Ratio 1 and Ratio 2

Richard Adhika

1. Introduction

The first question is that given a prime p and a positive integer n, we want to calculate the ratio

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

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 $a, b, c, d \in \mathbb{Z}/p^n$. 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 correponding determinant constraint. This takes $\mathcal{O}(p^{3n})$, which is very slow. Hence, the function `build_matrices2` 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)

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

If $p \nmid b$, then b^{-1} exists and thus we have a unique solution $c \equiv \text{rhs} \cdot b^{-1} \bmod p^n$ (line 10-12). We don't need to loop through c for these b. If $p \mid b$, then we have three cases (the first two cases are essentially contained within line 14-28 of the code below):

1. $p \nmid$ rhs. This immediately tell us that (1) can't have any solution if $p \mid b$. This is indicated by the variable *pval* (line 16), which will be 0 so that the loop won't execute.

2. $v_p(\text{rhs}) = pval$ for $pval \geq 1$. If $v_p(b) > pval$, then (1) also can't have any solution. Hence, it suffices to check b with $1 \leq v_p(b) \leq pval$ (line 17 with $i = v_p(b)$). Also $pval < n$ so the modulo is well-defined. Then, for each i, let $b = p^i \cdot \text{ind}$ so that we can rewrite (1) as

$$p^i \cdot \text{ind} \cdot c \equiv \text{rhs} \bmod p^n$$

   Dividing both sides by p^i, we have

$$\text{ind} \cdot c \equiv (\text{rhs}/p^i) \bmod p^{n-i}.$$

   We can safely assume that $p \nmid \text{ind}$ (line 21) since otherwise it would either be considered by larger i in future loops or have no solution. Hence, $c \equiv (\text{rhs}/p^i) \cdot \text{ind}^{-1} \bmod p^{n-i}$ (line 22). Then, we lift the solution to modulo p^n by line 23-25.

Algorithm 1 build_matrices2

```
function BUILD_MATRICES2(n, p, target_t, target_det)
    result ← [ ]
    R ← Integers(p^n)
    trace ← R(target_t)
    D ← R(target_det)

    for all a ∈ R do
        d ← trace − a
        rhs ← a · d − D
        for all b ∈ R with p ∤ b do
            result.append(a, b, rhs/b, d)
        end for

        if rhs ≠ 0 then
            rhs ← Integer(rhs)
            pval ← rhs.valuation(p)
            for i = 1 to pval do
                rhs_new ← R(rhs/p^i)
                for ind = 1 to p^(n−i) − 1 do
                    b ← R(p^i · ind)
                    if not p divides ind then
                        c_start ← rhs_new/ind
                        for c = c_start to p^n with step size p^(n−i) do
                            result.append(a, b, c, d)
                        end for
                    end if
                end for
            end for

        else
            for all c ∈ R do
                result.append(a, 0, c, d)
            end for
            for b = p to p^n − 1 step p do
                b ← R(b)
                pval ← b.valuation(p)
                for c = 0 to p^n − 1 step p^(n−pval) do
                    result.append(a, b, c, d)
                end for
            end for
        end if
    end for
    return result
end function
```

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 31-33). Furthermore, for all b with $v_p(b) = pval$, all c with $v_p(c) = n - pval$ satisfy (1) (line 34 - 40).

Notice that in the second case, we have that the number of solutions with $b = b'$ satisfying $v_p(b') = v$ is the same as that of $b = p^v$ as the loops in line 23-25 has the same step size, only with different c_start. Assuming n is not large, this algorithm costs $\mathcal{O}(p^{2n})$, which is already quite an improvement from the brute force function. This is the most optimal function that I can think of so far if we want to genuinely generate every single matrix satifying the numerator condition.

Now, we can modify this function 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 `count_matrices1` that count the number of matrices satisfying the numerator condition in $\mathcal{O}(p^n)$, and thus the first ratio. The code is as follows,

Algorithm 2 count_matrices1

function COUNT_MATRICES(n, $target\_trace$, $target\_det$)
    $count \leftarrow p^n \cdot p^{n-1} \cdot (p-1)$
    $R \leftarrow \text{Integers}(p^n)$
    $trace \leftarrow R(target\_trace)$
    $D \leftarrow R(target\_det)$
    for all $a \in R$ **do**
        $d \leftarrow trace - a$
        $rhs \leftarrow a \cdot d - D$
        if $rhs \neq 0$ **then**
            $pval \leftarrow \text{rhs.valuation}(p)$
            $count \leftarrow count + pval \cdot p^{n-1} \cdot (p-1)$
        else
            $count \leftarrow count + p^n$
            $count \leftarrow count + (n-1) \cdot (p^n - p^{n-1})$
        end if
    end for
    return $count$
end function

Moving on, we also want to compute the second ratio as follows

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

For the second ratio, however, we still need to generate the matrices as the reduction from $kmax$ to n may not include every matrix that satisfies the constraint in $\mathbb{Z}/p^n$ (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 $kmax = 5$, we have around 11 million matrices and for $kmax = 6$, 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 $\mathbb{Z}/p^{kmax}$ and that $p \nmid b$, we only need to consider $a, b \in [0, p^n - 1]$.*

Proof. Consider two matrices $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ and $\begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix}$ with $p \nmid a_2, b_2$. Then, the modular equation $a_2 a_3 \equiv \text{rhs} \bmod p^{kmax}$ has a unique solution $a_3 \equiv \text{rhs} \cdot a_2^{-1} \bmod p^{kmax}$. Furthermore, the modular equation $b_2 b_3 \equiv \text{rhs} \bmod p^{kmax}$ also has a unique solution $b_3 \equiv \text{rhs} \cdot b_2^{-1} \bmod p^{kmax}$. Assume that $b_2 \equiv a_2 \bmod p^n$ and $b_1 \equiv a_1 \bmod p^n$. We therefore have

$$b_4 \equiv t - b_1 \equiv t - a_1 \equiv a_4 \bmod p^n$$

and

$$b_3 \equiv (b_1 b_4 - D) b_2^{-1} \equiv (a_1 a_4 - D) a_2^{-1} \equiv a_3 \bmod p^n.$$

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

This means that the reduction does not affect the uniqueness of solution 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 $p^{2n-1}(p-1)$. Moving on to the next observation, we have

Lemma 2. *Let A be the set of all $M \in GL_2(\mathbb{Z}/p^n)$ such that there exists $M' \in GL_2(\mathbb{Z}/p^{kmax})$ having trace t and determinant D, and that $M = M' \bmod p^n$ (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 $M' \in \mathrm{GL}_2(\mathbb{Z}/p^{kmax})$ 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 $N' \in \mathrm{GL}_2(\mathbb{Z}/p^{kmax})$ be defined by

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

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

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

Finally, we want to show that the map is surjective. Let $N = \begin{pmatrix} a & p^v u \\ cu^{-1} & d \end{pmatrix} \in S'$. Let $M = \begin{pmatrix} a & p^v \\ cu^{-1} & d \end{pmatrix}$. 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 $M \equiv M' \mod p^n$. 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 $1 \le v \le kmax - 1$ and $b' = 0$. The function `count_matrices2`, which counts the number of matrices satisfying the numerator condition, first go through j from 1 to $n-1$, and then for each j, we loop through a from 0 to $p^{kmax} - 1$ and check all corresponding b in $[0, p^{kmax}]$ with $v_p(b) = j$ and $b \equiv p^j \mod p^n$. The complete version of the code will not be included since it is too long (look at the corresponding CoCalc file). However, we will take a look more closely at two helper functions: `helper_fn` and `helper_fn2`, which reduce a lot of repetitions (of b and c) to make the code more efficient.

Below is the code for the function `helper_fn`.

Algorithm 3 Helper Function

function HELPER_FN($p, n, k_{\max}$, projset, rhs_new, a, ind, d, 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 $\mathbb{Z}/p^{kmax}$ 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 $c + sp^{kmax-j}$ for any $s \in \mathbb{N}$ since it will also be congruent to M' with $c' + sp^{kmax-j}$. 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 $c \equiv gr \bmod p^{kmax}$ and $c' \equiv gr' \bmod p^{kmax}$. 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

$$g(1+(k+k_0)p^{l-j})^{-1} \equiv g(1+k_0p^{l-j})^{-1} \bmod p^n,$$

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

□

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 $\textsc{Helper\_fn2}(p, n, k_{\max}, \text{projset}, \text{lift}, \text{rhs\_new}, a, \text{ind}, d, \exp)$

$R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
$R_{k_{\max}} \leftarrow \mathbb{Z}/p^{k_{\max}}\mathbb{Z}$
$c_{\text{start}} \leftarrow \left(\frac{\text{rhs\_new}}{\text{ind}}\right) \bmod p^{k_{\max}-\exp}$
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 $\mathbb{Z}/p^{kmax}$). 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