Mod 2 linear algebra and tabulation of rational eigenforms

Kiran S. Kedlaya

Department of Mathematics, University of California, San Diego
kedlaya@ucsd.edu
http://kskedlaya.org/slides/ (see also this SageMathCloud project)

Automorphic forms: theory and computation
King's College, London
September 9, 2016

Joint work *in progress* with Anna Medvedovsky (MPI, Bonn).

Kedlaya was supported by NSF grant DMS-1501214 and UCSD (Warschawski chair).

Contents

A fool's errand?

Over the past two decades, Cremona has developed a highly efficient algorithm for enumerating rational $\Gamma_0(N)$-newforms of weight 2 and their associated elliptic curves (which we now know exhausts all elliptic curves over $\mathbb{Q}$), documented in his book *Algorithms for Modular Elliptic Curves*.

Cremona also has developed a highly efficient C/C++ implementation of this algorithm, which to date has enumerated all elliptic curves over $\mathbb{Q}$ of conductor ≤ 379998 (see Pari, Magma, Sage, or LMFDB).

Further extension of these tables would have, among other applications, consequences for the effective solution of S-unit equations; see arXiv:1605.06079 (von Känel-Matschke).

Is there room for improvement here? It is unlikely that any easy optimization in the algorithm or implementation has been missed!

A fool's errand?

Over the past two decades, Cremona has developed a highly efficient algorithm for enumerating rational $\Gamma_0(N)$-newforms of weight 2 and their associated elliptic curves (which we now know exhausts all elliptic curves over $\mathbb{Q}$), documented in his book *Algorithms for Modular Elliptic Curves*.

Cremona also has developed a highly efficient C/C++ implementation of this algorithm, which to date has enumerated all elliptic curves over $\mathbb{Q}$ of conductor ≤ 379998 (see Pari, Magma, Sage, or LMFDB).

Further extension of these tables would have, among other applications, consequences for the effective solution of S-unit equations; see arXiv:1605.06079 (von Känel-Matschke).

Is there room for improvement here? It is unlikely that any easy optimization in the algorithm or implementation has been missed!

A fool's errand?

Over the past two decades, Cremona has developed a highly efficient algorithm for enumerating rational $\Gamma_0(N)$-newforms of weight 2 and their associated elliptic curves (which we now know exhausts all elliptic curves over $\mathbb{Q}$), documented in his book *Algorithms for Modular Elliptic Curves*.

Cremona also has developed a highly efficient C/C++ implementation of this algorithm, which to date has enumerated all elliptic curves over $\mathbb{Q}$ of conductor ≤ 379998 (see Pari, Magma, Sage, or LMFDB).

Further extension of these tables would have, among other applications, consequences for the effective solution of S-unit equations; see arXiv:1605.06079 (von Känel-Matschke).

Is there room for improvement here? It is unlikely that any easy optimization in the algorithm or implementation has been missed!

A fool's errand?

Over the past two decades, Cremona has developed a highly efficient algorithm for enumerating rational $\Gamma_0(N)$-newforms of weight 2 and their associated elliptic curves (which we now know exhausts all elliptic curves over $\mathbb{Q}$), documented in his book *Algorithms for Modular Elliptic Curves*.

Cremona also has developed a highly efficient C/C++ implementation of this algorithm, which to date has enumerated all elliptic curves over $\mathbb{Q}$ of conductor ≤ 379998 (see Pari, Magma, Sage, or LMFDB).

Further extension of these tables would have, among other applications, consequences for the effective solution of S-unit equations; see arXiv:1605.06079 (von Känel-Matschke).

Is there room for improvement here? It is unlikely that any easy optimization in the algorithm or implementation has been missed!

Perhaps not...

Most positive integers do not occur as conductors of rational elliptic curves. For example, in the range 378000-378999, this LMFDB query returns 5885 curves of 566 different conductors:

```
sage: load("ec-378000-378999.sage");
sage: l = [EllipticCurve(i) for i in data];
sage: l2 = [i.conductor() for i in l];
sage: s = set(l2);
sage: len(s)
566
```

This is consistent with the expectation that the number of positive integers up to X which occur as conductors is $\sim CX^{5/6}$ (this being true for heights).

Perhaps not...

Most positive integers do not occur as conductors of rational elliptic curves. For example, in the range 378000-378999, this LMFDB query returns 5885 curves of 566 different conductors:

```
sage: load("ec-378000-378999.sage");
sage: l = [EllipticCurve(i) for i in data];
sage: l2 = [i.conductor() for i in l];
sage: s = set(l2);
sage: len(s)
566
```

This is consistent with the expectation that the number of positive integers up to X which occur as conductors is $\sim CX^{5/6}$ (this being true for heights).

Perhaps not...

Most positive integers do not occur as conductors of rational elliptic curves. For example, in the range 378000-378999, this LMFDB query returns 5885 curves of 566 different conductors:

```
sage: load("ec-378000-378999.sage");
sage: l = [EllipticCurve(i) for i in data];
sage: l2 = [i.conductor() for i in l];
sage: s = set(l2);
sage: len(s)
566
```

This is consistent with the expectation that the number of positive integers up to X which occur as conductors is $\sim CX^{5/6}$ (this being true for heights).

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

TSA Precheck for conductors?)

For a given N, the rate-limiting step in Cremona's computation of the elliptic curves of conductor N occurs at the very beginning, before one knows whether or not any such curves exist. (More on this shortly.)

Consequently, one can try to speed up the tabulation by prefixing a fast computation that cuts down the list of eligible conductors. For example, Cremona already excludes N divisible by 2^9, 3^6, or p^3 for any prime $p > 3$; but these form only 1.6% of all levels.

We discuss some precomputations based on:

- linear algebra over $\mathbb{F}_2$;
- results about mod 2 modular forms, including Serre reciprocity.

This will serve as an excuse to discuss some questions about mod 2 Hecke algebra multiplicities to which we have not found complete answers.

Contents

A high-level description

Positive integer N (whose divisors are already done)

$\downarrow$

Rational (old and new) Hecke eigensystems for $S_2(\Gamma_0(N), \mathbb{Q})$

$\downarrow$

Rational newforms for $S_2(\Gamma_0(N), \mathbb{Q})$

$\downarrow$

Elliptic curves over $\mathbb{Q}$ of conductor N

The first step is rate-limiting because very few possibilities survive to the later steps. We thus focus on this step; see Cremona's book for discussion of the others.

A high-level description

$$
\begin{array}{c}
\text{Positive integer } N \text{ (whose divisors are already done)} \\
\downarrow \\
\text{Rational (old and new) Hecke eigensystems for } S_2(\Gamma_0(N), \mathbb{Q}) \\
\downarrow \\
\text{Rational newforms for } S_2(\Gamma_0(N), \mathbb{Q}) \\
\downarrow \\
\text{Elliptic curves over } \mathbb{Q} \text{ of conductor } N
\end{array}
$$

The first step is rate-limiting because very few possibilities survive to the later steps. We thus focus on this step; see Cremona's book for discussion of the others.

Computation of eigensystems

Cremona computes not with $S_2(\Gamma_0(N), \mathbb{Q})$, but with the homology of $X_0(N)$ as represented via Manin's modular symbols. For $p \nmid N$, the action of T_p is given by a sparse[1] integer[2] matrix. By strong multiplicity one, for the purpose of distinguishing eigensystems we may ignore T_p for $p|N$ (which are not implemented by Cremona).

Let p be the smallest prime not dividing N. The rate-limiting step is to compute the kernel of $T_p - a_p$ for each $a_p \in [-2\sqrt{p}, 2\sqrt{p}] \cap \mathbb{Z}$. This involves matrices of size $\sim N/12$.

By contrast, the dimensions of these kernels are far smaller. Thus, further decomposing these kernels into joint eigenspaces is of negligible difficulty.

[1] This crucial property would be lost if we restricted to newforms; we must thus identify new eigensystems as such solely by comparing them to old eigensystems.

[2] In some cases, Cremona's code returns $2T_p$ because the computed matrix of $2T_p$ is not integral. However, we only work with the minus eigenspace for complex conjugation, where we have yet to observe a failure of integrality.

Computation of eigensystems

Cremona computes not with $S_2(\Gamma_0(N), \mathbb{Q})$, but with the homology of $X_0(N)$ as represented via Manin's modular symbols. For $p \nmid N$, the action of T_p is given by a sparse[1] integer[2] matrix. By strong multiplicity one, for the purpose of distinguishing eigensystems we may ignore T_p for $p|N$ (which are not implemented by Cremona).

Let p be the smallest prime not dividing N. The rate-limiting step is to compute the kernel of $T_p - a_p$ for each $a_p \in [-2\sqrt{p}, 2\sqrt{p}] \cap \mathbb{Z}$. This involves matrices of size $\sim N/12$.

By contrast, the dimensions of these kernels are far smaller. Thus, further decomposing these kernels into joint eigenspaces is of negligible difficulty.

[1] This crucial property would be lost if we restricted to newforms; we must thus identify new eigensystems as such solely by comparing them to old eigensystems.

[2] In some cases, Cremona's code returns $2T_p$ because the computed matrix of $2T_p$ is not integral. However, we only work with the minus eigenspace for complex conjugation, where we have yet to observe a failure of integrality.

Computation of eigensystems

Cremona computes not with $S_2(\Gamma_0(N), \mathbb{Q})$, but with the homology of $X_0(N)$ as represented via Manin's modular symbols. For $p \nmid N$, the action of T_p is given by a sparse[1] integer[2] matrix. By strong multiplicity one, for the purpose of distinguishing eigensystems we may ignore T_p for $p|N$ (which are not implemented by Cremona).

Let p be the smallest prime not dividing N. The rate-limiting step is to compute the kernel of $T_p - a_p$ for each $a_p \in [-2\sqrt{p}, 2\sqrt{p}] \cap \mathbb{Z}$. This involves matrices of size $\sim N/12$.

By contrast, the dimensions of these kernels are far smaller. Thus, further decomposing these kernels into joint eigenspaces is of negligible difficulty.

[1] This crucial property would be lost if we restricted to newforms; we must thus identify new eigensystems as such solely by comparing them to old eigensystems.

[2] In some cases, Cremona's code returns $2T_p$ because the computed matrix of $2T_p$ is not integral. However, we only work with the minus eigenspace for complex conjugation, where we have yet to observe a failure of integrality.

Linear algebra (not) over $\mathbb{Q}$

The complexity of linear algebra over a field is typically costed in terms of field operations. This gives reasonable results over a finite field.

However, this costing model does not work well over $\mathbb{Q}$: the cost of arithmetic operations depends on the heights of the operands. Moreover, direct use of conventional algorithms (e.g., Gaussian elimination) tends to incur *intermediate coefficient blowup*: heights of matrix entries increase steadily throughout the computation.

However, one can typically bound the height of the result of a computation (e.g., determinant) directly in terms of the heights of the entries. One can then use a *multimodular* approach: reduce from $\mathbb{Q}$ to various finite fields, do the linear algebra there, and reconstruct the answer using the Chinese remainder theorem. For instance, this is implemented in Magma and FLINT (the latter wrapped in Sage).

Linear algebra (not) over $\mathbb{Q}$

The complexity of linear algebra over a field is typically costed in terms of field operations. This gives reasonable results over a finite field.

However, this costing model does not work well over $\mathbb{Q}$: the cost of arithmetic operations depends on the heights of the operands. Moreover, direct use of conventional algorithms (e.g., Gaussian elimination) tends to incur *intermediate coefficient blowup*: heights of matrix entries increase steadily throughout the computation.

However, one can typically bound the height of the result of a computation (e.g., determinant) directly in terms of the heights of the entries. One can then use a *multimodular* approach: reduce from $\mathbb{Q}$ to various finite fields, do the linear algebra there, and reconstruct the answer using the Chinese remainder theorem. For instance, this is implemented in Magma and FLINT (the latter wrapped in Sage).

Linear algebra (not) over $\mathbb{Q}$

The complexity of linear algebra over a field is typically costed in terms of field operations. This gives reasonable results over a finite field.

However, this costing model does not work well over $\mathbb{Q}$: the cost of arithmetic operations depends on the heights of the operands. Moreover, direct use of conventional algorithms (e.g., Gaussian elimination) tends to incur *intermediate coefficient blowup*: heights of matrix entries increase steadily throughout the computation.

However, one can typically bound the height of the result of a computation (e.g., determinant) directly in terms of the heights of the entries. One can then use a *multimodular* approach: reduce from $\mathbb{Q}$ to various finite fields, do the linear algebra there, and reconstruct the answer using the Chinese remainder theorem. For instance, this is implemented in Magma and FLINT (the latter wrapped in Sage).

Short-circuiting the multimodular approach

To compute the kernel of the matrix representing $T_p - a_p$ on modular symbols, it is not necessary to use as many primes as theoretically required by the height bound. One can instead guess the kernel based on fewer primes, and then directly verify the result by multiplying with the original matrix. This is particularly cheap because the matrix is sparse.

In practice, Cremona works modulo the single prime $\ell = 2^{30} - 35$; experimentally, this always suffices to determine the kernel over $\mathbb{Q}$. It would be worth comparing with a multimodular approach starting from $\ell = 2$ and guessing after each prime.

Short-circuiting the multimodular approach

To compute the kernel of the matrix representing $T_p - a_p$ on modular symbols, it is not necessary to use as many primes as theoretically required by the height bound. One can instead guess the kernel based on fewer primes, and then directly verify the result by multiplying with the original matrix. This is particularly cheap because the matrix is sparse.

In practice, Cremona works modulo the single prime $\ell = 2^{30} - 35$; experimentally, this always suffices to determine the kernel over $\mathbb{Q}$. It would be worth comparing with a multimodular approach starting from $\ell = 2$ and guessing after each prime.

Linear algebra over finite fields (Magma)

How does the complexity of linear algebra over $\mathbb{F}_\ell$ vary with ℓ? A sensible behavior is exhibited by Magma 2.21-11:

```
> C := ModularSymbols(100001, 2, -1);
> M := HeckeOperator(C, 2);
> M2 := Matrix(GF(2), M); time Rank(M2);
9047
Time: 1.710
> M3 := Matrix(GF(3), M); time Rank(M3);
9085
Time: 4.220
> p := 2^30 - 35;
> Mp := Matrix(GF(p), M); time Rank(Mp);
9091
Time: 17.160
```

Linear algebra over finite fields (Magma)

How does the complexity of linear algebra over $\mathbb{F}_\ell$ vary with ℓ? A sensible behavior is exhibited by Magma 2.21-11:

```
> C := ModularSymbols(100001, 2, -1);
> M := HeckeOperator(C, 2);
> M2 := Matrix(GF(2), M); time Rank(M2);
9047
Time: 1.710
> M3 := Matrix(GF(3), M); time Rank(M3);
9085
Time: 4.220
> p := 2^30 - 35;
> Mp := Matrix(GF(p), M); time Rank(Mp);
9091
Time: 17.160
```

Linear algebra over finite fields (Sage)

By contrast, in Sage, linear algebra over $\mathbb{F}_\ell$ is far worse than Magma for $\ell > 2$ (and essentially unusable for $p > 2^{16}$), but notably better for $\ell = 2$ (see this demo).

This is because for $\ell = 2$, Sage uses the m4ri library by Gregory Bard, which implements the "Method of four Russians" algorithm. This algorithm makes special[3] use of the graph-theoretic interpretation of binary matrices, in order to save some logarithmic factors ahead of the Strassen crossover.

This raises the question: can we gain useful prescreening information by working solely over $\mathbb{F}_2$? A precise analysis of this question involves some interesting ingredients!

[3] There is a bitslicing approach that adapts the method to other small finite fields, but serious implementation seems not to have been pursued. See arXiv:0901.1413.

Linear algebra over finite fields (Sage)

By contrast, in Sage, linear algebra over $\mathbb{F}_\ell$ is far worse than Magma for $\ell > 2$ (and essentially unusable for $p > 2^{16}$), but notably better for $\ell = 2$ (see this demo).

This is because for $\ell = 2$, Sage uses the m4ri library by Gregory Bard, which implements the "Method of four Russians" algorithm. This algorithm makes special[3] use of the graph-theoretic interpretation of binary matrices, in order to save some logarithmic factors ahead of the Strassen crossover.

This raises the question: can we gain useful prescreening information by working solely over $\mathbb{F}_2$? A precise analysis of this question involves some interesting ingredients!

[3] There is a bitslicing approach that adapts the method to other small finite fields, but serious implementation seems not to have been pursued. See arXiv:0901.1413.

Linear algebra over finite fields (Sage)

By contrast, in Sage, linear algebra over $\mathbb{F}_\ell$ is far worse than Magma for $\ell > 2$ (and essentially unusable for $p > 2^{16}$), but notably better for $\ell = 2$ (see this demo).

This is because for $\ell = 2$, Sage uses the m4ri library by Gregory Bard, which implements the "Method of four Russians" algorithm. This algorithm makes special[3] use of the graph-theoretic interpretation of binary matrices, in order to save some logarithmic factors ahead of the Strassen crossover.

This raises the question: can we gain useful prescreening information by working solely over $\mathbb{F}_2$? A precise analysis of this question involves some interesting ingredients!

[3] There is a bitslicing approach that adapts the method to other small finite fields, but serious implementation seems not to have been pursued. See arXiv:0901.1413.

Contents

A general framework for prescreening

To simplify matters, hereafter we only consider odd N, so that we can take $p = 2$ in Cremona's algorithm. In this case, it is natural to modify our high-level description as follows:

$$\begin{array}{c}
\text{Odd positive integer } N, \text{ integer } e \in \{0,1\} \\
\downarrow \\
\text{Rational Hecke eigensystems for } S_2(\Gamma_0(N), \mathbb{Q}) \text{ with } a_2 \equiv e \pmod 2 \\
\downarrow \\
\text{Rational newforms for } S_2(\Gamma_0(N), \mathbb{Q}) \text{ with } a_2 \equiv e \pmod 2 \\
\downarrow \\
\text{Elliptic curves over } \mathbb{Q} \text{ of conductor } N \text{ with } a_2 \equiv e \pmod 2
\end{array}$$

Reminder: the options for a_2 are $-2, 0, 2$ if $e = 0$, and $-1, 1$ if $e = 1$.

A general framework for prescreening

To simplify matters, hereafter we only consider odd N, so that we can take $p = 2$ in Cremona's algorithm. In this case, it is natural to modify our high-level description as follows:

$$\begin{array}{c}
\text{Odd positive integer } N, \text{ integer } e \in \{0,1\} \\
\big\downarrow \\
\text{Rational Hecke eigensystems for } S_2(\Gamma_0(N), \mathbb{Q}) \text{ with } a_2 \equiv e \pmod{2} \\
\big\downarrow \\
\text{Rational newforms for } S_2(\Gamma_0(N), \mathbb{Q}) \text{ with } a_2 \equiv e \pmod{2} \\
\big\downarrow \\
\text{Elliptic curves over } \mathbb{Q} \text{ of conductor } N \text{ with } a_2 \equiv e \pmod{2}
\end{array}$$

Reminder: the options for a_2 are $-2, 0, 2$ if $e = 0$, and $-1, 1$ if $e = 1$.

Hecke matrices mod 2: some stupid models

If the matrix of the $\mathbb{Z}$-matrix $T_2 - e$ is invertible mod 2, then its determinant is odd, so T_2 has no $\mathbb{Q}$-eigenvalues congruent to e mod 2. How often does this occur?

Baseline: a random matrix over $\mathbb{F}_2$ fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty}(1 - 2^{-n}) \approx 71.1\%.$$

Since T_2 is self-adjoint in some basis, a better baseline is a random *symmetric* matrix over $\mathbb{F}_2$, which fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty}(1 - 2^{1-2n}) \approx 58.1\%.$$

Hecke matrices mod 2: some stupid models

If the matrix of the $\mathbb{Z}$-matrix $T_2 - e$ is invertible mod 2, then its determinant is odd, so T_2 has no $\mathbb{Q}$-eigenvalues congruent to e mod 2. How often does this occur?

Baseline: a random matrix over $\mathbb{F}_2$ fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty} (1 - 2^{-n}) \approx 71.1\%.$$

Since T_2 is self-adjoint in some basis, a better baseline is a random *symmetric* matrix over $\mathbb{F}_2$, which fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty} (1 - 2^{1-2n}) \approx 58.1\%.$$

Hecke matrices mod 2: some stupid models

If the matrix of the $\mathbb{Z}$-matrix $T_2 - e$ is invertible mod 2, then its determinant is odd, so T_2 has no $\mathbb{Q}$-eigenvalues congruent to e mod 2. How often does this occur?

Baseline: a random matrix over $\mathbb{F}_2$ fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty} (1 - 2^{-n}) \approx 71.1\%.$$

Since T_2 is self-adjoint in some basis, a better baseline is a random *symmetric* matrix over $\mathbb{F}_2$, which fails to be invertible with probability

$$1 - \prod_{n=1}^{\infty} (1 - 2^{1-2n}) \approx 58.1\%.$$

Why are these models stupid?

These models are stupid for (at least) two reasons.

- For N composite, we get a contribution from oldforms, so the probability that $T_2 - e$ has nontrivial kernel mod 2 is much higher than for N prime. (This also makes this test nearly useless for N compossite.)
- The existence of a nontrivial kernel mod 2 is explained by Serre reciprocity. Consequently, the correct probability modeling will be given by certain heuristics concerning the distribution of number fields.

Why are these models stupid?

These models are stupid for (at least) two reasons.

- For N composite, we get a contribution from oldforms, so the probability that $T_2 - e$ has nontrivial kernel mod 2 is much higher than for N prime. (This also makes this test nearly useless for N compossite.)
- The existence of a nontrivial kernel mod 2 is explained by Serre reciprocity. Consequently, the correct probability modeling will be given by certain heuristics concerning the distribution of number fields.

Why are these models stupid?

These models are stupid for (at least) two reasons.

- For N composite, we get a contribution from oldforms, so the probability that $T_2 - e$ has nontrivial kernel mod 2 is much higher than for N prime. (This also makes this test nearly useless for N compossite.)
- The existence of a nontrivial kernel mod 2 is explained by Serre reciprocity. Consequently, the correct probability modeling will be given by certain heuristics concerning the distribution of number fields.

Ranks mod 2: data for prime levels

For prime $N < 500000$ and $e = 0, 1$, we used Sage (calling Cremona's eclib and Bard's m4ri) to determine whether $T_2 - e$ has nontrivial kernel mod 2. Estimated runtime: about 3 weeks on 24 Intel Xeon X5690 cores (3.47GHz).

Results (see this demo for some data analysis):

$N \pmod 8$	$e = 0$	$e = 1$
1	16.8%	Always
3	Always for $N > 3$	Always for $N > 163$
5	42.2%	Always for $N > 37$
7	17.3%	47.9%

We will explain the "always" statements a bit later. In any case, for prime N, 38.7% of the kernel calculations over $\mathbb{Q}$ can be short-circuited by working over $\mathbb{F}_2$; that said, prime levels are already handled by Stein-Watkins and Bennett well beyond the range of interest.

Ranks mod 2: data for prime levels

For prime $N < 500000$ and $e = 0, 1$, we used Sage (calling Cremona's eclib and Bard's m4ri) to determine whether $T_2 - e$ has nontrivial kernel mod 2. Estimated runtime: about 3 weeks on 24 Intel Xeon X5690 cores (3.47GHz).

Results (see this demo for some data analysis):

$N \pmod 8$	$e = 0$	$e = 1$
1	16.8%	Always
3	Always for $N > 3$	Always for $N > 163$
5	42.2%	Always for $N > 37$
7	17.3%	47.9%

We will explain the "always" statements a bit later. In any case, for prime N, 38.7% of the kernel calculations over $\mathbb{Q}$ can be short-circuited by working over $\mathbb{F}_2$; that said, prime levels are already handled by Stein-Watkins and Bennett well beyond the range of interest.

Ranks mod 2: data for prime levels

For prime $N < 500000$ and $e = 0, 1$, we used Sage (calling Cremona's eclib and Bard's m4ri) to determine whether $T_2 - e$ has nontrivial kernel mod 2. Estimated runtime: about 3 weeks on 24 Intel Xeon X5690 cores (3.47GHz).

Results (see this demo for some data analysis):

$N \pmod 8$	$e = 0$	$e = 1$
1	16.8%	Always
3	Always for $N > 3$	Always for $N > 163$
5	42.2%	Always for $N > 37$
7	17.3%	47.9%

We will explain the "always" statements a bit later. In any case, for prime N, 38.7% of the kernel calculations over $\mathbb{Q}$ can be short-circuited by working over $\mathbb{F}_2$; that said, prime levels are already handled by Stein-Watkins and Bennett well beyond the range of interest.

Ranks mod 2: data for prime levels

For prime $N < 500000$ and $e = 0, 1$, we used Sage (calling Cremona's eclib and Bard's m4ri) to determine whether $T_2 - e$ has nontrivial kernel mod 2. Estimated runtime: about 3 weeks on 24 Intel Xeon X5690 cores (3.47GHz).

Results (see this demo for some data analysis):

$N \pmod 8$	$e = 0$	$e = 1$
1	16.8%	Always
3	Always for $N > 3$	Always for $N > 163$
5	42.2%	Always for $N > 37$
7	17.3%	47.9%

We will explain the "always" statements a bit later. In any case, for prime N, 38.7% of the kernel calculations over $\mathbb{Q}$ can be short-circuited by working over $\mathbb{F}_2$; that said, prime levels are already handled by Stein-Watkins and Bennett well beyond the range of interest.

Contents

Eigenvalue multiplicities

This time, instead of simply testing whether $T_2 - e$ is invertible mod 2, let us compute the multiplicity of 0 as a generalized eigenvalue of the reduced matrix. This equals the number of eigenvalues of T_2 in $\overline{\mathbb{Q}}_2$ in the open unit ball around e. (This computation is a bit more expensive than testing invertibility, but still quite efficient.)

This time, we can rule out (N, e) if we can account for the entire multiplicity using mod 2 representations which cannot lift to $\mathbb{Q}$ (e.g., because they take values in a larger field than $\mathbb{F}_2$). For N composite, we also remove the multiplicity coming from divisors of N.

Warning: the dimension of the kernel mod 2 is not mathematically significant! It is an artifact of the choice of basis used to express T_2, which is not the one coming from the integral Hecke algebra.

Eigenvalue multiplicities

This time, instead of simply testing whether $T_2 - e$ is invertible mod 2, let us compute the multiplicity of 0 as a generalized eigenvalue of the reduced matrix. This equals the number of eigenvalues of T_2 in $\overline{\mathbb{Q}}_2$ in the open unit ball around e. (This computation is a bit more expensive than testing invertibility, but still quite efficient.)

This time, we can rule out (N, e) if we can account for the entire multiplicity using mod 2 representations which cannot lift to $\mathbb{Q}$ (e.g., because they take values in a larger field than $\mathbb{F}_2$). For N composite, we also remove the multiplicity coming from divisors of N.

Warning: the dimension of the kernel mod 2 is not mathematically significant! It is an artifact of the choice of basis used to express T_2, which is not the one coming from the integral Hecke algebra.

Eigenvalue multiplicities

This time, instead of simply testing whether $T_2 - e$ is invertible mod 2, let us compute the multiplicity of 0 as a generalized eigenvalue of the reduced matrix. This equals the number of eigenvalues of T_2 in $\overline{\mathbb{Q}}_2$ in the open unit ball around e. (This computation is a bit more expensive than testing invertibility, but still quite efficient.)

This time, we can rule out (N, e) if we can account for the entire multiplicity using mod 2 representations which cannot lift to $\mathbb{Q}$ (e.g., because they take values in a larger field than $\mathbb{F}_2$). For N composite, we also remove the multiplicity coming from divisors of N.

Warning: the dimension of the kernel mod 2 is not mathematically significant! It is an artifact of the choice of basis used to express T_2, which is not the one coming from the integral Hecke algebra.

Some data analysis

Data collection using Google Compute Engine

Google Compute Engine is a cloud platform (like Amazon EC2) which seems particularly well-adapted for mathematics research. SageMathCloud is built on GCE, and LMFDB is hosted using GCE.

Using GCE, one can easily[4] run a trivially parallel computation on large numbers of virtual machines. Pricing is based on memory, disk usage, and CPU-minutes, with hugely preferential pricing for *preemptible* VMs.

We used VMs totaling 128 cores[5], to compute eigenvalue multiplicities of $T_2 - e$ for $e = 0, 1$ for all odd $N < 200000$. This took 5.5 days[6] at a cost[7] of about $250. See this demo for some data analysis.

[4] At least using free software! Using Magma this way is not straightforward.

[5] These only ran at 2.2GHz, but had much bigger L3 cache than my “faster” 24-core machine; in practice, this seemed to provide some advantage.

[6] Wall time. Due to preemptibility and other factors, CPU uptime was somewhat less.

[7] This “cost” was actually a promotional credit; we did not optimize it heavily.

Data collection using Google Compute Engine

Google Compute Engine is a cloud platform (like Amazon EC2) which seems particularly well-adapted for mathematics research. SageMathCloud is built on GCE, and LMFDB is hosted using GCE.

Using GCE, one can easily[4] run a trivially parallel computation on large numbers of virtual machines. Pricing is based on memory, disk usage, and CPU-minutes, with hugely preferential pricing for *preemptible* VMs.

We used VMs totaling 128 cores[5], to compute eigenvalue multiplicities of $T_2 - e$ for $e = 0, 1$ for all odd $N < 200000$. This took 5.5 days[6] at a cost[7] of about $250. See this demo for some data analysis.

[4] At least using free software! Using Magma this way is not straightforward.

[5] These only ran at 2.2GHz, but had much bigger L3 cache than my "faster" 24-core machine; in practice, this seemed to provide some advantage.

[6] Wall time. Due to preemptibility and other factors, CPU uptime was somewhat less.

[7] This "cost" was actually a promotional credit; we did not optimize it heavily.

Data collection using Google Compute Engine

Google Compute Engine is a cloud platform (like Amazon EC2) which seems particularly well-adapted for mathematics research. SageMathCloud is built on GCE, and LMFDB is hosted using GCE.

Using GCE, one can easily[4] run a trivially parallel computation on large numbers of virtual machines. Pricing is based on memory, disk usage, and CPU-minutes, with hugely preferential pricing for *preemptible* VMs.

We used VMs totaling 128 cores[5], to compute eigenvalue multiplicities of $T_2 - e$ for $e = 0, 1$ for all odd $N < 200000$. This took 5.5 days[6] at a cost[7] of about $250. See this demo for some data analysis.

[4] At least using free software! Using Magma this way is not straightforward.

[5] These only ran at 2.2GHz, but had much bigger L3 cache than my "faster" 24-core machine; in practice, this seemed to provide some advantage.

[6] Wall time. Due to preemptibility and other factors, CPU uptime was somewhat less.

[7] This "cost" was actually a promotional credit; we did not optimize it heavily.

Contents

Lower bounds for multiplicities

Suppose (for convenience) that N is squarefree. We will obtain the following lower bounds on the eigenvalue multiplicities mod 2:

$N \pmod 8$	Multiplicity for $e=0$	Multiplicity for $e=1$
1	0	$2\overline{\#}\frac{K(N)}{\langle\mathfrak{p}_2\rangle} + \overline{\#}K(-N) + 1$
3	$\overline{\#}K_2(-N) - \overline{\#}K(-N)$	$\overline{\#}K(N) + 2\overline{\#}K(-N)$
5	$\overline{\#}K_2(N) - \overline{\#}K(N)$	$2\overline{\#}K(N) + \overline{\#}K(-N)$
7	0	$\overline{\#}K(N) + 2\overline{\#}\frac{K(-N)}{\langle\mathfrak{p}_2\rangle}$

Notation in this table:

- for any abelian group G, $\overline{\#}G = \frac{1}{2}(\#G_{\text{odd}} - 1)$;
- $K(\pm N), K_2(\pm N)$ are the class group, 2-ray class group of $\mathbb{Q}(\sqrt{\pm N})$;
- $\mathfrak{p}_2$ is a prime of $\mathbb{Q}(\sqrt{\pm N})$ above 2.

We will also see from data that these bounds are *very often* not best possible.

Lower bounds for multiplicities

Suppose (for convenience) that N is squarefree. We will obtain the following lower bounds on the eigenvalue multiplicities mod 2:

$N \pmod 8$	Multiplicity for $e = 0$	Multiplicity for $e = 1$
1	0	$2\overline{\#}\frac{K(N)}{\langle \mathfrak{p}_2 \rangle} + \overline{\#}K(-N) + 1$
3	$\overline{\#}K_2(-N) - \overline{\#}K(-N)$	$\overline{\#}K(N) + 2\overline{\#}K(-N)$
5	$\overline{\#}K_2(N) - \overline{\#}K(N)$	$2\overline{\#}K(N) + \overline{\#}K(-N)$
7	0	$\overline{\#}K(N) + 2\overline{\#}\frac{K(-N)}{\langle \mathfrak{p}_2 \rangle}$

Notation in this table:

- for any abelian group G, $\overline{\#}G = \frac{1}{2}(\#G_{\mathrm{odd}} - 1)$;
- $K(\pm N), K_2(\pm N)$ are the class group, 2-ray class group of $\mathbb{Q}(\sqrt{\pm N})$;
- $\mathfrak{p}_2$ is a prime of $\mathbb{Q}(\sqrt{\pm N})$ above 2.

We will also see from data that these bounds are *very often* not best possible.

Lower bounds for multiplicities

Suppose (for convenience) that N is squarefree. We will obtain the following lower bounds on the eigenvalue multiplicities mod 2:

$N \pmod 8$	Multiplicity for $e = 0$	Multiplicity for $e = 1$
1	0	$2\overline{\#}\frac{K(N)}{\langle \mathfrak{p}_2 \rangle} + \overline{\#}K(-N) + 1$
3	$\overline{\#}K_2(-N) - \overline{\#}K(-N)$	$\overline{\#}K(N) + 2\overline{\#}K(-N)$
5	$\overline{\#}K_2(N) - \overline{\#}K(N)$	$2\overline{\#}K(N) + \overline{\#}K(-N)$
7	0	$\overline{\#}K(N) + 2\overline{\#}\frac{K(-N)}{\langle \mathfrak{p}_2 \rangle}$

Notation in this table:

- for any abelian group G, $\overline{\#}G = \frac{1}{2}(\#G_{\mathrm{odd}} - 1)$;
- $K(\pm N), K_2(\pm N)$ are the class group, 2-ray class group of $\mathbb{Q}(\sqrt{\pm N})$;
- $\mathfrak{p}_2$ is a prime of $\mathbb{Q}(\sqrt{\pm N})$ above 2.

We will also see from data that these bounds are *very often* not best possible.

Contributors to eigenvalue multiplicity

- Excluding the $+1$ for $N \equiv 1 \pmod 8$, each lower bound for $e = 1$ is a sum of contributions arising (via Serre reciprocity) from dihedral representations associated to characters of $G = \mathrm{Gal}(H/E)$, where $E = \mathbb{Q}(\sqrt{\pm N})$ and H is the maximal odd-order abelian unramified extension of K in which the primes above 2 split completely.
- Each lower bound for $e = 0$ is a sum of contributions arising from dihedral representations associated to characters of $G_2 = \mathrm{Gal}(H_2/E)$ not factoring through G, where H_2 is analogous to H except that ramification at 2 is now allowed.
- The extra contribution of 1 for $N \equiv 1 \pmod 8$, $e = 1$ comes from Eisenstein ideals above 2 in the Hecke algebra.

Contributors to eigenvalue multiplicity

- Excluding the $+1$ for $N \equiv 1 \pmod 8$, each lower bound for $e = 1$ is a sum of contributions arising (via Serre reciprocity) from dihedral representations associated to characters of $G = \mathrm{Gal}(H/E)$, where $E = \mathbb{Q}(\sqrt{\pm N})$ and H is the maximal odd-order abelian unramified extension of K in which the primes above 2 split completely.
- Each lower bound for $e = 0$ is a sum of contributions arising from dihedral representations associated to characters of $G_2 = \mathrm{Gal}(H_2/E)$ not factoring through G, where H_2 is analogous to H except that ramification at 2 is now allowed.
- The extra contribution of 1 for $N \equiv 1 \pmod 8$, $e = 1$ comes from Eisenstein ideals above 2 in the Hecke algebra.

Contributors to eigenvalue multiplicity

- Excluding the $+1$ for $N \equiv 1 \pmod 8$, each lower bound for $e = 1$ is a sum of contributions arising (via Serre reciprocity) from dihedral representations associated to characters of $G = \mathrm{Gal}(H/E)$, where $E = \mathbb{Q}(\sqrt{\pm N})$ and H is the maximal odd-order abelian unramified extension of K in which the primes above 2 split completely.
- Each lower bound for $e = 0$ is a sum of contributions arising from dihedral representations associated to characters of $G_2 = \mathrm{Gal}(H_2/E)$ not factoring through G, where H_2 is analogous to H except that ramification at 2 is now allowed.
- The extra contribution of 1 for $N \equiv 1 \pmod 8$, $e = 1$ comes from Eisenstein ideals above 2 in the Hecke algebra.

Additional multiplicity, explained and unexplained

The previous discussion does not explain the factors of 2 appearing in the $e = 1$ multiplicities. These arise from an observation of Edixhoven: there is a "degeneracy map"

$$S_1(\Gamma_0(N), \overline{\mathbb{F}}_2)^{\oplus 2}_{\mathrm{Katz}} \to S_2(\Gamma_0(N), \overline{\mathbb{F}}_2)_{\mathrm{Katz}}$$

which ensures that each representation which is unramified at 2 contributes at least 2. This completes the explanation of the table.

However, experimentally it seems that additional multiplicities appear. For example:

- for $e = 1$, *all* of the class group terms should carry a factor of 2;
- for $N \equiv 5 \pmod 8$, the $e = 0$ terms should also carry a factor of 2;
- there should be additional contributions from even parts of class groups (possibly explained by exhibiting suitable Galois deformations);
- there are failures of strong multiplicity 1 mod 2 (Kilford, Wiese).

Additional multiplicity, explained and unexplained

The previous discussion does not explain the factors of 2 appearing in the $e = 1$ multiplicities. These arise from an observation of Edixhoven: there is a "degeneracy map"

$$S_1(\Gamma_0(N), \overline{\mathbb{F}}_2)^{\oplus 2}_{\mathrm{Katz}} \to S_2(\Gamma_0(N), \overline{\mathbb{F}}_2)_{\mathrm{Katz}}$$

which ensures that each representation which is unramified at 2 contributes at least 2. This completes the explanation of the table.

However, experimentally it seems that additional multiplicities appear. For example:

- for $e = 1$, *all* of the class group terms should carry a factor of 2;
- for $N \equiv 5 \pmod 8$, the $e = 0$ terms should also carry a factor of 2;
- there should be additional contributions from even parts of class groups (possibly explained by exhibiting suitable Galois deformations);
- there are failures of strong multiplicity 1 mod 2 (Kilford, Wiese).

Additional multiplicity, explained and unexplained

The previous discussion does not explain the factors of 2 appearing in the $e = 1$ multiplicities. These arise from an observation of Edixhoven: there is a "degeneracy map"

$$S_1(\Gamma_0(N), \overline{\mathbb{F}}_2)^{\oplus 2}_{\mathrm{Katz}} \to S_2(\Gamma_0(N), \overline{\mathbb{F}}_2)_{\mathrm{Katz}}$$

which ensures that each representation which is unramified at 2 contributes at least 2. This completes the explanation of the table.

However, experimentally it seems that additional multiplicities appear. For example:

- for $e = 1$, *all* of the class group terms should carry a factor of 2;
- for $N \equiv 5 \pmod 8$, the $e = 0$ terms should also carry a factor of 2;
- there should be additional contributions from even parts of class groups (possibly explained by exhibiting suitable Galois deformations);
- there are failures of strong multiplicity 1 mod 2 (Kilford, Wiese).

A basic example

For $N = 89$, all 7 of the eigenvalues of T_2 on $S_2(\Gamma_0(N), \mathbb{F}_2)$ equal 1. As per LMFDB, this includes one rational Eisenstein-at-2 newform (89.2.1.b), plus two others which are congruent to each other mod 2, one rational (89.2.1.a) and one not (89.2.1.c).

We thus have a unique dihedral representation contributing 6 to the multiplicity of $e = 1$. Is there a generic reason for this?

A basic example

For $N = 89$, all 7 of the eigenvalues of T_2 on $S_2(\Gamma_0(N), \mathbb{F}_2)$ equal 1. As per LMFDB, this includes one rational Eisenstein-at-2 newform (89.2.1.b), plus two others which are congruent to each other mod 2, one rational (89.2.1.a) and one not (89.2.1.c).

We thus have a unique dihedral representation contributing 6 to the multiplicity of $e = 1$. Is there a generic reason for this?

Eisenstein ideals revisited

There is a further source of additional multiplicity for N composite: for $e = 1$, there is *always* an Eisenstein contribution no matter how N reduces mod 8 (Takagi, Yoo).

This means that as it stands, for N composite, this precomputation is of some use for $e = 0$ but useless for $e = 1$. However, the work of Yoo gives a detailed description of Eisenstein ideals (at least for N squarefree). Perhaps this can be used to make 2-adic computations of forms which are Eisenstein mod 2?

Eisenstein ideals revisited

There is a further source of additional multiplicity for N composite: for $e = 1$, there is *always* an Eisenstein contribution no matter how N reduces mod 8 (Takagi, Yoo).

This means that as it stands, for N composite, this precomputation is of some use for $e = 0$ but useless for $e = 1$. However, the work of Yoo gives a detailed description of Eisenstein ideals (at least for N squarefree). Perhaps this can be used to make 2-adic computations of forms which are Eisenstein mod 2?

Contents

An alternative to modular symbols

In his 2016 Dartmouth PhD thesis (under John Voight, with additional contributions from Gonzalo Tornaría), Jeffery Hein develops a construction of Birch into an algorithm for computing Hecke operators on $S_k(\Gamma_0(N), \mathbb{Q})$ for $k \geq 2$ and N squarefree[8] using an analogue of the "method of graphs" replacing isogenies of supersingular elliptic curves with p-neighbors of ternary quadratic forms.

In this approach, one gets direct access to spaces of newforms of specified Atkin-Lehner involution type; this is highly advantageous for calculations in large composite (but squarefree) level. Moreover, the matrices that are obtained are automatically defined over $\mathbb{Z}$, so one may work directly mod 2 without having to change basis (unlike in the current Sage or Magma packages).

[8] This condition has since been relaxed to require only that N is not a perfect square.

An alternative to modular symbols

In his 2016 Dartmouth PhD thesis (under John Voight, with additional contributions from Gonzalo Tornaría), Jeffery Hein develops a construction of Birch into an algorithm for computing Hecke operators on $S_k(\Gamma_0(N), \mathbb{Q})$ for $k \geq 2$ and N squarefree[8] using an analogue of the "method of graphs" replacing isogenies of supersingular elliptic curves with p-neighbors of ternary quadratic forms.

In this approach, one gets direct access to spaces of newforms of specified Atkin-Lehner involution type; this is highly advantageous for calculations in large composite (but squarefree) level. Moreover, the matrices that are obtained are automatically defined over $\mathbb{Z}$, so one may work directly mod 2 without having to change basis (unlike in the current Sage or Magma packages).

[8] This condition has since been relaxed to require only that N is not a perfect square.

Higher weights

As David Roberts described in his talk, for weights above 2 one expects rational newforms to occur rather infrequently. The methods we have described could in principle be used to investigate this further.

One catch is that matrices of higher weight Hecke operators computed using modular symbols, as in Magma and Sage, tend to have nontrivial denominators. The method of Birch-Hein-Tornaría–Voight does not suffer from this defect.

Higher weights

As David Roberts described in his talk, for weights above 2 one expects rational newforms to occur rather infrequently. The methods we have described could in principle be used to investigate this further.

One catch is that matrices of higher weight Hecke operators computed using modular symbols, as in Magma and Sage, tend to have nontrivial denominators. The method of Birch-Hein-Tornaría–Voight does not suffer from this defect.