Ratio 3 and Ratio 4

Richard Adhika

1. Introduction

Since in $\mathbf{GL}_2(\mathbb{Q}_p)$, conjugacy is the same as equality of characteristic polynomials, saying that M' has the same trace and determinant as M is equivalent to saying that there exists $\gamma \in \mathbf{GL}_2(\mathbb{Q}_p)$ such that $M' = C^{-1}MC$, or $CM' = MC$. Now, in the latter equality we can clear denominators, and think of C as a matrix in $c_2(\mathbb{Z}_p)$. This, however, creates a problem: if we just want such a C from $c_2(\mathbb{Z}_p)$, it could make us lose information: for example if we take $C = 0$, any M and M' would satisfy this condition. So we introduce a notion of 'approximate k-conjugacy': we say that M is approximately k-conjugate to M' if $MC \equiv CM' \bmod p^k$. To resolve the issue of *losing* information, we want to restrict C to having $v_p(\det(C)) \leq e$ for some $e \in \mathbb{Z}_0$, which defines how invertible C is.

With this, given a matrix $M' \in \mathbf{GL}_2(\mathbb{Q}_p)$ having trace t and determinant D, we are interested in counting the number of $M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z})$ satisfying the following,

1. M is $\mathbf{GL}_2(\mathbb{Q}_p)$ conjugate to M' (this is exactly the numerator condition of ratio 1), and
2. there exists $C \in c_2(\mathbb{Z}/p^n\mathbb{Z})$ such that $MC \equiv CM' \bmod p^k$ with $v_p(\det(C)) \leq e$.

we are interested in the following ratio

$$\eta_{p,k,e}(t, D) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) \mid M \text{ satisfies (1) and (2)}\}}{p^{2n-2}(p^2-1)}.$$

The conjecture is that the addition of the approximate conjugacy constraint (the second point) does not affect the ratio (give the same number as the first ratio) if $n \gg k$ and large enough e. Similar with the relationship between ratio 1 and ratio 2, we also wants to count the number of matrices $M \in \mathbf{GL}_2(\mathbb{Q}_p)$ that lift to $\mathbb{Z}_p$, that is,

1. there exists $N \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ such that N has trace t and determinant D, and $M' \equiv M \bmod p^n$ (this is exactly the numerator condition of ratio 2), and
2. there exists $C \in c_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ such that $MC \equiv CM' \bmod p^{kbig}$ with $v_p(\det(C)) \leq e$.

We then define ratio 4 as

$$\theta_{p,kbig,e}(t, D) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) \mid M \text{ satisfies (1) and (2)}\}}{p^{2n-2}(p^2-1)}.$$

Again, it is expected that ratio 4 is equal to ratio 2 if $kbig \gg n$ and large enough e. In both ratios, if $p \nmid t^2 - 4D$, then $e = 0$ works and that the second condition does not reduce the count.

2. Ratio 3

We first generate all the matrices satisfying (1) as in ratio 1. Then, finding the matrix C satisfying the second condition is tricky as the brute force method of checking all C in its domain will take forever. Let

$$M' = \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}, M = \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix}, \text{ and } C = \begin{pmatrix} x & y \\ z & w \end{pmatrix}.$$

We then want to solve for C in the equation

$$\begin{aligned} & MC \equiv CM' \bmod p^k \\ \iff & \begin{pmatrix} a_2x + b_2z & a_2y + b_2w \\ c_2x + d_2z & c_2y + d_2w \end{pmatrix} \equiv \begin{pmatrix} a_1x + c_1y & b_1x + d_1y \\ a_1z + c_1w & b_1z + d_1w \end{pmatrix} \bmod p^k. \end{aligned}$$

We can rewrite these as the following system of linear equations,

$$\begin{pmatrix} a_2 - a_1 & -c_1 & b_2 & 0 \\ c_2 & 0 & d_2 - a_1 & -c_1 \\ -b_1 & a_2 - d_1 & 0 & b_2 \\ 0 & c_2 & -b_1 & d_2 - d_1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \equiv 0 \bmod p^k. \tag{1}$$

We can solve this by finding the kernel basis (mod p^k) of the 4×4 matrix. Using `Oscar` package from `Julia` programming language, we have the built-in function that does exactly this (line 16 in the function `ratio3`). Finding ring kernel for each matrix takes up a lot of time and space, and is the main problem of computing ratio 3. I have not found any optimization that allows me to avoid doing so. All code in this section can be found on the file *ratio3.ipynb*.

Algorithm 1 Ratio 3

```
1: function RATIO3($p, n, k$, equations_template, matrices)
2:     $R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
3:     ker_hist $\leftarrow \{\}$, res $\leftarrow \{\}$
4:     for all $(M, num) \in$ matrices do
5:         equations $\leftarrow$ equations_template$[M[1]+1]$
6:         equations[1,3], equations[3,4], equations[2,1], equations[4,2] $\leftarrow M[2], M[2], M[3], M[3]$
7:         ker $\leftarrow$ kernel(equations, right)
8:         if ker $\in$ ker_hist then
9:             res$[M] \leftarrow$ ker_hist[ker]
10:        else
11:            minVal $\leftarrow$ det_check$(p, k, \text{ker}, R_n)$
12:            ker_hist[ker], res$[M] \leftarrow$ minVal, minVal
13:        end if
14:    end for
15:    return res
16: end function
```

The parameter 'equations_template' denotes the set of 4×4 matrices as in (1). We only need to create and store p^n of all such 4×4 matrices (for $a \in [0, p^n - 1]$), rather than creating one each time we are checking a new matrix. For any fixed a, there are only 4 entries of the corresponding matrix that vary (line 5-6). Lastly, we have the dictionary 'ker_hist' that stores a kernel vector as its key and the minimal p-valuation of the corresponding determinant as its value. This is done because I notice there are repeated kernels for different matrices, hence checking the determinant only once may be more efficient.

The parameter 'matrices' is a dictionary storing matrices $M \in \mathbf{GL}_2(\mathbb{Z}/p^k\mathbb{Z})$ that reduced from mod p^n as the key and the corresponding count as the value. We do this because if $k < n$, there may be a lot of repeated matrices when reduced to mod p^k and that the equation works in mod p^k as well. But how do we get this just from the function `build_matrices1`. If $k = n$, then there is nothing to reduce and we get the dictionary directly from the `build_matrices1`. For $k < n$, then we loop through all the matrices in mod p^{k+1} and then increase the count of the reduced matrices accordingly. We then do this from $k = n - 1$ to $k = 1$. Note that this is only effective if we are interested in computing ratio 3 for $k = 1$ up to $k = n$. Otherwise, there is a function to generate the matrices more efficiently in the last part of ratio3.ipynb (won't be discussed here).

After obtaining the kernel basis, we then try to construct the matrix C from these kernel by trying different combinations of the vectors and lifting them up to mod p^n, which is implemented in the function `det_check` (the pseudocode is not provided here as it is just a direct translation from the steps described below). We find C with minimal p-valuation of its determinant in two steps:

1. Let $\begin{pmatrix}\alpha & \beta & \gamma & \delta\end{pmatrix}$ be one of the kernel basis vectors. We then check the following value

$$\Delta = (\alpha + i_1 p^k)(\delta + i_4 p^k) - (\beta + i_2 p^k)(\gamma + i_3 p^k)$$

with $i_1, i_2, i_3, i_4 \in [0, 1]$. Here, Δ is the disriminant of the vector, with attempts to lift the entry(ies) to $\mathbb{Z}/p^n/Z$ by adding the i's as $p^k \equiv 0 \bmod p^k$.

2. For any two kernel vectors in the basis

$$v_1 = \begin{pmatrix}\alpha_1 & \beta_1 & \gamma_1 & \delta_1\end{pmatrix} \text{ and } v_2 = \begin{pmatrix}\alpha_2 & \beta_2 & \gamma_2 & \delta_2\end{pmatrix},$$

we form the matrix

$$C = \begin{pmatrix}\alpha_1 + \alpha_2 & \beta_1 + \beta_2 \\ \gamma_1 + \gamma_2 & \delta_1 + \delta_2\end{pmatrix}.$$

and then check the p-valuation of its determinant. Note that the new term whose p-valuation is being considered here is given by

$$\alpha_1\delta_2 - \beta_1\gamma_2 + \alpha_2\delta_1 - \beta_2\gamma_1.$$

Does this cover all possibilities of the p-valuation of the conjugating matrices? Wtih v_1 and v_2 defined as the second step above, consider the matrix

$$C' = \begin{pmatrix}i\alpha_1 + j\alpha_2 & i\beta_1 + j\beta_2 \\ i\gamma_1 + j\gamma_2 & i\delta_1 + j\delta_2\end{pmatrix}$$

for some constants i, j. Its determinant is given by

$$\begin{aligned}\det(C') &= \big(i\alpha_1 + j\alpha_2\big)\big(i\delta_1 + j\delta_2\big) - \big(i\beta_1 + j\beta_2\big)\big(i\gamma_1 + j\gamma_2\big) \\ &= i^2(\alpha_1\delta_1 - \beta_1\gamma_2) + j^2(\alpha_2\delta_2 - \beta_2\gamma_2) + ij(\alpha_1\delta_2 - \beta_1\gamma_2 + \alpha_2\delta_1 - \beta_2\gamma_1).\end{aligned}$$

The p-valuation of the first and second term has been calculated in the first step and of the last term has also been considered in the second step. Hence, there we don't need to consider the determinant of C' Furthermore, there is no need to lift the matrices in the second step as if it makes the p-valuation of its determinant equals k, then we must have encountered so in the first step. Lastly, any combination of three or more kernel basis vectors also have its p-valuation of determinant already considered as part of step 2.

3. Ratio 4

Only a few things are different than the process of finding ratio 3. First, instead of using `build_matrices1` from ratio1, we will use `ratio2` function from computing ratio 2. The slight modification in the function is that we need to store all the reduced matrices instead of just counting them, and that we need to consider every $b \in [0, p^{kbig-i}]$, not just $b = p^i$ for $1 \leq i \leq n-1$. Everything else is very similar to that of ratio 3.