J^{(1,1)} \cdot W^{(1,1)} & J^{(1,2)} \cdot W^{(1,2)} \\ J^{(2,1)} \cdot W^{(2,1)} & J^{(2,2)} \cdot W^{(2,2)} \end{pmatrix} - \begin{pmatrix} D^{(1)} & 0 \\ 0 & D^{(2)} \end{pmatrix} \right) \cdot \begin{pmatrix} D^{(1)} & 0 \\ 0 & D^{(2)} \end{pmatrix}^{-1}, \tag{4.9}$$

$$= \begin{pmatrix} J^{(1,1)} \cdot W^{(1,1)} & J^{(1,2)} \cdot W^{(1,2)} \\ J^{(2,1)} \cdot W^{(2,1)} & J^{(2,2)} \cdot W^{(2,2)} \end{pmatrix} \cdot \begin{pmatrix} D^{(1)} & 0 \\ 0 & D^{(2)} \end{pmatrix}^{-1} - \begin{pmatrix} \mathbb{1} & 0 \\ 0 & \mathbb{1} \end{pmatrix}. \tag{4.10}$$

The linear system therefore becomes

$$\tilde{V} \cdot c = b\,, \tag{4.11}$$

$$\underbrace{\tilde{V} \cdot D^{-1}}_{=\tilde{V}_{\mathrm{sc}}} \cdot \underbrace{D \cdot c}_{:=c'} = b\,, \tag{4.12}$$

which we can solve for c' to compute $c = D^{-1}c'$. The eigenvalues of $\tilde{V}_{\mathrm{sc}}$ are clustered closer together, so we can expect faster convergence rates. In fact, by working with $\tilde{V}_{\mathrm{sc}}$, the number of iterations of GMRES for the previous example can be reduced to 13. A comparison of the residues after each iteration for both approaches can be found in fig. 4.1.

A transformation that reduces the number of iterations of an iterative solver is called *preconditioning transformation*, and working with $\tilde{V}_{\mathrm{sc}}$ instead of $\tilde{V}$ could be considered a preconditioning step. However, the highlighted example of $\Gamma_0(2)$ is very simple. We will see later in this section that working with $\tilde{V}_{\mathrm{sc}}$ is not sufficient to compute larger examples in reasonable time. To further reduce the number of iterations, a *preconditioner* matrix M is needed, which allows to solve the better conditioned system of equations

$$M \cdot \tilde{V}_{\mathrm{sc}} \cdot c' = M \cdot b\,, \tag{4.13}$$

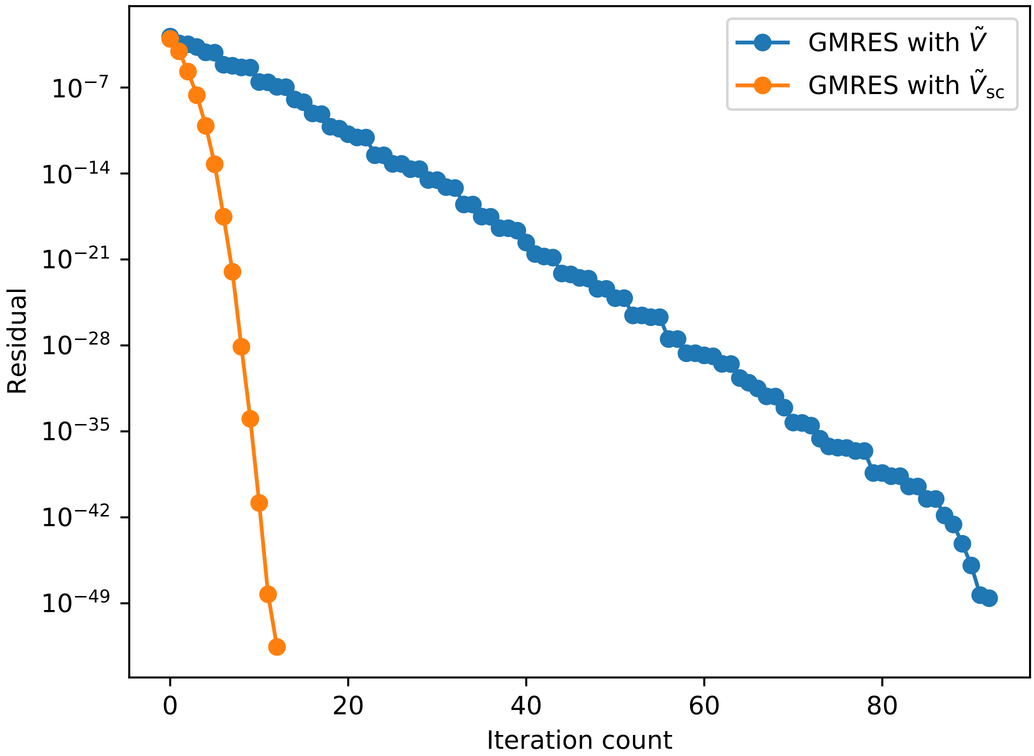

Figure 4.1: Illustration of the iterative computation of $f_0 \in S_8(\Gamma_0(2))$ to 50 digits precision using GMRES (taking $M_0 = 47$). Working with $\tilde{V}_{sc}$ reduced the number of iterations from 93 to 13.

(where we obviously evaluate $M \cdot \tilde{V}_{sc} \cdot c$ as $M \cdot (\tilde{V}_{sc} \cdot c)$ instead of $(M \cdot \tilde{V}_{sc}) \cdot c$ to avoid $O(N^3)$ matrix multiplication). However, obtaining such a preconditioner seems non-trivial for our application because $\tilde{V}_{sc}$ is non-hermitian, non-symmetric, and dense. In fact, we do not even know $\tilde{V}$ explicitly, and, as discussed earlier, constructing it is a $O(N^3)$ operation, so we would be in the same order of magnitude as just using a direct method to compute the solution. The key observation in resolving this dilemma is that $\tilde{V}_{sc}$ can be safely inverted at low precision. This can be seen from eq. (4.10): The entries of the block matrices W decay and become effectively zero at low precision. Since J does not change the order of magnitude, $J \cdot W$ also has decaying columns. However, by subtracting the unit diagonal matrix, we ensure that each column has at least one non-zero entry. This means that if the Fourier expansion order M_0 is very large, we asymptotically approach the unit matrix, which is (and remains) well-conditioned for inversion. Our approach is therefore to set the preconditioner M to a low-precision inverse (or something similar) of $\tilde{V}_{sc}$. Such an approach uses *mixed-precision* arithmetic, which is a relatively new concept that originated in high-performance computing.

Algorithm 1 Algorithm for computing Fourier expansion coefficients using GMRES

1: Compute block-factored form of $\tilde{V}_{\mathrm{sc}}$ at full precision
2: Construct $\tilde{V}_{\mathrm{sc,double}}$ at double-precision
3: Compute $\bar{L} \cdot \bar{U} = \tilde{V}_{\mathrm{sc,double}}$ at double-precision
4: Cast $\bar{L}, \bar{U}$ to full precision
5: Solve $\left(\bar{L} \cdot \bar{U}\right)^{-1} \tilde{V}_{\mathrm{sc}} \cdot a' = \left(\bar{L} \cdot \bar{U}\right)^{-1} b$ at full precision using GMRES
6: Return $a = D^{-1} \cdot a'$ at full precision

4.3.1 Mixed-Precision Arithmetic

For an overview of various methods and applications that use mixed-precision arithmetic, see [63]. The basic concept of mixed-precision arithmetic is to perform computationally expensive parts of an algorithm in faster, low-precision arithmetic without sacrificing the precision of the end result. So far, applications of mixed-precision arithmetic have typically replaced double (64-bit) arithmetic with 32-/16-bit arithmetic, which has faster memory bandwidth and vectorization potential and is supported by specialized hardware such as GPUs and tensor cores. In this project, we switch between arbitrary-precision arithmetic and double arithmetic, and according to the results in section 4.1.1, the speedup when switching between them is even greater because arbitrary-precision arithmetic is not natively supported by hardware and is therefore very slow. In the context of iterative solvers, it has been shown and analyzed that low-precision inverses (of possibly even highly ill-conditioned matrices) can serve as good preconditioners for iterative methods [64–66]. (In general, inverses are good preconditioners because one approximates the unit matrix which has the maximally clustered eigenvalue spectrum. Of course, if one knows the inverse to full precision, the problem can be solved in one iteration, but obtaining such an inverse is more expensive and numerically unstable than solving the problem directly.) Our approach is therefore to explicitly construct $\tilde{V}$ in 64-bit double arithmetic and compute an approximate inverse using a direct method. Since these operations are performed in double arithmetic, their contribution to CPU time can be neglected in our examples.

4.3.2 Preconditioned GMRES

To precondition the GMRES solver with a low-precision inverse, we first construct $\tilde{V}_{\mathrm{sc}}$ in double precision (which we will denote as $\tilde{V}_{\mathrm{sc,double}}$) and compute its LU decomposition

$$\bar{L} \cdot \bar{U} = \tilde{V}_{\mathrm{sc,double}} \,, \tag{4.14}$$

where $\bar{L}$ and $\bar{U}$ denote the L and U factors up to double precision. To compute the action of the inverse of $\tilde{V}_{\mathrm{sc,double}}$, it is advantageous not to explicitly form $\tilde{V}^{-1}_{\mathrm{sc,double}}$, which is computationally expensive, ill-conditioned, and destroys potential sparseness. A better approach is to use [66]

$$\tilde{V}^{-1}_{\mathrm{sc,double}} x = \bar{U}^{-1} \bar{L}^{-1} x \,. \tag{4.15}$$

The actions of $\bar{L}^{-1}$ and $\bar{U}^{-1}$ on a vector can be computed using $O(N^2)$ triangular solves. Although the inverse is never explicitly formed, we will refer to this approach as *computing the inverse* for simplicity. The algorithm for preconditioned GMRES is illustrated in Algorithm 1. The advantage of this algorithm is that GMRES gains *at least* 16 digits during each iteration (assuming $\tilde{V}_{\mathrm{sc,double}}$ is

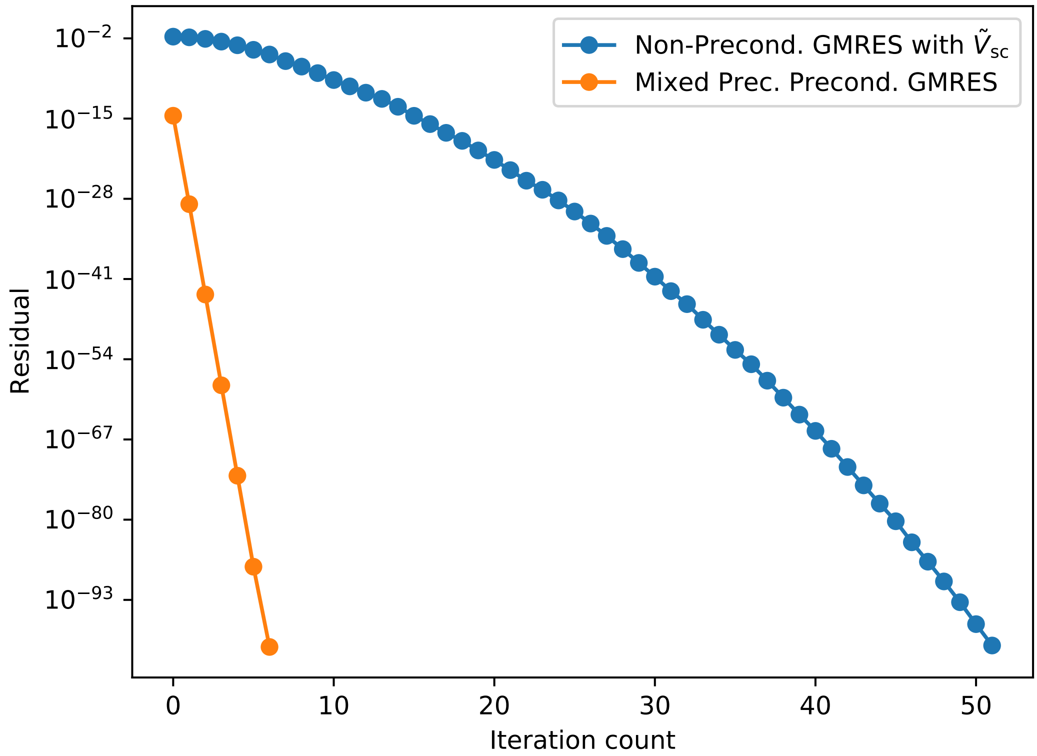

Figure 4.2: Comparison of precond. and non-precond. GMRES for the application of computing $f_0 \in S_2(\Gamma_0(20))$ to 100 digits precision (taking $M_0 = 868$). The preconditioned version reduces the iteration count from 52 to 7 iterations.

well-conditioned). The reason for this upper bound on the number of iterations (at least heuristically) comes from the fact that the inverse is known to 16 digits precision, which means that the solution can be refined to 16 digits precision during each iteration. This convergence rate is not only very fast, but it is also remarkable that the upper bound on the number of iterations is (in principle) independent of the problem and the size of the matrices involved. (We say "in principle" only because we assume here that the inverse of the matrix can be computed with 16-digit precision). For our previous example of $f_0 \in S_8(\Gamma_0(2))$ with 50 digits precision, this means that the number of iterations can be reduced from 13 iterations to only 3 iterations. The advantage of the preconditioned GMRES algorithm becomes even more apparent for larger examples. For example, computing $f_0 \in S_2(\Gamma_0(20))$ to 100 digits of precision takes 52 iterations with non-precond. GMRES and only 7 with precond. GMRES, as shown in fig. 4.2.

4.3.3 Iterative Refinement

Because GMRES must form a Krylov subspace, the action of $\tilde{V}_{sc}$ on a vector must be evaluated with full precision during each iteration, which is (comparatively) expensive. An alternative iterative

Algorithm 2 Algorithm for computing Fourier expansion coefficients using mixed-precision iterative refinement

1: Compute block-factored form of $\tilde{V}_{\mathrm{sc}}$ at full precision
2: Construct $\tilde{V}_{\mathrm{sc,double}}$ at double-precision
3: Compute $\bar{L} \cdot \bar{U} = \tilde{V}_{\mathrm{sc,double}}$ at double-precision
4: Use $\bar{L} \cdot \bar{U}$ to solve $\tilde{V}_{\mathrm{sc}} \cdot a' = b$ at 64-bit
5: **for** i=0:max_iter-1 **do**
6:     Compute $r = b - \tilde{V}_{\mathrm{sc}} \cdot a'$ at $(i+2) \cdot 16$ digits precision
7:     Use $\bar{L} \cdot \bar{U}$ to solve $\tilde{V}_{\mathrm{sc}} \cdot d = r$ at 64-bit
8:     Compute $a' = a' + d$
9:     **if** converged **then**
10:         break
11:     **end if**
12: **end for**
13: Return $a = D^{-1} \cdot a'$ at full precision

algorithm that does not create a Krylov subspace is given by *iterative refinement*. Iterative refinement (IR) is a relatively old technique, first applied by Wilkinson [67] in 1948, and can be viewed as Newton's method on the function $r(x) = A \cdot x - b$ [68]. In our application, the low-precision inverse can be used to iteratively refine the solution vector during each iteration. Since we do not form a Krylov subspace, we can gradually increase the precision during each iteration and do not need to perform all iterations at full precision. Thus, not only do we switch between double precision and arbitrary precision arithmetic, but we also choose different bit precisions when using arbitrary precision arithmetic. This approach makes even more use of *mixed precision* and is described in alg. 2. Using the (little) Gauss summation formula $\sum_{i=0}^{N} i = \frac{N(N+1)}{2}$, and assuming for simplicity that the complexity of arbitrary precision arithmetic grows linearly wrt. the precision, we can estimate that gradually increasing the precision in line 6 of alg. 2 should give a speedup of about a factor of two. However, this estimate only holds if the ring operations of arbitrary precision are considered independently. In practice, the matrices involved will typically have decaying columns, which means that working at a lower precision will not only reduce the complexity of the ring operations themselves, but also (and more importantly) reduce the number of ring operations to be performed, since many columns can be neglected from a lower precision perspective (more details on this can be found in section 4.3.5). This means that IR can in principle be more than twice as fast as GMRES.

If the approximate inverse is computed to 16 digits precision then iterative refinement gains 16 digits precision during each iteration. Unlike GMRES, the convergence rate can only be linear, which means that the number of iterations of IR is greater than or equal to that of GMRES.

4.3.4 GMRES vs. Iterative Refinement

As discussed in the previous section, GMRES can have a lower iteration count than IR, while the iterations of IR are on average *cheaper* because they do not have to be performed at the target precision. It is therefore interesting to examine which of these tradeoffs is advantageous in practice. As an example, we compute $\Delta \in S_{12}(\Gamma)$ to 1000 digits precision. This takes 39 iterations with GMRES, which achieves superlinear convergence and gains an impressive 48 digits in the last iteration. IR converges linearly and takes 64 iterations for the same example. However, despite the larger number of iterations, IR takes only $\sim 16s$ to compute while GMRES takes $\sim 39s$. For higher index examples, GMRES typically converges superlinearly only for the last iterations, bringing its iteration count closer to that of IR. This makes the speedup of mixed-precision IR over GMRES even larger. For this reason, we have used mixed-precision IR as the numerical solver throughout this work.

4.3.5 Optimizing the Action of W

The action of W (given by eq. (6.4)) can be interpreted as the evaluation of a polynomial at different points q_m^* times factors $(y_m^*)^{\frac{k}{2}}$. It is a well-known result that the evaluation of a polynomial at different points can be achieved in $O(N \cdot \ln(N)^2)$ asymptotic complexity (see for example [69]). However, this asymptotic growth comes with a large constant, which makes this algorithm in practice slower than the classical $O(N^2)$ algorithms for the problems considered in this work (in addition, these asymptotically fast algorithms are usually quite ill-conditioned).

For the classical $O(N^2)$ algorithms, the most common choice would be Horner's method, which evaluates a polynomial at a single point using N multiplications and $N+1$ additions as well as $O(1)$ memory. However, since the powers of q_m^* decay relatively quickly, it is in practice much faster to use Arb's optimized dot-product routines [59], which, among other technical optimizations, evaluate each term with the lowest possible precision (note that smaller terms can be evaluated with less precision than larger terms without affecting the precision of the result). In addition, the dot product routines neglect all terms that do not affect the result. This is particularly useful because the iterative refinement algorithm (see algorithm 2) starts with much lower precisions (from 32 digits) than the target precision, which means that the polynomials can be truncated on average to lower orders, with many terms being neglected. Recall also that M_0 is chosen based on the lowest point inside the fundamental domain, so quite pessimistically, which means that the polynomials converge faster for many τ_m^*. We note, however, that the naive approach of using the dot product, which assembles the entries of the matrix W and computes its action by using the dot product row by row, is not ideal for two reasons: First, the construction of W is comparatively expensive because it requires N^2 multiplications at full precision, which cannot be accelerated any further. Second, and more importantly, storing W as a matrix requires N^2 of memory space, which becomes inconvenient for larger problems. For this reason, we use modular splitting (see for example [52, Section 4.4.3]), for which only some of the powers of q_m^* need to be precomputed and stored. Modular splitting evaluates a polynomial $P(x)$ by using the relations

$$P(x) = \sum_{n=0}^{N} a_n x^n = \sum_{l=0}^{j-1} x^l P_l(x) = \sum_{l=0}^{j-1} x^l \left(\sum_{m=0}^{k-1} a_{jm+l} y^m \right), \tag{4.16}$$

where $y = x^j$. Thus, by choosing j and k of size $O(\sqrt{N})$, we only need to store $O(N^{3/2})$ values and can evaluate $P_l(x)$ using dot products. Note that we do not use classical rectangular splitting here, because we do not want the terms of $P_l(x)$ to be uniformly distributed in order to make best use of the dot-product optimizations. We find that using Arb's dot product often leads to a speedup that is close to an order of magnitude compared to a naive Horner scheme.

4.3.6 Optimizing the Action of J

It is immediately apparent that the entries of J (given by eq. (3.37)) are uniform and cannot be truncated when working at a lower precision which makes matrix-vector multiplication of J very slow compared to W. Note, however, that J can be factorized further:

$$J = D_L \cdot F \cdot D_R\,, \tag{4.17}$$

where

$$(D_L)_{n',m} = \exp\left(\frac{\pi i(2Q-1)}{2Q}\cdot n'\right), \tag{4.18}$$

$$F_{n',m} = \exp\left(\frac{-2\pi i}{2Q}\cdot n'\cdot m\right), \tag{4.19}$$

$$(D_R)_{n',m} = \frac{1}{2Q}\left(\frac{|c_m z_m + d_m|}{(c_m z_m + d_m)}\right)^k \exp\left(\frac{\pi i M_s(2Q-1)}{2Q}\right)\exp\left(\frac{-2\pi i M_s}{2Q}\cdot m\right). \tag{4.20}$$

M_s denotes the index of the first non-zero coefficient (in general, M_s depends on the cusp, so we should write $M_s(j)$ instead, but for the sake of simpler notation, we assume M_s is equal for all cusps here) and $n' := n - M_s$ with the property $0 \leq n' \leq M - M_s$. D_L and D_R are diagonal matrices whose action can be computed in $O(N)$ operations. F is similar to the matrix of the classical discrete Fourier transform (DFT), but (in general) with some rows and columns missing. Nevertheless, we can compute the action of F by a DFT. To illustrate, suppose $M = 3$, $2Q = 4$ (of course, in practice we need $Q > M$), and we have a missing column at $m = 2$. Then the action of F on a vector can be written as

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & (\zeta_4)^{-1} & (\zeta_4)^{-3} \\ 1 & (\zeta_4)^{-2} & (\zeta_4)^{-6} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}, \tag{4.21}$$

where $\zeta_4 = \exp\left(\frac{2\pi i}{4}\right)$ is the 4-th root of unity. This is equivalent to calculating:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & (\zeta_4)^{-1} & (\zeta_4)^{-2} & (\zeta_4)^{-3} \\ 1 & (\zeta_4)^{-2} & (\zeta_4)^{-4} & (\zeta_4)^{-6} \\ 1 & (\zeta_4)^{-3} & (\zeta_4)^{-6} & (\zeta_4)^{-9} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ 0 \\ x_2 \end{pmatrix}, \tag{4.22}$$

and selecting the first 3 entries of the output vector. Thus, our strategy for computing the action of F on a vector is to zero-pad all entries of the input vector for which $I(m, j) \neq i$, perform a DFT, and then select the first M entries of the output vector. The advantage of using a DFT to compute the action of F is that fast Fourier transform (FFT) algorithms are available that have asymptotic complexity $O(N\ln(N))$ [70]. Unlike the polynomial multipoint evaluation algorithms mentioned in section 4.3.5, the FFT algorithms typically have only a small asymptotic constant. In practice, we found the running time to be approximately $c \cdot Q\ln(Q)$, where $c < 10$, if the largest prime factor of Q is reasonably small (we used the Arb implementation to compute the FFT, contributed by Pascal Molin). Since we have a free choice of $Q > M$, we choose Q *slightly* larger than M and with small prime factors to speed up the FFT. Compared to the direct approach of computing the action of J by matrix-vector multiplications, which has complexity $O(Q \cdot M_0)$ (the exact number of operations also depends on the number of cusps), it is usually much faster to use FFTs, and the bottleneck of the algorithm becomes the action of W. Additional advantages of factoring J in the form of eq. (4.17) are that the memory consumption becomes much lower because we only need to store diagonals and $2Q$ roots of unity, and we avoid the N^2 operation to compute the entries of J.

4.3.7 Construction of $\tilde{V}_{\mathbf{sc,double}}$

To construct $\tilde{V}_{\text{sc,double}}$, we truncate the columns of W so that terms that are effectively zero in double precision are ignored. Then we compute the action of J on the remaining columns of W by FFTs (using NumPy [71]), similar to section 4.3.6. The construction of $\tilde{V}_{\text{sc,double}}$ therefore requires $O(N^2 \ln(N))$ double precision operations.

4.3.8 Computing the LU Decomposition of $\tilde{V}_{\mathbf{sc,double}}$

$\tilde{V}_{\text{sc,double}}$ is a sparse matrix because all entries below the double machine epsilon are neglected. To compute its LU decomposition, we therefore use the sparse linear algebra routines of SciPy [72]. We are unaware of the computational complexity of these routines (which should depend on the sparseness and structure of $\tilde{V}_{\text{sc,double}}$ and its LU factors), but in practice they take only negligible CPU time (see section 4.3.10 for more details).

4.3.9 Performing the LU Solves

As discussed in the previous section, we use SciPy's sparse linear algebra routines to compute an LU decomposition of $\tilde{V}_{\text{sc,double}}$ in double arithmetic. When using this precomputed LU decomposition to perform the solves within the iterative refinement algorithm, one must be careful not to over-/underflow the double exponent range, which is finite (see section 4.1.1) and can easily be exceeded for elements within the residue vectors. One way to avoid underflow is to convert the LU decomposition to 53-bit arbs, which have an unlimited exponent range. However, storing the LU decomposed matrix as an Arb matrix is quite memory consuming because Arb does not currently provide sparse matrices and because the memory footprint of an Arb object is larger than that of a double. A preferable approach is based on the observation that the input vectors to the LU solvers have relatively uniformly distributed entries. For this reason, we scale all entries by a constant factor 2^e to bring them into double range, convert them to doubles, perform the LU solve in double arithmetic using SciPy, convert the result back to Arb, and scale the result back again. This approach uses much less memory and is faster.

4.3.10 Runtime Profile of the Algorithm

To understand the bottlenecks and limitations of the mixed-precision IR algorithm, it is useful to measure the time that is spent on each stage of the algorithm. As a first example, we consider the computation of $\Delta \in S_{12}(\Gamma)$ to 1000 digits precision (taking $M_0 = 429$). Then, the runtime profile can be summarized as follows (we decided not to create a plot of this profile because the double computation parts would not be visible):

1. Initialization of $\tilde{V}_{\text{sc}}$ in block-factored form: ≈ 0.75s

2. Construction of $\tilde{V}_{\text{sc,double}}$: ≈ 0.011s
3. Sparse LU factorization of $\tilde{V}_{\text{sc,double}}$: ≈ 0.0026s
4. IR iterations: ≈ 14.84s

During the first IR iteration, the action of W took ≈ 0.019s and the action of J took ≈ 0.011s. During the last IR iteration, the action of W took ≈ 0.46s and the action of J took ≈ 0.070s. As a more complicated example, we compute $f_0 \in S_4(\Gamma_0(6))$ to 1000 digits precision (taking $M_0 = 2553$). The runtime profile can then be summarized as follows:

1. Initialization of $\tilde{V}_{\text{sc}}$ in block-factored form: ≈ 17.98s
2. Construction of $\tilde{V}_{\text{sc,double}}$: ≈ 0.44s
3. Sparse LU factorization of $\tilde{V}_{\text{sc,double}}$: ≈ 1.69s
4. IR iterations: ≈ 963s

During the first IR iteration, the actions of W took a total of ≈ 0.83s and the actions of J took ≈ 0.73s. During the last IR iteration, the actions of W took about 30.99s and the actions of J took about 4.18s.

There are two important observations from these two runtime profiles. First, the bottleneck of the algorithm is given by the actions of W during the last iterations (i.e., at high precision). Second, the double-precision parts take a negligible amount of CPU time.

4.3.11 Restarting the Algorithm

Since the iterative refinement algorithm does not need to form a Krylov subspace, it can be restarted without losing convergence. One approach we have experimented with is to gradually increase the values of Q and M_0. For example, if a target precision of 500 digits is desired, one can first choose Q and M_0 so that convergence is achieved up to 100 digits of precision. One can then use these approximations of the lower coefficients to 100 digits precision to restart the algorithm with a larger choice of Q and M_0 to refine the residue from 10^{-100} to 10^{-250}, and then again to go from 10^{-250} to 10^{-500}. However, the performance gain from this approach seems to be quite limited, since the bottlenecks are given by the last iterations anyway. In addition, each restart creates some extra computations to set up J, W, and the preconditioner. Although there are some restart configurations that are faster than simply starting with the target values of Q and M_0, the performance impact is very small and finding these configurations can be inconvenient, so we did not use this approach for our computations.

4.3.12 Performance Comparison to Previous Methods

To examine how the different approaches perform in practice, we ran several benchmarks that compute modular forms on congruence subgroups numerically. Obviously, it makes little sense to compute modular forms on congruence subgroups numerically, since they can be computed efficiently explicitly. However, congruence groups serve as useful benchmarks, because their results can be easily verified, and because it does not make a difference for the algorithm whether the group is congruence or not. The first benchmark computes $\Delta \in S_{12}(\Gamma)$ with different precisions. These results can be found in tab. 4.1.

Digits / $n(c) \cdot M_0$	Classical	Non-Precond. GMRES	Mixed Precision IR
250 / $1 \cdot 110$	1.92s (0.3GB)	1.25s (0.24GB), 18 iter.	**0.37s (0.22GB), 16 iter.**
500 / $1 \cdot 216$	30.7s (0.68GB)	20.3s (0.34GB), 26 iter.	**3.21s (0.22GB), 32 iter.**
1000 / $1 \cdot 429$	6min55s (3.58GB)	4min22s (1.62GB), 36 iter.	**33.3s (0.27GB), 64 iter.**

Table 4.1: Benchmarks for the numerical computation of $\Delta \in S_{12}(\Gamma)$. The first column lists the precision with which the coefficients were computed (up to a loss of precision for higher coefficients), as well as the number of cusps and the expansion order M_0. The remaining columns list the elapsed CPU time, the peak memory usage, and the number of iterations for the iterative methods. For details on how the benchmarks were run, see remark 4.3.1.

Digits / $n(c) \cdot M_0$	Classical	Non-Precond. GMRES	Mixed Precision IR
100 / $3 \cdot 533$	7min15s (1.91GB)	1min38s (1.5GB), 45 iter.	**7s (0.32GB), 8 iter.**
200 / $3 \cdot 1043$	1h9min31s (9.48GB)	15min3s (5.84Gb), 63 iter.	**43s (0.47GB), 15 iter.**
400 / $3 \cdot 2061$	-	4h24min56s (29.19GB), 90 iter.	**6min19s (0.94GB), 30 iter.**

Table 4.2: Benchmarks for the numerical computation of $f_0 \in S_4(G)$ where G is a subgroup of signature $(17, 0, 3, 1, 2)$ that is generated by $\sigma_S = (1)(2\,4)(3\,7)(5\,10)(6\,11)(8\,14)(9\,15)(12\,13)(16\,17)$ and $\sigma_R = (1\,7\,4)(2\,11\,10)(3\,15\,14)(5)(6\,12\,13)(8\,17\,9)(16)$. For details on how the benchmarks were performed, see remark 4.3.1.

Remark 4.3.1. Some more context about how the benchmarks were run: All implementations are highly optimized from a technical point of view. The *classical* version constructs the matrices J and W (using recursive multiplications to compute the powers of q) and multiplies them (using Arb's matrix multiplication) to construct $\tilde{V}$. It then uses Arb's implementation of the LU decomposition to solve the linear system of equations. The *non-precond. GMRES* version constructs the matrices J and W and stores $\tilde{V}_{\mathrm{sc}}$ in a block-factored form. The actions of J and W are computed using Arb's matrix-vector multiplications. Then GMRES is used to iteratively solve the linear system of equations. The *mixed precision IR* version uses the mixed precision iterative refinement approach with optimized actions of W and J, which are presented in sections 4.3.5 and 4.3.6. The benchmarks were run on a `Intel Xeon E5-2680 v4 @ 2.40GHz` CPU and run in a single thread. Note that CPU times may vary depending on the load on the machine. The reported memory consumption is the peak memory consumption of the program, which can also vary slightly due to different garbage collector behavior.

As we can see, the mixed-precision algorithm outperforms the other algorithms in all categories. This outperformance becomes even more obvious in a second benchmark where we computed a cusp form of an index 17 non-congruence subgroup with signature $(17, 0, 3, 1, 2)$. The benchmarks for this group are listed in tab. 4.2. As one can see, the mixed-precision IR algorithm runs more than 40 times faster than non-precond. GMRES at 400 digits precision while using significantly less memory. For larger examples this ratio becomes even larger because the IR approach has a lower asymptotic complexity. Computing the examples of chapter 7 to 1500 digits precision would therefore be infeasible with the previous algorithms.

4.3.13 Numerical Stability for Large Examples

Increasing the target precision (and hence the values of M_0 and Q) does not affect the condition number of $\tilde{V}_{\mathrm{sc,double}}$ (up to some noise), as shown in fig. 4.3. This seems to be due to the fact that that the additional columns are similar to those of a unit matrix.

The index and the number of cusps of the considered subgroup influence the conditioning more noticeably. Although large-index examples were not the focus of this work, it is interesting to investigate whether they are well-conditioned enough to apply mixed precision iterative precision iterative refinement on them as well. To do this, we consider the subgroup $\Gamma_0(120)$ of signature $(288, 17, 16, 0, 0)$. It is immediately apparent that with an index of 288 and 16 cusps, the largest of which has a width of 120, $\Gamma_0(120)$ is significantly larger than the other examples considered. As a test of our algorithm we performed the numerical computation of $f_0 \in S_2(\Gamma_0(120))$ to 50 digits precision. To achieve convergence, we choose $M_0 = 2725$, which means that the resulting linear system of equations is of dimension 43600×43600, which is enormous in the context of in the context of arbitrary precision arithmetic. Nevertheless, we have found that iterative refinement converges quickly, as can be seen in fig. 4.4. Unlike the other examples, the size of $\tilde{V}$ reduces the precision gain per iteration to about to about 9 digits per iteration instead of 16. We also noticed that the resulting coefficients have *only* been computed to about 44 digits precision instead of 50, but of course this can be easily overcome by using a buffer for large index examples. The computation used 60 GB of memory and took 2 hours and 30 minutes of CPU time. We note that, in contrast to the other computations, we had to use dense linear algebra to perform the LU decomposition because the sparse routines returned a memory error. We conclude that mixed-precision iterative refinement can be efficiently applied to large index examples too.

4.3.14 Complexity of the Algorithm

Studying the complexity of the mixed-precision IR algorithm is relatively difficult. First of all, it makes sense to ignore all computations that can be done in double precision, because due to their technical optimization, their contribution can be neglected compared to the parts that use arbitrary precision arithmetic (at least for the size of problems considered in this work, and taking the limit $N \to \infty$ would lead to conditioning problems at some point anyway), see section 4.3.10. When analyzing the performance with respect to N (we use N synonymously for Q and M_0 because they are usually proportional to each other), the asymptotic bottleneck is given both in theory and in practice by the action of W. The complexity of this computation is $O(N^2)$ (at least in practice, as discussed in section 4.3.5, the theoretical asymptotic complexity is $O(N \cdot \ln(N)^2)$), but with a very small constant due to the decaying columns of W. We also note that, unlike most iterative methods, the iteration count of our method depends only on the precision and is thus truly independent of N.

A more meaningful measure would be the bit complexity of the algorithm. However, this seems impossible to calculate due to the constantly changing working precision, floating point types, and decay rates of the dot product terms.

4.3.15 Summary

We have shown in this section how mixed-precision iterative refinement can be used to make the numerical computation of modular forms for general subgroups significantly faster which makes it feasible to compute examples of non-congruence modular forms. Because of its generality, we expect that the idea of computing a low precision inverse to iteratively solve the linear system of equations can also be beneficial for the arbitrary precision computation of other types of modular forms, such as Maass cusp forms and Taylor expansions of modular forms. We have also shown that the use of

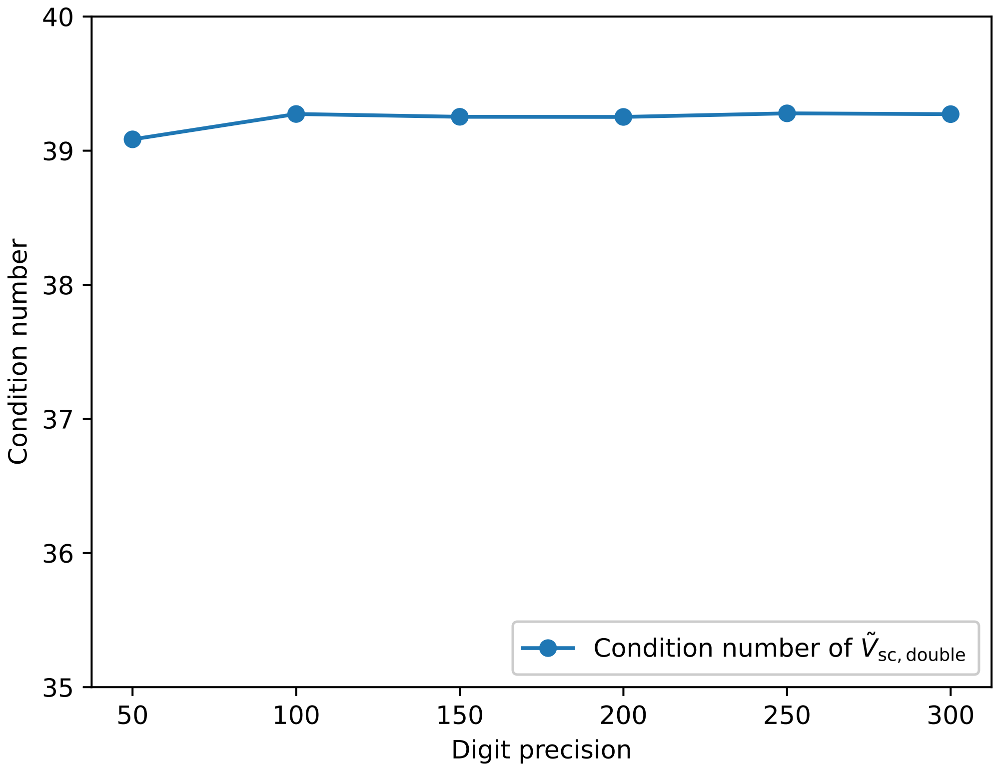

Figure 4.3: Illustration of the condition number of $\tilde{V}_{\mathrm{sc,double}}$ for varying target precisions. For the example we used the cusp form that was considered in tab. 4.2.

mixed-precision arithmetic can lead to enormous performance gains, which may make it even more attractive, especially in the field of arbitrary precision arithmetic.

4.4 Recognizing Fourier Coefficients as Algebraic Numbers

We can use the algorithm of section 4.3 to compute numerical estimates of the Fourier coefficients of modular forms (and cusp forms) with high precision. These expressions can then be recognized as algebraic numbers using the LLL algorithm.

4.4.1 The LLL Algorithm

The LLL algorithm (named after its authors Lenstra-Lenstra-Lovász [51]) is a lattice reduction algorithm that tries to find the *shortest* lattice in polynomial time. This algorithm has many applications, but in our context we use it to detect linear and algebraic dependencies between floating-point numbers. Given a set of N complex numbers $\{z_1, z_2, ..., z_N\}$, the LLL algorithm returns integers $c_1, c_2, c_3, ..., c_N$

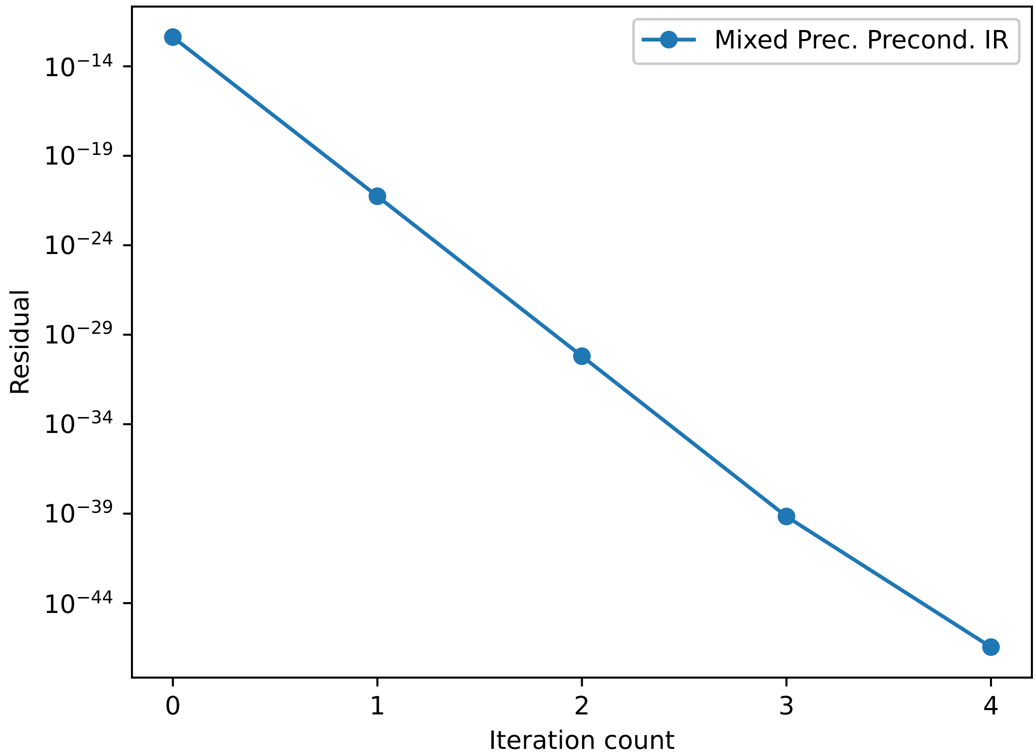

Figure 4.4: Illustration of the iterative computation of $f_0 \in S_2(\Gamma_0(120))$ to 50 digits precision using mixed precision IR (taking $M_0 = 2725$).

such that

$$\sum_{n=1}^{N} c_n z_n \approx 0\,. \tag{4.23}$$

More details about this algorithm can be found in Cohen's book [73, Section 2.7.2] and we use its implementation within the `lindep` routine in PARI [36]. Obviously, up to finite precision, c_i can always be found for arbitrary complex numbers. It is therefore important to check whether the result of the LLL algorithm corresponds to a *true* solution. This can be done rigorously if bounds on the coefficients c_i are known in advance. For our examples this is not the case (and in fact the input numbers will not be rigorous in general). To discard invalid solutions, we append to the input vector a number known to be non-algebraic and of similar size. If the LLL algorithm then detects that the coefficient of this number is equal to zero (and thus that this number is not part of the solution), this provides high heuristic evidence that the result is indeed correct. The LLL algorithm can also be used to reduce the size of the coefficients of a polynomial defining a number field, thus finding a simpler representation of that number field. More details about this algorithm can be found in [73, Section 4.4.2], and we have used the PARI [36] routine `polredabs` to try to reduce all occurring number fields to their simplest form.

4.4.2 Determining K

We have seen in section 2.3 that modular forms on noncongruence subgroups are defined over a field K times a N-th root (where N denotes the cusp width at infinity) of an expression in K. To determine K, we first choose a cusp form that has the lowest weight and is not an old form. Then we choose an expansion coefficient that is linear in u (and nonzero). Raising this expression to the Nth power yields an expression in K. We then use the LLL algorithm to determine an algebraic dependence of this expression (an upper bound on the degree of the algebraic number is given by the size of the passport). Then we use `polredabs` (see section 4.4.1) to reduce this number field to a simpler form.

Example 4.4.1 (Determining $\boldsymbol{K}$)**.** It may be useful to illustrate the above procedure with an example. Let G be a noncongruence subgroup with signature $(16, 1, 2, 0, 1)$, which is generated by $\sigma_S = (1\,4)(2\,5)(3\,8)(6\,11)(7\,10)(9\,14)(12\,15)(13\,16)$ and $\sigma_R = (1)(2\,10\,11)(3\,7\,14)(4\,8\,5)(6\,16\,15)(9\,13\,12)$. From this we get that $\sigma_T = (1\,8\,7\,11\,16\,12\,6\,2\,4)(3\,5\,10\,14\,13\,15\,9)$, which means that the cusp at infinity has width 9. We have $\dim(S_2(G)) = 1$, so we choose the second coefficient of this cusp form, which is given by $c_2 = -2.057184... - 0.677479...i$. Let $t = c_2^9$. Using the LLL algorithm, we can guess that t is a root of the polynomial

$$
\begin{aligned}
&327968553690211870345147021367286169 6x^3 - 661460321992970732459602769307398622412 8x^2 \\
&+ 3607523620681479138330555369588007533376401x - 13065325542025101569450431888392725591 04\,,
\end{aligned}
$$

(it takes about 150 digits of precision to detect this algebraic dependency). Using `polredabs` on this polynomial yields in the more convenient number field $K = \mathbb{Q}(v)$, where

$$
v^3 - 6v - 16 = 0\,,
$$

with embedding $v = -1.647426... + 1.463572...i$.

We note that it would also be possible to determine K without using the LLL algorithm by computing the modular forms for all Galois conjugates in the passport, from which one can guess the polynomial from its roots, for example by using continued fractions to recognize its coefficients. Such an approach was taken by Richards [74]. This approach could reduce the precision needed to determine K while increasing the number of modular forms to compute. We have not yet used this approach, but it would be interesting to investigate its efficiency.

4.4.3 Determining u

u is an N-th root of an expression in K (see section 2.3). Note that u is not unique, and a good choice of u makes the expressions of the factors in K small which is not only nicer to read, but also makes it easier to recognize these factors with the LLL algorithm. To determine u, we use an expression that is linear in u and write it as $c \cdot u$, where $c \in \mathbb{Q}$. We then try to find a good choice of c that *absorbs* common factors and denominators.

Example 4.4.2 (Determining $\boldsymbol{u}$)**.** Continuing with example 4.4.1, we use the LLL algorithm to compute

$$
t(v) = \frac{-1}{2^{42} \cdot 7^7}(-2^6 \cdot 3^{12} \cdot 49667 \cdot 1452815993
$$

$$+ 2^4 \cdot 3^{13} \cdot 5 \cdot 14543 \cdot 393024407v + 3^{13} \cdot 167 \cdot 9697 \cdot 1862489v^2) .$$

The only common prime factor with power greater than 9 is given by 3. So we factor the expression linearly in u into $c_2 = 3 \cdot u$ where

$$u = (\frac{-1}{2^{42} \cdot 7^7}(-2^6 \cdot 3^3 \cdot 49667 \cdot 1452815993 \\ + 2^4 \cdot 3^4 \cdot 5 \cdot 14543 \cdot 393024407v + 3^4 \cdot 167 \cdot 9697 \cdot 1862489v^2))^{1/9} .$$

Next, we try to improve on this initial choice of u. We do this by looking at the next coefficient that is quadratic in u. We then recognize the expression

$$t_2(v) = \frac{c_3}{u^2} = -\frac{3^3 \cdot 11 \cdot 19 \cdot 79}{2^2 \cdot 137^2} - \frac{2^3 \cdot 3 \cdot 5 \cdot 23^2}{137^2}v - \frac{3 \cdot 5669}{137^2}v^2 ,$$

for which a new denominator of 2 and 137 *appears*. We therefore improve u in an additional iteration where we update u so that $t_2(v)$ has only trivial denominators. This is done by *absorbing* the additional factor of $2 \cdot 137$ into u, which results in our final choice:

$$u = \left(\frac{3^3 \cdot 49667 \cdot 1452815993}{2^{45} \cdot 7^7 \cdot 137^9} - \frac{3^4 \cdot 5 \cdot 14543 \cdot 393024407}{2^{47} \cdot 7^7 \cdot 137^9}v - \frac{3^4 \cdot 167 \cdot 9697 \cdot 1862489}{2^{51} \cdot 7^7 \cdot 137^9}v^2\right)^{1/9} .$$

This approach is rather tedious, but can be automated. However, the choice of u can be considered experimental, and there is no guarantee that the best choice of u has been found.

4.4.4 Determining the Expansion Coefficients

Once u and K have been determined, the expansion coefficients can be found by dividing the numerical expressions by the appropriate powers of u. The resulting expressions can then be identified as elements in K that are (hopefully) relatively small.

Example 4.4.3 (Recognizing expansion coefficients)**.** Using the same subgroup as in the previous examples, we get the following expansion for $f_0 \in S_2(G)$

$$f(q_9) = q_9 + (822u)q_9^2 + ((-68028v^2 - 253920v - 445797)u^2)q_9^3 + \dots . \tag{4.24}$$

CHAPTER 5

Numerical Computation of Belyi Maps and Modular Forms for Genus Zero Subgroups

Parts of this chapter were used in the paper [25, Section 5].

In this chapter, we use well-known and highly efficient algorithms to compute Belyi maps for genus zero noncongruence subgroups. From these we obtain the Hauptmodul (see section 2.8) and show how bases of cusp forms and modular forms can be obtained from it. This provides an alternative method to the approach taken in chapter 4, which is restricted to subgroups of genus zero, but provides rigorous results and is usually faster.

5.1 Computation of Genus Zero Belyi Maps

Theorem 5.1.1 (Atkin-Swinnerton-Dyer)**.** A necessary and sufficient condition for $f(\tau)$ to be a modular function on a subgroup of finite index in Γ is that $f(\tau)$ is an algebraic function of j and that its only branch points should be branch points of order 2, where $j = 1728$, and branching points of order 3, where $j = 0$, and branching points at which j is infinite.

Proof. See Atkin-Swinnerton-Dyer [8, Theorem 1]. □

In particular, note that j, when viewed as a function on the modular curve $X(G)$ of some finite index subgroup $G \leqslant \Gamma$, gives an example of a *Belyi map*.

Definition 5.1.1 (Belyi map)**.** Let X be a compact Riemann surface. Then a holomorphic function

$$f : X \to \mathbb{P}^1(\mathbb{C}), \tag{5.1}$$

is said to be a *Belyi map* if it is unramified away from three points.

Belyi maps derive their name from a famous theorem of Belyi [75]

Theorem 5.1.2 (Belyi)**.** A compact Riemann surface X (equivalently, an algebraic curve) over $\mathbb{C}$ can be defined over $\overline{\mathbb{Q}}$ if and only if there exists a Belyi map on X.

Proof. See Belyi [76, Theorem 1]. □

Belyi maps and their computation are an interesting topic in their own right, with numerous applications in number theory and algebraic geometry. For an overview see the survey by Sijsling and Voight [13], we will only recall some of the computational details here.

Let G be a finite index subgroup of Γ. Then the covering map

$$R\,:\,X(G)\to X(\Gamma)\overset{j}{\cong}\mathbb{P}^1(\mathbb{C})\,, \tag{5.2}$$

is a Belyi map, where $X(G)=G\backslash\overline{\mathcal{H}}$ is the modular curve. If G is a subgroup of genus zero then the covering map $R(j_G)$ is a rational function in j_G, and branches over the images of the elliptic points and cusps. This means that R can be written as

$$R(j_G)=\frac{p_3(j_G)}{p_c(j_G)}=1728+\frac{p_2(j_G)}{p_c(j_G)}\,. \tag{5.3}$$

The ramification structure (i.e., the roots of p_2, p_3 and p_c) can be determined from the cycle type of σ_S, σ_R and σ_T. Let us illustrate this with an example.

Example 5.1.1 (Determining the ramification structure from the permutation triple)**.** Let G be a noncongruence subgroup with signature $(7,0,2,1,1)$ corresponding to the permutations $\sigma_S=(1)(2\,4)(3\,5)(6\,7)$, $\sigma_R=(1\,5\,4)(2\,7\,3)(6)$ and $\sigma_T=(1\,5\,2)(3\,4\,7\,6)$. By definition, p_2 must be of the form

$$p_2(j_G(\tau))=\prod_{i=1}^{7}\left(j_G(\tau)-j_G(e_{2,i})\right)\,, \tag{5.4}$$

where we denote $e_{2,i}$ to be the elliptic point of order two, located at the coset of index i. Since some of the evaluations at the elliptic points are equal, we can write this as

$$p_2(j_G(\tau))=(j_G(\tau)-j_G(e_{2,1}))(j_G(\tau)-j_G(e_{2,2}))^2(j_G(\tau)-j_G(e_{2,3}))^2(j_G(\tau)-j_G(e_{2,6}))^2\,. \tag{5.5}$$

This means that p_2 can be written in the form

$$p_2(j_G)=\left(j_G^3+A_2j_G^2+B_2j_G+C_2\right)^2\left(j_G+D_2\right)\,, \tag{5.6}$$

where (by Belyi's theorem) $A_2,B_2,C_2,D_2\in\overline{\mathbb{Q}}$. Similarly, p_3 and p_c can be factored into

$$p_3(j_G)=\left(j_G^2+A_3j_G+B_3\right)^3\left(j_G+C_3\right)\,, \tag{5.7}$$

and

$$p_c(j_G)=\left(j_G+A_c\right)^4\,, \tag{5.8}$$

where the roots are given by $j_G(e_{3,i})$, resp. $j_G(c_i)$.

Once the structure of p_2, p_3 and p_c has been determined, we can transform eq. (5.3) into

$$P(j_G):=p_3(j_G)-p_2(j_G)-1728p_c(j_G)=0\,, \tag{5.9}$$

where $P(j_G)$ is a polynomial whose coefficients are defined over symbolic expressions. The coefficients of $P(j_G)$ must vanish, which gives $\deg(P)=\deg(p_3)=\deg(p_2)$ polynomial equations in the unknowns $A_2, A_3, \ldots$. An additional equation is obtained by expanding $R(j_G)$ in $j_G(q_N)$ and by asserting that the constant term is equal to 744 if the cusp width at infinity is equal to one and vanishes otherwise.

Example 5.1.2 (Obtaining a system of nonlinear equations)**.** Continuing our example and substituting the appropriate expressions for p_2, p_3, and p_c into the coefficients of $P(j_G)$ in eq. (5.9) we obtain

$$
\begin{aligned}
0 &= -1728A_c^4 + B_3^3C_3 - C_2^2D_2\,,\\
0 &= 3A_3B_3^2C_3 - 6912A_c^3 + B_3^3 - 2B_2C_2D_2 - C_2^2\,,\\
0 &= 3A_3B_3^2 - 10368A_c^2 - 2B_2C_2 + (2A_3^2B_3 + (A_3^2 + 2B_3)B_3 + B_3^2)C_3 - (B_2^2 + 2A_2C_2)D_2\,,\\
0 &= 2A_3^2B_3 - B_2^2 + (A_3^2 + 2B_3)B_3 + B_3^2 - 2A_2C_2 + ((A_3^2 + 2B_3)A_3 + 4A_3B_3)C_3 - 2(A_2B_2 + C_2)D_2 - 6912A_c\,,\\
0 &= (A_3^2 + 2B_3)A_3 - 2A_2B_2 + 4A_3B_3 + 3(A_3^2 + B_3)C_3 - (A_2^2 + 2B_2)D_2 - 2C_2 - 1728\,,\\
0 &= -A_2^2 + 3A_3^2 + 3A_3C_3 - 2A_2D_2 - 2B_2 + 3B_3\,,\\
0 &= -2A_2 + 3A_3 + C_3 - D_2\,,
\end{aligned}
$$

and get the additional linear equation

$$3A_3 + C_3 - 4A_c = 0\,.$$

One can try to solve these systems of equations directly, for example using Gröbner bases [13, Section 2]. However, this quickly becomes infeasible for all but the simplest examples. A much more efficient approach is to use a numerical method to compute approximations to the evaluation of the Hauptmodul at the elliptic points and cusps. These approximations can then be used as initial values for Newton iterations to determine the unknown coefficients with high accuracy. The LLL algorithm can then be used to identify the expressions as algebraic numbers (analogous to section 4.4.4). This approach was proposed by Atkin-Swinnerton-Dyer [8], and its effectiveness was demonstrated by Monien [14, 15], who used it to compute Belyi maps for genus-zero noncongruence subgroups of large index and degree of the number field. Similar approaches using approximations of modular forms as initial values for Newton iterations have been used in [12, 40, 77, 78].

5.1.1 Finding Initial Values for Newton's Method

Throughout this work, we have computed the initial values for Newton's method using the algorithm that is described in chapter 4. The Fourier expansion of the Hauptmodul at infinity can be normalized to be of the form $q_N^{-1} + 0 + a_1q_N + a_2q_N^2 + \ldots$. The values of the Hauptmodul at the other cusps are finite which means that their expansions are of the form $a_0 + a_1q_{N_c} + \ldots$.

Remark 5.1.3. It is important to note that the q_N^{-1} terms form the right-hand side of the linear system of equations and therefore do not enter $\tilde{V}$. This means that the largest entries for each column of $\tilde{V}$ are still on the diagonal and therefore the mixed-precision iterative solving techniques of chapter 4 can also be used to compute j_G.

For the examples considered in this work, it is sufficient to numerically compute the Fourier expansion of the Hauptmodul to 50 digits of precision (although computations at lower precision would probably have worked as well). The evaluations of the Hauptmodul at the elliptic points can then be computed by evaluating the Hauptmodul at $\gamma_i(i)$ and $\gamma_i(\rho)$ where γ_i denotes the coset representative of the corresponding coset (it is preferable to choose the cusp expansion with the fastest convergence for the evaluation at these points in order to maximize the precision). The values at the cusps outside infinity are simply given by the leading coefficients in the cusp expansions.

5.1.2 Applying Newton's Method

Once the initial values have been obtained the multivariate Newton method can be used to improve the accuracy of these values. For simplicity, we will use $x = j_G$ in this section. We also use $[x^n]P$ to denote the coefficient of x^n in P. Then the Jacobian of the system of polynomial equations is given by a $(\mu+1) \times (\mu+1)$ matrix (where μ is the index of G), which has the form

$$J(P) = \begin{pmatrix} \frac{\partial}{\partial A_2}[x^0]P & \frac{\partial}{\partial B_2}[x^0]P & \cdots \\ \frac{\partial}{\partial A_2}[x^1]P & \frac{\partial}{\partial B_2}[x^1]P & \cdots \\ \vdots & \vdots & \vdots \\ \frac{\partial}{\partial A_2}[x^\mu]P & \frac{\partial}{\partial B_2}[x^\mu]P & \cdots \end{pmatrix}. \tag{5.10}$$

Let $X^{[m]} \in \mathbb{C}^\mu$ be the vector containing the numerical approximations of the unknowns $A_2, B_2, \ldots$ at the m-th iteration. Then we can use the update steps

$$X^{[m+1]} = X^{[m]} - [J(P(X^{[m]}))]^{-1} P(X^{[m]}), \tag{5.11}$$

to iteratively increase the precision of the approximations of X. As is standard with Newton's method, this procedure achieves quadratic convergence.

We note that, from a numerical point of view, it is preferable to perform the update steps by solving the linear system of equations

$$J(P(X^{[m]})) \cdot d^{[m]} = P(X^{[m]}), \tag{5.12}$$

instead of computing the matrix inverse of the Jacobian (see the discussion in section 4.3.2). Analogous to iterative refinement, the update steps are then given by

$$X^{[m+1]} = X^{[m]} + d^{[m]}. \tag{5.13}$$

We used Arb's LU decomposition to solve eq. (5.12) (i.e., a direct solving technique). For large index examples it may be preferable to perform this solving iteratively, for example by using preconditioned GMRES (see section 4.3.2).

As an additional implementation detail, we note that instead of computing the entries of the Jacobian matrix by symbolic computation of the partial derivatives and evaluating them by plugging in the corresponding approximations of the variables, it is instead preferable to compute the columns of the Jacobian by univariate polynomial multiplication, which was used by Monien in [14, 15]. To illustrate, suppose that P is of the form

$$P = (a_0 + a_1 x + a_2 x^2 + \ldots)^{k_a} \cdot (b_0 + b_1 x + b_2 x^2 + \ldots)^{k_b} \cdot \ldots + \ldots, \tag{5.14}$$

then

$$\frac{\partial}{\partial a_i} P = k_a x^i (a_0 + a_1 x + a_2 x^2 + \dots)^{k_a - 1} \cdot (b_0 + b_1 x + b_2 x^2 + \dots)^{k_b} . \tag{5.15}$$

Constructing this polynomial by using multiplications of univariate polynomials over $\mathbb{C}$ (for which we used Arb's polynomial implementation) then yields in a polynomial whose coefficients correspond to a column of J, since

$$\frac{\partial}{\partial a_i} [x^j] P = [x^j] \left(\frac{\partial}{\partial a_i} P \right) . \tag{5.16}$$

By applying this procedure to all unknowns (and possibly reusing terms for optimization), all entries of J can be efficiently assembled.

5.1.3 Identifying the Belyi Map

Once the coefficients of the Belyi map have been computed with sufficient precision, the LLL algorithm can be used to identify K and u.

Example 5.1.3. Continuing the example of this section, we find that the Belyi map is given by

$$\begin{aligned} R(x) &= \frac{(x^2 + 444ux - 148284u^2)^3 (x + 516u)}{(x + 462u)^4} , \\ &= 1728 + \frac{(x - 996u)(x^3 + 1422ux^2 + 822204u^2 x + 185029704u^3)^2}{(x + 462u)^4} , \end{aligned} \tag{5.17}$$

where $u = (2/823543)^{1/3}$ which means that $K = \mathbb{Q}$.

We can verify that the result of the Belyi map is correct by checking that eq. (5.9) holds for the recognized polynomials.

5.1.4 The Number Field L

To perform closed-form arithmetic over the number field of the Belyi map (or general modular forms of noncongruence subgroups), we introduce a new number field $L = \mathbb{Q}(w)$. If the cusp width at infinity is one, then $L = K$. Otherwise we choose L as the number field of u (note that we do not reduce this number field with `polredabs` here). The advantage of this choice of L is that one can efficiently convert elements of L into *u-v-factored* expressions (and vice versa). To do this, we compute $v(w)$ (i.e., the generator of K written in terms of the generator of L) using the LLL algorithm. Once this is found, converting elements from K to L is simply a substitution of $v(w)$. Similarly, converting elements from L to K can be done by substituting

$$w^N = u_{\text{interior}}(v) , \tag{5.18}$$

where $u_{\text{interior}}(v)$ denotes the term whose N-th root is being computed (i.e., $u = \left(u_{\text{interior}}(v)\right)^{1/N}$).

Example 5.1.4 (Conversions between K and L)**.** Let G be the subgroup generated by $\sigma_S = (1)(2\,4)(3\,7)(5)(6\,8)$, $\sigma_R = (1\,7\,4)(2\,8\,5)(3)(6)$ and $\sigma_T = (1\,7\,3\,4\,8\,6\,5\,2)$. Then $K = \mathbb{Q}(v)$ with

$$v^2 - 2 = 0 , \tag{5.19}$$

with embedding $v = -1.4142...$ and

$$u = (-99376/823543v - 140492/823543)^{1/8} . \tag{5.20}$$

We find $L = \mathbb{Q}(w)$ with

$$-823543w^{16} - 280984w^{8} + 16 = 0 , \tag{5.21}$$

with embedding $w = -0.2084... - 0.2084...i$, which leads to

$$v(w) = -823543/99376w^{8} - 35123/24844 . \tag{5.22}$$

Then the following expressions are equivalent

$$(-28v + 56)u^{2} \Longleftrightarrow 5764801/24844w^{10} + 593677/6211w^{2} . \tag{5.23}$$

5.2 Computing Fourier Expansions of the Hauptmodul from the Belyi Map

The result of the Belyi map can be used to explicitly compute Fourier expansions of the Hauptmodul.

5.2.1 Computing Fourier Expansions at Infinity

We have seen that

$$j = R(x) = \frac{p_3(x)}{p_c(x)} , \tag{5.24}$$

which we have to solve for $x = j_G$. To do this, we work with the reciprocal

$$\frac{1}{j} = \frac{1}{R(x)} = \frac{p_c(x)}{p_3(x)} =: \frac{1}{R(1/\tilde{x})} = \frac{p_c(1/\tilde{x})}{p_3(1/\tilde{x})} , \tag{5.25}$$

where we set $\tilde{x} := 1/x$. Expanding $1/R(1/\tilde{x})$ as a power series in $\tilde{x}$ results in

$$\sqrt[N]{\frac{1}{j}} = \sqrt[N]{\frac{1}{R(1/\tilde{x})}} =: s(\tilde{x}) , \tag{5.26}$$

where N is the width of the cusp at infinity and the roots are the roots of the power series. The power series $s(\tilde{x})$ has valuation one, so we can compute the reversion

$$\tilde{x} = s^{-1}(\sqrt[N]{1/j}) , \tag{5.27}$$

to get

$$x = 1/s^{-1}(\sqrt[N]{1/j}) . \tag{5.28}$$

Substituting the q-expansion of $\sqrt[N]{1/j}$ (which is a power series in q_N) then gives the q-expansion of j_G at infinity.

5.2.2 Computing Fourier Expansions at other Cusps

To compute the Fourier expansion at a cusp $\neq i\infty$, we perform the transformation

$$x \mapsto x + j_G(c_i) := \tilde{x}\,, \tag{5.29}$$

where $j_G(c_i)$ denotes the evaluation at the cusp. Then

$$\sqrt[N]{\frac{1}{j}} = \sqrt[N]{\frac{1}{R(\tilde{x})}} =: s(\tilde{x})\,, \tag{5.30}$$

where N is the width of the considered cusp (not at infinity) and

$$x = j_G(c_i) + s^{-1}(\sqrt[N]{1/j})\,. \tag{5.31}$$

5.2.3 Computing Fourier Expansions over L

Once the Belyi map is explicitly recognized over L, the expansions at infinity can be computed by performing the arithmetic of section 5.2.1 over L. For this, we use the generic routines provided by Sage [35]. The advantage of this approach is that the Fourier coefficients of the Hauptmodul are rigorous. Note that expansions of cusps outside infinity cannot be computed in general over L, since they are defined over number fields $\mathbb{Q}(v^{1/N_c})\mathbb{Q}(w)$, where N_c denotes the cusp width of the considered cusp outside infinity. The disadvantage of this approach is that it can be slow, because arithmetic over L can be slow, and because Sage's generic series reversion routines do not seem to use asymptotically fast algorithms.

5.2.4 Computing Fourier Expansions over $\mathbb{C}$

To compute Fourier coefficients of the Hauptmodul over $\mathbb{C}$ (more precisely, using arbitrary precision arithmetic), we use Arb [54] to perform the computations of sections 5.2.1 and 5.2.2. The bottleneck of these calculations is the reversion of the power series. We found that series reversion is significantly faster in Arb than in Sage [35] or PARI [36]. Arb has implemented the algorithm of [79], which reduces the asymptotic complexity from $O(N^3)$ to $O(N^{1/2}M(N) + N^2)$, where $M(N)$ is the complexity of the polynomial multiplication. Arb also provides implementations of the fast power series composition algorithms from [80], which we use for the substitutions.

We note, however, that the approach of sections 5.2.1 and 5.2.2 can be very ill-conditioned which means that one may have to use a higher working precision than the target precision. This seems to be caused by the fact that the reversed series s^{-1} can have very large coefficients, making the substitution ill-conditioned. We are unaware of any transformation that improves the conditioning, so the best we could come up with is an approach where we choose the working precision *sufficiently large* in order to overcome the ill-conditioning. We first compute s^{-1} to low precision (typically to 64 bits, but not in double precision because the exponents might over/underflow). The size of the resulting coefficients gives an estimate of the required precision. If the computation fails at the estimated precision, we try again with higher precision. The interval arithmetic of Arb is very useful for this, since it shows whether the working precision was high enough. While this strategy obviously always leads to correct

Digit Precision	Series Reversion	Mixed Precision IR
200	4.63s	**4.28s**
400	**22.3s**	38.1s
600	**59s**	150s

Table 5.1: Benchmarks for the numerical computation of j_G at all cusps for the noncongruence subgroup with signature $(7, 0, 2, 1, 1)$ generated by $\sigma_S = (1\,6)(2)(3\,4)(5\,7)$ and $\sigma_R = (1\,7\,6)(2\,3\,5)(4)$. The expansion order is chosen to achieve convergence up to the specified digit precision. For the series reversion approach the expansion orders depend on the cusp width of the considered cusp. The benchmarks were run on a `Intel i7 4770k @ 3.50GHz` CPU and run on a single thread.

Digit Precision	Series Reversion	Mixed Precision IR
100	2.74s	**1.97s**
200	**9.24s**	11.0s
400	**56s**	109s

Table 5.2: Benchmarks for the numerical computation of j_G for the noncongruence subgroup with signature $(17, 0, 3, 1, 2)$ generated by $\sigma_S = (1\,8)(2\,17)(3\,9)(4\,5)(6)(7\,12)(10\,16)(11\,13)(14\,15)$ and $\sigma_R = (1\,9\,4)(2\,12\,8)(3\,10\,17)(5)(6\,7\,13)(11\,14\,16)(15)$. For further remarks on the benchmarks, see Table 5.1.

results, it is not very elegant, and it would be useful to find a way to rewrite the problem so that all computations can be done at the target precision.

5.2.5 Power Series Reversion vs. Hejhal's Method

It is interesting to examine how the performance of the approach of section 5.2.4 compares to that of the mixed precision IR algorithm from chapter 4. Benchmarks for this can be found in tables 5.1 and 5.2. From these benchmarks, it can be seen that computing the Hauptmodul from the Belyi map is typically faster than using mixed precision IR. An additional benefit is that the results have rigorous error bounds.

5.3 Constructing Modular Forms and Cusp Forms from the Hauptmodul

In the previous section, we saw how the Fourier expansion of the Hauptmodul can be computed from the Belyi map. In this section we will discuss how to construct complete bases of S_k and M_k from this result.

5.3.1 Constructing Modular Functions that are Holomorphic at all Cusps Except Infinity

By theorem 2.8.2, every modular function on G that is holomorphic outside infinity can be written as a polynomial in the Hauptmodul j_G. Since j_G is a modular function (i.e., weight zero form), its derivative $j'_G(\tau) := \frac{1}{2\pi i}\frac{\partial}{\partial\tau} j_G(\tau)$ is a (weakly holomorphic) modular form of weight two. Higher weight forms can be constructed by computing powers of $j'_G(\tau)$, and the monomial $(j'_G(\tau))^{k/2}$ is therefore of weight k. If f is a holomorphic modular form of weight k, then $f(\tau)/(j_G'(\tau))^{k/2}$ is a (meromorphic) modular function with poles at the zeros of $j_G'(\tau)^{k/2}$, which are located at the elliptic points and cusps other than infinity. To make this modular function holomorphic outside infinity we cancel its poles by multiplying it with the polynomial

$$B(j_G(\tau)) = B_e(j_G(\tau)) \cdot B_c(j_G(\tau)), \tag{5.32}$$

which is designed to cancel out all the poles up to the correct order. Since j_G is a modular function on G, multiplying a modular form by a polynomial in j_G does not destroy the modularity.

Note that $j_G'(\tau)$ has zeros of order one at the cusps that are not infinity. Therefore, we can use

$$B_c(j_G(\tau)) = \prod_{c \neq i\infty} (j_G(\tau) - j_G(c))^{\alpha_c}, \tag{5.33}$$

with

$$\alpha_c = k/2. \tag{5.34}$$

At the elliptic points, $j_G'(\tau)$ has zeros of order $n_{e_i} - 1$, where n_{e_i} denotes the order of the elliptic point which is either 2 or 3. From this, we construct

$$B_e(j_G(\tau)) = \prod_e (j_G(\tau) - j_G(e))^{\beta_e}, \tag{5.35}$$

with

$$\beta_e = \left\lfloor \frac{k(n_e - 1)}{2n_e} \right\rfloor. \tag{5.36}$$

(Note that we have to divide by the order of the elliptic point since $(j_G(\tau) - j_G(e))$ has a zero of order n_e, see for example [38, pp. 227–228].) By construction, $f(\tau)/(j_G'(\tau))^{k/2} \cdot B(j_G(\tau))$ is a modular function, which is holomorphic outside infinity, and thus by theorem 2.8.2

$$f(\tau)/(j_G'(\tau))^{k/2} \cdot B(j_G(\tau)) = P(j_G(\tau)). \tag{5.37}$$

5.3.2 Prescribing Cusp Valuations to Construct Bases of Modular Forms

In the previous section we saw how to construct modular functions that are holomorphic outside infinity. We now use eq. (5.37) to construct modular forms with prescribed valuations at the cusps that can be used to construct bases of S_k and M_k. These constructed forms have valuations at the cusps that are equivalent to those of a Victor Miller basis and are therefore linearly independent. From eq. (5.37) we get that

$$f(\tau) = (j_G'(\tau))^{k/2} \cdot \frac{P(j_G(\tau))}{B(j_G(\tau))}. \tag{5.38}$$

To get a basis of forms of M_k, the i-th form f_i should have valuation i at infinity, where $i = 0, 1, \ldots$. Note that $j_G(\tau)$ and $j_G'(\tau)$ both have a pole of order 1 at infinity (i.e., valuation -1 with respect to q_N). So we get the desired behavior at infinity by choosing $P_i(j_G(\tau))$ as a monomial

$$P_i(j_G(\tau)) = j_G(\tau)^{\deg(B) - k/2 - i}. \tag{5.39}$$

The construction of cusp forms $f_i \in S_k$ works similarly. In this case, f_i should have valuation 1 at all cusps outside infinity and valuation $i+1$ at infinity. We hence get

$$\deg(P_i) = \deg(B) - k/2 - i - 1\,. \tag{5.40}$$

To force vanishing at the cusps outside infinity, we simply need to multiply by the factors $\big(j_G(\tau) - j_G(c)\big)$. Let $n(c)$ be the number of cusps of G. So we get

$$P_i(j_G(\tau)) = \prod_{c \neq i\infty} \big(j_G(\tau) - j_G(c)\big) \cdot j_G(\tau)^{\deg(B)-k/2-i-1-(n(c)-1)}\,. \tag{5.41}$$

Example 5.3.1 (Constructing cusp form from Hauptmodul)**.** Continuing with the example from this section, let us assume that we want to construct $f_0 \in S_4(G)$. By using the result of section 5.2, we compute the q-expansion of the Hauptmodul

$$j_G(\tau) = q_3^{-1} + 148932u^2 q_3 + 35666932u^3 q_3^2 + 7392301056u^4 q_3^3 + \dots\,. \tag{5.42}$$

The space $S_4(G)$ is one-dimensional and we get

$$f(\tau) = (j_G'(\tau))^2 \cdot \frac{P(j_G(\tau))}{B(j_G(\tau))} \tag{5.43}$$

$$= (j_G'(\tau))^2 \cdot \frac{(j_G(\tau) + 462u)}{(j_G(\tau) + 462u)^2 (j_G(\tau) - 996u)(j_G(\tau) + 516u)}\,, \tag{5.44}$$

which results in the expansion

$$f(\tau) = q_3 + 18uq_3^2 - 8640u^2 q_3^3 - 1823860u^3 q_3^4 + \dots\,. \tag{5.45}$$

The approach presented in this section can be used to explicitly compute modular and cusp forms over L, which means that the results are rigorous. An additional advantage from a performance point of view is that once the Fourier expansion of the Hauptmodul has been computed, the remaining forms can be obtained without additional expensive solving or series reversion. We note, however, that the division of power series may be ill-conditioned when working over $\mathbb{C}$ for the problems involved. For this reason, it is useful to use Arb's interval arithmetic to ensure that the coefficients have been computed with sufficient accuracy. Once a basis of forms has been constructed, linear algebra can be used to transform the basis into reduced row echelon form.

Remark 5.3.1. It would be interesting to investigate the practicality and effectiveness of an approach where higher genus Newton methods (see for example [40, 78]) are used to compute the curve from which the modular forms can then be constructed.

CHAPTER 6

Numerical Computation of Noncongruence Eisenstein Series

In this chapter we present a new approach to obtain high precision numerical approximations of the Fourier expansions of Eisenstein series on noncongruence subgroups. We exploit the fact that we have developed efficient methods for computing modular forms and cusp forms in chapters 4 and 5. By theorem 2.6.1, the Eisenstein series E_k can then be computed by computing the orthogonal complement of S_k in M_k with respect to the Petersson scalar product. This provides a very efficient way to compute Eisenstein series on noncongruence subgroups. These are particularly interesting to study because very few results are known about them. For this reason they were declared as one of the main goals for further investigation at the AIM workshop on noncongruence modular forms and modularity [81]. The only major result was obtained by Scholl [49, 82], who showed that Eisenstein series of weight two on noncongruence subgroups can be non-algebraic (unlike their congruence counterparts). However, Scholl's result could not yet be extended to Eisenstein series of higher weight. The computational perspective is arguably even worse: the only example of a computation of a non-trivial noncongruence Eisenstein series was recently given by Fiori and Franc [83], who computed the first coefficients of an Eisenstein series on a index 7 noncongruence subgroups to a few digits precision, despite spending over a month of CPU time. Their method is based on the evaluation of Dirichlet series, which converge however very slowly and is therefore not suitable for more involved computations.

6.1 Numerical Evaluation of Petersson Scalar Products

A key component of the approach described above is an efficient algorithm for computing Petersson scalar products, so in the following we will compare three different approaches.

6.1.1 Direct Numerical Integration

Recall from section 2.6 that the Petersson product on a subgroup $G \leqslant \Gamma$ with index μ is defined by the integral

$$\langle f, g \rangle_G := \frac{1}{\mu} \int_{G \backslash \mathcal{H}} f(\tau)\overline{g(\tau)} y^k d\tau \,. \tag{6.1}$$

One can therefore try to compute the double integral

$$\int_{-1/2}^{1/2} \left(\int_{\sqrt{1-x^2}}^{i\infty} f(\tau)\overline{g(\tau)} y^{k-2} dy \right) dx \,, \tag{6.2}$$

for each coset. However, even when using efficient doubly exponential integration methods, this approach requires $O(N^2 \log(N)^2)$ evaluations of modular forms and is quite slow in practice, see Cohen [16] for more details and benchmarks.

6.1.2 Nelson-Collins Formula

Recently, faster and more sophisticated approaches to the numerical computation of Petersson products have been developed. One of these alternative approaches is the use of a formula by Nelson [84] (see also Collins [85] for the case of multiple cusps). Using the cusp-normalizers as in the previous sections, Nelson's formula can be written as

$$\langle f, g \rangle_G = \frac{4\,(8\pi)^{-(k-1)}}{\mu} \sum_c \sum_{n=1}^{N} \frac{a_n^{(c)} \overline{b_n^{(c)}}}{n^{k-1}} W_k\left(4\pi\sqrt{n/N_c}\right) \,, \tag{6.3}$$

where N_c is the cusp-width of cusp representative c and

$$W_k(x) = \sum_{m\geq 1} (mx)^{k-1} \left(mxK_{k-2}(mx) - K_{k-1}(mx) \right) \,, \tag{6.4}$$

where $K_\nu(x)$ denotes the K-Bessel function. The Bessel functions can be evaluated using the implementation of Arb [54], although if performance is a concern, it is preferable to evaluate $W_k(x)$ as described in [86].

The elegance of this approach is that it can be used for very general types of modular forms, such as Maass cusp forms, and is straightforward to implement. A sufficient condition for the application of eq. (6.3) (assuming integer $k \geq 1$) is that $f \cdot g$ vanishes at all cusps. This means that at least one of f and g must be a cusp form. The $n = 0$ term can be omitted in any case for the examples considered in this work, since $a_0^{(c)} \overline{b_0^{(c)}} = 0$ (see the discussion in [86]).

The disadvantage of this approach is the convergence rate. Note that

$$K_\nu(z) \to \sqrt{\frac{\pi}{2z}} \exp(-z) \,, \tag{6.5}$$

for large z, which means that $W_k(x)$ decays exponentially. The arguments of eq. (6.3) are only proportional to $\sqrt{n/N_c}$, which means that the series converges like $\exp\left(-\sqrt{n/N_c}\right)$ and like

$$M_{\text{trunc}} = \left(\frac{\ln(10)D}{4\pi} \right)^2 \cdot N_c \,, \tag{6.6}$$

coefficients are needed to achieve convergence up to D digits of precision. This expansion order is higher than the expansion order M_0 needed to achieve convergence within the fundamental domain, since q-expansions of modular forms converge like $\exp\left(-n/N_C\right)$. The set of coefficients obtained by the method of chapter 4 is therefore a priori insufficient to compute the Petersson product with D digits of precision. One could try to compute higher order coefficients as well, e.g. by using eq. (3.19), but this would be computationally expensive because the number of coefficients required is considerable.

6.1.3 Cohen-Haberland Formula

An alternative method has recently been presented by Cohen [6, 16, 86] and is based on the Haberland formula [87], which transforms the double integral of eq. (6.2) into *simpler* integrals that can be evaluated more efficiently (a generalization of Haberland's formula to finite index subgroups can be found in [6, Chapter 12.5]). This approach is restricted to holomorphic modular forms of weight $k \geq 2$ and is therefore less general than the method of Nelson and Collins (see [86] for a comparison between the use cases of both methods), but sufficient for the examples considered in this work.

Remark 6.1.1. Before giving the formulas, we should note that Cohen does not use width-absorbing cusp normalizers (see definition 3.3.1). Therefore, to follow Cohen's notation, we multiply the Fourier expansions obtained using the numerical methods of chapters 4 and 5 by factors $N_c^{-k/2}$ to obtain forms $\tilde{f}(\tau)$.

We follow Cohen [86] to define

$$I_n(A, B, \tilde{f}) := \int_A^B \tau^n \tilde{f}(\tau) d\tau \,, \tag{6.7}$$

and

$$G_j(A, B; C, D) := \int_A^B \int_C^D \tilde{f}_j(\tau) \overline{\tilde{g}_j(\tau_2)} (\tau - \overline{\tau_2})^{k-2} d\tau \overline{d\tau_2} \,, \tag{6.8}$$

where $A, B, C, D \in \overline{\mathcal{H}}$ and $\tilde{f}_j, \tilde{g}_j$ denote expansions at the j-th coset representative. Let j denote a coset in the cycle of cusp c. Let $\gamma_j \in \Gamma$ denote the coset representative and $A_c \in \Gamma$ denote the (non width absorbing) cusp normalizer. Then there exists an integer $m \in \mathbb{Z}$ such that

$$\gamma_j = A_c T^m \,, \tag{6.9}$$

where T is defined as in eq. (2.9). Then

$$\tilde{f}_j := \left(\tilde{f}|_k \gamma_j\right)(\tau) = \left(\tilde{f}|_k A_c T^m\right)(\tau) = \sum_{n=0} a_n^{(c)} \zeta_{N_c}^{nm} q_{N_c}^n \,, \tag{6.10}$$

where $\zeta_{N_c} := \exp\left(2\pi i / N_c\right)$. Thus, coset expansions can be efficiently obtained by multiplying the cusp expansions by roots of unity.

We now assume, without loss of generality, that f is a cusp form (this does not lose generality, since f and g can be exchanged by eq. (2.38)). Applying the formulas of [86, Section 3.3] we get

$$\mu (2i)^{k-1} \langle \tilde{f}, \tilde{g} \rangle_G = \sum_{1 \leq j \leq \mu} G_j(\rho + 1, i\infty; i, i+1) \,, \tag{6.11}$$

$$= \sum_{1 \leq j \leq \mu} \sum_{0 \leq n \leq k-2} (-1)^n \binom{k-2}{n} \int_{\rho+1}^{i\infty} \tau^{k-2-n} \tilde{f}_j(\tau) d\tau \overline{\int_i^{i+1} \tau^n \tilde{g}_j(\tau) d\tau} \,, \tag{6.12}$$

$$= \sum_{1\le j\le \mu} \sum_{0\le n\le k-2} (-1)^n \binom{k-2}{n} I_{k-2-n}(\rho+1, i\infty, \tilde{f}_j) \overline{I_n(i, i+1, \tilde{g}_j)} \,. \tag{6.13}$$

To compute the resulting partial periods we follow Cohen [86, Section 3.4]:

1. $\int_{\rho+1}^{i\infty} \tau^{k-2-n} \tilde{f}_j(\tau) d\tau = \sum_{l=1} a_l^{(j)} \int_{\rho+1}^{i\infty} \tau^{k-2-n} \exp\big(2\pi i l\tau/N_c\big) d\tau$:
   We evaluate these integrals using
   $$\int_a^{i\infty} \exp(2\pi i m\tau)\tau^n d\tau = \exp(2\pi i m a) \sum_{s=0}^{n} \left(\frac{(-1)^s a^{n-s}}{(2\pi i m)^{s+1}} \prod_{j=n+1-s}^{n} j \right), \tag{6.14}$$
   where $a \in \mathcal{H}$, see Stein [34, Lemma 10.4].
2. $\int_i^{i+1} \tau^n \tilde{g}_j(\tau) d\tau = \sum_{l=0} a_l^{(j)} \int_i^{i+1} \tau^n \exp\big(2\pi i l\tau/N_c\big) d\tau$:
   For the case $l > 0$ we can evaluate these integrals by using
   $$\int_i^{i+1} \tau^n \exp\big(2\pi i l\tau/N_c\big) d\tau = \int_i^{i\infty} \tau^n \exp\big(2\pi i l\tau/N_c\big) d\tau - \int_{i+1}^{i\infty} \tau^n \exp\big(2\pi i l\tau/N_c\big) d\tau \,, \tag{6.15}$$
   and use eq. (6.14) for the remaining integrals. For the case $l = 0$ we use classical polynomial integration instead.

From a technical perspective we remark that the evaluation of the period integrals has a large overhead which means that many terms can be cached and reused. Despite this we have not implemented any technical optimizations because the evaluation of Petersson products typically takes negligible amount of CPU time compared to the computation of Fourier expansions.

6.1.4 Concluding Remarks

We have seen that the numerical evaluation of the Petersson product requires numerical approximations of the Fourier expansions at all cusps. These are in general non-trivial to obtain, even for congruence subgroups (see for example Cohen [86] and Collins [85]). In our case however, the algorithms of chapters 4 and 5 provide them as a *byproduct*.

For the examples considered in this work, the Cohen-Haberland formula is the preferred approach due to its superior convergence rate and performance. This is particularly useful when Hejhal's method is used to compute the Fourier expansions, since this result can then be used to compute the Petersson product (usually) without loss of precision. However, we also use the Nelson-Collins formula (at low precision) as an additional independent check that the computed products are correct.

6.2 Numerical Computation of Fourier Expansions of Noncongruence Eisenstein Series

We can now make use of the numerical evaluation of Petersson products to compute the orthogonal complement of S_k in M_k which gives a numerical approximation of Eisenstein series. Since $E_k \subset M_k$

we can make the ansatz

$$e_i = \sum_{j=0}^{\dim(M_k)-1} c_{i,j} m_j\,, \tag{6.16}$$

where $e_i \in E_k$ denotes a basis of E_k, $m_j \in M_k$ denotes a basis of M_k and $c_{i,j} \in \mathbb{C}$. We call $c_{i,j}$ *Eisenstein basis factors*. Let s_i denote the basis forms of S_k. To obtain the Eisenstein basis factors, we compute the kernel elements of the matrix

$$\begin{pmatrix} \langle s_0, m_0\rangle & \langle s_0, m_1\rangle & \dots & \langle s_0, m_{\dim(M_k)-1}\rangle \\ \langle s_1, m_0\rangle & \langle s_1, m_1\rangle & \dots & \langle s_1, m_{\dim(M_k)-1}\rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle s_{\dim(S_k)-1}, m_0\rangle & \langle s_{\dim(S_k)-1}, m_1\rangle & \dots & \langle s_{\dim(S_k)-1}, m_{\dim(M_k)-1}\rangle \end{pmatrix}, \tag{6.17}$$

using linear algebra.

Example 6.2.1 (Computation of Eisenstein series)**.** Let us illustrate the above procedure for the example $E_4(\Gamma_0(11))$ (working with a congruence subgroup leads to simpler expressions, but it should be obvious that the described approach also works for noncongruence subgroups). We get $\dim(M_4(\Gamma_0(11))) = 4$ and $\dim(S_4(\Gamma_0(11))) = 2$. So we get

$$e_i = c_{i,0} \cdot m_0 + c_{i,1} \cdot m_1 + c_{i,2} \cdot m_2 + c_{i,3} \cdot m_3\,, \tag{6.18}$$

and

$$\langle s_0, e_i\rangle = c_{i,0}\langle s_0, m_0\rangle + c_{i,1}\langle s_0, m_1\rangle + c_{i,2}\langle s_0, m_2\rangle + c_{i,3}\langle s_0, m_3\rangle \overset{!}{=} 0\,, \tag{6.19}$$

$$\langle s_1, e_i\rangle = c_{i,0}\langle s_1, m_0\rangle + c_{i,1}\langle s_1, m_1\rangle + c_{i,2}\langle s_1, m_2\rangle + c_{i,3}\langle s_1, m_3\rangle \overset{!}{=} 0\,, \tag{6.20}$$

which gives 2 equations in 4 unknowns for $i = 0, 1$. We now impose the (linearly independent) normalizations ($c_{0,0} = 1$, $c_{0,1} = 0$) and ($c_{1,0} = 0$, $c_{1,1} = 1$). (Note that by this normalization, the constructed Eisenstein series are in reduced row echelon form.) The case $i = 0$ now therefore corresponds to

$$c_{0,2}\langle s_0, m_2\rangle + c_{0,3}\langle s_0, m_3\rangle \overset{!}{=} -\langle s_0, m_0\rangle\,, \tag{6.21}$$

$$c_{0,2}\langle s_1, m_2\rangle + c_{0,3}\langle s_1, m_3\rangle \overset{!}{=} -\langle s_1, m_0\rangle\,. \tag{6.22}$$

An analogous linear system of equations can be obtained for the case $i = 1$. Evaluating the Petersson products and solving the linear system of equations results in

$$c_{0,0} = 1,\ c_{0,1} = 0,\ c_{0,2} = 0,\ c_{0,3} = 0\,, \tag{6.23}$$

and

$$c_{0,0} = 0,\ c_{0,1} = 1,\ c_{0,2} = 9,\ c_{0,3} = 28\,. \tag{6.24}$$

Substituting these results and the computed expansions for m_i into eq. (6.16) we get

$$e_0 = 1 + 240q^{11} + 2160q^{22} + 6720q^{33} + \dots\,, \tag{6.25}$$

$$e_1 = q + 9q^2 + 28q^3 + 73q^4 + \dots . \tag{6.26}$$

6.2.1 Canonical Normalization

The above procedure produces a basis for E_k in reduced row echelon form. A more natural normalization may be given by the *canonical normalization*, for which each basis element has a leading order coefficient that is 1 at one of the cusps and 0 at the others (except in the weight two case, where we must break the symmetry and leave the normalization at one of the cusps undefined). We can transform to this basis by using linear algebra.

6.2.2 Results

We have found this to be a very efficient approach to computing Eisenstein series. For example, reproducing the computation of [83] takes less than 0.2s of CPU time, and extending the computation from 19 digits precision to 1500 digits takes 13min and 20s, from which only about 9s were used to compute the Petersson products. We therefore used this algorithm to compute the Eisenstein series of weight $k \leq 6$ for 221 noncongruence subgroups with index $\mu \leq 17$, of which 200 where of genus zero and 21 where of genus one, to 1500 digits precision (see chapter 7). To our surprise, we did not find any non-trivial algebraic Eisenstein series in this data. It seems that the Petersson products introduce a non-algebraicity, that might be overcome by choosing a different normalization. We leave this for future work.

Remark 6.2.1. We say that an Eisenstein series on a noncongruence subgroup G is non-trivial if it is not an oldform and if $M_k(G) \neq E_k(G)$ (which happens when $\dim(S_k(G)) \neq 0$).

6.3 Example of a Non-Trivial Algebraic Eisenstein Series

One noncongruence subgroup for which it is known by Scholl's theorem [82, Theorem 3] that the weight two Eisenstein series are algebraic is given by the subgroup H_5 [81]

$$H_5 := \left\{ \gamma \in \Gamma \quad \middle| \quad \left(\frac{\eta^{12}(11\tau)}{\eta^{12}(\tau)}\right)^{1/5} |_\gamma = \left(\frac{\eta^{12}(11\tau)}{\eta^{12}(\tau)}\right)^{1/5} \right\}, \tag{6.27}$$

where $\eta(\tau) = q_{24} \prod_{n\geq 1}(1-q^n)$ denotes the *Dedekind eta function*. (Note that $\eta^{12}(11\tau)/\eta^{12}(\tau)$ is a modular function on $\Gamma_0(11)$ which means that H_5 is a character group of $\Gamma_0(11)$.) The subgroup H_5 can be generated by $\sigma_S = (1\,2)(3\,4)(5\,8)(6\,9)(7\,12)(10\,14)(11\,15)(13\,17)(16\,19)(18\,21)(20\,24)(22\,28)(23\,30)$ $(25\,32)(26\,33)(27\,36)(29\,37)(31\,38)(34\,40)(35\,41)(39\,43)(42\,45)(44\,49)(46\,51)(47\,53)(48\,55)(50\,57)$ $(52\,58)(54\,60)(56\,59)$ and $\sigma_R = (1\,3\,2)(4\,5\,9)(6\,13\,14)(7\,15\,8)(10\,18\,17)(11\,19\,12)(16\,23\,24)(20\,31\,32)$ $(21\,22\,33)(25\,39\,38)(26\,34\,40)(27\,41\,28)(29\,37\,30)(35\,45\,36)(42\,50\,51)(43\,44\,53)(46\,52\,58)(47\,59\,60)$ $(48\,55\,49)(54\,57\,56)$. This means that H_5 has signature $(60, 1, 10, 0, 0)$, five cusps of width 1 and five cusps of width 11. We get that $\dim(S_2(H_5)) = 1$ and the unique cusp form is given by the $\Gamma_0(11)$ oldform

$$s_0 = q_1 - 2q_1^2 - q_1^3 + 2q_1^4 + q_1^5 + 2q_1^6 - 2q_1^7 + \dots . \tag{6.28}$$

We have computed a basis for $M_2(H_5)$ to about 500 digits precision and found that

$$\begin{aligned}
m_0 &= 1 + \tfrac{1536887748}{390625} q^9 + \tfrac{-6115630700124}{48828125} q^{11} + \tfrac{94006270544076}{244140625} q^{12} + \tfrac{657558573754632}{1220703125} q^{13} + \dots\,, \\
m_1 &= q + \tfrac{607665972}{390625} q^9 + \tfrac{-2302295704656}{48828125} q^{11} + \tfrac{34833599928379}{244140625} q^{12} + \tfrac{239116450435908}{1220703125} q^{13} + \dots\,, \\
m_2 &= q^2 + \tfrac{198762176}{390625} q^9 + \tfrac{-764199359708}{48828125} q^{11} + \tfrac{11638490025852}{244140625} q^{12} + \tfrac{80950723119894}{1220703125} q^{13} + \dots\,, \\
m_3 &= q^3 + \tfrac{-15815546}{78125} q^9 + \tfrac{72859283252}{9765625} q^{11} + \tfrac{-46278038828}{1953125} q^{12} + \tfrac{-8234433934296}{244140625} q^{13} + \dots\,, \\
m_4 &= q^4 + \tfrac{-3679062}{15625} q^9 + \tfrac{2940544281}{390625} q^{11} + \tfrac{-225430190916}{9765625} q^{12} + \tfrac{-1525694815502}{48828125} q^{13} + \dots\,, \\
m_5 &= q^5 + \tfrac{-110956}{625} q^9 + \tfrac{85686617}{15625} q^{11} + \tfrac{-1297214623}{78125} q^{12} + \tfrac{-45883527858}{1953125} q^{13} + \dots\,, \\
m_6 &= q^6 + \tfrac{-9232}{125} q^9 + \tfrac{31877317}{15625} q^{11} + \tfrac{-469169468}{78125} q^{12} + \tfrac{-3234820494}{390625} q^{13} + \dots\,, \\
m_7 &= q^7 + \tfrac{-672}{25} q^9 + \tfrac{2068478}{3125} q^{11} + \tfrac{-5906684}{3125} q^{12} + \tfrac{-221436797}{78125} q^{13} + \dots\,, \\
m_8 &= q^8 + \tfrac{-28}{5} q^9 + \tfrac{10079}{125} q^{11} + \tfrac{-664248}{3125} q^{12} + \tfrac{-3672708}{15625} q^{13} + \dots\,, \\
m_9 &= q^{10} + \tfrac{-38}{5} q^{11} + \tfrac{367}{25} q^{12} + \tfrac{6278}{125} q^{13} + \dots\,,
\end{aligned}$$

up to about 300 terms.

Remark 6.3.1. For $M_2(H_5)$ we could not impose the Victor-Miller normalization. Instead, m_9 has valuation 10 and the remaining forms have a zero at the 10th coefficient.

Remark 6.3.2. We noticed that the linear system used to compute $M_2(H_5)$ is significantly less well conditioned than the other examples considered in this work. We expect this to be due to the relatively high dimensionality of the space, combined with the moderate size of μ. For this reason, we used the preconditioned GMRES solver (see section 4.3.2) instead of the iterative refinement approach (see section 4.3.3).

Computing the Petersson products between the forms, we determine from this that a basis for $E_2(H_5)$ is given by

$$\begin{aligned}
e_0 &= m_0 + \tfrac{-37283794635953844}{2166748046875} m_9\,, \\
e_1 &= m_1 + \tfrac{-2583966146013468}{433349609375} m_9\,, \\
e_2 &= m_2 + \tfrac{-178810564897398}{86669921875} m_9\,, \\
e_3 &= m_3 + \tfrac{40215838981851}{34667968750} m_9\,, \\
e_4 &= m_4 + \tfrac{6786803544201}{6933593750} m_9\,, \\
e_5 &= m_5 + \tfrac{99702868152}{138671875} m_9\,, \\
e_6 &= m_6 + \tfrac{13112388597}{55468750} m_9\,, \\
e_7 &= m_7 + \tfrac{859023267}{11093750} m_9\,, \\
e_8 &= m_8 + \tfrac{11512941}{2218750} m_9\,,
\end{aligned}$$

which results in the Fourier expansions

$$e_0 = 1 + \tfrac{1536887748}{390625} q^9 + \tfrac{-37283794635953844}{2166748046875} q^{10} + \tfrac{59878634576233572}{10833740234375} q^{11} + \tfrac{7174488645571801752}{54168701171875} q^{12} + \dots\,,$$

$$
\begin{aligned}
e_1 &= q + \tfrac{607665972}{390625} q^9 + \tfrac{-2583966146013468}{433349609375} q^{10} + \tfrac{-3973658345598216}{2166748046875} q^{11} + \tfrac{597425421234875369}{10833740234375} q^{12} + \dots, \\
e_2 &= q^2 + \tfrac{198762176}{390625} q^9 + \tfrac{-178810564897398}{86669921875} q^{10} + \tfrac{12532148692624}{433349609375} q^{11} + \tfrac{37668121662091434}{2166748046875} q^{12} + \dots, \\
e_3 &= q^3 + \tfrac{-15815546}{78125} q^9 + \tfrac{40215838981851}{34667968750} q^{10} + \tfrac{-117474801793669}{86669921875} q^{11} + \tfrac{-5776666823585683}{866699218750} q^{12} + \dots, \\
e_4 &= q^4 + \tfrac{-3679062}{15625} q^9 + \tfrac{6786803544201}{6933593750} q^{10} + \tfrac{1537385129556}{17333984375} q^{11} + \tfrac{-1510628988037233}{173339843750} q^{12} + \dots, \\
e_5 &= q^5 + \tfrac{-110956}{625} q^9 + \tfrac{99702868152}{138671875} q^{10} + \tfrac{13634639599}{693359375} q^{11} + \tfrac{-20972946283841}{3466796875} q^{12} + \dots, \\
e_6 &= q^6 + \tfrac{-9232}{125} q^9 + \tfrac{13112388597}{55468750} q^{10} + \tfrac{33775805032}{138671875} q^{11} + \tfrac{-3515511441901}{1386718750} q^{12} + \dots, \\
e_7 &= q^7 + \tfrac{-672}{25} q^9 + \tfrac{859023267}{11093750} q^{10} + \tfrac{2036300177}{27734375} q^{11} + \tfrac{-208956666011}{277343750} q^{12} + \dots, \\
e_8 &= q^8 + \tfrac{-28}{5} q^9 + \tfrac{11512941}{2218750} q^{10} + \tfrac{228509746}{5546875} q^{11} + \tfrac{-7565152653}{55468750} q^{12} + \dots.
\end{aligned}
$$

Alternatively, we could also transform $E_2(H_5)$ into the canonical normalization. This basis can be found in appendix A.

We have also computed $M_4(H_5)$ and from this numerical expressions for $E_4(H_5)$ to about 500 digits precision. Interestingly, the non-trivial Eisenstein series of weight 4 seem to be non-algebraic. This indicates that no non-trivial noncongruence algebraic Eisenstein spaces of weight $k > 2$ are known so far. The data corresponding to forms on the H_5 subgroup can be found at [88].

CHAPTER 7

A Database of Modular Forms on Noncongruence Subgroups

Parts of this chapter were used in the paper [89].

In this chapter we apply the algorithms of chapters 4, 5 and 6 to build a database of modular forms of weight $k \leq 6$ on noncongruence subgroups of index < 18.

7.1 Introduction

The theory of modular forms on noncongruence subgroups is still not well understood. We therefore provide a large number of examples in the hope that these will lead to new conjectures and observations. A database of modular forms on noncongruence subgroups was formulated as one of the goals for future research in the AIM Workshop *Noncongruence modular forms and modularity* [81]. The usefulness of computer-generated data for studying modular forms on noncongruence subgroups has been demonstrated by Atkin and Swinnerton-Dyer [8], who formulated key conjectures based on only a handful of computed examples. Additional computations have recently been made by Fiori and Franc [83], who computed the Hauptmodul and some modular forms for the three noncongruence subgroups of index 7. Somewhat related databases are the database of classical modular forms on congruence subgroups [90], the database of Hilbert modular forms [91] and the database of Belyi maps [78].

7.2 Signatures, Passports and Conjugation

We have defined the signature in definition 2.1.2 to distinguish between different *types* of subgroups. Another distinction can be made by defining the *passport*:

Definition 7.2.1 (Passport)**.** Given a set of subgroups with equal cycle types of $(\sigma_S, \sigma_R, \sigma_T)$, we say that two subgroups belong to the same passport if their *monodromy group* (i.e., the permutation group generated by σ_S and σ_R) is equivalent [13].

Remark 7.2.1. We identify the monodromy group by locating it in the database of transitive groups [92] using GAP [93].

Passports are useful for sorting subgroups because the size of the passport (i.e., the number of elements in a passport) provides an upper bound on the degree of $K(v)$. Furthermore, expressions for subgroups that are in the same Galois orbit are equivalent up to the embedding of $K(v)$ in $\mathbb{C}$.

Example 7.2.1. To illustrate this, consider the first passport of length greater than one which contains the representatives $\sigma_S = (2\,6)(1)(3\,4)(5\,7)$, $\sigma_R = (2\,7\,6)(1\,3\,5)(4)$ and $\sigma_S = (2\,4)(1)(3\,5)(6\,7)$, $\sigma_R = (2\,5\,4)(1\,3\,6)(7)$ for both representatives we also have $\sigma_T = (1\,3\,4\,5\,6\,7)(2)$. The Belyi map for the first representative is given by

$$R = \frac{(x^2 + ((2v - 10)u)x + (184/3v - 164/3)u^2)^3(x + 14u)}{(x + (6v - 16)u)}, \tag{7.1}$$

where $u = (125248356/96889010407v - 199546416/96889010407)^{1/6}$ and $v = 1/2 - \sqrt{3}/2i$ is an embedding of $K(v) = v^2 - v + 1$. Then the Galois conjugate second passport element has the same expression for the Belyi map as in eq. (7.1), but with the *other* embedding of $K(v)$ into $\mathbb{C}$ which is given by $1/2 + \sqrt{3}/2i$ (the same obviously holds true for the Fourier expansions of modular forms). Note that conjugation in Γ amounts to choosing different roots of u.

7.3 Database Structure

7.3.1 Database Labels

For compatibility, we use the same labeling structure as the BelyiDB [78] in the LMFDB. This means that we start with the index of the subgroup, followed by the label of the monodromy group in the transitive group database [92]. Afterwards we display the cycle type of the permutations σ_T, σ_R and σ_S. The last letter is a letter indicating the Galois orbit.

Example 7.3.1. Consider the passport of signature $(15, 1, 2, 1, 0)$ and monodromy group *T104* containing three elements. This passport decomposes into two Galois orbits: The first one with label *15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-a* for which $K = \mathbb{Q}(v)$ where $v^2 - v - 1 = 0$ and the second one with label *15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-b* for which $K = \mathbb{Q}$.

7.3.2 Permutation Triple Normalization

Each database entry is unique up to a normalization of the permutation triple (i.e., conjugation in Γ). We usually normalize σ_T so that the cycle type is sorted in descending order and that the labels are sorted in ascending order. This means that the largest cusp is placed at infinity. While this causes the number field L to be of larger degree (and thus the arithmetic to be slower), the factored expressions in u and v typically involve smaller factors and are therefore easier to read.

Example 7.3.2. Consider the permutation $(1\,2\,3\,4)(5\,6\,7)(8\,9)$ which has a cycle type of $(4, 3, 2)$, sorted in descending order.

There are three exceptions to the above strategy. The first is given for the case when the subgroup G has multiple cusps of equal width. Placing one of these cusps at infinity may cause K to have a larger degree, due to the broken symmetry. In this case we therefore normalize σ_T so that the largest cusp with unique cusp width is placed at infinity. The second exception occurs when G has a cusp of (unique) width 1. In this case, we place the cusp of width 1 at infinity because it is convenient that $K = L$. The third exception occurs for genus one subgroups, where we do not sort the labels of σ_T in ascending order, in case another normalization leads to better precision when computing the elliptic curve (see [89, Section 3]).

7.3.3 Data that has been Computed for each Database Element

For each database element we have computed the curve (either the Belyi map for genus zero subgroups or the elliptic curve for genus one subgroups), as well as Fourier expansions for the Hauptmoduls, cusp forms, and modular forms up to and including weight 6. For genus zero subgroups we achieved this by applying the algorithms of sections 5.2 and 5.3 over L. For higher genera subgroups we have computed numerical approximations of the corresponding Fourier expansions to an accuracy of 1500 digits using the method of section 4.3 and from these the closed-form solutions using the LLL algorithm. Note that for reasons of efficiency it is not necessary to compute all forms *from scratch*. Instead, by section 2.7, we can compute products between cusp forms and modular forms to generate (some of the) higher weight forms. In addition, we have computed numerical approximations of the basis factors that transform M_k into E_k to 1500 digits precision using the algorithm of chapter 6.

7.3.4 A Complete Example

For an explicit example, let us take a look at the database entry *15T61-10.5_3.3.3.3.3_2.2.2.2.2.2.2.1-a*. For the (echelonized) basis forms we use lower case letters.

- G: The subgroup G corresponds to the permutation group generated by

$$\begin{aligned}
\sigma_S &= (1\,15)(2\,12)(3\,7)(4\,9)(5\,13)(6\,10)(8\,14)(11)\,, \\
\sigma_R &= (1\,11\,12)(2\,13\,6)(3\,8\,15)(4\,10\,7)(5\,14\,9)\,, \\
\sigma_T &= (1\,3\,4\,5\,6\,7\,8\,9\,10\,2)(11\,12\,13\,14\,15)\,.
\end{aligned}$$

- Monodromy Group: $(C_3 \times C_3 \times C_3 \times C_3) : (C_2 \times A_5)$.
- K: $v = 1$ (i.e., $K = \mathbb{Q}$).
- Embeddings: Trivial embedding.
- u: $(-3125/14348907)^{1/10}$ with an embedding $-0.409269... - 0.132979... \cdot i$.
- L: $w^8 - 5/27w^6 + 25/729w^4 - 125/19683w^2 + 625/531441$ with $w = -0.409269... - 0.132979... \cdot i$.
- Curve: $y^2 + xy + y = x^3 - x^2 + 7x - 103$.
- Fourier expansions (up to $\sim$ 700 terms):
  - M_2:
    - $m_0 = 1 + 48/5u^2q_{10}^2 - 528/5u^3q_{10}^3 - 2448/25u^4q_{10}^4 - 3096/25u^5q_{10}^5 + ...$
    - $m_1 = q_{10} + uq_{10}^2 - 21/5u^2q_{10}^3 + 101/5u^3q_{10}^4 + 816/25u^4q_{10}^5 + ...$
  - S_2:

    * $s_0 = q_{10} + uq_{10}^2 - 21/5u^2q_{10}^3 + 101/5u^3q_{10}^4 + 816/25u^4q_{10}^5 + \ldots$ (this is the same as m_1 in this case).
- E_2:
    * e_0: We give the Eisenstein basis matrix $[1, 0.356085\ldots + 0.115699\ldots \cdot i]$ (up to 1500 terms), meaning that $e_0 = m_0 + (0.356085\ldots + 0.115699\ldots \cdot i)m_1$. Note that this basis is already in canonical normalization.
- M_4:
    * $m_0 = 1 - 688747536/625u^{10}q_{10}^{10} + \ldots$ (this is an oldform from Γ, namely the weight 4 Eisenstein series for $\mathrm{SL}(2, \mathbb{Z})$).
    * $m_1 = q_{10} - 70111/110u^4q_{10}^5 + \ldots$.
    * $m_2 = q_{10}^2 - 621/22u^3q_{10}^5 + \ldots$.
    * $m_3 = q_{10}^3 + 237/22u^2q_{10}^5 + \ldots$.
    * $m_4 = q_{10}^4 - 115/22uq_{10}^5 + \ldots$.
- S_4:
    * $s_0 = q_{10} - 65u^3q_{10}^4 - 1488/5u^4q_{10}^5 + \ldots$.
    * $s_1 = q_{10}^2 - 27/5u^2q_{10}^4 + \ldots$.
    * $s_2 = q_{10}^3 - uq_{10}^4 + 16u^2q_{10}^5 + \ldots$.
- E_4:
    * $e_0 = m_0$.
    * $e_1 = m_1 + (12.068314\ldots + 3.921233\ldots \cdot i)m_2 + (106.205344\ldots + 77.162699\ldots \cdot i)m_3 + (-61.318023\ldots - 84.397019\ldots \cdot i)m_4$.
    * $e_{0,\mathrm{can}}$: For the canonical normalizations we get $e_{0,\mathrm{can}} = e_0 + (-0.014977\ldots - 0.004866\ldots \cdot i)e_1$.
    * $e_{1,\mathrm{can}} = (0.059909\ldots + 0.019465\ldots \cdot i)e_1$.

(The weight six spaces are also given in the database, but are not listed here.)

7.4 Some Interesting Examples

7.4.1 Largest Degree of $K(v)$

The largest degree of K occurs for the passport *16T1954-12.3.1_3.3.3.3.3.1_2.2.2.2.2.2.2.1.1-a* for which we get $K = \mathbb{Q}(v)$ where $v = 1.531805\ldots + 0.185685\ldots i$ is a root of

$$\begin{aligned}0 = v^{24} - 6v^{23} + 21v^{22} - 60v^{21} + 184v^{20} - 478v^{19} + 651v^{18} - 1220v^{17} + 2230v^{16} + 1226v^{15} \\ - 947v^{14} + 804v^{13} - 7092v^{12} - 6862v^{11} + 3971v^{10} - 15340v^{9} + 7975v^{8} + 36044v^{7} + 7134v^{6} \\ + 14896v^{5} + 13928v^{4} - 2372v^{3} + 3970v^{2} - 584v + 22\,,\end{aligned}$$

with discriminant $2^{52}3^{15}5^{10}7^{4}13^{4}$ and Galois group S_{24}. While it is certainly possible to use the method of [25, Section 5.1] to compute Belyi maps for higher number field degrees (see for example

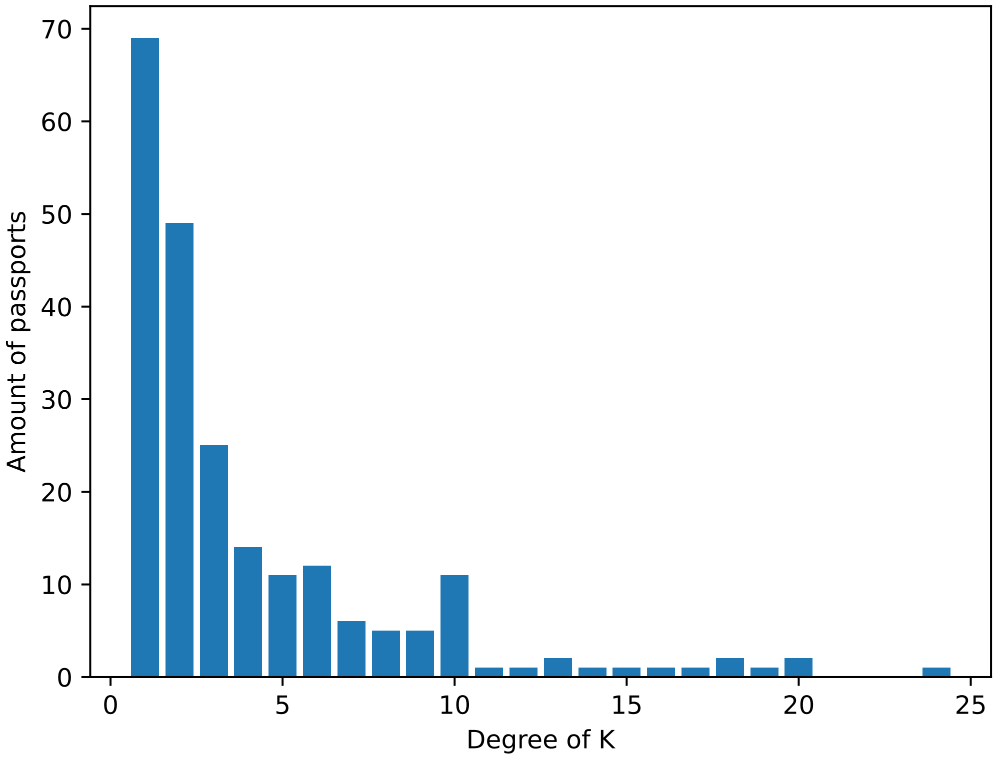

Figure 7.1: Distribution of the degrees of K in the database.

[15]), computing non-trivial amounts of Fourier coefficients becomes infeasible, which is the reason why these large passports have not been included in the database. To give an idea of the size of the coefficients involved, note that the first non-trivial coefficient of the Hauptmodul starts with

$$
\begin{aligned}
&18035144800333385601165709955931694753948569033237338483618065475555557037\ldots \\
&\qquad \ldots 3958967421781810036632024163158272374698352311806451683704 v^{23} + \ldots,
\end{aligned}
$$

which has 132 decimal digits and factors into

$$
\begin{aligned}
2^3 \cdot 3^4 \cdot 7^2 \cdot 13^3 \cdot 179 \cdot 1178062360621513 \\
\cdot 1515687995725535658492175921 \cdot 15731038963473855359968899262003 \\
\cdot 514197500928774955284304452843050396002488377491 \,.
\end{aligned}
$$

The distribution of degrees of K is given in figure 7.1. The defining polynomials of K can be found in appendix B.

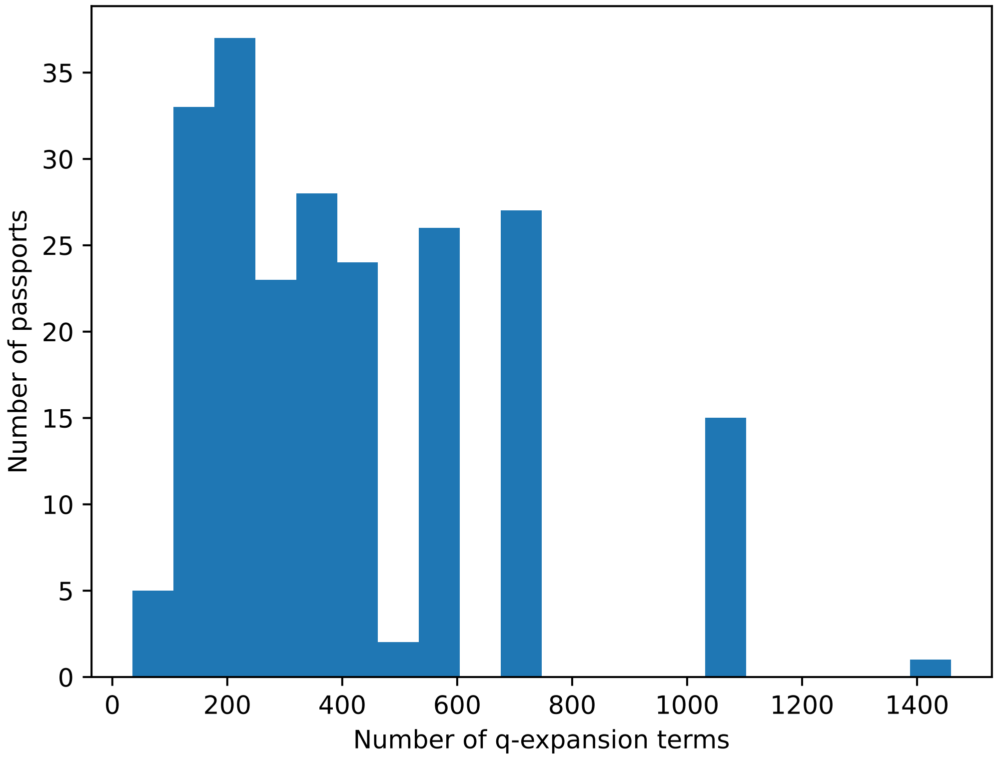

Figure 7.2: Distribution of the number of terms of the Fourier expansions in the database.

7.4.2 Most Fourier Expansion Terms

The largest number of Fourier expansion coefficients that have been computed for the database occurs for the passport *9T27-9_3.3.3_2.2.2.2.1-a* for which we list 1459 coefficients. A distribution of the number of Fourier expansion terms is given in figure 7.2. For genus zero subgroups, we chose the number of terms based on some heuristic guesses that depend on the degree of L and ensure that the computation does not take longer than a few days. For higher genera subgroups, we included as many terms as the LLL could reliably determine from the numerical approximations. For certain passports, we also had to reduce the number of coefficients in order to satisfy the maximal GitHub file size limit of 100MB.

7.4.3 Elliptic Curves Defined over $\mathbb{Q}$

In table 7.1 we give all of the examples of genus 1 noncongruence subgroups from the database that correspond to elliptic curves over $\mathbb{Q}$, together with their defining equations and their conductors.

Passport Label	Elliptic Curve	Conductor
9T27-9_3.3.3_2.2.2.2.1-a	$y^2 + xy + y = x^3 - x^2 - 95x - 697$	162
10T30-10_3.3.3.1_2.2.2.2.2-a	$y^2 + xy + y = x^3 + x^2 - 110x - 880$	15
15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$y^2 + xy + y = x^3 - x^2 + 7x - 103$	270
15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-b	$y^2 + xy = x^3 - x^2 - 41370x + 2022196$	2970

Table 7.1: Noncongruence subgroups corresponding to elliptic curves over $\mathbb{Q}$.

index \ genus	0	1
7	3/3	0
8	1/1	0
9	9/9	1/1
10	9/9	1/1
11	6/6	0
12	23/27	2/3
13	22/23	1/1
14	21/29	1/2
15	54/62	7/9
16	36/65	7/9
17	16/35	1/2
total	200/269	21/28

Table 7.2: Number of computed noncongruence passports that are currently in the database (the second number is the total number of available noncongruence passports). The passports that have not been computed are typically defined over very large number fields and have therefore been left out. (Note that there are no noncongruence subgroups with $\mu \leq 17$ and $g > 1$.)

7.5 Reliability of the Results

For genus zero, all coefficients have been computed using rigorous arithmetic over number fields or rigorous interval arithmetic. The only exception are the Eisenstein series, for which we can only heuristically estimate the precision. For higher genera, the coefficients are non-rigorous (but supported by very convincing numerical evidence) because they have been guessed using the LLL algorithm. As an additional verification, we have also compared the numerical values of the rigorous expressions and the Eisenstein series with a computation using Hejhal's method with a different horocycle height to verify that the results match to at least 10 digits of precision.

7.6 Status of the Database

The number of computed passports in the current version can be found in table 7.2. For the sake of completeness, the database also contains congruence passports, which are not listed here. The current size of the database is ~ 6GB in compressed form and ~ 16GB in uncompressed form. In total, about 25 000 hours of CPU time on `Intel Xeon E5-2680 v4 @ 2.40GHz` have been used to compute the data.

7.7 How to Access the Database

The database is currently available as a GitHub repository at [94] and is planned to be released to the LMFDB soon. The database entries can be loaded by running Sage scripts that return a Python dictionary with the results. This provides more portability between versions than storing the results as pickled objects. We also provide the results in printed form as strings inside JSON files. To save memory, we have not stored the numerical approximations of the Fourier coefficients. For the same reason, we have not explicitly stored the expressions over the number field L, but instead generate them when loading the Sage scripts by substituting the values of v and w.

CHAPTER 8

Further Applications

This chapter briefly lists additional examples of numerical computations of modular forms. These are not related to noncongruence subgroups but still benefit from the improvements to Hejhal's method that have been developed in chapter 4, therefore demonstrating the wide applicability of these ideas.

8.1 Computation of Eigenvalues of Maass Cusp Forms on Hecke Triangle Groups

Motivated by the results of the author's master's thesis [18], we improve and further analyze the *modified MPS* approach (see section 8.1.5) in this thesis and also establish a connection to the results of Judge [95, 96].

8.1.1 Motivation

In this section we consider eigenvalues of Maass cusp forms on Hecke triangle groups at arbitrary precision arithmetic. Our motivation for this comes from a result about eigenvalues of regular polygons (and other cycloidal shapes) in Euclidean space. Suppose Ψ is a dihedrally symmetric eigenfunction with eigenvalue λ inside a regular n-gon with Dirichlet (or Neumann) boundary condition. Since Ψ is dihedrally symmetric, it is sufficient to compute Ψ on a triangle with angles $(\pi/2, \pi/2n, \pi(1/2-1/2n))$ with Dirichlet and/or Neumann boundary conditions, see fig. 8.1.

Let $\lambda_i(n)$ denote the i-th eigenvalue corresponding to a dihedrally symmetric eigenfunction of a regular n-gon with Dirichlet boundary conditions. It has been shown in a number of papers [19–23] that these eigenvalues can be expanded as an asymptotic series in $1/n$

$$\lambda_i(n) \sim \lambda_i(\infty) \sum_{k=0}^{\infty} \frac{C_k(\lambda_i(\infty))}{n^k}, \tag{8.1}$$

where $\lambda_i(\infty)$ is the eigenvalue of the circle, which can be computed explicitly as a root of the Bessel function. Interestingly, the coefficients $C_k(\lambda_i(\infty))$ are polynomials which involve values of the Riemann zeta function and single valued multiple zeta values [22, 23]. For example for the case of

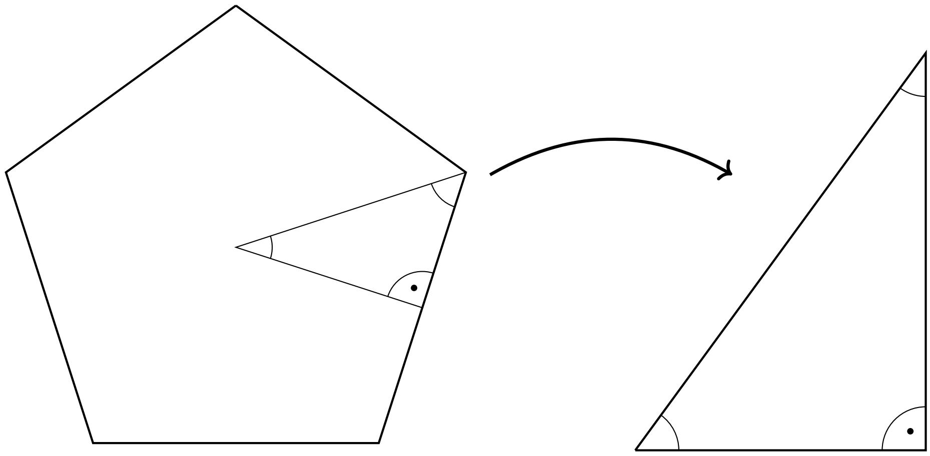

Figure 8.1: Fundamental region for dihedrally symmetric eigenfunctions of regular n-gons. This figure is based on a figure used in [18].

regular polygons with Dirichlet boundary conditions the eigenvalue expansion is of the form

$$\lambda_1(n) = \lambda_1(\infty)\left(1 + \frac{4\zeta(3)}{n^3} + \frac{\zeta(5)(-2\lambda_1(\infty)+12)}{n^5} + \ldots\right), \tag{8.2}$$

where $\lambda_1(\infty)$ denotes the first eigenvalue of the circle. Other examples (including regular n-gons with Neumann boundary condition, star shapes, and other cycloidal shapes) for which the expansion coefficients can be given explicitly are computed in [23].

The question that we want to investigate in this section is the following: Do the eigenvalues of Hecke triangle groups provide a similar expansion formula? If so, what are the expansion coefficients?

8.1.2 Hecke Triangle Groups

The Hecke triangle group $G_{n\geq 3}$ is generated by the $\mathrm{PSL}(2,\mathbb{R})$ elements

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad T_n = \begin{pmatrix} 1 & w \\ 0 & 1 \end{pmatrix}, \tag{8.3}$$

where $w = 2\cos(\pi/n)$, corresponding to the actions

$$\tau \to -1/\tau \quad \text{and} \quad \tau \to \tau + w\,. \tag{8.4}$$

Note that for $n = 3$ one gets $w_3 = 1$ which means that $G_3 = \Gamma$. As a fundamental domain for Hecke triangle groups we choose

$$\mathcal{F}(G_n) = \{\tau \in \overline{\mathcal{H}}\,,\ |\tau| \geq 1 \text{ and } |\mathrm{Re}(\tau)| \leq w/2\} \cup \{i\infty\}\,, \tag{8.5}$$

which has a minimal height of $Y_0 = \sin(\pi/n)$ and corresponds to a hyperbolic triangle with angles $(0, \pi/2, \pi/n)$. Note that Hecke triangle groups only have a single cusp of width w and are of genus zero.

8.1.3 Maass Cusp Forms

Unlike their *classical* holomorphic counterparts, Maass cusp forms are modular forms that are eigenfunctions of the hyperbolic Laplacian

$$\Delta = -y^2 \left(\partial_x^2 + \partial_y^2\right) , \tag{8.6}$$

that satisfy

$$\Delta f(\tau) = \lambda f(\tau) , \tag{8.7}$$

where λ is referred to as the *eigenvalue* of f. In addition, Maass cusp forms vanish at the cusps, transform like weight-zero modular functions ($f(\gamma(\tau)) = f(\tau)$) and have a polynomial growth condition on the expansion coefficients. As a Fourier expansion basis for Maass cusp forms we choose [45]

$$f(\tau) = \sum_{n>0} a_n \sqrt{y} \kappa_{iR} \left(\frac{2\pi n y}{w}\right) e\left(\frac{nx}{w}\right) , \tag{8.8}$$

where

$$e\left(\frac{nx}{w}\right) = \begin{cases} \sin\left(\frac{2\pi nx}{w}\right) & | \quad \text{odd eigenfunctions} \\ \cos\left(\frac{2\pi nx}{w}\right) & | \quad \text{even eigenfunctions} \end{cases} , \tag{8.9}$$

and

$$\kappa_{iR}(u) = \exp(\pi R/2) K_{iR}(u) . \tag{8.10}$$

(The exponential term is just a constant factor that is added for numerical convenience, since it prevents the decay for larger R). K denotes the K-Bessel function and R denotes the spectral parameter which is related to the eigenvalue λ by $\lambda = 1/4 + R^2$.

In the context of Hecke triangle groups, a famous conjecture of Phillips and Sarnak [97] states that no even Maass cusp forms should exist for non-arithmetic (that is $n \neq 3, 4, 6$) Hecke triangle groups. This conjecture is still open, but numerical searches for even eigenvalues of non-arithmetic triangle groups have not been successful, which makes it relatively likely that the conjecture is true. For this reason, we will restrict ourselves to odd Maass cusp forms throughout this project.

8.1.4 Localization of Eigenvalues

While Hejhal's method can be applied analogously to chapter 3 (but with a modified expansion basis), the computation of Maass cusp forms requires an additional step which is the localization of the eigenvalue (or spectral parameter). This means that we need to identify the values of R that correspond to a *true* Maass cusp form. A heuristic approach to this has been described by Hejhal [45]:

1. Given an interval $R_{\min} \leq R \leq R_{\max}$, choose a *sufficiently* small grid of values $R_{\min}, R_{\min} + \delta, R_{\min} + 2\delta, \ldots, R_{\max}$.

2. Select two values Y, Y' with $Y' < Y < Y_0$.

3. Solve for the expansion coefficients $a(R, Y)$ and $a(R, Y')$ for each value within the grid of R values.

4. Compare the coefficients of $a(R, Y)$ and $a(R, Y')$ to look for sign changes and minima in the difference function.

5. Once an interval containing an eigenvalue has been found, use a root-finding technique (such as the secant method) to find the parameter R that minimizes the difference between $a(R, Y)$ and $a(R, Y')$ up to numerical precision.

This procedure has several inconveniences:

- It is not immediately clear what a sufficiently small grid size is (smaller grids lead to more computations, while larger grids may miss eigenvalues).
- Step 4 can lead to *false triggers* that need to be identified, for example, by considering additional pairs of Y values.
- The computation of the expansion coefficients requires a normalization, and it is not clear a priori whether the correct normalization has been chosen (one should therefore impose several normalizations for different multiplicities).

8.1.5 Applying the Modified Method of Particular Solutions

We have experimented with an alternative approach to locate eigenvalues that is based on the *modified method of particular solutions* that was developed by Betcke and Trefethen [24] to compute eigenvalues of shapes in two-dimensional Euclidean space. Our approach can be summarized as follows: We treat the automorphy condition by introducing a matrix $A_B(\lambda)$ containing entries of the form $f(\tau_m) - f(\tau_m^*)$, as described in section 3.1, which can be viewed as a boundary condition. The automorphy condition should only be satisfied near the eigenvalues however in practice sporadic solutions can appear for which $f(\tau)$ approximates the zero function. To avoid these cases, Betcke and Trefethen [24] introduce a second condition, namely that the function should be non-zero in the interior. For this we introduce a second matrix $A_I(\lambda)$ that contains entries of $f(\tau_m)$.

Remark 8.1.1. We have also tested a variant where $A_B(\lambda)$ is set to the matrix $\tilde{V}$ resulting from Hejhal's method, see section 3.2. Both approaches seem to work equally well.

Remark 8.1.2. Instead of assembling A_I by evaluations at random points inside the fundamental domain, we choose the values at the horocycle points because this seems to work equally well and does not require additional evaluations of Bessel functions.

Following Betcke and Trefethen [24] we now introduce a matrix

$$A(\lambda) = \begin{bmatrix} A_B(\lambda) \\ A_I(\lambda) \end{bmatrix}, \tag{8.11}$$

and compute its column-pivoted QR factorization

$$Q(\lambda) = \begin{bmatrix} Q_B(\lambda) \\ Q_I(\lambda) \end{bmatrix}. \tag{8.12}$$

We then compute the singular values $\sigma_i(\lambda)$ of $Q_B^*(\lambda)$ in ascending order, where $Q_B^*(\lambda)$ corresponds to a reduced version of $Q_B(\lambda)$ for which columns corresponding to elements of the factorization

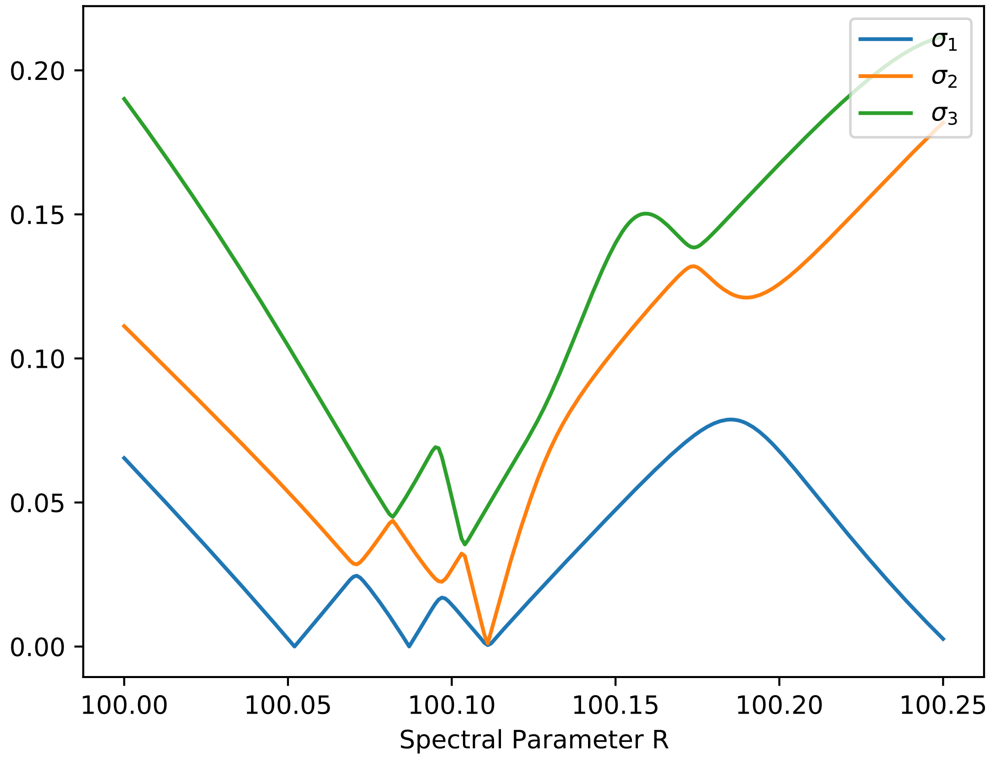

Figure 8.2: Locating the eigenvalues of Γ near $R = 100$ using the modified method of particular solutions. We have chosen $Y_0 = 0.75$ and $M_0 = 30$. By looking at the second singular value, we can also identify the eigenvalues $R = 100.1106656...$ and $R = 100.1112787...$, which have a very small gap.

matrix R below a certain threshold are removed to improve the numerical stability. The minima of the smallest singular value $\sigma_1(\lambda)$ now correspond to the eigenvalues λ.

Remark 8.1.3. Alternatively, one could also compute the generalized singular value decomposition of the matrices A_B and A_I, see [98]. This has been used in [18]. We however found the version that uses a reduced QR decomposition to be more numerically stable.

An example where this approach works very well can be found in figure 8.2, which shows the potential advantages of this method:

- We do not need to impose any normalizations and tests for different multiplicities.
- The method is very robust against *false triggers*.
- The *V-shaped* behavior of $\sigma_1(\lambda)$ can be used to efficiently *jump* close to eigenvalues by extrapolation.
- The higher singular values *point* to the corresponding next eigenvalues, which can also be used to optimize the localization and also helps to detect eigenvalues with very small gaps.

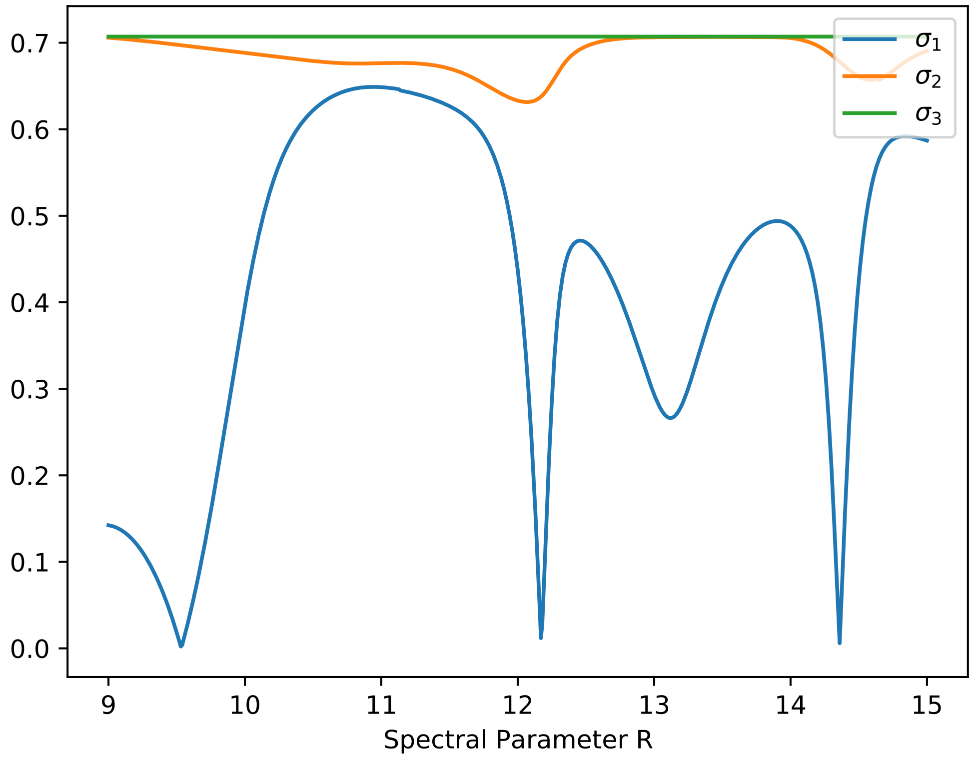

Figure 8.3: Locating the first eigenvalues of Γ using the modified method of particular solutions. We have chosen $Y_0 = 0.75$ and $M_0 = 15$. In this regime, the singular value plot is not as smooth and contains local minima outside the roots.

Unfortunately, not all cases work as well as in figure 8.2. Figure 8.3 shows an example where the plot looks less *V-shaped*, making it more difficult to localize the eigenvalues. It is interesting that the method seems to work better for higher eigenvalues than for lower ones, because the eigenfunctions become more oscillatory for larger spectral parameters. This may be due to the fact that the Bessel functions decay less quickly for larger spectral parameters, causing A_B and A_I to be of a more similar magnitude. Further research is needed to improve the stability of this approach before it can be used in production and for examples with multiple cusps.

8.1.6 Eigenvalue Expansion

To test if the eigenvalues of Hecke triangle groups can be expanded as a $1/n$ series

$$\lambda_i(n) \approx \sum_{j=0}^{N} \frac{c_j}{n^j}, \tag{8.13}$$

we computed $\lambda_i(n)$ for $i = 1, 2, ..., 10$ and $3 \leq n \leq 100$ to 150 digits precision (this computation has been performed in [18]).

Remark 8.1.4. Because the iterative mixed precision solving techniques of section 4.3 had not been developed at the point of performing these computations we had to use direct solving techniques. For future computations the results of section 4.3 should improve the performance of this computation significantly.

Afterwards we performed linear regression to determine the expansion coefficients c_j (similar to [23]). As a heuristic test to check if the eigenvalues offer a $1/n$ expansion we removed different entries to verify that the expressions are converging to the same expansion coefficients. This procedure additionally indicates the accuracy of the results.

Our numerical results indicate that all of the first ten eigenvalues can be expanded as $1/n$ series. This seems to be consistent with the results of Judge [95, 96]. Moreover, we find that odd $1/n$-expansion coefficients vanish up to the estimated accuracy. We also find that the first ten *limiting* eigenvalues (i.e., the values of c_0 which correspond to the eigenvalues of G_∞) are given by the first ten eigenvalues of $\Gamma_0(2)$ (which is conjugate to the theta group and thus has the same spectrum) as given in table 8.1 (note that according to Atkin-Lehner theory [99] the $\mathrm{PSL}(2,\mathbb{Z})$ eigenvalues correspond to multiplicity two eigenvalues on $\Gamma_0(2)$). We also find that the Fourier coefficients a_n converge to the limiting expressions of G_∞. Analogous to the eigenvalues, the odd $1/n$-expansion coefficients for a_n also vanish. However, unlike the eigenvalues, the expressions do not converge absolutely to their limiting values.

Unfortunately, we could not find the expansion coefficients in closed form. The corresponding numerical data can be found in [100].

Eigenvalue order	Limiting Eigenvalue
1	$\lambda_1(\Gamma_0(2))$
2	$\lambda_2(\Gamma_0(2))$
3	$\lambda_3(\Gamma_0(2))$
4	$\lambda_1(\mathrm{PSL}(2,\mathbb{Z}))$
5	$\lambda_1(\mathrm{PSL}(2,\mathbb{Z}))$
6	$\lambda_4(\Gamma_0(2))$
7	$\lambda_5(\Gamma_0(2))$
8	$\lambda_2(\mathrm{PSL}(2,\mathbb{Z}))$
9	$\lambda_2(\mathrm{PSL}(2,\mathbb{Z}))$
10	$\lambda_6(\Gamma_0(2))$

Table 8.1: Limiting eigenvalues (i.e. c_0 of eq. (8.13))

8.2 Traces of Real Singular Moduli

In this section, we demonstrate that the method developed in chapter 4 can be applied in various contexts by using it to compute traces of real singular moduli with very high precision. These were first computed in the work of Duke, Imamoglu, and Toth [101].

8.2.1 Background and Notation

The mathematical motivation is beyond the scope of this thesis, so we will just define the quantities that appear and refer to [27, 101] for a proper introduction. Let $Q(x,y) := ax^2 + bxy + cy^2$ denote a binary quadratic form with coefficients $a, b, c \in \mathbb{Z}$ and discriminant $d = b^2 - 4ac$. Let $\mathcal{Q}_d$ denote the set of binary quadratic forms with discriminant d. To each $Q \in \mathcal{Q}_d$, we associate the root τ_Q of $Q(\tau, 1)$ in $\mathcal{H}$. An element $\gamma \in \Gamma$ acts on Q by [101]

$$(\gamma Q)(x,y) = Q(\gamma_4 x - \gamma_2 y, -\gamma_3 x + \gamma_1 y), \quad \text{where} \quad \gamma = \begin{pmatrix} \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_4 \end{pmatrix} \in \Gamma\,. \tag{8.14}$$

Let $j_1(\tau) = j - 744 = q^{-1} + 196884q + \ldots$ be the normalized Hauptmodul for Γ. DIT [101] studied the quantity [101, Eq. 1.8]

$$\mathrm{Tr}_d(j_1) = \tfrac{1}{2\pi} \sum_{Q \in \Gamma / \mathcal{Q}_d} \int_{C_Q} j_1(\tau) \tfrac{d\tau}{Q(\tau,1)}\,, \tag{8.15}$$

for $d > 0$ and $d \equiv 0, 1 \pmod 4$ for a suitably defined smooth curve C_Q. More generally, let j_m denote the unique basis elements for $\mathbb{C}[j]$ of the form $j_m(\tau) = q^{-m} + O(q)$. Then $\mathrm{Tr}_d(j_m)$ are the so-called *traces of real singular moduli*. DIT [101, Theorem 5] showed that the generating function

$$F_d(\tau) = - \sum_{m \geq 0} \mathrm{Tr}_d(j_m) q^m\,, \tag{8.16}$$

satisfies the functional equation

$$F_d(\tau) - \tau^{-2} F_d(-\tfrac{1}{\tau}) = \tfrac{1}{\pi} \sum_{c<0<a\,,\, b^2-4ac=d} (a\tau^2 + b\tau + c)^{-1}\,. \tag{8.17}$$

8.2.2 Numerical Computation

DIT have numerically computed $\mathrm{Tr}_d(j_m)$ for $d = 5, 8, 12, 13, 17, 20$ and $m = 0, 1, 2, 3$ to six digits precision [101, Table 2]. An alternative approach is to use Hejhal's method. Using equations 8.16 and 8.17 we can compute the coefficients of F_d analogously to chapter 4. We define $r_d(\tau) := \frac{1}{\pi} \sum_{c<0<a\,,\, b^2-4ac=d} (a\tau^2 + b\tau + c)^{-1}$ and similar to equation 3.4 work with $yF_d(\tau)$ to improve the numerical stability. Then we get

$$\begin{aligned} a_n Y \exp(-2\pi n Y) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} F_d(\tau) \exp(-2\pi i n x) dx\,, & (8.18)\\ &\approx \frac{1}{2Q} \sum_{m=0}^{2Q-1} F_d(\tau_m) \exp\bigl(-2\pi i n x_m\bigr)\,, & (8.19)\\ &= \frac{1}{2Q} \sum_{m=0}^{2Q-1} \left(\left(\frac{|\tau_m|}{\tau_m} \right)^2 y_m^* F_d(\tau_m^*) + Y r_d(z_m) \right) \exp\bigl(-2\pi i n x_m\bigr)\,, & (8.20) \end{aligned}$$

where $\tau_m^* = -1/\tau_m = x_m^* + i y_m^*$, which we solve efficiently using the iterative methods of section 4.3.

For $d = 5, 8, 12, 13, 17, 20, 21, 24, 28, 29, 32, 33, 37, 40, 41, 44, 45, 48$ we have computed $F_d(\tau)$ to 2500 digits precision, resulting in coefficients for $m = 0, \ldots, 1063$ with between 500 and 2500 digits precision. This computation took just a few hours on a standard desktop computer. Some of the data can be found in appendix C, while the full data is available in [102].

Remark 8.2.1. We get a different result than DIT [101, Table 2] for $d = 20$, $m = 0$. Since our data agree for $m = 1, 2, 3$, we expect our result to be correct, since it would not make much sense for our algorithm to get only the leading order wrong.

We were not able to find closed-form expressions for the computed data and leave this for further research.

CHAPTER 9

Conclusion and Outlook

The main focus of this thesis was the numerical computation of noncongruence modular forms. For this, we have significantly improved the performance of Hejhal's method in chapter 4 by using mixed-precision iterative solving techniques and by optimizing the linear algebra involved. An alternative approach for subgroups of genus zero was presented in chapter 5. This more restricted approach has the advantage of producing rigorous Fourier expansions and is usually faster. We then considered noncongruence Eisenstein series and provided an efficient method for computing them in chapter 6. Our approach is to use the computed bases of cusp forms and modular forms to project out the Eisenstein space using Petersson products. Our results indicate that non-trivial noncongruence Eisenstein series are algebraic only for a few special cases, and furthermore, no non-trivial algebraic noncongruence Eisenstein series of weight $k > 2$ have yet been found. We have computed a large number of modular forms, cusp forms and Eisenstein series of weight $k \leq 6$ on noncongruence subgroups of index $\mu \leq 17$ and presented a database for them in chapter 7. This data is planned to be added to the LMFDB database and will hopefully help to gain a deeper understanding of the still very mysterious modular forms on noncongruence subgroups. Finally, in chapter 8 we have highlighted that the methods developed in this thesis can be efficiently applied to applications from various fields. For this, we have briefly considered the computation of Maass cusp forms on Hecke triangle groups in section 8.1 which benefit from the improved mixed-precision iterative solving techniques and investigated an alternative approach for locating the spectral parameters. In addition, we have shown in section 8.2 that the generating function of traces of real singular moduli can be very efficiently computed using the methods developed in chapter 4.

This thesis has focused mainly on numerical work, so the most obvious area for further research is to analyze the computed data of chapter 7 and to further develop the underlying theory. Hejhal's method, which was discussed in chapters 3 and 4, probably cannot be improved significantly anymore, since the problem sizes are limited by the convergence rates in the evaluation of modular forms, and the iteration counts of the iterative methods with mixed-precision preconditioners are already very low. As discussed in remark 5.3.1, a potentially more efficient numerical approach could be to compute the forms from the curve also for subgroups of higher genus. It would also be interesting to apply the developed methods to other problems, such as other types of modular forms, analogous to what we have done in chapter 8.

Also, although we have already computed hundreds of examples of noncongruence modular forms in chapter 7, it might be useful to have more examples (especially of genus > 1, which was not present for the index < 18 subgroups). For this, we have created a database of noncongruence subgroups with index < 32 [103] and plan to compute modular forms for some of them in the future.

In addition, this thesis has produced many high-precision floating-point approximations for which we have been unable to find closed-form expressions. In particular, it would be very interesting to recognize the noncongruence Eisenstein series computed in chapters 6 and 7, possibly using non-algebraic constants. Furthermore, it would be interesting to find closed-form expressions for the $1/n$ expansion coefficients of the eigenvalues of Maass cusp forms on Hecke triangle groups (see section 8.1), as well as for the traces of real singular moduli (see section 8.2).

APPENDIX A

Algebraic Eisenstein Series for H_5 in Canonical Normalization

In the canonical normalization, the weight two Eisenstein series on H_5 can no longer be defined over $\mathbb{Q}$, but instead are defined over the numberfield $K = \mathbb{Q}(v)$, where $v^4 - v^3 + v^2 - v + 1 = 0$ with embedding (for our representation) $v = -0.3090... - 0.9510...i$ if the Eisenstein series correspond to cusps of width 1. For the remaining cusps (which have cusp width 11) we get that the Eisenstein series are expressions over $\zeta_5 = \exp(2\pi i/5)$ and $a = \sqrt[5]{11}$. In our representation the cusp representatives are given by

$$\{i\infty, 0, -1/3, 1/4, -4/13, 7/27, -4/11, 3/11, -27/88, 20/77\}\,, \tag{A.1}$$

which means that the i-th Eisenstein series is associated to the cusp of the i-th entry of the set in Eq. (A.1). In the canonical normalization we then get

$$\begin{aligned}
\tilde{e}_0 = e_0 &+ (\tfrac{-21}{25}v^3 + \tfrac{-13}{25}v^2 + \tfrac{-8}{25}v^1 + \tfrac{-21}{25})e_1 + (\tfrac{209}{125}v^3 + \tfrac{131}{125}v^2 + \tfrac{16}{25}v^1 + \tfrac{-502}{125})e_2 \\
&+ (\tfrac{523}{625}v^3 + \tfrac{306}{625}v^2 + \tfrac{187}{625}v^1 + \tfrac{3399}{625})e_3 + (\tfrac{-193}{125}v^3 + \tfrac{-651}{625}v^2 + \tfrac{-348}{625}v^1 + \tfrac{-1073}{625})e_4 \\
&+ (\tfrac{-83719}{78125}v^3 + \tfrac{-25967}{78125}v^2 + \tfrac{-27127}{78125}v^1 + \tfrac{-83624}{78125})e_5 + (\tfrac{-760188}{390625}v^3 + \tfrac{-527357}{390625}v^2 + \tfrac{-77863}{78125}v^1 + \tfrac{-427501}{390625})e_6 \\
&+ (\tfrac{5592851}{1953125}v^3 + \tfrac{1634437}{1953125}v^2 + \tfrac{2435354}{1953125}v^1 + \tfrac{5753183}{1953125})e_7 + (\tfrac{-1632279}{1953125}v^3 + \tfrac{91947}{78125}v^2 + \tfrac{412568}{1953125}v^1 + \tfrac{29459218}{1953125})e_8\,, \\
\tilde{e}_1 = &\;(-\tfrac{1}{5}a^4 - \tfrac{1}{5}a^3 - \tfrac{1}{5}a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})e_1 \\
&+ (-\tfrac{3}{25}a^4 - \tfrac{11}{25}a^3 - \tfrac{24}{25}a^2 - \tfrac{11}{5}a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})e_2 \\
&+ (-\tfrac{2}{125}a^4 - \tfrac{18}{125}a^3 - \tfrac{43}{125}a^2 - \tfrac{22}{25}a - \tfrac{523}{625}\zeta_5^3 + \tfrac{306}{625}\zeta_5^2 - \tfrac{187}{625}\zeta_5 - \tfrac{601}{625})e_3 \\
&+ (-\tfrac{472}{625}a^4 - \tfrac{621}{625}a^3 - \tfrac{849}{625}a^2 - \tfrac{187}{125}a + \tfrac{193}{125}\zeta_5^3 - \tfrac{651}{625}\zeta_5^2 + \tfrac{348}{625}\zeta_5 - \tfrac{10911}{3125})e_4 \\
&+ (-\tfrac{1764}{3125}a^4 - \tfrac{2444}{3125}a^3 - \tfrac{3789}{3125}a^2 - \tfrac{1078}{625}a + \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 + \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})e_5 \\
&+ (-\tfrac{83657}{78125}a^4 - \tfrac{109274}{78125}a^3 - \tfrac{147876}{78125}a^2 - \tfrac{7854}{3125}a + \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2 + \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})e_6 \\
&+ (-\tfrac{107166}{390625}a^4 - \tfrac{326469}{390625}a^3 - \tfrac{593579}{390625}a^2 - \tfrac{206448}{78125}a - \tfrac{5592851}{1953125}\zeta_5^3 + \tfrac{1634437}{1953125}\zeta_5^2 - \tfrac{2435354}{1953125}\zeta_5 - \tfrac{7628117}{1953125})e_7 \\
&+ (-\tfrac{1274709}{1953125}a^4 - \tfrac{2152392}{1953125}a^3 - \tfrac{4373433}{1953125}a^2 - \tfrac{1542541}{390625}a + \tfrac{1632279}{1953125}\zeta_5^3 + \tfrac{91947}{78125}\zeta_5^2 - \tfrac{412568}{1953125}\zeta_5 - \tfrac{37334896}{9765625})e_8\,, \\
\tilde{e}_2 = &\;(-\tfrac{1}{5}\zeta_5 a^4 - \tfrac{1}{5}\zeta_5^2 a^3 - \tfrac{1}{5}\zeta_5^3 a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})e_1 \\
&+ (-\tfrac{3}{25}\zeta_5 a^4 - \tfrac{11}{25}\zeta_5^2 a^3 - \tfrac{24}{25}\zeta_5^3 a^2 + \left(\tfrac{11}{5}\zeta_5^3 + \tfrac{11}{5}\zeta_5^2 + \tfrac{11}{5}\zeta_5 + \tfrac{11}{5}\right) a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})e_2
\end{aligned}$$

$$+ (-\frac{2}{125}\zeta_5 a^4 - \frac{18}{125}\zeta_5^2 a^3 - \frac{43}{125}\zeta_5^3 a^2 + \left(\frac{22}{25}\zeta_5^3 + \frac{22}{25}\zeta_5^2 + \frac{22}{25}\zeta_5 + \frac{22}{25}\right) a - \frac{523}{625}\zeta_5^3 + \frac{306}{625}\zeta_5^2 - \frac{187}{625}\zeta_5 - \frac{601}{625})e_3$$

$$+ (-\frac{472}{625}\zeta_5 a^4 - \frac{621}{625}\zeta_5^2 a^3 - \frac{849}{625}\zeta_5^3 a^2 + \left(\frac{187}{125}\zeta_5^3 + \frac{187}{125}\zeta_5^2 + \frac{187}{125}\zeta_5 + \frac{187}{125}\right) a + \frac{193}{125}\zeta_5^3 - \frac{651}{625}\zeta_5^2 + \frac{348}{625}\zeta_5 - \frac{10911}{3125})e_4$$

$$+ (-\frac{1764}{3125}\zeta_5 a^4 - \frac{2444}{3125}\zeta_5^2 a^3 - \frac{3789}{3125}\zeta_5^3 a^2 + \left(\frac{1078}{625}\zeta_5^3 + \frac{1078}{625}\zeta_5^2 + \frac{1078}{625}\zeta_5 + \frac{1078}{625}\right) a$$

$$+ \frac{83719}{78125}\zeta_5^3 - \frac{25967}{78125}\zeta_5^2 + \frac{27127}{78125}\zeta_5 - \frac{224749}{78125})e_5$$

$$+ (-\frac{83657}{78125}\zeta_5 a^4 - \frac{109274}{78125}\zeta_5^2 a^3 - \frac{147876}{78125}\zeta_5^3 a^2 + \left(\frac{7854}{3125}\zeta_5^3 + \frac{7854}{3125}\zeta_5^2 + \frac{7854}{3125}\zeta_5 + \frac{7854}{3125}\right) a + \frac{760188}{390625}\zeta_5^3 - \frac{527357}{390625}\zeta_5^2$$

$$+ \frac{77863}{78125}\zeta_5 - \frac{2211071}{390625})e_6 + (-\frac{107166}{390625}\zeta_5 a^4 - \frac{326469}{390625}\zeta_5^2 a^3 - \frac{593579}{390625}\zeta_5^3 a^2$$

$$+ \left(\frac{206448}{78125}\zeta_5^3 + \frac{206448}{78125}\zeta_5^2 + \frac{206448}{78125}\zeta_5 + \frac{206448}{78125}\right) a$$

$$- \frac{5592851}{1953125}\zeta_5^3 + \frac{1634437}{1953125}\zeta_5^2 - \frac{2435354}{1953125}\zeta_5 - \frac{7628117}{1953125})e_7$$

$$+ (-\frac{1274709}{1953125}\zeta_5 a^4 - \frac{2152392}{1953125}\zeta_5^2 a^3 - \frac{4373433}{1953125}\zeta_5^3 a^2 + \left(\frac{1542541}{390625}\zeta_5^3 + \frac{1542541}{390625}\zeta_5^2 + \frac{1542541}{390625}\zeta_5 + \frac{1542541}{390625}\right) a$$

$$+ \frac{1632279}{1953125}\zeta_5^3 + \frac{91947}{78125}\zeta_5^2 - \frac{412568}{1953125}\zeta_5 - \frac{37334896}{9765625})e_8\,,$$

$$\tilde{e}_3 = (\left(\frac{1}{5}\zeta_5^3 + \frac{1}{5}\zeta_5^2 + \frac{1}{5}\zeta_5 + \frac{1}{5}\right) a^4 - \frac{1}{5}\zeta_5^3 a^3 - \frac{1}{5}\zeta_5^2 a^2 + \frac{21}{25}\zeta_5^3 - \frac{13}{25}\zeta_5^2 + \frac{8}{25}\zeta_5 - \frac{13}{25})e_1$$

$$+ (\left(\frac{3}{25}\zeta_5^3 + \frac{3}{25}\zeta_5^2 + \frac{3}{25}\zeta_5 + \frac{3}{25}\right) a^4 - \frac{11}{25}\zeta_5^3 a^3 - \frac{24}{25}\zeta_5^2 a^2 - \frac{11}{5}\zeta_5 a - \frac{209}{125}\zeta_5^3 + \frac{131}{125}\zeta_5^2 - \frac{16}{25}\zeta_5 - \frac{312}{125})e_2$$

$$+ (\left(\frac{2}{125}\zeta_5^3 + \frac{2}{125}\zeta_5^2 + \frac{2}{125}\zeta_5 + \frac{2}{125}\right) a^4 - \frac{18}{125}\zeta_5^3 a^3 - \frac{43}{125}\zeta_5^2 a^2 - \frac{22}{25}\zeta_5 a - \frac{523}{625}\zeta_5^3 + \frac{306}{625}\zeta_5^2 - \frac{187}{625}\zeta_5 - \frac{601}{625})e_3$$

$$+ (\left(\frac{472}{625}\zeta_5^3 + \frac{472}{625}\zeta_5^2 + \frac{472}{625}\zeta_5 + \frac{472}{625}\right) a^4 - \frac{621}{625}\zeta_5^3 a^3 - \frac{849}{625}\zeta_5^2 a^2 - \frac{187}{125}\zeta_5 a + \frac{193}{125}\zeta_5^3 - \frac{651}{625}\zeta_5^2 + \frac{348}{625}\zeta_5 - \frac{10911}{3125})e_4$$

$$+ (\left(\frac{1764}{3125}\zeta_5^3 + \frac{1764}{3125}\zeta_5^2 + \frac{1764}{3125}\zeta_5 + \frac{1764}{3125}\right) a^4 - \frac{2444}{3125}\zeta_5^3 a^3 - \frac{3789}{3125}\zeta_5^2 a^2 - \frac{1078}{625}\zeta_5 a$$

$$\frac{83719}{78125}\zeta_5^3 - \frac{25967}{78125}\zeta_5^2 + \frac{27127}{78125}\zeta_5 - \frac{224749}{78125})e_5$$

$$+ (\left(\frac{83657}{78125}\zeta_5^3 + \frac{83657}{78125}\zeta_5^2 + \frac{83657}{78125}\zeta_5 + \frac{83657}{78125}\right) a^4 - \frac{109274}{78125}\zeta_5^3 a^3 - \frac{147876}{78125}\zeta_5^2 a^2 - \frac{7854}{3125}\zeta_5 a$$

$$+ \frac{760188}{390625}\zeta_5^3 - \frac{527357}{390625}\zeta_5^2 + \frac{77863}{78125}\zeta_5 - \frac{2211071}{390625})e_6$$

$$+ (\left(\frac{107166}{390625}\zeta_5^3 + \frac{107166}{390625}\zeta_5^2 + \frac{107166}{390625}\zeta_5 + \frac{107166}{390625}\right) a^4 - \frac{326469}{390625}\zeta_5^3 a^3 - \frac{593579}{390625}\zeta_5^2 a^2 - \frac{206448}{78125}\zeta_5 a$$

$$- \frac{5592851}{1953125}\zeta_5^3 + \frac{1634437}{1953125}\zeta_5^2 - \frac{2435354}{1953125}\zeta_5 - \frac{7628117}{1953125})e_7$$

$$+ (\left(\frac{1274709}{1953125}\zeta_5^3 + \frac{1274709}{1953125}\zeta_5^2 + \frac{1274709}{1953125}\zeta_5 + \frac{1274709}{1953125}\right) a^4 - \frac{2152392}{1953125}\zeta_5^3 a^3 - \frac{4373433}{1953125}\zeta_5^2 a^2 - \frac{1542541}{390625}\zeta_5 a$$

$$+ \frac{1632279}{1953125}\zeta_5^3 + \frac{91947}{78125}\zeta_5^2 - \frac{412568}{1953125}\zeta_5 - \frac{37334896}{9765625})e_8\,,$$

$$\tilde{e}_4 = (-\frac{1}{5}\zeta_5^2 a^4 + \left(\frac{1}{5}\zeta_5^3 + \frac{1}{5}\zeta_5^2 + \frac{1}{5}\zeta_5 + \frac{1}{5}\right) a^3 - \frac{1}{5}\zeta_5 a^2 + \frac{21}{25}\zeta_5^3 - \frac{13}{25}\zeta_5^2 + \frac{8}{25}\zeta_5 - \frac{13}{25})e_1$$

$$+ (-\frac{3}{25}\zeta_5^2 a^4 + \left(\frac{11}{25}\zeta_5^3 + \frac{11}{25}\zeta_5^2 + \frac{11}{25}\zeta_5 + \frac{11}{25}\right) a^3 - \frac{24}{25}\zeta_5 a^2 - \frac{11}{5}\zeta_5^3 a - \frac{209}{125}\zeta_5^3 + \frac{131}{125}\zeta_5^2 - \frac{16}{25}\zeta_5 - \frac{312}{125})e_2$$

$$+ (-\frac{2}{125}\zeta_5^2 a^4 + \left(\frac{18}{125}\zeta_5^3 + \frac{18}{125}\zeta_5^2 + \frac{18}{125}\zeta_5 + \frac{18}{125}\right) a^3 - \frac{43}{125}\zeta_5 a^2 - \frac{22}{25}\zeta_5^3 a - \frac{523}{625}\zeta_5^3 + \frac{306}{625}\zeta_5^2 - \frac{187}{625}\zeta_5 - \frac{601}{625})e_3$$

$$+ (-\frac{472}{625}\zeta_5^2 a^4 + \left(\frac{621}{625}\zeta_5^3 + \frac{621}{625}\zeta_5^2 + \frac{621}{625}\zeta_5 + \frac{621}{625}\right) a^3 - \frac{849}{625}\zeta_5 a^2 - \frac{187}{125}\zeta_5^3 a + \frac{193}{125}\zeta_5^3 - \frac{651}{625}\zeta_5^2 + \frac{348}{625}\zeta_5 - \frac{10911}{3125})e_4$$

$$+ (-\frac{1764}{3125}\zeta_5^2 a^4 + \left(\frac{2444}{3125}\zeta_5^3 + \frac{2444}{3125}\zeta_5^2 + \frac{2444}{3125}\zeta_5 + \frac{2444}{3125}\right) a^3 - \frac{3789}{3125}\zeta_5 a^2 - \frac{1078}{625}\zeta_5^3 a$$

$$+ \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 + \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})e_5$$
$$+ (-\tfrac{83657}{78125}\zeta_5^2 a^4 + \left(\tfrac{109274}{78125}\zeta_5^3 + \tfrac{109274}{78125}\zeta_5^2 + \tfrac{109274}{78125}\zeta_5 + \tfrac{109274}{78125}\right) a^3 - \tfrac{147876}{78125}\zeta_5 a^2 - \tfrac{7854}{3125}\zeta_5^3 a$$
$$+ \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2 + \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})e_6$$
$$+ (-\tfrac{107166}{390625}\zeta_5^2 a^4 + \left(\tfrac{326469}{390625}\zeta_5^3 + \tfrac{326469}{390625}\zeta_5^2 + \tfrac{326469}{390625}\zeta_5 + \tfrac{326469}{390625}\right) a^3 - \tfrac{593579}{390625}\zeta_5 a^2 - \tfrac{206448}{78125}\zeta_5^3 a$$
$$- \tfrac{5592851}{1953125}\zeta_5^3 + \tfrac{1634437}{1953125}\zeta_5^2 - \tfrac{2435354}{1953125}\zeta_5 - \tfrac{7628117}{1953125})e_7$$
$$+ (-\tfrac{1274709}{1953125}\zeta_5^2 a^4 + \left(\tfrac{2152392}{1953125}\zeta_5^3 + \tfrac{2152392}{1953125}\zeta_5^2 + \tfrac{2152392}{1953125}\zeta_5 + \tfrac{2152392}{1953125}\right) a^3 - \tfrac{4373433}{1953125}\zeta_5 a^2 - \tfrac{1542541}{390625}\zeta_5^3 a$$
$$+ \tfrac{1632279}{1953125}\zeta_5^3 + \tfrac{91947}{78125}\zeta_5^2 - \tfrac{412568}{1953125}\zeta_5 - \tfrac{37334896}{9765625})e_8\,,$$
$$\tilde{e}_5 = (-\tfrac{1}{5}\zeta_5^3 a^4 - \tfrac{1}{5}\zeta_5 a^3 + \left(\tfrac{1}{5}\zeta_5^3 + \tfrac{1}{5}\zeta_5^2 + \tfrac{1}{5}\zeta_5 + \tfrac{1}{5}\right) a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})e_1$$
$$+ (-\tfrac{3}{25}\zeta_5^3 a^4 - \tfrac{11}{25}\zeta_5 a^3 + \left(\tfrac{24}{25}\zeta_5^3 + \tfrac{24}{25}\zeta_5^2 + \tfrac{24}{25}\zeta_5 + \tfrac{24}{25}\right) a^2 - \tfrac{11}{5}\zeta_5^2 a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})e_2$$
$$+ (-\tfrac{2}{125}\zeta_5^3 a^4 - \tfrac{18}{125}\zeta_5 a^3 + \left(\tfrac{43}{125}\zeta_5^3 + \tfrac{43}{125}\zeta_5^2 + \tfrac{43}{125}\zeta_5 + \tfrac{43}{125}\right) a^2 - \tfrac{22}{25}\zeta_5^2 a - \tfrac{523}{625}\zeta_5^3 + \tfrac{306}{625}\zeta_5^2 - \tfrac{187}{625}\zeta_5 - \tfrac{601}{625})e_3$$
$$+ (-\tfrac{472}{625}\zeta_5^3 a^4 - \tfrac{621}{625}\zeta_5 a^3 + \left(\tfrac{849}{625}\zeta_5^3 + \tfrac{849}{625}\zeta_5^2 + \tfrac{849}{625}\zeta_5 + \tfrac{849}{625}\right) a^2 - \tfrac{187}{125}\zeta_5^2 a + \tfrac{193}{125}\zeta_5^3 - \tfrac{651}{625}\zeta_5^2 + \tfrac{348}{625}\zeta_5 - \tfrac{10911}{3125})e_4$$
$$+ (-\tfrac{1764}{3125}\zeta_5^3 a^4 - \tfrac{2444}{3125}\zeta_5 a^3 + \left(\tfrac{3789}{3125}\zeta_5^3 + \tfrac{3789}{3125}\zeta_5^2 + \tfrac{3789}{3125}\zeta_5 + \tfrac{3789}{3125}\right) a^2 - \tfrac{1078}{625}\zeta_5^2 a + \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 + \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})e_5$$
$$+ (-\tfrac{83657}{78125}\zeta_5^3 a^4 - \tfrac{109274}{78125}\zeta_5 a^3 + \left(\tfrac{147876}{78125}\zeta_5^3 + \tfrac{147876}{78125}\zeta_5^2 + \tfrac{147876}{78125}\zeta_5 + \tfrac{147876}{78125}\right) a^2 - \tfrac{7854}{3125}\zeta_5^2 a + \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2$$
$$+ \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})e_6$$
$$+ (-\tfrac{107166}{390625}\zeta_5^3 a^4 - \tfrac{326469}{390625}\zeta_5 a^3 + \left(\tfrac{593579}{390625}\zeta_5^3 + \tfrac{593579}{390625}\zeta_5^2 + \tfrac{593579}{390625}\zeta_5 + \tfrac{593579}{390625}\right) a^2 - \tfrac{206448}{78125}\zeta_5^2 a - \tfrac{5592851}{1953125}\zeta_5^3 + \tfrac{1634437}{1953125}\zeta_5^2$$
$$- \tfrac{2435354}{1953125}\zeta_5 - \tfrac{7628117}{1953125})e_7$$
$$+ (-\tfrac{1274709}{1953125}\zeta_5^3 a^4 - \tfrac{2152392}{1953125}\zeta_5 a^3 + \left(\tfrac{4373433}{1953125}\zeta_5^3 + \tfrac{4373433}{1953125}\zeta_5^2 + \tfrac{4373433}{1953125}\zeta_5 + \tfrac{4373433}{1953125}\right) a^2 - \tfrac{1542541}{390625}\zeta_5^2 a + \tfrac{1632279}{1953125}\zeta_5^3 + \tfrac{91947}{78125}\zeta_5^2$$
$$- \tfrac{412568}{1953125}\zeta_5 - \tfrac{37334896}{9765625})e_8\,,$$
$$\tilde{e}_6 = (\tfrac{-8}{25}v^3 + \tfrac{-34}{25}v^2 + \tfrac{26}{25}v^1 + \tfrac{-13}{25})e_1$$
$$+ (\tfrac{78}{125}v^3 + \tfrac{342}{125}v^2 + \tfrac{-52}{25}v^1 + \tfrac{131}{125})e_2$$
$$+ (\tfrac{217}{625}v^3 + \tfrac{799}{625}v^2 + \tfrac{-642}{625}v^1 + \tfrac{306}{625})e_3$$
$$+ (\tfrac{-314}{625}v^3 + \tfrac{-66}{25}v^2 + \tfrac{1268}{625}v^1 + \tfrac{-651}{625})e_4$$
$$+ (\tfrac{-57752}{78125}v^3 + \tfrac{-79061}{78125}v^2 + \tfrac{82559}{78125}v^1 + \tfrac{-25967}{78125})e_5$$
$$+ (\tfrac{-232831}{390625}v^3 + \tfrac{-1444029}{390625}v^2 + \tfrac{179646}{78125}v^1 + \tfrac{-527357}{390625})e_6$$
$$+ (\tfrac{3958414}{1953125}v^3 + \tfrac{5704228}{1953125}v^2 + \tfrac{-4791934}{1953125}v^1 + \tfrac{1634437}{1953125})e_7$$
$$+ (\tfrac{-3930954}{1953125}v^3 + \tfrac{5009918}{1953125}v^2 + \tfrac{-253828}{1953125}v^1 + \tfrac{91947}{78125})e_8\,,$$
$$\tilde{e}_7 = (\tfrac{-34}{25}v^3 + \tfrac{8}{25}v^2 + \tfrac{-42}{25}v^1 + \tfrac{21}{25})e_1$$
$$+ (\tfrac{338}{125}v^3 + \tfrac{-78}{125}v^2 + \tfrac{84}{25}v^1 + \tfrac{-209}{125})e_2$$
$$+ (\tfrac{859}{625}v^3 + \tfrac{-217}{625}v^2 + \tfrac{1016}{625}v^1 + \tfrac{-523}{625})e_3 + (\tfrac{-1582}{625}v^3 + \tfrac{314}{625}v^2 + \tfrac{-1964}{625}v^1 + \tfrac{193}{125})e_4$$
$$+ (\tfrac{-140311}{78125}v^3 + \tfrac{57752}{78125}v^2 + \tfrac{-136813}{78125}v^1 + \tfrac{83719}{78125})e_5$$

$$
\begin{aligned}
&+ (\tfrac{-1131061}{390625} v^3 + \tfrac{232831}{390625} v^2 + \tfrac{-335372}{78125} v^1 + \tfrac{760188}{390625}) e_6 \\
&+ (\tfrac{8750348}{1953125} v^3 + \tfrac{-3958414}{1953125} v^2 + \tfrac{9662642}{1953125} v^1 + \tfrac{-5592851}{1953125}) e_7 \\
&+ (\tfrac{-3677126}{1953125} v^3 + \tfrac{3930954}{1953125} v^2 + \tfrac{1078964}{1953125} v^1 + \tfrac{1632279}{1953125}) e_8\,, \\
\tilde{e}_8 &= (\tfrac{-42}{25} v^3 + \tfrac{-26}{25} v^2 + \tfrac{-16}{25} v^1 + \tfrac{8}{25}) e_1 \\
&+ (\tfrac{84}{25} v^3 + \tfrac{52}{25} v^2 + \tfrac{32}{25} v^1 + \tfrac{-16}{25}) e_2 \\
&+ (\tfrac{1016}{625} v^3 + \tfrac{642}{625} v^2 + \tfrac{374}{625} v^1 + \tfrac{-187}{625}) e_3 \\
&+ (\tfrac{-1964}{625} v^3 + \tfrac{-1268}{625} v^2 + \tfrac{-696}{625} v^1 + \tfrac{348}{625}) e_4 \\
&+ (\tfrac{-136813}{78125} v^3 + \tfrac{-82559}{78125} v^2 + \tfrac{-54254}{78125} v^1 + \tfrac{27127}{78125}) e_5 \\
&+ (\tfrac{-335372}{78125} v^3 + \tfrac{-179646}{78125} v^2 + \tfrac{-155726}{78125} v^1 + \tfrac{77863}{78125}) e_6 \\
&+ (\tfrac{9662642}{1953125} v^3 + \tfrac{4791934}{1953125} v^2 + \tfrac{4870708}{1953125} v^1 + \tfrac{-2435354}{1953125}) e_7 \\
&+ (\tfrac{1078964}{1953125} v^3 + \tfrac{253828}{1953125} v^2 + \tfrac{825136}{1953125} v^1 + \tfrac{-412568}{1953125}) e_8\,.
\end{aligned}
$$

which yields in the expansions

$$
\begin{aligned}
\tilde{e}_0 &= 1 + (-\tfrac{21}{25} v^3 - \tfrac{13}{25} v^2 - \tfrac{8}{25} v - \tfrac{21}{25}) q^1 \\
&+ (\tfrac{209}{125} v^3 + \tfrac{131}{125} v^2 + \tfrac{16}{25} v - \tfrac{502}{125}) q^2 \\
&+ (\tfrac{523}{625} v^3 + \tfrac{306}{625} v^2 + \tfrac{187}{625} v + \tfrac{3399}{625}) q^3 \\
&+ (-\tfrac{193}{125} v^3 - \tfrac{651}{625} v^2 - \tfrac{348}{625} v - \tfrac{1073}{625}) q^4 \\
&+ (-\tfrac{83719}{78125} v^3 - \tfrac{25967}{78125} v^2 - \tfrac{27127}{78125} v - \tfrac{83624}{78125}) q^5 \\
&+ (-\tfrac{760188}{390625} v^3 - \tfrac{527357}{390625} v^2 - \tfrac{77863}{78125} v - \tfrac{427501}{390625}) q^6 + \dots\,, \\
\tilde{e}_1 &= (-\tfrac{1}{5} a^4 - \tfrac{1}{5} a^3 - \tfrac{1}{5} a^2 + \tfrac{21}{25} \zeta_5^3 - \tfrac{13}{25} \zeta_5^2 + \tfrac{8}{25} \zeta_5 - \tfrac{13}{25}) q^1 \\
&+ (-\tfrac{3}{25} a^4 - \tfrac{11}{25} a^3 - \tfrac{24}{25} a^2 - \tfrac{11}{5} a - \tfrac{209}{125} \zeta_5^3 + \tfrac{131}{125} \zeta_5^2 - \tfrac{16}{25} \zeta_5 - \tfrac{312}{125}) q^2 \\
&+ (-\tfrac{2}{125} a^4 - \tfrac{18}{125} a^3 - \tfrac{43}{125} a^2 - \tfrac{22}{25} a - \tfrac{523}{625} \zeta_5^3 + \tfrac{306}{625} \zeta_5^2 - \tfrac{187}{625} \zeta_5 - \tfrac{601}{625}) q^3 \\
&+ (-\tfrac{472}{625} a^4 - \tfrac{621}{625} a^3 - \tfrac{849}{625} a^2 - \tfrac{187}{125} a + \tfrac{193}{125} \zeta_5^3 - \tfrac{651}{625} \zeta_5^2 + \tfrac{348}{625} \zeta_5 - \tfrac{10911}{3125}) q^4 \\
&+ (-\tfrac{1764}{3125} a^4 - \tfrac{2444}{3125} a^3 - \tfrac{3789}{3125} a^2 - \tfrac{1078}{625} a + \tfrac{83719}{78125} \zeta_5^3 - \tfrac{25967}{78125} \zeta_5^2 + \tfrac{27127}{78125} \zeta_5 - \tfrac{224749}{78125}) q^5 \\
&+ (-\tfrac{83657}{78125} a^4 - \tfrac{109274}{78125} a^3 - \tfrac{147876}{78125} a^2 - \tfrac{7854}{3125} a + \tfrac{760188}{390625} \zeta_5^3 - \tfrac{527357}{390625} \zeta_5^2 + \tfrac{77863}{78125} \zeta_5 - \tfrac{2211071}{390625}) q^6 + \dots\,, \\
\tilde{e}_2 &= (-\tfrac{1}{5} \zeta_5 a^4 - \tfrac{1}{5} \zeta_5^2 a^3 - \tfrac{1}{5} \zeta_5^3 a^2 + \tfrac{21}{25} \zeta_5^3 - \tfrac{13}{25} \zeta_5^2 + \tfrac{8}{25} \zeta_5 - \tfrac{13}{25}) q^1 \\
&+ (-\tfrac{3}{25} \zeta_5 a^4 - \tfrac{11}{25} \zeta_5^2 a^3 - \tfrac{24}{25} \zeta_5^3 a^2 + \left(\tfrac{11}{5} \zeta_5^3 + \tfrac{11}{5} \zeta_5^2 + \tfrac{11}{5} \zeta_5 + \tfrac{11}{5}\right) a - \tfrac{209}{125} \zeta_5^3 + \tfrac{131}{125} \zeta_5^2 - \tfrac{16}{25} \zeta_5 - \tfrac{312}{125}) q^2 \\
&+ (-\tfrac{2}{125} \zeta_5 a^4 - \tfrac{18}{125} \zeta_5^2 a^3 - \tfrac{43}{125} \zeta_5^3 a^2 + \left(\tfrac{22}{25} \zeta_5^3 + \tfrac{22}{25} \zeta_5^2 + \tfrac{22}{25} \zeta_5 + \tfrac{22}{25}\right) a - \tfrac{523}{625} \zeta_5^3 + \tfrac{306}{625} \zeta_5^2 - \tfrac{187}{625} \zeta_5 - \tfrac{601}{625}) q^3 \\
&+ (-\tfrac{472}{625} \zeta_5 a^4 - \tfrac{621}{625} \zeta_5^2 a^3 - \tfrac{849}{625} \zeta_5^3 a^2 + \left(\tfrac{187}{125} \zeta_5^3 + \tfrac{187}{125} \zeta_5^2 + \tfrac{187}{125} \zeta_5 + \tfrac{187}{125}\right) a + \tfrac{193}{125} \zeta_5^3 - \tfrac{651}{625} \zeta_5^2 + \tfrac{348}{625} \zeta_5 - \tfrac{10911}{3125}) q^4 \\
&+ (-\tfrac{1764}{3125} \zeta_5 a^4 - \tfrac{2444}{3125} \zeta_5^2 a^3 - \tfrac{3789}{3125} \zeta_5^3 a^2 + \left(\tfrac{1078}{625} \zeta_5^3 + \tfrac{1078}{625} \zeta_5^2 + \tfrac{1078}{625} \zeta_5 + \tfrac{1078}{625}\right) a \\
&+ \tfrac{83719}{78125} \zeta_5^3 - \tfrac{25967}{78125} \zeta_5^2 + \tfrac{27127}{78125} \zeta_5 - \tfrac{224749}{78125}) q^5
\end{aligned}
$$

$$
\begin{aligned}
&+ (-\tfrac{83657}{78125}\zeta_5 a^4 - \tfrac{109274}{78125}\zeta_5^2 a^3 - \tfrac{147876}{78125}\zeta_5^3 a^2 + \left(\tfrac{7854}{3125}\zeta_5^3 + \tfrac{7854}{3125}\zeta_5^2 + \tfrac{7854}{3125}\zeta_5 + \tfrac{7854}{3125}\right) a \\
&+ \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2 + \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})q^6 + \dots, \\
\tilde{e}_3 &= (\left(\tfrac{1}{5}\zeta_5^3 + \tfrac{1}{5}\zeta_5^2 + \tfrac{1}{5}\zeta_5 + \tfrac{1}{5}\right) a^4 - \tfrac{1}{5}\zeta_5^3 a^3 - \tfrac{1}{5}\zeta_5^2 a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})q^1 \\
&+ (\left(\tfrac{3}{25}\zeta_5^3 + \tfrac{3}{25}\zeta_5^2 + \tfrac{3}{25}\zeta_5 + \tfrac{3}{25}\right) a^4 - \tfrac{11}{25}\zeta_5^3 a^3 - \tfrac{24}{25}\zeta_5^2 a^2 - \tfrac{11}{5}\zeta_5 a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})q^2 \\
&+ (\left(\tfrac{2}{125}\zeta_5^3 + \tfrac{2}{125}\zeta_5^2 + \tfrac{2}{125}\zeta_5 + \tfrac{2}{125}\right) a^4 - \tfrac{18}{125}\zeta_5^3 a^3 - \tfrac{43}{125}\zeta_5^2 a^2 - \tfrac{22}{25}\zeta_5 a - \tfrac{523}{625}\zeta_5^3 + \tfrac{306}{625}\zeta_5^2 - \tfrac{187}{625}\zeta_5 - \tfrac{601}{625})q^3 \\
&+ (\left(\tfrac{472}{625}\zeta_5^3 + \tfrac{472}{625}\zeta_5^2 + \tfrac{472}{625}\zeta_5 + \tfrac{472}{625}\right) a^4 - \tfrac{621}{625}\zeta_5^3 a^3 - \tfrac{849}{625}\zeta_5^2 a^2 - \tfrac{187}{125}\zeta_5 a + \tfrac{193}{125}\zeta_5^3 - \tfrac{651}{625}\zeta_5^2 + \tfrac{348}{625}\zeta_5 - \tfrac{10911}{3125})q^4 \\
&+ (\left(\tfrac{1764}{3125}\zeta_5^3 + \tfrac{1764}{3125}\zeta_5^2 + \tfrac{1764}{3125}\zeta_5 + \tfrac{1764}{3125}\right) a^4 - \tfrac{2444}{3125}\zeta_5^3 a^3 - \tfrac{3789}{3125}\zeta_5^2 a^2 - \tfrac{1078}{625}\zeta_5 a + \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 + \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})q^5 \\
&+ (\left(\tfrac{83657}{78125}\zeta_5^3 + \tfrac{83657}{78125}\zeta_5^2 + \tfrac{83657}{78125}\zeta_5 + \tfrac{83657}{78125}\right) a^4 - \tfrac{109274}{78125}\zeta_5^3 a^3 - \tfrac{147876}{78125}\zeta_5^2 a^2 - \tfrac{7854}{3125}\zeta_5 a + \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2 \\
&+ \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})q^6 + \dots, \\
\tilde{e}_4 &= (-\tfrac{1}{5}\zeta_5^2 a^4 + \left(\tfrac{1}{5}\zeta_5^3 + \tfrac{1}{5}\zeta_5^2 + \tfrac{1}{5}\zeta_5 + \tfrac{1}{5}\right) a^3 - \tfrac{1}{5}\zeta_5 a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})q^1 \\
&+ (-\tfrac{3}{25}\zeta_5^2 a^4 + \left(\tfrac{11}{25}\zeta_5^3 + \tfrac{11}{25}\zeta_5^2 + \tfrac{11}{25}\zeta_5 + \tfrac{11}{25}\right) a^3 - \tfrac{24}{25}\zeta_5 a^2 - \tfrac{11}{5}\zeta_5^3 a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})q^2 \\
&+ (-\tfrac{2}{125}\zeta_5^2 a^4 + \left(\tfrac{18}{125}\zeta_5^3 + \tfrac{18}{125}\zeta_5^2 + \tfrac{18}{125}\zeta_5 + \tfrac{18}{125}\right) a^3 - \tfrac{43}{125}\zeta_5 a^2 - \tfrac{22}{25}\zeta_5^3 a - \tfrac{523}{625}\zeta_5^3 + \tfrac{306}{625}\zeta_5^2 - \tfrac{187}{625}\zeta_5 - \tfrac{601}{625})q^3 \\
&+ (-\tfrac{472}{625}\zeta_5^2 a^4 + \left(\tfrac{621}{625}\zeta_5^3 + \tfrac{621}{625}\zeta_5^2 + \tfrac{621}{625}\zeta_5 + \tfrac{621}{625}\right) a^3 - \tfrac{849}{625}\zeta_5 a^2 - \tfrac{187}{125}\zeta_5^3 a + \tfrac{193}{125}\zeta_5^3 - \tfrac{651}{625}\zeta_5^2 + \tfrac{348}{625}\zeta_5 - \tfrac{10911}{3125})q^4 \\
&+ (-\tfrac{1764}{3125}\zeta_5^2 a^4 + \left(\tfrac{2444}{3125}\zeta_5^3 + \tfrac{2444}{3125}\zeta_5^2 + \tfrac{2444}{3125}\zeta_5 + \tfrac{2444}{3125}\right) a^3 - \tfrac{3789}{3125}\zeta_5 a^2 - \tfrac{1078}{625}\zeta_5^3 a + \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 \\
&+ \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})q^5 \\
&+ (-\tfrac{83657}{78125}\zeta_5^2 a^4 + \left(\tfrac{109274}{78125}\zeta_5^3 + \tfrac{109274}{78125}\zeta_5^2 + \tfrac{109274}{78125}\zeta_5 + \tfrac{109274}{78125}\right) a^3 - \tfrac{147876}{78125}\zeta_5 a^2 - \tfrac{7854}{3125}\zeta_5^3 a + \tfrac{760188}{390625}\zeta_5^3 - \tfrac{527357}{390625}\zeta_5^2 \\
&+ \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})q^6 + \dots, \\
\tilde{e}_5 &= (-\tfrac{1}{5}\zeta_5^3 a^4 - \tfrac{1}{5}\zeta_5 a^3 + \left(\tfrac{1}{5}\zeta_5^3 + \tfrac{1}{5}\zeta_5^2 + \tfrac{1}{5}\zeta_5 + \tfrac{1}{5}\right) a^2 + \tfrac{21}{25}\zeta_5^3 - \tfrac{13}{25}\zeta_5^2 + \tfrac{8}{25}\zeta_5 - \tfrac{13}{25})q^1 \\
&+ (-\tfrac{3}{25}\zeta_5^3 a^4 - \tfrac{11}{25}\zeta_5 a^3 + \left(\tfrac{24}{25}\zeta_5^3 + \tfrac{24}{25}\zeta_5^2 + \tfrac{24}{25}\zeta_5 + \tfrac{24}{25}\right) a^2 - \tfrac{11}{5}\zeta_5^2 a - \tfrac{209}{125}\zeta_5^3 + \tfrac{131}{125}\zeta_5^2 - \tfrac{16}{25}\zeta_5 - \tfrac{312}{125})q^2 \\
&+ (-\tfrac{2}{125}\zeta_5^3 a^4 - \tfrac{18}{125}\zeta_5 a^3 + \left(\tfrac{43}{125}\zeta_5^3 + \tfrac{43}{125}\zeta_5^2 + \tfrac{43}{125}\zeta_5 + \tfrac{43}{125}\right) a^2 - \tfrac{22}{25}\zeta_5^2 a - \tfrac{523}{625}\zeta_5^3 + \tfrac{306}{625}\zeta_5^2 - \tfrac{187}{625}\zeta_5 - \tfrac{601}{625})q^3 \\
&+ (-\tfrac{472}{625}\zeta_5^3 a^4 - \tfrac{621}{625}\zeta_5 a^3 + \left(\tfrac{849}{625}\zeta_5^3 + \tfrac{849}{625}\zeta_5^2 + \tfrac{849}{625}\zeta_5 + \tfrac{849}{625}\right) a^2 - \tfrac{187}{125}\zeta_5^2 a + \tfrac{193}{125}\zeta_5^3 - \tfrac{651}{625}\zeta_5^2 + \tfrac{348}{625}\zeta_5 - \tfrac{10911}{3125})q^4 \\
&+ (-\tfrac{1764}{3125}\zeta_5^3 a^4 - \tfrac{2444}{3125}\zeta_5 a^3 + \left(\tfrac{3789}{3125}\zeta_5^3 + \tfrac{3789}{3125}\zeta_5^2 + \tfrac{3789}{3125}\zeta_5 + \tfrac{3789}{3125}\right) a^2 - \tfrac{1078}{625}\zeta_5^2 a + \tfrac{83719}{78125}\zeta_5^3 - \tfrac{25967}{78125}\zeta_5^2 \\
&+ \tfrac{27127}{78125}\zeta_5 - \tfrac{224749}{78125})q^5 \\
&+ (-\tfrac{83657}{78125}\zeta_5^3 a^4 - \tfrac{109274}{78125}\zeta_5 a^3 + \left(\tfrac{147876}{78125}\zeta_5^3 + \tfrac{147876}{78125}\zeta_5^2 + \tfrac{147876}{78125}\zeta_5 + \tfrac{147876}{78125}\right) a^2 - \tfrac{7854}{3125}\zeta_5^2 a + \tfrac{760188}{390625}\zeta_5^3 \\
&- \tfrac{527357}{390625}\zeta_5^2 + \tfrac{77863}{78125}\zeta_5 - \tfrac{2211071}{390625})q^6 + \dots, \\
\tilde{e}_6 &= (-\tfrac{8}{25}v^3 - \tfrac{34}{25}v^2 + \tfrac{26}{25}v - \tfrac{13}{25})q^1 \\
&+ (\tfrac{78}{125}v^3 + \tfrac{342}{125}v^2 - \tfrac{52}{25}v + \tfrac{131}{125})q^2
\end{aligned}
$$

$$
\begin{aligned}
&+ (\tfrac{217}{625}v^3 + \tfrac{799}{625}v^2 - \tfrac{642}{625}v + \tfrac{306}{625})q^3 \\
&+ (-\tfrac{314}{625}v^3 - \tfrac{66}{25}v^2 + \tfrac{1268}{625}v - \tfrac{651}{625})q^4 \\
&+ (-\tfrac{57752}{78125}v^3 - \tfrac{79061}{78125}v^2 + \tfrac{82559}{78125}v - \tfrac{25967}{78125})q^5 \\
&+ (-\tfrac{232831}{390625}v^3 - \tfrac{1444029}{390625}v^2 + \tfrac{179646}{78125}v - \tfrac{527357}{390625})q^6 + \dots, \\
\tilde{e}_7 &= (-\tfrac{34}{25}v^3 + \tfrac{8}{25}v^2 - \tfrac{42}{25}v + \tfrac{21}{25})q^1 \\
&+ (\tfrac{338}{125}v^3 - \tfrac{78}{125}v^2 + \tfrac{84}{25}v - \tfrac{209}{125})q^2 \\
&+ (\tfrac{859}{625}v^3 - \tfrac{217}{625}v^2 + \tfrac{1016}{625}v - \tfrac{523}{625})q^3 \\
&+ (-\tfrac{1582}{625}v^3 + \tfrac{314}{625}v^2 - \tfrac{1964}{625}v + \tfrac{193}{125})q^4 \\
&+ (-\tfrac{140311}{78125}v^3 + \tfrac{57752}{78125}v^2 - \tfrac{136813}{78125}v + \tfrac{83719}{78125})q^5 \\
&+ (-\tfrac{1131061}{390625}v^3 + \tfrac{232831}{390625}v^2 - \tfrac{335372}{78125}v + \tfrac{760188}{390625})q^6 + \dots, \\
\tilde{e}_8 &= (-\tfrac{42}{25}v^3 - \tfrac{26}{25}v^2 - \tfrac{16}{25}v + \tfrac{8}{25})q^1 \\
&+ (\tfrac{84}{25}v^3 + \tfrac{52}{25}v^2 + \tfrac{32}{25}v - \tfrac{16}{25})q^2 \\
&+ (\tfrac{1016}{625}v^3 + \tfrac{642}{625}v^2 + \tfrac{374}{625}v - \tfrac{187}{625})q^3 \\
&+ (-\tfrac{1964}{625}v^3 - \tfrac{1268}{625}v^2 - \tfrac{696}{625}v + \tfrac{348}{625})q^4 \\
&+ (-\tfrac{136813}{78125}v^3 - \tfrac{82559}{78125}v^2 - \tfrac{54254}{78125}v + \tfrac{27127}{78125})q^5 \\
&+ (-\tfrac{335372}{78125}v^3 - \tfrac{179646}{78125}v^2 - \tfrac{155726}{78125}v + \tfrac{77863}{78125})q^6 + \dots.
\end{aligned}
$$

APPENDIX B

K for all Noncongruence Passports

This section lists all computed noncongruence passports together with the corresponding defining polynomials of the number fields K.

B.1 $\mu = 7$

7T4-6.1_3.3.1_2.2.2.1-a	$v^2 - v + 1$
7T7-4.3_3.3.1_2.2.2.1-a	$\mathbb{Q}$
7T7-5.2_3.3.1_2.2.2.1-a	$\mathbb{Q}$

B.2 $\mu = 8$

8T43-8_3.3.1.1_2.2.2.1.1-a	$v^2 - 2$

B.3 $\mu = 9$

9T11-6.2.1_3.3.3_2.2.2.2.1-a	$\mathbb{Q}$
9T13-6.3_3.3.3_2.2.2.1.1.1-a	$\mathbb{Q}$
9T26-8.1_3.3.3_2.2.2.1.1.1-a	$v^2 + 2$
9T27-7.1.1_3.3.3_2.2.2.2.1-a	$\mathbb{Q}$
9T27-9_3.3.3_2.2.2.2.1-a	$\mathbb{Q}$
9T30-4.3.2_3.3.3_2.2.2.2.1-a	$\mathbb{Q}$
9T32-9_3.3.1.1.1_2.2.2.2.1-a	$v^2 - v + 1$
9T33-5.3.1_3.3.3_2.2.2.2.1-a	$\mathbb{Q}$
9T34-5.4_3.3.3_2.2.2.1.1.1-a	$\mathbb{Q}$
9T34-7.2_3.3.3_2.2.2.1.1.1-a	$\mathbb{Q}$

B.4 $\mu = 10$

10T30-10_3.3.3.1_2.2.2.2.2-a	$\mathbb{Q}$
10T30-8.1.1_3.3.3.1_2.2.2.2.2-a	$\mathbb{Q}$
10T44-6.4_3.3.3.1_2.2.2.2.1.1-a	$v^3 - v^2 - 3v - 3$
10T44-7.3_3.3.3.1_2.2.2.2.1.1-a	$v^3 - v^2 + 2v - 3$
10T44-8.2_3.3.3.1_2.2.2.2.1.1-a	$v^3 - v^2 + 2v + 2$
10T44-9.1_3.3.3.1_2.2.2.2.1.1-a	$v^6 - 3v^5 + 6v^4 - 3v^3 + 3v^2 - 3v + 2$
10T45-10_3.3.3.1_2.2.2.1.1.1.1-a	$v^5 - 2$
10T45-5.3.2_3.3.3.1_2.2.2.2.2-a	$\mathbb{Q}$
10T45-5.4.1_3.3.3.1_2.2.2.2.2-a	$\mathbb{Q}$
10T45-7.2.1_3.3.3.1_2.2.2.2.2-a	$\mathbb{Q}$

B.5 $\mu = 11$

11T7-11_3.3.3.1.1_2.2.2.2.1.1.1-a	$v^8 + 2v^6 - 3v^5 + 10v^4 - 14v^3 + 14v^2 - 8v + 1$
11T8-10.1_3.3.3.1.1_2.2.2.2.2.1-a	$v^6 - 3v^5 + 5v^4 - 5v^3 + v + 2$
11T8-6.5_3.3.3.1.1_2.2.2.2.2.1-a	$v^2 - v + 3$
11T8-7.4_3.3.3.1.1_2.2.2.2.2.1-a	$v^3 - v^2 + v + 1$
11T8-8.3_3.3.3.1.1_2.2.2.2.2.1-a	$v^2 - 22$
11T8-9.2_3.3.3.1.1_2.2.2.2.2.1-a	$v^3 + 6v - 1$

B.6 $\mu = 12$

12T157-8.4_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^2 + 2$
12T177-12_3.3.3.3_2.2.2.2.2.1.1-a	$v^2 - 3$
12T178-12_3.3.3.1.1.1_2.2.2.2.2.1.1-a	$\mathbb{Q}$
12T182-10.2_3.3.3.1.1.1_2.2.2.2.2.2-a	$\mathbb{Q}$
12T182-8.4_3.3.3.1.1.1_2.2.2.2.2.2-a	$\mathbb{Q}$
12T218-10.1.1_3.3.3.3_2.2.2.2.2.1.1-a	$v^2 - v - 1$
12T218-12_3.3.3.3_2.2.2.2.2.1.1-a	$v^2 - 3$
12T251-6.4.2_3.3.3.3_2.2.2.2.2.1.1-a	$\mathbb{Q}$
12T253-12_3.3.3.3_2.2.2.1.1.1.1.1.1-a	$\mathbb{Q}$
12T284-9.3_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^3 - 3v - 4$
12T291-8.3.1_3.3.3.3_2.2.2.2.2.1.1-a	$\mathbb{Q}$
12T295-10.2_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^2 + 5$
12T295-11.1_3.3.3.1.1.1_2.2.2.2.2.2-a	$v^2 - v + 3$
12T295-11.1_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^2 - v + 3$
12T300-11.1_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^3 - v^2 + 4v + 2$
12T300-7.5_3.3.3.1.1.1_2.2.2.2.2.2-a	$\mathbb{Q}$

12T300-7.5_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^2 - 7$
12T301-12_3.3.3.1.1.1_2.2.2.2.2.2.1.1-a	$v^9 - 9v^7 - 12v^6 + 39v^5 + 48v^4 + 21v^3 - 36v^2 - 156v - 192$
12T301-5.4.3_3.3.3.3_2.2.2.2.2.2.1.1-a	$\mathbb{Q}$
12T301-6.5.1_3.3.3.3_2.2.2.2.2.2.1.1-a	$v^2 - 3$
12T301-7.3.2_3.3.3.3_2.2.2.2.2.2.1.1-a	$v^2 - 7$
12T301-7.4.1_3.3.3.3_2.2.2.2.2.2.1.1-a	$v^2 - v + 2$
12T301-8.3.1_3.3.3.3_2.2.2.2.2.2.1.1-a	$v^2 + 2$
12T301-9.2.1_3.3.3.3_2.2.2.2.2.2.1.1-a	$v^3 - 3v - 4$
12T60-6.6_3.3.3.3_2.2.2.2.1.1.1.1-a	$v^2 - 3$

B.7 $\mu = 13$

13T5-6.6.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^2 - v + 1$
13T7-13_3.3.3.3.1_2.2.2.2.1.1.1.1.1-a	$v^4 - v^3 + 2v^2 + 4v + 3$
13T8-10.2.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^6 - 2v^5 + 5v^2 + 5$
13T8-11.1.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^6 - 3v^5 + 9v^4 - 11v^3 + 30v^2 - 18v + 46$
13T8-13_3.3.3.1.1.1.1_2.2.2.2.2.2.1-a	$v^5 - v^4 + v^3 - 5v^2 + 2v - 1$
13T8-13_3.3.3.3.1_2.2.2.1.1.1.1.1-a	$v^{10} - v^9 + 3v^8 - 3v^7 + 5v^6 - 4v^5 + 4v^4 - 18v^3 + 4v^2 + 12v + 9$
13T8-13_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^{10} - 3v^9 + 5v^8 - 12v^7 + 24v^6 - 46v^5 + 68v^4 - 60v^3 + 96v^2 - 144v + 72$
13T8-5.4.4_3.3.3.3.1_2.2.2.2.2.2.1-a	$\mathbb{Q}$
13T8-5.5.3_3.3.3.3.1_2.2.2.2.2.2.1-a	$\mathbb{Q}$
13T8-6.4.3_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^2 - v - 3$
13T8-6.5.2_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^2 - v + 10$
13T8-7.3.3_3.3.3.3.1_2.2.2.2.2.2.1-a	$\mathbb{Q}$
13T8-7.4.2_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^3 + 4v - 2$
13T8-7.5.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^4 - v^3 + 9v^2 + 5v + 4$
13T8-8.3.2_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^3 - v - 2$
13T8-8.4.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^4 - 3v^2 - 2v + 6$
13T8-9.2.2_3.3.3.3.1_2.2.2.2.2.2.1-a	$\mathbb{Q}$
13T8-9.3.1_3.3.3.3.1_2.2.2.2.2.2.1-a	$v^4 - v^3 + 3v^2 + v + 2$
13T9-10.3_3.3.3.3.1_2.2.2.2.2.1.1.1-a	$v^{10} - 3v^9 - 4v^8 - 2v^7 + 29v^6 + 38v^5 + 3v^4 - v^3 + 22v^2 + 6v + 36$
13T9-11.2_3.3.3.3.1_2.2.2.2.2.1.1.1-a	$v^{10} - v^9 + 3v^8 + 4v^7 + 41v^6 + 224v^5 + 515v^4 + 828v^3 + 777v^2 + 348v + 164$
13T9-12.1_3.3.3.3.1_2.2.2.2.2.1.1.1-a	$v^{20} - 10v^{19} + 64v^{18} - 284v^{17} + 952v^{16} - 2510v^{15} + 5396v^{14} - 9710v^{13}$
	$+14927v^{12} - 19828v^{11} + 22932v^{10} - 23186v^9 + 20570v^8$
	$-15896v^7 + 10384v^6 - 5396v^5 + 2200v^4 - 804v^3 + 288v^2 - 48v + 18$
13T9-8.5_3.3.3.3.1_2.2.2.2.2.1.1.1-a	$v^7 - 2v^6 - 2v^5 - 4v^4 + 3v^3 - 2v^2 - 6v - 4$
13T9-9.4_3.3.3.3.1_2.2.2.2.2.1.1.1-a	$v^9 - 3v^8 - 6v^7 + 30v^6 - 90v^4 + 86v^3 + 30v^2 - 9v - 87$

B.8 $\mu = 14$

14T16-8.4.2_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^2 - 2$
14T33-7.7_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$v^4 - v^3 + 3v^2 - 4v + 2$
14T39-12.1.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^2 - 3$
14T39-14_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^3 - v^2 - 2v + 1$
14T46-10.2.2_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T46-6.4.4_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T54-5.5.4_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T54-8.3.3_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T55-10.4_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$\mathbb{Q}$
14T55-8.6_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$\mathbb{Q}$
14T58-14_3.3.3.1.1.1.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T62-10.4_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$v^{13} - v^{12} - 5v^{11} + 15v^{10} - 36v^8 + 51v^7 - 15v^6 - 20v^5$ $+80v^4 - 169v^3 + 189v^2 - 105v + 25$
14T62-11.3_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$v^{12} - v^{11} + 5v^{10} - 43v^9 + 20v^8 - 249v^7 - 63v^6 - 589v^5$ $-43v^4 - 79v^3 - 139v^2 - 116v + 36$
14T62-12.2_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$v^{14} + 9v^{12} - 2v^{11} + 28v^{10} - 6v^9 + 40v^8 + 32v^7 + 40v^6 + 144v^5$ $+36v^4 + 132v^3 - 18v^2 - 36v - 18$
14T62-9.5_3.3.3.3.1.1_2.2.2.2.2.2.1.1-a	$v^{10} - v^9 - 15v^8 + 78v^6 + 48v^5 - 150v^4 - 294v^3 - 267v^2$ $-130v - 26$
14T63-10.3.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T63-11.2.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^3 - v^2 + v + 1$
14T63-6.5.3_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T63-7.6.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T63-8.5.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^3 - v^2 + 2v + 2$
14T63-9.3.2_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$\mathbb{Q}$
14T63-9.4.1_3.3.3.3.1.1_2.2.2.2.2.2.2-a	$v^2 - v + 1$

B.9 $\mu = 15$

15T100-6.4.3.2_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T100-6.4.4.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T100-8.3.2.2_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T101-12.3_3.3.3.3.3_2.2.2.2.1.1.1.1.1.1-a	$v^8 + 4v^6 - 2v^5 + 5v^4 - 10v^3 - 10v - 5$
15T101-9.6_3.3.3.3.3_2.2.2.2.1.1.1.1.1.1-a	$v^6 - 3v^5 - 30v^3 + 45v^2 - 135v + 90$
15T103-10.3.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^5 + 5v^3 - 5v^2 + 2$
15T103-10.4.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^6 - 3v^5 + 10v^2 - 4v + 2$
15T103-11.2.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^2 - 55$
15T103-11.3.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^{10} - 3v^9 + 25v^8 - 81v^7 + 244v^6 - 732v^5 + 1400v^4$ $-2772v^3 + 4928v^2 - 3696v + 8624$
15T103-12.2.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^{10} - 5v^9 + 9v^8 - 15v^6 - 9v^5 + 123v^4$ $-270v^3 + 315v^2 - 205v + 65$

15T103-13.1.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^{10} - v^9 - 8v^8 + 16v^7 + 70v^6 + 2v^5 - 232v^4$ $+144v^3 + 889v^2 + 1375v + 880$
15T103-6.5.4_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^3 - v^2 - 3v - 3$
15T103-7.4.4_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^2 - v - 1$
15T103-7.5.3_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^4 - v^3 - 6v - 6$
15T103-7.6.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^4 - 5v^2 - 20$
15T103-7.7.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^3 - v^2 + 5v - 13$
15T103-8.4.3_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^4 - 20v - 45$
15T103-8.5.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^3 - v^2 + 2v + 2$
15T103-8.6.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^7 - 3v^6 - 2v^5 + 8v^4 - 14v^3 + 20v^2 + 45v + 17$
15T103-9.4.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^4 - v^3 + 6v^2 - 6v + 6$
15T103-9.5.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^7 - 9v^4 + 15v^3 + 30v^2 + 22v - 3$
15T104-10.5_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^5 - v^4 + 4v^3 - 5v^2 + 5v - 5$
15T104-10.5_3.3.3.3.3_2.2.2.2.2.1.1.1.1.1-a	$v^5 + 5v^3 - 5v^2 + 2$
15T104-11.2.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^3 - v^2 + v + 1$
15T104-11.4_3.3.3.3.3_2.2.2.2.2.1.1.1.1.1-a	$v^6 - 3v^5 + 10v^4 + 10v^3 - 75v^2 + 233v - 224$
15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 - v - 1$
15T104-11.4_3.3.3.3.3_2.2.2.2.2.2.2.1-b	$\mathbb{Q}$
15T104-12.1.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 - v + 1$
15T104-12.3_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^6 - 3v^5 + 15v^4 - 20v^3 + 30v^2 + 40$
15T104-12.3_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^4 - 3v^2 - 9$
15T104-13.2_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^{10} - 3v^9 + 5v^8 - 11v^7 + 6v^6 + 53v^5 - 2v^4 + 30v^3 + 56v^2 - 126v + 126$
15T104-13.2_3.3.3.3.3_2.2.2.2.2.1.1.1.1.1-a	$v^7 - v^6 - 20v^5 + 45v^4 + 170v^3 - 230v^2 - 280v + 600$
15T104-13.2_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^5 - 2v^4 + 7v^3 - 14v^2 + 16v - 12$
15T104-14.1_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^{20} - 8v^{19} + 38v^{18} - 130v^{17} + 331v^{16} - 630v^{15} + 1113v^{14}$ $-2003v^{13} + 3977v^{12} - 7591v^{11} + 11365v^{10} - 12945v^9 + 13293v^8$ $-9674v^7 + 8192v^6 - 8038v^5 + 8392v^4$ $-5748v^3 + 6500v^2 - 168v + 1680$
15T104-5.4.3.3_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T104-6.5.3.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 - v - 1$
15T104-7.4.3.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T104-7.5.2.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^3 + 2v - 2$
15T104-7.6.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 - v + 1$
15T104-8.3.3.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T104-8.5.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 + 1$
15T104-8.7_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^7 - v^6 - 14v^5 + 30v^4 + 60v^3 - 120v^2 - 180v + 420$
15T104-8.7_3.3.3.3.3_2.2.2.2.2.1.1.1.1.1-a	$v^5 - 5v^3 - 70v^2 - 90v - 28$
15T104-8.7_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 + 5$
15T104-9.3.2.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^3 - 2$
15T104-9.4.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T104-9.6_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^5 - 5v^3 - 10v^2 + 15v + 20$
15T104-9.6_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$v^2 - v - 1$

15T12-6.6.3_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^2 - v - 1$
15T14-10.2.2.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T39-15_3.3.3.3.3_2.2.2.2.1.1.1.1.1.1.1-a	$\mathbb{Q}$
15T48-5.4.4.2_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T61-10.5_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T76-10.5_3.3.3.3.1.1.1_2.2.2.2.2.2.2.1-a	$v^2 - v + 4$
15T92-9.3.3_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$v^3 - 12v - 14$
15T94-10.3.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$\mathbb{Q}$
15T94-10.4.1_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$\mathbb{Q}$
15T96-10.3.1.1_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T96-6.5.2.2_3.3.3.3.3_2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
15T99-6.5.4_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$\mathbb{Q}$
15T99-8.5.2_3.3.3.3.3_2.2.2.2.2.2.1.1.1-a	$\mathbb{Q}$

B.10 $\mu = 16$

16T1502-12.4_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 1$
16T1789-12.4_3.3.3.3.3.1_2.2.2.2.2.2.1.1.1.1-a	$v^2 + 2$
16T1789-12.4_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - 6$
16T1790-12.2.1.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1790-6.4.3.3_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1799-14.2_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 2$
16T1877-9.3.2.2_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1877-9.3.3.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1881-16_3.3.3.3.3.1_2.2.2.2.2.1.1.1.1.1.1-a	$v^2 + 2$
16T1881-8.4.4_3.3.3.3.3.1_2.2.2.2.2.2.2.1.1-a	$v^6 - 2v^5 - v^4 + 2v^2 + 4v + 2$
16T1900-12.4_3.3.3.3.3.1_2.2.2.2.2.2.1.1.1.1-a	$v^6 + 3v^4 - 4v^3 - 2$
16T1949-10.6_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-10.3.2.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 + 1$
16T1953-10.6_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - 6$
16T1953-11.2.2.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-11.3.1.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-11.5_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 14$
16T1953-11.5_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - 6$
16T1953-12.2.1.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 1$
16T1953-13.1.1.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 1$
16T1953-13.3_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-13.3_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^3 - v - 2$
16T1953-14.2_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^5 - v^4 + 4v^3 + 10v^2 - v + 31$
16T1953-15.1_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$v^5 - 3$
16T1953-15.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^{10} - 5v^7 + 5v^5 + 10v^4 - 20v^2 + 16$
16T1953-5.5.4.2_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$

16T1953-7.5.2.2_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-7.5.3.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-8.6.1.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^2 - v + 1$
16T1953-9.4.2.1_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^3 + 3v - 2$
16T1953-9.7_3.3.3.3.1.1.1.1_2.2.2.2.2.2.2.2-a	$\mathbb{Q}$
16T1953-9.7_3.3.3.3.3.1_2.2.2.2.2.2.2.2-a	$v^3 - 6v - 16$
16T1954-10.4.2_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{13} - 3v^{12} - 14v^{11} + 50v^{10} + 65v^9 - 307v^8 - 144v^7$ $+688v^6 + 320v^5 + 320v^4 - 1000v^3 + 2200v^2 - 3500v + 2500$
16T1954-11.3.2_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{15} - 4v^{14} + 72v^{13} - 223v^{12} + 1557v^{11} - 2892v^{10} + 11308v^9$ $-7535v^8 + 40931v^7 + 29964v^6 + 10824v^5 + 363803v^4$ $+103367v^3 + 125796v^2 + 933636v - 137005$
16T1954-12.2.2_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^6 - 3v^5 + 9v^4 - 6v^2 - 6v - 6$
16T1954-12.3.1_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{24} - 6v^{23} + 21v^{22} - 60v^{21} + 184v^{20} - 478v^{19} + 651v^{18} - 1220v^{17}$ $+2230v^{16} + 1226v^{15} - 947v^{14} + 804v^{13} - 7092v^{12} - 6862v^{11}$ $+3971v^{10} - 15340v^9 + 7975v^8 + 36044v^7 + 7134v^6$ $+14896v^5 + 13928v^4 - 2372v^3 + 3970v^2 - 584v + 22$
16T1954-6.5.5_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^4 + 4v^2 - 2$
16T1954-6.6.4_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^4 + 4v^2 - 2$
16T1954-8.5.3_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^9 - 4v^8 + 8v^6 - 16v^3 + 18v - 8$
16T1954-8.6.2_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{11} - 3v^{10} + 13v^9 - 27v^8 + 54v^7 - 98v^6 + 94v^5$ $-242v^4 + 77v^3 - 455v^2 - 7v - 463$
16T1954-8.7.1_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{19} - 6v^{18} + 17v^{17} - 22v^{16} + 24v^{15} - 322v^{14} - 345v^{13} - 1002v^{12}$ $+1070v^{11} - 718v^{10} + 6969v^9 - 4934v^8 + 6808v^7$ $-27018v^6 + 211v^5 - 60738v^4 + 90045v^3$ $-122168v^2 + 333600v - 362592$
16T1954-9.5.2_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{10} - 4v^9 - 12v^8 + 88v^7 + 8v^6 - 384v^5 - 944v^4$ $+4640v^3 + 8010v^2 + 5400v + 1800$
16T1954-9.6.1_3.3.3.3.3.1_2.2.2.2.2.2.1.1-a	$v^{18} - 6v^{17} + 18v^{16} - 36v^{15} - 21v^{14} + 312v^{13} - 311v^{12}$ $-780v^{11} + 2469v^{10} - 5892v^9 + 3657v^8 + 12180v^7 - 6851v^6$ $-11400v^5 - 1137v^4 + 5820v^3 + 26802v^2 + 23418v - 40563$

B.11 $\mu = 17$

17T6-15.1.1_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$\mathbb{Q}$
17T6-17_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^2 - v - 4$
17T9-10.4.3_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^8 - 2v^7 - 9v^6 + 33v^5 + 135v^4 - 579v^3 + 1018v^2 - 524v - 297$
17T9-10.5.2_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^9 - 2v^8 - 14v^7 + 32v^6 + 33v^5 - 246v^4$ $+387v^3 - 406v^2 + 308v - 168$
17T9-10.6.1_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^{16} - 4v^{15} + 24v^{14} - 61v^{13} + 232v^{12} - 482v^{11} + 2121v^{10} - 2469v^9$ $+12216v^8 - 14432v^7 + 35132v^6 - 57576v^5 + 60704v^4$ $-96296v^3 + 87652v^2 - 45396v + 67152$

17T9-11.3.3_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^4 - v^3 - 3v^2 - 63v - 189$
17T9-13.2.2_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^6 - v^5 - 60v^4 - 20v^3 + 2290v^2 + 8844v - 48554$
17T9-13.3.1_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^{18} - 5v^{17} + 9v^{16} - 29v^{15} + 253v^{14} - 1009v^{13} + 1995v^{12}$ $-2831v^{11} + 7366v^{10} - 19570v^9 + 30844v^8 - 34272v^7$ $+70708v^6 - 101660v^5$ $+109820v^4 - 125052v^3 + 186184v^2 - 28424v + 61744$
17T9-6.6.5_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^3 - v^2 + 6v - 12$
17T9-7.5.5_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^4 - v^3 + 5v^2 - 5v + 2$
17T9-7.7.3_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^3 - v^2 + 6v + 5$
17T9-8.5.4_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^8 - 10v^6 - 6v^5 + 30v^4 + 48v^3 - 9v^2 - 104v - 84$
17T9-8.6.3_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^8 - 4v^7 - 4v^6 + 14v^5 + 29v^4 - 18v^3 - 12v^2 + 50v - 83$
17T9-9.4.4_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^5 - v^4 - 2v^3 - 3v + 3$
17T9-9.5.3_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^7 - 2v^6 + 10v^5 - 116v^4 - 347v^3$ $+ 1834v^2 + 7956v + 15984$
17T9-9.6.2_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^9 - 3v^8 - 9v^7 - 4v^6 + 72v^5 + 192v^4$ $+399v^3 + 567v^2 + 477v + 144$
17T9-9.7.1_3.3.3.3.3.1.1_2.2.2.2.2.2.2.2.1-a	$v^{17} - 5v^{16} + 13v^{15} - 87v^{14} + 306v^{13} - 228v^{12} - 548v^{11}$ $+520v^{10} + 5086v^9 - 39042v^8 + 192006v^7$ $-665778v^6 + 1584964v^5$ $-2451512v^4 + 2221792v^3 - 986400v^2 + 98496v - 31104$

APPENDIX C

Traces of Real Singular Moduli: $\mathrm{Tr}_d(j_m)$

We list numerical approximations of traces of real singular moduli for $m = 0, \ldots, 9$ here to 500 digits precision. All data to full precision can be found in [102].

C.1 $d = 5$

```
#m Tr_d(j_m)
0   0.13700342101073800401806869981047721645341271206232290029311678094301915508145
     43190105439497636402604087305788910455526028024170979998082595364831820300377 38
     18760008275836535217685307913646104082681385030408647705565757924204287230559 00
     09852782092809034182192288354536775445656175872479815183046706190980840825270 59
     30167305214728185219903711748916439845097325196403483614430205903684893589156 86
     18067493255862811218483945487145305562562817617951767327539546620865349143079 19
     2952386177580114794968611 63
1   -5.1616294321261094420704640267223194956204840782420440329318357531803848208142
     39933685794097909250030929428892781873115077079769443475566734303109697833364 53
     39414539965586899173492417845784626507985895761575514422391209275482401977395 56
     91906251345289162735526773226651697645614247772898575484645341391078942045839 88
     13078366054336941459930851941503391581190764286478729451396884075912505369920 50
     83215675850469513678830686711122932328690175649366899897752848293915087854225 32
     3232296668415098107485747 11
2   -11.563432179935846669452999468309455074587279310650570710089589825377632546248
     19814168988269334813482212750751572395177669228001178883810061313613516283789 09
     65733538139535444784730644951955135647605665565329834842350383373671031086949 57
     36035975658255588755573932322696215561721834722576546355051376699509133134119 95
     38384347522678414968785430835501730170637967547244660733546627264098186380168 12
     56624723684251505563704993913877291668654549920458547768632321081043850511247 10
     5854038183908094064723188 97
3   -14.312248899223134316717113661022428117312100643750934198909227935992970202113
     91334922437120280525630688780426727484942254258424205758650047702240433772850 12
     30732623211391508790660842464424320176677946408171315480379581312136290688246 34
     57107747176921837710860633314598207555576132486663581100898111750197928835568 32
     44998143025459386597986904021202958401276528073561856296341774118493481117247 25
     58416670117906144527587941228124666702066661259631955166142231840565678862978 35
     3769465873751852242507462 51
```

```
4   -25.976338683745917531487238559322233794832938345745612829817991118243152390386
     5482486773976910104462419358040327472881684276719097015890192632996599132285328
```

C.2 $d = 8$

```
#m Tr_d(j_m)
0   0.19837875525590238413683140097291895202255006822603887292168327943980160303232
     6759600692376772926352940076225627215512292362298090232635448878766326766984891
```

1039674461434406189822576333559655725898782429874105365338826575860011325786999
7623671196308602601717635865851438161863358040823754312559692779510839822691010
3500968574074587516489633186303220611294370648855896301297725479229317788792274
369203991084277673063147078

1 -6.7661258446850764449092608542817582553035133218397456657385268260417793461793
7466859616219313866362512412073496898429485430391941920388783917608284971808030
9164693170328562672908529743240515629017172214346992979814988073896759456751455
2480358073599134992618280919541606752267497377545744778530908592234711414466370
6857615991556376713940680210151605985740560675449819054813708827545180682523188
4585703719572265144717654469280445835734121106618587613341914901393257801682931
944364034156789179048324581

2 -17.924341174746983909093345835054306113753752154713513081273469984798617700160
2339209018306950645156318312284183984632635217444644041162594047809056712633014
4841093376901667545013163482480636689991874595181212623760001403792148849355779
1613102009980161656245175900189377763365054387903041160593621466581628692207382
0448309312892147786050106912941008486554759567102189797364719140372165166340974
2322529071230433347479593130901078234057383488012167745521528797136007163999603
577205842282660906262067882

3 -19.501820695797172112816961817564438229283598641586185490131440008180300112203
7309534641202360267040624745839790763325305671057101321858537875071578019777573
0289399009330397872372883194194265262880536373914850803856647434211678684016631
6740210384808392165507329768838919844499651552417037721782467294675190142126399
5942354122678855314920278264980620394826266851253978829313990435433485873124112
3088616580421543281011923398119948277157465431811196233895251690953830758262210
643729453758190140784202449

4 -38.331805159587089049348047190168458776552882838536775144770675296460623793550
6500174095556067447072022701026133419956441444953575084125491718961695569834364
4800239627616302610772054574489952593195897363718329563246457860154071677385594
1679432515198828042685214464148424792462733918960582786990768760277925132805129
5604456209405938438041680343173273689276431758001529194968582552804655254429739
8486771109673494397475517081907041824227529394821510446220358232047790488228776
127634072922079283680870469

5 -27.403356758458077902593880953055157714631956596638859465099987984860300561461
9585805161535422932941428931978358484917601410460215190869352091700071576778487
9899981801960372985545629925294359022596938206403098001443473010732013718771562
8796908328673462974442325288482375489187779529455964987316359643637139377365567
6795571579129332583323076422511528662431843088876331131961008130743289340035797
7665055856580108042733690160310229830385082478606885413613107397682543037408621
116745779347397889525776001

6 -63.880229316235209579963762421623008000266830335684141896140306587517457761710
9343948566744755684051753257713161921000060224607950805487475013564548027384158
2648765326497829729488737682293899673088686630506306373764872454917202333655443
0841130321181839461231902762696310438567478604629237171101354046942753191645463
9173134191659191856034684591706431418784158991690042030031604755650325621506106
8126808954485502469393118702387814609244490793867589751444037577137238798574194
2932080008387718178884360782

7 -36.490390795597884875963257041725552255285953066787568443887397126778157362990
7849402657525896360730495036123667768463446228740438438290134393694051351459276
6937509495777291920153725749412033110743129022671722588405057619540111068482802
5661545844258680331457146469978266249399363542993715697585046228394395433356611

```
    1242891399550422959961078096026967893554498338246217151157268524289513912104661
    3117484015073603487687544994949603926017469455670878094820314735461357562879212
    4874914924445338715798788999
8   -75.205207444535773965279314170923455085682123203606378439767076113036009375174
    4804609921448600404540042013421634797302200401885163613996234187568062546959366
    2950156007901200350818707653949100398378680342324348148121678303714151201430969
    8254740111312876921855729249990523934505071373538390495769688487480394796093956
    7485936826898322486383504961615491414718407204210120128238295390276557325242557
    7775472274469626495078588662036472257222832219354152269898381308759569866824282
    4947445583348092838201783 24
9   -66.166740951863548131235335816144680091107446963570520403995556939352704051858
    0310755123692530680566356162742422284243835763038683929181041312109452054865747
    8280971936055251678011961798671898218570382138682789099308237514234199304890336
    3327379233212152943484820289882576841553402794519425107058892025271363409095298
    2502537085883047036365035670080299289890881687007564745537114251078804672860308
    2113443347138641906362473125111858603449960678952526984119925807734012462451892
    8862170026761550077957859 76
```

C.3 $d = 12$

```
#m Tr_d(j_m)
0   0.242025647542855154835942826775220604249609025393370610544921739907099178293 13
    3649945436547755976178116198605590491255217163055387716814896296661210465852736
    5835838531707334411803293768337349928780135018081626394031752402334098173226014
    2844233489781578411737844204912576395187541645406515779721621029735253601279420
    1712617496942278110990409248635367396379680986966698527526135875666558132934535
    7396254150531445237857674252186688964406208662675466378487174795020209300553797
    9371100415823247158036101 17
1   -8.279126461994568764891322182331986168816452803457720115099129186765407063405 0
    8436628031025522338728241683061685586887397993059186535910091208180031558266002
    0717223639179090724945947101210471550004766905418907031088819968203579662890152
    8386318190222202296746431235540164602106173590385008975681839901634933664771469
    2361159732241166190595604291270088388163510923798080045520012885954175032124940
    2998079886557700084116167152205930728761690494511344607736419425578096025404730
    164121667540640362578304342
2   -21.496009398561177193432937596041316844163434925999679179390698090214618748065
    4125879005805770098846100610376743303893081712278995482178153021496543486136462
    9462037053784664954563305329042766503057474269570913202686866491037055532554492
    6845635244931022615257264310009507358286975609604147877528532105734594877848999
    5628166957777712275377788924429898214413480965072613091828896965962417599502458
    9269541304904208284812597113847658857355905457399101422605689043329713072632030
    838173824213937000279975613
3   -24.845746832728498483341986250124351209099040572110082397040053305991531540075
    8464738548565861248485934143635324210249763546473667550776556826337170824103038
    1634251234270214594419392077406846520096372513458921858055901814714750626598544
    1624545587573495906489042830828908657906533644189009901143342398447724683104492
    9913237958300470830762836987546074450633218017573179010268438631641398526270106
    0031372335420581690875120934186432967832786003795194363957594443994972145451076
    407012658206878904885125948
```

```
4   -45.076835226801764638888846655904814496484439603010484598630256184307420010473
     7400754310138455616918821461489503721911826837502719502340603664164201284330453
     3196370891897205997337482148047915692781345799118261992659171373051801729625155
     2025628081984632929738317108848615599522236106076987469727238225598796189913883
     3996200088392676240551550227926127606314133763106833258673844920871442616629077
     9903167294279982507940333575256125096603637931383463768507269319295752424494892
     012157693900287806738354795
5   -35.175978705924172284989505433787434939472683124140260220816750347682424649087
     3362773323058185873848737947353979784994645783094467899033171705736898386760364
     0069572544945027977080227705668447782701588099394200982859857991730344829484752
     8048759129416013827489841026610769217176125469877294224672496320841847475156910
     0787406077599367401336282325444109653434859374129908727537890648522986221790104
     0505556047647177146229024188154839921766416290731085532419822099047977132028210
     466643085061309375125654644
6   -77.012536746146518973183968268187615276368394275381802069895079960800127350393
     5195521209851333121523942194298315970871808011672736791921146521860415183433758
     3273351020893580315093828248408080236565200155919476757032599756491109464152952
     1300607280239976611039445447641119828184064130604844476306066226312721700401278
     4773164209262545364276720617413282462922439167290510525704832179872818513024195
     9574642224899671414484505211607610772290968507007127129084518759291384270415951
     329628468781805062891135952
7   -43.973455189854278910234598495655000419286224485083217952136688413271550364841
     9231508435523198653747235044051346096004953911910546964992597152382693994101909
     5569405353217859314583516028040308178093253229448736490886423840239412089789148
     5794439894242313692657360146119899318624277576854904815225919760955254824151643
     8672108635183975457518351890981592250452576959135180651660523347890269160387555
     3103193469831554634005151659594549674678714536150949204759945116764497596436454
     153281860015785145438537338
8   -93.713223856442269031055669406677539811532160055363640849579008463666375975098
     3486079126886941222578613695364381481396487800918289522961385738472923672846052
     1513432803977994271194693733794479003853600251625631883202145284043558633574519
     5922410268482421288846469799783068681692971942169139295605207533719522948225222
     0175848684660289131621477114962585827029276374482831213210680581416983159841877
     3080813188048613753218239990994450231493420026690257609224497849663982318354308
     955795784970697300074400459
9   -78.172247122356433043634832169800527317635329125699368037807872172993753164121
     3755627298072551114035541239498581518402359710556670066854364282863710789403667
     3559348055766797775811239594833855039863448463045951570666675633158549286056531
     8154878881817911376739442488771024233786526022290986343012109176529867164046628
     9819971367832310728464586611811422279630626056955574609560302192547759823234458
     9988636361915214427711689076989353840427976687213429771761628704946675414687626
     902452187829804383974713957
```

C.4 $d = 13$

```
#m Tr_d(j_m)
0   0.21095522690204917004830899276501021676933190370514010311383535299500163339120
     7714481950748917170247741718684974055592557132794081539526200231343215684969394
     3273062911880693775566248717752682996152109264919664894157760731316420700446891
```

917199116552219118150859285987863487464576305780330277076175995566309425434566 2
370406535265729581405356641152858162925187443978163537853861613880572830757987 4
345395422765820044104114863939205847005857531681695667857174637328222377236060 2
40124907397973423260087065 8

1 -6.49262825186851493323057128454727497025461866668701746477915648774926227910 04
532642352209904219118396189039189387848522161370856523449337353063171919102754 6
358677966770178452462173170332682762561997075066784311032568177671676238461482 1
196617028887756031100102342962417925287434930048228286595074640729013502279784 6
844268665620491714108028503015527313508619496978363295110082319361627261367394 0
596075657272613491656316264591449492822190621054666921290083738459444950468897 9
2154957325854432909106898 93

2 -19.1642770333177024657710816715958624982489230414253384684531629704405899555 74
248928456979922287344305741440379794793305117405007755335160366966398738344701 7
508691358421931640068891680308335203311480242018911986417692382728830577308687 8
103130759882066293138819942626081559730360597626763351154267574038479161427634 6
578469821657190200707569953148661633613527811171618288716194745025252287323927 4
091084010101761563985002978002158916548990277110775274214141097084972502292624 8
8524627805288613174300261 08

3 -21.9974223434933582698639030163349433660455138305951208396097405337753519010 95
015568727711535723220185381861239919992747924315609072387675643582256538951917 5
868614500917516416990778897161477061318874080869140402494101275730080260815364 5
705006074966209364922147972829399242558752539272118677527019779287663614957791 7
424957334714826246932763622485554963039510028552287871959648570038142650778974 5
196936685095856078434681498363947204503022398042217489637265802531204511411849 5
7229242332639043830501814 38

4 -39.0529990909351387290574557280732428957051617018986962848733181921781683011 52
698335127127072258672153224021465946054802371861534498433166358804990135395739 3
046539190084113817769583917951908784674927343454104625734373181243350140914976 9
486062999621463675234689960996929071917210421861803261723784665205937023787621 1
491654436215699608332227267875925639021788618749167718637041659402594882228646 4
541826791164619965944320483905038306914915526480458566999986622612381135676502 9
1938122088699443099467758 22

5 -29.8481837418405815599645392779719160517689759489285344375159034880411800327 30
269208504294097910984066994034837680801359059479574117068169394081238825727905 6
079882480546343189999601274830019925476863396408570291500480234534342756755808 0
154233837582016593656443493700832276830176708137366824770994607014361773791133 9
824717292489658063370299489059619163918551670168427808749987638311248967178138 8
981162659905979691425838684864921201240492004926138505802875906601836536561430 7
2158148651066324273069230 84

6 -68.3979017737287711956433325620260375191232334474307464445447741105280199140 26
373206485907884641740975898850662411701442686505901078393222984651376219890845 3
638533614077084185997788449114814347191222420953571755250494698608260578216252 9
144572400657831228164584793805207330918723129000780893687415173487696796521938 2
110767425366737318041193788318328309922984173547650681713680842701633011268913 6
161493505597525454295839674240751658794676024248655802879106590992479849741493 3
1957382663630368750945931 67

7 -38.1565293380218154780356296917401773823400533135027346502875347319118116711 80
179227845147021857533190433754692913852671628157471062290582534461028101348065 4
300041301303288112219819861522591243358515773385707404044088913309700021855517 8
956355729440402681889063134211357804803479388451260467309132394980739614372997 5

```
     4763447942646352903768727101639409679595111829875399096066109910842697551620790
     4383571813276261631759533210333857834467520994378756895274884034315204094486583
     43217216047175631366976 8478
8   -80.45904361436896572172611692376119422557461366169120962661450962735027382 7548
     4067996733987091176865687383629681777202151974545408098608138608977910214457315
     3423066860226558012418396033474034391315640919197560933653516709882433846451121
     2285467336197154204332468354650084130908511739889765140005028914610094574199875
     4037537813279268912495715907403324978283762615459335187742453096602294668664601
     3745679634751181952824600426652565370881783169979989336949116563142290094897084
     08735616874896726782817 5966
9   -69.16082269016159803518989923192774443491304444034803499281990006494317261 6439
     1056294894904208739887628283794387969319025274002436852067062814694301094781586
     9873555925073655852141027725627456366083175625416066019291172547000428543189739
     2943690013969594680856224602454172151909076699589381792853620454628959624745388
     2212515338810358769457684648485045244736750264198409399103901941042828460165696
     1892975865036052720441405909139376808118928107948711307715672197812936383302392
     18060247855914151484190 7507
```

C.5 $d = 17$

```
#m Tr_d(j_m)
0    0.32342984781359049569855674560828360041434313093931898962857400715342630155133
     6869044195928633787860925988231756387132595755910840462578040035194977085660145
     1652877277142809137680663119608830772723263728478090519734867269922556047774783
     3378355418793876096341662595061398058829267644460645953404068219572054431609700
     2038092214988961943987199146373222967149833270586010113770418984989802559919063
     4864663000665061994593937557405265787121420446097788338377222521736308064523730
     902345185519688146477544642
1   -10.658279904247180234057303322336543140621123941847586681265225602338709505692
     1581347508233281216726289552489320256725118753055173486265635629804451537078125
     9409991131416858770152831298321838541227273015892916237832947583077444476723552
     9598047572994876802856215034372017026428489474220657967756988239856222126821709
     2494849838430784306074936824751665861017918463234867937470992833592060712112133
     2944411042343689324145902705936338797414557389000011186272080710410744065101179
     747086185332467051058804741
2   -28.468294738943589477494919665849709939423846676514471449935635046681645212985
     2639749272078791289336705511927688924300493555222182010100585564682648231565109
     3707731773230055471115627768510536608405119843108462659038532937282344874519847
     7969203226355284850622995522111499362213864344635102493474551964321964391016352
     4876373688446031273832106994686452123433104257722786904809707526408360238225201
     0620334724084343034565901835238154168358246091017831337669873922841321869605076
     484210824020919313752989943
3   -34.173100280173260522662680220387204401068795268643377905106231008670060845506
     8672455833895913415187282478534356293572782972264696739633019015665413945119363
     1112188742004143208055971798124867030917174257944779716990070577598979963768977
     1707846023400145632357778000008473736318773873731583886867490694257541170720904
     3328288479831639837891301122549914029438689345328152265433670480397778386554619
     1229005746814909582925263860844814067183560419909623455454637688372123524302407
     640861302621712397767429541
```

```
4   -59.98212047479068653379611704388342374136594467760015566630097338323549319695 7
    866185317793476972590062385362130529882415691307361813354833802813443034676056 4
    98634295517630636588671312602892063501314965417276279876951973288722728248022 29
    41930848032888991349361876795063758390038852790199514113091748839824161499942 77
    75088784496233618405599408442123709397282022335799596612799800720220229833728 78
    36355294097192421898735222939403258276589606998929512213122167702128376595348 72
    38753420742423713846085015 7
5   -45.95487969732467270840988990081623503555003188163889239592538071118896346006 9
    83271525467897406989293403470833312910125290371606142826405776827583606624401 50
    91122345567162905212921082580471955044524225067143415589696019499384756872878 96
    91555342677807074839503735923488018588330224791587277164736443291205528668830 68
    25674142600831845390180224782839999420385885737699960239437479961907469904138 20
    86911189139015258473659197601580213238451336525129014971869903045053641108499 79
    98638593099665372605325398 4
6   -103.6187243207649971501023325777493273043457028701425239934964674506475781228 0
    54501448494594727185979385539859193665901891863597845072653374895799882555380 51
    36809405854486907342850430997540341693912159892318926569714939260600267464565 96
    97281519120013879821905416803637486845725186611806471995477595953949280385595 78
    62517602646102260216697435292363506915844641515163361916622318668429715601821 19
    13136246716345183935226456666547696696431082009685499649438889006639334520182 57
    13454464895347601407903064 0
7   -60.16240772858137734100203601655410941099502656715224053767470520037498624229 5
    56224033792561306318645512125934332557102368553074492041112410792092378426964 92
    18553664359108138590256764031166594088255491412976748952969353669592112139539 63
    26745039063998900272942143810280739998370126493525445666480102958655385556359 82
    97920820640389351212349756952697851740397651656338590833828132278077424181457 64
    01513447672062608306478860740493406732234660985877702388844986575255442928949 00
    82075697309493754099128088 3
8   -122.6579673782616798561150048600119446504924150886898140653711314039747485421 3
    30978275908412518059526437776331975963468184222022271943359008765442129901406 48
    13529204930517093307514638606056515491644935726926154773624729908858879031067 08
    06327643460056704944291316356373855612447037917935545964214017432054313136141 51
    51613174182172393988098828088119228323574403026705579763892761449128229067079 23
    91133273242175078078127994371540798082890528909912924871538427264772203073475 68
    28479073713405108746703435 9
9   -105.6072937130210561484388376369056659904750023314424367353394322727702465829 7
    88075310655769711963183538350976883469418005794964089914133858568949094090226 32
    39770165425098269206626418236613595503664520903196571634808708170361673208346 44
    43388145019015453782894710133881908463385947332568611471185918588273877020987 89
    04919927569129070926099568060061190988555151713935129176898368891894308304890 34
    42770092999666690878690142094145489103283622944194575639978702006997500702935 52
    74373208261868772397444539 0
```

C.6 $d = 20$

```
#m Tr_d(j_m)
0   0.274006842021476008036137399620954432906825424124645800586233561886038310162 90
    863802108789952728052081746115778209110520560483419599961651907296636406007547 6
    375200165516730704353706158272922081653627700608172954111315158484085744611180 0
```

1970556418561806836438457670907355089131235174495963036609341238196168165054118
6033461042945637043980742349783287969019465039280696722886041180736978717831372
3613498651172562243696789097429061112512563523590353465507909324173069828615838
5904772355160229589937223 27

1 -8.3625308060309780557617317475158872851038816944463073715107127892790086835312
1903768783839562869242652846820425291244588467989061615683367371962243033562774
9837496068047067351039943368266799149202127570743693142294752150609635642344565
2613300396392252514563304822680692663141629749933201951757955419308513669351970
9846092064056054557389258014826034664378521987946266839343157835844718458580088
2473145634649228465794031292494792450761783742697618879203802955217679648334819
088633925374801937735881843

2 -23.931514863966991542540583040538163930330592906440135802885626224990777289131
6131288694342900885405629610846670174930876370557301886891266725210072358665764
5682695945244735532849062278767175437381654671176852264274680354205793539621893
9151223444529776592435493041427538847974200744473184813209931118972367878882967
2766847119371988568627222303637811039625907302680900327336333394865130480930230
2189037109070654344673584289020831800376148223801580155021883285342184509978035
040858443919687481189603456

3 -29.692966076354865329138496605594762066976319374011087392237406393320546248247
5011096876813113453890319844740504123154694311381099982707796503263944551160092
3760344419484308872763616458198809748246137355672477448790786708551237780847186
2914290403782468122186922715944169409841192392139276852566625507496327613788125
3014264331727357076137694305585416576926432087727524383537015861870529905718597
2950351408882939150727694369400980918764496631779255011526932423470070759708540
418046741760825840959134958

4 -50.893025980598200693925233244203608213817713233967428995267193464084034219465
0130980319222594368608901792761719572172999454660159912928750179931475666605833
4242294861246177821980769538539879394930231200394564889571932966699089180499082
5496524046156067385437403403795771304212382784970032341229751687191462841537764
3553478684999536155123195446498923219808367084507899783309015341107838924872832
1052607714367870283651814280876584949416199344370519678938466782395881379944828
279425414632456952941675980

5 -38.616992895251197638660285846288257716109313652936575931404885306832254033241
3283691441254272202890759800987134693301527446693601068157592190970618331975681
5524473886988025068998583990813206547966511692403499935431272449242945653123235
8253470801701096903770657500036647122541483153633513825754878632767556432808330
4997465296117535203741998972326133654867995710635035035659619876269469490859437
2156055627199944975165074391464676769529945837692418846728829653410986295376854
018229430247585810808504744

6 -87.453720352918139732506140995551226125284975329359770454657730432508274751226
9190110972803730887505237481481904373391923291633589334782105950514030264867717
1912001038992358613889852413723886574110118584136145814840544959004518893397570
2475887506773155247397716979063638564058578296047066327504617017100993332733306
1882194519960890235483519188512195910198210120638174946186232672364910816194913
0803660863672816955401942061853457020278432444492923963182213675638933439705569
036139948286715989640360650

7 -50.580926838451458248333485905327924980934501755850917986712555792236571734011
9406324878709393100724796278772064849214870433497676335062761901910702512452429
3174237089642743994862088361180002664813441243345979588744470082656610947963409
9325015979594741008593036405327686452830072287608194826440169558522557584384343

```
    6198939349187271079850208396439601935679682337386888057354926493256070453230826
    3173889813975620105877960597688294199928246597482515180705142850472384016509102
    94414944627394814304144l639
8   -105.81083796167665829733073163185297073403957457061034389840949668065764012481
    8919715310387334033612671557447723336219953999086721658087350405911552230360301
    5770679550924727571367043252281386601212204253987215781993878578942498287501581
    5845862001920819718182118460858936869415876961746149691066478546857196672803900
    2079805122246869291785992279069783761605892476325783296324684283282321700915650
    5742126351463764408423593895328012831981303806634109900993853435712750424423795
    8179422157500969831411419 75
9   -88.2773753505390442401559718820596495862860118350338148079219540184096492758 92
    0777617191762237754423517470533048931852033013781917946015142578018311252942494
    6546232656266453232864519540350414137291403318026158793560858592605971105297384
    2933434175242741340633458333169514184940621102285586164222603341706093740362102
    4232477607687354773095344473530800143429336597001598147545213144697873641643165
    5685579690992908543510736520193083529115850192960288732875657173276125212500273
    984862432384505514380552589
```

C.7 $d = 21$

```
#m Tr_d(j_m)
0   0.21766260722872542934540163948006412666651583216448779599930550834394685806949
    0069477252573549515946001479980322406651404870198769579463499352019914650530625
    4939641165010341495926521168313038335159757247432142870019189747933162944396185
    4074394128509100062623415897295082891368136037364176081389968864027580231882841
    4752505459379987383604614781247971355592161477474303230219417396842337973935583
    8649914241389869433595186358083122952971437755379488846074231216938808905398094
    655860689394401633263353900
1   -6.6939659437468350890685353148144044220556506749091579104237540333652260821986
    9321863904021823224025325663635956920991521770461988552139985217335535508163405
    2796020723081146915231624301071613906562580287964952272754898904133083094046732
    7098973827149687694244362276174886529357223215855156616746160482675584296161579
    9158467850695969246561080979781846658267742039472634572958209884221445449112534
    3463750029510300401789246691345723687926837162521193437006803486116714981832084
    491474813724247285364399213
2   -19.227415835390335861282523860050328298344825052018303632900903012773907829567
    8987395833322614967343298535570215607634381689674268971705468171551789186181801
    4489657946819246954385635500131502888279957764643210931188708585964856263360891
    8892606439600553813448009795017441381111522927416814029367272664965718716994580
    9382735866267105936480727964874780459050568845332911158355882495387758771150159
    8540230574429955901756977207948222091106303401390202206057097213472187366531783
    531124750836928558645486082
3   -23.612720832189125743696926709362913724675243164925861927946184223398967239375
    9484941352576076106158653148573493735353568709792435650133155408641322097057570
    2403889111323613316833228046412004860844465230934769558939578130020501047443617
    2254510812447627086638124807603740798436641153165453629171802731069765306780701
    1542027369356888028077327705900794967817923298240980478100031954119120628256742
    0396712712348024693383212017884081599916881120943044951815183978022031592200376
    319520217290566359550716759
```

```
4   -39.5930529772661592489010666962004824215368641315239325075237812781638570282 52
     3564550433380327518152749329553714457789556783885854617105651967760114052796928
     2518609242949125824669332195901147957429002481614598878268701026892376280353807
     4365896529699803558227639550869247851266868257390325642691035305741843094722200
     9621322199803012865271492797517244434966062328931493459621745304983175854421344
     3441328022200139349608957469880130508978224418951892474449922150254148904268734
     7371283131064535698610978 37
5   -31.7956406709966940561236037679440657884865372437995487085229002255688869303 30
     9526876747000147480037512402365234830939224662405426991826753067937621040260075
     5680806886980186545653610837707895326934670538837396302772600245439155506657586
     4227554927342085134000624909137008299581295888187216821143430604948500949670492
     7001550637765169851241102433707026134845163846752349574252366076558424269626079
     8339269006069970099890216033160717198024011203708475767297422438907789777866879
     0319263639027523811636916 92
6   -68.4738707939149274644718677668127417415168454194081963288062329075565664187 39
     0833870757417369147620819368499519682312826127799129391645241806877197018142870
     1197673712196325767766776346251075356716888304201680878692625103105739457287705
     8004497503607716432735961827695434130389984225370749192139904488081564135393958
     4848602100356432809459098302691976885325894618917071279175877285496183483994726
     3738153768685478152698124805218812053904654696085284181119376635088833694130302
     7421298756882742639600562 81
7   -41.3315939045644221038809447858553222396042845868579392935059853297197175644 97
     6829611537793661828713906640823720024046619297671676183547689625004033914849711
     5365869815646215112421371755401321482154886319210708926215324948869956684817562
     1286457495079597580199199939515772120235678110482252171548184998160887464030916
     3496187010241978571846308650059276960091841019453954856382771639816644174415432
     8832937687242848001834577425100184846085221525870543867108176458487314026250616
     5120561355261247504789616 45
8   -83.6046200852879581926830476134815504664815829607443867608064919332242333846 88
     6175194768600395589532933305523448641858121942593590678581567076291527516668391
     7481531419596894513815026025230756769845290055020869716211456953100695146576535
     7137887355747761074622790785237496578519122012722900468947049022233827969282212
     0580940221531654350098205489392187999958039977684018687173683486783691152554046
     7713184811055627685876709477642157294580359625725297445565994847708446458067866
     5530397999921138175581441 88
9   -68.4787812949118105932299939829633994259844905497909591844423354867903124145 11
     9807655182957784119091611089995086490644904228367733406744364221950156181903698
     9936186519642328433619181455525683404301414989819165123442724975524102243411784
     2165661994672497171190995693062170001568142455298725939413384003305483531296825
     4095805214790123379320628631948155696647850326965738645153872141412125435992982
     9468396430693283387562193111291564090048069494963895754235100708732010640859116
     0121271984720160471457403 29
```

C.8 $d = 24$

```
#m Tr_d(j_m)
0   0.2979002733006043045256762665416731401075209428058140861786847557822354438984 9
     8901247455590261600988466411187644548831775117669765561710461848779449365625624
     3171369055329782472503039037360190005190826898598568357579920583454097394973613
```

6670992420142368794863001465890569150369273895062739215290711479346525698595937
2870181192221277183057123346887373915497559063135444481204752673545989851775321
6800657558618761567354049882810460971177372012613556982017075897006752856577989
126844834445874649916367540

1 -8.9888832754103319652568536151065598717401882096631572111044072353805334845219
3711819824520449056495514268792416761214950797754622815942492011969417216204251
1346358390407907872893098685967339154990601110726010692123539614300439856076808
8157117142101962937695877971150492347390725534251233428240939794345016776718245
7732693246908824882623644833415169775258425806382914956331903771414081023125435
0348511004631659913840338119141066582150640461539432330713605477778341715980810
203493352501331376118784342

2 -26.290905039100051684235604998175787763700116714500174045750246429660262826996
0826008767555360404418747327977128917789077030687425942928548059767714360950886
3071623457148258514845597201398711394111612203218402849443026068912110065126416
5931584670364262684940049205998992486607605619965553917264515535127653193119705
6102023852867942845515763006397666149947598437177185799142406434028522082185257
5253809568535666142827895623622355857485800698877511063448592617191162949218066
572951940570136572566961608

3 -31.606100647072854836425198690272023854138268842837781474843393322125401999120
9966690954841496472075896392227155013178540294019898700457228126995848819838552
7232737333703479920615290995564381581226814869413876348170044111809538620166091
3922475768289392601832690695524270249170592216791491323662597382840882994801783
6594745796308433248135130376847538739300685856013879634328778843019982741930263
1569336623335521760940258572339450226590231314123819087362023404388869719625654
656687685980759686985147607

4 -56.138419493382544536742720178130344613242291295732438998994038674170994599376
4342157989873536678601286690928716975159911568184942855574828381044323369506009
3667887044880326753302064265864505638287803724672479699567847968348001268360169
3021419969504590425529764189792133326157927802602102388663101392695111708810707
4509658887000936156765776167364874969894894381553264613058279029912252555150310
9008164218281158239855784797396942873946273740365612537214592130388852907715319
405987596162651050117109692

5 -41.833687276106344951272926168903044091244065738700887811792676089068741693485
7302563164163382381938479492517533566177507519980798741835178103170057277353277
8681196589561729655057807934959316538834884397774216775977069238445379825084955
9181830748145021309922371084718127224433689535064920947349537284634794653403254
0837012459597925953063309882983547934811987484351227139787433309628990897543173
9956262178533735499822264695440839028187438955008455368027216764529333647042065
900079673640268668580636971

6 -94.399657045384560432023455513552413627946295813219645646149657804259412057077
8478918114295488051810298533430211552756776656880808648868365469621207572939391
8796514948672286803285086192973121384949258607630793999090241931819370991270639
9933963017117212641937643747333365335524358584571425909103833180702780932054402
8354141832536146027785527678972473355893966892830255142307093328075173603223726
1067558606957630867681335108131430631251680334744357266215873363889638221490022
895430628730074290160439927

7 -56.073042668887347287551554092523260352477959328486755507939524264886457471251
2213182301046830931898215517786113933401520834547574276075458401268049876871357
5356039993455319772591978412034085716255738284719387318432838074682777820101045
0408779448287723130729723822358452933904965114554135051447537714991759060002033

```
    1714851425624955152058254261437510891811133135845911987535684865394644145449911
    1016779685316585556742561225104344550234576419831527884671197989895909512489710
    0105553728915272131522270597
8   -113.87591004944564587189637179673127062805445680715652895857877184898546446706
    1970920114748040243580639647154895246642922413502193267695747757040143112273479
    8353776273652503157563702138600615070338680979068351507885350804754804454483926
    3229084441554421134725907034736649262790324412164941563154734174506061272377907
    3683015137944786079446227260240586310051231689041976006701700895655932659476660
    0297868643430080966205628345338622387138285536856150578034292810184914295381321
    499461768484907642852867648
9   -96.405355373179337271239352327263695094363179592570619803177339887669193293184
    5337011828660018919188828646156066755772348304815712442133153746417761005760484
    3302042676907655883312622834653548823905363758168544783476757856502971055437737
    3649482078641388460350028650024060615741050186663565814151496383022428243131207
    7262274850303864834694483286341769302760771166849879585458838377927865122630520
    8384828766093467787228769893808787428511616816931732464213432646175806906486956
    116001701387614732238913297
```

C.9 $d = 28$

```
#m Tr_d(j_m)
0   0.33309693526320729413601354523719345708683553653616224613382704121564762831286
    3392848352814135820903217628463083954684915485380005608209996623176157716486280
    3654875409660813705856227411449867455937416399873450464631509464895715813488812
    0564367747845742901189168448116102458746009652341578888100580572583679500670325
    4757251346639738127569515171995176551587804274872998120184232825582913138159423
    9654155157848355474020579689064890811744688575830755915688647788798873828464325
    519364658397096388013592175
1   -10.595738076435675226943961633532325697832276006218806367957552981690818933384
    7963570153612310030124324114429653732322543438704700761147366877939183582901468
    3454460971171381012076365274688508967956915731863680040991362173659688914861616
    0694241311129765905597345168763403902054510440974313929852070108047895588573061
    0419792690331765613318535895925698984228784479229029745077872398172378535802488
    8180985359459486538476121847900802181327989135391745199929570977982582872208098
    088591378632810309384269250
2   -28.581007481495506386621144254978321217871377883178951819461172319442440118057
    2869791497341543122357155181400527130416548349206626719084224517305857106205782
    7393835224462974529133196963138320179019865149044345159726232237929919581962800
    6800772893172687967927599112257628913824708264789933113832293838394153006977669
    4422474146297372324129406767428720899774154700226315803040522336124219705721677
    8168336207518641520318067213070481010906916263970638665182567022001492205650909
    743046516544171778610995445
3   -36.073704549319505819773963923078059294074195074049664374400908708591021082997
    1379926703883708009216300541021553508897995222132430297515269576019066307674793
    1973617720000048776988653170071297967135146535869903826777044122647118581528093
    2238742529477094770749773500762529924407980455798642352499370698706147833295674
    0999323605130592750172629789547456150017776384934828308404392707289368345344480
    7381990253667675278017122494581640982116365096099245125011611975780766817902709
    585605611038870602024970655
```

```
4   -62.009215188552632524049494920819715061082894498943810372172428574337733717980
     1016345211225202365942672468472579419260408126731757268848549513728420646343648
     7606983955849739248313076854139160979400529999728047404464257713662600593180252
     4318012006671781972441686721654660428301976383958879402823333346296702795916502
     0814996975134794212911149261996761446528191450734901120201004953289989886503451
     3743211480971762045248236734907607278060300148116364389543931111397148053415427
     8191270312276378812969847 94
5   -47.749860565468752621101674199315615890388818164219570795903213549769684421584
     0000044397933054074760020956144978448001850580014186274582586453453588568509246
     6671149534511000338154279861063581945462015548476218810004130102669646081657068
     3851295307610314546738694673824418706479983805392040464716602362435485085013056
     2590784331187166761280040016252477057140997375773942061126080933983739997537091
     2903147984143620284734019441881333375291146829010988763523621743666428099042945
     9102936008716608383508323 63
6   -104.90187607418289033120169869991979365792922184025484955765325318631614729356
     3674553922404579280461768936442027002534112896752968804191402680212948492549837
     6678521937487456905207871919724552017180801438742060956934505469053757376381410
     9520144929321279089774180629110778159929411726820143559400971889320191419185804
     2030326398027213101398106151161195354007465106096592940440507225271168004762716
     2065148109228732874143443025910181219804768477464482480364125781300439384105976
     8703362314340566564686760 62
7   -62.646985481175025146694313736102168874232887181026935877294658498538083811760
     8412033103424202634251404307257013775393175335002326755768969096289552800262976
     9102655858418530604076136884618989101533020559167984770289033373846976961508275
     4596114441755546211375458047165251143310463521034840011837224380382523428813408
     6910041730243103305216599847337730936664289000449451721673081686526734254741123
     7442699204211711716696707778357923610680400261070315224625901389308687549231992
     7376349883390399576062882 84
8   -127.46187092164146485519497352763747854229701206154016937827184663004046810908
     2873296470775411542438428061210457979912168543859741992923814305647264975920405
     6119487181642980261685919563261925924143296913745582744440017263301770413133816
     1521317722979636193544586511538183043321049626406048871304673122661887887166496
     1897521340698495018362987254633554315673239459167378613706673255032791655533175
     3769592552274588230954031195030132350407470724082919356371511735801373335923649
     5399462950105777495054971 88
9   -108.05550481154571023349568689245531765283624233165864042138094781650364434849
     1633281516791281953956535286425873257905761527510183498300672413077927497855197
     5780534155604815661236115623210872283456792183073512413928655305904031277056890
     5299329377006662643056506489553255304766266969624811833210365725496196557627862
     1969527326290412943833604699226819132262627067856397754689817922649975241576685
     3822454991455099807663092890480130619001560927025431035029610816684393068859978
     0588189143828940855825709 69
```

C.10 $d = 29$

```
#m Tr_d(j_m)
0   0.19473125800165782681409604345957578754319786085518021414333350324933244336832
     2636794567229540694521988189878377607537205322255129301058458780720958732508074
     8939919170681334065743008944230900749294860171696763332840655721079337026452426
```

147505692024271809799621302543210913579447442645180132124442111376903808184052З
9425531432765528518474668498730306687937615356542951451894052627244513809291330
4724782621481448522286243271380556202773653611513415026359824787731540385510531
455105583316593295700066294
1 -5.0792628195461559508034151409697742277360585549326731917469946340740799550977
8174974965002998035595204341369115785946372124781915480514471799718358828779828
08557290445359603723374557484065679952806391488244222319611142447425244445585617
8153942304633616579560432026723080806819641920331809920203691596993710125251786
3878349521160288144561407664919312523741657329553369846784426760969227696888377
9648944294765637976783207119726024585752204045214975661441440920918057355355465
340333362416779516974787913
2 -16.527809824896493967603141928565268900553499181938570760073617213092702151359
7642612221795762964734712581987774734091668807061159357179189184247303648046199
5993967701042966043450422175032650348473279132048394050120934926831981506598348
4192673410565663852095490886806266832549954392003190677358172301312131555252345
4247857278573107606934745434840920921807400812472694807744397685170000752407646
0647792166019068091355048564551976734071735741170205785841200723490572920169047
363859334208509084586102853
3 -21.801781898801744961380839611035765094030790662330139949865155302419435091231
9581849048291407023351331391922368020945892201358937325124636794334521576054624
3486760676406809860816344734822703732735832050476205973858515583477239645414811
51307242224185033863979407552391159461263602975724459370173957391837493286520875
3587553200546621663355033365686081810931564915491227542983499638309902354649919
2140938104996622711622587201604091504941399123494842204167284915373846246376360
772517095199186185313854226
4 -36.215738330892854764627362816479101372402088014115898181159185462449791139938
6836615920820216730435579900503417248614100841504215402937058523399017899003640
2880117482398995832754785000330860386283670252186875899720713796633303090074449
9790490222356631920014881309040006800283254185815330599650875166333512544228194
6353300523960904327053886005857616574385096519062687210106866272317855746719506
7084977547118472593890569965395120271256502017304285475246286451647210420149524
860284300809055588860588649
5 -28.725045599004104034374339133741169914112126191218572806712683983662387065829
1171265006213344211360265792771653375305497542746425741079813289502737208080216
5469892291850782020961651420885006308570880056676548572742400593089688161553948
2506022428530054130920116973863562342886037491081642177044146028811826157598519
4012614386846323344060702208070538125301136299675336984017160573362670221811266
4620579763863415099582035392971342203224616909252100594648541205943006698274643
292794881002444659491617723
6 -59.930677482886254813287998002184754720025669363338582004942681387631042588477
1517679790701794837598517305815634258319646868203170306397244586509437487720225
7331458918674009105995668470347406129149715651223460783879412419184964397027303
3987735531779574750522373008292905980532715280419281036133245040744625019269882
7403755399992910562693413622234897677339184923830718325164914770023717323602890
9668851993433739193012963585479227017487595659815977732427598465192883095675433
957040148062351266435080121
7 -36.835681334048251320234874627529969015685819790219726283994310871390995636373
5762536258960817498439589414475953119303293767559349401711352926210863527788171
0399172364850905988178870611900260936929616302629605827404194081176697281608885
0278845957809238611788550181046109052872437281686741457352368587422602624718487

```
    5099132988906101458386748681078054416143947079473262381548236919302149608826127
    8687031211588630194783678883072674163762848346180581596973434651754462703110636
    631528735530030475935024703
8   -75.248106619304870306128712290841630632192618287876013090552296702644054187061
    5436338298256647370428523679763392631384238769109099515968426676081297478790540
    6724680904581200244495377156296042724482145225999650485249648446647289747300223
    2336500851796229707259020204511829983941308699997640376036717890390187774776225
    8654688359250000545655418641795063937846822342571036409196498703318494025313061
    2183214370609498726224960225431374542518890555520127774609813590309358051791819
    319888864294606670917089996
9   -61.827322224317340080516653633020093732644309588935985076320322056222578219508
    1678616601370824799907475130584726150882716191926481576430124362756183835556815
    4731468188374862424975755386571702773347381148536689734816384956016957866247104
    8970626584220217446464605420381706504649984187645034404824310815216893960483984
    3692929557902704930006223727611021654627618154146081903909786406535132333232581
    6593820265910726345659479935739894648459371365123535525920214882749242594195165
    866984148987902985420475172
```

C.11 $d = 32$

```
#m Tr_d(j_m)
0   0.29756813288385357620524710145937842803382510233905830938252491915970240454849
    0139401038565159389529410114338440823268438543447135348953173318149490150477337
    6789075394053725659477008322595401040420266966718460382884140021295753557045180
    6559511692151609284733864500339483588848173644811158048008239863790016988680499
    6435506794462903902576453798777157242795037061235631468839539169266259734036515
    5251452861111881274734449779454830916941555973283844451946588218843976683188411
    553805986626416509594720617
1   -8.9621705873734919545466729175271530568768760773567565406367349923993088500801
    1696045091534753225781591561420919923163176087223220205812970239045283563165072
    4205466884508337725065817412403183449959372975906063118800007018960744246778895
    8065510049900808281225879500946888816825271939515205802968107332908143461036910
    2241546564460738930250534564705042432773797835510948986823595701860825831704871
    1612645356152166737397965654505391170286917440060838727607643985680035819998017
    886029211413304531310339409
2   -25.932028424478620969583284449365987643579954741108133238123864474272091242954
    6996773009399965110172262591720416399821169265515981734101624251241676282097985
    3316589130841007572676880261569027859499665903293864079604727737466711784367942
    6087752064959327520604435324028373071458116697234865871348475239362433707849201
    8487989703858606890414908192601797443212271946545746502965662159156845695467188
    7701955926793973713209523987881565495687176808072613984444370606163221024282681
    258253439876718559745267693
3   -31.940114658117604789981881210811504000133415167842070948070153293758728880855
    4671974283372377842025876628856580960500030112303975402743737506782274013692079
    1324382663248914864744368841146949836544343315253153186882436227458601166827721
    5420565160590919730615951381348155219283739302314618585550677023471376595822731
    9586567095829595928017342295853215709392079495845021015015802377825162810753053
    4063404477242751234696559351193907304622245396933794875722018788568619399287097
    146604004193859089442180391
```

```
4   -55.526944897014870891733002920516033656594813756516702301157008041316622387747
     4741513979031250847426339318995001383283735418387225848160711141593087986112697
```

C.12 $d = 33$

02536103373590069094875108880325580513936711895837877209481545790841736400011 63
8770730172312743020956676626037751010671053793793275078686742583025760611806815
3499288708578031096486041226415679061036627742073496130890913704298154879033227
7888468035513691203818592 91
1 -13.054690774569642082905586099142706530219788175899639859623013304989440473739
6559154006824529785184605105064086096747326397075680693261361815383953523825636
2093670393201176165836369493631818490736347587486209178518926728555400524848541
1947793311165546751909713618714507360549021663590539009644779962728129345381453
8504826800386529664956615635059989646012828737393027261405505410544575809728525
6421311918637747664009960382774465912326973700787660835394927676551603016222978
937060920315957624727449111
2 -36.845218414350576823097086070005324117448145487889312804217476271918457253978
5682594883551505294999685033013660452041476781976204805854652165814266411648659
5894874662297540994351083325101483056367373404694165168258971443722688151804780
6604592133978432951326054173462346725081405104071578854110677546164892705176053
1556180969630337055696785943215674904307697285607038302663994002098687221403460
3581734150779554783361806403327927762227413685989857654452906043361761177755407
477891275170303044880488332
3 -44.835007121189846985295018293840587436154895737599192719451038979606294303143
0079225892498351132887810939067276844947128005134684635602148825723321943741233
6595211283649891136217313547951380480446312929278776111382239924281934715265135
2142266711916914268840624326640578445976466231337499393434451353864114007502986
5895881421679928926664833226426109654857326851709128239938648921741791396849707
5851332013123966557712416744864353813012945880446867734271789324570695848622949
150706387625837400879629252
4 -80.035206093848721687608633157870860122043923336865265362517725725379844346608
7031098878082831179897446069447032508695838483108813867772747526583470995754300
9818066462090183558642459805632500455853469606020468491515047169032977245954265
7529020308683123725662028892640603501382642991742709806003076009063894922879930
9171149699988404644188460065372631454892139482153829443574393965950053349915915
1341116665722880443305417088772870555647116343103333820741145984962825481960768
432631411526115253441417729
5 -61.006082597251391987140787801093160332949337585370885158518983150275364313585
2854639421013062978607399799870148176259574777842290905136453054050062745189072
4749444223499872105213058408202371825487088645925223886579510380765774164520080
4652806309137641767812163132950644000897865972791816471705893754759967908595100
3603957748041904335031074042107816273140222460502130370093915190732397080394393
8202123794818696112202810780992863327945789985036326624719854571119993423841959
267296251786055961553921881
6 -133.75005092382696833524117039751522634670873471081483942224143363675720088150
0480539183616372853706844271596076869397276085033471622283225596061553480887770
5000448558612579651921616655742108105187113678178322789976658119200215691581884
7423376000329095042115994349299760390260676287300233542228260655053837721951768
5599406887096693582576262190389766408886111053044218684933140430348621048760580
7257558541087426396968034542793222367348897089343676024765013780799538519133427
937447669581840126581335471
7 -78.671729933255064725727633267941122366911746356109644123074318249805092456991
6629124313757194506360337477942041407664434023080086194376297389351762714112281
4382679258334781353787347608854036411032982058651090810827026266372415297204026
0739046570815744973780825203004914448748871402652840642613911805933109984125204

```
    9180557937488899894225751419004176640434299570031717623488739687344477313309376
    3851443295955752524202510485649451175715830945605539305669946052815070684660519
    219044041268727611140462922
8   -162.66451083111464898213485731013636209125571048582197671427789742853115280044
    8313693817843027060000702646066584200396694878514975786203608021545652271524477
    1896029557088861248468369643725610064952087551214559906119554946793210796303710
    7501985387186427321507518388433988400158409842854366230765636324107126293796336
    7922647299912066030028178428531345040534915391736278603293779056585892271360264
    7107008491685256766767803150124337235337728247778460415664068259948744286694927
    040713626885953742559025880
9   -137.57207218080551763530572959039642968728774050886630377113439584131458601508
    2453756665265148438699134164010187557004997917809642306249067951067093401065164
    6100724304830888453744420333754188378441964323551550285659033541669703271857845
    2160638918272062879051324228477706086205473435844255864903600690297784657441665
    4766524239579097678151948736391739366140249107303699024382702552355077761350928
    9888541085035335720068674624278435678187140528491454629819573541168290216360741
    197422596653901204312370606
```

C.13 $d = 37$

```
#m Tr_d(j_m)
0   0.26078879697826324423837097854472481617331609931714306643444707964520861866782
    1596672652625148027019580304071750708358642493676452270060653619485614030749810
    3829245736778762225589126411723949045278200236046380428737498931346890103061226
    4848717669850238835311683995805454804910480168974576778252627535686954850927668
    2644208841991044920431942321073861881139260501407507790702708267132132691284446
    9991502775642828510675669416385926213853572071660297041483155489006956246920881
    350768454100602150582436866
1   -7.5969281679915593093342242018797295085525733163366790133828906240710169506577
    0587463065516407442301842742707352485547721244761901218025358181766559620061139
    5261510751085828678736851864203690624162609079858193615177472095503505369235723
    3900010061729981643776969114941655146996400822490547154522854121526274818530766
    9258506144836897006373466330480812535870823623500203016498656879724577396296399
    8473868278149512771505624756886421900260181069827025809495215224521487221827281
    454457114286788518046641770
2   -21.938075356094395304899608336641906330554702192271385501724735971213367442080
    6549452899346243527601749380554587813221238351464957379842267748013673830247388
    4474552552508856709841069061736260754261527026740949751597920135413516622292167
    6113946748432589639670437007490930969147127245626711740834974437176272373777515
    5281196939338405756279104842490150941543965751151272928596318184511446024941976
    8544424370203442910557654277987187721524951753020013023397355022056923061462633
    350492233325254040891165485
3   -28.564102851767333678596661052886780510657806425895026800477327482004751369733
    0042780262455184253868992692443421142949238369083745807333686851591416074560557
    7292641051199150626777796139967587474598732242796631474184182713947200148066397
    4671794303266790654371495763033659257078032070533404413829873648103758746728061
    5383676878061182220450051643547320414827399584608515219242373833996363187594651
    2085556416719973759036855263470496954924202821976581048114825338061451165134248
    667186693661229465689848596
```

```
4   -48.7973310248913636955611280453717396115358487239357190828995959759035736220
34
    5440669429121380992643755531766715247375320967916533064438017709525490420837079
    9890708333895516657775981230790887617163771769218531776347455991217003149132123
    1388272218373185865092432221564123530971252146323351673666524782903496159274901
    8150053675404128672053533109945397948417894047170845592453941911981337840005108
    5146681174360893154145106593200151246738812258468975955164553044779741310288459
    912125885677820486846884756
5   -38.4621582037882795077824536747452581049819968297979652327173017678923029934
49
    2641233972935630068618318134666115978285511215878373621893235356408795741978805
    4401076712189411228238325460825775605763919240792486392141686332892540737317171
    9257128875729368607710186492069785440192791814405488703818598787926151559296676
    0507720065780070871653814356460603518504602704753216539630741161877226569093937
    7059749590155212832190863730470661000737116408101371032462174282646411272764524
    854893367137654833878288686
6   -80.5651914172992132802964404797385389962035511012249783858591318160048114212
82
    9138814884521282347075769142406921174525448683604907439694046143787025169629231
    1450848004049400054678075243889487653438898474850735615965045981780731794658680
    8101393810275298325315996260791594722253075476082569920979652258050570537106653
    2067779177868578177771157876736932264695206981132011615551819795142577538987108
    2213570702085456735312045035097254530269827821738966136744249649771313809607343
    460150479302285492573404764
7   -49.9687111642408999045795145811966474280924670614206666716234699994603129777
66
    7698910196900584785901249014726038616703054163236042580582933043189744260288829
    5342870348629184780785573906786582273091728026062953062806938970504237678954547
    3176158416460070759045613664907073327861409103315153654044611596534676295077559
    8233114190654163334870566314702839927460108794179189381853788297165184857931873
    0356029279030332157248113897115929322517407095679196743692502357426562540883776
    876909191564731280572998067
8   -98.6016638052909631123338005894068928637445560943645906563551587615860123894
38
    3603247447991471445887610372414535959882443464742400427940563452976405015291452
    3502281990404392447114161740211568408297000786072429471783265800292253730776080
    6481229096709511319534412826446518161289960371581260733876583941952645348127690
    3859534376844770334958463607482450332180620780505511448045693574828420910805588
    3675943212893323652130953281403743208018348217143991490764280948041886249597484
    368477853969049506235986978
9   -84.7946867601763596785346771820702554173419025091064249712662999663441727179
86
    1180429572750562276494304202243071568651974919211624541566592674584874130746165
    2434940416716750334675585601784744364618593703945062034536074112809755813094885
    0120228347644460611898093466562792633160967432316077785615059123506474517953309
    5842758285728302197182104019336619971309099834401496194032586610180070784481653
    7144327284077795690253629618103773805654785699669162654152323551352501467387218
    991735339817618154987169244
```

C.14 $d = 40$

```
#m Tr_d(j_m)
0   0.36608391018937650901825010761483002875336579433958510907214038679641840161
511
    5075030823514848834570866489369367831393323248124170390380730827960476610514921
    1473498307773677839913322249037245226299182672611919962922161959071172916793451
```

```
   5343342540768040389472161151029968154745629566169834383058976548529556971914022
   0086623304580303952809029838285264155183839237907420759557475454480806671990558
   1216846587028118330360121446886701243941240502657527153944855729408081866094801
   107856518123506736893471034
1  -10.974707335639615562615279284337190917719711041547843395166734281453123806749
   6491655296283796623816173662793356091009002809906489502227652052131642665400662
   7372803324103114481369794133032926181119802066231215054616092875583537847423475
   7047103957884398002512700554755289405612817749532459404279931342576613450565380
   2619044440408612251492528078342654271537269798685733443761204619490325962753968
   0540627178197354610600803036968329179156291280540650211057193909198394076453207
   3560330585917190691411362 75
2  -30.936588943018778424351055295967434667851135841351000880609346557895237903952
   6345628617762280596925682740275085566808870935989966729784439903050458194843673
   5033776955472180638448502455803790818270217814503558445953683753845624069512477
   0747879804353172447192716965175911096348862138292438883548468349634781402884591
   9223420404694005268392256001027967414607663111984192307519175451228302168045510
   9445812886264744121454357391968044070706335338991781202795889226334171119084524
   1326207618029786973632730 59
3  -39.196231339438473963016252587396055574566769854742568134266899190410932190627
   0030002752593155450888916494995741451439968241853875997765024232789006990366199
   0752210953361892529295375620901004949645153467145796145367986882704837705478792
   8990852240962609867820635665956957114077389697009297860526114745719418958554002
   7574482356364428580931497293932387220070583490019657961705578445971984720423711
   8440812614297682833444773808234971592411481408885736034685650245758429910954746
   5289442134857613729077 84996
4  -68.802376113992772564363907079777000178643526396531010211098115032740070805088
   6656214526994851071216358405717283410545626927022413777526978672315953573946100
   2495713638848242061831767866775423480663455051687566815548922678314911906773186
   9549352820522785693483411259753447033053657222356941679203034966358846234205259
   6670705798183923938632467087339886764966419264072519586769459370213234067244729
   3773200007924471348534914627647975498582117006829985049798928726414373399988407
   8595632221038496253658356 48
5  -53.955210887465195666160922866898145348731749126749764656106443835136385511017
   6159332308637350016847259129236632159782101482791143068325922624489660290775219
   8710314657514039284613977551083709488479117976490048112495492923225479572687613
   7907192330383248729570664399375056230642185821495895203755650979691811454239350
   8034464530655787321314613679199750245515811159787942178177225102022147428993050
   4809628008030270987769820714282088982544780611225520246750036001264672321807568
   2581196276068988265691422 57
6  -114.629050181332632089188932535278319465327587162202941415396886141353287689 19
   2850731285082261571880065716364609802085924643766135946007059499588507600561592
   8501937313686402238282651784922402392035724638106356498119897809579631402193720
   6806392138868232223148595823127038512807678905656545883694894095613599869030105 9
   2942614041834425054830109140399021595111460570660933936675175416153473135649909
   8016613039464715768762540916510191128339310417755386705896926832686979280210143
   7444276579125131566868216 43
7  -68.136031709453074337470257993515736221567452998545773156971235708496362759824
   1895329715434430493253029147810484281308369678429910334559333103450034885782034
   1277671885099965521337833108888939408558602264205317705839871770863560696291809
   9952254568644285922327766274420889599991919085733568709203502751236499512996243
```

```
   18333778066442781312624841625780191388692474026786228313286170391271553482411534
   54249292852926317879785822459354585719189611443293109658717432241846879921122210
   001714941479386291177653135
8  -139.869404300653862989341825925350328799423574109192637058728492067364709376040
   5124410720446387427409215474469444095429107145434101540971268141493447192610711
   4688058768565663941235650205059529827313196605542367764640709604408150978732638
   2270975150709533893904092123100717177521779105900641083951337672074201505953350
   0375308793894398152621935537365153624472796206096001654780973051242780584407611
   1552299793521686343596273942878132005553965933753254055785614942476453223818763
   297028866439088042397575617
9  -118.17885578804739997614093048953572109612663256198441670438320854635567392484
   8621915633745592244391540017319159671655679946710242972574081861442251055158895
   0358901566551966275817455343648559134784671091856841601706786984406351445155169
   0288564215953738220680193085343945150564005832509647326411113594056751693419951
   0424333976187414018301070168791051677363822275340431028751571295234461747106855
   9863721824928064906693613215274696225305921969038429984814568673564772204222596
   0648691758825380467448502201
```

C.15 $d = 41$

```
#m Tr_d(j_m)
0  0.4135141645721219043660102284384564102286456641985913557141971430664468558413939
   3836904626605954827244184826385377591875470107345133609921837530117275196215262
   8004228966597836648325086303723361125004222714110934614856642527467166854516927
   5367576246187738679406078908613868761140693129553873974604324342068051551714007
   3067719976926956088195231081432667501676642015512340631931181844825555318554742
   1547615325642956472345327387820221410702179466863756565752994711532858436567308
   47258403355348306508655092
1  -12.638275379124474145951429659952906217019962598411454967471417961900176114836
   0234492754967532072482180202388987304176137285726987773030948124790768790237078
   4087463255975487400702016249516560617792772294638745179794786642927321166506708
   8771017149203194817851357004724606591099500173253949971719257367027620511215432
   7628730378888294180962702285676142088839545960418199507139482946977512436212189
   2460807915867725897674893599795691711707245995148311961178191941435280535108884
   552149101323714457651681078
2  -35.699476261046594666304590777674243836524331026719500954004156853830168443131
   0665728450669162595278703463932932740728361705895828574936454610532629079867853
   0214826183342639093574972252499753418717928496951366679838690835643416589417877
   1727805876305760762357756374365644362923543403681484249805258863894795128384856
   7880570553425225656017585515557703082811862160478914414041571456040961599576824
   7895617752158095577019488530029236644812228733196173791174210713323736308823425
   56421828414938053502730065
3  -43.987293475197627562453247798582287263577033529856154166456487804905147418530
   6222169372287797948298691321192504205486982609184279000189875219786782396889967
   4862338441169074605012935914550615006483538156133840623527124536314931584279385
   9049662291633754768027997518564118858948652148143057071888536946676946325682544
   1149156497461181971117233593384713431779252979205782001478804503295105252760121
   9753195420768374070774771442321420783883151632839539820027665548028270522323417
   687674729597231166556856042
```

```
4   -77.3299796785001707637198743877546870522706834999269569353751032355159596331 82
     98373461032834567472176940825922422545301070055719054343800802381688686934290 04
```

```
4   -77.329979678500170763719874387754687052270683499926956935375103235515959633182
     9837346103283456747217694082592242254530107005571905434380080238168868693429004
     0801903179739773882888552002406975685553622826413652212964959374642691816183034
     8580090399988916849120156129028752958426105429243890220982896897999409877871379
     8003974904788778748867249654277621489971946902845394728124510332770196943451686
     8494771512008905699520836259719877438710221494953988199636404872196839522267089
     205259779009698882658863861
5   -60.851765928226299274358751591588284731190710956881948575011977449993870314129
     8717968670526174591999494674034192073484357235154456104442631071469635831459258
     5658122276211300889588608654747100855477764428528144422318698400041150813442023
     7131300684951846557373876872562355356164384164308263988614204903114272561030701
     6423011274469702800547389910835517979668661632283583740275125519360783886478453
     1493590679574609351341664410557152841909653144695081657836382062524859107378420
     543505669930420246974759963
6   -129.13414099707953631934463599859300809469390992585425331827991323807303634163
     7319867341124542808134874695358279463417542174703500418868233994482547472412159
     8366455407980198521300719741819302992520410597039618758582327144563365312008175
     3032347287402567695871875735926876734851312288764650981007935833888285681686269
     6521404723665472176434672616943886723879220236830698456431386128420020462104545
     1279289761843858341186268223453611965385511960393437535702164815731826644598533
     905353184687860312922643812
7   -77.456774810752149744456343954715547290327332524872913308589877399450697113292
     6955161419803152872198333935091017334473781126748924211993500541826064936077380
     2467694036247359351371554357864675194383226922206925296901253007657231317582352
     3676799002092086758655225094485992437651458503252232210144105078394190163800201
     3887863646872140611779278643295378682879870792054173978612414873728793436565670
     6063883182092222428957143481530613088518633331240278150032360944967732677010002
     921771586810054137147918108
8   -158.27010577776305394438923474666476961166422730242698240012869378322844566911
     9383528266181632381692007668711265959255899874314656123935920036612206247460503
     3772009804776615063367831744607449047037133128679978565112936189710374759508119
     1122235222558830671973689696203898010223100156120652087206193648788956863150802
     1808484492883887786191642783288581740770800193004595262552959981299731854915258
     4594209767945945988270753556105520275028581776840287415558716313067738195012509
     485374410027140308659805188
9   -133.32915384915472746512098567458355258868559202267952128199561433266438882860
     1874826698822123324310758195596396974178157783334170993448781590528170883037650
     3839387609213444881435599425871086821426880709584741146531513031221413275474039
     7672589379521697835510830997367799193054053545169900438865205856569731341353518
     9715623093277528646828325166732858761059052019529629061850471914477691662118512
     3214743652380526351007520585156484273324266474623722421464890915584099419142662
     164913514776549417147271472
```

C.16 $d = 44$

```
#m Tr_d(j_m)
0   0.2872716944780220271122179475253971673484240272746626799627175787809619265387 4
```

88695515484159105545369162898554596230724829914761713927679776141060489434611 73
6746234435719457894344266917446478371962637302949006008239626286216843444430088
4650109075098176073853840844874509276818179864725873084469374509337055555027002
0098219076957334958259508 63
1 -7.7394823553047604292771554414152681858944714896001257941860426510811296702106
9844428053322529186863388917769821821096515895060745222499355270237060149575427
5237569719551020974216756151681415843801075537558123569639257066592656179935189
8136551307698283656490100353500214009483212580767603252274533484517657479059882
7975333122223304992372941856424248844437966978251293804096243404375719693994066
2054088508746709121007019842771514929133460547452551598370140746008325611540151
3507798858737034121688 18792
2 -24.355145895933215805439163467187314024854185819690126140292417279986790066949
2082165600558153418621576426537933786119394475893468952398218109785729233310291
5367381401671272963836148129749246849434521810673323762874063514061708373651584
5488873213038663334820841899826705838362351323127463357970483153981624244130509
0149582056634989690615251754247153864102866845888272156088483437978339612437374
6331362459570224285960262724404352438656031879847851568118174524823319751009097
8338769923800241175140 78139
3 -31.184022193144325054597398208037130423275248166047529191157299504262096513322
0328268395615481422192207524294380237429129008412080444159835612766878621064719
8252068251103312937350793576120387065424355585053822715912084831640662776829239
0939484938485642336049433210263818541103208394065156329363096839522018570976909
4833170769193279839414729190129859396304131480413371922337388321491623249332088
3221119698668230683465272075195328333344508003941769374914022515976085863252150
5751490771333416984837 73447
4 -53.527585155443878822020198269811241365211904562271111986823742406455535970708
4210617710574686263700842946473173096592694961353406509512810476249386819043831
7799964603143165583553613368141715877523874546868743212174315453350848700591539
1908222683924300298146183641180555919730289223215042291079024962149276661528767
7871299230302634421842015579304519368829791122367045936894143679386850671970682
2707786620452080838473958217667466263918588051078274234659568846351905107584894
6086491147165687997403 00126
5 -43.027829979549735215474550259885613808224103731800733889944021788403506138499
8953074917661009604521498102639642940354832312434988263483847840944041461512593
9556792299505058549541858530158688201492324493116273820523356469041093078018356
6204186978968602732555914776380942681537946982876834066314579851103154205269153
6662826451677333174614464129361041818909846631681985009980379998949436778066893
1678366526070587158749966843606869407094058917380478636389109152492172788710292
4319700733994531995043 98771
6 -89.603713252302601313839831697959837740732230043558243553096474895177349146169
8964951037827310477678764139916776183945296453825060783960009952833304934311749
3278825342747655156553857625466855890648089465122196903992976196306334550713849
9476732738912832408498772853657388022034556243285432991824445234361729186131411
5419932059060183826175966078415780787157129853994018715994928479364104875482693
9693402348304216703455438294399994247138253113140086753057444944788800955118471
7852089499813349471233 43051
7 -53.722943801974008202687161895187859500682059870271190849824631826327716799800
6182641047801310865015973273972330819995753349887832520717832942371409240011270
8359112434122522521238482958379557588681853668667523751471365395965663075043726
1625179730919276871372489439543846753998487214391794737679459858655000265360926

```
    4504987335948403918910974447139761657188497668164115998925870262534179067429991
    2488174948155319661108345593582293503711427732271317010490607331661121818171662
    090170310327802980237749913
8   -109.36757319480754930906241348602361212524132234468350981435965807419101193720
    9965100405865568952891075744684017524583743961359442889085408072223797502495712
    2982934232867562486620257096679461409167305337238086413513108870264783819838730
    3027337911489653425535239283301273917766427570773971776003521041611589703256867
    0094065088641845499039394903474024723398642072961424622534731664976655766639196
    7676019178891359162263081361120084868135494429828302283480628784942343070624903
    935063548357541812957552995
9   -92.138146228599034608123842852076405643654310586386827111424003735359159863883
    5289344181752709155364890859966068871922571180645603948759220907801216796509462
    9854591199479589035683836080815032535323362493894184928242975164780108481671047
    7584261607795979816375721547137808262016062576933177332499634520470281619815696
    2372864263617329738007059015954180006230481278785075290109098540785543059976631
    8266172048573961368922164988984357292894169359312655299211542569167964478945475
    347040765178474604830836954
```

C.17 $d = 45$

```
#m Tr_d(j_m)
0   0.22833903501789667336344783301746202742235452010387150048852796823836525846909
    0531684239916272733767347884298151742587671337361829999680432560805303383396230
    3126668045972755869614217985607684013780230838401441284260959654034047871759833
    4975463682134839030365381392422795907609362645413302530507784365163473470878432
    1694550869121364203317285291486073307516220866067247269071700983947482264859476
    9677915542643801869747324247857550927093802936325294554589924436810891523846532
    158731029596685799161435272
1   -6.4912927771164145862625258959149158709775282406643260772803545630577850076427
    1776097005510023816877927241105335224084587322133716702068907044183801185395525
    4891359069316732902670028083000927609158845328109622307539567413228176961995300
    8766124103816917994804436879087792440045852421317812883120881963101941013384104
    2101993210297693581326663071784432519798534834069909747160487508694910551413102
    2246079234317698631823669966412319978311892941522881718639172223319062549466962
    030898513531120684418679075
2   -18.879038477807481003670959672825517030409272471640603765218391558675251613543
    0956706136247044112188597362171164245777643373239965759310531455888399117804694
    0340260110594337114978071055050233681195015542353602277926217874418649867188231
    7907134551899437116281725110899379873432318335956616743516192503307925219143804
    5122383037725538996451747097533570110263002707607776403598172346100510112576427
    1907169391466804801689031335792408111395927376547056427767624590329319561414652
    014850238179091954713280854
3   -25.324429527600340773974785529858226519540347765540482695356325815437824503695
    4707078702505686080987436145550550940197669515552723935129190341265923728948179
    2144001197051853423906112823026565036506519133252207524102661601612453641777203
    5078920699510805093172145773662649201185914979841190205430327371025437706318396
    5908471925361702081788073729408798992731012064318706314259280012364515694699329
    8648164950679721535222437700078501243968401437849517177512922013973359424029852
    767132158045902514837151686
```

```
4   -42.395199445883110673835137892737577931379383204230681585249803753541879731410
     4902340907415372306376395251160151741620638937127218381391268919090909060805188
     7077932731700379068980533123343622137967730895566471979976018851742589695759017
     4757714594872467177979527526943846579913697206793321440335996528120594052590223
     6223246079770174823356747949542881352326701941124291829318107530313868641891047
     8972080076831969671996585571507987106711414480678036640640605605660946561814358
     29696898725228622289626069
5   -34.498783144047351014657066983731318875190274482657197241629539733653749055775
     4318797170933100006483852144922814990764132898034834750306721578665104351802074 8
     3252799627883368320064346894086861347081963187178802285058682385525930540257007
     6150676313747491820649259266027708070174078175423734526198939622004591346064973
     0361423830964219728507869904643839666305448554507056097104402819516275353466934
     9219121861691452683114358302632987478532505942566211316358747490288711447310702
     32348677878424605520169268 8
6   -70.047526369057555050411656956943155819508969762715400181104673370940323212793
     3532817766645793965070319300662569488055733061487100340489442520648028747166100
     4694549400391401450053966516326277383692266830028955265924052362614279410788089
     9342222432451118384954102556949278394630286633095358862780749612304545930318816
     0891907780392650062511803726300552420381871456423295407664170892516697028591196
     3603751576167355954095322484816500211437519860516369094596231641567640487109653
     98586740979261210960314971 3
7   -43.500257328850051991554503294564630840457512073491716876792519596881897395556
     2227631902066344577515471408281135728130047022950313402354216268980587908882981
     6113387691999496774356454561966024721256548561604469819251960788416916565618543
     9658988073468038667261080831272195452734724326091962069746844325003184167415472
     1272077701548490980726489951520958487883094005761984006992996220526381966390775
     4694462370942455549012084517895136185080560517071568946259972705852061904143505
     50813225359870373408265039 2
8   -87.102191662266480915802352963125358841886751104842099996233673523250620026527
     4194910267532081508130991632812484181986953495873953677371052038041309294311977
     1833705000893091814792586162031231891604325078077875535424090927815314903616911
     4329985359501130625797697516362418926653033081669329916503377364481692456414878
     3763294641454443402953166438353484548713384769022052478403611180408997325998248
     9364222743642949846982455153347879140237355718119694378394352562439652310688628
     84670477292336859539225113 3
9   -73.043548835849113588830272133233060431567173393916358366690585428372099849649
     8011120498588545221431848111513892475723954785252220197025161274893732946330535 0
     6228300786899865483318694231684258823467991972362174789550243088244814033923623
     6392763949169163650545098237122275061488609490747274593378783031417474871586319
     6648180247144304951188192548338792776056230412514290441638104987127799710825656
     8874526852092017705682105338334327142745720766754446331322235668817349231768052
     98304977641537549945857481 1
```

C.18 $d = 48$

```
#m Tr_d(j_m)
0   0.363038471314282732253914240162830906374413538090055915817382609860648767439 70
     0474918154821633964267174297908385736882825744583081575222344444991815698779104
     8753757797561001617704940652506024893170202527122439591047628603501147259839021
```

```
    4266350234672367617606766307368864592781312468109773669582431544602880401919130
    2568926245413417166485613872953051094569521480450047791289203813499837199401803
    6094381225797167856786511378280033446609312994013199567730762192530313950830696
    905665062373487073705415175
1   -10.748004699280588596716468798020658422081717462999839589695349045107309374032
    7062939502902885049423050305188371651946540856139497741089076510748271743068231
    4731018526892332477281652664521383251528737134785456601343433245518527766277246
    3422817622465511307628632155004753679143487804802073938764266052867297438924499
    7814083478888856137688894462214949107206740482536306545914448482981208799751229
    4634770652452104142406298556923829428677952728699550711302844521664856536316015
    419086912106968500139987806
2   -30.817544075395451084335745510284393417058672604962962414414257278919117068641
    9544039958171780042332234899050920419644653218057278404761310952900103797991826
    8669907809866512071163335784145005001391149590101021699438467683346258831101592
    8851445860014536694543801677978324259971735412076994632431803102962891461434088
    6234216017420454739335335543090072641973417973933224633888923749031138811527033
    0251391635795761262381783502848655621177988015142866345027276602205685814787919
    022491013704207939627007832
3   -38.506268373073259486591984134093807638184197137690901034947539980400063675196
    7597760604925666560761971097149157985435904005836368395960573260930207591716879
    1636675510446790157546914124204040118282600077959738378516299878245554732076476
    0650303640119988305519722723820559914092032065302422238153033113156360850200639
    2386582104631272682138360308706641231461219583645255262852416089936409256512097
    9787321112449835707242252605803805386145484253503563564542259379645692135207975
    664814234390902531445567976
4   -68.352621326782311708960772299380086749929514953681499604180202322047806735614
    5868918569249240710135407458058934044591325612738140243658845890733005322559489
    0218753455773662090160652195940006004984274395383729144287939133058834849341752
    4806840379172233259680499209901041699133461580688717525331135872594356351961610
    5716091300107856841188527481911191127928119152314028698434237256670909179423397
    5809947898928515161421717109344883973102615470744230227217937968161704231809185
    316071716699285650317175842
5   -53.756824180362893857397221057107592000519392222541889598915925606349214312949
    2329104159310475483775010319991333872101994030455464036485101168694445456705270
    2165481288619644958140869796589742600116534066345139433478881787423935110932138
    1761614979488489381978774782663903344839120779955468212143290777236415123586656
    2487268811853425846480443318835699848410952814517381126857692693195290744583694
    1416770869336661853967239657167357497143722808190198287058523254267849274407275
    844277387454190363641475975
6   -113.44164080081785190887646344909083951415677933863542936552004710148722455784
    5136182471708720751884247053084648565697527203085953975591712904442761375703586
    6905917871169512796317166265680806822691719109311781551834889510712960622282575
    4940780982636150947383911826974267313228177185584399972632163533203856567051138
    2638027662315584072662333241516595574433861592350615776244428852868542077831159
    8826996232891921186743899798644442816016844535344445075736728746803918034570847
    410119919126267996438787928
7   -68.258750467585211692752216625016100469398361867146884349898885077281743078590
    2344309380658548945121101416795694377033674640747450557756291031466531254155829
    0429631733594472820030221061175024176995444928306805213156740692937180932650035
    5852245206406120919855060897621269435359860616598587984076357435245237199235742
```

```
    9563851010340055067211620839560525978634679118323504671248445391640046398913147
    0946337943189724800559913070073226437592547137515138012543206552932169026200977
    693303649542808645454389162
8   -138.01412988631971389255807264962935435362609846491174777000286524248333652028
    7298940455861483851547987429558419325863121936403604690511383858542873344477485
    0578664097421903728619635035689497727446087197136076845086081953526436019482674
    9612039818627947826250741113253483154818785370025945872832223538325329667305438
    2320956136058223338237933377351210268293595445067802685689295079551562742788730
    7385882912480136341124218596413263388695692078904225282204297932290751603254735
    359001484096772427110605978
9   -116.78247009543439821387746837703849887750085213213111540053043780145650944712
    6121956595320853337701314821663714627851355906565634058813527866247050653169945
    2209130855248844304321840297486566956244823161061407511653007075536258589629329
    5614817260862170718727760177361915647600300117192937408552359371877607241300108
    9708965954396621435594731821499175419427685414561036980943537156000875802268398
    1051367187680555901284179043949743863729052055870292767719368335323822827863163
    672371550355811257176973449
```

Bibliography

[1] N. M. Nikolov and I. T. Todorov,
Lectures on Elliptic Functions and Modular Forms in Conformal Field Theory, 2005,
arXiv: math-ph/0412039 [math-ph] (cit. on p. 1).

[2] E. D'Hoker and J. Kaidi, *Lectures on modular forms and strings*, 2022,
arXiv: 2208.07242 [hep-th] (cit. on p. 1).

[3] S. Murthy, *Black holes and modular forms in string theory*, 2023,
arXiv: 2305.11732 [hep-th] (cit. on p. 1).

[4] A. Beléndez, C. Villalobos, D. Méndez, T. Vázquez and C. Neipp,
Exact solution for the nonlinear pendulum, Revista Brasileira de Ensino de Física **29** (2007)
(cit. on pp. 1, 2).

[5] DLMF Editors, *NIST Digital Library of Mathematical Functions*,
https://dlmf.nist.gov/, [Online; accessed 15 June 2023], 2023 (cit. on p. 3).

[6] H. Cohen and F. Strömberg, *Modular Forms: A Classical Approach*,
Graduate Studies in Mathematics 179, American Mathematical Society, 2017,
ISBN: 978-0-8218-4947-7 (cit. on pp. 3, 5–8, 10–15, 55).

[7] A. J. Brizard, *A primer on elliptic functions with applications in classical mechanics*,
European Journal of Physics **30** (2009) 729,
URL: https://dx.doi.org/10.1088/0143-0807/30/4/007 (cit. on p. 3).

[8] A. O. L. Atkin and H. P. F. Swinnerton-Dyer, "Modular forms on noncongruence subgroups",
Combinatorics (Proc. Sympos. Pure Math., Vol. XIX, Univ. California, Los Angeles, Calif., 1968), 1971 1 (cit. on pp. 3, 9, 11, 14, 25, 43, 45, 61).

[9] F. Calegari, V. Dimitrov and Y. Tang, *The Unbounded Denominators Conjecture*, 2021,
URL: https://arxiv.org/abs/2109.09040 (cit. on pp. 3, 25).

[10] H. Magureanu, *Seiberg-Witten geometry, modular rational elliptic surfaces and BPS quivers*,
JHEP **2022** (2022), URL: https://doi.org/10.1007/JHEP05(2022)163 (cit. on p. 3).

[11] D. A. Hejhal, "On Eigenfunctions of the Laplacian for Hecke Triangle Groups",
Emerging Applications of Number Theory,
ed. by D. A. Hejhal, J. Friedman, M. C. Gutzwiller and A. M. Odlyzko,
New York, NY: Springer New York, 1999 291, ISBN: 978-1-4612-1544-8,
URL: https://doi.org/10.1007/978-1-4612-1544-8_11 (cit. on pp. 3, 17, 19, 20).

[12] M. Klug, M. Musty, S. Schiavone and J. Voight,
Numerical calculation of three-point branched covers of the projective line,
LMS Journal of Computation and Mathematics **17** (2014) 379 (cit. on pp. 3, 17, 28, 45).

[13] J. Sijsling and J. Voight, "On computing Belyi maps", *Numéro consacré au trimestre "Méthodes arithmétiques et applications", automne 2013*, vol. 2014/1, Publ. Math. Besançon Algèbre Théorie Nr. Presses Univ. Franche-Comté, Besançon, 2014 73 (cit. on pp. 3, 44, 45, 61).

[14] H. Monien, *The sporadic group J2, Hauptmodul and Belyi map*, 2017, URL: https://arxiv.org/abs/1703.05200 (cit. on pp. 3, 45, 46).

[15] H. Monien, *The sporadic group Co3, Hauptmodul and Belyi map*, 2018, URL: https://arxiv.org/abs/1802.06923 (cit. on pp. 3, 45, 46, 65).

[16] H. Cohen, "Haberland's formula and numerical computation of Petersson scalar products", *ANTS X—Proceedings of the Tenth Algorithmic Number Theory Symposium*, vol. 1, Open Book Ser. Math. Sci. Publ., Berkeley, CA, 2013 249, URL: https://doi.org/10.2140/obs.2013.1.249 (cit. on pp. 3, 54, 55).

[17] The LMFDB Collaboration, *The L-functions and modular forms database*, http://www.lmfdb.org, [Online; accessed 22 March 2022], 2022 (cit. on pp. 3, 11).

[18] D. Berghaus, *Computing Laplacian Eigenvalues at Arbitrary Precision Arithmetic*, MA thesis: University of Bonn, 2020 (cit. on pp. 4, 69, 70, 73, 75).

[19] P. Grinfeld and G. Strang, *Laplace eigenvalues on regular polygons: A series in 1/N*, Journal of Mathematical Analysis and Applications **385** (2012) 135, ISSN: 0022-247X, URL: http://www.sciencedirect.com/science/article/pii/S0022247X1100583X (cit. on pp. 4, 69).

[20] M. Boady, *Applications of Symbolic Computation to the Calculus of Moving Surfaces*, PhD thesis: Drexel University, 2016 (cit. on pp. 4, 69).

[21] R. S. Jones, *The fundamental Laplacian eigenvalue of the regular polygon with Dirichlet boundary conditions*, 2017, arXiv: 1712.06082 [math.NA] (cit. on pp. 4, 69).

[22] D. Berghaus, B. Georgiev, H. Monien and D. Radchenko, *On Dirichlet eigenvalues of regular polygons*, 2021, URL: https://arxiv.org/abs/2103.01057 (cit. on pp. 4, 69).

[23] D. Berghaus, R. S. Jones, H. Monien and D. Radchenko, *Computation of Laplacian eigenvalues of two-dimensional shapes with dihedral symmetry*, 2022, URL: https://arxiv.org/abs/2210.13229 (cit. on pp. 4, 69, 70, 75).

[24] T. Betcke and L. N. Trefethen, *Reviving the method of particular solutions*, SIAM Rev. **47** (2005) 469, ISSN: 0036-1445, URL: https://doi.org/10.1137/S0036144503437336 (cit. on pp. 4, 72).

[25] D. Berghaus, H. Monien and D. Radchenko, *On the computation of modular forms on noncongruence subgroups*, Submitted to Mathematics of Computation, 2022, URL: https://arxiv.org/abs/2207.13365 (cit. on pp. 5, 10, 17, 25, 43, 64).

[26] F. Diamond and J. Shurman, *A First Course in Modular Forms*, Graduate Texts in Mathematics, Springer New York, 2006, ISBN: 9780387272269, URL: https://books.google.de/books?id=EXZCAAAAQBAJ (cit. on pp. 5, 15).

[27] D. Zagier, *Elliptic Modular Forms and Their Applications*, The 1-2-3 of Modular Forms, Springer Berlin Heidelberg, 2008, ISBN: 978-3-540-74119-0 (cit. on pp. 5, 76).

[28] T. Miyake, *Modular Forms*, Springer Monographs in Mathematics, Springer, 1989 (cit. on pp. 5, 13).

[29] F. Strömberg, *Noncongruence subgroups and Maass waveforms*, Journal of Number Theory **199** (2019) 436, ISSN: 0022-314X, URL: https://nottingham-repository.worktribe.com/output/1125860 (cit. on pp. 7, 9, 17, 20, 21).

[30] W. W. Stothers, *Level and index in the modular group*, Proc. Roy. Soc. Edinburgh Sect. A **99** (1984) 115, ISSN: 0308-2105, URL: https://doi.org/10.1017/S0308210500025993 (cit. on p. 9).

[31] T. Hsu, *Identifying congruence subgroups of the modular group*, Proc. Amer. Math. Soc. **124** (1996) 1351, ISSN: 0002-9939, URL: https://doi.org/10.1090/S0002-9939-96-03496-X (cit. on p. 9).

[32] M. H. Millington, *Subgroups of the Classical Modular Group*, Journal of the London Mathematical Society **s2-1** (1969) 351, ISSN: 0024-6107, eprint: https://academic.oup.com/jlms/article-pdf/s2-1/1/351/2786190/s2-1-1-351.pdf, URL: https://doi.org/10.1112/jlms/s2-1.1.351 (cit. on p. 9).

[33] F. Strömberg, *Noncongruence subgroups and Maass Waveforms*, https://github.com/fredstro/noncongruence, [Online; accessed 22 March 2022], 2018 (cit. on p. 9).

[34] W. Stein, *Modular Forms, a Computational Approach*, Graduate Studies in Mathematics, American Mathematical Society, 2007, ISBN: 978-0-8218-3960-7, URL: https://bookstore.ams.org/gsm-79/ (cit. on pp. 11, 12, 56).

[35] W. A. Stein et. al., *Sage Mathematics Software (Version 9.2)*, https://www.sagemath.org/, [Online; accessed 22 March 2022], 2021 (cit. on pp. 11, 26, 49).

[36] The PARI Group, *PARI/GP version 2.13.2*, http://pari.math.u-bordeaux.fr/, [Online; accessed 22 March 2022], 2021 (cit. on pp. 11, 26, 40, 49).

[37] J. P. Serre, *A Course in Arithmetic*, Graduate Texts in Mathematics, Springer, 1973 (cit. on p. 11).

[38] D. A. Cox, *Primes of the Form x2+ny2: Fermat, Class Field Theory, and Complex Multiplication*, Wiley, 1989 (cit. on pp. 14, 51).

[39] J. Milne, *Modular Functions and Modular Forms*, 2017, URL: https://www.jmilne.org/math/CourseNotes/MF.pdf (cit. on p. 15).

[40] B. Selander and A. Strömbergsson, *Sextic coverings of genus two which are branched at three points*, Preprint (2003) (cit. on pp. 17, 20–22, 45, 52).

[41] F. Strömberg, *Maass Waveforms on ($\Gamma_0(N)$, χ) (Computational Aspects)*, Hyperbolic geometry and applications in quantum chaos and cosmology (2012) 187 (cit. on pp. 17, 20, 21).

[42] A. R. Booker, A. Strömbergsson and A. Venkatesh, *Effective computation of Maass cusp forms*, Int. Math. Res. Not. (2006) Art. ID 71281, 34, ISSN: 1073-7928, URL: https://doi.org/10.1155/IMRN/2006/71281 (cit. on pp. 17, 20).

[43] J. H. Bruinier and F. Strömberg, *Computation of Harmonic Weak Maass Forms*, Experimental Mathematics **21** (2012) 117, URL: https://doi.org/ (cit. on p. 17).

[44] J. Voight and J. Willis, "Computing Power Series Expansions of Modular Forms", *Computations with Modular Forms*, ed. by G. Böckle and G. Wiese, Springer, 2014 331 (cit. on pp. 17, 23).

[45] D. A. Hejhal, "On Eigenvalues of the Laplacian for Hecke Triangle Groups", *Zeta Functions in Geometry*, Tokyo, Japan: Mathematical Society of Japan, 1992 359, URL: https://doi.org/10.2969/aspm/02110359 (cit. on pp. 17, 71).

[46] W. A. Stein, F. Strömberg, S. Ehlen and N. Skoruppa et. al., *Purplesage (psage)*, https://github.com/fredstro/psage, [Online; accessed 22 March 2022], 2022 (cit. on pp. 22, 26).

[47] G. Berger, *Hecke operators on noncongruence subgroups*, C. R. Acad. Sci. Paris Sér. I Math. **319** (1994) 915, ISSN: 0764-4442 (cit. on p. 25).

[48] W.-C. W. Li, L. Long and Z. Yang, *Modular forms for noncongruence subgroups*, Q. J. Pure Appl. Math. **1** (2005) 205, ISSN: 1549-6724 (cit. on p. 25).

[49] A. J. Scholl, "Modular forms on noncongruence subgroups", *Séminaire de Théorie des Nombres, Paris 1985–86*, vol. 71, Progr. Math. Birkhäuser Boston, Boston, MA, 1987 199, URL: https://doi.org/10.1007/978-1-4757-4267-1_14 (cit. on pp. 25, 53).

[50] W. Y. Chen, *Moduli interpretations for noncongruence modular curves*, Math. Ann. **371** (2018) 41, ISSN: 0025-5831, URL: https://doi.org/10.1007/s00208-017-1575-6 (cit. on p. 25).

[51] A. K. Lenstra, H. W. Lenstra Jr. and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982) 515, ISSN: 0025-5831, URL: https://doi.org/10.1007/BF01457454 (cit. on p. 39).

[52] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2010 (cit. on pp. 25, 26, 33).

[53] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier and P. Zimmermann, *MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding*, ACM Trans. Math. Softw. **33** (2007) 13, ISSN: 0098-3500, URL: https://doi.org/10.1145/1236463.1236468 (cit. on p. 25).

[54] F. Johansson, *Arb: efficient arbitrary-precision midpoint-radius interval arithmetic*, IEEE Transactions on Computers **66** (8 2017) 1281 (cit. on pp. 25, 26, 49, 54).

[55] S. Behnel et al., *Cython: The best of both worlds*, Computing in Science & Engineering **13** (2011) 31 (cit. on p. 26).

[56] G. Van Rossum and F. L. Drake Jr, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995 (cit. on p. 26).

[57] B. W. Kernighan and D. M. Ritchie, *The C programming language*, 2006 (cit. on p. 26).

[58] F. Johansson, *Efficient implementation of elementary functions in the medium-precision range*, 2014, URL: https://arxiv.org/abs/1410.7176 (cit. on p. 26).

[59] F. Johansson, "Faster Arbitrary-Precision Dot Product and Matrix Multiplication", *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, 2019 15 (cit. on pp. 26, 33).

[60] D. Berghaus, *Code to compute Modular Forms and Eisenstein Series on Noncongruence Subgroups of PSL(2,$\mathbb{Z}$)*, https://github.com/David-Berghaus/q-expansion, [Online; accessed 15 June 2023], 2023 (cit. on p. 26).

[61] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, 1997, ISBN: 0898713617 (cit. on pp. 26, 27).

[62] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM Journal on Scientific and Statistical Computing **7** (1986) 856 (cit. on p. 27).

[63] A. Abdelfattah et al., *A survey of numerical linear algebra methods utilizing mixed-precision arithmetic*, The International Journal of High Performance Computing Applications **35** (2021) 344, URL: https://doi.org/10.1177/10943420211003313 (cit. on p. 30).

[64] S. M. Rump, *Approximate Inverses of Almost Singular Matrices Still Contain Useful Information*, Technical Report 90.1, Hamburg University of Technology (1990) (cit. on p. 30).

[65] S. M. Rump, *Inversion of extremely Ill-conditioned matrices in floating-point*, Japan Journal of Industrial and Applied Mathematics **26** (2009) 249 (cit. on p. 30).

[66] E. Carson and N. J. Higham, *A New Analysis of Iterative Refinement and Its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems*, SIAM Journal on Scientific Computing **39** (2017) 2834 (cit. on p. 30).

[67] J. H. Wilkinson, *Progress report on the Automatic Computing Engine*, Report MA/17/1024 (1948) (cit. on p. 32).

[68] J. Demmel et al., *Error bounds from extra-precise iterative refinement*, ACM Trans. Math. Software **32** (2006) 325, ISSN: 0098-3500, URL: https://doi.org/10.1145/1141885.1141894 (cit. on p. 32).

[69] I. Gohberg and V. Olshevsky, *Complexity of multiplication with vectors for structured matrices*, Linear Algebra and its Applications **202** (1994) 163, ISSN: 0024-3795, URL: https://www.sciencedirect.com/science/article/pii/0024379594901899 (cit. on p. 33).

[70] J. W. Cooley and J. W. Tukey,
An algorithm for the machine calculation of complex Fourier series,
Math. Comp. **19** (1965) 297, ISSN: 0025-5718, URL: https://doi.org/10.2307/2003354
(cit. on p. 34).

[71] C. R. Harris et al., *Array programming with NumPy*, Nature **585** (2020) 357,
URL: https://doi.org/10.1038/s41586-020-2649-2 (cit. on p. 35).

[72] P. Virtanen et al., *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*,
Nature Methods **17** (2020) 261 (cit. on p. 35).

[73] H. Cohen, *A Course in Computational Algebraic Number Theory*,
Graduate Texts in Mathematics 138, Springer, 1996 (cit. on p. 40).

[74] M. Richards, *Computing Covering Maps for subgroups of* $\Gamma(1)$ *with genus* ≥ 1,
PhD thesis: University of Oxford, 1995 (cit. on p. 41).

[75] G. V. Belyi, *Galois extensions of a maximal cyclotomic field*,
Izv. Akad. Nauk SSSR Ser. Mat. **43** (1979) 267, 479, ISSN: 0373-2436 (cit. on p. 43).

[76] G. V. Belyi, *Another proof of the three points theorem*, Sbornik: Mathematics **193** (2002) 329
(cit. on p. 43).

[77] S. Krämer,
Numerical calculation of automorphic functions for finite index subgroups of triangle groups,
PhD thesis: University of Bonn, 2015 (cit. on p. 45).

[78] M. Musty, S. Schiavone, J. Sijsling and J. Voight, "A database of Belyi maps",
Proceedings of the Thirteenth Algorithmic Number Theory Symposium, vol. 2, Open Book Ser.
Math. Sci. Publ., Berkeley, CA, 2019 375 (cit. on pp. 45, 52, 61, 62).

[79] F. Johansson, *A fast algorithm for reversion of power series*, Math. Comp. **84** (2015) 475,
ISSN: 0025-5718, URL: https://doi.org/10.1090/S0025-5718-2014-02857-3
(cit. on p. 49).

[80] R. P. Brent and H. T. Kung, *Fast algorithms for manipulating formal power series*,
J. Assoc. Comput. Mach. **25** (1978) 581, ISSN: 0004-5411,
URL: https://doi.org/10.1145/322092.322099 (cit. on p. 49).

[81] W. Li, T. Liu, L. Long and R. Ramakrishna et. al.,
Noncongruence modular forms and modularity,
https://aimath.org/pastworkshops/noncongruence.html,
[Online; accessed 22 March 2022], 2009 (cit. on pp. 53, 58, 61).

[82] A. J. Scholl, *Fourier coefficients of Eisenstein series on non-congruence subgroups*,
Mathematical Proceedings of the Cambridge Philosophical Society **99** (1986) 11
(cit. on pp. 53, 58).

[83] A. Fiori and C. Franc,
The unbounded denominators conjecture for the noncongruence subgroups of index 7,
Journal of Number Theory (2022), ISSN: 0022-314X, URL:
https://www.sciencedirect.com/science/article/pii/S0022314X22000130
(cit. on pp. 53, 58, 61).

[84] P. D. Nelson, *Evaluating modular forms on Shimura curves*, Math. Comp. **84** (2015) 2471, ISSN: 0025-5718, URL: https://doi.org/10.1090/S0025-5718-2015-02943-3 (cit. on p. 54).

[85] D. J. Collins, *Numerical computation of Petersson inner products and q-expansions*, 2018, URL: https://arxiv.org/abs/1802.09740 (cit. on pp. 54, 56).

[86] H. Cohen, "Expansions at cusps and Petersson products in Pari/GP", *Elliptic integrals, elliptic functions and modular forms in quantum field theory*, Texts Monogr. Symbol. Comput. Springer, Cham, 2019 161 (cit. on pp. 54–56).

[87] K. Haberland, *Perioden von Modulformen einer Variabler and Gruppencohomologie.*, Math. Nachr. **112** (1983) 245, ISSN: 0025-584X, URL: https://doi.org/10.1002/mana.19831120113 (cit. on p. 55).

[88] D. Berghaus, *Data of algebraic Eisenstein series for H_5 noncongruence subgroup*, https://github.com/David-Berghaus/algebraic-eisenstein-series, [Online; accessed 28 June 2023], 2023 (cit. on p. 60).

[89] D. Berghaus, H. Monien and D. Radchenko, *A Database of Modular Forms on Noncongruence Subgroups*, 2023, arXiv: 2301.02135 [math.NT] (cit. on pp. 61, 63).

[90] A. J. Best et al., "Computing Classical Modular Forms", *Arithmetic Geometry, Number Theory, and Computation*, ed. by J. S. Balakrishnan et al., Cham: Springer International Publishing, 2021 131, ISBN: 978-3-030-80914-0 (cit. on p. 61).

[91] S. Donnelly and J. Voight, "A Database of Hilbert Modular Forms", *Arithmetic Geometry, Number Theory, and Computation*, ed. by J. S. Balakrishnan et al., Cham: Springer International Publishing, 2021 365, ISBN: 978-3-030-80914-0 (cit. on p. 61).

[92] A. Hulpke, *GAP package TransGrp*, https://www.gap-system.org/Packages/transgrp.html, [Online; accessed 13 June 2023], 2022 (cit. on pp. 61, 62).

[93] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.12.2*, https://www.gap-system.org, [Online; accessed 13 June 2023], 2022 (cit. on p. 61).

[94] D. Berghaus, H. Monien and D. Radchenko, *Database of modular forms of Noncongruence Subgroups of PSL(2,$\mathbb{Z}$)*, https://github.com/David-Berghaus/noncong_database, [Online; accessed 11 October 2022], 2022 (cit. on p. 68).

[95] C. M. Judge, *Conformally converting cusps to cones*, Conform. Geom. Dyn. **2** (1998) 107, URL: https://doi.org/10.1090/S1088-4173-98-00024-1 (cit. on pp. 69, 75).

[96] C. M. Judge, *On the existence of Maass cusp forms on hyperbolic surfaces with cone points*, J. Amer. Math. Soc. **8** (1995) 715, ISSN: 0894-0347, URL: https://doi.org/10.2307/2152928 (cit. on pp. 69, 75).

[97] R. S. Phillips and P. Sarnak, *On cusp forms for co-finite subgroups of PSL(2,$\mathbb{R}$)*, Inventiones mathematicae **80** (1985) 339, ISSN: 1432-1297 (cit. on p. 71).

[98] T. Betcke,
The generalized singular value decomposition and the method of particular solutions,
SIAM J. Sci. Comput. **30** (2008) 1278, ISSN: 1064-8275,
URL: https://doi.org/10.1137/060651057 (cit. on p. 73).

[99] A. O. L. Atkin and J. Lehner, *Hecke operators on* $\Gamma_0(m)$, Math. Ann. **185** (1970) 134,
ISSN: 0025-5831, URL: https://doi.org/10.1007/BF01359701 (cit. on p. 75).

[100] D. Berghaus, *Master's thesis data*,
https://github.com/David-Berghaus/master-thesis-data,
[Online; accessed 11 October 2022], 2020 (cit. on p. 75).

[101] W. Duke, Ö. Imamoḡlu and Á. Tóth,
Cycle integrals of the j*-function and mock modular forms*, Ann. of Math. (2) **173** (2011) 947,
ISSN: 0003-486X, URL: https://doi.org/10.4007/annals.2011.173.2.8
(cit. on pp. 75–77).

[102] D. Berghaus, *Numerical approximations of traces of real singular moduli*,
https://github.com/David-Berghaus/real_singular_moduli_traces,
[Online; accessed 15 June 2023], 2023 (cit. on pp. 77, 95).

[103] D. Berghaus, H. Monien and D. Radchenko, *A database of subgroups of PSL(2,*$\mathbb{Z}$*)*,
https://github.com/David-Berghaus/psl2z_db, [Online; accessed 15 June 2023],
2023 (cit. on p. 80).

List of Figures

List of Tables