Modular forms modulo 2

Paul Dubois

March 3, 2020

Abstract

We are interested in Modular forms modulo 2, and computing thing about it. [temporary abstract]

Key words that should appear: Modular forms; Mod 2; Duality of definitions; Governing fields; Frobenian map?; Exact computations;

Contents

1 Modular forms

1.1 Modular forms of level 1

Let $\mathbb{H}$ denote the *upper-half plane*, that is, $\mathbb{H} = \{z = x + yi \in \mathbb{C} |\ y > 0\}$.

We say that a function $f : \mathbb{H} \to \mathbb{C}$ is *weakly modular* of *weight* $2k$ if f is meromorphic and

$$f(z) = (cz+d)^{-2k} f\left(\frac{az+b}{cz+d}\right) \qquad \text{for all } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z}).$$

The group $\mathrm{SL}_2(\mathbb{Z})$ of invertible $(2x2)$ matrices over $\mathbb{Z}$ with is generated by

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix};$$

see [Conrad, 2020, p.1-2].

From this property, we can derive an alternative definition of weakly modular functions: f is weakly modular of weight $2k$ if f is meromorphic and

$$f(z+1) = f(z) \quad \text{and} \quad f(-1/z) = z^k f(z)$$

for all $z \in \mathbb{C}$.

Moreover, we define a function $f : \mathbb{H} \to \mathbb{C}$ to be *modular* of weight $2k$ if f is holomorphic and weakly modular. Lastly, we say that a function $f : \mathbb{H} \to \mathbb{C}$ is a *modular form* of weight $2k$ if it modular and holomorphic at ∞, that is, $f(1/z)$ is holomorphic at $z = 0$.

It is straightforward to check, using the above definition, that the set of modular forms of weight $2k$ is closed under addition and multiplication by complex scalars. More precisely:

- If f_1 and f_2 are modular forms of weight $2k$, then $f_1 + f_2 : z \to f_1(z) + f_2(z)$ is modular of weight $2k$ as well.

- Similarly, if $\lambda \in \mathbb{C}$ and f is a modular form of weight $2k$, then so is $\lambda \cdot f : z \to \lambda f(z)$.

Therefore, modular forms of weight $2k$ over $\mathbb{C}$ form a space. We denote it M_k.

It is also possible to multiply modular forms, in which case the weights are additive: If f_1 and f_2 are modular forms of respective weights $2k_1$ and $2k_2$, then $f_1 f_2 : a \to f_1(z) f_2(z)$ is modular of weight $2k_1 + 2k_2$.

We deduce that we can take powers of modular forms, and the weight is then multiplied by the exponent: if $f(z)$ is modular of weight $2k$, then $f^n(z)$ is modular of weight $2k \cdot n$ (with $n \in \mathbb{N}$ [1]).

1.2 Typical Modular Forms

1.2.1 Eisenstein series G_k

The most famous class of modular forms is probably the *Eisenstein series*, usually denoted G_k. We define them as follows [Stein, 2007, Examples of Modular Forms]:

$$G_k(z) = \sum_{(m,n) \in \mathbb{Z}^2 \setminus \{(0,0)\}} \frac{1}{(mz+n)^{2k}}$$

for $k \geq 2$.

[1] The set of naturals $\mathbb{N}$ is taken to start from 0 in this paper.

It is easy to check that G_k are modular of weight $2k$ [Stein, 2007, Proposition 2.1], as:

$$G_k(z+1) = G_k(z)$$

(using $(m, n+m) \to (m, n)$, an invertible map)

$$G_k(-1/z) = z^k G_k(z)$$

(using $(m, -n) \to (m, n)$, an invertible map).

It is pleasant to remark that [Stein, 2007, Proposition 2.2]

$$G_k(\infty) = \sum_{n \in \mathbb{Z}^*} \frac{1}{n^{2k}} = 2\zeta(2k)$$

. Where $\zeta(k)$ is Riemann zeta function. The values of this function are well-known on positive even numbers, and we deduce [Lennart Rade, 2013, p.194] that:

$$G_k(\infty) = 2\zeta(2k) = \frac{(2\pi)^{2k}}{(2k)!} B_k$$

where $B_k = (-1)^{k+1} b_{2k}$ and b_k are Bernoulli numbers.

1.2.2 The Modular Discriminant Δ

We will be interested in one main modular form in the rest of this article: the *modular discriminant* Δ. We define Δ in terms of G_k as follows [Serre, 1973, p.84]:

$$\Delta = \left(\frac{1}{(2\pi)^{12}}\right)(g_2^3 - 27g_3^2) \in M_6^0 \qquad \text{with } g_2 = 40G_2 \text{ and } g_3 = 140G_3$$

As g_2^3 is modular of weight $4 \cdot 3 = 12$ and g_3^2 of weight $6 \cdot 2 = 12$, Δ is modular of weight 12. Multiplying by the scalar $(1/(2\pi)^{12})$ doesn't change the weight of the modular form, and it will we useful later for normalization purposes.

Now, using $G_2(\infty) = 2\zeta(4) = \frac{\pi^4}{45}$ and $G_3(\infty) = 2\zeta(6) = \frac{2\pi^4}{945}$, we get

$$\Delta(\infty) = \left(\frac{1}{(2\pi)^{12}}\right)\left[\left(\frac{4\pi^4}{3}\right)^3 - \left(\frac{8\pi^4}{27}\right)^2\right] = 0$$

so Δ has a zero at infinity.

1.3 Cusp Forms

A function $f : \mathbb{H} \to \mathbb{C}$ that is a modular form may in addition be a *cusp form*, if $f(\infty) = 0$. We will denote the *space of modular cusp forms* of weight $2k$ over $\mathbb{C}$ by M_k^0.

It is useful to note $G_k(\infty) = \sum_{n \in \mathbb{N}^*} \frac{2}{n^{2k}} > 2$ and in particular, $G_k(\infty) \neq 0$, so G_k are *not* cusp forms for any k. As we have shown it before, $\Delta(\infty) = 0$, so Δ is a modular cusp form of weight 12, so $\Delta \in M_6^0$. Using tools from complex analysis, we can prove that Δ has only one zero (at infinity), which has order one [Serre, 1973, p.88].

We have the following relation:

Theorem 1. $M_k \cong M_k^0 \oplus \mathbb{C} \cdot G_k$ *for all* $k \geq 2$ *[Serre, 1973, p.88]*

Proof. We let $\Phi : M_k \to \mathbb{C}$ such that if $f \in M_k$, $\Phi(f) = f(\infty)$.

Now, we have $\text{Ker}(\Phi) = M_k^0$, therefore, by the 1st Isomorphism Theorem, $M_k/M_k^0 \cong \text{Im}(\Phi) \subseteq \mathbb{C}$.

Note that $G_k \in M_k$, and $G_k(\infty) = \sum_{n\in\mathbb{Z}^*} \frac{1}{n^{2k}} \neq 0$, so $G_k \notin M_k^0$. As $G_k \neq 0$, $\dim(M_k/M_k^0) \geq 1$ and $\text{Im}(\Phi) = \mathbb{C}$. Thus, $G_k \in M_k$

M_k^0

Finally, we have $M_k \cong M_k^0 \oplus \mathbb{C}.G_k$ if $k \geq 2$. (The above argument fails for $k < 2$ as G_k is not well defined any more.) □

Therefore, the dimensions of M_k and M_k^0 are closely linked.

1.4 Dimensions of Spaces of Modular Forms

The fact that multiplying two modular forms gives a function that remains modular yields that we may map a set of modular forms to an other.

Theorem 2. *$M_{k-6} \cong M_k^0$. [Serre, 1973, p.88]*

Proof. We let $\Phi : M_{k-6} \to M_k^0$ such that if $f \in M_k$, $\Phi(f)(z) = \Delta(z)f(z)$.

This is well defined as if f has weight $2(k-6)$, $\Delta.f$ has weight $2k$ since Δ has weight 12. As Δ is a cusp from, $\Delta.f$ will also be a cusp form.

From definition, Φ is clearly homomorphic.

Now, if $g \in M_k^0$, we may define $\Psi : M_k^0 \to M_{k-6}$ such that $\Psi(g)(z) = g(z)/\Delta(z)$

This is well defined as if g has weight $2k$, $\Delta.f$ has weight $2k$ since Δ has weight 12. As Δ is a cusp from, $\Delta.f$ will also be a cusp form.

This is well defined as Δ has only one zero, at infinity, where g also has a zero (as g is a cusp form). The weights agree again as well.

It is then easy to remark that $\Psi = \Phi^{-1}$. So Φ is bijective, and thus isomorphic.

Finally, we have $M_{k-6} \cong M_k^0$. □

This theorem, combined with the previous one is very powerful: it shows that there must be a pattern (of 6) in the sequence of dimensions $\dim(M_k)$ and $\dim(M_k^0)$ for $k \geq 2$. We have $M_k \cong M_k^0 \oplus \mathbb{C}.G_k \cong M_{k-6} \oplus \mathbb{C}.G_k$, so $\dim(M_k) = \dim(M_{k-6}) + 1$ when $k \geq 2$. Thus, if we compute the dimensions of M_0, M_1, M_2, M_3, M_4, M_5, we can extrapolate dimensions of M_k and M_k^0 for all k.

Using complex analysis techniques again, we have:

- $\dim(M_k) = 0 \quad k < 0$
- $\dim(M_1) = 0$
- $\dim(M_0) = \dim(M_2) = \dim(M_3) = \dim(M_4) = \dim(M_5) = 1$

In the case $k = 0$, $\dim(M_0) = 1$. As $f(z) = 1$ is clearly a modular from of weight 0, $\{1\}$ is a basis for M_0. We deduce $\dim(M_k^0) = 0$ as 1 is clearly not a cusp form. In the case $k = 1$, $\dim(M_1) = 0$, which makes $\dim(M_1^0) = 0$ automatically. (Cases $k < 0$ are similar to $k = 1$.)

Other cases may be derived directly from the relations (using induction to get general formulas), and we obtain:

Space	$k < 0$	$k \geq 0,\ k \equiv 1 \bmod 6$	$k \geq 0,\ k \not\equiv 1 \bmod 6$
$\dim(M_k)$	0	$\lfloor k/6 \rfloor$	$\lfloor k/6 \rfloor + 1$
$\dim(M_k^0)$	0	$\max\{0, \lfloor k/6 \rfloor - 1\}$	$\lfloor k/6 \rfloor$

Note that the max is taken only to avoid negative dimensions.

1.5 Fourier Expansion

1.5.1 Definition

To study such function, we use Fourier Expansion. In the case of f being a modular form of weight $2k$, a *Fourier Expansion* is a representation of f as a power series of $e^{2\pi inz}$ i.e.

$$f(z) = \sum_{n\in\mathbb{Z}} a_n(f)e^{2\pi inz}.$$

We usually denote $q = e^{2\pi iz}$ so that $q^n = e^{2\pi inz}$ and the Fourier expansion of f become

$$f(q) = \sum_{n\in\mathbb{Z}} a_n(f)q^n.$$

When in this form, we may as well call it the *q-expansion*.

1.5.2 Typical Modular Forms Fourier Expansion

Fourier Expansions of G_k The modular forms G_k have the following q-expansion [Serre, 1973, p.92]:

$$G_k(q) = 2\zeta(2k) + 2\frac{(2\pi i)^{2k}}{(2k-1)!}\sum_{n=1}^{\infty}\sigma_{2k-1}(n)q^n$$

where σ_d is the generalized divisor function such that:

$$\sigma_d(n) = \sum_{m|n} m^d.$$

Fourier Expansion of Δ We also have [Serre, 1973, p.95]:

$$\Delta(q) = q\prod_{n=1}^{\infty}(1-q^n)^{24}$$

1.6 A Basis for Modular Forms

The set of modular forms that are weight $2k$ in fact form a vector space (we can add modular forms together, and multiply them with a constant) over the complex numbers. One may ask then a basis for this vector space.

We would like to find a basis for each set M_k. It turns out that the modular forms G_2 and G_3 introduced before in fact generate a basis for all M_k. It is not obvious and may in fact seems wrong at a first stage: G_2 and G_3 are modular forms of weight 4 and 6, whereas M_k in general have modular forms of weight $2k$. However, by taking combinations of G_2 and G_3, we may obtain modular forms of any weight $2k$. It is important to remember that when multiplied, the weight of modular forms add up.

Theorem 3. *The set $S = \{G_2^a G_3^b | a, b \in \mathbb{N}, 2a+3b = k\}$ is a basis for M_k. [Stein, 2007, Theorem 2.17]*

Proof. Of course, the cases when $\dim(M_k) = 0$ (for $k < 0$ and $k = 1$) are trivial, as the basis is empty, and $2a + 3b = k$ has no solution for $a, b \in \mathbb{N}$.

To show S is a basis, we need it to span M_k and to be linearly independent.

We start with spanning, and we proceed by induction on k, with step 6.

As $\dim(M_k) = 1$ for $k = 0, 2, 3, 4, 5, 7$, and the equation $2a + 3b = k$ has exactly one solution for $a, b \in \mathbb{N}$ (namely $(a, b) = (0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (2, 1)$), S has only one element, which must be the basis.

Now, for $k > 7$, take some $a, b \in \mathbb{N}$ such that $2a + 3b = k$. Let $f \in M_k$, and $g = G_2^a G_3^b \in M_k$. $g(\infty) \neq 0$ as none of G_2 or G_3 is a cusp form. So there must be a complex λ such that $f - \lambda g$ is a cusp form. Then $f - \lambda g \in M_k \cong M_{k-6}^0$ and we can find a $h \in M_{k-6}^*$ such that $h.\Delta = f - \lambda g$.

By induction, h must be a polynomial of G_2 and G_3; by definition, Δ is one as well (note that yet, we don't put any restriction on powers of G_2 and G_3, other then being positive integers). Therefore, $f = \Delta.h + \lambda g$ is a polynomial of G_2 and G_3. From the fact that $f \in M_k$ (i.e. f has weight $2k$), terms of f as a polynomial of G_2 and G_3 have the from $G_2^a G_3^b$ with $2a + 3b = k$.

We now want to show linear independence, we proceed by contradiction.

Suppose there is a non-trivial linear relation of terms $G_2^a G_3^b$. We can multiply it by suitable G_2 and G_3 so that all terms have the form $2a+3b = k \equiv 0 \mod 12$. Then, we can divide all terms by G_3^2, witch gives us that there is a polynomial for which ${G_2^3}/{G_3^2}$ is a root. In particular, this polynomial is constant when ${G_2^3}/{G_3^2}$ is plugged. This contradicts the fact that q-expansion of ${G_2^3}/{G_3^2}$ is not constant. □

This set of makes to be a basis, and one may even find it pleasant: given the two modular forms G_2 and G_3, this set generates all the modular forms of weight $2k$ that we could think of, if we only knew these two modular forms.

1.7 Hecke Operators

We define the *Hecke operators* for a modular form f as follows [Serre, 1973, p.100]:

$$T_n f(z) = n^{2k-1} \sum_{a \geq 1,\, ad=n,\, 0 \leq b < d} d^{-2k} f\left(\frac{az+b}{d}\right)$$

with $n \in \mathbb{N}$.

We can check that $T_n f$ is modular if f is (as the sum of modular forms).

We may as well write $T_n f$ as a Fourier Expansion of $q = e^{2\pi i z}$ as follows [Serre, 1973, p.100]:

$$T_n f(z) = \sum_{m \in \mathbb{Z}} \gamma(m) q^m \quad \text{with} \quad \gamma(z) = \sum_{a|(n,m),\, a \geq 1} a^{2k-1} c\left(\frac{mn}{a^2}\right)$$

$$\text{For modular forms } f \text{ s.t. } f(z) = \sum_{n \in \mathbb{Z}} \alpha(n) q^n$$

2 Modular Forms Modulo Two

2.1 Strategy to Reduce Modulo Two

It is not trivial, at this point, why and how we can reduce modulo 2 modular forms, objects that have coefficients in $\mathbb{C}$. In general, reduction modulo a number is only possible with whole numbers (integers). We would like to reduce modulo 2 coefficients of the Fourier series for modular forms. But at the moment, they lie in $\mathbb{C}$.

In fact, we will introduce a new basis for the modular forms: the so called Miller Basis. The coefficients of all the forms in this basis are integers. It is then possible to consider the space of modular forms over $\mathbb{Z}$ instead of $\mathbb{C}$. Once this is done, we will reduce all the newly integral coefficients modulo 2.

In this section, we will denote all objects reduced modulo 2 with an $\overline{\text{over-line}}$:

- The modular form f once reduced will be denoted $\overline{f}$.
- The coefficients of the q-expansion c will reduce to $\bar{c}$
- The Hecke operators T_n reduced will be denoted $\overline{T_n}$.

2.2 Integral Basis

2.2.1 Normalisation of Typical Modular Forms

Normalisation of Eisenstein series G_k We first recall the formula for q extension of G_k and the one for $\zeta(2k)$:

$$G_k(q) = 2\zeta(2k) + 2\frac{(2\pi i)^{2k}}{(2k-1)!}\sum_{n=1}^{\infty}\sigma_{2k-1}(n)q^n$$

and

$$2\zeta(2k) = \frac{(2\pi)^{2k}}{(2k)!}B_k$$

so overall:

$$G_k(q) = \frac{(2\pi)^{2k}}{(2k)!}B_k + 2\frac{(2\pi i)^{2k}}{(2k-1)!}\sum_{n=1}^{\infty}\sigma_{2k-1}(n)q^n$$

We would like to normalize this series, so that the coefficients become integers, so that we can ultimately reduce them modulo 2. Right now, coefficients are rational.

As we want to keep the series modular with same weight, the only tool we have to normalize the series is multiplication by a constant. The normalization is a crucial point: If we multiply by 2 all coefficients of a modular form that already lie in $\mathbb{Z}$, the reduction mod2 will always give zero.

First, let's normalize the series to have particular values on some coefficients of interest. There are two justified ways to do so: normalize to have constant coefficient set to one, and to have q coefficient is set to one. We will introduce both: Let E_k be such that:

$$E_k.2\zeta(2k) = G_k$$

so that

$$E_k = 1 + (-1)^k\frac{4k}{B_k}\sum_{n=1}^{\infty}\sigma_{2k-1}(n)q^n.$$

E_k then has constant coefficient set to one.

Let F_k be such that:

$$F_k.\left(2\frac{(2\pi i)^{2k}}{(2k-1)!}\right) = G_k$$

so that

$$F_k = (-1)^k \frac{B_k}{4k} + \sum_{n=1}^{\infty} \sigma_{2k-1}(n) q^n.$$

F_k then has q-coefficient set to one (as $\sigma_{2k-1}(1) = 1$).

Clearly, the coefficients of this expansion remain in $\mathbb{Q}$ at least, and we will show that for some specific k, the coefficients lie in fact in $\mathbb{Z}$. Both F_k and E_k are interesting, but for our purpose (reducing modulo 2), we will use E_k. Note that E_k are normalized versions of Eisenstein series G_k, but in literature, both are called Eisenstein series see [Shrivastava, 2017, p.6] for example.

The Modular Discriminant Δ Normalized Again, we recall the formula for q extension of Δ:

$$\Delta(q) = q \prod_{n=1}^{\infty} (1-q^n)^{24}$$

Clearly, the coefficients in expansion of Δ are integers (which we can reduce modulo 2). This is the reason why we defined Δ with the $\frac{1}{(2\pi)^{12}}$ factor in front.

2.2.2 Miller Basis

Basis with Integral Coefficients (in Fourier Series) Applying normalization $G_k \to E_k$ above for $k = 2, 3$, we get:

$$\begin{aligned} E_2 &= 1 + \frac{8}{B_2} \sum_{n=1}^{\infty} \sigma_3(n) q^n \qquad B_2 = \frac{1}{30} \\ &= 1 + 240 \sum_{n=1}^{\infty} \sigma_3(n) q^n \end{aligned}$$

and

$$\begin{aligned} E_3 &= 1 - \frac{12}{B_3} \sum_{n=1}^{\infty} \sigma_5(n) q^n \qquad B_3 = \frac{1}{42} \\ &= 1 - 504 \sum_{n=1}^{\infty} \sigma_5(n) q^n \end{aligned}$$

Now, we have shown that $\{G_2^a G_3^b | 2a + 3b = k\}$ is a basis for modular forms of weight $2k$ over the complex (see 1.6). As $E_2 = \lambda G_2,\ \lambda \in \mathbb{C}$ and $E_3 = \mu G_3,\ \mu \in \mathbb{C}$, we have that $\{E_2^a E_3^b | 2a + 3b = k\}$ remains a basis for M_k over $\mathbb{C}$.

It is clear, from the series, that coefficients of the q-expansion of both E_2 and E_3 are all integers. Thus, so are coefficients of combinations of E_2 and E_3. Therefore, we have found a basis for M_k such that all elements in the basis have only integral coefficients in their q-expansion.

Miller Basis for M_k^0 This is a nice result, but we can in fact do better, by forcing the first coefficients to chosen values.

Theorem 4. *For the space of modular cusp forms M_k^0, there exists a basis $\{f_1, \cdots, f_r\}$ such that:*

- $f_i \in Z[q]$

- $a_i^j = \delta_i j = \begin{cases} 1 & \textit{if } i = j \\ 0 & \textit{if } i \neq j \end{cases} \quad \forall 1 \leq i, j \leq r$
  where a_i^j *is the coefficient of* q^j *in expansion of* f_i.

This is commonly called the Miller basis for M_k^0, as it was first introduced by Victor Saul Miller Miller [1975].

Proof. • For $k < 6$, $k = 7$, we have $\dim(M_k^0) = 0$. Thus, $\emptyset$ is a basis which satisfies the Miller basis properties.

- For $k = 6$, we have $\dim(M_k^0) = 1$. Thus, $\{\Delta\}$ is a basis which satisfies the Miller basis properties.

- For $k \geq 7$, we let $r = \dim(M_k^0) \geq 1$. We then consider the set

$$\{g_j | 1 \leq j \leq r\}$$

where

$$g_j = \Delta^j E_3^{2(d-j)+b} E_2^a$$

with

$$2a + 3b \leq 7 \ \& \ 2a + 3b \cong k \bmod 6$$
$$\& \ \ d = \frac{k - (2a + 3b)}{6} \quad \in \mathbb{N} \text{ as } k \geq 7$$

Note that a and b are unique unless $k \cong 0 \bmod 6$. In witch case, we use by convention $a = 0$, $b = 0$.

As all E_2, E_3, and Δ have integral coefficients, g_j will as well.

We then look at the q series:

$$\Delta(q) = q + O(k^2) \implies \Delta^j(q) = q^j + O(k^{j+1})$$

As we normalized so,

$$E_2(q) = 1 + O(q) \implies E_2^{\alpha}(q) = 1 + O(q)$$
$$E_3(q) = 1 + O(q) \implies E_3^{\alpha}(q) = 1 + O(q)$$

This gives:

$$g_j(q) = q^j + O(q^{j+1}) \quad \forall 1 \leq j \leq r.$$

Therefore, $\{g_j, | 1 \leq j \leq r\}$ is clearly a linearly independent set. By dimension argument, it also spans M_k^0. Therefore, it forms a basis. Moreover, in this basis: $a_i^j = \delta_{ij} \quad i \leq j$.

Finally, we can use Gaussian elimination on $\{g_j\}$ to obtain a basis $\{f_j | 1 \leq j \leq r\}$ such that: $a_i^j = \delta_{ij} \quad \forall 1 \leq i, j \leq r$. The coefficients will remain in $\mathbb{Z}$ after Gaussian elimination.

□

Extension to all M_k We already have a basis for M_k^0, as $\dim(M_k) = \dim(M_k^0) + 1$ (over $\mathbb{C}$), we just need to adjoint one element of $M_k \setminus M_k^0$ to our basis.

It was shown before that $\{E_2^a E_3^b | 2a + 3b = k\}$ is a basis for M_k with integral coefficients (see 2.2.2) One may see from the q-expansion that $E_2^a E_3^b = 1 + O(q)$ so $E_2^a E_3^b \in M_k \setminus M_k^0$.

Therefore, we can just add one element of $\{E_2^a E_3^b | 2a + 3b = k\}$ to the Miller basis, and use Gaussian elimination again. We get a basis foe M_k of the form $\{f_j | 0 \leq j \leq r\}$ such that in this basis: $a_i^j = \delta_{ij} \quad \forall 0 \leq i, j \leq r$ (with $r = dim(M_k^0)$ i.e. $r + 1 = dim(M_k)$).

Miller Basis Examples

Miller basis for $k = 16$ We can calculate the Miller basis for $k = 16$: $k \cong 4 \bmod 12$ so $a = 2$ and $b = 0$; $d = 2$. We put $g_1 = \Delta^1 E_3^2 E_2^2$, so:

$$\begin{aligned} g_1(q) &= \Delta(q) E_2^2(q) E_3^2(q) \\ &= \left[q - 24q^2 + 252q^3 + O(q^4)\right] \\ &\quad \cdot \left[1 + 240q + 2160q^2 + 6720q^3 + O(q^4)\right]^2 \\ &\quad\quad \cdot \left[1 - 504q - 16632q^2 + 122976q^3 + O(q^4)\right]^2 \\ &= q - 552q^2 - 188244q^3 + O(q^4) \end{aligned}$$

and $g_2 = \Delta^2 E_3^0 E_2^2$, so:

$$\begin{aligned} g_2(q) &= \Delta^2(q) E_2^2(q) \\ &= \left[q - 24q^2 + 252q^3 + O(q^4)\right]^2 \\ &\quad \cdot \left[1 + 240q + 2160q^2 + 6720q^3 + O(q^4)\right]^2 \\ &= q^2 + 432q^3 + O(q^4) \end{aligned}$$

Then, $f_2 = g_2$ and $f_1 = g_1 + 552g_2$, so:

$$\begin{aligned} f_1(q) &= q - 552q^2 - 188244q^3 + O(q^4) \; + \; 552 \cdot \left[q^2 + 432q^3 + O(q^4)\right] \\ &= q + 50220q^3 + O(q^4) \\ f_2(q) &= q^2 + 432q^3 + O(q^4) \end{aligned}$$

Therefore, up to $O(q^4)$, $\{f_1, f_2\} = \{q + 50220q^3 + O(q^4), q^2 + 432q^3 + O(q^4)\}$ is a basis for M_{16}^0.

To extend this base to M_k, we adjoint a term of the form $g_0 = E_2^a E_3^b$ where $2a + 3b = 16$. We pick $g_0 = E_2^8$, so:

$$\begin{aligned} g_0(q) &= E_2^8(q) \\ &= \left[1 + 240q + 2160q^2 + 6720q^3 + O(q^4)\right]^8 \\ &= 1 + 1920q + 1630080q^2 + 803228160q^3 + O(q^4) \end{aligned}$$

Then, $f_0 = g_0 - 1920g_1 - 1630080g_2$, so:

$$\begin{aligned} f_0(q) &= g_0(q) - 1920g_1(q) - 1630080g_2(q) \\ &= \left[1 + 1920q + 1630080q^2 + 803228160q^3 + O(q^4)\right] \\ &\quad - 1920\left[q + 50220q^3 + O(q^4)\right] \\ &\quad\quad - 1630080\left[q^2 + 432q^3 + O(q^4)\right] \\ &= 1 + 2611200q^3 + O(q^4) \end{aligned}$$

Therefore, up to $O(q^4)$, $\{f_0, f_1, f_2\} = \{1+2611200q^3+O(q^4), q+50220q^3+O(q^4), q^2+432q^3+O(q^4)\}$ is a basis for M_{16}.

Miller basis for $k = 92$ The calculation of this basis may be interesting by hand once; However, it is possible to automate it. The procedure that calculates such coefficients is a standard in SageMath Contributors [2020]. Here is, up to $O(q^{10})$, the Miller basis for M_{92}:

$$\begin{aligned}
f_0 &= 1 + 3034192667130000q^8 + 137290127714549760000q^9 + O(q^{10})\\
f_1 &= q + 91578443563200q^8 + 2651503140376278561q^9 + O(q^{10})\\
f_2 &= q^2 + 2380310529376q^8 + 42238207588515840q^9 + O(q^{10})\\
f_3 &= q^3 + 51682260816q^8 + 530253459731160q^9 + O(q^{10})\\
f_4 &= q^4 + 896013480q^8 + 4882999541760q^9 + O(q^{10})\\
f_5 &= q^5 + 11516000q^8 + 28971735750q^9 + O(q^{10})\\
f_6 &= q^6 + 94680q^8 + 80990208q^9 + O(q^{10})\\
f_7 &= q^7 + 312q^8 - 4860q^9 + O(q^{10})
\end{aligned}$$

2.3 Basis Modulo Two

2.3.1 Reduced Modular Forms

Now that we have a basis with integral coefficients, it makes sense to reduce forms modulo 2. For a modular form f, we denote its reduced modulo 2 from $\overline{f}$. It is defined as follows:

If

$$f(q) = \sum_{n\in\mathbb{N}} c(n)q^n$$

then

$$\overline{f}(q) = \sum_{n\in\mathbb{N}} \overline{c}(n)q^n \qquad \text{with } \ \overline{c}(n) = c(n) \bmod 2.$$

We want to reduce Miller basis modulo 2. The reason is that as we know that some coefficients are ones, the reduction will not be trivial. We will reduce separately E_2, E_3 and Δ (witch together generate the Miller basis).

E_2 reduced We have:

$$E_2(q) = 1 + 240\sum_{n=1}^{\infty} \sigma_3(n)q^n \equiv 1 \bmod 2$$

Therefore, the reduction modulo 2 of E_2 is just 1. We write $\overline{E_2} = 1$, so $\overline{E_2^a} = 1$, for all $a \geq 0$.

E_3 reduced We have:

$$E_3(q) = 1 - 504\sum_{n=1}^{\infty} \sigma_5(n)q^n \equiv 1 \bmod 2$$

Therefore, the reduction modulo 2 of E_3 is 1 as well. We write $\overline{E_3} = 1$, so $\overline{E_3^b} = 1$, for all $b \geq 0$.

Δ reduced We defined before Δ, and we would now like to know its q-extension in the standard way. That is, an infinite sum of q^n, instead of an infinite product as we have at the moment.

We define the coefficients $\tau(n)$ to match in the equation:

$$\Delta(q) = q\prod_{n=1}^{\infty}(1-q^n)^{24} = \sum_{n=1}^{\infty}\tau(n)q^n$$

When this holds, τ is called the *Ramanujan function.*

We would like an explicit formula for $\tau(n)$. More precisely, we are interested in a formula for $\tau(n)$ mod 2.

We will calculate separately the coefficients $\tau(n)$ mod 2 for n even and odd.

Case n odd Remember $\sigma_s(n)$ as the sum of s^{th} powers of (positive) divisors of n. It is known from classical theory [Kolberg, 1962, p.8] that:

$$\tau(8n+l) \equiv a_l \sigma_{11}(8n+l) \pmod{2^{b_l}}$$

where $gcd(l,8) = 1$, $a_1 = 1$, $a_3 = 1217$, $a_5 = 1537$, $a_7 = 705$, $b_1 = 11$, $b_3 = 13$, $b_5 = 12$, $b_7 = 14$

We are interested in congruence class (mod 2) of the Ramanujan function $\tau(n)$. For n odd, we deduce the following:

$$\tau(n) \equiv \sigma_{11}(n) \equiv \sum_{d|n} d^{11} \equiv \sum_{d|n} 1 \equiv \begin{cases} 1 \bmod 2 & \text{if } n \text{ is a square} \\ 0 \bmod 2 & \text{else} \end{cases}$$

Case n even It is easy to calculate that $\tau(2) = -24 \equiv 0 \bmod 2$.
Using $\tau(p^{n+1}) = \tau(p^n)\tau(p) - p^{11}\tau(p^{n-1}) \quad p \in \mathbb{P}$ [Serre, 1973, p.97] with $p = 2$, it follows by induction that $\tau(2^k) \equiv 0 \bmod 2 \quad \forall k \in \mathbb{N}$.
Using $\tau(nm) = \tau(n)\tau(m) \quad$ if $gcd(n,m) = 1$ [Serre, 1973, p.97], it follows that for all n even, $\tau(n) \equiv 0 \bmod 2$.

Explicit series of the discriminant Therefore, the only non-zero coefficients (modulo 2) appears on odd squares, i.e.:

$$\tau(n) \equiv \begin{cases} 1 \bmod 2 & \text{if } n = (2m+1)^2 \quad \text{for } m \in \mathbb{N} \\ 0 \bmod 2 & \text{else} \end{cases}$$

Thus, we can write the power series of Δ as:

$$\Delta(q) \equiv \overline{\Delta}(q) = \sum_{m=0}^{\infty} q^{(2m+1)^2} \bmod 2$$

2.3.2 Reduced Basis

The Miller basis for M_k was obtained via the Gauss elimination of the set $\{\Delta^j E_3^{2(d-j)+b} E_2^a | 1 \leq j \leq \dim(M_k)\}$ (with some conditions on a,b,d).

But $\overline{E_2^a} = \overline{E_2}^a = 1^a = 1 \bmod 2$ and similarly, $\overline{E_3^{2(d-j)+b}} = 1 \bmod 2$. So once the above set is reduced modulo 2, we are left with $\{\overline{\Delta}^j | 1 \leq j \leq \dim(M_k)\}$. So the Miller basis just becomes the Gauss elimination of $\overline{\Delta}$ powers.

This is what motivates the next section.

2.4 Space of Modular Forms Modulo Two

We would like to have a definition for this space in a similar way as M_k was used for modular forms (of weight $2k$) before reduction.

By abuse of notation, we denote the reduction of Δ modulo 2 (written $\overline{\Delta}$ until here) also by Δ, that is,

$$\Delta(q) = \sum_{m=0}^{\infty} q^{(2m+1)^2} \bmod 2.$$

2.4.1 Weights of Modular Forms Modulo Two

We just saw that the Miller basis for $\overline{M_k}$ is (the Gaussian elimination of) $\{\Delta^j | 1 \leq j \leq \dim(M_k)\}$.

Now, if we look at this set not reduced modulo 2, we have: $\{\Delta^j | 1 \leq j \leq \dim(M_k)\}$. This is a set of modular forms that have different weights. However, we started with a modular forms in M_k, i.e. all modular forms having weight $2k$.

We understand now that modulo 2, the weight of modular form doesn't make sense any more. This is one of the consequences of reducing modulo 2: we lose some informations about the modular forms, such as the weight.

From this observation, we should study all modular forms together, modulo 2 (instead of separating by weights). This is why the space of modular forms modulo 2 will be denoted $\mathcal{F}$, with no dependence on k.

2.4.2 Powers of the Modular Discriminant Δ

Set of Powers of the Modular Discriminant Δ As we just saw, the Gaussian elimination of powers Δ^k up to $\dim(\overline{M_k})$ form the Miller basis of $\overline{M_k}$ (modular forms of weight $2k$ reduced modulo 2).

For simplicity again, we will just take the powers of Δ to be our basis for modular forms modulo 2 (i.e. drop the Gaussian elimination process).

We define the space $\mathbb{F}_2[\Delta]$ in the usual way:

$$\mathbb{F}_2[\Delta] = \left\{ \sum_{k=1}^{n} a_k \Delta^k | n \in \mathbb{N},\ a_k \in \mathbb{F}_2 \right\}$$

From 2.3.1 we had:

$$\Delta(q) = \sum_{n=0}^{\infty} \tau(n) q^n = \sum_{m=0}^{\infty} q^{(2m+1)^2}$$

Therefore, we define

$$\Delta^k(q) = \sum_{n=0}^{\infty} \tau_k(n) q^n = \left(\sum_{m=0}^{\infty} q^{(2m+1)^2} \right)^k \bmod 2$$

Thus, we have $\tau(n) = \tau_1(n)$.

Proportion of zeros In fact, most of the coefficients $\tau_k(n)$ are 0 modulo 2.

When $k = 1$, there is already few coefficients that are ones: only the odd squares. When raising to the k^{th} power, there are even "less".

Conditions on non-zero coefficients We can find conditions on coefficients that may not be zero.

We observe: We remark that odd squares are all 1 mod 8, and even squares are all 0 mod 8.

$a =$	0	1	2	3	4	5	6	7	mod8
$a^2 =$	0	1	4	1	0	1	4	1	mod8

Table 1: Squares modulo 8

We know from previous calculations that $\Delta(q)$ only has odd powers of q. Thus, raising to the k^{th} power give terms of power n such that:

$$\begin{aligned} n &= m_1^2 + m_2^2 + m_3^2 + \cdots + m_k^2 \\ &\equiv \ 1 \ + \ 1 \ + \ 1 \ + \cdots + \ 1 \bmod 8 \\ &\equiv k \bmod 8 \end{aligned}$$

Therefore: $\tau_k(n) \equiv 1 \mod 2 \implies n \equiv k \mod 8$
Equivalently: $n \not\equiv k \mod 8 \implies \tau_k(n) \equiv 0 \mod 2$ (by taking the contra-positive)

This means, that Δ^k may only have terms q^n such that $n \equiv k \bmod 8$, i.e. Δ^k may only have terms of power congruent to $k \bmod 8$. When $k = 1$, this is that Δ may only have terms of power $1 \bmod 8$, this matches with table 1: all odd squares are $1 \bmod 8$.

Even powers of Δ We compare $\Delta^{2k}(q)$ and $\Delta^k(q^2)$:

$$\begin{aligned} \Delta^{2k}(q) &= \left(\sum_{m=0}^{\infty} q^{(2m+1)^2}\right)^{2k} \\ &= \sum_{n=0}^{\infty} \#[(2m_1+1)^2 + (2m_2+1)^2 + ... + (2m_{2k}+1)^2 = n \mid m_0, m_1, ..., m_{2k} \in \mathbb{N}]\ q^n \\ &= \sum_{n\ even} \#[(2m_1+1)^2 + (2m_2+1)^2 + ... + (2m_k+1)^2 = n/2 \mid m_0, m_1, ..., m_k \in \mathbb{N}]\ q^n \\ &= \left(\sum_{m=0}^{\infty} q^{((2m+1)^2)\cdot 2}\right)^k \\ &= \left(\sum_{m=0}^{\infty} (q^2)^{(2m+1)^2}\right)^k = \Delta^k(q^2) \end{aligned}$$

Thus, $\Delta^{2k}(q) = \Delta^k(q^2)$. Therefore, we can write any modular form modulo 2 f as the following:

$$f = \sum_{s\geq 0} f_s^{2^s} \quad \text{with } f_s \text{ having only odd powers of } \Delta$$

[Nicolas and Serre, 2012a, (3)] So it is sufficient to study only the odd powers of Δ.

2.4.3 The Space $\mathcal{F}$

We define the *space of modular forms modulo 2* denoted $\mathcal{F}$ to be [Nicolas and Serre, 2012a, 2.1]:

$$\mathcal{F} = \left\langle \Delta^k | k \text{ odd} \right\rangle = \langle \Delta, \Delta^3, \Delta^5, \Delta^7, \dots \rangle$$

That is, all finite polynomials of Δ over $\mathbb{F}_2$, having only odd powers. We remark that the weight of modular forms do not appear, as it was discussed before in 2.4.1. The observations modulo 8 that we have done in 2.4.2 yields that it will be useful to denote:

$$\mathcal{F}_1 = \left\langle \Delta^k \mid k = 1 \bmod 8 \right\rangle = \langle \Delta, \Delta^9, \Delta^{17}, \Delta^{25}, \dots \rangle$$

$$\mathcal{F}_3 = \left\langle \Delta^k \mid k = 3 \bmod 8 \right\rangle = \langle \Delta^3, \Delta^{11}, \Delta^{19}, \Delta^{27}, \dots \rangle$$

$$\mathcal{F}_5 = \left\langle \Delta^k \mid k = 5 \bmod 8 \right\rangle = \langle \Delta^5, \Delta^{13}, \Delta^{21}, \Delta^{29}, \dots \rangle$$

$$\mathcal{F}_7 = \left\langle \Delta^k \mid k = 7 \bmod 8 \right\rangle = \langle \Delta^7, \Delta^{15}, \Delta^{23}, \Delta^{31}, \dots \rangle$$

Of course, we have:

$$\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_3 \oplus \mathcal{F}_5 \oplus \mathcal{F}_7$$

We will also introduce (as in [Nicolas and Serre, 2012b, 2.]):

$$\mathcal{F}(n) = \left\langle \Delta^k \mid k \text{ odd and } k \leq 2n-1 \right\rangle = \langle \Delta, \Delta^3, \Delta^5, \dots, \Delta^{2n-1} \rangle$$

This matches specifically $\overline{M_{6(2n-1)}} = \mathcal{F}(n)$.

2.4.4 Duality between Δ and q

As we defined $\mathcal{F}$ above, a modular form modulo 2 is an expression of powers Δ^k. But we had from before that $\Delta = \sum_{m=0}^{\infty} q^{(2m+1)^2} \bmod 2$. Therefore, we can translate a modular form given as a finite polynomial of Δ into an infinite polynomial of q. Thus, there are two ways to write a modular form modulo 2.

This duality between the two definitions is what makes the study of modular forms modulo 2 so interesting: we go back and forth between an infinite series and a finite polynomial. One is easy to express, the other easy to compute. This will lead to new reasoning. In particular, there is a new technique of computation ("exact computations") that uses equivalence between the two ways of writing a modular form.

2.5 Hecke Operators Modulo Two

2.5.1 Reduction Modulo Two

Definition Now that we have reduced modular forms modulo 2, we would like to study the Hecke operators on these reduced modular forms. We define *Hecke operators modulo 2* as follows:

With f a modular form modulo 2 with q-definition

$$f(q) = \sum_{n \in \mathbb{N}} c(n) q^n$$

we define

$$\overline{T_p} | f(q) = \sum_{n \in \mathbb{N}} \gamma(n) q^n$$

where

$$\gamma(n) = \begin{cases} c(np) & \text{if } p \nmid n \\ c(np) + c(n/p) & \text{if } p \mid n \end{cases} \qquad \& \ p \text{ an odd prime}$$

Well-definiteness We want to check that all the definitions make sense. When we look at $T_p|f$, there is a number of ways to to reduce it modulo 2: $\overline{T_p|f}$, $\overline{T_p|\overline{f}}$, $\overline{\overline{T_p}|f}$, $\overline{T_p}|\overline{f}$.

Let's compare coefficients:

$\overline{T_p|f}$:

$$\gamma(n) = \sum_{a|(n,p),\, a \geq 1} a^{2k-1} c\left(\frac{np}{a^2}\right) = \begin{cases} \overline{c}(np) & \text{if } p \nmid n \\ \overline{c}(np) + \overline{c}(n/p) & \text{if } p \mid n \end{cases}$$

Divisors of (n,p) are $\{1\}$ or $\{1,p\}$ since p is prime, so the sum split in two cases, with one or two terms. We see now that looking at Hecke operators modulo 2 only for primes simplifies the sum to a computable formula.

As both 1 and p are odd, the term a^{2k-1} reduces to 1 modulo 2. We understand why Hecke operators modulo 2 isn't defined for even numbers: many terms in the summation would become zero. It would not make sense to call it a Hecke operator any more.

It also makes sense why we look at modular forms modulo 2 and not say three or five: the coefficient a^{2k-1} collapse nicely modulo 2, which won't be the case modulo an other number then 2.

$\overline{T_p|\overline{f}}$: This is (very) similar to the case before.

$\overline{T_p}|\overline{f}$:

$$\gamma(n) = \begin{cases} \bar{c}(np) & \text{if } p \nmid n \\ \bar{c}(np) + \bar{c}(n/p) & \text{if } p \mid n \end{cases}$$

$\overline{\overline{T_n}|f}$: Again, this is (very) similar to the case before.

All reductions give in fact the same result, so it makes sense to reduce modular forms modulo 2, and still study the Hecke operators (but now only for odd primes). As this all makes sense, we will now write only consider modular forms modulo 2, and we will drop the over lines for simplicity. The fact that T_p and $\overline{T_p}$ have exactly the same action on the q-expansions of modular forms is only true when p is an odd prime. This is why we will concentrate on this case.

2.5.2 Basic Properties

When reduced modulo 2, Hecke operators $\overline{T_p}$ for primes p have more properties then the general T_p. The extra properties make the study modulo two interesting.

Inherited properties From the fact that $\overline{T_p}|\overline{f(q)} = \overline{T_p|f(q)}$, we get that the Hecke operators modulo 2 keep the properties they had before being reduced.

Modularity Remains From definition 1.7, a Hecke operators transform a modular form to an other. This is because from definition, $T_n f$ is a sum of modular forms (which remain modular). Therefore, Hecke operators modulo 2 will as well transform a modular form to an other. This was not clear from the definition modulo 2 that we had (which was in terms of q series).

Commutativity As in general [Serre, 1973, p.101]:

$$T_n T_m = T_{mn} \quad \text{if } \gcd(m,n) = 1$$

We get that:

$$\overline{T_p T_q} = \overline{T_q T_p} \quad \forall p, q \in \mathbb{P}$$

Therefore, the Hecke operators modulo 2 commute. This, as well, was not clear form definition. It will be very convenient for future calculations.

Linearity From definition 1.7, we have that the Hecke operators are immediately linear. That is:

$$T_p|(f+g) = T_p|f + T_p|g$$

(this follows directly from definition).

This property will also remain modulo 2.

Behaviour of $\mathcal{F}_i$ Suppose $f \in \mathcal{F}_i$ [2], using 2.4.2, we have:

$$f = \sum_{m \equiv i \bmod 8} \mu_m \Delta^m = \sum_{n \equiv i \bmod 8} c(n) q^n$$

From the definition of Hecke operator modulo 2 (2.5.1), we have:

$$\overline{T_p}|f = \sum_{n \in \mathbb{N}} \gamma(n) q^n \quad \text{with } \gamma(n) = \begin{cases} c(np) & \text{if } p \nmid n \\ c(np) + c(n/p) & \text{if } p \mid n \end{cases}$$

$c(np)$: We have $np \not\equiv i \bmod 8 \implies c(np) = 0$.

$c(n/p)$: As p is an odd prime, it is an odd number, so from 2.4.2, $p^2 \equiv 1 \bmod 8$, so $p^{-2} \equiv 1 \bmod 8$ as well (with p^{-2} seen mod8).

Therefore, $np \not\equiv i \bmod 8 \implies n/p \equiv {}^{np}/_{p^2} \equiv np \not\equiv i \bmod 8$.

$\gamma(n)$: We conclude that $n \equiv np^2 \not\equiv pi \bmod 8 \implies \gamma(n) = 0$

Using 2.4.2 again, we deduce that $\overline{T_p}|f \in \mathcal{F}_j$ with $j \equiv pi \bmod 8$.

Overall, we have the following:

$$f \in \mathcal{F}_i \implies \overline{T_p}|f \in \mathcal{F}_j \text{ with } j \equiv pj \bmod 8$$

Non-Nullity of Hecke Opereator We will prove a property that directly implies the non nullity of Hecke Operators.

This property follows the idea developed in [Ono, 2004, p.33].

Property 2.1. *If $f \in \mathcal{F}$, and $\overline{T_p}|f = 0$ for all odd primes p, then either $f = 0$ or $f = \Delta$. That is, only Δ and 0 give zero after applying any Hecke operator.*

Proof. Let's denote by $a(n)$ the coefficients of the q-expansion of f in the usual way ($f(z) = \sum_{n=0}^{\infty} a(n) q^n$, with $q = e^{2\pi i z}$). With p an odd prime, we similarly define $\overline{T_p}|f(z) = \sum_{n=0}^{\infty} \gamma(n) q^n$ with $\gamma(n) = c(np) + c(n/p)$.

1. if r simple odd:
   $p \nmid n$ gives $0 = \gamma(n) = a(np)$, so $a(r) = 0$
2. If r odd of power 3 or more:
   Putting $n = mp^2$, we get: $0 = \gamma(mp^2) = a(mp^3) + a(mp) = a(mp^3)$.
   Thus, $a(r) = 0$

Thus, $a(r) \neq 0$ implies r is an odd square. Note that $0 = \gamma(np) = a(np^2) + a(n)$, so $a(1) = 1$ will implies $a(r) = 1$ for all odd squares r. In this case, $f = \Delta$.

Similarly, $a(1) = 0$ makes $a(n) = 0$ for all n. Therefore, f may only be Δ or 0. $\square$

An immediate consequence (by taking the contra-positive) of this property is that if $f \neq 0, \Delta$, then there exists a p such that $\overline{T_p}|f \neq 0$. Thus, for any $k > 1$, we get that there is a prime p such that $\overline{T_p}|\Delta^k \neq 0$. This means that $\overline{T_p}$ is never the null operator (so reduction modulo two doesn't become trivial).

Non-nullity of Hecke operators is also implied by the following property, which is new to this paper:

[2] By abuse of notation, we denote by f a modular forms modulo 2 (instead of $\overline{f}$).

Property 2.2. *In fact, we have* $\overline{T_p}|\Delta^p = \Delta + \mathcal{O}(\Delta^9)$ *for any odd primes* p.
That is, $\overline{T_p}|\Delta^p = \Delta + \left[\Delta^{k_1} + \Delta^{k_2} + \cdots + \Delta^{k_r}\right]$ *with* $k_i \geq 9 \quad \forall 1 \leq i \leq r$ *(in fact, we also have* $k_i \equiv 1 \bmod 8 \quad \forall 1 \leq i \leq r$*).*

Proof. We denote $c(n)$ the coefficients of the q-expansion of Δ^p and $\gamma(n)$ the ones of $T_p|\Delta^p$. We recall form definition that $\gamma(1) = c(p)$ (since $p \nmid 1$). Now, $c(p) = 1$ (in fact, p is the smallest power of q that appear in the q-expansion of Δ^p). So $\overline{T_p}|\Delta^p = \Delta + \mathcal{O}(\Delta^3)$.

But now using Behaviour of Hecke operators in $\mathcal{F}_i$, we have:

$$\overline{T_p}|\Delta^p = \Delta^{k_0} + \Delta^{k_1} + \Delta^{k_2} + \cdots + \Delta^{k_r} \qquad k_i \neq k_j \text{ if } i \neq j$$

with $k_0 \equiv k_j \bmod 8$ for all $0 \leq j \leq r$. As $k_0 = 1$, this means that $k_j \equiv 1 \bmod 8$. Therefore, the smallest power of Δ appearing in $\overline{T_p}|\Delta^p$ apart from 1 is 9. This gives the proposition statement. $\square$

Note that with this proof, we know an explicit modular form such that the Hecke operator doesn't vanish.

2.5.3 Nil-potency

The properties of Hecke operators is that, given a modular form f, if we apply a Hecke operators enough times, the form will become zero (i.e. they are nilpotent). The strategy to show this is to prove that for any k (odd), and any prime p, we have:

$$\overline{T_p}|\Delta^k = \sum_{j<k} \mu_j \Delta^j$$

The proof of this property will be divided in two main steps:

Order of Δ doesn't increase We first want to show that:

$$\overline{T_p}|\Delta^k = \sum_{j\leq k} \mu_j \Delta^j$$

Let f be a modular form modulo 2 of maximum degree $2n_0 - 1$ (in Δ), i.e.

$$f = \Delta^{2n_0-1} + \Delta^{2n_1-1} + \cdots + \Delta^{2n_m-1}$$

with $n_0 > n_1 > \cdots > n_m$. So $f \in \mathcal{F}(n_0)$ Then, there must exist a modular forms $g \in M_{6n_0}$ such that $\overline{g} = f$. For example, we can take

$$g = \Delta^{2n_0-1} + E_2^{6(n_0-n_1)}\Delta^{2n_1-1} + \cdots + E_2^{6(n_0-n_m)}\Delta^{2n_m-1}$$

(it is straightforward to check that $g \in M_{6n_0}$ and $\overline{g} = f$, as it was designed for). We know that $\overline{T_p}|f = \overline{T_p}|\overline{g} = \overline{T_p|g}$. Remark as well that as $T_p : M_{6n_0} \to M_{6n_0}$, we have $T_p|g \in M_{6n_0}$, so the maximum degree of Δ that appear in $T_p|g$ is $2n_0 - 1$. Thus, the maximum degree of $\overline{\Delta}$ that appear in $\overline{T_p|g} = \overline{T_p}|f$ is $2n_0 - 1$. Therefore, the degree of f doesn't increase when applying $\overline{T_p}$ to it.

Order of Δ decrease Since T_p and $\overline{T_p}$ have exactly the same action on q-expansions of modular forms, we can interchange them as we want. By abuse of notation (again), we denote the reduction of T_p modulo 2 (usually denoted $\overline{T_p}$) by T_p as well.

Now that we have proved that

$$T_p|\Delta^k = \sum_{j\leq k} \mu_j \Delta^j$$

we need to show that $\mu_k = 0$, so that the maximum order of Δ in fact effectively decrease.

Let's look at $\mathcal{F}(k)$ as a vector space over $\mathbb{F}_2$ with basis $\{\Delta, \Delta^3, \cdots, \Delta^k\}$. We may represent a modular form modulo 2 by a k-vector over $\mathbb{F}_2$ (note that even powers of Δ will always be zero, but we keep track of them to lighten notation). Then, as T_p are linear (see 2.5.2), we can represent each operator T_p with a matrix. Let A_p be the $(k \times k)$-matrix (over $\mathbb{F}_2$) representing the action of T_p on $\mathcal{F}(k)$. Since the order of Δ doesn't increase when applying a Hecke operator, the matrix A_p should be upper-triangular, i.e.:

$$A_p = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,k} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,k} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{k,k} \end{pmatrix}$$

We need to show that the coefficients $a_{i,i}$ are zero.

We will do this by induction. Suppose we know T_p decrease the degree of Δ^j for all $j \leq k-1$. Translating this information to the matrix, it means that $a_{j,j} = 0$ for all $j \leq k-1$. Then, we only need to show that $a_{k,k} = 0$.

Now that we have all this information on the diagonal, it makes sense to study the trace: $\mathrm{Tr}(A_p) = a_{k,k}$. A nice interpretation of the Trace should give us an equation for $a_{k,k}$.

We can interpret the trace as the sum of eigenvalues of the matrix A_p, i.e. eigenvalues of the Hecke operator T_p. Some knowledge about eigenvalues of Hecke operators has been proved already (see Hatada [1979]), we have: For p an odd prime, if λ_p is an eigenvalue of T_p, we have the congruence: $\lambda_p \equiv 1 + p \bmod 8$ Since p is an odd (prime) number, we get: $\lambda_p \equiv 0 \bmod 2$. As this is true for all eigenvalues of T_p, we have that the sum of eigenvalues (which corresponds to the trace of the matrix) is zero over $\mathbb{F}_2$. Thus: $a_{k,k} = \mathrm{Tr}(A_p) \equiv 0 \bmod 2$

Now, this is a proof by induction, but the first case really is $k = 0$, in which case, all modular forms are just 0, so all Hecke operators are obviously zero so nilpotent in this case.

Therefore, we proved the nilpotence modulo 2 of Hecke operators T_p for all p odd primes.

2.5.4 Expression as a Sum of Powers of Δ

As the degree of a modular form doesn't increase after applying a Hecke operator, we can apply this the modular form Δ^k to get:

$$T_p|\Delta^k = \sum_{\substack{j \leq k \\ j \text{ odd}}} \mu_j \Delta^j$$

As we know, moreover, that the degree of a modular form will in fact decrease, we deduce that in fact:

$$T_p|\Delta^k = \sum_{\substack{j \leq k-2 \\ j \text{ odd}}} \mu_j \Delta^j \tag{*}$$

The observation on $\mathcal{F}_i$ (in 2.5.2) leads us to the formula:

$$T_p|\Delta^k = \sum_{\substack{j \leq k-2 \\ j \equiv pk \bmod 8}} \mu_j \Delta^j \tag{**}$$

(since $\Delta^k \in \mathcal{F}_i$ with $k \equiv i \bmod 8$)

2.5.5 Examples (for Small Powers of Δ)

We will describe the behaviour of Hecke operators when applied to Δ^k with k odd, $k \leq 7$.

Δ Clearly, from (*), we have $T_p|\Delta = 0$, since the sum is empty (for any p odd prime).

Δ^3 From (*), we have $T_p|\Delta^3 = \Delta$ or 0.
Moreover, (**) gives $T_p|\Delta^3 = 0$ if $1 \not\equiv 3p \bmod 8$ i.e. if $p \not\equiv 3 \bmod 8$.

Now, if $p \equiv 3 \bmod 8$, we may only look at the coefficient q^1 of $T_p|\Delta^3$ (if it is 1, $T_p|\Delta^3 = \Delta$ and if it is 0, $T_p|\Delta^3 = 0$, as there is no other possibilities).

From definition (in 2.5.1), we have that the coefficient of q^1 is $\gamma(1) = c(p)$ (since $p \nmid 1$) with c the q coefficients of Δ^3.

From (2.3.1), the non-zero coefficients of Δ are odd squares. So we have:

$$(\Delta(q))^3 = \left(\sum_{m=0}^{\infty} q^{(2m+1)^2}\right)^3 = \sum_{n=0}^{\infty} \#\{m_1, m_2, m_3 \text{ odds } | m_1^2 + m_2^2 + m_3^2 = n\} q^n$$

As, $c(p)$ is th p^{th} coefficient of Δ^3, $c(p) = \#\{m_1, m_2, m_3 \text{ odds } | m_1^2 + m_2^2 + m_3^2 = p\} \bmod 2$ corresponds (mod2) to the number of ways to write p as sum of three odd squares.

Now, there are a few possible cases: If $p = 3$, then $m_1 = m_2 = m_3 = 1$ is the only solution, and $c(p) = 1$ so $T_p|\Delta^3 = \Delta$. Now, we consider

- $m_1 = m_2 = m_3$: in this case, $m_1^2 + m_2^2 + m_3^2$ isn't prime, unless $p = 3$ (but we handled this case already). So there is no solution of this form.

- $m_1 \neq m_2 \neq m_3 \neq m_1$ (i.e. all distinct): in this case, if (m_1, m_2, m_3) is a solution, the so is (m_1, m_3, m_2), (m_2, m_1, m_3), (m_2, m_3, m_1), (m_3, m_1, m_2) and (m_3, m_2, m_1). Therefore, there is an even number of solutions, so the contribution to $c(p)$ of solutions of this type is 0.

- $m_1 = m_2 \neq m_3$ in this case, if (m_1, m_2, m_3) is a solution, the so is (m_1, m_3, m_2), (m_3, m_1, m_2). Therefore, there is an odd number of solutions, so the contribution to $c(p)$ of solutions of this type is 1 (if such kind of solution exist).

We want to know if there are solutions to $a^2 + 2b^2 = p$ with a and b odds. We look at the integers $\mathbb{Z}[\sqrt{-2}]$: If p splits, then there is a pair (a, b) of integers such that $(a + b\sqrt{-2})(a - b\sqrt{-2}) = p$, i.e. $a^2 + 2b^2 = p$. Now, p splits in $\mathbb{Z}[\sqrt{-2}]$ if $\left(\frac{-2}{p}\right) = +1$. As $p \equiv 3 \bmod 8$, $\left(\frac{2}{p}\right) = -1$ and $\left(\frac{-1}{p}\right) = -1$, so $\left(\frac{-2}{p}\right) = +1$ (by the first and second supplement to the law of quadratic reciprocity.) Therefore, there are two integers a and b such that $a^2 + 2b^2 = p$. Suppose a is even: then p is even, which is even (as p is an odd prime). Thus, a must be odd. Suppose b is even: then $a^2 + 2b^2 \equiv a^2 \equiv 1 \bmod 8$, but $p \equiv 3 \bmod 8$ Thus, b must be odd.

Therefore, if $p \equiv 3 \bmod 8$, we found a pair of odd numbers a and b such that $a^2 + 2b^2 = p$, so $c(p) = 1$, and thus $T_p|\Delta^3 = \Delta$.

Δ^5 From (*), we have $T_p|\Delta^5 = \Delta^3$ or Δ or 0.
Moreover, (**) gives:

$$\begin{aligned} p \equiv 7 \bmod 8: &\quad T_p|\Delta^5 = \Delta^3 \text{ or } 0 &&\text{if } 3 \equiv 5p \bmod 8 \quad \text{i.e. } p \equiv 7 \bmod 8 \\ p \equiv 5 \bmod 8: &\quad T_p|\Delta^5 = \Delta \text{ or } 0 &&\text{if } 1 \equiv 5p \bmod 8 \quad \text{i.e. } p \equiv 5 \bmod 8 \\ p \equiv 1 \text{ or } 3 \bmod 8: &\quad T_p|\Delta^5 = 0 &&\text{else} \end{aligned}$$

$p \equiv 7 \bmod 8$ Now, if $p \equiv 7 \bmod 8$, we may only look at the coefficient q^3 of $T_p|\Delta^5$ (if it is 1, $T_p|\Delta^5 = \Delta^3$ and if it is 0, $T_p|\Delta^3 = 0$, as there is no other possibilities).

From definition (in 2.5.1), we have that the coefficient of q^3 is $\gamma(3) = c(3p)$ (since $p \nmid 3$) with c the q coefficients of Δ^5.

From (2.3.1), the none zero coefficients of Δ are odd squares. Now, $c(3p)$ is th p^{th} coefficient of Δ^5. We have:

$$(\Delta(q))^5 = \left(\sum_{m=0}^{\infty} q^{(2m+1)^2}\right)^5 = \sum_{n=0}^{\infty} \#\{m_1, m_2, m_3, m_4, m_5 \text{ odds } | m_1^2 + m_2^2 + m_3^2 + m_4^2 + m_5^2 = n\} q^n$$

So $c(3p) = \#\{m_1, m_2, m_3, m_4, m_5 \text{ odds } | m_1^2 + m_2^2 + m_3^2 + m_4^2 + m_5^2 = 3p\} \bmod 2$ corresponds (mod2) to the number of ways to write $3p$ as sum of five odd squares.

Now, we look at the decomposition of m_1, m_2, m_3, m_4, m_5:

- (1+1+1+1+1): in this case, there are $5! = 120$ symmetric solutions, so an even number of solution, so the contribution to $c(3p)$ is 0.
- (1+1+1+2): in this case, there are $3!\binom{5}{2} = 60$ symmetric solutions, (which is even), so the contribution to $c(3p)$ is 0.
- (1+2+2): in this case, there are $\binom{5}{2}\binom{3}{2} = 30$ symmetric solutions, (again, even), so the contribution to $c(3p)$ is 0.
- (1+1+3): in this case, there are $2\binom{5}{3} = 20$ symmetric solutions, so the contribution to $c(3p)$ is 0.
- (1+4): in this case, there are 5 symmetric solutions, so an odd number of solution, so the contribution to $c(3p)$ is 1, if such a solution exist.
- (5): in this case, $m_1^2 + m_2^2 + m_3^2 + m_4^2 + m_5^2 = 5n$ which is never $3p$.

Now, we are looking for pairs of odd integers (a, b) such that $a^2 + 4b^2 = 3p$. that is, $a^2 + c^2 = 3p$ with $c = 2b$ so $c \equiv 2 \bmod 4$. We look at factorization of $3p$ in $\mathbb{Z}[\sqrt{(-1)}]$. Now, $p \equiv 7 \bmod 8$, so $\left(\frac{-1}{p}\right) = -1$ (by the first supplement to the law of quadratic reciprocity). Thus, p remains prime in the Gaussian integers $\mathbb{Z}[\sqrt{(-1)}]$. Therefore, there is no pair of (a, b) integers solution $a^2 + 4b^2 = 3p$.

Thus, $T_p|\Delta^5 \neq \Delta^3$ for any odd prime p.

$p \equiv 5 \bmod 8$ Now, if $p \equiv 5 \bmod 8$, we want to know if $T_p\Delta = \Delta$ or 0. Again, we look at the coefficient q^1 of $T_p|\Delta$, which is the coefficient q^p of Δ^5, i.e. $c(p) = \#\{m_1, m_2, m_3, m_4, m_5 \text{ odds } | m_1^2 + m_2^2 + m_3^2 + m_4^2 + m_5^2 = p\} \bmod 2$, which corresponds (mod2) to the number of ways to write p as sum of five odd squares.

If $p = 5$, then $c(5) = 1$ clearly, and we have $T_5|\Delta^5 = \Delta$. Now, we use again the decomposition of m_1, m_2, m_3, m_4, m_5 from above: we are looking for pairs of odd integers (a, b) such that $a^2 + 4b^2 = p$. that is, $a^2 + c^2 = 3p$ with $c = 2b$ so $c \equiv 2 \bmod 4$. We again look at factorization of p in $\mathbb{Z}[\sqrt{(-1)}]$. Now, $p \equiv 5 \bmod 8$, so $\left(\frac{-1}{p}\right) = 1$ (by the first supplement to the law of quadratic reciprocity). Thus, p remains prime in the Gaussian integers $\mathbb{Z}[\sqrt{(-1)}]$. Therefore, there is a pair of (a, b) integers solution $a^2 + c^2 = p$.

Suppose both a and c are odd: then $a^2 + c^2$ is even, so it may not be any odd prime p. Suppose both a and c are even: then $a^2 + c^2$ is even, so it may not be any odd prime p. So, without loss of generalities, a is odd and c is even. We can write $b = c/2$. Suppose c is even: then $a^2 + 4b^2 \equiv 1 \bmod 8$. As $p \equiv 5 \bmod 8$, $a^2 + 4b^2 \neq p$. So if (a, b) is a solution, then a and b are odds. Thus, $T_p|\Delta^5 = \Delta$ for any odd prime p.

Δ^7 From (*), we have $T_p|\Delta^7 = \Delta^5$ or Δ^3 or Δ or 0. In fact, we have [Nicolas and Serre, 2012a, 2.3]:

$$\begin{array}{lll} T_p|\Delta^7 = & \Delta & \text{if } p \equiv 7 \bmod 16 \\ T_p|\Delta^7 = & \Delta^3 & \text{if } p \equiv 5 \bmod 8 \\ T_p|\Delta^7 = & \Delta^5 & \text{if } p \equiv 3 \bmod 8 \\ T_p|\Delta^7 = & 0 & \text{if } p \equiv 1 \bmod 8 \text{ or } p \equiv -1 \bmod 16 \end{array}$$

2.5.6 Table of Hecke Operators

Here is a table of Hecke operators:

	Δ^1	Δ^3	Δ^5	Δ^7	Δ^9	Δ^{11}	Δ^{13}	Δ^{15}	Δ^{17}	Δ^{19}
T_3	0	Δ	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5+\Delta^{13}$	0	$\Delta^9+\Delta^{17}$
T_5	0	0	Δ	Δ^3	0	0	Δ^9	$\Delta^3+\Delta^{11}$	Δ^5	Δ^7
T_7	0	0	0	Δ	0	0	Δ^3	Δ^9	0	Δ^5
T_{11}	0	Δ	0	Δ^5	Δ^3	$\Delta+\Delta^9$	Δ^7	Δ^{13}	0	$\Delta^9+\Delta^{17}$
T_{13}	0	0	Δ	Δ^3	0	0	$\Delta+\Delta^9$	Δ^{11}	Δ^5	Δ^7
T_{17}	0	0	0	0	Δ	Δ^3	Δ^5	Δ^7	Δ	0
T_{19}	0	Δ	0	Δ^5	Δ^3	$\Delta+\Delta^9$	Δ^7	Δ^{13}	0	$\Delta+\Delta^9+\Delta^{17}$
T_{23}	0	0	0	Δ	0	0	Δ^3	$\Delta+\Delta^9$	0	Δ^5
T_{29}	0	0	Δ	Δ^3	0	0	Δ^9	$\Delta^3+\Delta^{11}$	Δ^5	Δ^7
T_{31}	0	0	0	0	0	0	0	Δ	0	0
T_{37}	0	0	Δ	Δ^3	0	0	$\Delta+\Delta^9$	Δ^{11}	Δ^5	Δ^7
T_{41}	0	0	0	0	0	0	0	0	Δ	Δ^3
T_{43}	0	Δ	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5+\Delta^{13}$	0	$\Delta^9+\Delta^{17}$
T_{47}	0	0	0	0	0	0	0	Δ	0	0

Action of Primes Hecke Operators (primes up to 50) on Modular Forms Modulo 2 (up to Δ^{19}).

A larger table may be found in the appendix (see A.1).

It seems quite random, which makes sense since the Hecke operators depend on prime, and primes appear at random. However, it is interesting to try to find patterns and rules for this table.

In the second column (of Δ^3), we get 1/4 of the primes giving Δ (the other 3/4 giving 0), this is a consequence of Dirichlet Density Theorem, that will be discussed later in this paper. Similarly, in the third column (of Δ^5), we get 1/4 of the primes giving Δ. For similar reason, in the fourth column (of Δ^7), we get 1/4 of the primes giving Δ^3; 1/4 of the primes giving Δ^5; 1/8 of the primes giving Δ; and 3/8 of the primes giving 0.

2.6 Nil-potency Order

In this subsection, we follow the construction from Nicolas and Serre [2012a].
As we know that the Hecke operators are nilpotent, we may want to study the order of nil potentness.

2.6.1 Introduction

Definition For a modular from modulo 2 $f \in \mathcal{F}$, we define the *nil potentness order* to be the smallest integer $g(f)$ such that we have

$$T_{p_1} T_{p_2} \cdots T_{p_{g(f)}} | f = 0$$

for any set of primes numbers $p_1, p_2, \ldots, p_{g(f)} \in \mathbb{P}$. The primes p_i involved do not need to be distinct. Note as well that from commutativity of the Hecke operators, the order of the primes p_i doesn't matter.

By convention, we write $g(0) = -\infty$. With a slight abuse of notation, we will write $g(k)$ for $g(\Delta^k)$.

Properties

Maximum Order All Hecke operators lower by at least two the maximum degree of Δ in the Δ-expansion of a modular form modulo 2 2.5.3. We deduce that $g(f) \leq g(T_p|f) + 1$. Applied to Δ^k, we get: $g(k) \leq g(k-2) + 1$. Therefore, by induction, we have $g(k) \leq \lfloor \frac{k+1}{2} \rfloor$. This implies by the same occasion, the well definiteness of the order of nil potentness for all modular form modulo two.

Minimum Order If the degree of f is strictly greater than 1 (i.e. $f \neq 0, \Delta$), then $g(f) \geq 2$.

We deduce this from the fact that Hecke operators are not null operators in general (2.5.2): Remember that one consequence of non nullity is that if $f \neq 0, \Delta$, then there exists an odd prime p such that $T_p|f \neq 0$.

This directly implies that $g(f) > 1$ if $f \neq \Delta, 0$.

Examples We can compute a few nil potentness "by hand" (we will see later that there is a more direct method):

- $g(0) = -\infty$
- $g(\Delta) = 1$:
  $T_p(\Delta) = 0$ as order of Δ decrease, see 2.5.3
- $g(\Delta^3) = 2$:
  $T_p|\Delta^3 = \Delta$ or 0
  thus: $g(\Delta^3) = 1 + \max(g(\Delta), g(0)) = 2$
- $g(\Delta^3 + \Delta) = 2$
  similarly
- $g(\Delta^5) = 2$:
  $T_p|\Delta^5 = \Delta$ or 0
  thus: $g(\Delta^5) = 1 + \max(g(\Delta), g(0)) = 2$
- $g(\Delta^5 + \Delta^3 + \Delta) = g(\Delta^5 + \Delta^3) = g(\Delta^5 + \Delta) = 2$
  similarly
- $g(\Delta^7) = 3$:
  $T_p(\Delta^7) = \Delta^5$ or Δ^3 or Δ or 0
  thus: $g(\Delta^7) = 1 + \max(g(\Delta^5), g(\Delta^3), g(\Delta), g(0)) = 3$

2.6.2 Code and Height of Natural Numbers

Definition of n_3, n_5 and h We consider a natural number $k \in \mathbb{N}$, and we let α_i be the digits of it's binary representation, i.e.

$$k = \sum_{i=0}^{\infty} \alpha_i 2^i \qquad \text{with } \alpha_i \in \{0, 1\}.$$

We then define $n_3(k)$ and $n_5(k)$ as follows:

$$n_3(k) = \sum_{i=1}^{\infty} \alpha_{2i+1} 2^i = \sum_{i \text{ odd}} \alpha_i 2^{\frac{i-1}{2}} \qquad \text{and} \qquad n_5(k) = \sum_{i=1}^{\infty} \alpha_{2i+2} 2^i = \sum_{i \text{ even}} \alpha_i 2^{\frac{i-2}{2}}$$

We also define h (the *height*)to be the sum of n_3 and n_5, that is:

$$h(k) = n_3(k) + n_5(k)$$

The definition of n_3 and n_5 is equivalent to the followings:

$n_3(k)$ Take k and write it in binary base. Ignore the units digit. Select only the digits corresponding to odd powers of 2. This forms $n_3(k)$.

$n_5(k)$ Again, write k in binary base. Ignore the unit digit. Select only the digits corresponding to even powers of 2. This forms $n_5(k)$.

Example We look at the example $k = 91$: $91 = 1 + 2 + 8 + 16 + 64 = 2^0 + 2^1 + 2^3 + 2^4 + 2^6$

We construct $n_3(k)$ and $n_5(k)$ graphically as follows:

$k = 91$	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
k	1	0	1	1	0	1	1	$1011011 = 91$
$n_3(k)$	1	$\boxed{0}$	1	$\boxed{1}$	0	$\boxed{1}$	1	$011 = 3$
$n_5(k)$	$\boxed{1}$	0	$\boxed{1}$	1	$\boxed{0}$	1	1	$110 = 6$

$$\left.\begin{matrix} n_3 \\ n_5 \end{matrix}\right\} = h(k) = 3 + 6 = 9$$

Basic Properties

Variation Note that neither of h, n_3, n_5 is monotone. That is, in general, it is not true that $h(n+1) \geq h(n)$, neither $n_3(n+1) \geq n_3(n)$ nor $n_5(n+1) \geq n_5(n)$.

$2k$ and $2k+1$ It is straightforward, form definition, that n_3 and n_5 (therefore also h) are the same for an even $2m$ and the next number $2m+1$. Explicitly:

$$\left.\begin{array}{rl} \text{for all } m \in \mathbb{N}: & n_3(2m) = n_3(2m+1) \\ \text{and} & n_5(2m) = n_5(2m+1) \end{array}\right\} \quad h(2m) = h(2m+1)$$

Addition of powers of 2 Moreover, it is clear from definition that m_3, m_5 and h preserve addition of powers of 2, that is:

$$\begin{array}{rcl} n_3(m_1) + n_3(m_2) & = & n_3(m_1 + m_2) \\ n_5(m_1) + n_5(m_2) & = & n_5(m_1 + m_2) \\ h(m_1) + \ h(m_2) & = & h(m_1 + m_2) \end{array} \qquad \text{for all} \qquad \begin{array}{c} m_1 = 2^{k_1},\ k_1 \in \mathbb{N} \\ m_2 = 2^{k_2},\ k_2 \in \mathbb{N} \\ k_1 \neq k_2 \end{array}$$

Powers of 2 For $m = 2^k$, $k > 0$ even:

$$n_3(m) = 0 \quad \text{and} \quad n_5(m) = 2^{\frac{k-2}{2}} = {}^1\!/\!_2\sqrt{m} \quad \text{so} \quad h(m) = {}^1\!/\!_2\sqrt{m}$$

For $m = 2^k$, $k > 0$ odd:

$$n_3(m) = \sqrt{m/2} \quad \text{and} \quad n_5(m) = 0 \quad \text{so} \quad h(m) = \sqrt{m/2}$$

Therefore, if $m = 2^k$, $m \geq 2$, we have:

$$1/2\sqrt{m} \leq h(m) \leq \sqrt{m/2}$$

Lower Bound for h Now, combining with the property above and properties of square root, we have:

$$1/2\sqrt{m} \leq h(m) \quad \text{for all } m \in \mathbb{N},\ m \geq 2.$$

Upper Bound for h Let $m = 2^{k_1} + 2^{k_2} + \cdots + 2^{k_r}$ with $k_1 > k_2 > \cdots > k_r$.
By construction, we have:

- If k_1 is even, $n_3(m) < \sqrt{2^{k_1}} = \sqrt{m} < \sqrt{2m}$
- If k_1 is odd, $n_3(m) < \sqrt{2^{k_1+1}} < \sqrt{2m}$

Thus, in any case $n_3(m) < \sqrt{2m}$.
Again, by construction, we have:

- If k_1 is even, $n_5(m) < \sqrt{2^{k_1}} < \sqrt{m}$
- If k_1 is odd, $n_5(m) < \sqrt{2^{k_1-1}} = \sqrt{m/2} < \sqrt{m}$

Thus, in any case $n_5(m) < \sqrt{m}$.
Therefore, we have $h(m) < \sqrt{2m} + \sqrt{m} = (1+\sqrt{2})\sqrt{m} < 5/2\sqrt{m}$.

Asymptotic Behaviour of h Note that if $m = 0, 1$, we have $h(m) = \sqrt{m} = m$. Thus, we have:

$$1/2\sqrt{m} \leq h(m) < (1+\sqrt{2})\sqrt{m} < 5/2\sqrt{m}$$

for all $m \in \mathbb{N}$. Using the big $\mathcal{O}$ notation (asymptotic behaviour), this is:

$$h(m) = \mathcal{O}(\sqrt{m})$$

Code of Natural Numbers Given a natural number $m \in \mathbb{N}$, we define it's *code* as the pair $[n_3(m), n_5(m)]$.

A Representation of Odd/Even We define the maps:

$$\phi : 2\mathbb{N} \to \mathbb{N} \times \mathbb{N} \quad \text{such that} \quad \phi(2m) = [n_3(2m), n_5(2m)]$$

and

$$\psi : 2\mathbb{N}+1 \to \mathbb{N} \times \mathbb{N} \quad \text{such that} \quad \psi(2m+1) = [n_3(2m+1), n_5(2m+1)]$$

so that

$$\phi^{-1} : \mathbb{N} \times \mathbb{N} \to 2\mathbb{N} \quad \text{such that:}$$

$$\phi^{-1}([m_1, m_2]) = 2\sum_{i=0}^{\infty} \alpha_i 2^{2i} + 4\sum_{i=0}^{\infty} \alpha_i 2^{2i} \qquad \text{with} \begin{cases} m_1 = \sum_{i=0}^{\infty} \alpha_i 2^i \\ m_2 = \sum_{i=0}^{\infty} \beta_i 2^i \end{cases}$$

and

$$\psi^{-1} : \mathbb{N} \times \mathbb{N} \to 2\mathbb{N}+1 \quad \text{such that:}$$

$$\psi^{-1}([m_1, m_2]) = 2\sum_{i=0}^{\infty} \alpha_i 2^{2i} + 4\sum_{i=0}^{\infty} \alpha_i 2^{2i} + 1 \qquad \text{with} \begin{cases} m_1 = \sum_{i=0}^{\infty} \alpha_i 2^i \\ m_2 = \sum_{i=0}^{\infty} \beta_i 2^i \end{cases}$$

(i.e. squaring each power of two composing m_1 and m_2 separately.)
Since ϕ (respectively ψ) has an inverse, it defines a bijection between even (respectively odd) natural numbers and their code.

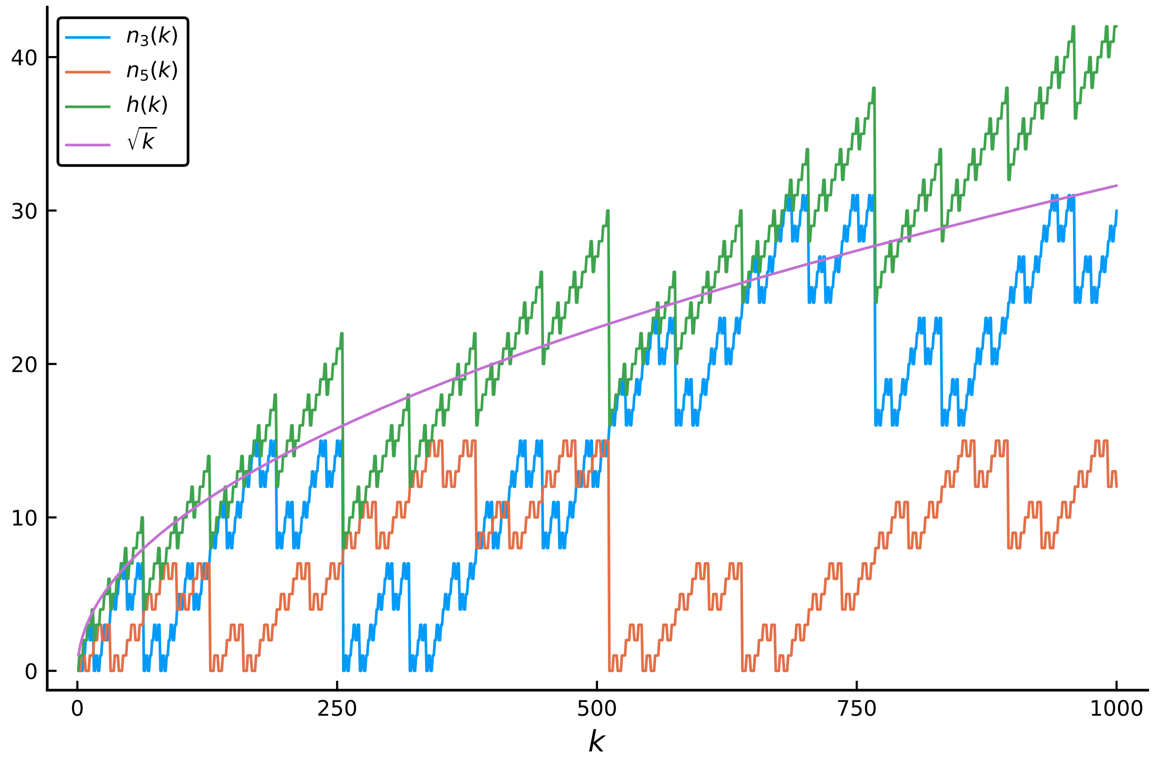
Behaviour

Height's Behaviour We observe the following behaviour:

k	0 , 1	2 , 3	4 , 5	6 , 7	8 , 9	10 , 11	12 , 13	14 , 15
Code of k	[1 , 0]	[0 , 1]	[1 , 1]	[2 , 0]	[3 , 0]	[2 , 1]	[3 , 1]	[0 , 2]
h(k)	0	1	1	2	2	3	3	4

(Larger table in A.3.)

On smaller scale, the behaviour is as follows:



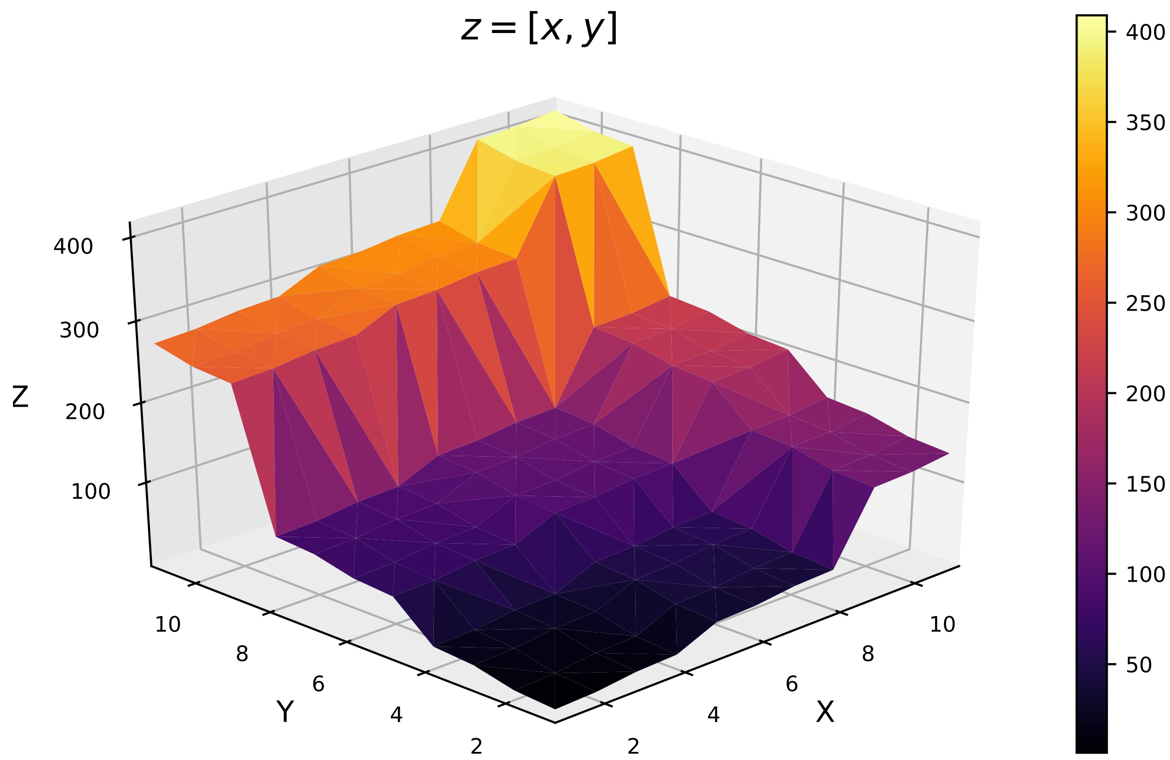
(Plot with other scales in A.4.)

Code's Behaviour It may be interesting to represent how the code grows according to both parameters. First, we can make a small table of values:

	0	1	2	3	4	5	6	7	8	9	10
0	0	4	16	20	64	68	80	84	256	260	272
1	2	6	18	22	66	70	82	86	258	262	274
2	8	12	24	28	72	76	88	92	264	268	280
3	10	14	26	30	74	78	90	94	266	270	282
4	32	36	48	52	96	100	112	116	288	292	304
5	34	38	50	54	98	102	114	118	290	294	306
6	40	44	56	60	104	108	120	124	296	300	312
7	42	46	58	62	106	110	122	126	298	302	314
8	128	132	144	148	192	196	208	212	384	388	400
9	130	134	146	150	194	198	210	214	386	390	402
10	136	140	152	156	200	204	216	220	392	396	408

(Larger table in A.4.)

But it isn't very visual, so another way to view it is as a surface (obtained by linking with triangles the points plotted). On the grid $[0, 10]^2$, we plot the surface $z = [x, y]$ i.e. z is the integer with code $[x, y]$.



(Other scales of plots available in A.4.)

Order Relation We define the following order relation on natural numbers: For $m, l \in \mathbb{N}$, $m \prec l$ if $h(m) < h(l)$ or $h(m) = h(l)$ and $n_5(m) \prec n_5(l)$. This relation is a total order, this is straightforward to check.

2.6.3 Action of T_3 and T_5

h on modular forms For the rest of the section, we will write a modular form $f \in \mathcal{F}$ as follows: $f = \Delta^{m_1} + \Delta^{m_2} + \cdots + \Delta^{m_r}$ with $m_1 > m_2 > \cdots > m_r$. In this case, m_1 is the *degree* of f, and we will denote it by ∂f. We write $\partial f = -\infty$ if $f = 0$. We define $h(f) = h(m_1)$, and $h(f) = -\infty$ if $f = 0$. Similarly, we define $n_3(f) = n_3(m_1)$ and $n_5(f) = n_5(m_1)$, and $n_3(f) = n_5(f) = -\infty$ if $f = 0$.

Action of T_3 Here, we want to give a description for the behaviour of $h(T_3|f)$ and $\partial T_3|f$.

Proposition 2.1. *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2.*

1. *In general, $h(T_3|f) \leq h(f) - 1$.*

2. *If $n_3(f) > 0$, then $h(T_3|f) = h(f) - 1$ and $\partial T_3|f$ has code $[n_3(f) - 1, n_5(f)]$.*

This proposition is stated in [Nicolas and Serre, 2012a, §4], and a complete proof is given in Gerbelli-Gauthier [2014].

Action of T_5 Similarly to the above for T_3, we want to give a description for the behaviour of $h(T_5|f)$ and $\partial T_5|f$.

Proposition 2.2. *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2.*

1. *In general, $h(T_5|f) \leq h(f) - 1$.*

2. *If $n_5(f) > 0$, then $h(T_5|f) = h(f) - 1$ and $\partial T_5|f$ has code $[n_3(f), n_5(f) - 1]$.*

Again, this proposition is stated in [Nicolas and Serre, 2012a, §4], and a complete proof is given in Gerbelli-Gauthier [2014].

2.6.4 Formula for the Order of Nil-potency

Lower bound

Property 2.3. *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2 of degree m_1. Then we have:*

$$T_3^{n_3(m_1)} T_5^{n_5(m_1)} | f = \Delta$$

Proof. We apply $n_3(m_1)$ times the proposition about T_3 above, and $n_5(m_1)$ times the proposition about T_5, to get that the degree of $g = T_3^{n_3(m_1)} T_5^{n_5(m_1)} | f$ has code $[0, 0]$. The propositions also implie that $h(g) > -\infty$, i.e. $g \neq 0$. Therefore, $\partial g > -\infty$, so it must be odd. since it has code $[0, 0]$, $\partial g = 1$, thus $g = \Delta$. $\square$

Corollary 2.1. *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2. We have $g(f) \geq h(f) + 1$.*

Proof. This is directly implied by the proposition: As $T_3^{n_3(m_1)} T_5^{n_5(m_1)} | f = \Delta \neq 0$, we have $g(f) \geq n_3(f) + n_5(f) + 1 = h(k) + 1$. $\square$

Representation of $\mathcal{F}$

Table This means that for each k odd, there is pair $[a, b] = [n_3(k), n_5(k)] \in \mathbb{N} \times \mathbb{N}$ (which corresponds to the code of k), such that $T_3^a T_5^b | \Delta^k = \Delta$. Explicitly, $T_3^{n_3(k)} T_5^{n_5(k)} | \Delta^k = \Delta$.

Thus, we can arrange all odd powers of the discriminant Δ in a table, such that applying the corresponding Hecke operators give exactly Δ^1:

	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	T_5^5	T_5^6	T_5^7	T_5^8	T_5^9	T_5^{10}
T_3^0	Δ^{1}	Δ^{5}	Δ^{17}	Δ^{21}	Δ^{65}	Δ^{69}	Δ^{81}	Δ^{85}	Δ^{257}	Δ^{261}	Δ^{273}
T_3^1	Δ^{3}	Δ^{7}	Δ^{19}	Δ^{23}	Δ^{67}	Δ^{71}	Δ^{83}	Δ^{87}	Δ^{259}	Δ^{263}	Δ^{275}
T_3^2	Δ^{9}	Δ^{13}	Δ^{25}	Δ^{29}	Δ^{73}	Δ^{77}	Δ^{89}	Δ^{93}	Δ^{265}	Δ^{269}	Δ^{281}
T_3^3	Δ^{11}	Δ^{15}	Δ^{27}	Δ^{31}	Δ^{75}	Δ^{79}	Δ^{91}	Δ^{95}	Δ^{267}	Δ^{271}	Δ^{283}
T_3^4	Δ^{33}	Δ^{37}	Δ^{49}	Δ^{53}	Δ^{97}	Δ^{101}	Δ^{113}	Δ^{117}	Δ^{289}	Δ^{293}	Δ^{305}
T_3^5	Δ^{35}	Δ^{39}	Δ^{51}	Δ^{55}	Δ^{99}	Δ^{103}	Δ^{115}	Δ^{119}	Δ^{291}	Δ^{295}	Δ^{307}
T_3^6	Δ^{41}	Δ^{45}	Δ^{57}	Δ^{61}	Δ^{105}	Δ^{109}	Δ^{121}	Δ^{125}	Δ^{297}	Δ^{301}	Δ^{313}
T_3^7	Δ^{43}	Δ^{47}	Δ^{59}	Δ^{63}	Δ^{107}	Δ^{111}	Δ^{123}	Δ^{127}	Δ^{299}	Δ^{303}	Δ^{315}
T_3^8	Δ^{129}	Δ^{133}	Δ^{145}	Δ^{149}	Δ^{193}	Δ^{197}	Δ^{209}	Δ^{213}	Δ^{385}	Δ^{389}	Δ^{401}
T_3^9	Δ^{131}	Δ^{135}	Δ^{147}	Δ^{151}	Δ^{195}	Δ^{199}	Δ^{211}	Δ^{215}	Δ^{387}	Δ^{391}	Δ^{403}
T_3^{10}	Δ^{137}	Δ^{141}	Δ^{153}	Δ^{157}	Δ^{201}	Δ^{205}	Δ^{217}	Δ^{221}	Δ^{393}	Δ^{397}	Δ^{409}

(A larger table can be found in A.4.)

Exact Formula We derive here an explicit formula for the order of nil-potency of a modular form modulo 2.

Theorem 5 (Order of Nil-potency of Modular Forms Modulo 2). *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2. The order of nil-potency is exactly $g(f) = h(f) + 1$.*

What remains to prove is that $g(f) \leq h(f) + 1$. This is proved in [Nicolas and Serre, 2012a, §5].

From this follows a new remark, which is useful to estimate computations times: As $g(f) = h(f)+1$, and $h(f) = h(\partial f) = \mathcal{O}(\sqrt{\partial f})$, we have $g(f) = \mathcal{O}(\sqrt{\partial f})$, i.e. the nil-potency order of a modular form behaves asymptotically as the square root of its degree.

Corollary 2.2. *Let $f \in \mathcal{F}$ be a non-zero modular form modulo 2. If $T_3|f = T_5|f = 0$, then $f = \Delta$.*

Proof. By the previous proposition, we have both $n_3(f) = 0$ and $n_5(f) = 0$. Thus, ∂f has code $[0, 0]$. Since $f \neq 0$ and $f \in \mathcal{F}$, this means $f = \Delta$. □

3 Hecke Algebra

This section follows from Nicolas and Serre [2012b].

We recall $\mathcal{F} = \langle \Delta^k \mid k \text{ odd} \rangle$, i.e. $\mathcal{F} = \langle \Delta, \Delta^3, \Delta^5, \Delta^7, \Delta^9, \dots \rangle$

3.1 Definition

We redefine $\mathcal{F}(n) = \langle \Delta, \Delta^3, \Delta^5, \dots, \Delta^{2n-1} \rangle$ so that $\dim(\mathcal{F}(n)) = n$

We define $A(n)$ as the $\mathbb{F}_2$-subalgebra of $\mathrm{End}(\mathcal{F}(n))$ given by $\mathbb{F}_2$ and T_p. That is, if $\mathfrak{m}(n) = \{T_{p_1} \cdot T_{p_2} \cdots T_{p_k} | p_1, p_2, \dots, p_k \in \mathbb{P}, k \geq 1\}$ is a sub-vector-space of $\mathcal{F}$, we get $A(n) = \mathbb{F}_2 \oplus \mathfrak{m}(n)$.

Property 3.1. *$\mathfrak{m}(n)$ is the only maximal ideal of $A(n)$.*

Proof. Firstly, we note that $A(n)/\mathfrak{m} \cong \mathbb{F}_2$. Since $\mathbb{F}_2$ is a field, $\mathfrak{m}$ must be a maximal ideal.

Now, suppose I is an other (i.e. $I \neq \mathfrak{m}$) maximal ideal of $A(n)$. Then there is an operator $u \in \mathfrak{m}$ such that $(1 + u) \in I$. Since Hecke operators are nilpotent 2.5.3, there exists $n \in \mathbb{N}$ such that $u^n = 0$. By induction, $(1 + u^n) \in I$ for all $n \geq 1$. □

Note that as Hecke operators are all nilpotent 2.5.3, the ideal $\mathfrak{m}(n)$ is itself nilpotent. In fact, from the minimum 2.6.1 and maximum 2.6.1 nil-potency order property extend to the ideal $\mathfrak{m}$.

Let the dual of $\mathbb{F}(n)$ be $\mathcal{F}(n)^* = \{F : \mathcal{F}(n) \to \mathbb{F}_2\}$. Then $\mathbb{F}(n)^*$ is an $A(n)$-module with operation $(u \cdot F)(f) = F(u|f)$ for $u \in A(n)$ and $F \in \mathcal{F}(n)$.

We define e_n to be the element of $\mathcal{F}(n)$ such that $e_n(\Delta) = 1$ and $e_n(\Delta^{2j+1}) = 0$ for all $1 \leq j < n$ (i.e. characteristic of Δ).

Denote the q-coefficients of a modular form f by $a_m(f)$, so $f = \sum_{m>0} a_m(f)q^m$. Then $e_n(f) = a_1(f)$. For an odd prime p, have $a_1(T_p|f) = a_p(f)$, so $T_p \cdot e_n(f) = a_p(f)$. By induction, this gives for odd primes $p_1, p_2, \dots, p_k$:

$$T_{p_1} T_{p_2} \cdots T_{p_k} \cdot e_n(f) = a_{p_1 p_2 \cdots p_k}(f)$$

To fit naturally with the above definitions, we define the Hecke algebra A as follows: As $\mathcal{F}(n) \subset \mathcal{F}(n+1)$, we can restrict elements of $A(n+1)$ to $\mathcal{F}$ to obtain an element of $A(n)$. If we consider the map $\phi_n : A(n+1) \to A(n)$ to be the restriction to $\mathcal{F}(n)$, then ϕ_n is a homomorphism. As $A(1)$ is either identity or zero, $A(1) \cong \mathbb{F}_2$. Therefore, we have the chain:

$$\cdots \to A(n+1) \to A(n) \to A(n-1) \to \cdots \to A(2) \to A(1) \cong \mathbb{F}_2$$

We then define the Hecke algebra A to be the projective limit of the above $A(n)$ as $n \to \infty$. Explicitly, this means

$$A = \varprojlim_{n \in \mathbb{N}} A(n) = \{T_{p_1} \cdot T_{p_2} \cdots T_{p_k} | p_1, p_2, \dots, p_k \in \mathbb{P}, k \geq 0\}.$$

.

3.2 Basic Properties

Property 3.2. *Note that for a non zero modular forms $f \in \mathcal{F}(n)$, there exists an operator $u \in A(n)$ such that $e_n(u|f) = 1$.*

Proof. We can write $f = q^m + \mathcal{O}(q^{m+1})$ for some m odd (as $\mathcal{F}(n)$ is generated by odd powers of Δ). Now, as m is odd, $m = p_1 p_2 \cdots p_k$ with p_i odd primes for all $1 \leq i \leq k$. Then, by the above, $T_{p_1} T_{p_2} \cdots T_{p_k} \cdot e_n(f) = a_m(f) = 1$. Letting $u = T_{p_1} T_{p_2} \cdots T_{p_k}$, we have $e_n(u|f) = u \cdot e_n(f) = 1$. □

Property 3.3. *$\mathcal{F}(n)^*$ [3] is free as an $A(n)$-module, with basis e_n; i.e., $\mathcal{F}(n)^* = A(n) \cdot e_n$.*

Proof. By contradiction, suppose (e_n) (the $A(n)$-module generated by e_n) isn't $\mathcal{F}(n)^*$. The there must be a non-zero modular form $f \in \mathcal{F}(n)$ such that $u \cdot e_n(f) = 0$ for all $u \in A(n)$. This would contradict the last property. Therefore, we have $\mathcal{F}(n)^* = A(n) \cdot e_n$. □

Corollary 3.1. *From last property, we deduce:*

- *The map $\phi : A(n) \to \mathcal{F}(n)^*$ such that $\phi(u) = u \cdot e_n$ is a bijection.*
- *The dimension of $A(n)$ is n.*
- *There is a bijection $\phi : A(n) \to \mathcal{F}(n)^*$.*

Proof. We prove separately:

- This follows directly from the fact that $\mathcal{F}(n)^* = A(n) \cdot e_n$.
- We have $\dim(A(n)) = \dim(\mathcal{F}(n)^*)$ by the above, and $\dim(\mathcal{F}(n)^*) = \dim(\mathcal{F}(n))$ by duality.
- Since $\mathcal{F}(n)^* \leftrightarrow A(n)$ [4], $\mathcal{F}(n)^{**} \leftrightarrow A(n)^*$. And as $\mathcal{F}(n)^{**} \cong \mathcal{F}(n)$, we have $\mathcal{F}(n) \leftrightarrow A(n)^*$.

□

3.3 As generated by T_3 and T_5

The Hecke algebra is in fact generated by powers of T_3 and T_5, i.e., we have:

$$A = \mathbb{F}_2 [[T_3, T_5]]$$

The strategy to show this splits in two parts: First, we show that $A(n) = \mathbb{F}_2 [T_3, T_5]$ (with T_3 and T_5 seen in $A(n)$). Then, we show that this equation remains when taking the limit.

Property 3.4. *We have $A(n) = \mathbb{F}_2 [T_3, T_5]$.*

Proof. We define $A'(n) = \mathbb{F}_2 [T_3, T_5]$ as a subalgebra of $A(n)$. This is a local algebra (i.e. it has a unique maximal ideal). The idea here is similar to the one involved in A being a local algebra (see 3.1). We denote the maximal ideal of $A'(n)$ by $\mathfrak{m}'$.

Suppose, for contradiction, that $A' \neq A$, so $\dim(A') < n$.

If $\mathcal{F}(n)^*$ (seen as a $A'(n)$-module), was cyclic (i.e. generated by a single element), then $\dim(\mathcal{F}(n)^*) < n$. However, we know $\dim(\mathcal{F}(n)^*) = n$, so $\mathcal{F}(n)^*$ isn't cyclic.

We define $V = {}^{\mathcal{F}(n)^*}/_{\mathfrak{m}'\mathcal{F}(n)^*}$. Nakayama lemma under the statement for maximal ideals implies that V has dimension > 1.

Now we put $U = \{f \in \mathcal{F}(n) \mid \ a|f = 0 \quad \forall a \in \mathfrak{m}'\}$, i.e. the modular forms that are zero after application of any operator in $\mathfrak{m}'$. We want to show that this vector space U has dimension > 1.

As vector spaces, we know: $({}^{W_1}/_{W_2})^* \cong W_2^0$, so:

$$\left(\frac{\mathcal{F}(n)^*}{\mathfrak{m}'\mathcal{F}(n)^*}\right)^* \cong \left(\mathfrak{m}'\mathcal{F}(n)^*\right)^0$$

Where

$$\left(\mathfrak{m}'\mathcal{F}(n)^*\right)^0 = \{\tilde{f} \in \mathcal{F}(n)^{**} \mid \tilde{f}(a) = 0 \quad \forall a \in \mathfrak{m}'\mathcal{F}(n)^*\}$$

[3] $\mathcal{F}(n)^*$ denotes the dual of $\mathcal{F}(n)$.

[4] $A \leftrightarrow B$ means there exists a bijection from A to B

is the inhalator of $\mathfrak{m}'\mathcal{F}(n)^*$. We know there is an isomorphism between $\mathcal{F}(n)^{**}$ and $\mathcal{F}(n)$[5]. Thus, we have:

$$\left(\mathfrak{m}'\mathcal{F}(n)^*\right)^0 \cong \{f \in \mathcal{F}(n) \mid \ a|f = 0 \quad \forall a \in \mathfrak{m}'\mathcal{F}(n)^*\}$$

But one may recognize that this is exactly U.

Therefore, $\dim(U) = \dim((\mathfrak{m}'\mathcal{F}(n)^*)^0) = \dim(V) = \dim(V^*) > 1$

$\{0, \Delta\}$ is a subspace of U with dimension 1. As $\dim(U) > 1$, there must exists a modular form $f \in \mathcal{F}(n)$ such that: f is neither 0 or Δ, and $a|f = 0$ for all $a \in \mathfrak{m}'$. In particular, this means that $T_3|f = 0$ and $T_5|f = 0$. However, this contradicts the Order of Nil-potency of Modular Forms Modulo 2 corollary 2.6.4.

Thus we have $A(n) = A'(n) = \mathbb{F}_2\left[T_3, T_5\right]$. □

Property 3.5. *We have* $A = \mathbb{F}_2\left[[T_3, T_5]\right]$.

Proof. For any n, there is a homomorphism $\psi_n : \mathbb{F}_2\left[x, y\right] \to A(n)$ with $\psi_n(x) = T_3$ and $\psi_n(x) = T_5$. Therefore, we can take the limit as $n \to \infty$ to get a homomorphism $\psi : \mathbb{F}_2\left[[x, y]\right] \to A$ such that $\psi(x) = T_3$ and $\psi(x) = T_5$.

The fact that ψ is surjective follows directly from the last property.

Now we want to show that ψ is injective. Since is is already homomorphic, it suffices to show that for any element $u = \sum_{i,j} \lambda_{ij} T_3^i T_5^j$, there exist a modular form modulo 2 f such that $u|f = \sum_{i,j} \lambda_{ij} T_3^i T_5^j |f = \Delta$. Note that this sum is finite, as Hecke operators are nilpotent.
If $\lambda_{00} = 1$, take $f = \Delta$. Suppose $\lambda_{00} = 0$: Consider $S = \{(i,j) \in \mathbb{N}^2 \mid \lambda_{ij} = 1\}$.
Let $S_{min} = \{(m,n) \mid m+n \leq i+j \quad \forall (i,j) \in S\}$ and $(a,b) \in S_{min}$ be such that $b \leq n \quad \forall (m,n) \in S_{min}$.
Let $k = [a,b]$ (k is the odd integer with code $[a,b]$ as in 2.6.2), and let $f = \Delta^k$. Then, by the theorem of Order Of Nil-potency 2.6.4, we have $T_3^a T_5^b|f = \Delta$. Moreover, by proposition on action of T_3 and T_5 (2.6.3 and 2.6.3), we have that $T_3^i T_5^j|f = 0$ for all $(i,j) \in S$. Thus, $u|f = \Delta$.

Therefore, ψ is an isomorphism, which completes the proof. □

3.4 Table of Powers of Hecke Operators

Δ^1	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^1	0	0	$\dots$
T_3^1	0	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^3	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^3	0	0	$\dots$
T_3^1	Δ^1	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^5	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^5	Δ^1	0	$\dots$
T_3^1	0	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^7	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^7	Δ^3	0	$\dots$
T_3^1	Δ^5	Δ^1	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^9	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^9	0	0	0	$\dots$
T_3^1	Δ^3	0	0	0	$\dots$
T_3^2	Δ^1	0	0	0	$\dots$
T_3^3	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{11}	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^{11}	0	0	0	$\dots$
T_3^1	Δ^9	0	0	0	$\dots$
T_3^2	Δ^3	0	0	0	$\dots$
T_3^3	Δ^1	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{13}	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^{13}	Δ^9	0	0	$\dots$
T_3^1	Δ^7	Δ^3	0	0	$\dots$
T_3^2	Δ^5	Δ^1	0	0	$\dots$
T_3^3	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Action of Powers Hecke Operators T_3 and T_5 on Modular Forms Modulo 2 (up to Δ^{13}).

A larger table may be found in the appendix (see A.2).

[5] Just take $\phi : \mathcal{F}(n) \to \mathcal{F}(n)^{**}$ such that $f \to [F \to F(f)]$.

4 Frobenian Elements

4.1 Context

Let R be a commutative ring, M and P ideals in R. We can then prove the followings:

Theorem 6. • *M is maximal $\iff$ R/M is a field*

- *P is prime $\iff$ R/P is an integral domain*

Property 4.1. *Maximal ideals are prime.*

Proof.

$$\begin{aligned} M \text{ maximal ideal } &\iff R/M \text{ field} \\ &\implies \text{R/M Integral Domain} \iff M \text{ prime ideal} \end{aligned}$$

□

If L/K is a Galois extension, then we will denote it's Galois group by $\mathrm{Gal}(L/K)$.

———–

Let K be a number field, and $\mathcal{O}_K$ be the corresponding ring of integers. Let $\mathfrak{p}$ be a non-zero prime ideal in $\mathcal{O}_K$. Let L/K be a finite extension and again, $\mathcal{O}_L$ be the ring of integers in L.

Then we know that $\mathcal{O}_L$ is the integral closure of $\mathcal{O}_K$ in L

We have $\mathfrak{p}\mathcal{O}_L$ an ideal in $\mathcal{O}_L$. It is not a prime ideal in general, but as L/K is finite, there exists a factorization as the following:

$$\mathfrak{p}\mathcal{O}_L = \prod_{i=1}^{r} \mathfrak{P}_i^{e_i}$$

Where the integers e_i are called the ramification indexes. We also have $\mathfrak{P}_i \cap \mathcal{O}_K = \mathfrak{p}$, and we say that the ideals $\mathfrak{P}_i$ in L extend the ideal $\mathfrak{p}$ in K.

Then, there are three possibilities for an ideal: it may split, ramify of be inert.

Definition 4.1 (Ideal Ramifies)**.** *We say that an ideal $\mathfrak{p}$ ramifies in L/K if a ramification index e_i is greater then one, i.e. if $e_i > 1$ for some $1 \leq i \leq r$.*

Definition 4.2 (Ideal Splits)**.** *We say that $\mathfrak{p}$ splits in L/K if none of the ramification indexes e_i is greater then one, and r is a least two; i.e. if $e_i = 1 \quad \forall 1 \leq i \leq r$ and $r \geq 2$.*

Definition 4.3 (Ideal Inert)**.** *We say that $\mathfrak{p}$ is inert in L/K if there is only one ramification index e_1 and it is equal to one; i.e. if $e_1 = 1$ and $r = 1$.*

We know that the extension L/K is ramified in the primes that divide the discriminant. Therefore, the extension is unramified in all but finitely many prime ideals.

4.2 Residue Fields Extensions

The ideal $\mathfrak{p}$ defines the *residue field* $F = \mathcal{O}_K/\mathfrak{p}$. The ideals $\mathfrak{P}_i$ define the *residue fields* $F_i = \mathcal{O}_L/\mathfrak{P}_i$. The field F then naturally embeds to F_i (so each $\mathfrak{P}_i$ defines a field extension). The *inertia degree* of $\mathfrak{P}_i$ is the degree $f_i = [F_i : F] = [\mathcal{O}_L/\mathfrak{P}_i : \mathcal{O}_K/\mathfrak{p}]$ of this extension.

We then observe that $[L : K] = \sum_{i=1}^{r} e_i f_i$

We can then specify when an ideal splits or ramifies completely.

Definition 4.4 (Ideal Splits Completely). *We say that $\mathfrak{p}$ splits completely in L/K if all ramification indexes e_i and inertia degrees f_i are one. i.e. if $e_i = f_i = 1 \quad \forall 1 \le i \le r$.*
In this case, $r = [L : K]$.

Definition 4.5 (Ideal Ramifies Completely). *We say that $\mathfrak{p}$ ramifies completely in L/K if the inertia degrees f_1 is one, and r is one. i.e. if $r = 1$ and $f_1 = 1$.*
In this case, $e_1 = [L : K]$.

4.3 Norms of Ideals

We define the *norm of an ideal I* in $\mathcal{O}_K$ as $N(I) = |\mathcal{O}_K/I|$.

If $\mathfrak{p} \subset \mathcal{O}_K$ is a prime ideal, then we can put $(p) = \mathfrak{p} \cap \mathbb{Z}$. It follows that $p\mathcal{O}_K \subset \mathfrak{p}$. $\mathcal{O}_K$ is a free $\mathbb{Z} -$ *module* of rank $[K : \mathbb{Q}] = q$, i.e. $\exists \alpha_1, \dots, \alpha_q$ s.t. $\mathcal{O}_K = \mathbb{Z}\alpha_1 \oplus \dots \oplus \mathbb{Z}\alpha_q$. Thus, $|\mathcal{O}_K/\mathfrak{p}| \le |\mathcal{O}_K/(p)| \le p^q$.

We have $\mathrm{Norm}(\mathfrak{p}) = |\mathcal{O}_K/\mathfrak{p}| = p^m$ and $\mathrm{Norm}_{L/\mathbb{Q}}(\mathfrak{P}_i) = \mathrm{Norm}_{K/\mathbb{Q}}(\mathfrak{p})^{f_i}$. This implies $\mathrm{Norm}(\mathfrak{P}_i) = |\mathcal{O}_L/\mathfrak{P}_i| = p^{mf_i}$.
We also have: $\mathcal{O}_K/\mathfrak{p} \cong \mathbb{F}_{\mathrm{Norm}(\mathfrak{p})}$ and $\mathcal{O}_L/\mathfrak{P}_i \cong \mathbb{F}_{\mathrm{Norm}(\mathfrak{P}_i)}$.

4.4 Galois Extensions Simplifications

When the extension L/K is Galois, the ramification indexes e_i are all the same ($e_i = e$), as well as the inertia degrees $f_i = f$. We then have

$$\mathfrak{p}\mathcal{O}_L = \prod_{i=1}^{r} \mathfrak{P}_i^e \text{ and } [L : K] = ref.$$

The Galois group $\mathrm{Gal}(L/K)$ is often denoted G.

We define the *decomposition group* $G_{\mathfrak{P}}$ of the ideal $\mathfrak{P}$ to be $\{\sigma \in G | \sigma(\mathfrak{P}) = \mathfrak{P}\}$. It turns out that $G_{\mathfrak{P}} \cong \mathrm{Gal}({}^{\mathcal{O}_L/\mathfrak{P}}/_{\mathcal{O}_K/\mathfrak{p}}) \cong \mathrm{Gal}(\mathbb{F}_{p^{mf}}/\mathbb{F}_{p^f})$. Moreover, it is a cyclic group, so $G_{\mathfrak{P}} =< \tilde{\sigma} >$.

4.5 Unramified Prime Simplifications

When the ideal $\mathfrak{p}$ is unramified, $e = 1$, so we get:

$$\mathfrak{p}\mathcal{O}_L = \prod_{i=1}^{r} \mathfrak{P} \text{ and } [L : K] = rf$$

4.6 The Frobenius Element

4.6.1 Definition

We can construct the *Frobenius element* (sometimes also called the *Artin symbol*, or the *Frobenius map*) that depend on the extension L/K and ideal $\mathfrak{P}$ in $\mathcal{O}_L$. It is denoted $\mathrm{Frob}_{L/K}(\mathfrak{P})$, and is *the* element $\sigma \in G$ such that:

$$\sigma\mathfrak{P} = \mathfrak{P} \quad \text{and} \quad \sigma(\alpha) \equiv \alpha^{\mathrm{Norm}_{K/\mathbb{Q}}(\mathfrak{p})} \bmod \mathfrak{P} \quad \forall \alpha \in \mathcal{O}_L.$$

The second condition is the interesting one; while the first is only useful to make the Frobenius element unique. The second condition defines a unique element only up to conjugacy class. Most of the time, we will consider abelian extensions, so the conjugacy classes will only have one element, and the first condition will be dropped.

We define the *Frobenius element* for $\mathfrak{p}$ (denoted $\mathrm{Frob}_{L/K}(\mathfrak{p})$) in a meaning full manner, to be the set

$$\{\mathrm{Frob}_{L/K}(\mathfrak{P}) | \mathfrak{P} \text{ extending } \mathfrak{p}\} \subset G.$$

The following properties imply that $\mathrm{Frob}_{L/K}(\mathfrak{p})$ is in fact a conjugacy class in the Galois group $\mathrm{Gal}(L/K)$. Hence, we refer to $\mathrm{Frob}_{L/K}(\mathfrak{p})$ as the Frobenius conjugacy class.

Property 4.2. *If $\tau \in G$, then $Frob_{L/K}(\tau\mathfrak{P}) = \tau Frob_{L/K}(\mathfrak{P})\tau^{-1}$.*

Proof. For all $x \in \mathcal{O}_L$, we have:

$$\mathrm{Frob}_{L/K}(\mathfrak{P})x = x^{\mathrm{Norm}_{K/\mathbb{Q}}(\mathfrak{p})} \mod \mathfrak{P}$$

But all such x may be written as $\tau^{-1}(x)$, so we have:

$$\mathrm{Frob}_{L/K}(\mathfrak{P})\tau^{-1}(x) = (\tau^{-1}x)^{\mathrm{Norm}_{K/\mathbb{Q}}(\mathfrak{p})} \mod \mathfrak{P}$$

Which gives:

$$\tau\mathrm{Frob}_{L/K}(\mathfrak{P})\tau^{-1}(x) = x^{\mathrm{Norm}_{K/\mathbb{Q}}(\mathfrak{p})} \mod \mathfrak{P}$$

$\square$

Property 4.3. *If $\mathfrak{P}_1$ and $\mathfrak{P}_2$ extend $\mathfrak{p}$, then $Frob_{L/K}(\mathfrak{P}_1)$ and $Frob_{L/K}(\mathfrak{P}_2)$ are conjugates.*

Proof. We have the following scheme:

$$\begin{tikzcd} L \arrow[d, dash] & \supseteq & \mathfrak{P}_1 \arrow[dr, dash] & & \mathfrak{P}_2 \arrow[dl, dash] \\ K & \supseteq & & \mathfrak{p} & \end{tikzcd}$$

There is an element $\tau \in G$ such that $\tau(\mathfrak{P}_1) = \mathfrak{P}_2$. Then using last property, we deduce that $\mathrm{Frob}_{L/K}(\mathfrak{P}_1)$ and $\mathrm{Frob}_{L/K}(\mathfrak{P}_2)$ are conjugates. $\square$

Never the less, is important to notice at this point that if L/K is an abelian extension (i.e. G is abelian), then every conjugacy class in $\mathrm{Gal}(L/K)$ are made up of only one element. In this case, we sometimes use $\mathrm{Frob}_{L/K}(\mathfrak{p})$ to denote the Frobenius element $\mathrm{Frob}_{L/K}(\mathfrak{P})$, where $\mathfrak{P}$ is any prime lying above $\mathfrak{p}$.

4.6.2 Examples

$\mathbb{Q}[\sqrt{7}]/\mathbb{Q}$ (quadratic field extension) We have minimum polynomial $m(x) = x^2 - 7$, the discriminant is $\Delta = 4.7 = 28$.
We write

$$G = \mathrm{Gal}(\mathbb{Q}[\sqrt{7}] : \mathbb{Q}) =< \sigma \mid \sigma^2 = 1_G >\cong C_2.$$

As C_2 is abelian, we will have no problem defining Frobenius elements.

The prime ideal (3) As $m(x) = (x+1)(x-1) \mod 3$, we have $(3) = (3, \sqrt{7}+1)(3, \sqrt{7}-1)$. As well, $\mathrm{Norm}_{\mathbb{Q}[\sqrt{7}]/\mathbb{Q}}((3)) = 3$ and $\mathrm{Norm}_{\mathbb{Q}[\sqrt{7}]/\mathbb{Q}}((3, \sqrt{7}+1)) = \mathrm{Norm}_{\mathbb{Q}[\sqrt{7}]/\mathbb{Q}}((3, \sqrt{7}-1)) = 3$, but $\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((3)) = 3$. So we have:

$$\begin{aligned} \mathrm{Frob}_{\mathbb{Q}[\sqrt{7}]:\mathbb{Q}}((3, \sqrt{7}+1)) : \alpha \to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((3))} &\mod (3, \sqrt{7}+1) \\ \sqrt{7} \to \left(\sqrt{7}\right)^3 \equiv \sqrt{7} &\mod (3, \sqrt{7}+1) \end{aligned}$$

Thus, $\mathrm{Frob}_{\mathbb{Q}[\sqrt{7}]:\mathbb{Q}}((3, \sqrt{7}+1)) = 1_G \in G$. Similarly, $\mathrm{Frob}_{\mathbb{Q}[\sqrt{7}]:\mathbb{Q}}((3, \sqrt{7}-1)) = 1_G \in G$.

The prime ideal (5) As $m(x)$ has no root mod2. So $m(x)$ is irreducible mod5 and (5) is inert in $\mathbb{Q}[\sqrt{7}]$ As well, $\mathrm{Norm}_{\mathbb{Q}[\sqrt{7}]/\mathbb{Q}}((5)) = 5^2 = 25$ but $\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((5)) = 5$. So we have:

$$\mathrm{Frob}_{\mathbb{Q}[\sqrt{7}]:\mathbb{Q}}((5)) : \alpha \to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((5))} \bmod (5)$$
$$\sqrt{7} \to \left(\sqrt{7}\right)^5 \equiv -\sqrt{7} \bmod (5)$$

Thus, $\mathrm{Frob}_{\mathbb{Q}[\sqrt{7}]:\mathbb{Q}}((5)) = \sigma \in G$.

$\mathbb{Q}[\zeta_n]/\mathbb{Q}$ **(n^{th} Cyclotomic Field Extensions)** We have minimum polynomial:

$$\Phi(x) = \prod_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} \left(x - e^{2i\pi \frac{k}{n}}\right)$$

(so degree of the extension is $\varphi(n)$, where φ is Euler totient function).

Discriminant of the extension is:

$$\Delta = (-1)^{\varphi(n)/2} \frac{n^{\varphi(n)}}{\prod_{p|n} p^{\varphi(n)/(p-1)}}$$

see [Washington, 1997, Proposition 2.7].

The Galois group G consist of σ_k such that $\sigma_k(\zeta_n^i) = \zeta_n^{ik}$, with $\gcd(k,n) = 1$.) Note as well that G is abelian, so it is simple to calculate the Frobenius element. It is straightforward that G is naturally isomorphic to the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^\times$. Note that $\sigma \in G$ is determined by $\sigma(\zeta_n)$. Note as well that this group is abelian.

With $p \in \mathbb{P}$, a prime that is unramified in $\mathbb{Q}[\zeta_n]/\mathbb{Q}$, let P be an ideal lying above (p). We want to look at $\mathrm{Frob}_{\mathbb{Q}[\zeta_n]/\mathbb{Q}}(P)$:

We have:

$$\mathrm{Frob}_{\mathbb{Q}[\zeta_n]/\mathbb{Q}}(P) : \alpha \to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((p))} \bmod P$$
$$\zeta_n \to \zeta_n^p \bmod P$$

Case $\mathbb{Q}[\zeta_{10}]/\mathbb{Q}$ **(10^{th} cyclotomic field extension)** We denote by $\zeta_{10} = e^{\pi i/5}$ the 10^{th} root of unity. We have minimum polynomial $m(x) = x^4 - x^3 + x^2 - x + 1$ (so degree of the extension is 4), the discriminant is $\Delta = 5^3$.

We write $G = \mathrm{Gal}(\mathbb{Q}[\zeta_{10}] : \mathbb{Q}) =< \sigma : \zeta_{10} \to \zeta_{10}^3 \mid \sigma^4 = Id > \cong C_4$.

The prime ideal (3) As $m(x)$ has no root mod3, so (3) is inert. We have:

$$\mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]/\mathbb{Q}}((3)) : \alpha \to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((3))} \bmod (3)$$
$$\zeta_{10} \to (\zeta_{10})^3 \bmod (3)$$

Thus, $\mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]:\mathbb{Q}}((3)) = \sigma \in G$.

The prime ideal (7) As $m(x)$ has no root mod7, so (7) is inert. We have:

$$\mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]/\mathbb{Q}}((7)) : \alpha \to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((7))} \bmod (7)$$
$$\zeta_{10} \to (\zeta_{10})^7 \bmod (7)$$

Thus, $\mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]:\mathbb{Q}}((7)) = \sigma^3 \in G$.

The prime ideal (11) As $m(x) = (x-2)(x+3)(x+4)(x+4) \bmod 11$, so (11) splits. We have:

$$\begin{aligned} \mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]/\mathbb{Q}}((11)) : \alpha &\to \alpha^{\mathrm{Norm}_{\mathbb{Q}/\mathbb{Q}}((11))} \bmod (11) \\ \zeta_{10} &\to (\zeta_{10})^{11} = \zeta_{10} \bmod (11) \end{aligned}$$

Thus, $\mathrm{Frob}_{\mathbb{Q}[\zeta_{10}]:\mathbb{Q}}((11)) = \sigma^4 = Id \in G$.

4.6.3 Behaviour in Chained Extensions

We will consider the following scheme:

$$\begin{array}{ccc} \mathcal{O}_M \subset M & \supseteq & \mathfrak{P} \\ | & & | \\ \mathcal{O}_L \subset L & \supseteq & \mathfrak{p} \\ | & & | \\ \mathcal{O}_K \subset K & \supseteq & p \end{array}$$

In such a situation, we can define (for M/K Galois) $\mathrm{Frob}_{M/K}(\mathfrak{P})$, $\mathrm{Frob}_{M/K}(p)$, $\mathrm{Frob}_{M/L}(\mathfrak{P})$, $\mathrm{Frob}_{M/L}(\mathfrak{p})$. If, in addition, L/K is normal: $\mathrm{Frob}_{L/K}(\mathfrak{p})$, and $\mathrm{Frob}_{L/K}(p)$. [Janusz, 1996, p.99]

We will look at properties of these Frobenius elements (relation between each others).

Property 4.4.

$$Frob_{M/K}(\mathfrak{P})^{f(\mathfrak{P}/\mathfrak{p})} = Frob_{M/L}(\mathfrak{P})$$

[what is this $f(\mathfrak{P}/\mathfrak{p})$? is it the fields of $\mathfrak{P}$ over $\mathfrak{p}$?]

to write... [Janusz, 1996, p.99] □

Property 4.5.

$$Frob_{L/K}(\mathfrak{p}) = Frob_{M/K}(\mathfrak{P})\big|_L$$

Proof. Let $\sigma = \mathrm{Frob}_{M/K}(\mathfrak{P}) \in \mathrm{Gal}(M/K)$ so $\sigma : M \to M$ s.t. $\sigma|_K = Id$ and σ is an autotomorphism. Similarly, let $\tau = \mathrm{Frob}_{L/K}(\mathfrak{p}) \in \mathrm{Gal}(L/K)$ so $\tau : L \to L$ s.t. $\tau|_K = Id$ and τ is an autotomorphism.

As M extends L, σ being an automorphism of M makes it an automorphism of L as well. The restriction condition stays the same. □

Property 4.6.

$$Gal(L/K) \cong {}^{Gal(M/K)}\!/_{Gal(M/L)}$$

Proof. Let $\sigma \in \mathrm{Gal}(M/K)$, i.e. $\sigma : M \to M$ s.t. $\sigma|_K = Id$ and σ is an autotomorphism.

Let $\phi : \mathrm{Gal}(M/K) \to \mathrm{Gal}(L/K)$ be such that: $\phi(\sigma) = \sigma|_L$. This is well defined as an automorphism of M restricts to an automorphism of L when M extends L.

It is trivial to check that ϕ is a homomorphism.

The kernel of ϕ is clearly $\mathrm{Gal}(M/L)$.

The image of ϕ is $\mathrm{Gal}(L/K)$ as every element of $\mathrm{Gal}(L/K)$ may be extended to $\mathrm{Gal}(M/K)$.

Therefore, the property follows via the 1^{st} isomorphism theorem. □

Property 4.7. *We have:*

$$\mathfrak{p} \textit{ splits complitely in } L \iff Frob_{L/K}(\mathfrak{P}) = 1$$

[Janusz, 1996, p.100]

We will consider the following scheme:

$$\begin{tikzcd}[column sep=small]
 & L_1L_2 \supseteq \mathfrak{P} \arrow[dl, dash] \arrow[dr, dash] & \\
L_1 \supseteq \mathfrak{p}_1 \arrow[dr, dash] & & L_2 \supseteq \mathfrak{p}_2 \arrow[dl, dash] \\
 & K \supseteq \mathfrak{p} &
\end{tikzcd}$$

Property 4.8. *We have:*

$$Frob_{L_1L_2/K}(\mathfrak{P}) = Frob_{L_1/K}(\mathfrak{p}_1) \times Frob_{L_2/K}(\mathfrak{p}_2)$$

[Janusz, 1996, p.100]

Property 4.9. *We have:*

$$\mathfrak{p} \text{ splits complitely in } L_1L_2 \iff \mathfrak{p} \text{ splits complitely in } L_1 \text{ and } L_2$$

Proof. Combine the last two proposition. [Janusz, 1996, p.100] □

4.7 The Chebotarev's Density Theorem

4.7.1 Motivations

If we look at the distribution of primes numbers modulo a number (15 in the next example), we get a table as follows:

Table mod 15:

mod15	primes (up to 500)
0	
1	31, 61, 151, 181, 211, 241, 271, 331, 421,
2	2, 17, 47, 107, 137, 167, 197, 227, 257, 317, 347, 467,
3	3,
4	19, 79, 109, 139, 199, 229, 349, 379, 409, 439, 499,
5	5,
6	
7	7, 37, 67, 97, 127, 157, 277, 307, 337, 367, 397, 457, 487,
8	23, 53, 83, 113, 173, 233, 263, 293, 353, 383, 443,
9	
10	
11	11, 41, 71, 101, 131, 191, 251, 281, 311, 401, 431, 461, 491,
12	
13	13, 43, 73, 103, 163, 193, 223, 283, 313, 373, 433, 463,
14	29, 59, 89, 149, 179, 239, 269, 359, 389, 419, 449, 479,

It looks like there are classes of primes. We would like to characterize this repartition: that is, decide if classes are finite or infinite, and quantify the repartitions.

4.7.2 Notions of Density

As discussed previously, we are interested in subsets of $\mathbb{P}$ (the set of primes numbers). Euler proved that there are infinitely many primes. Therefore, there are two types of subsets of $\mathbb{P}$: the ones that are infinite, and the finites ones. For finite sets, we can characterise the size by just counting elements. In fact, we will mainly be interested in sets that have infinitely many primes, and again, we would like a notion of size.

A suitable way would be to compare the subset with the set of all primes, and, say look at the proportions of primes included in the subset.

We call this the density, there are two rigorous ways to define it:

Definition 4.6 (Natural density). *We say that $S \subseteq \mathbb{P}$ has natural density δ when:*

$$\lim_{x \to +\infty} \frac{\#\{p \in \mathbb{P} | p \in S\}}{\#\{p \in \mathbb{P} | p \in \mathbb{P}\}} = \delta$$

Definition 4.7 (Analytic density or Dirichlet density). *We say that $S \subseteq \mathbb{P}$ has analytical (or Dirichlet) density δ when:*

$$\lim_{s \to 1^+} \left(\sum_{p \in S} \frac{1}{p^s} \right) \left(\sum_{p \in \mathbb{P}} \frac{1}{p} \right)^{-1} = \delta$$

Note that the natural density may not exist. However, when both exist, the two densities are the same.

4.7.3 Statement

One of the most important results that use Frobenian maps is probably the Chebotarev density theorem.

Theorem 7 (Chebotarev Density Theorem). *With L/K an extension of Galois group $G = Gal(L/K)$. Let C be a conjugacy class in G.*

Then, the proportion of unramified primes ideals $\mathfrak{p}$ in K that have Frobenius element $Frob_{L/K}(\mathfrak{p}) = C$ [6] *is $|C|/|G|$.*

We see that Frobenius elements are in the heart of this theorem. It was proved by Nikolai Chebotarev in his thesis (Tschebotareff [1926]).

4.7.4 Example

We go through an example of Chebotarev Density Theorem for an extension of order 3.

We look at $K/\mathbb{Q}$ with $K \cong \mathbb{Q}[x]/x^3-3x-1$ (i.e. the number field with defining polynomial $x^3 - 3x - 1$).

Using SageMath [7], we have: The discriminant of this extension is $3^4 = 81$, and the extension is Galois. We define $G = \mathrm{Gal}(K\mathbb{Q})$ the Galois group of the extension, and we have $G \cong C_3 = \langle \sigma \mid \sigma^2 = 1 \rangle$ since the order of the extension is 3).

Then, an unramified prime in $\mathbb{Q}$ may remain irreducible in $K/\mathbb{Q}$, split in $K/\mathbb{Q}$. If p splits in $K\mathbb{Q}$, then $\mathrm{Frob}_{K/\mathbb{Q}}(p) = 1$ (the identity of the Galois group G). If p remains inert in $K\mathbb{Q}$, then $\mathrm{Frob}_{K/\mathbb{Q}}(p) = \sigma$ or σ^2.

As the discriminant is finite, there are finitely many primes that ramifies. Applying the Chebotarev Density theorem: one third of the primes will split in this extension, and two third will remain inert.

[6] When depending on a prime in the "lower" field, the Frobenius element is a conjugacy class to be well defined.

[7] See the code in appendix, D

4.7.5 Special Case

Here, we want to apply Chebotarev theorem in the case of a quadratic field extension. We are looking at the field extension $L/K = \mathbb{Q}[\sqrt{d}]/\mathbb{Q}$ for $d \in \mathbb{Z}$ a square-free integer. Denote by $G = \mathrm{Gal}(\mathbb{Q}[\sqrt{d}]/\mathbb{Q}) \cong C_2$ the Galois group of this extension. This group is abelian (so all conjugacy classes are made of a single element), and for any conjugacy class C, $|C|/|G| = 1/2$.

Now, for a prime p unramified, we want to calculate the Frobenius element. If p is unramified, either $p\mathcal{O}_{\mathbb{Q}[\sqrt{d}]} = R_1 R_2$ or $p\mathcal{O}_{\mathbb{Q}[\sqrt{d}]} = R$.

In the first case, we have $\left(\frac{d}{p}\right) = 1$ (i.e. $\sqrt{d} \in \mathbb{F}_p$, so d is a square modulo p). In this case, $\sqrt{d}^p \equiv \sqrt{d} \bmod p$ so $\mathrm{Frob}_{\mathbb{Q}[\sqrt{d}]}(p) = \left\{Id : \sqrt{d} \to \sqrt{d}\right\} \in G$.

In the second case, $\left(\frac{d}{p}\right) = -1$ (i.e. $\sqrt{d} \notin \mathbb{F}_p$, so d is not a square modulo p). In this case, $\sqrt{d}^p \not\equiv \sqrt{d} \bmod p$ as there is no other choice, $\mathrm{Frob}_{\mathbb{Q}[\sqrt{d}]}(p) = \left\{\sigma : \sqrt{d} \to -\sqrt{d}\right\} \in G$.

Then by Chebotarev's density theorem, we have that the density of primes p such that $\left(\frac{d}{p}\right) = \pm 1$ is $1/2$ in both cases.

Therefore, we have the following summary:

Primes $p \in \mathbb{P}$ such that:	Density:
$\left(\frac{d}{p}\right) = +1$	$1/2$
$\left(\frac{d}{p}\right) = 0$	0
$\left(\frac{d}{p}\right) = -1$	$1/2$

Thus, for a square free d, $\left(\frac{d}{p}\right)$ is as often $+1$ as -1 (for a prime p), and $\left(\frac{d}{p}\right) = 0$ happens only finitely many times.

4.8 The Dirichlet's Density Theorem

4.8.1 Statement

The most common application of Chebotarev density theorem is probably the Dirichlet's density theorem.

Theorem 8 (Dirichlet's Density Theorem). *Let $n \in \mathbb{N}^*$, $a \in \mathbb{N}$ such that $\gcd(a,n) = 1$. If $S = \{p \in \mathbb{P} \mid p \equiv a \mod n\}$, then S has density $1/\varphi(n)$.*

4.8.2 Link with Chebotarev

This is a direct application of Chebotarev's density theorem for the field extension $\mathbb{Q}[\zeta] : \mathbb{Q}$ where ζ is the n^{th} root of unity (this is the cyclotomic field).

The Galois group is abelian (it is precisely $G = \mathbb{Z}_n^\times$ and has order $\varphi(n)$). The abelian property implies that all conjugacy classes are made of a single element. Thus, for any conjugacy class C, the fraction $|C|/|G|$ is just $1/\varphi(n)$. Primes ideals in $\mathbb{Q}$ are just primes numbers. Therefore, Chebotarev gives Dirichlet's density theorem in the particular case of cyclotomic extensions.

4.8.3 Example

Here, look at the example of Dirichlet theorem in the case $n = 15$ from the motivation subsection above (see 4.7.1).

We apply the last theorem in the case of $n = 15$: $\varphi(15) = 8$. We define $S_k = \{p \in \mathbb{P} \mid p \equiv k \bmod 15\}$. By Dirichlet density theorem, the density of S_k is $1/8$ if k and 15 are co-prime (i.e. if

$k = 1, 2, 4, 7, 8, 11, 13, 14$), otherwise (if $k = 0, 3, 5, 6, 9, 10, 12$) it is 0. This is what we could conjecture from the observations.

5 Frobenian Maps and Governing Fields

5.1 Frobenian Maps

Class functions Let G be a group, Ω a set, and $f : G \to \Omega$. We say that f is a *class function* (of G) if f is constant on conjugacy classes of G. That is, if f remains unchanged under conjugation map of G.

S-Frobenian Maps This definition is taken from [Serre, 2012, §3.3]. Let K be a number field. Let P be the set of primes ideals in K. Let $S \subseteq P$ be a subset of primes ideal of K. We say that a function $f : P \backslash S \to \Omega$ is S-Frobenian if there exists an M, extending K and a class function $\phi : \mathrm{Gal}(M/K) \to \Omega$ such that $f = \phi \circ \mathrm{Frob}_{M/K}()$, i.e. $f(\mathfrak{p}) = \phi \circ \mathrm{Frob}_{M/K}(\mathfrak{p}) \quad \forall \mathfrak{p} \in P$ [8].

Frobenian Maps With the same setting as above, $f : P \setminus S \to \Omega$ is Frobenian if there exists a finite set $S \subset P$ such that f is S-Frobenian.
In general, we will take S to be the set of ramified primes (there are finitely many, since they divide the Discriminant, which is finite).

In the case of $K = \mathbb{Q}$, the set of primes ideals becomes just the set of primes $\mathbb{P}$. And a map $f : \mathbb{P} \to \Omega$ is said to be *Frobenian* if there exists a field extension $M/\mathbb{Q}$ and a class function $\phi : \mathrm{Gal}(M/\mathbb{Q}) \to \Omega$ such that for all but finitely many (all unramified) primes $p \in \mathbb{P}$, we have $f(p) = \phi(\mathrm{Frob}_{M/\mathbb{Q}}(p))$.

$a_{ij}(p)$ Frobenian We recall that for all p odd prime,

$$T_p = \sum_{i,j \geq 0} a_{ij}(p) T_3^i T_5^j \qquad \text{with } a_{ij}(p) \in \mathbb{F}_2.$$

Theorem 9. *The map $p \to a_{ij}(p)$ is Frobenian (for all $i, j \geq 0$).*
That is, for all $i, j \geq 0$, there exists an extension $M_{ij}/\mathbb{Q}$ and a class function $\phi_{ij} : Gal(M_{ij}/\mathbb{Q}) \to \mathbb{F}_2$ such that $a_{ij}(p) = \phi_{ij}(Frob_{M_{ij}/\mathbb{Q}}(p))$ for all $p \in \mathbb{P}$ unramified in $M_{ij}/\mathbb{Q}$.

[here, what paper do I cite? Bellaiche's "une représentation Galoisienne universelle attachée aux formes modulaires modulo 2?] In such a configuration, M_{ij} are called *governing fields*.

5.2 Governing Fields

s It is nice to know that the maps $p \to a_{ij}(p)$ are Frobenius, but to compute $a_{ij}(p)$, we need to know explicitly the governing field (which will depend on i and j).

5.2.1 Basics

Notations We denote M_{ij} the first governing field of the map $p \to a_{ij}(p)$ From the theorem above, we know that such a field exist, the first means that we take the one with smallest degree extension. [does this definition make sense?]

[8]Note that $\phi(\mathrm{Frob}_{M/K}(\mathfrak{p}))$ is well defined since ϕ is a class function, and $\mathrm{Frob}_{M/K}(\mathfrak{p})$ is a conjugacy class of $\mathrm{Gal}(M/K)$.

Properties Using properties of Frobenius elements, we have that if L extends M_{ij}, then L will also be a governing fields for the map $p \to a_{ij}(p)$. [other properties following from Frobenius properties?]

We also have:

- $T_p \in \mathbb{F}_2[[x^2, y^2]]$ if $p \equiv 1 \bmod 8$
- $T_p \in x.\mathbb{F}_2[[x^2, y^2]]$ if $p \equiv 3 \bmod 8$
- $T_p \in y.\mathbb{F}_2[[x^2, y^2]]$ if $p \equiv 5 \bmod 8$
- $T_p \in xy.\mathbb{F}_2[[x^2, y^2]]$ if $p \equiv 7 \bmod 8$

[Nicolas and Serre, 2012b, §7] [proof??]

Examples [include from comp + link to table for more]

5.2.2 Known Governing Fields

For i,j small, we can compute explicitly the maps a_{ij}.

Known $a_{ij}(p)$ We have:

- $a_{10}(p) = 1 \equiv p \equiv 3 \bmod 8$
- $a_{01}(p) = 1 \equiv p \equiv 5 \bmod 8$
- $a_{11}(p) = 1 \equiv p \equiv 7 \bmod 8$
- $a_{20}(p) = 1 \equiv \exists a, b \in \mathbb{Z}$ and b odd, such that $p = a^2 + 8b^2, p \equiv 3 \bmod 8$
- $a_{02}(p) = 1 \equiv \exists a, b \in \mathbb{Z}$ and b odd, such that $p = a^2 + 16b^2, p \equiv 3 \bmod 8$

[Nicolas and Serre, 2012b, §7] [proof??]

Corresponding Governing Fields We then have the following corresponding governing fields:

- $M_{10} = \mathbb{Q}(\zeta_8)$ with ζ_8 the 8^{th} root of unity
- $M_{01} = \mathbb{Q}(\zeta_8)$
- $M_{11} = \mathbb{Q}(\zeta_8, \sqrt{\zeta_8}) = \mathbb{Q}(\zeta_{16})$ with ζ_{16} the 16^{th} root of unity.
- $M_{10} = \mathbb{Q}(\zeta_8, \sqrt{1+i})$
- $M_{10} = \mathbb{Q}(\zeta_8, \sqrt[4]{2})$

[Nicolas and Serre, 2012b, §7] [proof??]

5.2.3 Research of Governing Fields

[NEW TOPIC !!!]

6 Numerics

6.1 High Performance Computations

It is important to make the program as fast as possible. Indeed, the faster the program goes, the more data it will generate (within the same amount of time). This data will be used for numerical analysis and we will also use it for interpretation. Therefore, with more data, we have more knowledge, and we can make smarter guesses.

There are two main ways to make a program faster: use a better algorithm, or use a faster implementation. A better algorithm means, for example, test factors only up to square root (in the case of primality a test). A better implementation simply means optimisation inside the computer (i.e. on operations that are made, types that are used...). We will try to optimise both.

6.1.1 Algorithm Optimisation

We can optimize an algorithm by optimizing (decreasing) the number of operations, or by using mathematical scheme (usually cancellations).

Optimize instructions Optimizing instructions usually comes through optimizing loops (stopping loops as soon as possible, avoiding extra loops...). For example, the following two algorithms create the same list of coefficients for the q-series of Δ.

Algorithm 1:

Require: $L \geq 1$
  $f \leftarrow zeros(L)$ ▷ Empty list of length L
  n=0
  while $n < L$ **do**
    if $(\sqrt{n} - 1)\%2 = 0$ **then**
      $f[n] = 1$
    end if
  end while

Algorithm 2:

Require: $L \geq 1$
  $f \leftarrow zeros(L)$ ▷ Empty list of length L
  $id = 1$
  $i = 1$
  while $id < L$ **do**
    $f[id] = 1$
    $i+ = 2$
    $id = i^2$
  end while

However, the second algorithm is significantly more efficient: the loop is faster as it only goes through odd squares instead of all numbers, and it has no condition to check. The algorithms may be *harder* to understand, but it in fact is *better* (in terms of performance).

Mathematical ruse As we are working modulo 2, there are obviously many cancellations, which will make the calculations *faster*. It is an opportunity we shouldn't miss to make the algorithms *stronger*.

6.1.2 Implementation Approach

As explained above, investigations on which tool will be the more suitable for the computations is an important part. Of course, the best would be to find a programming language that can already deal with modular forms modulo two. Unfortunately, this (yet) doesn't exist. There are packages that have modular forms implemented, but none with modular forms modulo two specifically. The goal of looking at modulo two is to conclude more than what we know in general. So using what has already been done in general to make computations modulo two won't give any thing interesting.

We realize that there is no other way than just creating a package for modular forms modulo two on our own. In fact, this is what we will do later, but before, we want to determine the tools to build this package. Modular forms modulo 2 come from maths, so it makes sense to use a high level programming language. For scientific computing nowadays, there are two main open source languages: Python and Julia. Each having various packages to work with.

We will test a selection of major ones.

6.1.3 Choice of Implementation

Now it is time to wonder how to represent modular forms modulo 2. We have seen above that a modular form modulo 2 in fact have two representations: one as an infinite q-series, and one as a finite Δ-polynomial. As we want (later on) to compute Hecke operators of these forms, we will need, at some point to use the q-series representation. In fact, this will be one of the crucial points, since it is an infinite series. The way we represent infinite objects in computers, which have only a finite amount of components (memory addresses, say), is to only store informations up to a cutting point. This is equivalent (somewhat) to the asymptotic notation in mathematics. In the case of q-series of modular forms, we will store only the few first hundred/thousand/million coefficients.

This means that we will represent a modular form via its q-series, witch will be stored as a list. We investigate the best ways (timewise) to do basic operations to decide what technology to use. The operations tested are creating the q-series of Δ, and squaring it (both storing coefficients up to some power `LENGTH`, the length of the list used).

There are various techniques to store lists in a computed, the main ones are continuous list, linked list, and sparse list. Continuous and linked lists can be aggregated as dense lists.

Dense Technique Dense storage means that we store each values of the list (next to each other, or with a link to the next). No element of the list is skipped. There are various ways to implement this technique:

Pure Python Using the Python language, this is the most elementary way to go. It represents all the q-coefficients with the default linked list python object. (code in appendix B.1).

NumPy Python NumPy is the most well known scientific computing library for Python. It interfaces with C objects to provide very fast features (such as lists). (code in appendix B.2).

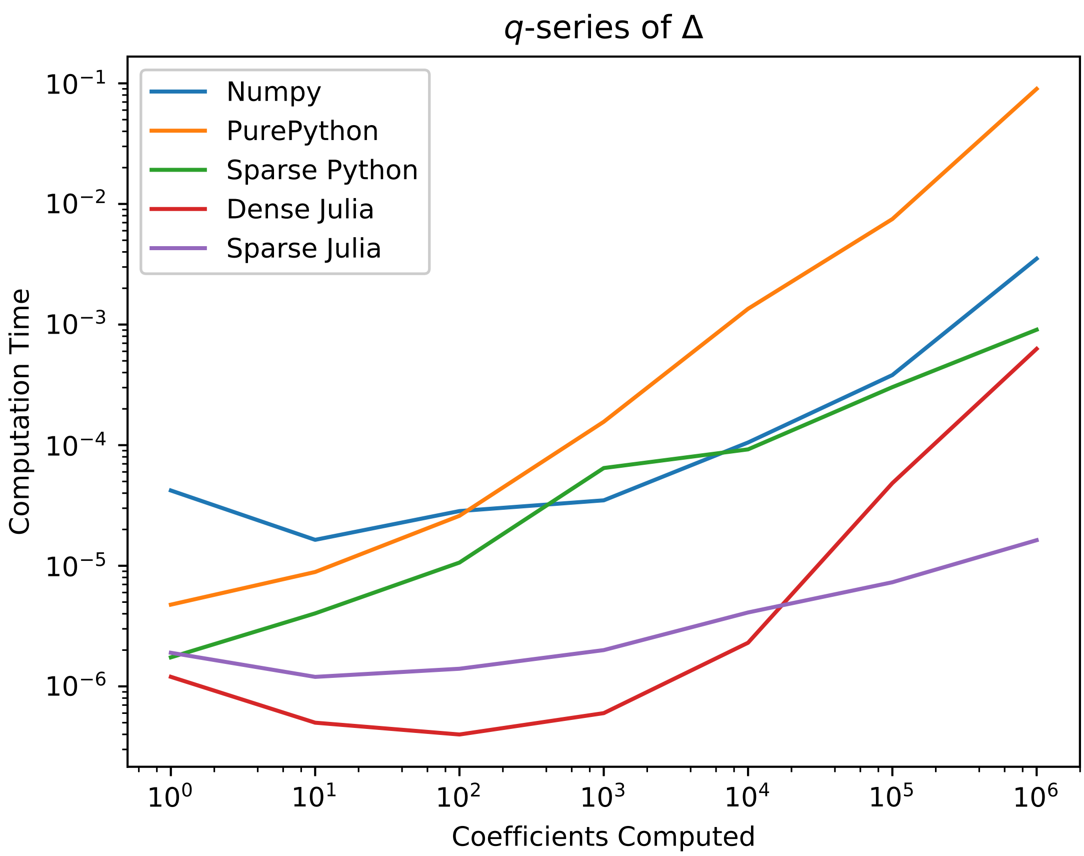
Dense Julia Julia is well-known as both high level and very fast language. Julia naturally supports lists, that we can use to represent modular forms. (code in appendix B.3).

Sparse Technique As all coefficients of the q-series are just 0 or 1, and that most of the time, they are 0, we can represent a modular forms by storing only the coefficients for which it is non-zero. This method is (storing only non-zero values) is known as sparse representation. We can implement this technique in both Python and Julia:

Sparse Python We can adapt the previous code to use Python's linked lists as index of a sparse list. Note that in general, we would need a second list to store values, but there ar only 0s and 1s, we can take as convention that all stored indices have value 1 and all non-store have value 0. (code in appendix B.4).

Sparse Julia Julia has a very convenient built-in sparse module. This is particularly interesting, since the built-in type already have nice methods. (code in appendix B.5).

Speed Comparison We can now compare the speed of each implementation to compute q-series. If we do that for various number of coefficients, we may obtain a graph of the following type (it is sightly dependent on the machine that execute the code, but the shape remains).



For small computations, the implementation doesn't make a big difference. However, for large computations, it seems that the sparse methods do better. It makes sense, since sparse representations are typically used for objects with more than 95% of zeros, which is the case for modular forms modulo 2.

For a more precise analysis, we now compare the speed of each implementation to compute q-series of Δ and Δ^2 [9]. The following table is obtained for 10^6 coefficients computed (i.e. up to q^{10^6}). Note that $\mathcal{O}(q^{10^6})$ will be standard for the rest of this paper.

[9] Δ^2 itself isn't part of our space $\mathcal{F}$, but it will be useful as we will compute $\Delta^{2k+1} = \Delta^{2k-1} \cdot \Delta^2$. So it makes sense to be concerned about it.

	Δ	Δ^2
Pure Python	0.08263147	0.26249526
NumPy Python	0.00138761	0.16163688
Dense Julia	0.000648	0.001698
Sparse Python	0.00095099	0.00134479
Sparse Julia	0.000021	0.000034

From this table, it is clear that the fastest implementation is the one using sparse lists (so called "sparse vectors") in Julia. Therefore, we will use this technique. It is nice to remark that the Pure Python implementation was 7720 times slower than the Sparse Julia one. We see here the importance of choosing the right tool to implement an algorithm.

This ratio would even be greater considering the bad algorithm presented before 6.1.1.

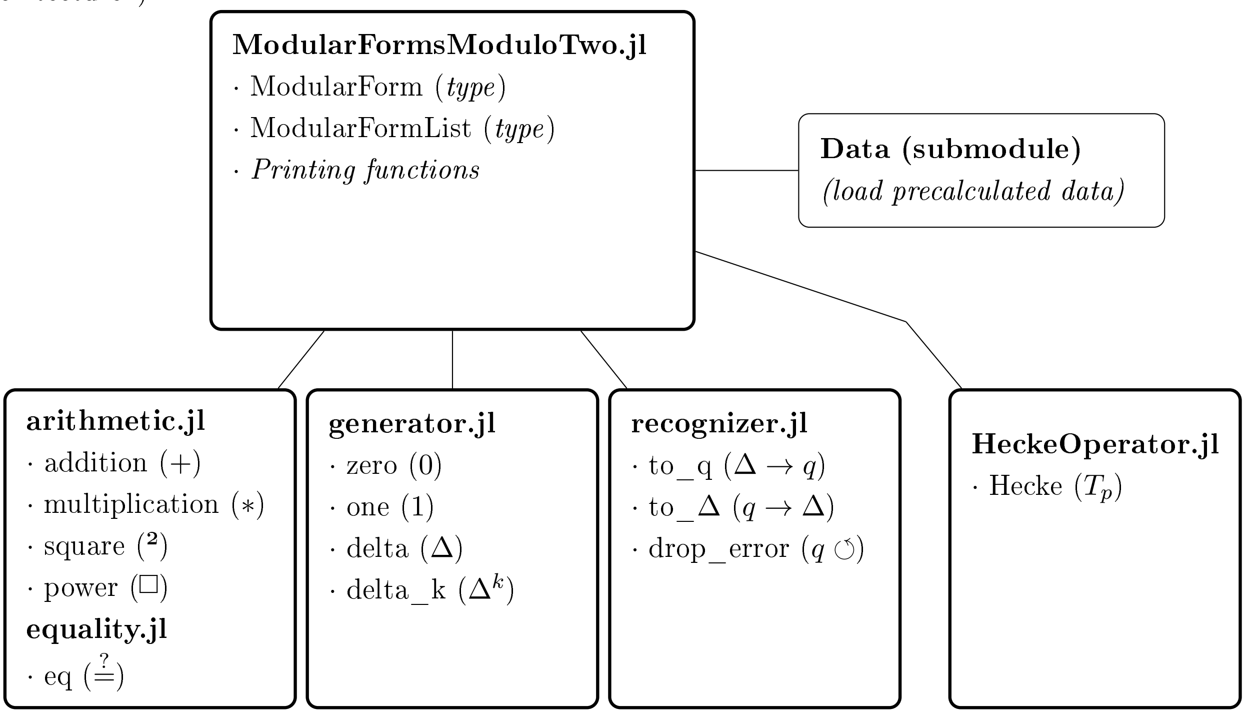
6.2 Creating the library

It is clear now that the code should be done with Julia and it's Sparse objects. Now, as all the library should be created from the beginning, it is a good idea to pack all of it in a Julia module. Doing so, no code will be repeated for each small task.

6.2.1 ModularFormsModuloTwo.jl

The code will be divided in a many files, for convenience.

Code Architecture The main function are direct parts of the module, and the pre-calculated data part is written in a subfolder, detailed next paragraph. Here is a visualization of the organisation (the "architecture"):



Files Details We will detail here the important and interesting parts of the code. For more details, the reader may refer to the source code, witch is commented and documented.

Basics Operations Basic operations on modular forms modulo 2 are defined in `arithmetic.jl` (see C.2.1 or https://github.com/paulduboisg98/ModularFormsModuloTwo.jl/blob/master/src/arithmetic.jl). All of these algorithms have been optimised as much as possible (i.e. cutting loops as soon as possible, and iterate through "ones" only, as most coefficients are zero).

Equality up to known It might be useful to compare two modular form up to some coefficients (perhaps if f_1 is known up to q^n and f_2 up to q^m, we can compare them up to $q^{\min n,m}$). This equality test is defined in `equality.jl` (see C.2.1 or https://github.com/paulduboisg98/ModularFormsModuloTwo.jl/blob/master/src/equality.jl).

Generators of $\mathcal{F}$ As modular forms modulo 2 are polynomials of Δ, we need to ba able to generate the q-series of powers of Δ. Such functions are defined in `generator.jl` [10] (see C.2.1 or https://github.com/paulduboisg98/ModularFormsModuloTwo.jl/blob/master/src/generators.jl). Here, the optimization is not trivial: To calculate Δ^{31} (say), the most trivial way is to do $\Delta \cdot \Delta \cdots \Delta$ with 30 multiplications. But what is done in fact inside the library is much faster: we calculate Δ, $\Delta^2 = \Delta \cdot \Delta$, $\Delta^4 = \Delta^2 \cdot \Delta^2$, ... until Δ^{16}, and then $\Delta^{31} = \Delta^{16} \cdot \Delta^8 \cdot \Delta^4 \cdot \Delta^2 \cdot \Delta$. Doing like this, only 8 multiplications were needed (against 30 with the naïve method). On may think that 3 was an example particularly good for this technique, but if we take an other example, say 32, we would turn 31 multiplication to 5. In fact, in general, to calculate Δ^n this second method need a maximum of $\log_2(n)^2$ multiplications against $n-1$ for the first one. So it really is a much faster algorithm.[11]

Hecke Operators The only formula we have to calculate Hecke operators, is using the q-representation of modular forms. Therefore, the Hecke operators are taking a modular forms under q-representation as input (together with a prime p). Note that when applying a Hecke operator, we loose a lot of informations on it's expansion: If f is known up to q^n, then $T_p|f$ will only be known up to $q^{n/p}$. This means that we should be careful when applying Hecke operator.

Hecke operators are implemented in the file `HeckeOperator.jl` (see C.2.1 or https://github.com/paulduboisg98/ModularFormsModuloTwo.jl/blob/master/src/HeckeOperator.jl)

Recognise the trick? This part of the module us probably the most important, and it is the one that differ the most form any other computing technique.

We start by noting that it is easy to go from the Δ representation of a modular form to the q representation: it suffices to choose an arbitrary n, and calculate all coefficients up to q^n, using the series expansion of Δ. This step is necessary to calculate Hecke operators.

Now, one may ask if it is possible to go back from the q-representation to the Δ-representation of a modular form. In general, this is not possible. **But** If we have some assumptions on the modular form, then it may become possible. For example, if we know that the maximum degree (in terms of Δ) of f is n, and that we know it's q-coefficients up to n: then f may be written as $f = \sum_{k \le n} \mu_k \Delta^k$, so the set $\{\Delta^k | k \le n\}$ acts as a basis, and it is just a matter of finding the matching coefficients μ_k. This is in fact possible, and that is how the function `to_delta()` is made possible.

Now, once we have the Δ representation of a modular form, we potentially have as many q-coefficients as we want. This may seem weird or even magical, since we assumed only finitely many q-coefficients were known. In fact, we can use this fact to drop the numerical error that calculating a Hecke operator may have produced.

[10] `generators.jl` generates Δ and then uses the power function from `arithmetic.jl`.

[11] In fact, half of the operations are squaring modular forms, wich is much faster than a usual multiplication. This this algorithm is even better than stated.

Such kind of calculations are rather unusual in computer science: usually computers approximates objects, and this approximation error is never given back. But with modular forms modulo two, we can take back the approximation error.

This method will now be referred as exact calculation. [may I create this name?]

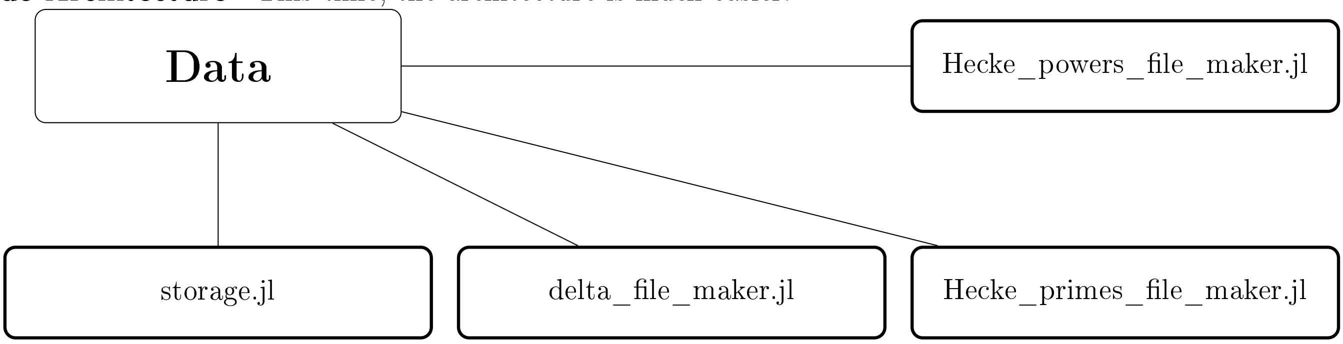
It is implemented in the file `recognizer.jl` (see C.2.1 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/recognizer.jl).

Global The last file, `ModularFormsModuloTwo.jl` is the one that creates the link between all the small part of programs written in other files. It also defines a few general objects, such as types and printing functions. For more details, see (C.2.1 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/ModularFormsModuloTwo.jl)

6.2.2 (Pre-calculated) Data

Data is natively part of the ModularFormModuloTwo.jl module, but it is treated internally as a sub-component, for better organisation.

Code Architecture This time, the architecture is much easier:



Files Details We will detail here an overview of what each file achieve. For more details, the reader may look at the source code, which is highly commented, and quite explicit in general.

- `storage.jl` (see C.2.2 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/data/storage.jl): This is the only file to be called outside of the Data submodule.
- `delta_file_maker.jl` (see C.2.2 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/data/delta_file_maker.jl): The program in this file generates the q-coefficients lists for a range of powers of Δ.
- `Hecke_primes_file_maker.jl` (see C.2.2 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/data/Hecke_primes_file_maker.jl): The program in this file calculates the Δ-representation lists for a range of Hecke operators T_p and range of powers of Δ.
- `Hecke_powers_file_maker.jl` (see C.2.2 or https://github.com/pauldubois98/ModularFormsModuloTwo.jl/blob/master/src/data/Hecke_powers_file_maker.jl): The program in this file calculates the Δ-representation lists for a range of powers of Hecke operators T_3 and T_5 and range of powers of Δ.

6.2.3 Open-Source

This library is completely open-source. Anyone is welcome to contribute. Anyone can use it (for free).

6.2.4 Official

This module is now registered as an official Julia package. To use all the code form this library, a new user will just need to type:

```
julia> using Pkg
julia> Pkg.add(PackageSpec(url="https://github.com/pauldubois98/ModularFormsModuloTwo.jl"))
```

It is convenient that all the algorithms developed in this module can be used just by importing the package, which can be done in a minute.

6.2.5 Online Documentation

As it usually comes wit open-sources packages, ModularFormsModuloTwo.jl has an online documentation (see `https://pauldubois98.github.io/ModularFormsModuloTwo.jl/`).

6.3 Finding coefficients of Hecke operators

We want to find the coefficients a_{ij} such that

$$\sum_{i,j} a_{ij} T_3^i T_5^j = T_p$$

(with $a_{ij} \in \mathbb{F}_2$).

Let $k \geq 1$ an integer. Then there exists an integer $N(k) > 0$ such that, for all pairs of non-negative integers (i, j) with $i + j \geq N(k)$, we have $T_3^i T_5^j | \Delta^k = 0$.

This allows us to write:

$$\sum_{i+j<N(k)} a_{ij} T_3^i T_5^j | \Delta^k = T_p | \Delta^k \qquad (*)$$

Now, suppose that we want to calculate the table of the $a_{ij}(p)$ for $p \in \mathbb{P}$:

1. Take an odd power for Δ (say k, we usually start with the smallest: 1 and the increase gradually)
2. Plug Δ^k in the equation above, ie:
3. Calculate $T_3^i T_5^j | \Delta^k \forall i + j < N(k)$
4. Calculate $T_p | \Delta^k \forall i + j < N(k)$
5. Equate both sides of $(*)$, if not zero (which unfortunately happens often), use the equation to deduce $a_{ij}(p)$

6.4 Finding Governing Fields

Implementation Strategy For algebraic computations (such as coputing Frobenius elements), many libraries already exist. However, many (such as SageMath) are limited in terms of degree of field extensions. This is why we will use a very powerful and respected library: PARI GP. PARI GP may be used through C, but for simplicity, we will use the the GP language that PARI GP developers suggest.

It isn't easy to connect the Julia computations for the maps a_{ij} and the PARI GP language. So we will export results from both sides in text files, and analyse it with Python. Note here that the analysis is just checking if whether or not, the field is a governing field. All the "difficult" computations are made in very efficient languages (Julia and the highly optimized PARI GP library). So it is fine, for convenience, to use a slower language as Python for the last step, which isn't computationally fast.

6.4.1 Frobenian Map Test

To test if a field extension is a governing field for a_{ij}, we need to check if the map a_{ij} is Frobenian (we take the set S to be ramified primes) with respect to this field. Here is the algorithm we use to test this:

[Algorithm to be included.]

Note that this algorithm doesn't give a proof, since it only make some tests for primes less than 10^4.

Checking Known Governing Fields [It is always good to check, to make the theory stronger (this is usually true in Science, a little bit less in math)] This will test the algorithms that we use: if the output agrees with the theory, then it is less likely to contain errors. [results: Serre is right]

Finding Possible New Governing Fields [NEW RESULTS!!!] [results: a04, a03, a30, a40] [more results? (a05, a50, ...)? => if time]

Probabilistic Analysis Here, we look for the probability that the results found appeared randomly. We assume that the Frobenius element of a prime is random for each prime. then the proba [bla]. [this part is written, but only in my head, at the moment.]

A Hecke Operators

A.1 Primes Hecke Operators

	Δ^1	Δ^3	Δ^5	Δ^7	Δ^9	Δ^{11}	Δ^{13}	Δ^{15}	Δ^{17}	Δ^{19}	Δ^{21}
T_3	0	Δ^1	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5 + \Delta^{13}$	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_5	0	0	Δ^1	Δ^3	0	0	Δ^9	$\Delta^3 + \Delta^{11}$	Δ^5	Δ^7	$\Delta^9 + \Delta^{17}$
T_7	0	0	0	Δ^1	0	0	Δ^3	Δ^9	0	Δ^5	Δ^3
T_{11}	0	Δ^1	0	Δ^5	Δ^3	$\Delta^1 + \Delta^9$	Δ^7	Δ^{13}	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{13}	0	0	Δ^1	Δ^3	0	0	$\Delta^1 + \Delta^9$	Δ^{11}	Δ^5	Δ^7	$\Delta^9 + \Delta^{17}$
T_{17}	0	0	0	0	Δ^1	Δ^3	Δ^5	Δ^7	Δ^1	0	0
T_{19}	0	Δ^1	0	Δ^5	Δ^3	$\Delta^1 + \Delta^9$	Δ^7	Δ^{13}	0	$\Delta^1 + \Delta^9 + \Delta^{17}$	Δ^7
T_{23}	0	0	0	Δ^1	0	0	Δ^3	$\Delta^1 + \Delta^9$	0	Δ^5	Δ^3
T_{29}	0	0	Δ^1	Δ^3	0	0	Δ^9	$\Delta^3 + \Delta^{11}$	Δ^5	Δ^7	$\Delta^1 + \Delta^9 + \Delta^{17}$
T_{31}	0	0	0	0	0	0	0	Δ^1	0	0	0
T_{37}	0	0	Δ^1	Δ^3	0	0	$\Delta^1 + \Delta^9$	Δ^{11}	Δ^5	Δ^7	$\Delta^9 + \Delta^{17}$
T_{41}	0	0	0	0	0	0	0	0	Δ^1	Δ^3	Δ^5
T_{43}	0	Δ^1	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5 + \Delta^{13}$	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{47}	0	0	0	0	0	0	0	Δ^1	0	0	0
T_{53}	0	0	Δ^1	Δ^3	0	0	$\Delta^1 + \Delta^9$	Δ^{11}	Δ^5	Δ^7	$\Delta^1 + \Delta^9 + \Delta^{17}$
T_{59}	0	Δ^1	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5 + \Delta^{13}$	0	$\Delta^1 + \Delta^9 + \Delta^{17}$	Δ^7
T_{61}	0	0	Δ^1	Δ^3	0	0	$\Delta^1 + \Delta^9$	Δ^{11}	Δ^5	Δ^7	$\Delta^1 + \Delta^9 + \Delta^{17}$
T_{67}	0	Δ^1	0	Δ^5	Δ^3	$\Delta^1 + \Delta^9$	Δ^7	Δ^{13}	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{71}	0	0	0	Δ^1	0	0	Δ^3	$\Delta^1 + \Delta^9$	0	Δ^5	Δ^3
T_{73}	0	0	0	0	Δ^1	Δ^3	Δ^5	Δ^7	0	Δ^3	Δ^5
T_{79}	0	0	0	0	0	0	0	0	0	0	0
T_{83}	0	Δ^1	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5 + \Delta^{13}$	0	$\Delta^1 + \Delta^9 + \Delta^{17}$	Δ^7
T_{89}	0	0	0	0	Δ^1	Δ^3	Δ^5	Δ^7	0	Δ^3	Δ^5
T_{97}	0	0	0	0	Δ^1	Δ^3	Δ^5	Δ^7	Δ^1	0	0
T_{101}	0	0	Δ^1	Δ^3	0	0	$\Delta^1 + \Delta^9$	Δ^{11}	Δ^5	Δ^7	$\Delta^9 + \Delta^{17}$
T_{103}	0	0	0	Δ^1	0	0	Δ^3	$\Delta^1 + \Delta^9$	0	Δ^5	Δ^3
T_{107}	0	Δ^1	0	Δ^5	Δ^3	$\Delta^1 + \Delta^9$	Δ^7	Δ^{13}	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{109}	0	0	Δ^1	Δ^3	0	0	Δ^9	$\Delta^3 + \Delta^{11}$	Δ^5	Δ^7	$\Delta^9 + \Delta^{17}$
T_{113}	0	0	0	0	0	0	0	0	0	0	0
T_{127}	0	0	0	0	0	0	0	Δ^1	0	0	0
T_{131}	0	Δ^1	0	Δ^5	Δ^3	$\Delta^1 + \Delta^9$	Δ^7	Δ^{13}	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{137}	0	0	0	0	0	0	0	0	Δ^1	Δ^3	Δ^5
T_{139}	0	Δ^1	0	Δ^5	Δ^3	Δ^9	Δ^7	$\Delta^5 + \Delta^{13}$	0	$\Delta^9 + \Delta^{17}$	Δ^7
T_{149}	0	0	Δ^1	Δ^3	0	0	Δ^9	$\Delta^3 + \Delta^{11}$	Δ^5	Δ^7	$\Delta^1 + \Delta^9 + \Delta^{17}$

Action of Primes Hecke Operators (primes up to 150) on Modular Forms Modulo 2 (up to Δ^{21}).

A larger table may be found online at https://pauldubois98.github.io/ModularFormsModuloTwo.jl/tables/Hecke_primes_table.html.

A.2 Powers of Hecke Operators

Δ^1	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^1	0	0	$\dots$
T_3^1	0	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^3	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^3	0	0	$\dots$
T_3^1	Δ^1	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^5	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^5	Δ^1	0	$\dots$
T_3^1	0	0	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^7	T_5^0	T_5^1	T_5^2	$\dots$
T_3^0	Δ^7	Δ^3	0	$\dots$
T_3^1	Δ^5	Δ^1	0	$\dots$
T_3^2	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^9	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^9	0	0	0	$\dots$
T_3^1	Δ^3	0	0	0	$\dots$
T_3^2	Δ^1	0	0	0	$\dots$
T_3^3	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{11}	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^{11}	0	0	0	$\dots$
T_3^1	Δ^9	0	0	0	$\dots$
T_3^2	Δ^3	0	0	0	$\dots$
T_3^3	Δ^1	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{13}	T_5^0	T_5^1	T_5^2	T_5^3	$\dots$
T_3^0	Δ^{13}	Δ^9	0	0	$\dots$
T_3^1	Δ^7	Δ^3	0	0	$\dots$
T_3^2	Δ^5	Δ^1	0	0	$\dots$
T_3^3	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{15}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{15}	$\Delta^3+\Delta^{11}$	0	0	0	$\dots$
T_3^1	$\Delta^5+\Delta^{13}$	$\Delta^1+\Delta^9$	0	0	0	$\dots$
T_3^2	Δ^7	Δ^3	0	0	0	$\dots$
T_3^3	Δ^5	Δ^1	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{17}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{17}	Δ^5	Δ^1	0	0	$\dots$
T_3^1	0	0	0	0	0	$\dots$
T_3^2	0	0	0	0	0	$\dots$
T_3^3	0	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{19}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{19}	Δ^7	Δ^3	0	0	$\dots$
T_3^1	$\Delta^9+\Delta^{17}$	Δ^5	Δ^1	0	0	$\dots$
T_3^2	Δ^3	0	0	0	0	$\dots$
T_3^3	Δ^1	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{21}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{21}	$\Delta^9+\Delta^{17}$	Δ^5	Δ^1	0	$\dots$
T_3^1	Δ^7	Δ^3	0	0	0	$\dots$
T_3^2	Δ^5	Δ^1	0	0	0	$\dots$
T_3^3	0	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{23}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{23}	$\Delta^{11}+\Delta^{19}$	Δ^7	Δ^3	0	$\dots$
T_3^1	$\Delta^{13}+\Delta^{21}$	Δ^{17}	Δ^5	Δ^1	0	$\dots$
T_3^2	0	0	0	0	0	$\dots$
T_3^3	0	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{25}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{25}	$\Delta^5+\Delta^{13}$	$\Delta^1+\Delta^9$	0	0	$\dots$
T_3^1	$\Delta^{11}+\Delta^{19}$	Δ^7	Δ^3	0	0	$\dots$
T_3^2	Δ^{17}	Δ^5	Δ^1	0	0	$\dots$
T_3^3	0	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{27}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{27}	$\Delta^7+\Delta^{15}$	Δ^{11}	0	0	$\dots$
T_3^1	$\Delta^9+\Delta^{17}+\Delta^{25}$	Δ^{13}	Δ^9	0	0	$\dots$
T_3^2	$\Delta^3+\Delta^{11}+\Delta^{19}$	Δ^7	Δ^3	0	0	$\dots$
T_3^3	$\Delta^1+\Delta^{17}$	Δ^5	Δ^1	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Δ^{29}	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	$\dots$
T_3^0	Δ^{29}	$\Delta^{17}+\Delta^{25}$	Δ^{13}	Δ^9	0	$\dots$
T_3^1	Δ^{23}	$\Delta^{11}+\Delta^{19}$	Δ^7	Δ^3	0	$\dots$
T_3^2	$\Delta^{13}+\Delta^{21}$	Δ^{17}	Δ^5	Δ^1	0	$\dots$
T_3^3	0	0	0	0	0	$\dots$
T_3^4	0	0	0	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Action of Powers Hecke Operators T_3 and T_5 on Modular Forms Modulo 2 (up to Δ^{31}).

A larger table may be found online at https://pauldubois98.github.io/ModularFormsModuloTwo.jl/tables/Hecke_powers_table.html.

A.3 Behaviour of Code of Integers

Code of Integers up to 150 Codes, as a function of integers.

k	code of k	h(k)
0 , 1	[0 , 0]	0
2 , 3	[1 , 0]	1
4 , 5	[0 , 1]	1
6 , 7	[1 , 1]	2
8 , 9	[2 , 0]	2
10 , 11	[3 , 0]	3
12 , 13	[2 , 1]	3
14 , 15	[3 , 1]	4
16 , 17	[0 , 2]	2
18 , 19	[1 , 2]	3
20 , 21	[0 , 3]	3
22 , 23	[1 , 3]	4
24 , 25	[2 , 2]	4
26 , 27	[3 , 2]	5
28 , 29	[2 , 3]	5
30 , 31	[3 , 3]	6
32 , 33	[4 , 0]	4
34 , 35	[5 , 0]	5
36 , 37	[4 , 1]	5
38 , 39	[5 , 1]	6
40 , 41	[6 , 0]	6
42 , 43	[7 , 0]	7
44 , 45	[6 , 1]	7
46 , 47	[7 , 1]	8
48 , 49	[4 , 2]	6

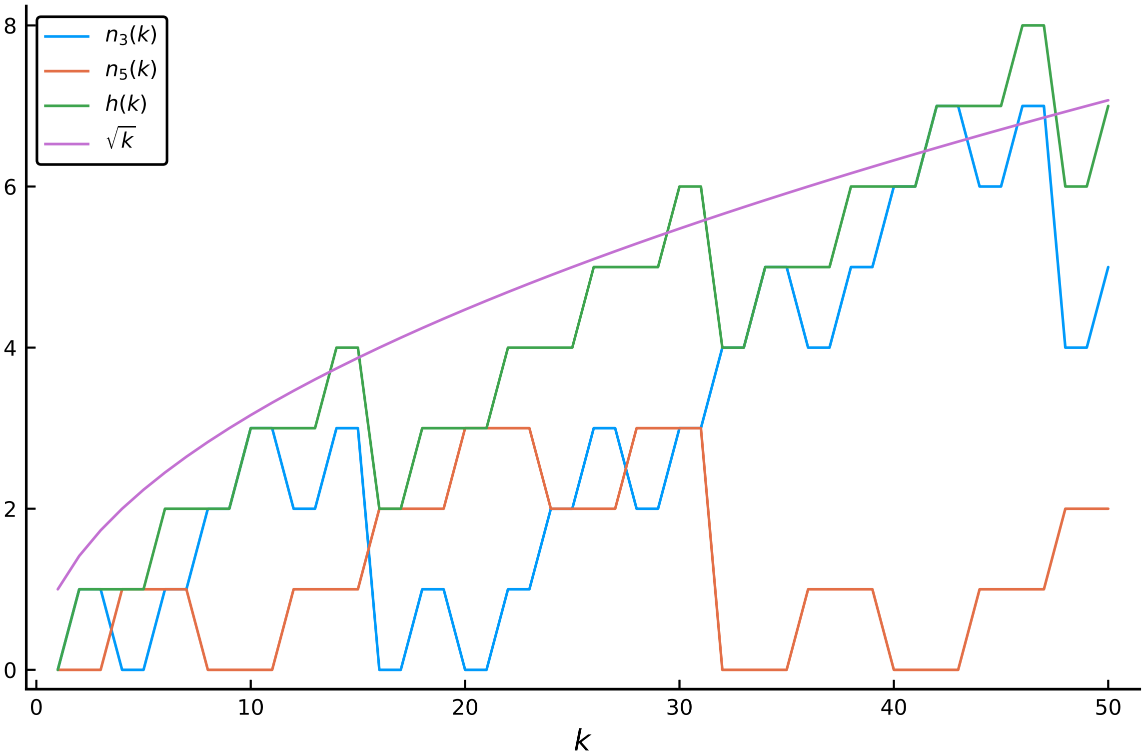
k	code of k	h(k)
50 , 51	[5 , 2]	7
52 , 53	[4 , 3]	7
54 , 55	[5 , 3]	8
56 , 57	[6 , 2]	8
58 , 59	[7 , 2]	9
60 , 61	[6 , 3]	9
62 , 63	[7 , 3]	10
64 , 65	[0 , 4]	4
66 , 67	[1 , 4]	5
68 , 69	[0 , 5]	5
70 , 71	[1 , 5]	6
72 , 73	[2 , 4]	6
74 , 75	[3 , 4]	7
76 , 77	[2 , 5]	7
78 , 79	[3 , 5]	8
80 , 81	[0 , 6]	6
82 , 83	[1 , 6]	7
84 , 85	[0 , 7]	7
86 , 87	[1 , 7]	8
88 , 89	[2 , 6]	8
90 , 91	[3 , 6]	9
92 , 93	[2 , 7]	9
94 , 95	[3 , 7]	10
96 , 97	[4 , 4]	8
98 , 99	[5 , 4]	9

k	code of k	h(k)
100 , 101	[4 , 5]	9
102 , 103	[5 , 5]	10
104 , 105	[6 , 4]	10
106 , 107	[7 , 4]	11
108 , 109	[6 , 5]	11
110 , 111	[7 , 5]	12
112 , 113	[4 , 6]	10
114 , 115	[5 , 6]	11
116 , 117	[4 , 7]	11
118 , 119	[5 , 7]	12
120 , 121	[6 , 6]	12
122 , 123	[7 , 6]	13
124 , 125	[6 , 7]	13
126 , 127	[7 , 7]	14
128 , 129	[8 , 0]	8
130 , 131	[9 , 0]	9
132 , 133	[8 , 1]	9
134 , 135	[9 , 1]	10
136 , 137	[10, 0]	10
138 , 139	[11, 0]	11
140 , 141	[10, 1]	11
142 , 143	[11, 1]	12
144 , 145	[8 , 2]	10
146 , 147	[9 , 2]	11
148 , 149	[8 , 3]	11

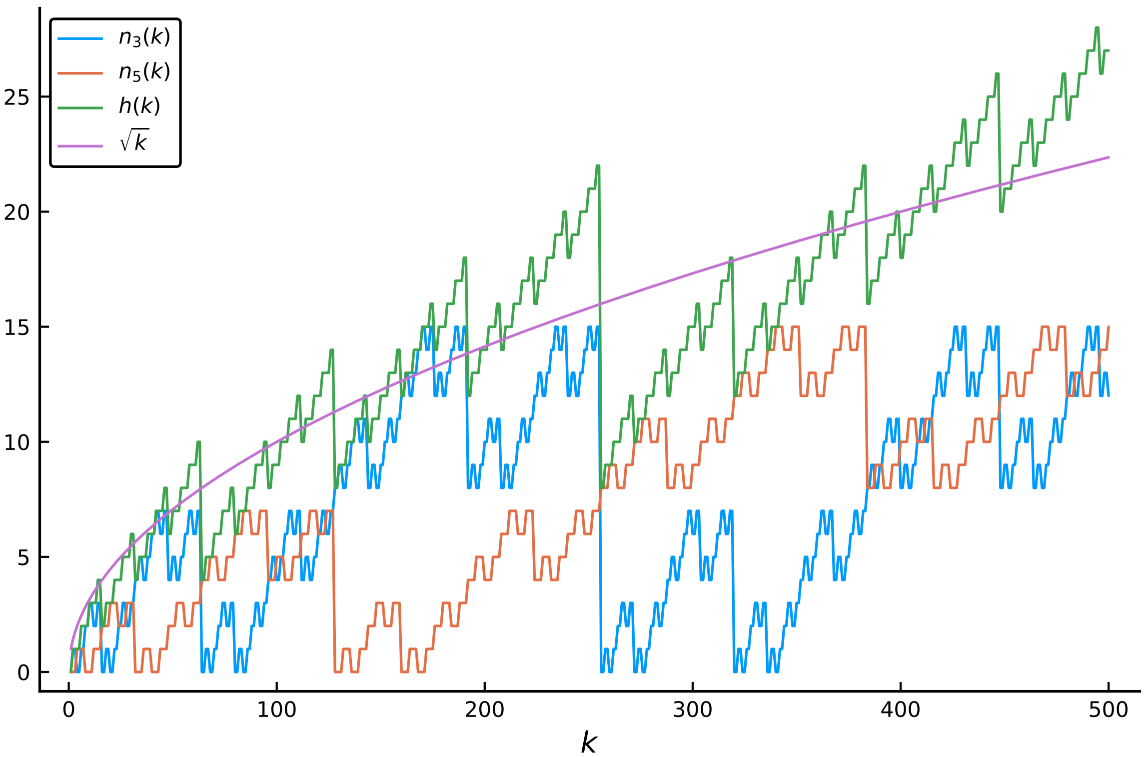
A larger table may be found online at `https://pauldubois98.github.io/HeckeOperatorsModuloTwo/int_to_code/`.

A.4 Behaviour of h on Various Scales

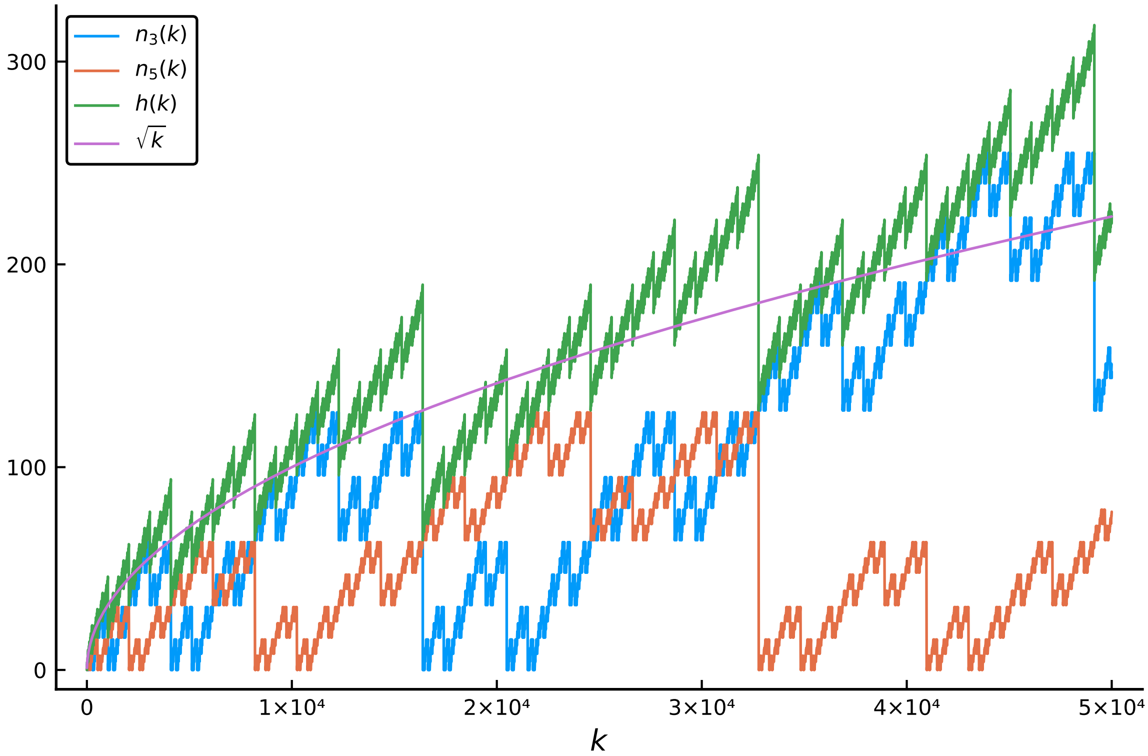
Range 0 **to** $5 * 10^1$



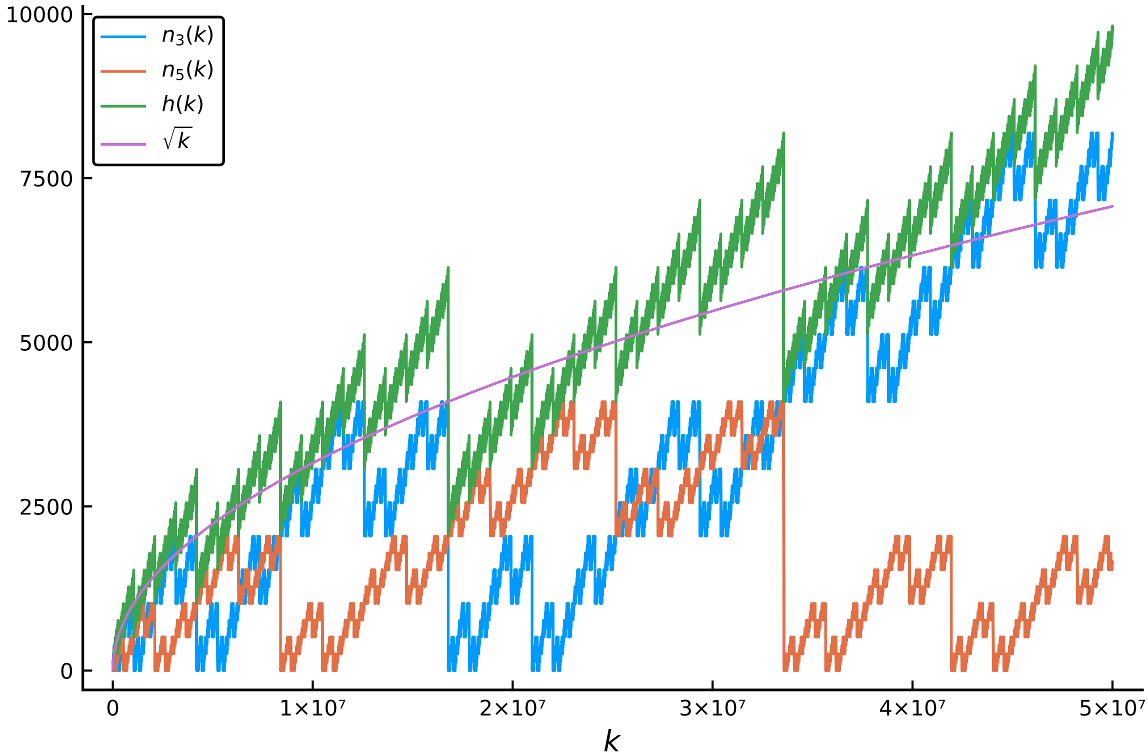
Range 0 **to** $5 * 10^2$



Range 0 **to** $5 * 10^4$



Range 0 **to** $5 * 10^7$



Other pictures (including animated ones) may be found online at `https://pauldubois98.github.io/HeckeOperatorsModuloTwo/behaviour_h/`.

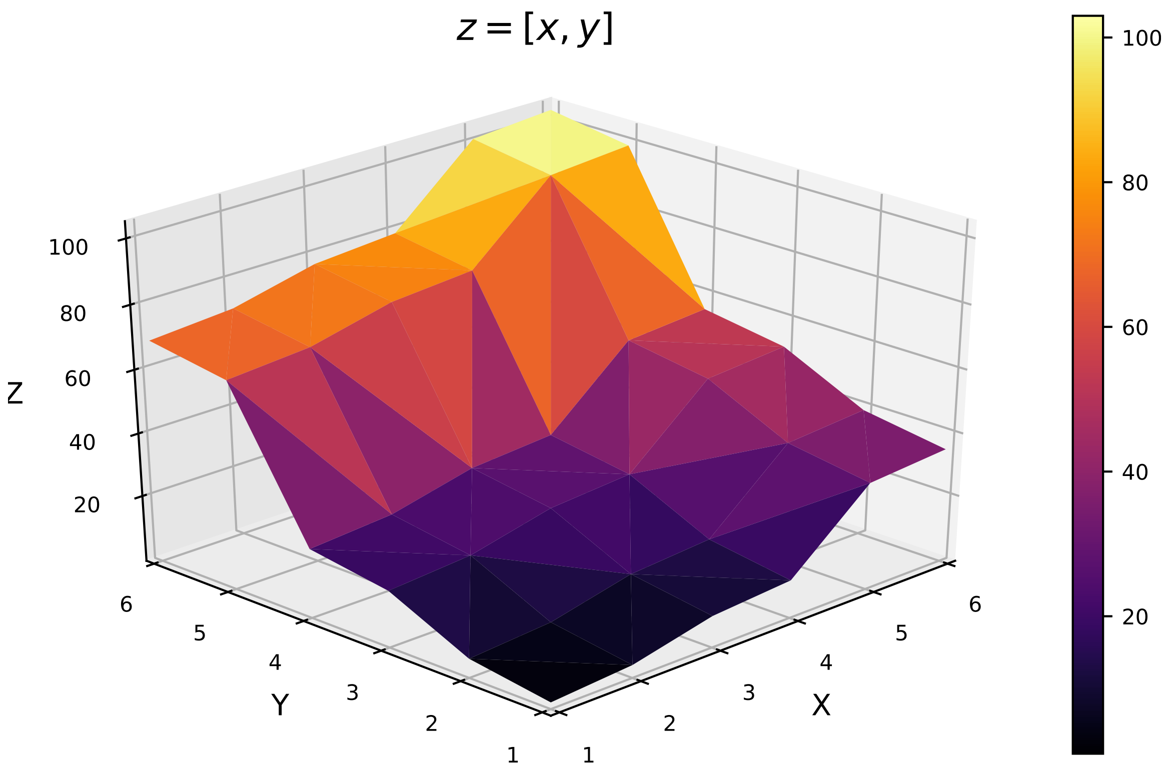
Integers with Small Code

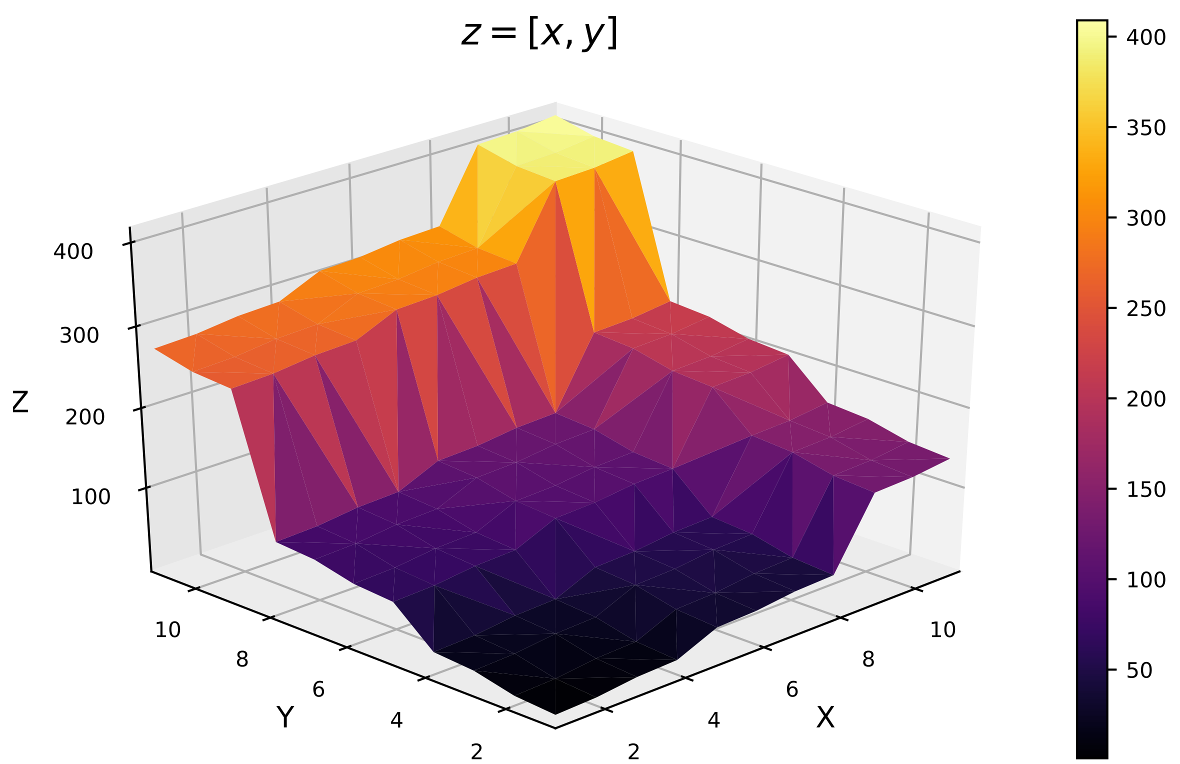
Table As the codes of integers are in bijection with even numbers (or odd numbers), we can also plot even (or odd) integers as function of their code.

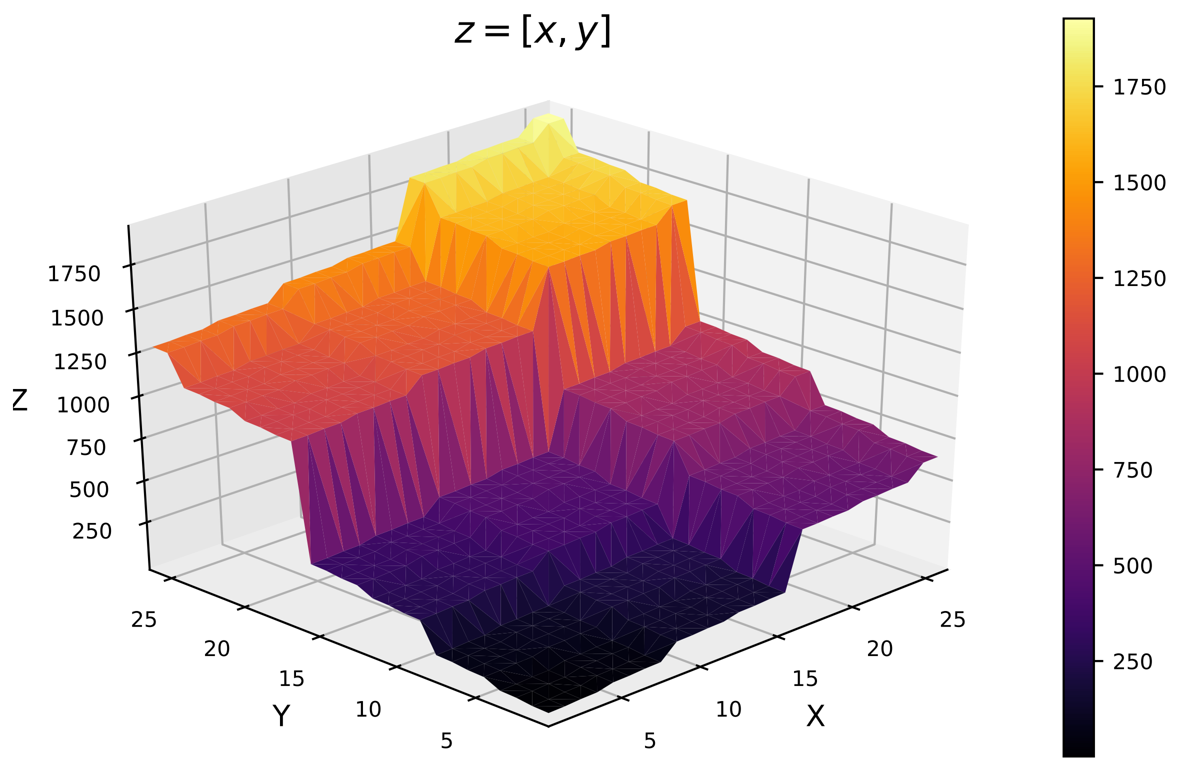
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	4	16	20	64	68	80	84	256	260	272	276	320	324	336	340
1	2	6	18	22	66	70	82	86	258	262	274	278	322	326	338	342
2	8	12	24	28	72	76	88	92	264	268	280	284	328	332	344	348
3	10	14	26	30	74	78	90	94	266	270	282	286	330	334	346	350
4	32	36	48	52	96	100	112	116	288	292	304	308	352	356	368	372
5	34	38	50	54	98	102	114	118	290	294	306	310	354	358	370	374
6	40	44	56	60	104	108	120	124	296	300	312	316	360	364	376	380
7	42	46	58	62	106	110	122	126	298	302	314	318	362	366	378	382
8	128	132	144	148	192	196	208	212	384	388	400	404	448	452	464	468
9	130	134	146	150	194	198	210	214	386	390	402	406	450	454	466	470
10	136	140	152	156	200	204	216	220	392	396	408	412	456	460	472	476

A larger table may be found online at `https://paulduboisj98.github.io/HeckeOperatorsModuloTwo/code_to_even/`. Note that the same table may be produced for odd integers, find it online at `https://paulduboisj98.github.io/HeckeOperatorsModuloTwo/code_to_odd/`.

Plots We plot the surface $z = [x, y]$ i.e. z is the integer with code $[x, y]$.







Other pictures (including animated ones) may be found online at `https://pauldubois98.github.io/HeckeOperatorsModuloTwo/plot_code_to_int/`.

A.4.1 Powers of T_3 and T_5 giving Δ^1

Here, wee look at the power of Δ^k such that when $T_3^a T_5^b$ is applied to it, we have $T_3^a T_5^b | \Delta^k = \Delta^1$.

	T_5^0	T_5^1	T_5^2	T_5^3	T_5^4	T_5^5	T_5^6	T_5^7	T_5^8	T_5^9	T_5^{10}	T_5^{11}	T_5^{12}	T_5^{13}	T_5^{14}
T_3^0	Δ^{1}	Δ^{5}	Δ^{17}	Δ^{21}	Δ^{65}	Δ^{69}	Δ^{81}	Δ^{85}	Δ^{257}	Δ^{261}	Δ^{273}	Δ^{277}	Δ^{321}	Δ^{325}	Δ^{337}
T_3^1	Δ^{3}	Δ^{7}	Δ^{19}	Δ^{23}	Δ^{67}	Δ^{71}	Δ^{83}	Δ^{87}	Δ^{259}	Δ^{263}	Δ^{275}	Δ^{279}	Δ^{323}	Δ^{327}	Δ^{339}
T_3^2	Δ^{9}	Δ^{13}	Δ^{25}	Δ^{29}	Δ^{73}	Δ^{77}	Δ^{89}	Δ^{93}	Δ^{265}	Δ^{269}	Δ^{281}	Δ^{285}	Δ^{329}	Δ^{333}	Δ^{345}
T_3^3	Δ^{11}	Δ^{15}	Δ^{27}	Δ^{31}	Δ^{75}	Δ^{79}	Δ^{91}	Δ^{95}	Δ^{267}	Δ^{271}	Δ^{283}	Δ^{287}	Δ^{331}	Δ^{335}	Δ^{347}
T_3^4	Δ^{33}	Δ^{37}	Δ^{49}	Δ^{53}	Δ^{97}	Δ^{101}	Δ^{113}	Δ^{117}	Δ^{289}	Δ^{293}	Δ^{305}	Δ^{309}	Δ^{353}	Δ^{357}	Δ^{369}
T_3^5	Δ^{35}	Δ^{39}	Δ^{51}	Δ^{55}	Δ^{99}	Δ^{103}	Δ^{115}	Δ^{119}	Δ^{291}	Δ^{295}	Δ^{307}	Δ^{311}	Δ^{355}	Δ^{359}	Δ^{371}
T_3^6	Δ^{41}	Δ^{45}	Δ^{57}	Δ^{61}	Δ^{105}	Δ^{109}	Δ^{121}	Δ^{125}	Δ^{297}	Δ^{301}	Δ^{313}	Δ^{317}	Δ^{361}	Δ^{365}	Δ^{377}
T_3^7	Δ^{43}	Δ^{47}	Δ^{59}	Δ^{63}	Δ^{107}	Δ^{111}	Δ^{123}	Δ^{127}	Δ^{299}	Δ^{303}	Δ^{315}	Δ^{319}	Δ^{363}	Δ^{367}	Δ^{379}
T_3^8	Δ^{129}	Δ^{133}	Δ^{145}	Δ^{149}	Δ^{193}	Δ^{197}	Δ^{209}	Δ^{213}	Δ^{385}	Δ^{389}	Δ^{401}	Δ^{405}	Δ^{449}	Δ^{453}	Δ^{465}
T_3^9	Δ^{131}	Δ^{135}	Δ^{147}	Δ^{151}	Δ^{195}	Δ^{199}	Δ^{211}	Δ^{215}	Δ^{387}	Δ^{391}	Δ^{403}	Δ^{407}	Δ^{451}	Δ^{455}	Δ^{467}
T_3^{10}	Δ^{137}	Δ^{141}	Δ^{153}	Δ^{157}	Δ^{201}	Δ^{205}	Δ^{217}	Δ^{221}	Δ^{393}	Δ^{397}	Δ^{409}	Δ^{413}	Δ^{457}	Δ^{461}	Δ^{473}
T_3^{11}	Δ^{139}	Δ^{143}	Δ^{155}	Δ^{159}	Δ^{203}	Δ^{207}	Δ^{219}	Δ^{223}	Δ^{395}	Δ^{399}	Δ^{411}	Δ^{415}	Δ^{459}	Δ^{463}	Δ^{475}
T_3^{12}	Δ^{161}	Δ^{165}	Δ^{177}	Δ^{181}	Δ^{225}	Δ^{229}	Δ^{241}	Δ^{245}	Δ^{417}	Δ^{421}	Δ^{433}	Δ^{437}	Δ^{481}	Δ^{485}	Δ^{497}
T_3^{13}	Δ^{163}	Δ^{167}	Δ^{179}	Δ^{183}	Δ^{227}	Δ^{231}	Δ^{243}	Δ^{247}	Δ^{419}	Δ^{423}	Δ^{435}	Δ^{439}	Δ^{483}	Δ^{487}	Δ^{499}
T_3^{14}	Δ^{169}	Δ^{173}	Δ^{185}	Δ^{189}	Δ^{233}	Δ^{237}	Δ^{249}	Δ^{253}	Δ^{425}	Δ^{429}	Δ^{441}	Δ^{445}	Δ^{489}	Δ^{493}	Δ^{505}
T_3^{15}	Δ^{171}	Δ^{175}	Δ^{187}	Δ^{191}	Δ^{235}	Δ^{239}	Δ^{251}	Δ^{255}	Δ^{427}	Δ^{431}	Δ^{443}	Δ^{447}	Δ^{491}	Δ^{495}	Δ^{507}
T_3^{16}	Δ^{513}	Δ^{517}	Δ^{529}	Δ^{533}	Δ^{577}	Δ^{581}	Δ^{593}	Δ^{597}	Δ^{769}	Δ^{773}	Δ^{785}	Δ^{789}	Δ^{833}	Δ^{837}	Δ^{849}
T_3^{17}	Δ^{515}	Δ^{519}	Δ^{531}	Δ^{535}	Δ^{579}	Δ^{583}	Δ^{595}	Δ^{599}	Δ^{771}	Δ^{775}	Δ^{787}	Δ^{791}	Δ^{835}	Δ^{839}	Δ^{851}
T_3^{18}	Δ^{521}	Δ^{525}	Δ^{537}	Δ^{541}	Δ^{585}	Δ^{589}	Δ^{601}	Δ^{605}	Δ^{777}	Δ^{781}	Δ^{793}	Δ^{797}	Δ^{841}	Δ^{845}	Δ^{857}
T_3^{19}	Δ^{523}	Δ^{527}	Δ^{539}	Δ^{543}	Δ^{587}	Δ^{591}	Δ^{603}	Δ^{607}	Δ^{779}	Δ^{783}	Δ^{795}	Δ^{799}	Δ^{843}	Δ^{847}	Δ^{859}
T_3^{20}	Δ^{545}	Δ^{549}	Δ^{561}	Δ^{565}	Δ^{609}	Δ^{613}	Δ^{625}	Δ^{629}	Δ^{801}	Δ^{805}	Δ^{817}	Δ^{821}	Δ^{865}	Δ^{869}	Δ^{881}
T_3^{21}	Δ^{547}	Δ^{551}	Δ^{563}	Δ^{567}	Δ^{611}	Δ^{615}	Δ^{627}	Δ^{631}	Δ^{803}	Δ^{807}	Δ^{819}	Δ^{823}	Δ^{867}	Δ^{871}	Δ^{883}
T_3^{22}	Δ^{553}	Δ^{557}	Δ^{569}	Δ^{573}	Δ^{617}	Δ^{621}	Δ^{633}	Δ^{637}	Δ^{809}	Δ^{813}	Δ^{825}	Δ^{829}	Δ^{873}	Δ^{877}	Δ^{889}
T_3^{23}	Δ^{555}	Δ^{559}	Δ^{571}	Δ^{575}	Δ^{619}	Δ^{623}	Δ^{635}	Δ^{639}	Δ^{811}	Δ^{815}	Δ^{827}	Δ^{831}	Δ^{875}	Δ^{879}	Δ^{891}
T_3^{24}	Δ^{641}	Δ^{645}	Δ^{657}	Δ^{661}	Δ^{705}	Δ^{709}	Δ^{721}	Δ^{725}	Δ^{897}	Δ^{901}	Δ^{913}	Δ^{917}	Δ^{961}	Δ^{965}	Δ^{977}
T_3^{25}	Δ^{643}	Δ^{647}	Δ^{659}	Δ^{663}	Δ^{707}	Δ^{711}	Δ^{723}	Δ^{727}	Δ^{899}	Δ^{903}	Δ^{915}	Δ^{919}	Δ^{963}	Δ^{967}	Δ^{979}
T_3^{26}	Δ^{649}	Δ^{653}	Δ^{665}	Δ^{669}	Δ^{713}	Δ^{717}	Δ^{729}	Δ^{733}	Δ^{905}	Δ^{909}	Δ^{921}	Δ^{925}	Δ^{969}	Δ^{973}	Δ^{985}
T_3^{27}	Δ^{651}	Δ^{655}	Δ^{667}	Δ^{671}	Δ^{715}	Δ^{719}	Δ^{731}	Δ^{735}	Δ^{907}	Δ^{911}	Δ^{923}	Δ^{927}	Δ^{971}	Δ^{975}	Δ^{987}
T_3^{28}	Δ^{673}	Δ^{677}	Δ^{689}	Δ^{693}	Δ^{737}	Δ^{741}	Δ^{753}	Δ^{757}	Δ^{929}	Δ^{933}	Δ^{945}	Δ^{949}	Δ^{993}	Δ^{997}	Δ^{1009}
T_3^{29}	Δ^{675}	Δ^{679}	Δ^{691}	Δ^{695}	Δ^{739}	Δ^{743}	Δ^{755}	Δ^{759}	Δ^{931}	Δ^{935}	Δ^{947}	Δ^{951}	Δ^{995}	Δ^{999}	Δ^{1011}
T_3^{30}	Δ^{681}	Δ^{685}	Δ^{697}	Δ^{701}	Δ^{745}	Δ^{749}	Δ^{761}	Δ^{765}	Δ^{937}	Δ^{941}	Δ^{953}	Δ^{957}	Δ^{1001}	Δ^{1005}	Δ^{1017}

A larger table may be found online at `https://pauldubois98.github.io/HeckeOperatorsModuloTwo/T3T5_powers_to_delta/`.

B Speed Comparison

B.1 Pure Python

```
def delta(LENGTH):
    f=[0 for i in range(LENGTH)]
    indice=1
    i=1
    while indice<LENGTH:
        f[indice] = 1
        i+=2
        indice = i**2
    return f

def square(f):
    f_sq = [0 for i in range(len(f))]
    i = 0
    while 2*i < len(f):
        if f[i]:
            f_sq[2*i] = 1
        i += 1
    return f_sq
```

B.2 NumPy Python

```
import numpy as np

def delta(LENGTH):
    f=np.zeros(LENGTH, dtype=np.int8)
    indice=1
    i=1
    while indice<LENGTH:
        f[indice] = 1
        i+=2
        indice = i**2
    return f

def square(f):
    f_sq=np.zeros(len(f), dtype=np.int8)
    i = 0
    while 2*i < len(f):
        if f[i]:
            f_sq[2*i] = 1
        i += 1
    return f_sq
```

B.3 Dense Julia

```
function delta(LENGTH)
    f = zeros(Int8, LENGTH)
    indice = 2
    i = 1
    while indice < LENGTH
        f[indice] = 1
        i += 2
        indice = i^2 + 1
    end
    return f
end

function square(f)
    f_sq = zeros(Int8, length(f))
    i = 1
    while 2 * i - 1 < length(f)
        if f[i] == 1
            f_sq[2 * i - 1] = 1
        end
        i += 1
    end
    return f_sq
end
```

B.4 Sparse Python

```
def delta(LENGTH):
    f = []
    indice = 1
    i = 1
    while indice < LENGTH:
        f.append(indice)
        i += 2
        indice = i**2
    return (f, LENGTH)

def square(form):
    f_sq = []
    f = form[0]
    for n in f:
        if 2*n-1 <= form[1]:
            f_sq.append(2*n-1)
    return (f_sq, form[1])
```

B.5 Sparse Julia

```
using SparseArrays: SparseVector, spzeros

function delta(LENGTH)
    f = spzeros(Int8, LENGTH)
    indice::Int = 2
    i::Int = 1
    while indice <= f.n
        f[indice] = Int8(1)
        i += 2
        indice = i^2 + 1
    end
    return f
end

function square(f)
    f_sq = spzeros(Int8, f.n)
    for n in f.nzind
        if 2n - 1 <= f_sq.n
            f_sq[2n - 1] = 1
        end
    end
    return f_sq
end
```

C ModularFormsModuloTwo.jl

C.1 Module structure

```
ModularFormsModuloTwo.jl (module)
├── docs
│   └── (automatic documentation)
├── examples
│   ├── Hecke_primes_table.jl
│   └── Hecke_powers_table.j
└── src
    ├── data
    │   ├── storage.jl
    │   ├── delta_file_maker.jl
    │   ├── Hecke_primes_file_maker.jl
    │   ├── Hecke_powers_file_maker.jl
    │   └── data files (.JLD2)
    ├── arithmetic.jl
    ├── equality.jl
    ├── generators.jl
    └── HeckeOperator.jl
```

```
│
├── ModularFormsModuloTwo.jl
└── recognizer.jl
```

Full module on GitHub (https://github.com/pauldubois98/ModularFormsModuloTwo.jl). The official documentation (https://pauldubois98.github.io/ModularFormsModuloTwo.jl/) has more details on how to use.

C.2 Files details

C.2.1 Main Sources

ModularFormsModuloTwo.jl

```
module ModularFormsModuloTwo
    """
    A standard module for computations on modular forms modulo two.
    """

    using SparseArrays: SparseVector, spzeros, dropzeros!, sparse
    import Base: +, *, ^

    """
    We can represent a modular forms mod 2 by it's coefficients as a polynomial in q
↪   or D.
    The routines in this file are made for q-series.
    Modular forms modulo 2 have coefficients in q-series being 0 most of the times,
↪   and 1 otherwise.
    Thus, we will represent them as sparse 1-dimensional arrays (sparse vectors) of
↪   type SparseVector{Int8,Int}.
    """
    ModularForm = SparseVector{Int8,Int}
    ModularFormOrNothing = Union{ModularForm, Nothing}
    ModularFormList = Array{SparseVector{Int8,Int}, 1}
    """
    Lists of Modular Forms will be useful for storage.
    """
    ModularFormOrNothingList = Array{ModularFormOrNothing, 1}


    """
    disp(f[, maxi])

    Display details of f, a modular forms mod 2.

    Displays what type of data the object id,  up to which coefficient is the form
↪   known.
    Then displays the first few coefficients. Coefficients are displayed until maxi
↪   (50 by default).
```

```julia
# Example
```julia-repl
[f is a modular form mod 2]
julia> disp(f)
MF mod 2 (coef to 100) - 0100000001000000000000000100000000000000000000001...
```
"""
function disp(f::ModularForm, maxi::Int = 50)
    print("MF mod 2 (coef to " * string(f.n) * ") - ")
    for i = 1:min(maxi, f.n)
        print(f[i])
    end
    if maxi<f.n
        println("...")
    else
        println()
    end
end

function brackets(k::Int, brackets_level::Int=1)::String
    if brackets_level==0 # never put brackets
        return string(k)
    elseif brackets_level==1 # put brackets if more than one digit
        if k>9
            return "{"*string(k)*"}"
        else
            return string(k)
        end
    else # always put brackets
        return "{"*string(k)*"}"
    end
end

function delta_repr(f::ModularFormOrNothing, Delta_symbol::String="\\Delta",
↪ brackets_level::Int=1, math_mode::Bool=true)::String
    if f===nothing
        return "error"
    end
    if length(f.nzind)==0
        return "0"
    else
        k = f.nzind[1]
        s = Delta_symbol*"^"*brackets(k-1, brackets_level)
        for k in f.nzind[2:end]
            s *= " + "*Delta_symbol*"^"*brackets(k-1, brackets_level)
        end
```

```
            if math_mode
                return "\$"*s*"\$"
            else
                return s
            end
        end
    end


    include("arithmetic.jl")
    include("equality.jl")
    include("generators.jl")
    include("HeckeOperator.jl")
    include("data/storage_text.jl")
    include("data/storage.jl")
    include("recognizer.jl")





end
```

arithmetic.jl

```
"""
We can represent a modular forms mod 2 by it's coefficients as a polynomial in q or
↪  D.
The routines in this file are made for q-series.
"""





### ARITHMETIC OPERATIONS of modular forms modulo 2
"""
    +(f1, f2)

Compute the addition of two modular forms (with mathematical accuracy).

# Example
```julia-repl
[f1 & f2 are modular forms mod 2]
julia> f1+f2
1000-element SparseVector{Int8,Int64} with 27 stored entries:
  [...]
```
```

````
```
"""
function +(f1::ModularForm, f2::ModularForm)::ModularForm
    m = min(f1.n, f2.n)
    f = truncate(f1, m)
    for n in f2.nzind
        if n<=m
            f[n] = 1-f[n]
        end
    end
    return dropzeros!(f)
end

"""
    *(f1, f2)

Compute the multiplication of two modular forms (with mathematical accuracy).

# Example
```julia-repl
[f1 & f2 are modular forms mod 2]
julia> f1*f2
1000-element SparseVector{Int8,Int64} with 86 stored entries:
  [...]
```
"""
function *(f1::ModularForm, f2::ModularForm)::ModularForm
    n::Int = min(f1.n, f2.n)
    f = spzeros(Int8, n)
    for n in f1.nzind
        for m in f2.nzind
            if n+m-1 <= f.n
                f[n+m-1]=1-f[n+m-1]
            else
                break
            end
        end
    end
    return dropzeros!(f)
end

"""
    sq(f)

Compute the square of a modular form (with mathematical accuracy).

This is a much more effcient method then computing the square with multiplication.
````
````

```
sq(f) is (much) more effcient then f*f, time wise and memory wise.

# Example
```julia-repl
[f is a modular form mod 2]
julia> @time f*f
  0.169466 seconds (37 allocations: 1.127 MiB)

julia> @time sq(f)
  0.000020 seconds (23 allocations: 9.875 KiB)
```
"""
function sq(f::ModularForm)::ModularForm
    f_sq = spzeros(Int8, f.n)
    for n in f.nzind
        if 2n-1 <= f_sq.n
            f_sq[2n-1] = 1
        end
    end
    return f_sq
end

"""
    ^(f, k)

Compute f^k (with mathematical accuracy).

Example
```julia-repl
[f is a modular form mod 2]
julia> f^5
1000-element SparseVector{Int8,Int64} with 75 stored entries:
  [...]
```
"""
function ^(f::ModularForm, k::Int)::ModularForm
    # we use binary decomposition of k for effciency
    f_pow=one(f.n)
    while k != 0
        if k&1 != 0
            f_pow *= f
        end
        f = sq(f)
        k >>= 1
    end
    return f_pow
end
```

equality.jl

```julia
"""
Routines to check equality of modular forms modulo two up to known coefficients.
Also defines useful truncations methods.
"""





### TRUNCATION of modular forms modulo 2
"""
    truncate(f, LENGTH)

Truncate f to the LENGTH first coefficients with no error.

# Example
```julia-repl
[f is a modular form mod 2]
julia> f
1000-element SparseVector{Int8,Int64} with 16 stored entries:
  [...]

julia> truncate(f, 100)
100-element SparseVector{Int8,Int64} with 5 stored entries:
  [...]
```
"""
function truncate(f::ModularForm, LENGTH::Int=10^3)::ModularForm
    if f.n>LENGTH
        return f[1:LENGTH]
    else
        return f
    end
end

"""
    truncate(f1, f2[, LENGTH])

Truncate f1 and f2 to LENGTH first coefficients with no error.
Truncate to min length of f1 & f2 if LENGTH = -1.

Example
```julia-repl
[f1 & f2 are modular forms mod 2]
julia> disp(f1)
```
```

```
MF mod 2 (coef to 1000) - 01000000010000000000000001000000000000000000000001...
julia> disp(f2)
MF mod 2 (coef to 100) - 01000000010000000000000001000000000000000000000001...

julia> f1, f2 = truncate(f1,f2)

julia> disp(f1)
MF mod 2 (coef to 100) - 01000000010000000000000001000000000000000000000001...
julia> disp(f2)
MF mod 2 (coef to 100) - 01000000010000000000000001000000000000000000000001...
```
"""
function truncate(f1::ModularForm, f2::ModularForm,
↪   LENGTH::Int=-1)::Tuple{ModularForm,ModularForm}
    if LENGTH == -1
        if f1.n>f2.n
            return (f1[1:f2.n], f2)
        elseif f1.n==f2.n
            return (f1, f2)
        else
            return (f1, f2[1:f1.n])
        end
    else
        n=min(LENGTH, f1.n, f2.n)
        return (f1[1:n], f2[1:n])
    end
end

EQUALITY up to size
"""
    eq(f1, f2)

Up to maximum coefficient known for both f1 and f2, tell equality.
"""
function eq(f1::ModularForm, f2::ModularForm)::Bool
    f1, f2 = truncate(f1, f2)
    return f1==f2
end
```

## generators.jl

```
"""
This file contains routines to generete standard modular forms mod 2.
Essentialy, this is 0 form, 1 form, and powers of D.
"""
```
```

```julia
"""
    zero([LENGTH])

Create a zero form of length LENGTH

# Example
```julia-repl
julia> zero()
1000-element SparseVector{Int8,Int64} with 0 stored entries

julia> zero(1)
1-element SparseVector{Int8,Int64} with 0 stored entries
```
"""
function zero(LENGTH::Int=10^3)::ModularForm
    return spzeros(Int8, LENGTH)
end

"""
    one([LENGTH])

Create a one form of length LENGTH

Example
```julia-repl
julia> one()
1000-element SparseVector{Int8,Int64} with 1 stored entry:
  [1   ]  =  1

julia> one(1)
1-element SparseVector{Int8,Int64} with 1 stored entry:
  [1]  =  1
```
"""
function one(LENGTH::Int=10^3)::ModularForm
    f = spzeros(Int8, LENGTH)
    f[1] = 1
    return f
end

"""
    delta([LENGTH])

Create the standard D form, with coefficients up to LENGTH
```

```
=> as a D-series!

# Example
```julia-repl
julia> disp(delta())
MF mod 2 (coef to 1000) - 01000000010000000000000001000000000000000000000001...

julia> disp(delta(10^6))
MF mod 2 (coef to 1000000) - 01000000010000000000000001000000000000000000000001...
```
"""
function delta(LENGTH::Int=10^3)::ModularForm
    f = spzeros(Int8, LENGTH)
    indice::Int = 2
    i::Int = 1
    while indice <= f.n
        f[indice] = Int8(1)
        i += 2
        indice = i^2+1
    end
    return f
end

"""
    delta_k(k[, LENGTH])

Create the standard D^k form, with coefficients up to LENGTH
=> as a q-series!

Example
```julia-repl
julia> disp(delta_k(0))
MF mod 2 (coef to 1000) - 10000000000000000000000000000000000000000000000000...

julia> disp(delta_k(1))
MF mod 2 (coef to 1000) - 01000000010000000000000001000000000000000000000001...

julia> disp(delta_k(2))
MF mod 2 (coef to 1000) - 00100000000000000010000000000000000000000000000000...

julia> disp(delta_k(3))
MF mod 2 (coef to 1000) - 00010000000100000001000000000000000000000001000000...

julia> disp(delta_k(5))
MF mod 2 (coef to 1000) - 00000100000001000000000000000100000001000000010000...
```
"""
```

```
function delta_k(k::Int, LENGTH::Int=10^3)::ModularForm
    if k==0
        return one(LENGTH)
    elseif k==1
        return delta(LENGTH)
    elseif k==2
        return sq(delta(LENGTH))
    else
        return delta(LENGTH)^k
    end
end




"""
    Delta(k[, LENGTH])

Create the standard D form, with coefficients up to LENGTH
=> as a D-series!

# Example
```julia-repl
julia> disp(Delta(1))
MF mod 2 (coef to 100) - 01000000000000000000000000000000000000000000000000...
```
"""
function Delta_k(k::Int, LENGTH::Int=10^2)::ModularForm
    df = zero(LENGTH)
    df[k+1]=1
    return df
end
```

HeckeOperator.jl

```
"""
Routines for Hecke operators on modular forms mod 2
"""




"""
    Hecke(p, f)

Compute Tp|f (with mathematical accuracy).

# Example
```julia-repl
```
```

```
julia> d=delta()
julia> disp(d)
MF mod 2 (coef to 1000) - 01000000010000000000000001000000000000000000000001...

julia> disp(Hecke(2, d))
MF mod 2 (coef to 500) - 00100000000000000010000000000000000000000000000000...

julia> disp(Hecke(3, d))
MF mod 2 (coef to 333) - 00000000000000000000000000000000000000000000000000...
```
"""
function Hecke(p::Int, f::ModularForm)::ModularForm
    Tpf = spzeros(Int8, f.n÷p)
    i::Int = 0
    while i+1 <= Tpf.n
        if i % p == 0
            Tpf[i+1] = (f[i*p+1] + f[div(i, p)+1]) % 2
        else
            Tpf[i+1] = f[i*p+1]
        end
        i += 1
    end
    return Tpf
end
```

### recognizer.jl

```
"""
Routines to change between representation of modular forms modulo two.
Uses a list of pre-calculated powers of delta to speed up the calculations.

Note that for speed purposes, no checks of any kind are made.
The user should make sure that the form may be written in terms for form from the
↪ precalculated variable.
"""

"""
    to_q(df, precalculated)

Compute the q-series represenstaion of f (using precalculated).

Example
```julia-repl
```

```
julia> precalculated = loadFormListBinary(10^2, 10^6)
julia> df = Delta_k(5)
julia> disp(df)
MF mod 2 (coef to 100) - 00000100000000000000000000000000000000000000000000...

julia> f = to_q(df, precalculated)
julia> disp(f)
MF mod 2 (coef to 1000000) - 00000100000001000000000000000100000001000000010000...

```
"""
function to_q(df::ModularForm, precalculated::ModularFormOrNothingList,
↪   LENGTH::Int=length(precalculated[2]))::ModularForm
    f = zero(LENGTH)
    for k in df.nzind
        f += precalculated[k]
    end
    return f
end

"""
    to_D(f, precalculated)
    -- or --
    to_delta(f, precalculated)

Compute the D-series represenstaion of f (using precalculated).

Example
```julia-repl
julia> precalculated = loadFormListBinary(10^2, 10^6)
julia> f = delta(10^6) + delta_k(3, 10^6)
julia> disp(f)
MF mod 2 (coef to 1000000) - 01010000010100000001000001000000000000000001000001...

julia> df = to_delta(f, precalculated)
100-element SparseArrays.SparseVector{Int8,Int64} with 2 stored entries:
  [2  ]  =  1
  [4  ]  =  1

julia> disp(df)
MF mod 2 (coef to 100) - 01010000000000000000000000000000000000000000000000...
```
"""
function to_D(mf::ModularForm, precalculated::ModularFormOrNothingList,
↪   LENGTH::Int=length(precalculated))::ModularForm
    f = deepcopy(mf)
    k = 1
```
```

```julia
    df = zero(LENGTH)
    for k in 1:2:LENGTH
        if f[k+1]==1
            #there is D^k if f
            df[k+1] = 1
            f += precalculated[k+1]
        else
            #there is no D^k if f
            #pass
        end
        k += 1
    end
    return df
end
to_delta = to_D




"""
    drop_error(f, precalculated, LENGTH)

Drops the numerical error that f might have (as long as this error isn't too large).

# Example
```julia-repl
julia> precalculated = loadFormListBinary(10^2, 10^6)
julia> f = delta(10^6) + delta_k(3, 10^6)
julia> disp(f)
MF mod 2 (coef to 1000000) - 01010000010100000001000001000000000000000001000001...

julia> T11f = MFmod2.Hecke(11, f)
julia> disp(T11f)
MF mod 2 (coef to 90909) - 01000000010000000000000001000000000000000000000001...

julia> T11f_exact = drop_error(T11f, precalculated)
julia> disp(T11f_exact)
MF mod 2 (coef to 1000000) - 01000000010000000000000001000000000000000000000001...

```
"""
function drop_error(f::ModularForm, precalculated::ModularFormOrNothingList,
↪   LENGTH::Int=length(precalculated[2]))::ModularForm
    df = to_D(f, precalculated)
    f = to_q(df, precalculated)
    return f
end
```

C.2.2 Data Submodule

storage.jl

```
using JLD2, FileIO

"""
Load the modular forms lists from binary .jdl2 files, using this module's standard
↪ naming system.
"""



"""
    loadFormListBinary(MAXI, LENGTH)

Loads the list of q-coefficients of D powers form file.
"""
function loadFormListBinary(MAXI::Int, LENGTH::Int)
    # standard naming
    file_name = "delta_q-"*"maxi"*string(MAXI)*"-"*"length"*string(LENGTH)*".jdl2"
    # load
    return load(joinpath(@__DIR__, file_name), "list")
end

"""
    loadHeckePrimesListBinary(MAX_PRIME, MAX_DELTA)

Loads the list of D-coefficients of prime Hecke operators applied to powers of D.
"""
function loadHeckePrimesListBinary(MAX_PRIME::Int, MAX_DELTA::Int)
    # standard naming
    file_name =
     ↪ "Hecke_primes-"*"max_prime"*string(MAX_PRIME)*"-"*"max_delta"*string(MAX_DELTA)*".jdl2"
    # load
    return load(joinpath(@__DIR__, file_name), "list")
end

"""
    loadHeckePowersListBinary(MAX_POWER, MAX_DELTA)

Loads the list of D-coefficients of powers of Hecke operators applied to powers of D.
"""
function loadHeckePowersListBinary(MAX_POWER::Int, MAX_DELTA::Int)
    # standard naming
    file_name =
     ↪ "Hecke_powers-"*"max_power"*string(MAX_POWER)*"-"*"max_delta"*string(MAX_DELTA)*".jdl2"
    # load
    return load(joinpath(@__DIR__, file_name), "list")
```

```
end
```

delta_file_maker.jl

```
include("../ModularFormsModuloTwo.jl")
using .ModularFormsModuloTwo
MFmod2 = ModularFormsModuloTwo
using JLD2, FileIO

# parameters
LENGTH = 10^6
MAXI = 10^3

# list
list = MFmod2.ModularFormOrNothingList(nothing, MAXI)

# D^2
d2 = MFmod2.delta_k(2, LENGTH)
 # 1st iteration
println("Calculating: ", "D^1")
d = MFmod2.delta(LENGTH)
list[2] = d
k = 1
# main loop
while k < MAXI-2
    global k, d, d2
    k += 2
    println("Calculating: ", "D^"*string(k))
    d *= d2
    list[k+1] = d
end


# final saving (standard naming)
@save joinpath(@__DIR__,
↪   "delta_q-"*"maxi"*string(MAXI)*"-"*"length"*string(LENGTH)*".jdl2") list
```

Hecke_primes_file_maker.jl

```
include("../ModularFormsModuloTwo.jl")
using .ModularFormsModuloTwo
MFmod2 = ModularFormsModuloTwo
using JLD2, FileIO
using Primes

# parameters
MAXI = 10^2
LENGTH = 10^6
```

```
# binary read
precalculated = MFmod2.loadFormListBinary(MAXI, LENGTH)

# parameters
MAX_DELTA = length(precalculated)
MAX_PRIME = 10^4

# list
list = Array{Union{MFmod2.ModularFormOrNothingList, Nothing}, 1}(nothing, MAX_PRIME)




for p in Primes.primes(3, MAX_PRIME) #avoid prime 2
    print("Calculating: T", p, "| ")
    l = MFmod2.ModularFormOrNothingList(nothing, MAX_DELTA)
    for k in 1:2:MAX_DELTA-2
        print("D^", k, " ")
        f = MFmod2.delta_k(k, LENGTH)
        Tf = MFmod2.Hecke(p, f)
        dTf = MFmod2.to_delta(Tf, precalculated)
        l[k+1] = dTf
    end
    println()
    list[p] = l
end


# final saving (standard naming)
@save joinpath(@__DIR__,
↪   "Hecke_primes-"*"max_prime"*string(MAX_PRIME)*"-"*"max_delta"*string(MAX_DELTA)*".jdl2")
↪   list
```

Hecke_powers_file_maker.jl

```
include("../ModularFormsModuloTwo.jl")
using .ModularFormsModuloTwo
MFmod2 = ModularFormsModuloTwo
using JLD2, FileIO
using Primes

# parameters
MAXI = 10^3
LENGTH = 10^6

# binary read
```

```
precalculated = MFmod2.loadFormListBinary(MAXI, LENGTH)

# parameters
MAX_DELTA = length(precalculated)
MAX_POWER = 20

# list
list = Array{Union{MFmod2.ModularFormOrNothingList, Nothing}, 2}(nothing, MAX_POWER,
↪  MAX_POWER)



for i in 1:MAX_POWER
    for j in 1:MAX_POWER
        print("Calculating: T3^", i-1, "T5^", j-1, "| ")
        l = MFmod2.ModularFormOrNothingList(nothing, MAX_DELTA)
        # use previous calculations
        if i == 1
            if j == 1
                for k in 1:2:MAX_DELTA-2
                    print("D^", k, " ")
                    df = MFmod2.Delta_k(k, MAX_DELTA)
                    l[k+1] = df
                end
            else
                for k in 1:2:MAX_DELTA-2
                    print("D^", k, " ")
                    df = list[i,j-1][k+1]
                    f = MFmod2.to_q(df, precalculated)
                    T5f = MFmod2.Hecke(5, f)
                    dT5f = MFmod2.to_delta(T5f, precalculated)
                    l[k+1] = dT5f
                end
            end
        else
            for k in 1:2:MAX_DELTA-2
                print("D^", k, " ")
                df = list[i-1,j][k+1]
                f = MFmod2.to_q(df, precalculated)
                T3f = MFmod2.Hecke(3, f)
                dT3f = MFmod2.to_delta(T3f, precalculated)
                l[k+1] = dT3f
            end
        end
        println()
        # save
        list[i,j] = l
```

```
    end
end



# final saving (standard naming)
@save joinpath(@__DIR__,
↪   "Hecke_powers-"*"max_power"*string(MAX_POWER)*"-"*"max_delta"*string(MAX_DELTA)*".jdl2")
↪   list
```

D Chebotarev Example

```
m = 0
f = x^3 + m*x^2 + (m-3)*x - 1
K = NumberField(f, 'a')

print("Defining polynomial:")
print(f)
> x^3 - 3*x - 1
print("Discriminant:")
print(K.discriminant())
> 81
print("Extension Degree:")
print(K.degree())
> 3
print("Galois extension:")
print(K.is_galois())
> True
print("Basis:")
True
print(K.maximal_order().basis())
> [1, a, a^2]
```

E T_p as series of T_3 and T_5

E.1 $a_ij(p)$ Computations

E.2 $a_ij(p)$ Graphs

F Governing Fields

F.1 Check of Known Governing Fields

F.2 Frobenian Elements Computations in Extensions

F.3 Analysis of Extensions

References

Keith Conrad. $\mathrm{SL}_2(\mathbb{Z})$. [Online], 2020. URL https://kconrad.math.uconn.edu/blurbs/grouptheory/SL(2,Z).pdf. Available from https://kconrad.math.uconn.edu/blurbs/grouptheory/SL(2,Z).pdf.

William Stein. *Modular forms, a computational approach*, volume 79 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2007. ISBN 978-0-8218-3960-7; 0-8218-3960-8. doi: 10.1090/gsm/079. URL https://doi.org/10.1090/gsm/079. With an appendix by Paul E. Gunnells.

Bertil Westergren Lennart Rade. *Mathematics Handbook for Science and Engineering*. Springer Science and Business Media, 2013, 2013. 5th edition, illustrated.

J.-P. Serre. *A course in arithmetic*. Springer-Verlag, New York-Heidelberg, 1973. Translated from the French, Graduate Texts in Mathematics, No. 7.

Sagar Shrivastava. Introduction to modular forms, 2017.

Victor Saul Miller. DIOPHANTINE AND P-ADIC ANALYSIS OF ELLIPTIC CURVES AND MODULAR FORMS, 1975. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:0295447. Thesis (Ph.D.)–Harvard University.

SageMath Contributors. Sagemath. Software tool, 2020. URL https://www.sagemath.org/.

O. Kolberg. Congruences for Ramanujan's function $\tau(n)$. *Arbok Univ. Bergen Mat.-Natur. Ser.*, 1962 (11), 1962. ISSN 0522-9189.

Jean-Louis Nicolas and Jean-Pierre Serre. Formes modulaires modulo 2: l'ordre de nilpotence des opérateurs de Hecke. *C. R. Math. Acad. Sci. Paris*, 350(7-8):343–348, 2012a. ISSN 1631-073X. doi: 10.1016/j.crma.2012.03.013. URL https://doi.org/10.1016/j.crma.2012.03.013.

Jean-Louis Nicolas and Jean-Pierre Serre. Formes modulaires modulo 2: structure de l'algèbre de Hecke. *C. R. Math. Acad. Sci. Paris*, 350(9-10):449–454, 2012b. ISSN 1631-073X. doi: 10.1016/j.crma.2012.03.019. URL https://doi.org/10.1016/j.crma.2012.03.019.

Ken Ono. *The web of modularity: arithmetic of the coefficients of modular forms and q-series*, volume 102 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 2004. ISBN 0-8218-3368-5.

Kazuyuki Hatada. Eigenvalues of Hecke operators on $\mathrm{SL}(2, \mathbf{Z})$. *Math. Ann.*, 239(1):75–96, 1979. ISSN 0025-5831. doi: 10.1007/BF01420494. URL https://doi.org/10.1007/BF01420494.

Mathilde Gerbelli-Gauthier. Modular forms and galois representations mod p, and the nilpotent action of hecke operators mod 2, 2014. URL http://www.math.mcgill.ca/darmon/theses/gerbelli-gauthier/mathilde-gg.pdf.

Lawrence C. Washington. *Introduction to cyclotomic fields*, volume 83 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1997. ISBN 0-387-94762-0. doi: 10.1007/978-1-4612-1934-7. URL https://doi.org/10.1007/978-1-4612-1934-7.

Gerald J. Janusz. *Algebraic number fields*, volume 7 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 1996. ISBN 0-8218-0429-4.

N. Tschebotareff. Die Bestimmung der Dichtigkeit einer Menge von Primzahlen, welche zu einer gegebenen Substitutionsklasse gehören. *Math. Ann.*, 95(1):191–228, 1926. ISSN 0025-5831. doi: 10.1007/BF01206606. URL https://doi.org/10.1007/BF01206606.

Jean-Pierre Serre. *Lectures on* $N_X(p)$, volume 11 of *Chapman & Hall/CRC Research Notes in Mathematics*. CRC Press, Boca Raton, FL, 2012. ISBN 978-1-4665-0192-8.