ETH zürich

ETA PRODUCTS

RYAN RUEGER

A semester thesis submitted to The Swiss Federal Technical Institute of Technology in Zurich, March 2023.

Supervised by Dr. M. Schwagenscheidt.

Contents

1 Preliminaries on modular functions

Let R be a ring, denote by $\mathrm{SL}_2(R)$ the set of 2×2 matrices with entries in R and determinant 1. We call $\mathrm{SL}_2(\mathbb{Z})$ the *modular group*, $\mathrm{SL}_2(\mathbb{R})$ the *real modular group* and, for brevity, subgroups of the (real) modular group *(real) modular subgroups*. Two real modular subgroups Γ, Γ' are said to be *commensurable* if their intersection $\Gamma \cap \Gamma'$ has finite index in both Γ and Γ'. We will use the inline notation[1]

$$(a, b; c, d) \quad \text{for the } 2 \times 2 \text{ matrix} \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Moreover, we denote by $\mathcal{H}$ the complex upper half plane $\{z \in \mathbb{C} \mid \mathrm{im}(z) > 0\}$ and by $\overline{\mathcal{H}}$ its closure $\{z \in \mathbb{C} \mid \mathrm{im}(z) \geq 0\} \cup \{\infty\}$.

The real modular group acts on $\overline{\mathcal{H}}$ via *Möbius transformations* or *broken linear transformations*. Let z lie in $\overline{\mathcal{H}}$ and $(a, b; c, d)$ in $\mathrm{SL}_2(\mathbb{R})$, then we define

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} z \overset{\text{def.}}{=} \begin{cases} a/c & \text{if } z = \infty, c \neq 0 \\ \infty & \text{elif } z = \infty, c = 0 \\ \infty & \text{elif } cz + d = 0 \\ (az + b)/(cz + d) & \text{else } (cz + d \neq 0). \end{cases}$$

This action is still well defined when restricted to the complex upper half plane $\mathcal{H}$. Finally, we call the expression $j(\tau, (a, b; c, d)) = c\tau + d$ the *automorphy factor* of $(a, b; c, d)$ at τ.

An important example of this action, is the *translation by one* matrix $T = (1, 1; 0, 1)$ acting on any $z \neq \infty$ to yield $Tz = z + 1$. Notably also, T stabilises ∞.

Another important (counter)example is that, in contrast to regular matrix multiplication, this action does not commute with scalar multiplication. More precisely, if λ is in $\mathbb{R}$ and $\gamma = (a, b; c, d)$ an $\mathrm{SL}_2(\mathbb{R})$ matrix, then $\gamma\lambda\tau \neq \lambda\gamma\tau$ in general. This can be seen through direct calculation, but also understood intuitively: after noting that

$$\lambda\tau = \begin{pmatrix} \lambda & 0 \\ 0 & 1 \end{pmatrix} \tau \quad \text{so} \quad \lambda\gamma\tau = \begin{pmatrix} \lambda & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tau \quad \text{and} \quad \gamma\lambda\tau = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \lambda & 0 \\ 0 & 1 \end{pmatrix} \tau.$$

Indeed recall that matrices do not commute in general, so we do not expect equality.

Throughout this work, we will be working with complex numbers of the form $\exp(2\pi\iota z/n)$ for complex z where ι is the complex unit. To simplify this notation we will write $e_n(z)$ instead and when $n = 1$ simply $e(z)$. Using $q = \exp(2\pi\iota\tau)$ then becomes $q = e(\tau)$ and $e_n(\alpha\tau) = q^{\alpha/n}$.

Now let Γ be a subgroup of $\mathrm{GL}_2^+(\mathbb{R}) = \{\gamma \in \mathrm{GL}_2(\mathbb{R}) \mid \det(\gamma) > 0\}$ commensurable with $\mathrm{SL}_2(\mathbb{Z})$, k a real number and $\nu\colon \Gamma \to \mathbb{C}$ a function with $|\nu(\gamma)| = 1$ for all γ in Γ. A meromorphic function $f\colon \mathcal{H} \to \mathbb{C}$ is called a *weakly modular function* of *weight* k, *level* Γ and *multiplier system* ν if

$$f(\gamma\tau) = \nu(\gamma)\det(\gamma)^{-k/2} j(\tau, \gamma)^k f(\tau) \tag{1.1}$$

[1] This notation is common to computer algebra systems such as GNU Octave and MATLAB

holds for all γ in Γ and τ in $\mathcal{H}$. The space of all such functions f forms a complex vector space which we will denote by $M^!_{k,\nu}(\Gamma)$.

1.1 Commensurability and Expansions at cusps

We require commensurability of Γ with $\mathrm{SL}_2(\mathbb{Z})$ because it ensures a sort of periodicity condition for the functions in $M^!_{k,\nu}(\Gamma)$. More precisely, since $\Gamma \cap \mathrm{SL}_2(\mathbb{Z})$ has finite index in $\mathrm{SL}_2(\mathbb{Z})$, there exists a minimal positive power h for which the element $T^h = (1, h; 0, 1)$ of $\mathrm{SL}_2(\mathbb{Z})$ lies in $\Gamma \cap \mathrm{SL}_2(\mathbb{Z}) \subseteq \Gamma$. (This is true in general: for every group G and subgroup H of finite index: every element g in G has a power k for which g^k lies in H). From the modular invariance condition (Eq. 1.1) we garner that f is "almost" h-periodic

$$f(\tau + h) = f(T^h\tau) = \nu(T^h)\det(T)^{-k/2}j(\tau, T^h)^k f(\tau) = \nu(T^h)f(\tau).$$

Since $|\nu(\gamma)| = 1$ for all γ in Γ, we can choose a real θ such that $\nu(T^h) = e(\theta)$. Consequently $F(\tau) = e(-\theta\tau)f(h\tau)$ is a 1-periodic meromorphic function on $\mathcal{H}$ and so has a Fourier expansion of the form

$$F(\tau) = \sum_{n\in\mathbb{Z}} f_n e(n\tau) \quad \text{so} \quad f(\tau) = \sum_{n\in\mathbb{Z}} f_n e_h((n+\theta)\tau).$$

We call the coefficients f_n the *Fourier coefficients* of f, and $F(\tau)$ the *expansion of f at infinity*. When written in the form

$$F(\tau) = \sum_{n\in\mathbb{Z}} f_n q^n$$

the *q-expansion* of f (at infinity). Sometimes, we will refer to the expansion of $f(\tau)$ directly as the expansion at infinity by abuse of notation. We say that f is holomorphic (meromorphic) at infinity if the Fourier coefficients vanish for $n \leq 0$ ($n \leq n_0$ for some $n_0 < 0$). If f is holomorphic at infinity, we define f at infinity to be f_0, as this is the only term which does not vanish when taking the limit of $f(\iota y)$ with real y going to $+\infty$. Moreover, we define the *order* of f at infinity to be $(n_0 + \theta)/h$ where n_0 is the first index for which f_{n_0} is non-vanishing.

It is worth noting that this expansion is often presented as the Laurent expansion of $\tilde{f}(q) = F(\log(q)/2\pi\iota)$ at $q = 0$; explaining the nomenclature "expansion at infinity", since the map $q \mapsto \log(q)/2\pi\iota$ (the left-inverse of $e(\cdot)$) sends 0 to infinity. First we assure ourselves that this is well defined. We note that $q \mapsto \log(q)/2\pi\iota$ is a holomorphic map from the open punctured unit disc D^* to the upper complex half-plane. Indeed, since the logarithm is multi-valued, we have for some integral k that

$$\frac{\log(q)}{2\pi\iota} = \frac{\log(|q|) + i(\arg(z) + 2\pi k)}{2\pi\iota} = \frac{\arg(z)}{2\pi} + k - \iota\frac{\log(|q|)}{2\pi}.$$

On D^*, we have that $0 < |q| < 1$ so $\log(|q|) < 0$ and $\log(q)/2\pi\iota$ lies in $\mathcal{H}$. Moreover, since F is 1-periodic, $F(\log(q)/2\pi\iota)$ does not depend on k and $\tilde{f}$ is well-defined meromorphic on D^*. Therefore $\tilde{f}$ has a Laurent expansion at $q = 0$

$$\tilde{f}(q) = \sum_{n\geq n_0} \tilde{f}_n q^n \quad \text{so} \quad \tilde{f}(e(\tau)) = F(\tau) = \sum_{n\geq n_0} \tilde{f}_n e(\tau n).$$

This procedure of finding a way to extend f to infinity can also be used to extend f to any rational number on the real line.

To that end, consider a rational number $r = a/c$ in reduced fraction form. Bézout's identity delivers us integers b, d such that $M = (a, b; c, d)$ has determinant 1. In particular, $M\infty = r$. Since Γ is commensurable with $\mathrm{SL}_2(\mathbb{Z})$, $M^{-1}\Gamma M$ is commensurable with $M^{-1}\,\mathrm{SL}_2(\mathbb{Z})M = \mathrm{SL}_2(\mathbb{Z})$. So, as before, there exists a minimal positive power h such that T^h lies in $M^{-1}\Gamma M$. So T^h is of the form $M^{-1}LM$ for some $L = (\alpha, \beta; \gamma, \delta)$ in Γ. In particular then, $L = MT^hM^{-1}$ fixes r.

Calculating

$$L = MT^hM^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \begin{pmatrix} 1-ach & a^2h \\ -c^2h & 1+ach \end{pmatrix}$$

we obtain

$$\begin{aligned} L\tau - r &= \frac{(1-ach)\tau + a^2h}{-c^2h\tau + 1 + ach} - \frac{a}{c} \\ &= \frac{c((1-ach)\tau + a^2h) - a(-c^2h\tau + 1 + ach)}{c(-c^2h\tau + 1 + ach)} \\ &= \frac{c\tau - a}{c(-c^2h\tau + 1 + ach)} \\ &= \frac{\tau - r}{\tau\gamma + \delta}. \end{aligned}$$

So the function $(\tau - r)^k f(\tau)$ satisfies

$$(L\tau - r)^k f(L\tau) = (L\tau - r)^k \nu(L)\det(L)^{-k/2}(\gamma\tau + \delta)^k f(\tau) = \nu(L)(\tau - r)^k f(\tau);$$

or equivalently, after replacing τ with $M\tau$ and choosing a real Λ so that $\nu(L) = e(\Lambda)$,

$$(MT^h\tau - r)^k f(MT^h\tau) = e(\Lambda)(M\tau - r)^k f(M\tau).$$

So $F(\tau) = e(-\Lambda\tau)(Mh\tau - r)^k f(Mh\tau)$ is a 1-periodic meromorphic function. Indeed,

$$\begin{aligned} F(\tau+1) &= e(-\Lambda(\tau+1))(Mh(\tau+1) - r)^k f(Mh(\tau+1)) \\ &= e(-\Lambda)e(-\Lambda\tau)(MT^h(h\tau + h) - r)^k f(MT^h(h\tau + h)) \\ &= e(-\Lambda)e(-\Lambda\tau)(MT^hh\tau - r)^k f(MT^hh\tau) \\ &= e(-\Lambda)e(-\Lambda\tau)(MT^hM^{-1}Mh\tau - r)^k f(MT^hM^{-1}Mh\tau) \\ &= e(-\Lambda)e(-\Lambda\tau)(LMh\tau - r)^k f(LMh\tau) \\ &= e(-\Lambda)e(-\Lambda\tau)(LMh\tau - r)^k \nu(L)\det(L)^{k/2} j(Mh\tau, L)^{-k} f(Mh\tau) \\ &= e(-\Lambda\tau)(LMh\tau - r)^k (Mh\tau\gamma + \delta)^{-k} f(Mh\tau) \\ &= e(-\Lambda\tau)(Mh\tau - r)^k f(Mh\tau) \\ &= F(\tau). \end{aligned}$$

Therefore $F(\tau)$ has a Fourier series expansion

$$F(\tau) = e(-\Lambda\tau)(Mh\tau - r)^k f(Mh\tau) = \sum_{n\in\mathbb{Z}} f_n e(n\tau)$$
$$\text{so} \quad f(\tau) = (\tau - r)^{-k} \sum_{n\in\mathbb{Z}} f_n e_h((n+\Lambda)M^{-1}\tau).$$

We call $F(\tau)$ this *expansion of f at r*. We define f to be holomorphic (meromorphic) in the same way as we did at infinity. The order is also defined similarly, by $(n_0 + \Lambda)/h$ where n_0 is the first index for which f_{n_0} is non-vanishing.

Importantly, we see that the expansion of f at r is the expansion of

$$\tilde{f}(\tau) = e((\theta - \Lambda)\tau)(\tau - r)^k f(M\tau)$$

at infinity. Indeed, as in the expansion at infinity, we define

$$\begin{aligned} F(\tau) = e(-\theta\tau)\tilde{f}(h\tau) &= e(-\theta\tau)e((\theta-\Lambda)\tau)(Mh\tau - r)^k f(Mh\tau) \\ &= e(-\Lambda\tau)(Mh\tau - r)^k f(Mh\tau) \end{aligned}$$

and obtain the same function F as in the case in which we directly expand at r.

1.2 The slash operator

If k is an integer, then an $\mathrm{SL}_2(\mathbb{Z})$-commensurable subgroup $\Gamma \subseteq \mathrm{GL}_2^+(\mathbb{R})$ acts on space M of meromorphic functions on $\mathcal{H}$ by the *weight-k, multiplier-system-ν slash operator*

$$(f|_{k,\nu}\gamma)(\tau) \overset{\text{def.}}{=} \nu(\gamma)^{-1}\det(\gamma)^{k/2} j(\tau,\gamma)^{-k} f(\gamma\tau).$$

if and only if ν is multiplicative (i.e. a morphism of groups $\Gamma \to \mathbb{C}^\times$). In this terminology, $M^!_{k,\nu}(\Gamma)$ is the $\cdot|_{k,\nu}$-invariant subspace of M.

An important consequence in the integer-weight, multiplicative multiplier-system case is that we now only need to know how the function f transforms under *generators* of Γ to determine whether f is weakly modular of level Γ. As an example, consider the full modular group $\mathrm{SL}_2(\mathbb{Z})$. It is generated by $T = (1,1;0,1)$ and $S = (0,1;-1,0)$. Therefore we must only verify that

$$f(T\tau) = f(\tau+1) = \nu(T)f(\tau) \quad \text{and} \quad f(S\tau) = f(-1/\tau) = \nu(T)\tau^k f(\tau)$$

to assert whether f is in $M^!_{k,\nu}(\mathrm{SL}_2(\mathbb{Z}))$ or not.

To verify that the slash operator defines a group action, we must verify that $f|_{k,\nu}\gamma$ is still well-defined meromorphic, that $f|_{k,\nu}\,\mathrm{id} = f$ and that $(f|_{k,\nu}\gamma)|_{k,\nu}\gamma' = f|_{k,\nu}(\gamma\gamma')$. Well-definedness is clear, since the Möbius transformations are a well defined action. Moreover, $f|_{k,\nu}\,\mathrm{id}(\tau) = \nu(\mathrm{id})^{-1}\det(\mathrm{id})^k j(\tau,\mathrm{id})^{-k} f(\mathrm{id}\,\tau) = f(\tau)$ follows from $\nu(\mathrm{id}) = 1$ since ν is a morphism of groups.

Finally, with γ, γ' elements of Γ we have

$$\begin{aligned}(f|_{k,\nu}\gamma)|_{k,\nu}\gamma'(\tau) &= \nu(\gamma')^{-1}\det(\gamma')^{k/2}j(\tau,\gamma')^{-k}(f|_{k,\nu}\gamma)(\gamma'\tau)\\ &= \nu(\gamma')^{-1}\det(\gamma')^{k/2}j(\tau,\gamma')^{-k}\nu(\gamma)^{-1}\det(\gamma)^{k/2}j(\gamma'\tau,\gamma)^{-k}f(\gamma\gamma'\tau)\\ &= (\nu(\gamma)\nu(\gamma'))^{-1}(\det(\gamma)\det(\gamma'))^{k/2}(j(\tau,\gamma')j(\gamma'\tau,\gamma))^{-k}f(\gamma\gamma'\tau)\\ &= \nu(\gamma\gamma')^{-1}\det(\gamma\gamma')^{k/2}j(\tau,\gamma\gamma')^{-k}f(\gamma\gamma'\tau)\\ &= f|_{k,\nu}(\gamma\gamma')\end{aligned}$$

where the equality $j(\tau,\gamma')j(\gamma'\tau,\gamma) = j(\gamma\gamma',\tau)$ is called the *cocycle relation* and is always true. Indeed, if $\gamma = (a,b;c,d)$ and $\gamma' = (a',b';c',d')$, then

$$\begin{aligned}j(\tau,\gamma')j(\gamma'\tau,\gamma) &= (c'\tau+d')\left(c\left(\frac{a'\tau+b'}{c'\tau+d'}\right)+d\right)\\ &= c(a'\tau+b')+d(c'\tau+d')\\ &= (ca'+dc')\tau+(cb'+dd')\\ &= j(\tau,\gamma\gamma')\end{aligned}$$

because

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}a' & b'\\ c' & d'\end{pmatrix} = \begin{pmatrix}* & *\\ ca'+dc' & cb'+dd'\end{pmatrix}.$$

It is at this point, that we see the importance of Γ being a subgroup of $\mathrm{GL}_2^+(\mathbb{R})$ and k being an integer. Else $\det(\gamma)^k\det(\gamma')^k \neq \det(\gamma\gamma')^k$. Here the classic counter example goes as follows. Suppose $k=1$ and $\gamma = \gamma' = (1,2;1,1)$. Then $\det(\gamma) = \det(\gamma') = -1$ and $\det(\gamma)^{1/2}\det(\gamma')^{1/2} = \sqrt{-1}\sqrt{-1} = \imath\cdot\imath = -1$ but $\det(\gamma\gamma')^{1/2} = (\det(\gamma)\det(\gamma'))^{1/2} = \sqrt{-1\cdot -1} = \sqrt{1} = 1$.

Common modular subgroups of study are

$$\begin{aligned}\Gamma(N) &= \{(a,b;c,d)\in\mathrm{SL}_2(\mathbb{Z}) \mid a\equiv d\equiv 1 \pmod N \text{ and } b\equiv c\equiv 0 \pmod N\}\\ \Gamma_1(N) &= \{(a,b;c,d)\in\mathrm{SL}_2(\mathbb{Z}) \mid a\equiv d\equiv 1 \pmod N \text{ and } c\equiv 0 \pmod N\}\\ \Gamma_0(N) &= \{(a,b;c,d)\in\mathrm{SL}_2(\mathbb{Z}) \mid c\equiv 0 \pmod N\}\end{aligned}$$

We call $\Gamma(N)$ the *principal subgroup of level* N and any modular subgroup $\Gamma \subseteq \mathrm{SL}_2(\mathbb{Z})$ containing $\Gamma(N)$ a *congruence subgroup of level* N. Since $\Gamma(N) \subseteq \Gamma_1(N) \subseteq \Gamma_0(N)$, both $\Gamma_1(N)$ and $\Gamma_0(N)$ are congruence subgroups of level N. Importantly, any congruence subgroup is commensurable with $\mathrm{SL}_2(\mathbb{Z})$.

These (congruence) subgroups lend themselves to the following natural example. Consider a Dirichlet character χ modulo N, we can extend this to a multiplicative multiplier system ν

on $\Gamma_0(N)$ by defining $\nu((a,b;c,d)) = \chi(d)$. Indeed,

$$\begin{aligned}\nu((a,b;c,d)(e,f;g,h)) &= \nu((*,*;*,cf+dh))\\ &= \chi(cf+dh)\\ &= \chi(dh)\\ &= \chi(d)(h)\\ &= \nu((a,b;c,d))\,\nu((e,f;g,h))\end{aligned}$$

since cf is divisible by N. Importantly, d is a unit modulo N because $c \equiv 0 \pmod N$ and $ad - bc = 1$. Consequently ν is non-vanishing for all matrices in $\Gamma_0(N)$ and therefore is compatible with our definition of a multiplier system. Ono says that forms in $M^!_{k,\nu}(\Gamma_0(N))$ have a *Nebentypus character* χ [Ono04, Def. 1.15].

1.3 A lemma on the generators on congruence subgroups

Finally, we finish the preliminaries with a

Lemma 1.1 (On generators of congruence subgroups)**.** [Bor99, Lem. 5.1, pg. 9] *Let $N > 1$ be an integer. Let G_N describe the matrices $(a,b;c,d)$ in $\Gamma_0(N)$ for which $c,d > 0$ and $d \equiv 1 \pmod 4$. Let $Z = (-1,0;0,1)$. If $4 \nmid N$, then G_N generates $\Gamma_0(N)$. Conversely, if $4 \mid N$, then matrices in G_N together with $Z = (-1,0;0,1)$ generate $\Gamma_0(N)$.*

Proof. The strategy of this proof goes as follows. We begin with an arbitrary element γ of $\Gamma_0(N)$ and multiply it by elements $g_1,\dots,g_n$ in G_N ($G_N \cup \{Z\}$) until we obtain an element g in G_N ($G_N \cup \{Z\}$). Then $g_n \cdots g_1 \gamma = g$ so $\gamma = {g_1}^{-1} \cdots {g_n}^{-1} g$ lies in $\langle G_N \rangle$ ($\langle G_N \cup \{Z\} \rangle$).

Consider for $(a,b;c,d)$ in G_N the product

$$\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}\begin{pmatrix}a & b\\ c & d\end{pmatrix} = \begin{pmatrix}a+c & b+d\\ c & d\end{pmatrix}.$$

It clearly also lies in G_N, so $T \overset{\text{def.}}{=} (1,1;0,1)$ is generated by G_N.

Now we will show that $\Gamma_0(N)$ is generated by matrices $(a,b;c,d)$ in $\Gamma_0(N)$ for which d is odd. Suppose d is even (then c must be odd for $(a,b;c,d)$ to lie in $\Gamma_0(N) \subseteq \mathrm{SL}_2(\mathbb{Z})$) and consider

$$\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix} = \begin{pmatrix}a & a+b\\ c & c+d\end{pmatrix}, \quad \text{therefore} \quad \begin{pmatrix}a & b\\ c & d\end{pmatrix} = \begin{pmatrix}a & a+b\\ c & c+d\end{pmatrix}\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}.$$

Since $c+d$ is odd, we have shown that matrices with d even can be generated from matrices $(\tilde{a},\tilde{b};\tilde{c},\tilde{d})$ in $\Gamma_0(N)$ with $\tilde{d}$ odd.

It remains to be shown that it suffices for $d \equiv 1 \pmod 4$ (not just that d is odd). To that end, let $(a,b;c,d)$ be a matrix in $\Gamma_0(N)$ with $d \equiv 3 \pmod 4$. In particular then d is odd. Now observe the following case distinction.

Case: $4 \nmid N$. Consider with k integral the product

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ N & 1 \end{pmatrix}^k \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ kN & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} a + bNk & a + bNk + b \\ c + dNk & c + dNk + d \end{pmatrix}.$$

Notice that 4 and dN are coprime because $4 \nmid N$ and d is odd. So there exists a k such that $c + dNk \equiv 2 \pmod 4$. In turn, $c + dNk + d \equiv 2 + 3 = 1 \pmod 4$. This shows that matrices $(a, b; c, d)$ in $\Gamma_0(N)$ with $d \equiv 3 \pmod 4$ can be generated from matrices from which $d \equiv 1 \pmod 4$ via T and $(1, 0; N, 1)$.

Case: $4 \mid N$. We simply multiply $(a, b; c, d)$ by Z to obtain $(a, -b; c, -d)$ with $-d \equiv 3 \pmod 4$ which is equivalent to $d \equiv 1 \pmod 4$. This shows that matrices $(a, b; c, d)$ in $\Gamma_0(N)$ with $d \equiv 3 \pmod 4$ can be generated from matrices from which $d \equiv 1 \pmod 4$ via Z.

Finally, we must show that matrices $(a, b; c, d)$ with c or d negative can be generated from matrices with c, d positive. Calculating

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ N & 1 \end{pmatrix}^k = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ kN & 1 \end{pmatrix} = \begin{pmatrix} a + kNb & b \\ c + kNd & d \end{pmatrix}$$

for suitable integral k, we see that matrices with negative c are generated from matrices with positive c. With c positive, we can multiply

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{4l} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 4l \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & 4al + b \\ c & 4cl + d \end{pmatrix}$$

to find another matrix with positive d and $d \equiv 1 \pmod 4$ that generates $(a, b; c, d)$. $\square$

2 Number-theoretic tools

Here we present a short interlude to introduce some more notation and useful formulae.

For an odd prime p and an integer n, we define the *Legendre symbol*

$$\left[\frac{n}{p}\right] = \begin{cases} 0 & \text{if } p \text{ divides } n \\ 1 & \text{elif } n \text{ is a non-zero quadratic residue mod } p \\ -1 & \text{else.} \end{cases}$$

We extend this definition to become the *Jacobi symbol* by replacing p by an odd positive integer $m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ and multiplicatively defining

$$\left[\frac{n}{m}\right] = \left[\frac{n}{p_1}\right]^{\alpha_1} \left[\frac{n}{p_2}\right]^{\alpha_2} \cdots \left[\frac{n}{p_k}\right]^{\alpha_k}$$

where the p_i are (odd) primes. Finally, we can extend this to the *Kronecker symbol* for arbitrary integers m and n in the same multiplicative way by defining

$$\left[\frac{n}{2}\right] = \begin{cases} 0 & \text{if } 2 \text{ divides } n \\ 1 & \text{if } a = \pm 1 \pmod 8 \\ -1 & \text{if } a = \pm 3 \pmod 8 \end{cases}, \quad \left[\frac{n}{\pm 1}\right] = \begin{cases} 1 & \text{if } n \geq 0 \\ \pm 1 & \text{if } n < 0 \end{cases}, \quad \left[\frac{n}{0}\right] = \begin{cases} 1 & \text{if } a = \pm 1 \\ 0 & \text{else} \end{cases}.$$

From this definition, we can derive the following properties

(i) The Kronecker symbol is multiplicative in the upper argument.

(ii) The Kronecker symbol is multiplicative in the lower argument.

(iii) Fixing n we obtain a (real) Dirichlet character modulo n, $m \mapsto [m/n]$.

(iv)

$$\left[\frac{-1}{n}\right] = \begin{cases} 1 & n \equiv 1 \pmod 4 \\ -1 & n \equiv 3 \pmod 4 \end{cases}$$

(v) $[n/m] = 0$ if and only if $\gcd(n, m) > 1$.

Definition 2.1 (Jacobi Theta Function)**.** [Sch21, Lem. 6.4.1, pg. 42] For complex τ, z with τ in the upper complex half plane, we define the *Jacobi Theta Function*

$$\vartheta(z;\tau) = \sum_{n\in\mathbb{Z}} e(n^2\tau/2 + nz) = \sum_{n\in\mathbb{Z}} q^{n^2/2}\xi^n$$

where $q = e(\tau), \xi = e(z)$. In defines an entire function in z and satisfies the transformation laws

$$\vartheta(z+1;\tau) = 1 \quad \text{and} \quad \vartheta(z+\tau;\tau) = e_{-1}(\tau/2+z)\vartheta(z;\tau).$$

Lemma 2.2 (Jacobi's Triple Product identity, Euler's Identity)**.** [Sch21, Th. 6.4.2, pg. 42; Köh11, Th. 1.1, pg.4] *For complex q, w with $|q| < 1$ and $w \neq 0$ we have* Jacobi's Triple product identity

$$\vartheta(z;\tau) = \sum_{n\in\mathbb{Z}} q^{n^2/2}\xi^n = \prod_{n\geq 1}(1-q^n)\left(1+\xi q^{n-(1/2)}\right)\left(1+\xi^{-1}q^{n-(1/2)}\right).$$

Furthermore, fixing m, replacing q with $q^{(m+1)}$ and choosing $\xi = -q^{(1-m)/2}$ we obtain

$$\sum_{n\in\mathbb{Z}}(-1)^n q^{n(n(m+1)+1-m)/2} = \prod_{n\geq 1}\left(1-q^{n(m+1)}\right)\left(1-q^{n(m+1)-m}\right)\left(1-q^{n(m+1)-1}\right).$$

In particular, with $m=2$ we obtain Euler's Identity

$$\sum_{n\in\mathbb{Z}}(-1)^n q^{n(3n-1)/2} = \prod_{n\geq 1}\left(1-q^{3n}\right)\left(1-q^{3n-2}\right)\left(1-q^{3n-1}\right) = \prod_{n\geq 1}(1-q^n). \tag{2.1}$$

3 The Dedekind Eta Function

On the complex upper half plane, we define the *Dedekind Eta-function*

$$\eta(\tau) = \exp\left(\frac{\pi\iota}{12}\tau\right)\prod_{n\geq 1}(1-\exp(2n\pi\iota\tau)) = q^{1/24}\prod_{n\geq 1}(1-q^n)$$

where, as usual, $q = \exp(2\pi\iota\tau) = e(\tau)$.

Before we begin showing properties of the function, we will motivate it a little.

3.1 Motivating the Dedekind Eta Function

In this section we summarise Chapters 63, 64, 65 of Rademacher [Rad70].

When studying modular forms, usually the first (almost canonical) examples are the *Eisenstein series of weight k* defined on the complex upper half plane

$$G_k(\tau) \overset{\text{def.}}{=} \sum_{\substack{m,n\in\mathbb{Z}\\(m,n)\neq(0,0)}} \frac{1}{(m\tau+n)^k}.$$

It is easy to show that the transformation behaviour of G_k under the action of the generators S, T in $\mathrm{SL}_2(\mathbb{Z})$ is $G(\tau+1) = G(\tau)$ and $G(-1/\tau) = \tau^k G(\tau)$. Moreover, one can show that these series converge uniformly absolutely for $k > 2$ on compact subsets of $\mathcal{H}$ and thus define modular forms for the full modular group $\mathrm{SL}_2(\mathbb{Z})$. These series clearly vanish for odd k.

To follow Rademacher more closely, we will rephrase these Eisenstein series from being defined as functions on $\mathcal{H}$ to being invariants attached to a lattice $\Omega = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ in $\mathbb{C}$. More precisely, we define

$$G_k(\omega_1, \omega_2) = \sum_{\substack{m,n\in\mathbb{Z}\\(m,n)\neq(0,0)}} \frac{1}{(m\omega_1 + n\omega_2)^k}$$

for complex numbers ω_1, ω_2 that are not $\mathbb{R}$-linearly dependent. Clearly, $G_k(\tau) = G_k(\tau, 1)$. The converse is also true in the following sense. Since ω_1, ω_2 are not $\mathbb{R}$-linearly dependent, $\tau = \omega_2/\omega_1$ lies in $\mathcal{H}$ (after replacing ω_2 with $-\omega_2$ if necessary) and $\Omega = \omega_1(\tau\mathbb{Z} + \mathbb{Z})$. Now we have that $G_k(\tau) = G_k(\tau, 1) = {\omega_1}^k G_k(\omega_1, \omega_2)$.

In the case of weight $k = 2$, the series converge conditionally. Hecke tried to fix this problem by defining

$$G_2^*(\omega_1, \omega_2; s) = \sum_{m,n} \frac{1}{(m\omega_1 + n\omega_2)^2 |m\omega_1 + n\omega_2|^s}$$

and then defining $G_2^*(\omega_1, \omega_2) = \lim_{s\to 0} G_2(\omega_1, \omega_2; s)$. Notably, however, $G_2^*(z; s)$ is not a meromorphic function.

Nevertheless, this is helpful because $G_2(\omega_1, \omega_2; s)$ is absolutely convergent and with

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\tau = \frac{a\tau + b}{c\tau + d} = \frac{\omega_2 a + \omega_1 b}{\omega_2 c + \omega_2 d} \overset{\text{def.}}{=} \frac{\omega_2'}{\omega_1'} \overset{\text{def.}}{=} \tau'$$

it satisfies $G_2(\omega_1, \omega_2; s) = G_2(\omega_1', \omega_2'; s)$ for ω_1', ω_2'. This delivers us the modular invariance for

$G_2(\omega_1, \omega_2)$. A direct consequence of the definnition of τ' is that

$$\frac{\omega_1}{\omega_1'} = \frac{\omega_1}{c\omega_2 + d\omega_1} = \frac{1}{c\tau + d}$$

Now some pages of analysis can be done to yield

$$\begin{aligned} G_2(\omega_1, \omega_2) &= -\frac{2(2\pi)^2}{\omega_1^2}\left(-\frac{1}{24} + \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) - \frac{2\pi\iota}{\omega_1^2(\tau - \overline{\tau})} \\ &= -\frac{1}{\omega_1^2}\left(-\frac{\pi^2}{3} - 2(2\pi\iota)^2 \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) - \frac{2\pi\iota}{\omega_1^2(\tau - \overline{\tau})} \\ &= \frac{1}{\omega_1^2}\left(\frac{\pi^2}{3} + 2\frac{(2\pi\iota)^2}{(2-1)!} \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) - \frac{2\pi\iota}{\omega_1^2(\tau - \overline{\tau})} \\ &= \frac{1}{\omega_1^2}\left(2\zeta(2) + 2\frac{(2\pi\iota)^2}{(2-1)!} \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) - \frac{2\pi\iota}{\omega_1^2(\tau - \overline{\tau})} \end{aligned}$$

Note that this form is very similar to the formula

$$G_k(\tau) = G_k(1, \tau) = 2\zeta(k) + 2\frac{(2\pi\iota)^k}{(k-1)!} \sum_{m=1}^{\infty} \sigma_{k-1}(m)e(m\tau)$$

proven for $k > 2$. Here we see that the additional trailing term in the G_2 case is causing problems.

Using this formula, these two different representations of $G_2(\omega_1, \omega_2) = G_2(\omega_1', \omega_2')$ give us the equality

$$\begin{aligned} &\frac{2(2\pi)^2}{\omega_1^2}\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) + \frac{2\pi\iota}{\omega_1^2(\tau - \overline{\tau})} \\ &= \frac{2(2\pi)^2}{{\omega_1'}^2}\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau')\right) + \frac{2\pi\iota}{{\omega_1'}^2(\tau' - \overline{\tau'})} \end{aligned}$$

Which rearranged becomes

$$\begin{aligned} &2(2\pi)^2\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) \\ &= 2(2\pi)^2\frac{\omega_1^2}{{\omega_1'}^2}\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau')\right) + 2\pi\iota\left(\frac{\omega_1}{{\omega_1'}^2}\frac{1}{(\tau' - \overline{\tau'})} - \frac{1}{\tau - \overline{\tau}}\right) \\ &= 2(2\pi)^2\frac{1}{(c\tau+d)^2}\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau')\right) + 2\pi\iota\left(\frac{1}{(c\tau+d)^2}\frac{1}{(\tau' - \overline{\tau'})} - \frac{1}{\tau - \overline{\tau}}\right) \end{aligned}$$

This can be rearranged again to

$$\begin{aligned} &2(2\pi)^2\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau)\right) \\ &= 2(2\pi)^2\frac{1}{(c\tau+d)^2}\left(\frac{1}{24} - \sum_{n=1}^{\infty} \sigma_1(n)e(n\tau')\right) - \frac{c\pi\iota}{c\tau + d} \end{aligned}$$

Viewing τ' as a function of τ, we have the relationship

$$\frac{\mathrm{d}\tau'}{\mathrm{d}\tau} = \frac{1}{(c\tau+d)^2}$$

which yields

$$2(2\pi)^2\left(\frac{1}{24}-\sum_{n=1}^{\infty}\sigma_1(n)e(n\tau)\right)$$
$$=2(2\pi)^2\left(\frac{1}{24}-\sum_{n=1}^{\infty}\sigma_1(n)e(n\tau')\right)\frac{\mathrm{d}\tau'}{\mathrm{d}\tau}-\frac{c\pi\iota}{c\tau+d}$$

Reversing the Lambert sum, we obtain

$$2(2\pi)^2\left(\frac{1}{24}-\sum_{n=1}^{\infty}\frac{ne(n\tau)}{1-e(n\tau)}\right)$$
$$=2(2\pi)^2\left(\frac{1}{24}-\sum_{n=1}^{\infty}\frac{ne(n\tau')}{1-e(n\tau')}\right)\frac{\mathrm{d}\tau'}{\mathrm{d}\tau}-\frac{c\pi\iota}{c\tau+d}$$

And for housekeeping, we divide by $4\pi\iota$ to resemble Rademacher's Form

$$2\pi\iota\left(\frac{1}{24}-\sum_{n=1}^{\infty}\frac{ne(n\tau)}{1-e(n\tau)}\right)$$
$$=2\pi\iota\left(\frac{1}{24}-\sum_{n=1}^{\infty}\frac{ne(n\tau')}{1-e(n\tau')}\right)\frac{\mathrm{d}\tau'}{\mathrm{d}\tau}+\frac{1}{2}\frac{c}{c\tau+d}$$

To keep track of the signs here, note that $2(2\pi)^2/(4\pi\iota)=8\pi^2/(4\pi\iota)=-2\pi\iota$.

Finally, we can now integrate this to obtain

$$\frac{\pi\iota}{12}\tau+\sum_{n=1}^{\infty}\log\left(1-e(n\tau)\right)$$
$$=\frac{\pi\iota}{12}\tau'+\sum_{n=1}^{\infty}\log(1-e(n\tau'))+\frac{1}{2}\log(c\tau+d)+K(a,b,c,d)$$

for a constant K independent of τ. Which when we apply the exponential function to gives us

$$e_{24}(\tau)\prod_{n=1}^{\infty}(1-e(n\tau))$$
$$=e_{24}(\tau')\prod_{n=1}^{\infty}(1-e(n\tau'))\exp\left(\frac{1}{2}\log(c\tau+d)\right)\exp(K)$$
$$=\exp(K)(c\tau+d)^{1/2}e_{24}(\tau')\prod_{n=1}^{\infty}(1-e(n\tau'))$$

Which is to say, if we define the function

$$\eta(\tau)=e_{24}(\tau)\prod_{n=1}^{\infty}(1-e(n\tau))$$

it transforms under the action of $\mathrm{SL}_2(\mathbb{Z})$ matrices $\gamma=(a,b;c,d)$ as follows

$$\eta(\gamma\tau)=\exp(-K)(c\tau+d)^{-1/2}\eta(\tau)$$

for some constant K depending only on γ.

3.2 Expansion of the Eta Function

We can obtain a series form of the Eta-function from the following

Lemma 3.1 (The q-expansion of the Eta-function).

$$\eta(\tau) = \sum_{n\geq 1} \left[\frac{12}{n}\right] e_{24}(n^2\tau) = \sum_{n\geq 1} \left[\frac{12}{n}\right] q^{n^2/24}$$

Therefore

$$\eta(\alpha\tau+\beta) = \sum_{n\geq 1} \left[\frac{12}{n}\right] e_{24}(n^2(\alpha\tau+\beta)) = \sum_{n\geq 1} \left[\frac{12}{n}\right] e_{24}(n^2\beta) q^{\alpha n^2/24}$$

Proof. Multiplying Euler's identity[2] (Eq. 2.1) by $q^{1/24}$ we obtain

$$\begin{aligned}
\eta(\tau) &= q^{1/24}\prod_{n\geq 1}(1-q^n) \\
&= \sum_{n\in\mathbb{Z}} (-1)^n q^{n(3n-1)/2+1/24} \\
&= \sum_{n\in\mathbb{Z}} (-1)^n q^{(6n-1)^2/24} \\
&= \sum_{n\in\mathbb{Z}} (-1)^n e_{24}((6n-1)^2\tau) \\
&= \sum_{n\leq 0} (-1)^n e_{24}((6n-1)^2\tau) + \sum_{n\geq 1} (-1)^n e_{24}((6n-1)^2\tau) \\
&= \sum_{n\geq 0} (-1)^n e_{24}((6n+1)^2\tau) + \sum_{n\geq 1} (-1)^n e_{24}((6n-1)^2\tau)
\end{aligned}$$

Rewriting the first sum into the form $\sum_{m\geq 1} \chi_1(m) e_{24}(m^2\tau)$ we require the character

$$\chi_1(m) = \begin{cases} (-1)^{(m-1)/6} & \text{if } m \equiv 1 \pmod 6 \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } m \equiv 1 \pmod{12} \\ -1 & \text{if } m \equiv 7 \pmod{12} \\ 0 & \text{else.} \end{cases}$$

The second sum can be rewritten to $\sum_{m\geq 1} \chi_2(m) e_{24}(m^2\tau)$ for the character

$$\chi_2(m) = \begin{cases} (-1)^{(m+1)/6} & \text{if } m \equiv -1 \pmod 6 \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } m \equiv -1 \pmod{12} \\ -1 & \text{if } m \equiv -7 \pmod{12} \\ 0 & \text{else.} \end{cases}$$

So we have that $\eta(\tau) = \sum_{m\geq 1}(\chi_1(m)+\chi_2(m)) e_{24}(m^2\tau)$. Writing out the sum

$$\begin{aligned}
\chi_1(m)+\chi_2(m) &= \begin{cases} 1 & \text{if } m \equiv \pm 1 \pmod{12} \\ -1 & \text{if } m \equiv \pm 7 \pmod{12} \\ 0 & \text{else.} \end{cases} \\
&= \left[\frac{12}{m}\right]
\end{aligned}$$

gives us the claimed expansion. □

[2] A direct consequence of Jacobi's triple product identity.

3.3 Transformation properties of the Eta-function

From our motivation of the Eta-function, we glean that for $(a,b;c,d)$ in $\mathrm{SL}_2(\mathbb{Z})$ and $\tau' = (a,b;c,d)\tau$ we have the functional equation $\eta(\tau') = K'\sqrt{c\tau+d}\,\eta(\tau)$. The dependence of K' on the transformation matrix $(a,b;c,d)$ is called the multiplier system of η and we denote it by $v_\eta((a,b;c,d))$.

Lemma 3.2 (The multiplier system of η)**.** *The multiplier system of the Dedekind-Eta function for $\mathrm{SL}_2(\mathbb{Z})$ matrices is is given by*

$$v_\eta\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right) = \begin{cases} [d/c]\, e_{24}\left(-3c + bd(1-c^2) + c(a+d)\right) & c > 0 \text{ odd}, \\ [-d/-c]\, e_{24}\left(3c - 6 + bd(1-c^2) + c(a+d)\right) & c < 0 \text{ odd}, \\ [c/d]\, e_{24}\left(3d - 3 + ac(1-d^2) + d(b-c)\right) & c \geq 0, d \text{ odd}, \\ [-c/d]\, e_{24}\left(-3d - 9 + ac(1-d^2) + d(b-c)\right) & c < 0, d \text{ even}. \end{cases}$$

$$= \begin{cases} e_{24}\left((a+d)c - bd(c^2-1) - 3c\right)[d/c] & c > 0 \text{ odd}, \\ e_{24}\left((a+d)c - bd(c^2-1) - 3c\right)[-d/-c]\, e_{24}(6(c-1)) & c < 0 \text{ odd}, \\ e_{24}\left(-3d - 9 + ac(1-d^2) + d(b-c)\right)[c/d]\, e_{24}(6(d+1)) & c \geq 0, d \text{ odd}, \\ e_{24}\left(-3d - 9 + ac(1-d^2) + d(b-c)\right)[-c/d] & c < 0, d \text{ even}. \end{cases}$$

We will not prove this theorem. Instead we will show two special cases.

Theorem 3.3. [Ono04, Th. 1.61, p. 17] *For τ in the upper complex half-plane, the Dedekind-Eta function satisfies*

$$\eta(\tau+1) = e_{24}(1)\eta(\tau)$$
$$\eta(-1/\tau) = \sqrt{-\iota\tau}\,\eta(\tau)$$

Proof. The first functional equation is shown through straightforward algebraic manipulations. Indeed

$$\begin{aligned} \eta(\tau+1) &= \exp\left(\frac{1}{12}\pi\iota(\tau+1)\right)\prod_{n\geq 1}(1-\exp(2n\pi\iota(\tau+1))) \\ &= \exp\left(\frac{1}{12}\pi\iota\right)\exp\left(\frac{1}{12}\pi\iota\tau\right)\prod_{n\geq 1}(1-\exp(2n\pi\iota)\exp(2n\pi\iota\tau)) \\ &= \exp\left(\frac{1}{12}\pi\iota\right)\exp\left(\frac{1}{12}\pi\iota\tau\right)\prod_{n\geq 1}(1-\exp(2n\pi\iota\tau)) \\ &= \exp\left(\frac{1}{12}\pi\iota\right)\eta(\tau) \\ &= e_{24}(1)\eta(\tau). \end{aligned}$$

The second is a little more involved. We will Follow Siegel's proof [Sie54] as presented in [Apo91, Th. 3.1, p. 48].

By the identity theorem of holomorphic functions, it suffices to show the second functional equation on the positive imaginary axis. So let $\tau = y\iota$ with $y > 0$. Then the equation to prove becomes $\eta(\iota/y) = \sqrt{y}\eta(\iota y)$, which when we apply the logarithm becomes $\log(\eta(\iota/y)) =$

$\log(y)/2 + \log(\eta(\iota y))$. After investigating

$$\begin{aligned}
\log(\eta(\tau)) &= \log\left(\exp\left(\frac{1}{12}\pi\iota\tau\right)\prod_{n\geq 1}(1-\exp(2n\pi\iota\tau))\right)\\
&= \frac{\pi\iota}{12}\tau + \sum_{n\geq 1}\log(1-\exp(2n\pi\iota\tau))\\
&= \frac{\pi\iota}{12}\tau + \sum_{n\geq 1}\sum_{k\geq 1}\frac{1}{k}\exp(2\pi\iota kn\tau)\\
&= \frac{\pi\iota}{12}\tau + \sum_{n\geq 0}\sum_{k\geq 1}\frac{1}{k}\exp(2\pi\iota k\tau)\exp(2\pi\iota kn\tau)\\
&= \frac{\pi\iota}{12}\tau + \sum_{k\geq 1}\frac{1}{k}\exp(2\pi\iota k\tau)\sum_{n\geq 0}\exp(2\pi\iota kn\tau)\\
&= \frac{\pi\iota}{12}\tau + \sum_{k\geq 1}\frac{1}{k}\exp(2\pi\iota k\tau)\frac{1}{1-\exp(2\pi\iota k\tau)}\\
&= \frac{\pi\iota}{12}\tau + \sum_{k\geq 1}\frac{1}{k}\frac{1}{\exp(-2\pi\iota k\tau)-1}
\end{aligned}$$

we see this amounts to proving

$$\begin{aligned}
\frac{1}{2}\log(y) &= \log\left(\eta\left(\frac{\iota}{y}\right)\right) - \log(\eta(\iota y))\\
&= \frac{\pi\iota}{12}\left(\frac{\iota}{y} - y\iota\right) + \sum_{k\geq 1}\frac{1}{k}\left(\frac{1}{\exp(-2\pi\iota k\iota/y)-1} - \frac{1}{\exp(-2\pi\iota k\iota y)-1}\right)\\
&= \frac{\pi}{12}\left(y - \frac{1}{y}\right) + \sum_{k\geq 1}\frac{1}{k}\left(\frac{1}{\exp(2\pi k/y)-1} - \frac{1}{\exp(2\pi k y)-1}\right)
\end{aligned}$$

If we now consider the function $f_N(z) = z^{-1}\cot(\pi\iota Nz)\cot(\pi Nz/y)$, with $N = n + 1/2$, we see that it has the following poles and residues: at the origin, f_N has a pole of order 3 with residue $-(\iota/3)(y - 1/y)$; at $z = \iota k/N$ for (non-zero) integral k, f_N has simple poles with residues $-\cot(k\pi\iota/y)/\pi k$; and at $z = ky/N$ for (non-zero) k integral, f_N has simple poles with residues $-\cot(k\pi\iota y)/\pi k$.

Calculating the residue at the origin is a tedious calculation, but the other two residues are manageable since they are simple poles: at $z = \iota k/N$, the residue is

$$\begin{aligned}
&\mathrm{Res}_{\iota k/N}(f_N)\\
&= \lim_{z\longrightarrow \iota k/N} (z - \iota k/N) z^{-1} \cot(\pi \iota N z) \cot(\pi N z/y)\\
&= \lim_{z\longrightarrow \iota k/N} z^{-1} \cot(\pi N z/y) \cos(\pi \iota N z) \lim_{z\longrightarrow \iota k/N} \frac{z - \iota k/N}{\sin(\pi \iota N z)}\\
&= \frac{N}{\iota k} \cot(\pi \iota k/y) \cos(-\pi k) \lim_{z\longrightarrow \iota k/N} \frac{z - \iota k/N}{\sin(\pi \iota N z)}\\
&= \frac{N}{\iota k} \cot(\pi \iota k/y)(-1)^k \lim_{z\longrightarrow \iota k/N} \frac{z - \iota k/N}{\sin(-\pi k) + \pi \iota N \cos(-\pi k)(z - \iota k/N) + O((z - \iota k/N)^2)}\\
&= \frac{N}{\iota k} \cot(\pi \iota k/y)(-1)^k \lim_{z\longrightarrow \iota k/N} \frac{z - \iota k/N}{\pi \iota N(-1)^k (z - \iota k/N) + O((z - \iota k/N)^2)}\\
&= \frac{N}{\iota k} \cot(\pi \iota k/y)(-1)^k \lim_{z\longrightarrow \iota k/N} \frac{1}{\pi \iota N(-1)^k + O(z - \iota k/N)}\\
&= \frac{N}{\iota k} \cot(\pi \iota k/y)(-1)^k \frac{1}{\pi \iota N(-1)^k}\\
&= -\frac{1}{\pi k} \cot(\pi \iota k/y);
\end{aligned}$$

and at $z = ky/N$, the residue is

$$\begin{aligned}
&\mathrm{Res}_{ky/N}(f_N)\\
&= \lim_{z\longrightarrow ky/N} (z - ky/N) z^{-1} \cot(\pi \iota N z) \cot(\pi N z/y)\\
&= \frac{N}{ky} \cot(\pi \iota k y) \cos(\pi k) \lim_{z\rightarrow ky/N} \frac{z - ky/N}{\sin(\pi N z/y)}\\
&= \frac{N}{ky} \cot(\pi \iota k y)(-1)^k \lim_{z\rightarrow ky/N} \frac{z - ky/N}{\pi N/y \cos(\pi k)(z - ky/N) + O((z - ky/N)^2)}\\
&= \frac{N}{ky} \cot(\pi \iota k y)(-1)^k \lim_{z\rightarrow ky/N} \frac{z - ky/N}{\pi N/y \cos(\pi k)(z - ky/N) + O((z - ky/N)^2)}\\
&= \frac{1}{\pi k} \cot(\pi \iota k y).
\end{aligned}$$

Now we notice that for any $z \neq k\pi$ in $\mathbb{C}$ we have

$$\frac{\cot(z)}{\iota} = \frac{\cos(z)}{\iota \sin(z)} = \frac{\exp(\iota z) + \exp(-\iota z)}{\exp(\iota z) - \exp(-\iota z)} = \frac{\exp(2\iota z) + 1}{\exp(2\iota z) - 1} = 1 + \frac{2}{\exp(2\iota z) - 1}.$$

Hence for arbitrary z, z' we can write

$$\frac{1}{\iota} (\cot(z) - \cot(z')) = 2 \left(\frac{1}{\exp(2\iota z) - 1} - \frac{1}{\exp(2\iota z') - 1} \right).$$

Therefore, if we consider the rhombus R with vertices $1, \iota y, -1, -\iota y$ we obtain with the residue theorem

$$
\begin{aligned}
\int_R f_N(z)\,\mathrm{d}\,z &= 2\pi\iota\left(-\frac{\iota}{3}\left(y-\frac{1}{y}\right)+\sum_{\substack{-n\le k\le n\\ k\neq 0}}\frac{1}{\pi k}\left(\cot(k\pi\iota y)-\cot\left(\frac{k\pi\iota}{y}\right)\right)\right)\\
&= 2\pi\iota\left(-\frac{\iota}{3}\left(y-\frac{1}{y}\right)+2\sum_{1<k\le n}\frac{1}{\pi k}\left(\cot(k\pi\iota y)-\cot\left(\frac{k\pi\iota}{y}\right)\right)\right)\\
&= \frac{2}{3}\pi\left(y-\frac{1}{y}\right)+4\sum_{1<k\le n}\frac{\iota}{k}\left(\cot(k\pi\iota y)-\cot\left(\frac{k\pi\iota}{y}\right)\right)\\
&= \frac{2}{3}\pi\left(y-\frac{1}{y}\right)-4\sum_{1<k\le n}\frac{1}{k\iota}\left(\cot(k\pi\iota y)-\cot\left(\frac{k\pi\iota}{y}\right)\right)\\
&= \frac{2}{3}\pi\left(y-\frac{1}{y}\right)-8\sum_{1<k\le n}\frac{1}{k}\left(\frac{1}{\exp(2k\pi y)-1}-\frac{1}{\exp(2k\pi/y)-1}\right)\\
&= \frac{2}{3}\pi\left(y-\frac{1}{y}\right)+8\sum_{1<k\le n}\frac{1}{k}\left(\frac{1}{\exp(2k\pi/y)-1}-\frac{1}{\exp(2k\pi y)-1}\right)\\
&= 8\left(\frac{1}{12}\pi\left(y-\frac{1}{y}\right)+\sum_{1<k\le n}\frac{1}{k}\left(\frac{1}{\exp(2k\pi/y)-1}-\frac{1}{\exp(2k\pi y)-1}\right)\right).
\end{aligned}
$$

Consequently, all that needs to be shown is that $\lim_{N\to\infty}\int_R f_N(z)/8\,\mathrm{d}\,z = \log(y)/2$. To that end, notice that $\lim_{N\to\infty} zf_N(z) = 1$ on the edges of R connecting ι, y and $-y, -\iota$, and $\lim_{N\to\infty} zf_N(z) = -1$ on the other two edges. Then using the theorem of bounded convergence ($zF_N(z)$ is bounded on the edges of the rhombus) we obtain

$$
\begin{aligned}
\lim_{N\to\infty}\int_R F_N(z)\,\mathrm{d}\,z &= \int_R\left(\lim_{N\to\infty} zF_N(z)\right)\cdot\frac{1}{z}\,\mathrm{d}\,z\\
&= \left(\int_y^\iota - \int_\iota^{-y} + \int_{-y}^{-\iota} - \int_{-\iota}^{y}\right)\frac{1}{z}\,\mathrm{d}\,z\\
&= 2\left(\int_y^\iota - \int_{-\iota}^{y}\right)\frac{1}{z}\,\mathrm{d}\,z\\
&= 2(\log(\iota)-\log(y)-(\log(y)-\log(-\iota)))\\
&= 4\log(y)-2(\log(\iota)+\log(-\iota))\\
&= 4\log(y).
\end{aligned}
$$

Therefore $\lim_{N\to\infty}\int_R f_N(z)/8\,\mathrm{d}\,z = \log(y)/2$ and we have completed the proof. □

4 Eta products

Let $N \geq 1$ be integral, and let r_δ be an integer for each positive divisor δ of N. The arising *eta product* of *level* N is

$$\prod_{\delta|N} \eta(\delta\tau)^{r_\delta}.$$

4.1 Transformation properties of eta products

Theorem 4.1. [Ono04, Th. 1.64, p. 18] *Let $f(\tau) = \prod_{\delta|N} \eta(\delta\tau)^{r_\delta}$ be an eta product of level N such that*

$$\sum_{\delta|N} r_\delta \equiv 0 \pmod 2, \quad \sum_{\delta|N} \delta r_\delta \equiv 0 \pmod{24}, \text{ and } \sum_{\delta|N} \frac{N}{\delta} r_\delta \equiv 0 \pmod{24}.$$

Then f is a modular form of level $\Gamma_0(N)$, weight $k = (1/2)\sum_\delta r_\delta$ and multiplier system

$$\chi\colon \Gamma_0(N) \to \mathbb{R}; \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \left[\frac{(-1)^k \prod_\delta \delta^{r_\delta}}{d}\right] = \left[\frac{-1}{d}\right]^k \prod_\delta \left[\frac{\delta}{d}\right]^{r_\delta}.$$

That is, f satisfies the transformation property

$$f\left(\frac{a\tau + b}{c\tau + d}\right) = \chi(d)(c\tau + d)^k f(\tau)$$

on the complex upper half-plane for all matrices $(a, b; c, d)$ in $\Gamma_0(N)$.

Proof. Before we begin with the proof proper, we notice that

$$\left[\frac{-1}{d}\right]^k = \begin{cases} 1 & d \equiv 1 \pmod 4 \text{ or } k \equiv 0 \pmod 2 \\ -1 & \text{else} \end{cases}$$
$$= \begin{cases} 1 & d \equiv 1 \pmod 4 \text{ or } \sum_\delta r_\delta \equiv 0 \pmod 4 \\ -1 & \text{else} \end{cases}$$

Now let let us begin by naively multiplying out the definition of f.

$$\begin{aligned} f\left(\frac{az+b}{cz+d}\right) &= \prod_\delta \eta\left(\frac{\delta az + \delta b}{cz+d}\right)^{r_\delta} \\ &= \prod_\delta \left(v_\eta((a, \delta b; c/\delta, d))\sqrt{c\tau + d}\eta(\delta\tau)\right)^{r_\delta} \\ &= (c\tau + d)^k f(z) \prod_\delta v_\eta((a, \delta b; c/\delta, d))^{r_\delta}. \end{aligned}$$

By Lemma 1.1, it suffices to verify

$$v_f((a, b\delta; c/\delta, d)) \overset{\text{def.}}{=} \prod_\delta v_\eta((a, \delta b; c/\delta, d))^{r_\delta} \overset{!}{=} \chi(d)$$

for $c/\delta, d$ positive and $d \equiv 1 \pmod 4$. Since δ is chosen to be positive, this is equivalent to c, d

positive and $d \equiv 1 \pmod 4$. In that case, Lemma 3.2 tells us that

$$\begin{aligned}v_f((a, b\delta, c/\delta, d)) &= \prod_\delta e_{24}\left(r_\delta\left(3d - 3 + a\left[\frac{c}{\delta}\right](1-d^2) + d\left(\delta b - \frac{c}{\delta}\right)\right)\right)\left[\frac{c/\delta}{d}\right]^{r_\delta} \\ &= \prod_\delta e_{24}\left(r_\delta\left(3d - 3 + a\left[\frac{c}{\delta}\right](1-d^2) + d\left(\delta b - \frac{c}{\delta}\right)\right)\right)\prod_\delta\left[\frac{c/\delta}{d}\right]^{r_\delta}.\end{aligned}$$

The first product is

$$\begin{aligned}&\prod_\delta e_{24}\left(r_\delta\left(3d - 3 + a\left[\frac{c}{\delta}\right](1-d^2) + d\left(\delta b - \frac{c}{\delta}\right)\right)\right) \\ &= \prod_\delta e_{24}\left(3r_\delta(d-1) + \frac{r_\delta}{\delta}ac(1-d^2) + r_\delta \delta db - \frac{r_\delta}{\delta}dc\right) \\ &= e_{24}\left(3(d-1)\sum_\delta r_\delta + ac(1-d^2)\sum_\delta \frac{r_\delta}{\delta} + db\sum_\delta r_\delta\delta - dc\sum_\delta\frac{r_\delta}{\delta}\right) \\ &= e_{24}\left(-3(d-1)\sum_\delta r_\delta\right) \\ &= \begin{cases}1 & (d-1)\sum_\delta r_\delta \equiv 0 \pmod 8 \\ -1 & \text{else}\end{cases} \\ &= \begin{cases}1 & d \equiv 1 \pmod 4 \text{ or } \sum_\delta r_\delta \equiv 0 \pmod 4 \\ -1 & \text{else}\end{cases} \\ &= \left[\frac{-1}{d}\right]^k.\end{aligned}$$

In the penultimate equality we used the fact that d is odd, and that $\sum_\delta r_\delta \equiv 0 \pmod 2$.

For the second product, we note that δ is coprime to d because δ is a divisor of N which in turn divides c. Therefore $[\delta/d] = \pm 1$ (importantly, non-zero) and by the multiplicativity of the Kronecker symbol, we have that

$$\left[\frac{c/\delta}{d}\right] = \left[\frac{c}{d}\right]\left[\frac{\delta}{d}\right].$$

So the product over the Kronecker symbols is

$$\prod_\delta\left[\frac{c/\delta}{d}\right]^{r_\delta} = \prod_\delta\left[\frac{c}{d}\right]^{r_\delta}\left[\frac{\delta}{d}\right]^{r_\delta} = \left[\frac{c}{d}\right]^{\sum_\delta r_\delta}\prod_\delta\left[\frac{\delta}{d}\right]^{r_\delta} = \prod_\delta\left[\frac{\delta}{d}\right]^{r_\delta}$$

and we can conclude that

$$v_f((a, b\delta; c/\delta, d)) = \left[\frac{-1}{d}\right]^k\prod_\delta\left[\frac{\delta}{d}\right]^{r_\delta} = \chi(d).$$

completing the proof. □

4.2 Expansions of Eta products

Proposition 4.2 (Expansion of $\eta(\delta\tau)$ at a cusp)**.** [Köh11, Prop. 2.1,p. 34] *Let $f_\delta(\tau) = \eta(\delta\tau)$ with $\delta \geq 1$ integral and let $r = a/c$ be a fraction in reduced form representing a cusp.*

Further, let b, d be integers chosen such that $M = (a, b; c, d)$ lies in $\mathrm{SL}_2(\mathbb{Z})$. Then we have with $g \overset{\text{def.}}{=} \gcd(c, \delta)$ and $A \overset{\text{def.}}{=} (x, y; -c/g, \delta a/g)$ in $\mathrm{SL}_2(\mathbb{Z})$

$$f_\delta(M\tau) = v_\eta(A^{-1})\Big(\frac{g}{\delta}(c\tau + d)\Big)^{1/2} \eta\left(\frac{g^2}{\delta}\tau + \frac{g(x\delta b + yd)}{\delta}\right).$$

Re-writing this, with $\nu = x\delta b + yd$ we obtain the q-expansion of f_δ at r

$$f_\delta(M\tau) = v_\eta(A^{-1})\Big(\frac{g}{\delta}(c\tau + d)\Big)^{1/2} \sum_{n=1}^{\infty} \left[\frac{12}{n}\right] e_{24}\left(\frac{n^2\nu g}{\delta}\right) q^{n^2 g^2/24\delta}$$

In particular, we have that the order of vanishing of f_δ at the cusp r is

$$\mathrm{ord}_r(f_\delta) = \frac{g^2}{24\delta} = \frac{1}{24\delta}\gcd(c, \delta)^2$$

Proof. Since r is a fraction in reduced form, a, c are coprime and Bézout's identity delivers us b, d such that M lies in $\mathrm{SL}_2(\mathbb{Z})$. We see that $M(\infty) = a/c = r$. Therefore

$$f_\delta(M\tau) = \eta(\delta M\tau) = \eta\left(\delta\frac{a\tau + b}{c\tau + a}\right) = \eta\left(\frac{\delta a\tau + \delta b}{c\tau + a}\right) = \eta(M'\tau)$$

for $M' = (\delta a, \delta b; c, d)$. As described in the preliminaries, the expansion of a weakly modular form g in $M^!_{k,\nu}(\Gamma)$ at a cusp r is the expansion of $e((\theta - \Lambda)\tau)(\tau - r)^k g(P\tau)$ where $P\infty = r$, $e(\theta) = \nu(T^h)$, $e(\Lambda) = \nu(QT^hQ)$ and QT^hQ lies in Γ. Since $T = (1, 1; 0, 1)$ already lies in $\Gamma_0(N)$, we know that $e(\Lambda) = e(\theta)$ and we only need to calculate the expansion of $(\tau - r)^k f(\delta M\tau) = (\tau - r)^k f(M'\tau)$.

We now want to use our knowledge of the expansion of $\eta(\alpha\tau + \beta)$ at infinity (Theorem 3.1). We do this by finding an *auxiliary* $\mathrm{SL}_2(\mathbb{Z})$ matrix $A = (x, y; u, v)$ so that AN has vanishing lower left entry. Indeed, then

$$\begin{pmatrix} x & y \\ u & v \end{pmatrix}\begin{pmatrix} \delta a & \delta b \\ c & d \end{pmatrix} \overset{!}{=} \begin{pmatrix} x\delta a + yc & x\delta b + yd \\ 0 & u\delta b + vd \end{pmatrix} \quad \text{so} \quad AN\tau = \frac{x\delta a + yc}{u\delta b + vd}\tau + \frac{x\delta b + yd}{u\delta b + vd}$$

is of the form $\alpha\tau + \beta$. Then we can use the expansion of $\eta(\alpha\tau + \beta)$ in

$$\begin{aligned} f_\delta(M\tau) &= \eta(N\tau) \\ &= \eta(A^{-1}AN\tau) \\ &= v_\eta(A^{-1})(-uAN\tau + x)^{1/2}\eta(AN\tau) \end{aligned}$$

To find A, consider now

$$\begin{pmatrix} x & y \\ u & v \end{pmatrix}\begin{pmatrix} \delta a & \delta b \\ c & d \end{pmatrix} = \begin{pmatrix} x\delta a + yc & x\delta b + yd \\ u\delta a + vc & u\delta b + vd \end{pmatrix}.$$

With $g \overset{\text{def.}}{=} \gcd(c, \delta a) = \gcd(c, \delta)$, the choice

$$v = \frac{\delta a}{g}, \quad u = -\frac{c}{g}$$

delivers coprime integers that satisfy $u\delta a + vc = 0$. Furthermore, the other entries of AN can be simplified to

$$\begin{aligned}
\text{(top left)} \quad & x\delta a + yc = xvg - yug = g(xv - yu) = g \\
\text{(bottom right)} \quad & u\delta b + vd = -\frac{c}{g}\delta b + \frac{\delta a}{g} d = \frac{\delta}{g}(-bc + ad) = \frac{\delta}{g}.
\end{aligned}$$

However, the top right entry cannot be worked any further. We call this $\nu \overset{\text{def.}}{=} x\delta b + yd$ to obtain $AN\tau = g^2\tau/\delta + \nu g/\delta$. Now we can calculate

$$\begin{aligned}
f_\delta(M\tau) = \cdots &= v_\eta(A^{-1})(-uAN\tau + x)^{1/2}\eta(AN\tau) \\
&= v_\eta(A^{-1})\left(-u\left(\frac{g^2}{\delta}\tau + \frac{\nu g}{\delta}\right) + x\right)^{1/2} \eta\left(\frac{g^2}{\delta}\tau + \frac{\nu g}{\delta}\right) \\
&= v_\eta(A^{-1})\left(\frac{-ug}{\delta}(g\tau + \nu) + x\right)^{1/2} \eta\left(\frac{g^2}{\delta}\tau + \frac{\nu g}{\delta}\right) \\
&= v_\eta(A^{-1})\left(\frac{c}{\delta}(g\tau + \nu) + x\right)^{1/2} \eta\left(\frac{g^2}{\delta}\tau + \frac{\nu g}{\delta}\right)
\end{aligned}$$

and working on the square root

$$\begin{aligned}
c(g\tau + \nu) + \delta x &= cg\tau + c\nu + \delta x \\
&= cg\tau + c(x\delta b + yd) + \delta x \\
&= cg\tau + cx\delta b + cyd + \delta x \\
&= cg\tau + x\delta(bc + 1) + cyd \\
&= cg\tau + x\delta ad + cyd \\
&= cg\tau + d(x\delta a + cy) \\
&= cg\tau + d(xvg - ugy) \\
&= cg\tau + dg(xv - uy) \\
&= cg\tau + dg \\
&= g(c\tau + d)
\end{aligned}$$

we obtain

$$f_\delta(M\tau) = v_\eta(A^{-1})\left(\frac{g}{\delta}(c\tau + d)\right)^{1/2} \eta\left(\frac{g^2}{\delta}\tau + \frac{\nu g}{\delta}\right).$$

Finally, using Theorem 3.1 we get the result

$$f_\delta(M\tau) = v_\eta(A^{-1})\left(\frac{g}{\delta}(c\tau + d)\right)^{1/2} \sum_{n=1}^{\infty} \left[\frac{12}{n}\right] e_{24}\left(\frac{n^2}{\delta}(g^2\tau + \nu g)\right)$$

proving the first assertion.

We note that $[12/0] = 0$, so the first non-vanishing term in the sum is $e_{24}(\nu g/\delta)e_{24}(g^2\tau/\delta)$. Therefore the order is indeed $g^2/(24\delta)$, proving the second claim. □

Corollary 4.3 (Expansion of an eta product)**.** *Let $f(\tau) = \prod_{\delta|N} \eta(\delta\tau)^{r_\delta}$ be an eta product and $r = a/c$ a fraction in reduced form representing a cusp. Let $M = (a, b; c, d)$ be a matrix in $\mathrm{SL}_2(\mathbb{Z})$. Then with $k = (1/2)\sum_\delta r_\delta$, $g_\delta = \gcd(c, \delta)$, $A_\delta = (x_\delta, y_\delta, -c/\delta, \delta a/g_\delta)$ in $\mathrm{SL}_2(\mathbb{Z})$ and*

$\nu_\delta = x_\delta \delta b + y_\delta d$ *we have*

$$\begin{aligned} f(M\tau) &= \prod_\delta \eta(\delta M\tau)^{r_\delta} \\ &= \prod_\delta v_\eta(A_\delta^{-1})^{r_\delta} \left(\frac{g_\delta}{\delta}(c\tau+d)\right)^{r_\delta/2} \eta\left(\frac{g_\delta^2}{\delta}\tau + \frac{\nu_\delta g_\delta}{\delta}\right)^{r_\delta} \\ &= (c\tau+d)^k \prod_\delta v_\eta(A_\delta^{-1})^{r_\delta} \left(\frac{g_\delta}{\delta}\right)^{r_\delta/2} \eta\left(\frac{g_\delta^2}{\delta}\tau + \frac{\nu_\delta g_\delta}{\delta}\right)^{r_\delta}. \end{aligned}$$

Theorem 4.4. [Köh11, Cor. 2.2, p. 36] *Let c, d and N be positive integers with d dividing N and c, d coprime. If f is is an eta product for N satisfying the conditions of Theorem 4.1, then the order of vanishing of f at the cusp c/d is*

$$\frac{N}{24} \sum_{\delta | N} \frac{\gcd(d,\delta)^2}{\delta} \cdot r_\delta$$

4.3 Atkin-Lehner operators and Fricke Involutions

Let $N \geq 1$ be an integer and Q an exact divisor thereof. That is, all powers of a prime p that divide N also divide Q. We define the *Atkin-Lehner* operator on $M_k(\Gamma_0)$ for integral k to be the slash operation with a matrix of the form

$$W(Q) = \begin{pmatrix} Q\alpha & \beta \\ N\gamma & Q\delta \end{pmatrix} \quad \text{with integral entries, having} \quad \det(W(Q)) = Q.$$

Furthermore, we define the *Fricke involution* to be operation of slashing with the matrix $W(N) = (0, -1; N, 0)$. If we only consider exact divisors $Q_p = p^k$ that are the power of some prime, this definition matches that of Ono in [Ono04, Def. 2.19, pg. 27].

For this definition to be sensible, we must verify that it is independent of the choice of $\alpha, \beta, \gamma, \delta$. To that end, note that any Atkin-Lehner matrix can be decomposed into the product

$$W(Q) = \begin{pmatrix} Q\alpha & \beta \\ N\gamma & Q\delta \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ N\gamma/Q & Q\delta \end{pmatrix} \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}$$

where importantly, $R(Q) \stackrel{\text{def.}}{=} (\alpha, \beta; N\gamma/Q, Q\delta)$ has determinant $Q\alpha\delta - \beta N\gamma/Q = (Q^2\alpha\delta - \beta N\gamma)/Q = Q/Q = 1$, and integral entries, since Q divides N. So we can use the transformation properties of f under $\Gamma_0(N)$ matrices to evaluate the expression

$$\begin{aligned} f|_k W(Q)(\tau) &= Q^k(N\gamma\tau + Q\delta)^{-k} f(W\tau) \\ &= Q^k(N\gamma\tau + Q\delta)^{-k} f(R(Q)(Q,0;0,1)\tau) \\ &= Q^k(N\gamma\tau + Q\delta)^{-k} f(R(Q)Q\tau) \\ &= Q^k(N\gamma\tau + Q\delta)^{-k}(N\gamma/Q(Q\tau) + Q\delta)^k f(Q\tau) \\ &= Q^k f(Q\tau) \end{aligned}$$

and conclude that the Atkin-Lehner operator clearly only depends on Q.

Proposition 4.5. [Ono04, Prop. 2.21, pg. 27] *The Atkin-Lehner operators (and Fricke* *involutions) are involutions on $M_k(\Gamma_0(N))$.*

Proof. Since k is integral, the slash operator defines a group action. Calculating

$$\begin{aligned} W(Q)^2 &= \begin{pmatrix} Q\alpha & \beta \\ N\gamma & Q\delta \end{pmatrix}^2 \\ &= \begin{pmatrix} Q^2\alpha^2 + \beta N\gamma & Q\alpha\beta + \beta Q\delta \\ N\gamma Q\alpha + Q\delta N\gamma & N\gamma\beta + Q^2\delta^2 \end{pmatrix} \\ &= \begin{pmatrix} Q\alpha^2 + \beta N/Q\gamma & \alpha\beta + \beta\delta \\ N(\gamma\alpha + \delta\gamma) & N/Q\gamma\beta + Q\delta^2 \end{pmatrix} \begin{pmatrix} Q & 0 \\ 0 & Q \end{pmatrix} \end{aligned}$$

The left matrix, call this γ, lies in $\Gamma_0(N)$. Now

$$\begin{aligned} f|_k(\gamma(Q,0;0,Q))(\tau) &= (f|_k(Q,0;0,Q))|_k\gamma(\tau) \\ &= (\det((Q,0;0,Q))^{k/2} j(\tau,(Q,0;0,Q))^k f)|_k\gamma(\tau) \\ &= f|_k\gamma \\ &= f \end{aligned}$$

□

Lemma 4.6 (The Atkin-Lehner operators send eta products to eta products)**.** *Let N be a positive integer and Q an exact divisor of N. Let r_δ be an integer for every divisor δ of N such that $k = \sum_\delta r_\delta/2$ is integral. Then*

$$\left(\prod_{\delta|N} \eta(\delta\tau)^{r_\delta}\right) |_k W_Q = \left(Q^k \prod_{\delta|N} \nu_\eta(A_\delta) {g_\delta}^{-r_\delta/2}\right) \prod_{\delta|N} \eta((\delta * Q)\tau)^{r_\delta}$$

where

$$W_Q = \begin{pmatrix} Qa & b \\ Nc & Qd \end{pmatrix}, \quad A_\delta = \begin{pmatrix} g_\delta a & b\delta/g_\delta \\ Ncg_\delta/Q\delta & dQ/g_\delta \end{pmatrix},$$

$g_\delta = \gcd(\delta, Q)$ and $a * b = ab/\gcd(a,b)^2$.

Proof. Before we begin, we notice that if l, m are divisors of n, then $(l * m)$ is again a divisor of n. Therefore we really do obtain an eta product of level N.

We have by definition that

$$\prod_{\delta|N} \eta(\delta\tau)^{r_\delta}|_k W_Q \overset{\text{def.}}{=} Q^k (Nc\tau + Qd)^{-k} \prod_{\delta|N} \eta(\delta W_Q \tau)^{r_\delta}.$$

Since $\det(\delta W_Q) = \delta Q$, we cannot directly pull this factor out of the argument of η. Using the same decomposition idea from $W_Q = R_Q Q$ we want to decompose $\delta W_Q = \delta R_Q Q$. At this point we recall the point made in the introduction, that $\delta W_Q \neq W_Q \delta$.

Trying

$$\delta R_Q = \begin{pmatrix} \delta a & \delta b \\ Nc/Q & dQ \end{pmatrix} = \begin{pmatrix} a & \delta b \\ Nc/\delta Q & dQ \end{pmatrix} \begin{pmatrix} \delta & 0 \\ 0 & 1 \end{pmatrix}$$

or

$$\delta R_Q = \begin{pmatrix} \delta a & \delta b \\ Nc/Q & dQ \end{pmatrix} = \begin{pmatrix} \delta a & b \\ Nc/Q & dQ/\delta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \delta \end{pmatrix}$$

may not yield integral matrices. However, this inspires the approach of splitting $\delta = \delta_1\delta_2$ into the product of integers δ_1, δ_2 and writing

$$\delta R_Q = \begin{pmatrix} \delta a & \delta b \\ Nc/Q & dQ \end{pmatrix} = \begin{pmatrix} \delta_1 a & \delta_2 b \\ Nc/Q\delta_2 & dQ/\delta_1 \end{pmatrix} \begin{pmatrix} \delta_2 & 0 \\ 0 & \delta_1 \end{pmatrix}.$$

Now we can choose $\delta_1 = \gcd(\delta, Q) \overset{\text{def.}}{=} g$ and $\delta_2 = \delta/g$ to ensure that $N/Q\delta_2$ and Q/δ_1 are integers. So we have the decomposition

$$\delta R_Q = \begin{pmatrix} \delta a & \delta b \\ Nc/Q & dQ \end{pmatrix} = \underbrace{\begin{pmatrix} ga & b\delta/g \\ Ncg/Q\delta & dQ/g \end{pmatrix}}_{\overset{\text{def.}}{=} A} \begin{pmatrix} \delta/g & 0 \\ 0 & g \end{pmatrix}.$$

Clearly, $\det(A) = 1$. Now using the notation $a * b = ab/\gcd(a, b)$ we have

$$\begin{aligned} \eta(\delta R_Q Q\tau) &= \eta\left(A \begin{pmatrix} \delta/g & 0 \\ 0 & g \end{pmatrix} Q\tau\right) \\ &= \eta\left(A \frac{\delta Q}{g^2}\tau\right) \\ &= \eta\left(A(\delta * Q)\tau\right) \\ &= \nu_\eta(A)\left(\frac{Ncg}{Q\delta}(d * Q)\tau + \frac{dQ}{g}\right)^{1/2} \eta((\delta * Q)\tau) \\ &= \nu_\eta(A)\left(\frac{Ncg}{Q\delta}\frac{\delta Q}{g^2}\tau + \frac{dQ}{g}\right)^{1/2} \eta((\delta * Q)\tau) \\ &= \nu_\eta(A) g^{-1/2}(Nc\tau + dQ)^{1/2}\eta((\delta * Q)\tau) \end{aligned}$$

Therefore, now denoting $g_\delta = \gcd(\delta, Q)$ and $A = A_\delta$ we have

$$\begin{aligned} & Q^k(Nc\tau + dQ)^{-k} \prod_{\delta | N} \eta(\delta W_Q \tau)^{r_\delta} \\ &= Q^k(Nc\tau + dQ)^{-k} \prod_{\delta | N} \nu_\eta(A_\delta) g_\delta^{-r_\delta/2} (Nc\tau + dQ)^{r_\delta/2} \eta((\delta * Q)\tau)^{r_\delta} \\ &= Q^k(Nc\tau + dQ)^{-k} \prod_{\delta | N} \nu_\eta(A_\delta) g_\delta^{-r_\delta/2} \prod_{\delta | N} (Nc\tau + dQ)^{r_\delta/2} \prod_{\delta | N} \eta((\delta * Q)\tau)^{r_\delta} \\ &= Q^k(Nc\tau + dQ)^{-k}(Nc\tau + dQ)^{\sum_{\delta | N} r_\delta/2} \prod_{\delta | N} \nu_\eta(A_\delta) g_\delta^{-r_\delta/2} \prod_{\delta | N} \eta((\delta * Q)\tau)^{r_\delta} \\ &= Q^k \prod_{\delta | N} \nu_\eta(A_\delta) g_\delta^{-r_\delta/2} \prod_{\delta | N} \eta((\delta * Q)\tau)^{r_\delta} \end{aligned}$$

completing the proof. □

5 An EtaProduct Implementation in Sagemath

Sage already includes an EtaProduct implementation [Loe] for modular curves. It has many features, and is efficient. One limitation of this implementation however, is that it requires some restrictions on the pairs δ, r_δ that form the eta product $\prod_{\delta|N} \eta(\delta\tau)^{r_\delta}$. For example

```
EtaProduct(8, {4: 2, 8: 2}).q_expansion()
...
ValueError: sum r_d (=4) is not 0
EtaProduct(8, {4: -2, 8: 2}).q_expansion()
ValueError: sum d r_d (=8) is not 0 mod 24
```

This is because EtaProduct verifies Ligozat's criteria of functions on the modular curve $X_0(N)$. More precisely, there is the

Theorem 5.1 (Ligozat, 1975)**.** [Lig75, Prop. 3.1.1][McM01, Th. 7.4] *An eta product of level N formed by the pairs (δ, r_δ) is the q-expansion of a function on $X_0(N)$ if and only if the following four conditions hold*

$$\sum_{\delta|N} r_\delta \equiv 0 \pmod 2, \quad \sum_{\delta|N} \delta r_\delta \equiv 0 \pmod{24}, \quad \sum_{\delta|N} \frac{N}{\delta} r_\delta \equiv 0 \pmod{24}, \quad \text{and} \quad \prod_{\delta|N} \frac{N}{\delta}^{r_\delta} \in \mathbb{Q}^2.$$

We can initially generalise this, and initially allow any eta product to calculate q-expansions at infinity. We can also calculate q-expansions at arbitrary cusps represented by rational numbers

Remark. At time of writing, the `.slashby()` method of `RhoProduct` has implementation issues.

5.1 Working with fractional powers in Sage

Unfortunately, at the time of writing, sage does not support fractional powers of symbols. Concretely, trying to do this will make sage raise a `ValueError`.

```
sage: R.<q> = LaurentSeriesRing(ZZ)
sage: q^(1/2)
...
ValueError: power series valuation would be fractional
```

To get around this limitation, there are two immediate solutions: either to implement a new `LaurentSeriesRing` subclass that allows fractional powers; or to crudely write a `repr` function to print fractional powers to `stdout` whilst internally maintaining integral powers. Although we will only need fractional powers with one denominator $(1/24)$ which makes the first option likely easier to implement, I opted for the second crude `repr` method.

The large downside, of course, of this approach, is that it does not produce a very clean api. Users must remember that the q that they are working with, represents $q^{1/24}$. As an example

```
from RhoProduct import repr_fractional_power
sage: R = LaurentSeriesRing(ZZ, names=('q',))
sage: (q,) = R._first_ngens(1)
sage: print(repr_fractional_power(q, 24))
q^(1/24)
```

5.2 (Efficiently) Managing the precision of the expansion

Lemma 3.1 tells us that the powers of q in the expansion grows quadratically. That is

$$\eta(\tau) = \sum_{n\geq 1} \left[\frac{12}{n}\right] q^{n^2/24}$$

Therefore, if we want a precision of q^k, we need to calculate this sum for $n = 1, \dots, \left\lceil \sqrt{24k} \right\rceil$ terms and then truncate at $O(q^k)$.

The precision of Eta products are a little trickier to manage since we allow negative exponents. That is to say, we must increase the precision when calculating the product, since negative powers may produce terms within our precision in a product.

5.3 Installing RhoProduct

The source is available at https://github.com/rrueger/RhoProduct.

RhoProduct has been packaged as a regular Python module. It can be installed for usage with Python with

```
pip install https://github.com/rrueger/RhoProduct/raw/main/dist/sage_rhoproduct-0.9.0-py3-
  none-any.whl
```

Sage uses its own package hierarchy to match the version of Python it is shipped with. To install for usage with Sage call

```
sage -pip install https://github.com/rrueger/RhoProduct/raw/main/dist/sage_rhoproduct-0.9.0-
  py3-none-any.whl
```

Sage is not officially available as a python module on PyPi, therefore it is not listed as a dependency in this module and will *not* be automatically installed alongside `RhoProduct`. Sage must be installed on the system independently, before `RhoProduct` can be installed with `sage -pip` as above.

After installation, one can simply `import` the module.

```
sage: from RhoProduct import RhoProduct
sage: print(RhoProduct(8, {4: 2, 8: 2}).q_expansion(20))
q - 2*q^5 - 3*q^9 + 6*q^13 + 2*q^17 + O(q^21)
```

5.4 Testing and benchmarking RhoProduct

Distributed along with the package, there is a short testing and benchmarking script. Calling `pip show sage_rhoproduct` will show the path under which the module files are installed. For example

```
$ pip show sage_rhoproduct
Name: sage-rhoproduct
Version: 0.9.0
Summary: A sage module implementing a slightly more generalised EtaProduct
Home-page:
Author:
Author-email: Ryan Rueger <rrueger@ethz.ch>, Markus Schwagenscheidt <info@markus-
  schwagenscheidt.de>
```

```
License: AGPL-3.0-only
Location: /home/rrueger/.local/lib/python3.10/site-packages
Requires: datetime
Required-by:
```

The test script is available at

```
<Location:>/RhoProduct/test/tests.py
```

which is

```
/home/rrueger/.local/lib/python3.10/site-packages/RhoProduct/test/tests.py
```

in the example above.

The testing script will print various q-expansions of `RhoProducts` and compare them with Sage's `EtaProduct`.

A RhoProduct source code

For completeness the source of the implementation is listed below. It implements a Python module which can be natively imported in both Python and Sage.

```python
#!/usr/bin/env python3

import sage.all
from sage.arith.functions import lcm
from sage.arith.misc import gcd
from sage.arith.misc import kronecker_symbol
from sage.arith.misc import xgcd
from sage.functions.other import ceil
from sage.matrix.constructor import Matrix
from sage.misc.functional import sqrt
from sage.misc.misc_c import prod
from sage.rings.big_oh import O
from sage.rings.infinity import Infinity
from sage.rings.integer_ring import ZZ
from sage.rings.laurent_series_ring import LaurentSeriesRing
from sage.rings.universal_cyclotomic_field import UniversalCyclotomicField


def simplify(a, b):
    g = gcd(a, b)
    return (a/g, b/g)


def coefficients(series):
    coeffs = dict()
    # Extract coefficients and exponents from the expansion
    # Whilst there is a simple list() interface for PowerSeriesRing, there
    # is not (a priori) one for LaurentSeriesRing
    # list() on a LaurentSeriesRing element will return the list of
    # coefficients, starting at the smallest non-zero exponent
    # Compare
    # R.<q> = LaurentSeriesRing(ZZ); list(q**5 + q**3 + q)
    # R.<q> = PolynomialRing(ZZ); list(q**5 + q**3 + q)
    # We use .exponents()[0] to extract the smallest non-zero exponent
    # So we can shift the coefficient
    lowest_exponent = series.exponents()[0]
    for exponent, coefficient in enumerate(list(series)):
        if coefficient:
            coeffs[exponent+lowest_exponent] = coefficient
    # Sort dictionary to ensure that we get the powers in ascending order
    return sorted(coeffs.items(), key=lambda i: [0])


def repr_fractional_power(series, k=24):
    """Print series in fractional exponents from whole exponents

    This is to overcome a Sage limitation of not allowing fractional powers in
```

```
    LaurentSeriesRings.
    """

    # Set precision of the input series
    # If a series is not truncated by O(z^k), then series.prec() returns
    # infinity. This is not what we want in thise case
    if series.prec() == Infinity:
        # Coefficients returns a sorted list of pairs (exponent, coefficient)
        prec = max(coefficients(series), key=lambda i: i[0])[0]
    else:
        prec = series.prec()

    # If integral exponents, we can return this as a LaurentSeriesRing object
    if all([i % k == 0 for i in series.exponents()]):
        reduced_series = 0
        R = LaurentSeriesRing(ZZ, names=('q',))
        (q,) = R._first_ngens(1)
        for coef, exp in zip(series.coefficients(),
                             [exp//k for exp in series.exponents()]):
            reduced_series += coef*q**exp

        reduced_series += O(q**ceil(prec/k))
        return reduced_series

    # Else, the exponents are not integral, and we cannot return a
    # LaurentSeriesRing, so we return a string
    fmt_str = []
    for exponent, coefficient in coefficients(series):
        # Ensuring that the fraciton n**2/k is simplified
        new_exp_numerator, new_exp_denominator = simplify(exponent, k)
        new_exponent = new_exp_numerator / new_exp_denominator

        # This case distinction is ugly, but required to avoid
        # 4x^2 + 3x^1 + 4x^0
        if exponent == 0:
            fmt_str += [f'{coefficient}']
        else:
            if coefficient == 1:
                c_str = ''
            elif coefficient == -1:
                c_str = '-'
            else:
                c_str = f'{coefficient}*'

            if new_exponent == 1:
                q_str = 'q'
            elif new_exp_denominator == 1:
                q_str = f'q^{new_exp_numerator}'
            else:
                t = new_exp_numerator/new_exp_denominator
                q_str = f'q^({t})'
```

```
            fmt_str += [c_str + q_str]

    # Sage cannot do calculations inside fstrings
    prec = ceil(prec/k)
    if series.prec() != Infinity:
        fmt_str += [f'O(q^{prec})']
    fmt_str = " + ".join(fmt_str)
    # Clean up negative coefficients and multiplying by 1
    fmt_str = fmt_str.replace('+ -', '- ')
    return fmt_str


class NonSlashableEtaProduct(Exception):
    pass


class NotEtaProduct(Exception):
    pass


class RhoProduct:
    """RhoProduct: Ryan's Eta Product

    Calling this RhoProduct allows us to compare to Sage's implementation of
    EtaProduct"""

    def __init__(self, level, exponents):
        self.level = level
        # Alias
        self.N = level
        self.exponents = exponents

        ndiv = [delta for delta in self.exponents.keys() if level % delta]
        if ndiv:
            if len(ndiv) == 1:
                err = f"{ndiv[0]} does not divide level ({level})"
            else:
                ndiv_str = ", ".join(map(str, ndiv))
                err = f"{ndiv_str} do not divide level ({level})"

            rhoproduct = f"RhoProudct({level}, {repr(exponents)})"
            raise NotEtaProduct(f"Wrong level/exponents in {rhoproduct}: {err}")

        self.k = sum(exponents.values())//2
        # Requires int() call to prevent sage from casting the elements as
        # floats and runing into overflow errors (prod is a sage function)
        self.char = prod([int(d**exp) for (d, exp) in self.exponents.items()])
        self.char *= (-1)**self.k

    def __eta_function(self, prec, alpha=1, beta=0):
```

```
        # Alias for debugging performance
        level = self.level
        if beta:
            U = UniversalCyclotomicField()
            E = U.gen
            R = LaurentSeriesRing(U, names=('q',))
            (q,) = R._first_ngens(1)
            self.expansion = 0
            for n in range(ceil(sqrt(prec*level*24))):
                # Here, use q, since we want a power of q^(1/24)
                self.expansion += kronecker_symbol(12, n) \
                                  * E(24)**(n**2*beta)    \
                                  * q**(alpha*level*n**2)
        else:
            R = LaurentSeriesRing(ZZ, names=('q',))
            (q,) = R._first_ngens(1)
            self.expansion = 0
            for n in range(ceil(sqrt(prec*level*24))):
                self.expansion += kronecker_symbol(12, n) * q**(alpha*level*n**2)

        # Here we do need q**24 since we want prec to represent in integral
        # powers of q
        self.expansion += O(q**(24*level*prec))
        return self.expansion

    def __coefficients(self):
        return coefficients(self.expansion)

    def __repr__(self):
        # Repr in the style of Sage's EtaProduct with addl true repr
        product = " ".join([f'(eta_{d})^{e}' for d, e in self.exponents.items()])
        interpretation = f'Rho Product of level {self.level} : {product}'
        true_repr = f'RhoProduct({self.N}, {repr(self.exponents)})'
        return interpretation + f' ({true_repr})'

        # # Informal __repr__, in the style of Sage
        # product_terms = [f'n(z)^{e}' if d == 1 else f'n({d}z)^{e}'
        #                  for d, e in self.exponents.items()]
        # product = "*".join(product_terms)
        # interpretation = f' is the RhoProduct {product} of level {self.N}'

        # # True __repr__, i.e. one can type this get the object
        # true_repr = f'RhoProduct({self.N}, {repr(self.exponents)})'
        # return true_repr + interpretation

    def __pow__(self, power):
        exponents = {d: e*power for d, e in self.exponents.items()}
        return RhoProduct(self.level, exponents)

    def __mul__(self, other):
        level = lcm(self.level, other.level)
```

```python
        # Create new exponents dict. We do not want to change self or other
        exponents = dict()
        for delta, exponent in self.exponents.items():
            exponents[delta] = exponent
        for delta, exponent in other.exponents.items():
            if delta in exponents:
                exponents[delta] += exponent
            else:
                exponents[delta] = exponent
        return RhoProduct(level, exponents)

    def isetaproduct(self):
        # For integral weight for \Gamma_0(N)
        if sum(self.exponents.values()) % 2:
            # print("The sum of exponents is not divisible by 2")
            return False
        elif sum([d*exp for d, exp in self.exponents.items()]) % 24:
            # print("The sum of d*a_d is not divisible by 24")
            return False
        elif sum([self.N/d*exp for d, exp in self.exponents.items()]) % 24:
            # print("The sum of N/d*a_d is not divisible by 24")
            return False

        return True

    def isslashable(self):
        # For integral weight for \Gamma_0(N)
        if sum(self.exponents.values()) % 2:
            print("The sum of exponents is not divisible by 2")
            return False
        return True

    def q_expansion(self, prec=10):
        self.expansion = prod([self.__eta_function(prec, alpha=delta)**exponent
                               for delta, exponent in self.exponents.items()])
        return repr_fractional_power(self.expansion, 24*self.level)

    def expansionatcusp(self, cusp):
        pass

    def order(self, cusp):
        d = cusp.denominator()
        s = sum([gcd(d, delta)*exp/delta for delta, exp in self.exponents.items()
    ])
        order = self.N/24 * 1/d * 1/gcd(d, self.N/d) * s
        return order

    def slashby(self, matrix, prec=10):
        a = matrix[0][0]
        b = matrix[0][1]
        c = matrix[1][0]
```

```
        d = matrix[1][1]

        if not a*d - b*c == 1:
            raise NonSlashableEtaProduct("Matrix is not in SL_2(Z)")

        if not self.isslashable():
            raise NonSlashableEtaProduct("Cannot slash by matrix")

        U = UniversalCyclotomicField()
        R = LaurentSeriesRing(U, names=('q',))
        (q,) = R._first_ngens(1)
        p = 1

        for delta, exponent in self.exponents.items():
            g = gcd(c, delta)
            _, y, x = xgcd(-c/delta, delta*a/g)
            nu = x*delta*b + y*d
            p *= self.__eta_function(prec,
                                     alpha=1/delta,
                                     beta=nu*g/delta)**int(exponent)
                                     # alpha=1,
                                     # alpha=g**2/delta,

        return repr_fractional_power(p, 24*self.level)

    # Since the fricke involution is only available as an expansion for now, we
    # must pass a precision paramater to it
    def al_involution(self, Q, matrix=None, prec=10):
        if gcd(self.level, Q) != Q:
            print("Q does not divide level!")
            exit(1)
        elif gcd(self.level/Q, Q) != 1:
            print("Q is not an _exact_ divisor of level!")
            exit(1)

        if matrix is None:
            a = Q
            b = 1
            _, d, c = xgcd(Q, -self.level/Q)
            c *= self.level
            d *= Q
        else:
            a = matrix[0][0]
            b = matrix[0][1]
            c = matrix[1][0]
            d = matrix[1][1]

        # First slash with (Q, 0; 0 ,1)
        # Do this by replacing z with Qz
        exponents = {Q * delta: exponent for delta, exponent in
                     self.exponents.items()}
```

```
        # Now try to slash the new "eta product" by the reduced matrix
        return RhoProduct(self.level*Q, exponents
                          ).slashby(Matrix([[a/Q, b], [c/Q, d]]))
```

B RhoProduct testing and benchmarking script

After having installed the RhoProduct Python module, this testing script will calculate expansions of different RhoProducts and compare them to Sage's EtaProduct implementation.

```
#!/usr/bin/env python3

from RhoProduct import RhoProduct, repr_fractional_power, NotEtaProduct
import datetime
import sage.all
from sage.misc.misc_c import prod
from sage.rings.laurent_series_ring import LaurentSeriesRing
from sage.rings.integer_ring import ZZ
from sage.rings.big_oh import O
from sage.modular.etaproducts import EtaProduct
from sage.matrix.constructor import Matrix

print("----------------------------------------------------------------------")
print("Testing __repr__ of RhoProduct")
print("----------------------------------------------------------------------")
print(RhoProduct(8, {1: 24, 2: -24}))
print(RhoProduct(8, {1: 24, 2: -24})*RhoProduct(6, {2: 24, 3: -24}))

R = LaurentSeriesRing(ZZ, names=('q',))
(q,) = R._first_ngens(1)
print("Fractional power repr example:", repr_fractional_power(q, 24))

# print("----------------------------------------------------------------------")
# print("Testing NotEtaProduct level/exponents check")
# print("----------------------------------------------------------------------")
# try:
#     RhoProduct(8, {3: 24, 7: -24})
# except NotEtaProduct as err:
#     print("RhoProduct(8, {3: 24, 7: -24})")
#     print(err)

print("----------------------------------------------------------------------")
print("Check that the Eta Function is correctly implemented")
print("----------------------------------------------------------------------")
# This is surprisingly difficult to do, for two reasons
# 1. Sage's EtaProducts have no way of just printing the expansion of the
#    EtaFunction
# 2. Naively calculating (1-q^n) converges at a different rate that the fourier
#    expansion. This means that one has to calculate both terms to a high
#    precision to get comparable results.
# Must be LaurentSeriesRing for coefficients() interface to work
R = LaurentSeriesRing(ZZ, names=('q',)); (q,) = R._first_ngens(1)
```

```
# Implicitly use q^24, so that we can properly divide later.
p = prod([1 - q**(24*n) for n in range(1, 10)])
# "Multiply" by q^(1/24)
print("Naive product expansion:", repr_fractional_power(q*p + O(q**(10*24))))
print("RhoProduct             :", RhoProduct(1, {1: 1}).q_expansion())

print("----------------------------------------------------------------------")
print("Directly comparing expansions of EtaProduct with RhoProduct")
print("----------------------------------------------------------------------")
print("EtaProduct:", EtaProduct(8, {1: 24, 2: -24}).q_expansion(10))
print("RhoProduct:", RhoProduct(8, {1: 24, 2: -24}).q_expansion(10))

print("----------------------------------------------------------------------")
print("Calculating non-EtaProduct examples from Web of Modularity")
print("----------------------------------------------------------------------")
print("Ex. 1.66 Web of Modularity:", RhoProduct(5, {5: 5, 1: -1}).q_expansion())
print("Ex. 1.66 Web of Modularity:", RhoProduct(8, {4: 2, 8: 2}).q_expansion())

# print("----------------------------------------------------------------------")
# print("Slashing by matrices")
# print("----------------------------------------------------------------------")
# print(RhoProduct(5, {5: 5, 1: -1}).slashby(Matrix([[4, 7], [1, 2]])))
# print()
# print(RhoProduct(1, {1: 24}).slashby(Matrix([[0, 1], [-1, 0]])))
# print(RhoProduct(1, {1: 24}).slashby(Matrix([[4, 7], [1, 2]])))
# print(RhoProduct(1, {1: 24}).q_expansion())
# print(RhoProduct(11, {1: 2, 11: 2}).q_expansion())
# print(RhoProduct(11, {1: 2, 11: 2}).isetaproduct())
# print(RhoProduct(11, {1: 2, 11: 2}).slashby(Matrix([[0, -1], [1, 0]])))
# print("AL Involution:", RhoProduct(8, {1: 24, 2: -24}).al_involution(8))

print("----------------------------------------------------------------------")
print("Test Accuracy and benchmarking speed of RhoProduct vs EtaProduct")
print("----------------------------------------------------------------------")
max_prec = 50
level = 12
exponents = {1: -336, 2: 576, 3: 696, 4: -216, 6: -576, 12: -144}
print(":: Comparing EtaProduct and RhoProducts")
print(EtaProduct(level, exponents))
print(RhoProduct(level, exponents))
print(f":: Calculating EtaProduct expansion up to precision {max_prec}...")
etaproducts = []
start = datetime.datetime.now()
for prec in range(1, max_prec+1):
    print(f'\r{prec}/{max_prec}', end='')
    etaproducts.append(EtaProduct(level, exponents).q_expansion(prec))
print()
eta_diff = (datetime.datetime.now() - start)
eta_time = eta_diff.seconds + float(eta_diff.microseconds / 10**9)
eta_timeper = round(float(eta_time/max_prec), 5)
print(f':: Time taken: {eta_time}s ({eta_timeper}s per calculation)')
```

```python

print(f":: Calculating RhoProduct expansion up to precision {max_prec}...")
rhoproducts = []
start = datetime.datetime.now()
for prec in range(1, max_prec+1):
    if datetime.datetime.now() - start > eta_diff:
        factor = round((datetime.datetime.now() - start)/eta_diff * max_prec/prec, 2)
        print(f'\r{prec}/{max_prec} (Slower than EtaProduct by factor {factor})', end='')
    else:
        print(f'\r{prec}/{max_prec}', end='')
    rhoproducts.append(RhoProduct(level, exponents).q_expansion(prec))
print()
rho_diff = (datetime.datetime.now() - start)
rho_time = rho_diff.seconds + float(rho_diff.microseconds / 10**9)
rho_timeper = round(float(rho_time/max_prec), 5)
print(f':: Time taken: {rho_time}s ({rho_timeper}s per calculation)')

if not etaproducts == rhoproducts:
    print(":: Not Equal")
    print(":: First 5 product expansions")
    for etaproduct, rhoproduct in zip(etaproducts[:5], rhoproducts[:5]):
        print("EtaProduct:",  etaproduct)
        print("RhoProduct:",  rhoproduct)
else:
    print(":: Both results are equal at all precision levels")

print(f'EtaProduct was {round(float(rho_time/eta_time), 5)}x faster')
```

References

[Apo91] Tom M. Apostol, *Modular Functions and Dirichlet Series in Number Theory*, Vol. Graduate Texts in Mathematics 41, Springer, 1991. ↑14

[Ono04] K. Ono, *The Web of Modularity: Arithmetic of the Coefficients of Modular Forms and q-series*, Regional conference series in mathematics, American Mathematical Society, 2004. ↑6, 14, 18, 22

[Sie54] Carl Ludwig Siegel, *A simple proof of $\eta(-1/\tau) = \eta(\tau)\sqrt{\tau/i}$*, Mathematika **1** (1954), no. 1, 4-4, DOI https://doi.org/10.1112/S0025579300000462. ↑14

[Köh11] Günter Köhler, *Eta products and theta series identities*, Springer monographs in mathematics, Springer, Berlin, 2011. ↑8, 19, 22

[Bor99] R. E. Borcherds, *Reflection groups of Lorentzian lattices*, posted on 1999, DOI 10.48550/ARXIV.MATH/9909123. ↑6

[Sch21] Markus Schwagenscheidt, *Lectures on Elliptic Functions* (2021), https://people.math.ethz.ch/~mschwagen/ellipticfunctions. ↑8

[Rad70] Hans Rademacher, *Topics in Analytic Number Theory*, Springer, 1970. ↑10

[Loe] David Loeffler, *Eta Products on Modular Curves*, https://doc.sagemath.org/html/en/reference/modfrm/sage/modular/etaproducts.html. ↑25

[Lig75] Gérard Ligozat, *Courbes Modulaires de Genre 1*, Mémoires de la S. M. F. **43** (1975). ↑25

[McM01] Ken McMurdy, *A Splitting Criterion for Galois Representations Associated to Exceptional Modular Forms*, 2001. ↑25