Ratio 1 and Ratio 2

Richard Adhika

1. Introduction

For reasons coming from number theory, we are interested in computing the following two kinds of (closely related) ratios, for different primes p: Let t and d be integers.

The first ratio is

$$\nu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | M \text{ has trace } t \text{ and } \det D \mod p^n\}}{p^{2n-2}(p^2-1)}.$$

This ratio is really interesting only when the *discriminant* of the characteristic polynomial, i.e., $t^2 - 4D$, is divisible by p (and they get more interesting as the power of p dividing the discriminant gets larger). These stabilize when n gets large (if p doesn't divide the discriminant, they stabilize right away). However, we understand these quite well. We are more interested in the ones that come no from counting solutions $\mod p^n$, but from counting the *reductions* $\mod p^n$ *of solutions in* $\mathbb{Z}_p$. More precisely, we define

$$\mu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | \exists M' \in \mathbf{GL}_2(\mathbb{Z}_p) : M' \text{ has trace } t \text{ and } \det D, \text{ and } M' \bmod p^n = M\}}{p^{2n-2}(p^2-1)}.$$

Since we cannot really handle full p-adic series in a computer code, we introduce another parameter, "kbig" or something (also user-defined), and compute

$$\mu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | \exists M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z}) : M' \text{ has trace } t \text{ and } \det D, \text{ and } M' \bmod p^n = M\}}{p^{2n-2}(p^2-1)}.$$

Again these stabilize eventually (as n gets large, but we need $kbig >> n$), and again they are only interesting when a power of p divides the discriminant (when it doesn't, these also stabilize at $n = 1$ and are equal to ν_p). We want to know these numbers for various values of t, d, and p. The problem is that even if you let $p = 5$ and $kbig = 6$, it is already taking forever to compute (using brute force method).

2. Ratio 1

We first want to try to generate all matrices of the form

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

with $d \equiv t - a \bmod p^n$ and $ad - bc \equiv D \bmod p^n$ for $a, b, c, d \in \mathbb{Z}/p^n\mathbb{Z}$. Here, we have 3 free parameters a, b, and c since d depends solely on a for the given t. The naive brute force method is to loop through $[0, p^n - 1]$ for each a, b, c and then check the determinant condition. This takes $\mathcal{O}(p^{3n})$, which is very slow. Hence, the function `build_matrices1` in the file *ratio1_and_ratio2.ipynb* aims to generate the matrices more efficiently (the pseudocode of the function is provided on the next page).

For the parameter a, we will still loop through $[0, p^n - 1]$. First, fix an a. Let rhs$= ad - D \bmod p^n$. Then, for any $b \in [0, p^n - 1]$, the problem turns into solving the equation (for c)

$$bc \equiv rhs \bmod p^n. \tag{1}$$

If $p \nmid b$, then b^{-1} exists and thus we have a unique solution $c \equiv rhs \cdot b^{-1} \bmod p^n$ (line 6). We don't need to loop through c for these b. If $p \mid b$, then we have three cases as follows,

1. $p \nmid rhs$. This immediately tell us that (1) can't have any solution if $p \mid b$. This corresponds to the facet that the loop in line 9 won't execute.

2. $v_p(rhs) \geq 1$. If $v_p(b) > v_p(rhs)$, then (1) also can't have any solution. Hence, it suffices to check b with $1 \leq v_p(b) \leq v_p(rhs)$ (line 10 with $v_p(b) = i$). Also $v_p(rhs) < n$ so the modulo is well-defined. Then, for each i, let $b = p^i j$ so that we can rewrite (1) as

   $$p^i j \cdot c \equiv rhs \bmod p^n,$$

   We then get $c \equiv (rhs/p^i) \cdot j^{-1} \bmod p^{n-i}$, for which we lift c to modulo p^n.

3. $rhs \equiv 0 \bmod p^n$. Then, we can set $b = 0$ so that all $c \in [0, p^n - 1]$ satisfy (1) (line 14-16). Furthermore, for any b, all c with $v_p(c) = n - v_p(b)$ satisfy (1) (line 17 - 19).

Algorithm 1 build_matrices1

1: **function** BUILD_MATRICES1(n, p, t, D)
2: $\quad res \leftarrow [\ ]$, $Rn \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
3: $\quad$**for all** $a \in Rn$ **do**
4: $\quad\quad rhs \leftarrow Rn(D - a(t - a))$
5: $\quad\quad$Add all tuples (a, b, c, d) to res such that $c \equiv rhs \cdot b^{-1} \bmod p^n$ with $p \nmid b$.
6:
7: $\quad\quad$**if** rhs $\neq 0$ **then**
8: $\quad\quad\quad$**for** $i = 1$ to $v_p(rhs)$ **do**
9: $\quad\quad\quad\quad$**for** $b \in Rn$ with $b = p^i j$ with $p \nmid j$ **do**
10: $\quad\quad\quad\quad\quad$Add (a, b, c, d) to res with $c \in Rn$ satisfying $c \equiv rhs \cdot j^{-1} \bmod p^{n-i}$
11: $\quad\quad\quad\quad$**end for**
12: $\quad\quad\quad$**end for**
13: $\quad\quad$**else**
14: $\quad\quad\quad$**for all** $c \in R$ **do**
15: $\quad\quad\quad\quad$Add $(a, 0, c, d)$ to res
16: $\quad\quad\quad$**end for**
17: $\quad\quad\quad$**for** $b \in Rn$ with $v_p(b) = i \geq 1$ **do**
18: $\quad\quad\quad\quad$Add (a, b, c, d) to res with $c \in Rn$ satisfying $v_p(c) = n - i$
19: $\quad\quad\quad$**end for**
20: $\quad\quad$**end if**
21: $\quad$**end for**
22: $\quad$**return** res
23: **end function**

Assuming n is not large, this algorithm costs $\mathcal{O}(p^{2n})$, Notice that in the second case, we have that the number of matrices with $b = b'$ satisfying $v_p(b') = v$ is the same as that of $b = p^v$ as the loops in line 9-11 has the same step size. which is already quite an improvement from the brute force algorithm.

Now, we can modify this algorithm a bit if we only want to *count* the number of matrices satisfying the numerator condition. The few points that we obtained so far are:

- For each a, there are $p^{n-1}(p-1)$ b's that are relatively prime to p. Each of these b produces exactly one matrix. Hence, we have $p^{2n-1}(p-1)$ matrices in total with $p \nmid b$.
- For each a, if rhs has p-valuation k, then there are

$$\sum_{i=1}^{k} p^{n-i-1} \cdot (p-1) \cdot p^i = kp^{n-1}(p-1)$$

  many solutions that corresponds to b with $v_p(b) \geq 1$. The term $p^{n-i-1}(p-1)$ comes from the number of b with $v_p(b) = i$ and the term p^i comes from lifting the modulo from p^{n-i} to p^n.
- For each a, if rhs $\equiv 0 \bmod p^n$, then there are

$$p^n + \sum_{i=1}^{n-1} p^i(p^{n-i} - p^{n-i-1}) = p^n + (n-1)(p^n - p^{n-1})$$

  many solutions corresponds to b with $b = 0$ and $v_p(b) \geq 1$.

With these in mind, we then write a function `ratio1` that count the number of matrices satisfying the numerator condition in $\mathcal{O}(p^n)$,. The code is as follows,

Algorithm 2 ratio1

function RATIO1(n, t, D)
  $count \leftarrow p^n \cdot p^{n-1} \cdot (p-1)$, $Rn \leftarrow \text{Integers}(p^n)$
  for all $a \in R$ **do**
    $rhs \leftarrow Rn(a(t-a) - D)$
    if $rhs \neq 0$ **then**
      $count \leftarrow count + v_p(b) \cdot p^{n-1} \cdot (p-1)$
    else
      $count \leftarrow count + p^n + (n-1) \cdot (p^n - p^{n-1})$
    end if
  end for
  return $count$
end function

3. Ratio 2

For the second ratio, however, we still need to generate the matrices as the reduction from $kbig$ to n may not include every matrix that satisfies the constraint in $\mathbb{Z}/p^n\mathbb{Z}$ (which is true if $p^l | t^2 - 4D$ for some $l \in \mathbb{N}$). The problem now is that generating and storing all such matrices takes up a lot of time and space (e.g. for $p = 5$ and $kbig = 5$, we have around 11 million matrices and for $kbig = 6$, CoCalc runs out of space).

What if we don't actually need to generate all the matrices? The first observation is that

Lemma 1. *For any matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with entries in $\mathbb{Z}/p^{kbig}\mathbb{Z}$, if $p \nmid b$, we only need to consider $a, b \in [0, p^n - 1]$.*

Proof. Consider two matrices $\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}$ and $\begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix}$ with $p \nmid b_1, b_2$. Let $rhs = a_1 d_1 - D \bmod p^{kbig}$. Then, the modular equation $b_1 c_1 \equiv rhs \bmod p^{kbig}$ has a unique solution $c_1 \equiv rhs \cdot b_1^{-1} \bmod p^{kbig}$. Furthermore, the modular equation $b_2 b_3 \equiv rhs \bmod p^{kbig}$ also has a unique solution $b_3 \equiv rhs \cdot b_2^{-1} \bmod p^{kbig}$. Assume that $b_2 \equiv b_1 \bmod p^n$ and $a_2 \equiv a_1 \bmod p^n$. We therefore have

$$d_2 \equiv t - a_2 \equiv t - a_1 \equiv d_1 \bmod p^n$$

and

$$c_2 \equiv (a_2 d_2 - D) b_2^{-1} \equiv (a_1 d_1 - D) b_1^{-1} \equiv c_1 \bmod p^n.$$

Hence all $a, b \geq p^n$ with $p \nmid b$ will not produce any new reduced matrix. □

This means that the reduction does not affect the existence and uniqueness of matrices for b with $p \nmid b$. Therefore, just as the first ratio, we don't need to consider these b's and that the count for these matrices is $p^{2n-1}(p-1)$. Next, we have

Lemma 2. *Let A be the set of all $M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z})$ such that there exists $M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ having trace t and determinant D, and that $M \equiv M' \bmod p^n$ (the numerator condition). Let $S \subset A$ with $b = p^v$ and $S' \subset A$ with $b = p^v u$ for some fixed u with $p \nmid u$ and $v < n$. Then, $|S| = |S'|$.*

Proof. Let $v \in [1, n-1]$ and $M = \begin{pmatrix} a & p^v \\ c & d \end{pmatrix} \in S$. Define a map $\phi : S \to S'$ by

$$\phi(M) = N = \begin{pmatrix} a & p^v u \\ cu^{-1} & d \end{pmatrix}.$$

First, we want to show that this mapping is well-defined by showing that $N \in S'$. By definition of M, there exists $M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ of the form $M' = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$ having trace t and determinant D and that $M' \equiv M \mod p^n$. Let

$$U = \begin{pmatrix} 1 & 0 \\ 0 & u \end{pmatrix},\ U^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & u^{-1} \end{pmatrix}.$$

Let $N' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ be defined by

$$N' = U^{-1}MU = \begin{pmatrix} a' & b'u \\ c'u^{-1} & d' \end{pmatrix}.$$

Then, N' has trace t and determinant D, and that $b'u \equiv p^v u \bmod p^n$. Hence, $N = N' \bmod p^n \in S'$.

We then want to show that this mapping is injective. Let $M_1, M_2 \in S$ of the form

$$M_1 = \begin{pmatrix} a_1 & p^v \\ c_1 & d_1 \end{pmatrix}, M_2 = \begin{pmatrix} a_2 & p^v \\ c_2 & d_2 \end{pmatrix}.$$

Assume that $\phi(M_1) = \phi(M_2)$. Hence, we have

$$\begin{pmatrix} a_1 & p^v u \\ c_1 u^{-1} & d_1 \end{pmatrix} = \begin{pmatrix} a_2 & p^v u \\ c_2 u^{-1} & d_2 \end{pmatrix}$$

Since $p \nmid u$ and $p \nmid u^{-1}$, the equality $M_1 = M_2$ follows.

Finally, we want to show that the map is surjective. Let $N = \begin{pmatrix} a & p^v u \\ cu^{-1} & d \end{pmatrix} \in S'$ (any matrix in S' can be written in this form). Let $M = \begin{pmatrix} a & p^v \\ c & d \end{pmatrix}$. If N' is the lift of N, then we define $M' = U^{-1}N'U$ and a similar argument from the previous paragraph shows that $M \equiv M' \bmod p^n$. Hence $M \in S$ and that $\phi(M) = N$ so that ϕ is surjective. Finally, we conclude that ϕ is a well-defined bijection between S and S' so that $|S| = |S'|$. $\square$

By Lemma 2, for ratio 2, we only need to consider the reduced matrices with $b = p^v$ for $1 \leq v < kbig$ and $b = 0$. Before going to the main function `ratio2`, we will take a look more closely at two helper functions: `ratio2_adder` and `ratio2_adder_with_check`, which reduce a lot of repetitions (of b and c) to make the program more efficient. Below is the code for the function `helper_fn`.

Algorithm 3 Helper Function 1

function RATIO2_ADDER($p, n, kbig$, projset, rhs_new, a, ind, exp)
  $R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
  $c_{\text{start}} \leftarrow \left(\frac{\text{rhs\_new}}{\text{ind}}\right) \bmod p^{k_{\max}-\text{exp}}$
  if $\big(R_n(a),\ R_n(c_{\text{start}}),\ R_n(d)\big) \in$ projset **then**
    return false
  end if
  Add $\big(R_n(a),\ R_n(c_{\text{start}}),\ R_n(d)\big)$ to projset
  for c from $c_{\text{start}} + p^{k_{\max}-\text{exp}}$ to $p^n - 1$ step $p^{k_{\max}-\text{exp}}$ **do**
    Add $\big(R_n(a),\ R_n(c),\ R_n(d)\big)$ to projset
  end for
  return true
end function

Note that we use this function only for $j < n$. The parameter rhs_new in this function denotes $\frac{ad-D}{p^j}$, ind denotes $\frac{b}{p^j}$ with $p \nmid$ ind, and exp denotes j (we will still use j in the explanation below, the use of *exp* is just for code readability). If this helper function returns false, then we stop the loop of b and immediately proceed to the next a in the function `count_matrices2`. What this is saying is that if we encounter one reduced matrix that has been considered before (either for smaller a or b), then there is no need to check further b or c for this fixed a.

Let $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be the matrix that we are considering with entries in $\mathbb{Z}/p^{kbig}$ and $M' = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$ with $M \equiv M' \bmod p^n$ and $v_p(b) = v_p(b') = j$. We start with a more obvious simplification first, which is that for this particular a and b, there is no need to consider M with $c + sp^{kbig-j}$ for any $s \in \mathbb{N}$ since it will also be congruent to M' with $c' + sp^{kbig-j}$. This also applies for $j \geq n$.

With this, we want to show that

Lemma 3. *If M is the first matrix that satisfies the congruence for this particular a, then $a > a'$ if and only if $b = p^j$. Furthermore, we don't need to consider any $b_0 > b$ for this particular a.*

Proof. It is clear that if $b = p^j$, then $a > a'$ since otherwise if $a = a'$, then $b > b'$ so that $b \neq p^j$. Now assume that $a > a'$ and $b = p^j u$ for some $u > 1$ and $p \nmid u$. We can then write $b' = p^j \beta' u$, $c' = \gamma' u^{-1}$, and $c = \gamma u$. We have

$$\begin{pmatrix} a & p^j u \\ \gamma u^{-1} & d \end{pmatrix} \equiv \begin{pmatrix} a' & p^j \beta' u \\ \gamma' u^{-1} & d' \end{pmatrix} \bmod p^n.$$

We can "remove" u and u^{-1} and the modular equation above will still hold. That is,

$$\begin{pmatrix} a & p^j \\ \gamma & d \end{pmatrix} \equiv \begin{pmatrix} a' & p^j \beta' \\ \gamma' & d' \end{pmatrix} \bmod p^n.$$

Hence, the reduced form of the matrix on the left hand side has also been considered before so that we get $b = p^j$. Then, for any $v \in \mathbb{N}$ with $p \nmid v$, we have

$$\begin{pmatrix} a & p^j v \\ \gamma v^{-1} & d \end{pmatrix} \equiv \begin{pmatrix} a' & p^j \beta' v \\ \gamma' v^{-1} & d' \end{pmatrix} \bmod p^n$$

so that we don't need to consider any $b > p^j$ anymore (as we don't need to consider all corresponding c's as well). $\square$

Furthermore, we also have

Lemma 4. *If $a = a'$, and M is the first matrix that satisfies the congruence, then we don't need to consider all $b_0 > b$.*

Proof. Since $b > b'$, let $b = b' + kp^l$ for some $k \in \mathbb{N}$. Note that for $j < n$, $l = n$, otherwise, $l = j$ as we only consider those b with reduced form being p^j. Let $g = \frac{ad-D}{p^j}, r = (b/p^j)^{-1} = (b'/p^j + kp^{l-j})^{-1}$, and $r' = (b'/p^j)^{-1}$. Then we have $c \equiv gr \bmod p^{kbig}$ and $c' \equiv gr' \bmod p^{kbig}$. Hence, we have

$$g(b'/p^j + kp^{l-j})^{-1} \equiv g(b'/p^j)^{-1} \bmod p^n$$

so that

$$kp^{l-j} \equiv 0 \bmod p^{\max(0, n - v_p(g))}.$$

Since M is the first such matrix, $k = p^{\max(0, n-l+j-v_p(g))}$ and $b' = p^j$. Since all $b_0 > b$ have the form $b_0 = b' + (k + k_0)p^l$ with $k_0 \in \mathbb{N}$, and that

$$g(1 + (k + k_0)p^{l-j})^{-1} \equiv g(1 + k_0 p^{l-j})^{-1} \bmod p^n,$$

all matrices of the form $\begin{pmatrix} a & b_0 \\ c_0 & d \end{pmatrix}$ have its reduced form been considered earlier.

$\square$

By Lemma 3 and Lemma 4, for a fixed a, we don't need to consider further b and c values once we know that we have encountered the reduced form before.

Below is the code for the function `helper_fn2`.

Algorithm 4 $\text{HELPER\_FN2}(p, n, k_{\max}, \text{projset}, \text{lift}, \text{rhs\_new}, a, \text{ind}, d, \text{exp})$

$R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
$R_{k_{\max}} \leftarrow \mathbb{Z}/p^{k_{\max}}\mathbb{Z}$
$c_{\text{start}} \leftarrow \left(\frac{\text{rhs\_new}}{\text{ind}}\right) \bmod p^{k_{\max}-\exp}$
if $(R_n(a),\ R_n(c_{\text{start}}),\ R_n(d)) \in \text{projset}$ **then**
    $\text{lifted} \leftarrow \text{lift}[(R_n(a),\ R_n(c_{\text{start}}),\ R_n(d))]$
    if $\text{lifted}[1] = 0$ **or** $\text{valuation}_p(\text{lifted}[1]) = \exp$ **then**
        return false
    end if
end if
$\text{projset} \leftarrow \text{projset} \cup \{(R_n(a), R_n(c_{\text{start}}), R_n(d))\}$
$\text{lift}[(R_n(a), R_n(c_{\text{start}}), R_n(d))] \leftarrow (a,\ R_{k_{\max}}(p^{\exp} \cdot \text{ind}),\ c_{\text{start}},\ d)$
for c **from** $c_{\text{start}} + p^{k_{\max}-\exp}$ **to** $p^n - 1$ **step** $p^{k_{\max}-\exp}$ **do**
    $\text{lift}[(R_n(a), R_n(c), R_n(d))] \leftarrow (a,\ R_{k_{\max}}(p^{\exp} \cdot \text{ind}),\ c,\ d)$
    $\text{projset} \leftarrow \text{projset} \cup \{(R_n(a), R_n(c), R_n(d))\}$
end for
return true

The only different things here is that we now have a dictionary called `lift` that keeps track on the pairing between the reduced matrix (entries in $\mathbb{Z}/p^n$) and the original matrix (entries in $\mathbb{Z}/p^{kbig}$). This is done to make sure that Lemma 2 and Lemma 3 work as we require the assumption that $v_p(b) = v_p(b')$. Furthermore, if $b' = 0$, this means that we have covered all possible reduced form of c, so we don't need to check anything for current a. As a sidenote, all the examples I ran works perfectly fine without this helper function, meaning that one of $v_p(b) = v_p(b')$ or $b' = 0$ is guaranteed in this case. However, I am still having trouble proving it so I am still using this