Appendix A. An algorithm for whether the $\sigma_{i,j}$'s span $\mathrm{Int}(o_L, o_L)$

Dragoș Crișan, Jingjie Yang

Acknowledgements. Shashidhara Balla

A.1. **Introduction.** Let $\mathbb{Q}_p \subseteq L \subsetneq \mathbb{C}_p$ be a field of finite degree d over $\mathbb{Q}_p$, o_L the ring of integers of L, $\pi \in o_L$ a fixed prime element, and $q := |o_L/\pi_L o_L|$ the dimension of the residue field.

For an o_L-submodule S of $L[Y]$ and an integer n, let $S_n = \{f \in S : \deg(f) < n\}$.

Recall that the polynomials $P_m(Y)$ are defined by

$$\exp(Y \cdot \log_{\mathrm{LT}}(Z)) = \sum_{i=0}^{\infty} P_m(Y) Z^m.$$

We will choose the coordinate Z such that $\log_{\mathrm{LT}}(Z) = \sum_{k=0}^{\infty} \pi^{-k} Z^{q^k}$.

Define the upper-triangular matrix $(\sigma_{i,j})_{i,j\geq 0}$ with entries in $L[Y]$ by

$$P_j(Ys) = \sum_{i=0}^{j} \sigma_{i,j}(Y) P_i(s).$$

By Lemma 9.9 in [KL], we know that $\sigma_{i,j}(Y) \in \mathrm{Int}(o_L, o_L)$ and that $\deg(\sigma_{i,j}(Y)) \leq j$. The question is whether the o_L-linear span of $\{\sigma_{i,j}(Y) : 0 \leq i \leq j\}$ equals $\mathrm{Int}(o_L, o_L)$. In this write-up we develop an algorithm to check whether $\big(\mathrm{Int}(o_L, o_L)\big)_n$ is contained in the o_L-linear span of $\{\sigma_{i,j}(Y) : 0 \leq i \leq j < N\}$ for some fixed N, where for convenience we require $q - 1 \mid N$.

A.2. **Theory.**

A.2.1. *Reduction to* $\tau_{i,j}^{(a)}$. To ease notation, for a fixed $a \in \{0, 1, \ldots, q-2\}$, we denote $\underline{i} = a + (q-1)i$.

By Proposition 10.8(2) in [KL], there exists upper-triangular matrices $\tau_{i,j}^{(a)}(Y)$ such that

OldToNew2
$$\sigma_{\underline{i},\underline{j}}(Y) = Y^a \cdot \tau_{i,j}^{(a)}(Y^{q-1}). \tag{16}$$

Definition A.1. For a polynomial $P(x)$, we denote by $\gamma_n(P)$ the coefficient of x^n in P.

Definition A.2. Let M be the o_L-linear span of $\{\sigma_{i,j}(Y) : 0 \leq i \leq j\}$. For a fixed a, let $M^{(a)}$ be the o_L-linear span of $\left\{\sigma_{\underline{i},\underline{j}}(Y) : 0 \leq i \leq j\right\}$. Let $S^{(a)}$ be the o_L-linear span of $\left\{\tau_{i,j}^{(a)}(Y) : 0 \leq i \leq j\right\}$.

nThroughCoeff
Lemma A.3. Let $(f_b^{(a)})_{b\geq 0}$ be a regular basis for $S^{(a)}$ — that is, each $f_b^{(a)}$ has degree b. Then, $M = \mathrm{Int}(o_L, o_L)$ if and only if for all $a \in \{0, 1, \ldots q-2\}$ and $b \geq 0$, we have

$$\nu_\pi(\gamma_b(f_b^{(a)})) = -w_q(a + b(q-1)).$$

Proof. For a fixed $a \in \{0, 1, \ldots q-2\}$, by (16), we have $\gamma_s(\sigma_{i,j}(Y)) = 0$ if $s \not\equiv j \pmod{q-1}$. So, by definition, $M = \bigoplus_{a=0}^{q-2} M^{(a)}$.

We write $S^{(a)}(Y^{q-1}) = \{f(Y^{q-1}) : f \in S^{(a)}\}$. Equation (16) shows that

$$M^{(a)} = Y^a \cdot N^{(a)}(Y^{q-1}).$$

Having chosen a regular basis $(f_b^{(a)})_{b\geq 0}$, these give regular bases $\left(f_b^{(a)}(Y^{q-1})\right)_{b\geq 0}$ for $S^{(a)}(Y^{q-1})$.

So, we get regular bases $\left(Y^a f_b^{(a)}(Y^{q-1})\right)_{b\geq 0}$ for $M^{(a)}$ and thus a regular basis $\{Y^a f_b^{(a)}(Y^{q-1}) : a \in \{0, 1, \dots q-2\}, b \geq 0\}$ for M.

Then, $M = \mathrm{Int}(o_L, o_L)$ is equivalent to $\nu_\pi(\gamma_{a+b(q-1)}(Y^a f_b^{(a)}(Y^{q-1}))) = -w_q(a + b(q-1))$, which is equivalent to $\nu_\pi(\gamma_b(f_b^{(a)})) = -w_q(a + b(q-1))$. □

Let $n = a + b(q-1)$, where a, b are integers, with $a \in \{0, 1, \dots q-2\}$. The proof above shows that a polynomial of degree n with π-valuation of leading term equal to $w_q(n)$ exists in M_N if and only a polynomial of degree b with the same valuation of leading term exists in $S^{(a)}_{N/(q-1)}$. So, the strategy will be to compute regular bases for $S^{(a)}_{N/(q-1)}$.

TauFormula

A.2.2. *A formula for $\tau^{(a)}_{i,j}$.* One advantage of this approach is that the matrices $\tau^{(a)}_{i,j}(Y)$ can be computed quickly. Recall Definition 10.1 of [KL] (where we merely change notation, calling m by a instead):

Definition A.4. For each $j \geq i \geq 0$, let

$$Q_a(i,j) := \left\{ \mathbf{k} \in \mathbb{N}^\infty : \sum_{\ell=0}^{\infty} k_\ell = \underline{i}, \sum_{\ell=1}^{\infty} k_\ell \left(\frac{q^\ell - 1}{q-1} \right) = j - i \right\};$$

$$r^{(a)}_{i,j} := \sum_{\mathbf{k} \in Q_a(i,j)} \binom{\underline{i}}{k_0; k_1; \dots} \cdot \pi^{-\sum_{\ell=1}^{\infty} \ell \cdot k_\ell}.$$

Moreover, define the following upper diagonal matrix of coefficients, which doesn't depend on a.

Definition A.5. Let

$$D_{i,j} = i! \gamma_i P_j(Y).$$

From Proposition 1.20 in outline9, we obtain the following recursion formula, valid for $i \geq 1$:

$$D_{i,j} = \sum_{r \geq 0} \pi^{-r} D_{i-1, j-q^r},$$

with the initial conditions being $D_{0,j} = \delta_{0,j}$.

Now, by Remark 10.4 in [KL] it follows that $r^{(a)}_{i,j} = D_{\underline{i},\underline{j}}$. To tie this back to $\tau^{(a)}_{i,j}$, we introduce one more notation.

Definition A.6.

$$\mathcal{D}_Y := \mathrm{diag}(1, Y, Y^2, \dots)$$

Then, Lemma 10.11 in [KL] gives $\tau^{(a)} = (r^{(a)})^{-1} \cdot \mathcal{D}_Y \cdot r^{(a)}$. This gives a fast algorithm to compute the matrices $\tau^{(a)}$, as the recurrence relation for D allows us to compute $r^{(a)}$ easily.

A.2.3. *Gaussian elimination over a (discrete) valuation ring.* Let R be a (discrete) valuation ring and let A be an $m \times n$ matrix with entries in R. We define notions of elementary row operations and row echelon form over R, similarly to the definitions over a field.

Definition A.7. Given a matrix A as above, the elementary row operations are as follows.

(1) Swap two rows.
(2) Multiply an entire row by a unit in R.
(3) Add an R-multiple of a row to another row.

reserves-span **Lemma A.8.** Performing elementary row operations on a matrix preserves its R-row span.

Proof. For each elementary row operation on A, we define an $m \times m$ matrix B with entries in R such that the result of applying the elementary row operation on A is BA. Observe that in each case, B is invertible, so BA has the same R-row span as A. $\square$

Lemma A.9 (Gaussian Elimination)**.** Let A be a matrix as above. Assume that $m \geq n$ and that A has rank n. Then, one can perform a sequence of elementary row operations on A to produce an upper-triangular matrix of rank n. n_elimination

Proof. We will exhibit an algorithm that puts A in the required form.

We start with the leftmost column. As A has rank n, there is a non-zero entry on column 1. Pick the one with minimal valuation and swap rows, so that the entry on column 0 with minimal valuation is on position $(0, 0)$. Let the new matrix be B.

Then, for each row $i \geq 1$, subtract $\frac{b_{i0}}{b_{00}} \times$ (row 0) from row i. After all of these operations, the matrix has block form:

$$\left[\begin{array}{c|c} b_{00} & * \\ \hline 0 & A' \end{array}\right]$$

where $*$ denotes some $1 \times (n-1)$ matrix, and A' is an $(m-1) \times (n-1)$ matrix. Observe that, as A had rank n and the elementary row operations don't change the rank, A' will have rank $n - 1$.

Now, we can inductively apply the same procedure to A'. Observe that all row operations on A' extend to row operations on the whole matrix that don't change the block structure (as the corresponding entries in the first column are all 0's). By construction, the end result is an upper-triangular matrix, which has the same rank as the initial matrix A. $\square$

A.3. **Implementation.** We focus on the totally ramified extension $L = \mathbb{Q}_p(p^{1/d})$ and the unramified extension of degree d, where we take the prime p, the degree d, and the cutoff N as input parameters.

Fix $a \in \{0, 1, \dots, q-2\}$. Firstly, we compute the matrices $(\tau^{(a)})_{0 \leq i \leq j < N/(q-1)}$ following the method discussed in Section A.2.2. Then, for $s = 0, \dots, N/(q-1) - 1$, we will appeal to the following result to inductively compute a basis $(g_b^{(a),s})_{0 \leq b \leq s}$ for the o_L-span of $\{\tau_{i,j}^{(a)} : 0 \leq i \leq j \leq s\}$, with each $g_b^{(a),s}$ having degree b.

Proposition A.10. Fix $s \geq 0$, and let $(g_b^{(a),s-1})_{0 \leq b \leq s-1}$ be a basis for the o_L-span of $\{\tau_{i,j}^{(a)} : 0 \leq i \leq j \leq s-1\}$ such that each $g_b^{(a),s-1}$ has degree b.

Record the coefficients of these polynomials $g_*^{(a),s-1}$ in s row vectors, and append $s+1$ new row vectors obtained from the coefficients of $\tau_{*,s}^{(a)}$ to obtain the $(2s+1)\times(s+1)$ matrix

$$B := \begin{array}{c@{}l} \begin{array}{cccc} Y^s & Y^{s-1} & & 1 \end{array} & \\ \begin{pmatrix} \bullet & * & \cdots & * \\ & \bullet & \cdots & * \\ & & \ddots & \vdots \\ & & & \bullet \\ * & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{pmatrix} & \begin{array}{l} \tau_{s,s}^{(a)} \\ g_{s-1}^{(a),s-1} \\ \\ g_0^{(a),s-1} \\ \tau_{0,s}^{(a)} \\ \\ \tau_{s-1,s}^{(a)} \end{array} \end{array}$$

with coefficients in L. The $\bullet$'s are non-zero (where $B_{s,0}\neq 0$ because $\sigma_{\underline{s},\underline{s}} = Y^{\underline{s}}$ by Lemma 9.9 of [KL] which by Equation 16 implies that $\tau_{s,s}^{(a)} = Y^s$), so B has rank $s+1$.

Bring the full-rank matrix B to upper-triangular form B' using Gaussian elimination over the discrete valuation ring o_L as per Lemma A.9. Then

(i) we can define the new polynomials $g_s^{(a),s}, g_{s-1}^{(a),s}, \ldots, g_0^{(a),s}$ by reading off the first $s+1$ rows of B', so that each $g_b^{(a),s}$ has degree b and $(g_b^{(a),s})_{0\leq b\leq s}$ form a basis for the o_L-span of $\{\tau_{i,j}^{(a)} : 0\leq i\leq j\leq s\}$;

(ii) for each $b = 0,\ldots,s-1$, the π-adic valuation of the leading coefficient in the new polynomial $g_b^{(a),s}$ is at most that of the old polynomial $g_b^{(a),s-1}$.

Proof. By Lemma A.9 the upper-triangular matrix B' still has rank $s+1$, so it has only non-zero elements on its main diagonal. Hence for each $b = 0,1,\ldots,s$, the polynomial $g_b^{(a),s}$ obtained by reading off the b-th row has degree b. Then of course these polynomials are linearly independent. Also they are the only non-zero rows in B', so by Lemma A.8 their o_L-span is the same as that of the rows of B, which by construction is precisely the o_L-span of $\{\tau_{i,j}^{(a)} : 0\leq 1\leq j\leq s\}$, giving (i).

Now fix $0\leq b\leq s-1$, and consider what happens to the b-th column when we reduce B to B'. Observe that in the proof of Lemma A.9, when we operate on the j-th column for $j = 0,\ldots,s-b-1$, as the row for $g_b^{(a),s-1}$ has a 0 entry in the j-th column, it is neither chosen to be the pivot row nor altered as we subtract off multiples of the pivot row. Thus when we operate on the $(s-b)$-th column to determine the $(s-b)$-th row and column of B', the leading coefficient of $g_b^{(a),s-1}$ must be a candidate for the pivot. But the pivot $B'_{s-b,s-b}$ is chosen to have minimal valuation, so $\nu_\pi(\gamma_b(g_b^{(a),s-1}))\geq \nu_\pi(B'_{s-b,s-b})$. Now $B'_{s-b,s-b} = \gamma_b(g_b^{(a),s})$ by definition, giving (ii). $\square$

For b fixed, it follows that

$$\nu_\pi(\gamma_b(g_b^{(a),s})),\quad s = b, b+1,\ldots$$

is a non-increasing sequence. Moreover, as $g_b^{(a),s}\in S^{(a)}$ can be written as an o_L-linear combination of the $f_i^{(a)}$'s and each $f_i^{(a)}$ is of degree i, we must have $g_b^{(a),s} = \sum_{0\leq i\leq b}\lambda_i f_i^{(a)}$ for some

$\lambda_i \in o_L$; by looking at the leading coefficient, it follows that

$$\nu_\pi(\gamma_b(g_b^{(a),s})) \geq \nu_\pi(\gamma_b(f_b^{(a)})) \geq -w_q(a+b(q-1)).$$

These observations motivate us to look at the following

s_0_def **Definition A.11.** For $n = a+b(q-1)$, let $s_0(n)$ be the minimal $s \geq b$ such that $(g_b^{(a),s})_{0\leq b\leq s}$ satisfies $\nu_\pi(\gamma_b(g_b^{(a),s})) = -w_q(n)$, if such s exists; otherwise set $s_0(n) = \infty$.

Then whenever $s \geq s_0(n)$ in the computations, we can immediately conclude that the equality $\nu_\pi(\gamma_b(f_b^{(a)})) = -w_q(a+b(q-1))$ in Lemma A.3 holds for this $n = a+b(q-1)$.

We may thus make a small optimisation: at any stage s, if $s \geq s_0(a+b(q-1))$ for all $0 \leq b < d$ then we can just drop the last d columns when carrying out Gaussian elimination. Indeed for all $s' > s$ it is unnecessary to compute $(g_b^{(a),s'})_{0\leq b<d}$ as the π-adic valuation of each leading term has already hit the desired minimum, and to compute the leading terms of $(g_b^{(a),s'})_{d\leq b\leq s'}$ we do not need the lower-order terms in the last d columns.

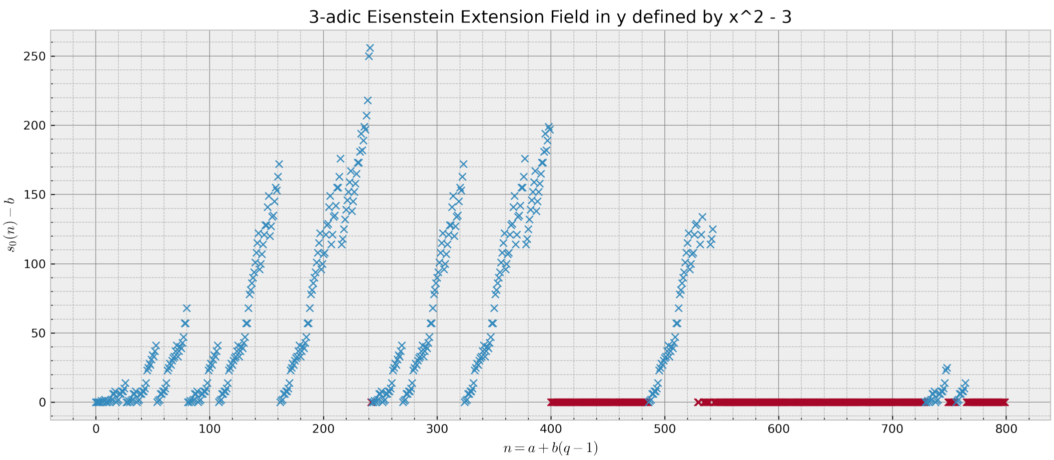

FIGURE 1. `extension = "3,2,800,ram"` — $s_0(n)$ in the quadratic ramified extension $\mathbb{Q}_3(\sqrt{3})$ for $n < 800$. Red points are the n's for which $s_0(n) \geq 800$. 3-2-ram

A.4. **Data.** For reference, the computations in Figure 1 took

- 227.04 seconds for D;
- 616.45 seconds for $\tau^{(0)}$ and 616.43 seconds for $\tau^{(1)}$;
- 0.20 seconds for $s = 50$, 1.89 seconds for $s = 100$, 6.15 seconds for $s = 150$, 12.09 seconds for $s = 200$, etc. for $a = 0$, and slightly less for $a = 1$.

We see that $s_0(n) - b$ seems to depend on the p-adic digits of n; we only managed to prove a special case of this pattern, which we will discuss below. Nonetheless, the data do suggest that $s_0(n)$ is finite for every n and hence that $\operatorname{Int}(o_L, o_L)$ is spanned by the $\sigma_{i,j}$'s as an o_L-module.

A similar pattern emerges for larger p and unramified extensions: see Figures 2 and 3 below.

More data and plots can be found on our GitHub repository.

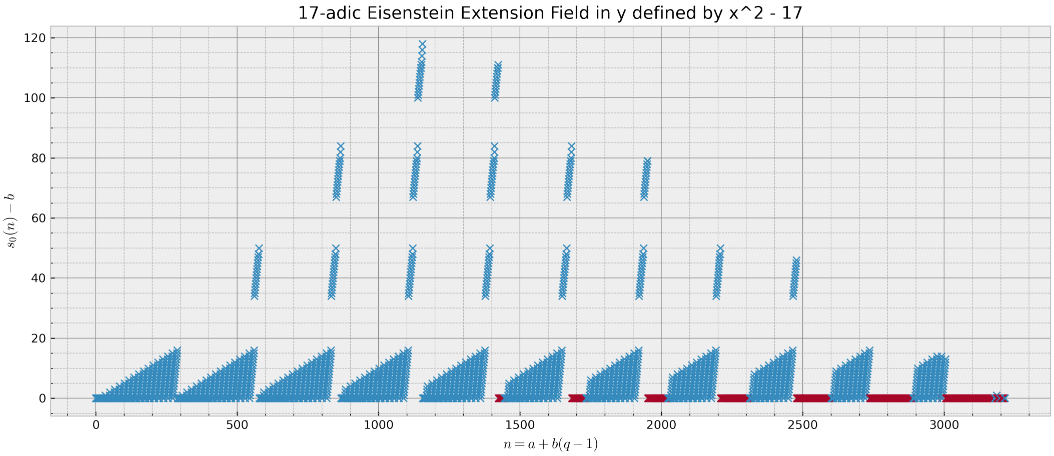

FIGURE 2. **extension = "17,2,3216,ram"** — $s_0(n)$ in the quadratic ramified extension $\mathbb{Q}_{17}(\sqrt{17})$ for $n < 3216$. Note that red points are the n's for which $s_0(n) \geq 3216$ — not enough computation was done to unveil the pattern for the larger n's! `17-2-ram`

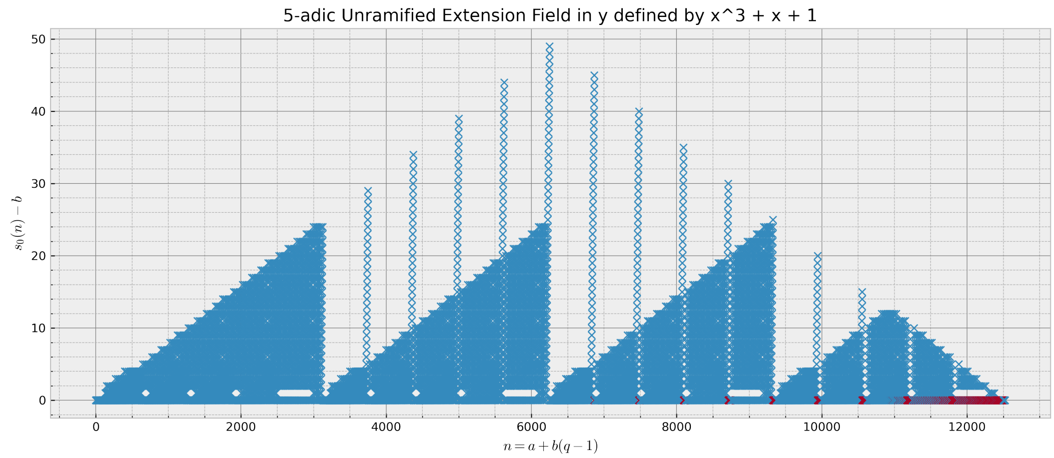

FIGURE 3. **extension = "5,3,12524,unram"** — $s_0(n)$ in the cubic unramified extension of $\mathbb{Q}_5$ for $n < 12524$. Again, note how the red points — the n's for which $s_0(n) \geq 12524$ — give the illusion of $s_0(n) - b$ decreasing. `5-3-unram`

A.5. **Some results.**

Definition A.12. Given a natural number n, let $s_q(n)$ be the sum of digits of n in base q.

Recall Definition A.11:

Definition. For $n = a + b(q-1)$, let $s_0(n)$ be the minimal $s \geq b$ such that $(g_b^{(a),s})_{0\leq b\leq s}$ satisfies $\nu_\pi(\gamma_b(g_b^{(a),s})) = -w_q(n)$, if such s exists; otherwise set $s_0(n) = \infty$.

We define the following more intuitive quantity:

Definition A.13. For $n = a + b(q-1)$, let $\mathrm{Cap}(n) = a + bs_0(n)$. Alternatively, $\mathrm{Cap}(n)$ is the minimal $N \geq n$ such that the o_L-span of $\{\sigma_{i,j} : 0 \leq i \leq j \leq N\}$ contains a polynomial of
Cap_n degree n and π-valuation of the leading term $-w_q(n)$.

Here, the equivalence of the two definitions follows from the definition of $s_0(n)$.

Let $n = a + b(q-1)$. Analysing the computational results, we are led to believe that, if $s_q(n) < p$, then $s_0(n) = b$. This is made clear by the following:

Theorem A.14. Let n be a positive integer such that $s_q(n) < p$. Let $j = n$ and $i = s_q(n)$.
s_q(n)<p Then $\sigma_{i,j}$ is a polynomial of degree n, with π-valuation of leading term equal to $-w_q(n)$.

Recall the definition of the polynomials $c_n(Y)$ from [dSEE09]:

$$[Y](t) = \sum_{n=1}^{\infty} c_n(Y)t^n$$

Translating the definition of the polynomials $\sigma_{i,j}(Y)$, Corollary 9.8 in [KL], we get:

$$([Y](t))^i = \left(\sum_{n=1}^{\infty} c_n(Y)t^n\right)^i = \sum_{j=i}^{\infty} \sigma_{i,j}(Y)t^j$$

Using the binomial theorem, this gives:

$$\sigma_{i,j} = \sum_{n_1+n_2+\ldots+n_i=j} c_{n_1}c_{n_2}\ldots c_{n_i}$$

Of course, for $i = 1$ we obtain $\sigma_{1,j} = c_j$. So, the proof of the Theorem 3.1 in [dSEE09] shows that $\mathrm{Cap}(n) = n$ for n equal to some power of q. We will extend this result to all n that have $s_q(n) < p$, where $s_q(n)$ is the sum of digits of n, written in base q. For this, we need the following lemma:

_q inequality **Lemma A.15.** Let $n_1, n_2, \ldots, n_i$ be positive integers. Then, $w_q(n_1)+w_q(n_2)+\ldots+w_q(n_i) \leq w_q(n_1 + n_2 + \ldots + n_i)$. Equality holds if and only if $s_q(n_1) + s_q(n_2) + \ldots + s_q(n_i) = s_q(n_1 + n_2 + \ldots + n_i)$, that is, if there is "no carrying" in the sum $n_1 + n_2 + \ldots + n_i$.

Proof. Direct calculations show that

$$w_q(n) = \frac{n - s_q(n)}{q-1}$$

Substituting into our inequality, we need to prove

$$s_q(n_1) + s_q(n_2) + \ldots + s_q(n_i) \geq s_q(n_1 + n_2 + \ldots + n_i)$$

which can be checked by direct calculations or by induction. Equality holds in the initial inequality if and only if it holds here, which is to say there is "no carrying" in the sum $n_1 + n_2 + \ldots + n_i$. $\square$

Now, we are ready for:

Proof of Theorem A.14. Recall that

$$\sigma_{i,j} = \sum_{n_1+n_2+\ldots+n_i=j} c_{n_1}c_{n_2}\ldots c_{n_i}$$

where each c_k is a polynomial of degree at most k, with π-valuation of the leading term at least $-w_q(n)$ (as it is in $\mathrm{Int}(o_L, o_L)$).

Let's look at each of the terms $c_{n_1}c_{n_2}\ldots c_{n_i}$. As each c_k has degree at most k, this contributes to the coefficient of Y^k in $\sigma_{i,j}$ if and only if $\deg(c_{n_1}) = n_1, \deg(c_{n_2}) = n_2, \ldots, \deg(c_{n_i}) = n_i$. For the moment, assume this is the case. Then, the coefficient of Y^n in this product is the product of leading coefficients of the c_{n_i}'s, which has π-valuation at least $-(w_q(n_1)+w_q(n_2)+\ldots+w_q(n_i))$. Now, using Lemma A.15, this is at least $-w_q(n_1+n_2+\ldots+n_i) = -w_q(n)$, with equality if and only if $s_q(n_1)+s_q(n_2)+\ldots+s_q(n_i) = s_q(n) = i$, so the n_i's are powers of q. That is, the only contribution to the coefficient of Y^n in $\sigma_{i,j}$ that has small enough valuation comes from permutations of the unique way of writing n as a sum of i powers of q. In other words, if $n = b_r b_{r-1}\ldots b_1 b_{0(q)}$ is the writing of n in base q, then the only terms that have a possible contribution are obtained when $(n_1, n_2, \ldots, n_i)$ is a permutation of $(q^0, q^0, \ldots, q^1, \ldots, q^r)$, where each q^k appears b_k times.

But, by [dSEE09], when k is a power of q, c_k is a polynomial of degree exactly k, with π-valuation of leading term exactly $-w_q(k)$. So, when $(n_1, n_2, \ldots, n_i)$ is a permutation as above, the product $c_{n_1}c_{n_2}\ldots c_{n_i}$ is a polynomial of degree n, with π-valuation of leading term equal to $-w_q(n)$. Moreover, as proved before, if $(n_1, n_2, \ldots, n_i)$ is not such a permutation, the product $c_{n_1}c_{n_2}\ldots c_{n_i}$ has the coefficient of Y^n either 0 or of π-valuation larger than $-w_q(n)$.

As there are $\binom{i}{b_0,b_1,\ldots,b_r}$ such permutations, with $p \nmid \binom{i}{b_0,b_1,\ldots,b_r}$ (because $i < p$ by the initial assumption on n), the final sum $\sigma_{i,j}$ has degree n, with π-valuation of leading term $-w_q(n)$. $\square$

Definition A.13 then gives:

Corollary A.16. Let n be a positive integer such that $s_q(n) < p$. Then $\mathrm{Cap}(n) = n$.

The numerical data suggests that this is the largest set on which $\mathrm{Cap}(n) = n$.

References

[dSEE09] de Shalit E. and Iceland E. Integer valued polynomials and lubin–tate formal groups. *Journal of Number Theory*, (129):632–639, 2009.

[KL] Ardakov K. and Berger L. Bounded functions on character varieties. "bounded26".

A.6. **SageMath Code.** (tested on Sage 9.4)

```
extension = "3,2,100,ram"  # Choose the extension to compute with
precision = 1000           # Choose the precision that Sage will use

parse = extension.split(',')
p = int(parse[0])  # Prime to calculate with
d = int(parse[1])  # Degree to calculate with
N = int(parse[2])  # Cutoff; must be divisible by q-1
ram = parse[3]


```

```
# Python imports
from time import process_time
import matplotlib.pyplot as plt
import numpy as np

# Definitions
from sage.rings.padics.padic_generic import ResidueLiftingMap
from sage.rings.padics.padic_generic import ResidueReductionMap
import sage.rings.padics.padic_extension_generic

power = p^d - 1
t_poly = ""

if ram == "ram":
    t_poly = f"x^{d}-{p}"
else:
    # generate poly for unramified case
    Fp = GF(p)
    Fp_t.<t> = PolynomialRing(Fp)
    unity_poly = t^(power) - 1
    factored = unity_poly.factor()
    factored_str = str(factored)
    start = factored_str.find("^"+str(d))
    last_brac_pos = factored_str.find(")",start)
    first_brac_pos = len(factored_str) \
                    - factored_str[::-1].find("(",len(factored_str)-start)
    t_poly = factored_str[first_brac_pos:last_brac_pos].replace('t','x')


# Define the polynomial to adjoin a root from
Q_p = Qp(p,precision)
R_Qp.<x> = PolynomialRing(Q_p)
f_poly = R_Qp(t_poly)

# Define the p-adic field, its ring of integers and its residue field
# These dummy objects are a workaround to force the precision wanted
dummy1.<y> = Zp(p).ext(f_poly)
dummy2.<y> = Qp(p).ext(f_poly)

o_L.<y> = dummy1.change(prec=precision)
L.<y> = dummy2.change(prec=precision)
k_L = L.residue_field()
print(L)

# Find the generator of the unique maximal ideal in o_L.
Pi = o_L.uniformizer()

# Find f, e and q
f = k_L.degree()              # The degree of the residual field extension
e = L.degree()/k_L.degree()   # The ramification index
```

```
q = p^f

# Do linear algebra over the ring of polynomials L[X]
# in one variable X with coefficients in the field L:
L_X.<X> = L[]
L_Y.<Y> = L[]

v = L.valuation()

# The subroutine Dmatrix calculates the following sparse matrix of coefficients.
# Let D[k,n] be equal to k! times the coefficient of Y^k in the polynomial P_n(Y).
# I compute this using the useful and easy recursion formula
#      D[k,n] = \sum_{r \geq 0} \pi^{-r} D[k-1,n-q^r]
# that can be derived from Laurent's Prop 1.20 of "outline9".
# The algorithm is as follows: first make a zero matrix with S rows and columns
# (roughly, S is (q-1)*Size), then quickly populate it one row at a time,
# using the recursion formula.
def Dmatrix(S):
    D = matrix(L, S,S)
    D[0,0] = 1
    for k in range(1,S):
        for n in range(k,S):
            r = 0
            while n >= q^r:
                D[k,n] = D[k,n] + D[k-1,n-q^r]/Pi^r  # the actual recursion
                r = r+1
    return D


# \Tau^{(m)} in Definition 10.10 of "bounded26":
def TauMatrix(Size, m, D=None):
    if D is None:
        D = Dmatrix((q - 1) * (Size + 1))
    R = matrix(L, Size,Size, lambda x,y: D[m + (q-1)*x, m + (q-1)*y])

    # Define a diagonal matrix:
    Diag = matrix(L_X, Size,Size, lambda x,y: kronecker_delta(x,y) * X^x)

    # Compute the inverse of R:
    S = R.inverse()

    # Compute the matrix Tau using Lemma 10.11 in "bounded26":
    Tau = S * Diag * R

    return Tau

def underscore(m, i):
    return m + i*(q-1)

def w_q(n):
```

```
    return (n - sum(n.digits(base=q))) / (q-1)

def compute_s(N, filename=None):
    assert N%(q-1) == 0

    t_start = process_time()
    D = Dmatrix(N)
    t_end = process_time()
    print(f"D matrix: {t_end-t_start : .2f} sec")

    s0_s = [-1 for _ in range(N)]

    for a in range(q-1):
        t_start = process_time()
        Tau_a = TauMatrix(N//(q-1), a, D)
        t_end = process_time()
        print(f"a={a}, Tau matrix: {t_end-t_start : .2f} sec")

        B_old = Matrix(0,0)
        d = 0
        for s in range(N // (q-1)):
            t_start = process_time()

            # 1. Use the non-zero rows from previous calculations
            # 2. Add a 0 column to its left
            # 3. Add rows corresponding to entries from the j_th column of Tau_a
            B = Matrix(L, 2*s-d+1, s-d+1)
            B[0,0] = 1  # Tau_a[s, s]
            B[1:s-d+1, 1:] = B_old
            for i in [0 .. s-1]:
                coeffs = Tau_a[i, s].list()
                B[s-d+1+i, B.ncols()-len(coeffs)+d:] = vector(L, reversed(coeffs[d:]))

            # Perform Gaussian elimination
            i0 = 0
            ks = []
            for k in range(B.ncols()):
                valuation_row_pairs = [
                    (v(B[i,k]), i) for i in range(i0, B.nrows()) if B[i,k] != 0]

                if not valuation_row_pairs:
                    raise ValueError("B is not full-rank")
                minv, i_minv = min(valuation_row_pairs)
                ks.append(k)

                # Swap the row of minimum valuation with the first bad row
                B[i0, :], B[i_minv, :] = B[i_minv, :], B[i0, :]

                # Divide the top row by a unit in o_L
                u = B[i0, k] / Pi^int(e * v(B[i0, k]))
```

```
                B[i0, :] /= u

                # Cleave through the other rows
                for i in range(i0 + 1, B.nrows()):
                    if v(B[i, k]) >= v(B[i0, k]):
                        B[i, :] -= B[i, k]/B[i0, k] * B[i0, :]

                i0 += 1

            d_is_updated = False
            for b in [d .. s]:
                n = a + b*(q-1)
                if v(B[s-b, s-b]) * e == -w_q(n):
                    if s0_s[n] == -1:
                        s0_s[n] = s
                else:
                    if not d_is_updated:
                        d = b
                        d_is_updated = True
            B_old = B[:s-d+1, :s-d+1]

            t_end = process_time()
            print(f"a={a}, s={s}: {t_end-t_start : .2f} sec", end='\r')
            if filename is not None:
                with open(filename, 'w') as f:
                    f.write("n,s0\n")
                    for n, s0 in enumerate(s0_s):
                        f.write(f"{n},{s0}\n")
        print()

    plt.style.use('bmh')
    fig = plt.figure(figsize=(15,6), dpi=300)
    for n, s0 in enumerate(s0_s):
        if s0 != -1:
            b = n // (q-1)
            plt.plot(n, s0-b, 'x', c='C0')
        else:
            plt.plot(n, 0, 'x', c='C1')
    plt.xlabel(r"$n = a + b(q-1)$")
    plt.ylabel("$s_0(n) - b$")
    plt.title(str(L))
    plt.minorticks_on()
    plt.grid(which='both')
    plt.grid(which='major', linestyle='-', c='grey')

    return s0_s, fig


s0_s = compute_s(N);
```