Towards Fast Encoding of Garcia-Stichtenoth Algebraic Geometry Codes

Matthew Weidner
Rita A. and Øistein Skjellum SURF Fellow
Mentor: Chris Umans
Co-mentor: Anand Narayanan

October 27, 2017

Abstract

Error-correcting codes are ways to "encode" a block of bits as a longer, redundant series of bits, such one that can decode the original message even if the encoded message has suffered some errors. Garcia-Stichtenoth algebraic geometry codes encode messages by interpreting them as functions in Riemann-Roch spaces of curves in the Garcia-Stichtenoth tower, then evaluating those functions at many rational points on their curve. They improve upon Reed-Solomon codes, which are used frequently in everyday life, by having arbitrarily long blocks while still allowing one to correct a constant proportion of errors. However, the current best encoding algorithm is slow, outputting encoded messages of length N in time $O((N \log(N))^3)$. In this project, we make progress towards a faster algorithm, by attempting to construct "explicit" bases for Riemann-Roch spaces of curves in the Garcia-Stichtenoth tower. We succeed in extending existing explicit bases to the third level of the tower. We also construct explicit bases for related but weaker codes over the field with four elements. Additionally, we propose several pathways towards solving the general problem.

1 Introduction

1.1 Background

Error correcting codes are everywhere in modern life. Basically, they are ways to "encode" a message by adding redundant information, so that the original message can be decoded even if one receives an error-laden copy of the encoded message. A simple example appears in the Universal Product Code (UPC) which appears on merchandise in most stores.

The bars on a UPC label correspond to the 12 numbers appearing along the bottom. The translation from numbers to bars does not constitute an error correcting code, but the 12 numbers do: only the first 11 numbers are arbitrary, while the 12th number is chosen so that

$$3(x_1 + x_3 + x_5 + x_7 + x_9 + x_{11}) + (x_2 + x_4 + x_6 + x_8 + x_{10} + x_{12})$$

Figure 1: A UPC-A label. Note that $3 \times (8+2+8+0+1+0) + (9+6+5+0+0+3) = 80$ is divisible by 10.[†]

is divisible by 10, i.e., its decimal representation ends in a 0.[‡] Here x_i indicates the i-th number. Because this is no longer true if any of the numbers changes, the barcode reader can detect when it has read one number incorrectly and ask the cashier to scan again. In other words, the UPC encoding is *1 error detecting*. The barcode reader can also correct certain kinds of errors: if a barcode is read incorrectly but the read message differs from an allowed message in only one digit, and there is only one such allowed message, then it is reasonable for the barcode reader to assume that the actual message equals that allowed message. However, there are some barcodes which cannot be corrected in this way even if only one error occurs.

One might wonder whether we can do better — is it possible to design error correcting codes such that one can detect or even correct a large number of errors? It turns out that the answer is yes, and very much so. In fact, *random codes* — where we fix an input message length and a larger encoded message length, then map each input message to a random message with the desired length — often allow one to detect a number of errors almost equal to the number of redundant symbols (i.e., the encoded message length minus the input message length).

However, random codes are somewhat impractical. There are three general parts to making a code "practical":

- A guarantee on how many errors the code can detect/correct;
- A fast algorithm (computer program) for encoding messages; and
- A fast algorithm for decoding received messages, i.e., finding a message (or messages) which most likely resulted in the received message after being encoded and possibly suffering errors.

All three parts are hard to implement for random codes, making them not always the best choice despite their good error correcting abilities. Additionally, if two parties might want to communicate several versions of a random error correcting code, then they must agree on each random code in advance, which requires storing a lot of data.

Instead, many practical applications use a more structured kind of error-correcting code called a *Reed-Solomon code*. These are used in CDs, QR codes, and many other places, to ensure that one can get the correct data even if the product is damaged or the reader makes occasional mistakes.[§]

[†]Image source: `https://images-na.ssl-images-amazon.com/images/G/02/rainier/make-money-css/UPC.gif`

[‡]Source: `https://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf`, §7.9

[§]Sources: Barry Cipra. The Ubiquitous Reed-Solomon Codes. *SIAM News*, 26(1), January 1993; and `http://www.qrcode.com/en/about/error_correction.html`

Figure 2: A QR code which can be read correctly despite many missing bits, due to Reed-Solomon error correction.[‖]

To define Reed-Solomon codes, fix a prime power q. There is a special set of q numbers or tuples of numbers, called the *finite field of size* q, such that one can "add", "subtract", "multiply", and "divide" elements of the set, and these operations satisfy the usual rules (e.g., $a(b+c) = ab+ac$)). We treat messages as sequences of elements in this finite field; this is not a restriction since, as usual, we can easily translate between sequences in the finite field and strings of bits. Also fix a number $1 \leq k \leq q-1$. To encode a message consisting of k finite field elements $a_1, \ldots, a_k$, consider the polynomial

$$f(x) = a_1 + a_2 x + \cdots + a_k x^{k-1},$$

and output f applied to each of the q elements in the finite field. Then it turns out that, if we start with two different messages, then the encoded messages differ in at least $q+1-k$ places. Hence we can detect if a received message has suffered up to $q-k$ errors (since then it will no longer equal the encoding of any input message), and we can recover the original message even if a received message has suffered up to $(q+1-k)/2$ errors.

It turns out that this is the best we can do for error correcting codes over the finite field of size q.[**] In addition, Reed-Solomon codes have fast encoding and decoding algorithms, and one can program all possible Reed-Solomon codes (for all choices of q and k) in a single algorithm. Thus it is not hard to understand why they are used so widely in practical applications.

1.2 A Problem

However, Reed-Solomon codes have one disadvantage: over the finite field of size q, one can only define codes which work on messages of $k < q$ finite field elements at a time. Of course, if we have a longer message, we can just break it up into "blocks" of k elements each. But if we have a long message which must be broken into blocks, then too many errors in one block can make the message unreadable even if the overall proportion of errors is very low. One way to get around this is to guess which kind of errors are likely to occur in a given situation and to spread out different blocks' encoded messages, so that any particular block is unlikely to have many errors.

Another solution to this problem is to increase q. However, this has two disadvantages:

[‖]Image source: `https://commons.wikimedia.org/wiki/File:QR_Code_Damaged.jpg`

[**]Technically, this is only true when we restrict ourselves to *linear codes* — error-correcting codes whose encoding function is a linear operator when the set of all element strings with a fixed length is considered naturally as a vector space over the finite field.

- Working with elements in larger finite fields is more computationally difficult, slowing down the encoding and decoding algorithms; and
- Elements of larger finite fields have longer representations as sequences of bits, so that even if the chance of making an error in each bit stays the same, the chance of making an error in each element (represented as a sequence of bits) increases.

Thus it is an interesting question, both practically and theoretically, to ask whether we can define error correcting codes over a finite field of size q which are as good (or almost as good) as Reed-Solomon codes, but can encode arbitrarily long messages in one block.

1.3 A Solution?

It turns out that the answer is yes, due to a class of codes called *algebraic geometry codes*. These codes generalize Reed-Solomon codes, as follows. We can think of the elements of the finite field of size q as the points on a line in a special kind of space. Reed-Solomon codes then treat messages as nice functions (namely polynomials) mapping points on the line to elements of the finite field, and encode messages by evaluating their functions at points on the line. Algebraic geometry codes replace the line with any curve, which consists of a special set of points in some n-dimensional space over the finite field of size q (i.e., the space of all n-tuples of elements of the finite field); then we encode messages by treating them as functions of points on the curve and evaluating those functions at many points.

Algebraic geometry codes are not quite as good as Reed-Solomon codes, but they are close. Recall that, for any fixed $1 \leq k \leq q-1$, we have a Reed-Solomon code which turns messages of length k into encoded messages of length q, such that any two encoded messages differ in at least $q+1-k$ symbols. We call k the *dimension*, the length of encoded messages (in this case q) the *length* N, and the minimum number of differing symbols (in this case $q+1-k$) the *minimum distance* d. We always have $d+k \leq N+1$, and for Reed-Solomon codes, $d+k = N+1$. We can rewrite this as $\delta + R = (N+1)/N \approx 1$, where $\delta = d/N$ is the *relative distance* and $R = k/N$ is the *rate*.

Using algebraic geometry codes, for any fixed q, we can construct codes of arbitrarily high N over the finite field with q elements, such that

$$\delta + R \geq 1 - \frac{1}{\sqrt{q} - 1}$$

for every such code, at least when q is a perfect square. Hence for moderate values of q, this is close to the performance of Reed-Solomon codes.

In fact, for $q \geq 49$, this performance is better than that of random codes! The possible δ and R values which random codes can achieve for arbitrarily large lengths N are given by the *Gilbert-Varshamov bound*, which we can plot as a curve in terms of δ and R. For many values of R and q, this curve lies "below" the line $\delta + R = 1 - \frac{1}{\sqrt{q}-1}$ (see Figure 1.3). This reinforces the fact that algebraic geometry codes are "good codes", and explains why they have attracted both practical and theoretical interest.

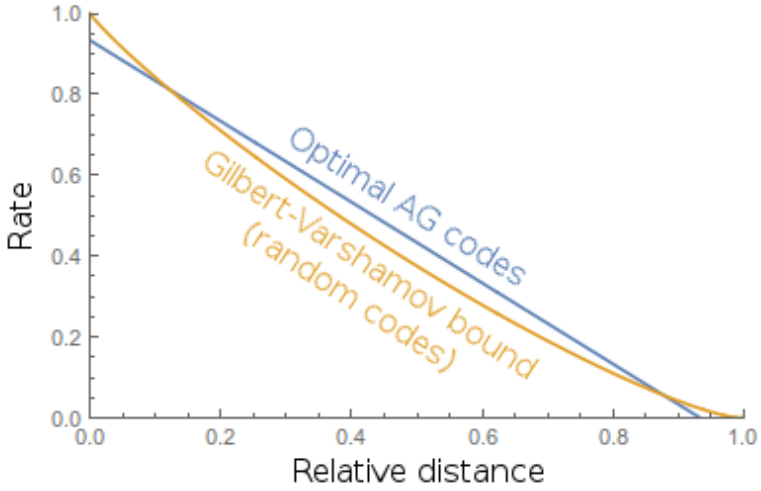

Figure 3: Performance comparision for optimal algebraic geometry codes vs. the Gilbert-Varshamov bound, which indicates the performance of random codes, over the finite field with 256 elements. Points on the graph indicate combinations of rate and relative distance which can be achieved for arbitrarily long codes.

1.4 Our Project

However, we are not done yet — we still need a way to encode and decode messages for algebraic geometry codes. As one might expect, this turns out to be much more difficult and mathematical than it is for Reed-Solomon codes, and existing algorithms are relatively slow. Our goal was to find new and faster ways to encode algebraic geometry codes, especially for a certain class of curves forming the *Garcia-Stichtenoth tower*; these curves give the best possible codes while having relatively simple graphs.

We have not succeeded in constructing a complete algorithm faster than the existing one. However, we have made promising progress on several approaches which differ substantially from existing algorithms. In particular, we have made a reasonably fast algorithm for codes which are about half as good as the best algebraic geometry codes (namely, $\delta + R \approx 1 - 2/(\sqrt{q} - 1)$ instead of $1 - 1/(\sqrt{q} - 1)$), we have developed two fast algorithms for encoding the easiest "unsolved" curve in the Garcia-Stichtenoth tower, and we have developed a fast algorithm for a related problem in the (unfortunately practically useless) case $q = 2$.

2 Results

We have focused exclusively on fast construction of regular functions in the Garcia-Stichtenoth tower. The Garcia-Stichtenoth tower[1] is a simple explicit tower of function fields meeting the Drinfeld-Vlăduţ bound. To define the tower, let q be a prime power. The Garcia-Stichtenoth

tower over $\mathbb{F}_{q^2}$ is defined by $F_0 = \mathbb{F}_{q^2}(x_0)$, and $F_{i+1} = F_i(x_{i+1})$ where x_{i+1} satisfies the relation

$$x_{i+1}^q + x_{i+1} = \frac{x_i^q}{x_i^{q-1} + 1}.$$

We let $P_\infty^{(i)}$ denote the unique pole of x_0 in F_i. Typically one constructs algebraic-geometry codes using the divisor $D = rP_\infty^{(i)}$ for some $r > 0$. In this case, the elements of $\bigcup_{r \geq 0} \mathcal{L}(rP_\infty^{(i)})$ are called regular functions, and for any regular function f, its pole order at $P_\infty^{(i)}$ is called its weight.

We had two related but distinct goals:

- Given a prime power q, $n \geq 0$, and $r \in [q^{n+1}, q^{n+1} + q^n - 1]$, construct a regular function of weight r in $F_n/\mathbb{F}_{q^2}$ in time $\text{poly}(n, q)$, where $F_n/\mathbb{F}_{q^2}$ is the n-th level of the Garcia-Stichtenoth tower over $\mathbb{F}_{q^2}$. This would mostly prove Conjecture 1.4 in Cohen & Ta-Shma.[2]
- Given a prime power q, $n \geq 0$, and an element of a q^n-dimensional $\mathbb{F}_{q^2}$-vector space, construct the image of that vector under some isomorphism with the space of regular functions of weight between q^{n+1} and $q^{n+1} + q^n - 1$ in $F_n/\mathbb{F}_{q^2}$. The current best algorithm[3] does so in time $O((nq^{n+2})^3) = (q^n)^3\text{poly}(q, n)$; my goal is $q^n\text{poly}(q, n)$, but even $(q^n)^2\text{poly}(q, n)$ would be a sizeable improvement.

We have made progress towards these goals by treating special cases or weaker versions, as detailed below.

2.1 Explicit Basis for Regular Functions in F_3

Mathematically "explicit" bases for regular functions in F_0 and F_1 are easy, and an explicit basis for F_2 was found independently by Shum et. al.[3] (see §VI) and Pellikaan.[4] We have found an explicit basis for F_3, at least when $q > 2$ is even. The regular functions are somewhat complicated to describe, but they can be written down in time $O(q^3)$.

2.2 Weaker, But Faster, Codes Beating the Gilbert-Varshamov Bound

Shum et. al.'s algorithm[3] takes time $(q^n)^3\text{poly}(q, n)$ to write down a generator matrix for codes over F_n, after which encoding is faster ($(q^n)^2\text{poly}(q, n)$ time to multiply the generator matrix by an input vector). One could eliminate this cubic time bottleneck if we could somehow build the generator matrix for F_n from the one for $F_{n/k}$ for some k. We have not found a way to do this, but we can make slightly weaker codes. Specifically, for any $k \geq 1$, we can make arbitrarily long codes lying on or above the line $R + \delta = 1 - \frac{kq+k-1}{q^2-q}$, with the encoding algorithm for the n-th level (length $\approx q^{n+2}$) requiring $O(n^3 q^{3\lceil n/k \rceil + 4} + q^{n+\lceil n/k \rceil + 4})$ finite field operations over $\mathbb{F}_{q^2}$. When $k = 2$, these parameters can exceed the Gilbert-Varshamov bound when $q \geq 19$, with encoding in time approximately $O(N^{1.5})$ where N is the length.

The main idea is to make mostly regular functions by taking regular functions from $F_{n/k}$, "shifting" them upwards by multiples of n/k (i.e., replacing x_i with $x_{i+n/k}$, etc.), and multiplying the shifted functions. It turns out that we can describe shifting in terms of a simple morphism, due to the symmetry in the definition of the Garcia-Stichtenoth tower, and use this isomorphism to reason about the pole degrees of our functions.

2.3 Mostly Regular Functions when $q = 2$ in time $O(n)$

Due to details of the Garcia-Stichtenoth tower, it is easier to make functions which are regular except for a certain pole degree at each place, as opposed to truly regular functions. Essentially, the idea is to avoid the x_i which contributes the most to each pole degree and leave the poles of those that contribute less. We have developed an $O(n)$ time algorithm for constructing such functions of arbitrary weight when $q = 2$. This case is not of practical interest, since the tower does not give good codes when $q = 2$, but we hope that it may serve as a model for general q.

We start by defining our output functions via a "verbose" recursion involving a sum over functions constructed in lower levels of the tower; see Section 3.3 for the definition. From the recursion and a rewritten form of it, we prove the pole degree bounds that we want. We then use similarities between the recursive forms in the n-th, $(n-1)$-st, and $(n-2)$-nd levels to give an alternate Fibonacci-style recursion for our functions, which we solve in $O(n)$ time using dynamic programming.

2.4 Iterated Corrections

A simple general idea for constructing regular functions is to start with an "obvious" non-regular function of a given weight, then add correction terms to "fix" this function at various places (i.e., the sum is regular at those places). These new terms may themselves need to be fixed at other places through more correction terms, etc.

To make such an approach work, we need two things:

- Correction terms to fix any pole that might arise; and
- An algorithm which implements the repeated fixing process quickly and without looping forever.

We have mostly addressed the first task in general, by carefully using results on power series expansions from Shum's thesis.[5] The second task appears more difficult. So far, we have succeeded in developing an algorithm for $n = 3$ which is guaranteed to terminate, giving us a nearly linear time encoding algorithm in that case. This method appears cleaner and more generalizable than the algorithm for constructing an explicit basis for F_3 described above; for example, it works for all q, not just $q > 2$ even.

2.5 A New, But Slower, Algorithm for Regular Functions

We have developed a new algorithm for constructing regular functions in general, which basically operates by moving all bad poles into bad poles at the infinite place. We do so in a way which ensures that we can cancel these bad poles using regular functions that we have constructed recursively. However, it turns out that the power series of the bad poles at infinity has order proportional to about q^{2n}, so it apparently takes time about q^{4n} to compute. Thus our algorithm can at best run in time $(q^n)^4\text{poly}(q, n)$, much worse than Shum et. al.'s algorithm.[3] By settling for not-quite-regular functions, we can reduce this bottleneck to about q^{2n}; however, these functions do not give good codes as n goes to infinity.

2.6 Future Work

Our immediate goal is to generalize the results described in Section 2.4 to $n > 3$. The $n = 4$ case already presents difficulties, and the general case appears hard. We hope that this might lead to an encoding algorithm with runtime $(q^n)^2 2^n \text{poly}(q, n)$.

We also hold out hope that the $O(n)$ time algorithm when $q = 2$ can generalize to $q > 2$, since that is our only result which hints at a $\text{poly}(n, q)$ time algorithm to construct regular functions.

3 Methods

3.1 Basis for Regular Functions in F_3, $q \geq 4$ even

Let $q \geq 4$ be an even prime power. Let F_3 be the third level of the Garcia-Stichtenoth tower over $\mathbb{F}_{q^2}$, with notation as in Shum et. al.[3] (In particular, $F_0 = \mathbb{F}_{q^2}(x_0)$.) By work of Pellikaan, Stichtenoth, and Torres,[6] the allowed weights of regular functions are exactly those coming from regular elements of F_2 together with all weights $\geq q^4 - q^2$. (The weight of a function is defined to be the negative of its valuation at $P_\infty^{(3)}$.) We here describe how to find, for all $r \geq q^4 - q^2$, a regular function of weight r in F_3. Throughout this section, all equalities will hold not just in F_3 but in the polynomial ring $\mathbb{F}_{q^2}[x_0, x_0^{-1}, \ldots, x_3, x_3^{-1}]$ — that is, we will not use any of the nontrivial relations defining F_3.

The range $q^4 - q^2 \leq r \leq q^4$ is easy: merely take a regular element of weight $r - q^4 + q^3$ in F_2, shift its indices up by 1, and multiply by π_0.

Now let $q^4 < r < q^4 + q^3$, $q \nmid r$. (We can obtain all higher weights through multiplication by powers of x_0.) Write $r = q^4 + e_1 q^2 + e_2 q + e_3$. Let $u_2 = x_2 + \frac{x_1^2}{x_0}$, which is regular at $S_1^{(3)}$. We know that the element

$$u_3 := x_3 + \frac{u_2^2}{x_1} = x_3 + \frac{x_2^2}{x_1} + \frac{x_1^3}{x_0^2}$$

is regular at $S_{2,4}^{(3)}$ and has a pole of order 1 at $S_1^{(3)}$. (Recall we are in characteristic 2.) We start by considering the element $v := \pi_0 x_0 x_1^{e_1} u_2^{e_2} u_3^{e_3}$, which has the desired weight and is regular except possibly at $S_0^{(3)}$. Binomial expand this element as

$$v = \pi_0 x_0 \sum_{k=0}^{e_2} \sum_{j=0}^{e_3} \binom{e_2}{k} \binom{e_3}{j} x_2^{e_2-k} \left(x_3 + \frac{x_2^2}{x_1}\right)^{e_3-j} \frac{x_1^{e_1+2k+3j}}{x_0^{k+2j}}. \tag{1}$$

Next, let w be the result of applying the following two rules to the above element:

(i) In the binomial expansion of $x_2^{e_2-k} \left(x_3 + \frac{x_2^2}{x_1}\right)^{e_3-j}$, whenever a term x_2^i appears with $i \geq q$, replace i with

$$i' := \begin{cases} q & \text{if } i = 2q - 1 \\ i' \in [1, q-1], i' \equiv i \pmod{q-1} & \text{else.} \end{cases}$$

(ii) Whenever $k+2j \geq q$, replace $\frac{1}{x_0^{k+2j}}$ with $\frac{1}{x_0^{i'}}$, where i' is as in (i).

This w has several important properties. To explain the first, write w in the form

$$w = \pi_0 x_0 \sum_{k=0}^{e_2} \sum_{j=0}^{e_3} z_{k,j} \frac{x_1^{e_1+2k+3j}}{x_0^{a_{k,j}}}, \tag{2}$$

where $a_{k,j}$ is the i' corresponding to $k+2j$, and where $z_{k,j}$ is a sum of monomials in x_1, x_2, x_3, none of which contain a positive power of x_1, and at least one of which contains no power of x_1.

I claim that each $z_{k,j}$ is regular at $S_{2,4}^{(3)}$. To see this, note that $z_{k,j}$ is just the result of applying rule (i) to $\binom{e_2}{k}\binom{e_3}{j} x_k^{e_2-k} \left(x_3 + \frac{x_2^2}{x_1}\right)^{e_3-j}$. The difference between this element and $z_{k,j}$ is a sum of monomials of the form $x_3^l \frac{x_2^s}{x_1^m}(x_2^q + x_2)$, where $l, s, m \geq 0$ and $l+m = e_3 - j < q$. Such a monomial is regular at $S_{2,4}^{(3)}$, so each $z_{k,j}$ is regular at $S_{2,4}^{(3)}$.

It follows (using $e_1 + 2k + 3j \geq 1$ and $a_{k,j} < q$ for $jk \neq 0$, and $a_{0,0} = 0$) that w is regular at $S_{2,4}^{(3)}$. I claim that w is regular at $S_1^{(3)}$ as well. In addition, I claim that in the full expansion of w as a sum of monomials $\pi_0 x_0 x_3^{i_3} x_2^{i_2} x_1^{i_1} x_0^{i_0}$, we always have $i_2 - i_0 \in [0, q]$.

To prove the above two claims, write

$$v = \pi_0 x_0 x_1^{e_1} \sum_{j=0}^{e_3} \binom{e_3}{j} x_3^{e_3-j} \frac{u_2^{e_2+2j}}{x_1^j}. \tag{3}$$

Let w_2 be the result of replacing each u_2^i such that $i \geq q$ with

$$\begin{cases} u_2^{i-q}\left(x_2 + \frac{x_1^{2q}}{x_0}\right) & \text{if } i \in [q, 2q-1] \\ u_2^{i-2q}\left(x_2^2 + \frac{x_1^{4q}}{x_0^2}\right) & \text{if } i \in [2q, 3q-3]. \end{cases}$$

(Note that $e_2 + 2j \in [0, 3q-3]$ always.) Then $w_2 = w$, as follows. The right-hand sides of (1) and (3) have the same expansion as sums of monomials times $\pi_0 x_0$, so the powers of x_2 at least q in some summand of (1) are in bijection with the powers of x_2 at least q in some $u_2^{e_2+2j}$ in (3). Now $u_2^{e_2+2j}$ only contains a power of x_2 at least q if $e_2 + 2j \geq q$, in which case we can write either

$$u_2^{e_2+2j} = u_2^q u_2^m = \left(x_2^q + \frac{x_1^{2q}}{x_0^q}\right) u_2^m$$

or

$$u_2^{e_2+2j} = u_2^{2q} u_2^m = \left(x_2^{2q} + \frac{x_1^{4q}}{x_0^{2q}}\right) u_2^m$$

because q is a power of the characteristic, for some $m \in [0, q-1]$ (note that in the second case, we in fact have $m \in [0, q-3]$). Thus in the expansion of v as a sum of monomials times $\pi_0 x_0$, at least with respect to powers of x_2, the rule (i) applied when yielding w and the substitution $u_2^{e_2+2j} \mapsto u_2^{i'}$ yielding w_2 perform the same substitution in all monomials containing a power of x_2 at least q. Similarly, the substitution yielding w_2 performs one of the substitutions

$$\frac{x_1^{2q}}{x_0^q} \mapsto \frac{x_1^{2q}}{x_0}$$

or

$$\frac{x_1^{4q}}{x_0^{2q}} \mapsto \frac{x_1^{4q}}{x_0^2},$$

giving the same result as the substitution caused by rule (ii).

Now the second claim about w — that we always have $i_2 - i_0 \in [0, q]$ — is obvious because it holds for w_2. Also, from its original definition, easily w_2 is regular at $S_1^{(3)}$, hence w is regular at $S_1^{(3)}$.

Next, consider again the expansion in (2), and let f be the result of replacing each $x_1^{e_1+2k+3j}$ such that $e_1 + 2k + 3j \geq q$ with $x_1^{c_{k,j}}$, where $c_{k,j} \in [1, q-1]$, $i' \equiv e_1 + 2k + 3j \pmod{q-1}$. We can write $f - w$ as a sum of terms of the form

$$\pi_0 x_0 x_3^{i_3} x_2^{i_2} x_1^t x_0^{i_0} (x_1^{q-1} + 1)$$

with $i_3, i_2 - i_0 \in [0, q]$, each of which is regular at $S_1^{(3)}$ (noting that the leading x_0 is sufficient to cancel the pole due to $x_3^{i_3}$). Hence f is regular at $S_1^{(3)}$ as well. Now writing

$$f = \pi_0 x_0 \sum_{k=0}^{e_2} \sum_{j=0}^{e_3} z_{k,j} \frac{x_1^{c_{k,j}}}{x_0^{a_{k,j}}}, \tag{4}$$

with all $a_{k,j} c_{k,j} \in [1, q]$ except for $a_{0,0} = c_{0,0} = 0$, it is easy to see that f is regular at $S_0^{(3)}$ and has weight r. It is also regular at $S_{2,4}^{(3)}$ for the same reasons as w. Finally, f is regular at all places outside $\{P_\infty^{(3)}\} \cup S_{0,4}^{(3)}$ because we can write it as a sum of monomials in the x_i, and any monomial is regular at all such places (because such places assign a value in $\overline{\mathbb{F}}_{q^2}^\times$ to each x_i). Hence f is the desired element. Easily we can write down f in time $O(q^3)$.

3.2 Weaker, But Faster, Codes Beating the Gilbert-Varshamov Bound

3.2.1 Shifting

Let $f = x_m^{e_m} \cdots x_0^{e_0}$ be a monomial in F_m. For $i \geq 0$, we define the *shift of f by i* to be the element

$$f[i] := x_{m+i}^{e_m} \cdots x_i^{e_0} \in F_{m+i}.$$

We extend the definition of shift $\mathbb{F}_{q^2}$-linearly to all of F_m.

Proposition 3.2.1. *(a) For any $f \in F_m$ and $i \geq 0$, $f[i]$ is well-defined, i.e., it does not depend on the representation of f as a sum of monomials.*

(b) Let $f \in F_m$ be regular of weight r. Then:

- *$f[i]$ has weight r*
- *$f[i]$ is regular at $S_{i,m+i+1}^{(m+i)}$*
- *For $t \in [0, i-1]$, for all $P \in S_t^{(m+i)}$, we have*

$$v_P(f[i]) = \begin{cases} -r & \text{if } t \leq \frac{m+i}{2} \\ -rq^{2t-(m+i)} & \text{if } t > \frac{m+i}{2} \end{cases}$$

- $\deg\left((f[-i])_\infty^{(m+i)}\right) = rq^i$.

Proof. For all j, we have the isomorphism

$$\begin{aligned}\phi_j : F_j &\xrightarrow{\sim} F_j \\ x_k &\mapsto x_{j-k}^{-1},\end{aligned}$$

which is its own inverse (see p. 2237[3]). It is easy to see that

$$f[i] = \phi_{m+i}(\phi_i(f)),$$

proving (a).

To prove (b), we use the fact that ϕ_j induces bijections $S_t^{(j)} \leftrightarrow S_{j-t}^{(j)}$ for each $t \in [0, j]$, together with a correspondence $P_\infty^{(j)} \leftrightarrow S_{j+1}^{(j)}$ (see p. 2237[3]). Thus letting $f \in F_m$ be regular of weight r, $\phi_m(f)$ is regular at all places (including $P_\infty^{(j)}$) except for a pole of order r at $S_{j+1}^{(j)}$. Then by the ramification behavior of the tower (see §II[3]), $\phi_m(f) \in F_{m+i}$ is regular at all places (including $P_\infty^{(j)}$) except for $S_{m,m+i+1}^{(m+i)}$. In particular, for $t \in [m, m+i+1]$, for all $P \in S_t^{(m+i)}$, we have

$$v_P(\phi_m(f)) = \begin{cases} -r & \text{if } t \geq \frac{m+i}{2} \\ -rq^{m+i-2t} & \text{if } t < \frac{m+i}{2} \end{cases}$$

Then $f[i] = \phi_{m+i}(\phi_m(f))$ easily has the first three properties in (b). The fourth property follows either from computing the total pole degree directly, or from noting that

$$\begin{aligned}\deg\left((f[i])_\infty^{(m+i)}\right) &= \deg\left((\phi_m(f))_\infty^{(m+i)}\right) \\ &= [F_{m+i} : F_m] \deg\left((\phi_m(f))_\infty^{(m)}\right) \\ &= q^i \deg\left((f)_\infty^{(m)}\right) \\ &= rq^i.\end{aligned}$$

□

3.2.2 Mostly Regular Functions

Fix $k \geq 1$. By shifting, we will construct mostly regular functions in F_n using regular functions in $F_{\lceil n/k \rceil}$.

Specifically, let $r \in [0, q^n - 1]$. Write $r = r_1 q^{n-\lceil n/k \rceil} + r_2 q^{n-2\lceil n/k \rceil} + \cdots + r_{k-1} q^{n-(k-1)\lceil n/k \rceil} + r_k$, with $r_1, \ldots, r_{k-1} \in [0, q^{\lceil n/k \rceil} - 1]$ and $r_k \in [0, q^{n-(k-1)\lceil n/k \rceil} - 1]$. For $i \in [1, k-1]$, let $f_i \in F_{\lceil n/k \rceil}$ be regular of weight $q^{\lceil n/k \rceil + 1} + r_i$, and let $f_k \in F_{n-(k-1)\lceil n/k \rceil}$ be regular of weight $q^{n-(k-1)\lceil n/k \rceil + 1} + r_k$. Set $f := \prod_{i=1}^k f_i[(i-1)\lceil n/k \rceil]$.

Using the above proposition, it is easy to see that f has weight $r + \sum_{i=1}^k q^{n-(i-1)\lceil n/k \rceil + 1}$ and is regular outside of $S_{0,(k-1)\lceil n/k \rceil - 1}^{(n)}$. It remains to bound the pole orders at these places.

For $i \in [1, k-1]$, f_i has weight less than $q^{\lceil n/k \rceil + 1} + q^{\lceil n/k \rceil}$. Thus by the above proposition, the pole divisor of $f_i[(i-1)\lceil n/k \rceil]$ in $F_{i\lceil n/k \rceil}$ satisfies

$$(f_i[(i-1)\lceil n/k \rceil])_\infty^{(i\lceil n/k \rceil)} - (q^{\lceil n/k \rceil + 1} + r_i)(P_\infty^{(i\lceil n/k \rceil)}) \leq (q+1)q^{\lceil n/k \rceil} \sum_{t=0}^{(i-1)\lceil n/k \rceil - 1} q^{\max\{0, 2t - i\lceil n/k \rceil\}} \left(S_t^{(i\lceil n/k \rceil)}\right).$$

Here we use $\left(S_t^{(i\lceil n/k \rceil)}\right)$ as a shorthand for the divisor $\sum_{P \in S_t^{(i\lceil n/k \rceil)}}(P)$. Let G_i denote the divisor on the right-hand side. By direct computation or by the same trick used in the proof of the proposition, we have $\deg(G_i) = (q+1)(q^{i\lceil n/k \rceil} - q^{\lceil n/k \rceil})$.

Next, f_k has weight less than $q^{n-(k-1)\lceil n/k \rceil + 1} + q^{n-(k-1)\lceil n/k \rceil}$. Hence again

$$(f_k[(k-1)\lceil n/k \rceil])_\infty^{(n)} - (q^{n-(k-1)\lceil n/k \rceil + 1} + r_k)(P_\infty^{(n)}) \leq (q+1)q^{n-(k-1)\lceil n/k \rceil} \sum_{t=0}^{(k-1)\lceil n/k \rceil - 1} q^{\max\{0, 2t-n\}} \left(S_t^{(i\lceil n/k \rceil)}\right).$$

Let G_k denote the divisor on the right-hand side. As above, we have $\deg(G_k) = (q+1)(q^n - q^{n-(k-1)\lceil n/k \rceil})$.

Thus

$$(f)_\infty^{(n)} - v_{P_\infty^{(n)}}(f)(P_\infty^{(n)}) \leq \sum_{i=1}^{k-1} \mathrm{Con}_{F_{i\lceil n/k \rceil}}^{F_n} G_i + G_k.$$

Define

$$G = \left(\sum_{i=1}^{k} q^{n-(i-1)\lceil n/k \rceil + 1}\right)(P_\infty^{(n)}) + \sum_{i=1}^{k-1} \mathrm{Con}_{F_{i\lceil n/k \rceil}}^{F_n} G_i + G_k.$$

Then the above remarks show that

$$f \in \mathcal{L}(G + r(P_\infty^{(n)})) \setminus \mathcal{L}(G + (r-1)(P_\infty^{(n)})),$$

and

$$\begin{aligned} \deg(G) &= \sum_{i=1}^{k} q^{n-(i-1)\lceil n/k \rceil + 1} + \sum_{i=1}^{k-1} (q+1)(q^n - q^{n-(i-1)\lceil n/k \rceil}) + (q+1)(q^n - q^{n-(k-1)\lceil n/k \rceil}) \\ &\geq k(q+1)q^n - q^n \\ &= q^n(kq + k - 1). \end{aligned}$$

3.2.3 Code Sequences Beating the Gilbert-Varshamov Bound

For $r \in [0, q^n - 1]$, let $f_r \in \mathcal{L}(G + r(P_\infty^{(n)})) \setminus \mathcal{L}(G + (r-1)(P_\infty^{(n)}))$ be the function f constructed in the previous section. We extend this definition to $r \geq q^n$ by setting

$$f_{sq^n + t} = x_0^s f_t;$$

then we still have $f_r \in \mathcal{L}(G + r(P_\infty^{(n)})) \setminus \mathcal{L}(G + (r-1)(P_\infty^{(n)}))$ because x_0 is regular of weight q^n.

Define an $\mathbb{F}_{q^2}$-linear map $\psi : \mathbb{F}_{q^2}^{\mathbb{N}^0} \to \bigcup_r \mathcal{L}(G + rP_\infty^{(n)})$ by sending the r-th basis vector to f_r (we zero-index the basis vectors). Because the f_r have distinct weights, ψ is injective.

Each f_r is regular at all of the code places used in Shum et. al.,[3] i.e., the (completely split) places lying above the zero of $x_0 - \alpha$ for some $\alpha \in \mathbb{F}_{q^2} \setminus \Omega$. Hence we can speak of the evaluation map $\text{ev} : \bigcup_r \mathcal{L}(G + rP_\infty^{(n)}) \to \mathbb{F}_{q^2}^{q^n(q^2-q)}$, mapping a function to its values at the code places. By a property of Riemann-Roch spaces and the bound on $\deg(G)$ in the previous section, ev is injective when restricted to $\mathcal{L}(G + q^n(q^2 - q - kq - k + 1))P_\infty^{(n)})$.

Let $K \in [1, q^n(q^2 - q - kq - k + 1) + 1]$. Then we can consider $(\text{ev} \circ \psi)|_{\mathbb{F}_{q^2}^K}$ to be the evaluation map for an $[N, K]$ code, with $N = q^n(q^2 - q)$. The codespace is $\text{ev}(\mathcal{L}(G + (K-1)P_\infty^{(n)}))$. By a property of Riemann-Roch spaces, this code has minimum distance $D \geq N - K - \deg(G) \geq N - K - q^n(kq + k - 1)$. Thus letting R be the rate and δ be the relative distance, we have

$$R + \delta \geq 1 - \frac{q^n(kq + k - 1)}{q^n(q^2 - q)} = 1 - \frac{kq + k - 1}{q^2 - q}.$$

To summarize, for any $n \geq 0$, we can define codes of length $q^n(q^2 - q)$ on or above the line

$$R + \delta = 1 - \frac{kq + k - 1}{q^2 - q},$$

with many possible choices of rate (specifically, any rate $K/(q^n(q^2 - q))$ with $K \in [1, q^n(q^2 - q - kq - k + 1) + 1]$). Depending on k and q, this line may exceed the Gilbert-Varshamov bound. For instance, when $k = 1$, it does so for $q \geq 7$, as noted in Shum et. al.;[3] when $k = 2$, it does so for $q \geq 19$; when $k = 3$, it does so for $q \geq 32$; and when $k = 4$, it does so for $q \geq 47$.

3.2.4 Fast Encoding Algorithm

Fix $k \geq 1$. Given $v \in \mathbb{F}_{q^2}^{q^n(q^2-q-kq-k+1)+1}$, we wish to output $\text{ev}(\psi(v))$. We begin by considering how to do so for $v \in \mathbb{F}_{q^2}^{q^n}$.

First, compute the evaluations of some $g_0, \dots, g_{q^{\lceil n/k \rceil}-1} \in F_{\lceil n/k \rceil}$ at the code places of $F_{\lceil n/k \rceil}$, where each g_s is regular of weight $q^{\lceil n/k \rceil + 1} + s$. Using the algorithm in Shum et. al.,[3] we can do so using $< (\lceil n/k \rceil)^3 q^{3\lceil n/k \rceil + 4}$ multiplications and divisions in $\mathbb{F}_{q^2}$.

Next, define the procedure Construct-Level-i as follows. Input $i \in [0, k-1]$ and a vector $w \in \mathbb{F}_{q^2}^{q^{i\lceil n/k \rceil}}$. We will output the evaluations of $\sum_{j=0}^{q^{i\lceil n/k \rceil}-1} w_j f_{jq^{n-i\lceil n/k \rceil}}$ at all code places in $F_{i\lceil n/k \rceil}$, where w_j denotes the coordinate of the j-th basis vector in w (we zero-index the basis vectors). Note that each $f_{jq^{n-i\lceil n/k \rceil}}$ indeed lies in $F_{i\lceil n/k \rceil}$, so this makes sense.

If $i = 0$, output 1. Else do the following. Uniquely write $w = \sum_{l=0}^{q^{\lceil n/k \rceil}-1} w^{(l)}_{+lq^{(i-1)\lceil n/k \rceil}}$, where each $w^{(l)} \in \mathbb{F}_{q^2}^{q^{(i-1)\lceil n/k \rceil}}$, and where the subscript $(+lq^{(i-1)\lceil n/k \rceil})$ means to shift all of the basis vectors' indices in $w^{(l)}$ up by $lq^{(i-1)\lceil n/k \rceil}$. Call Construct-Level-$i$ on $(i-1, w^{(l)})$ for each l and store the results. For a code place Q of $F_{(i-1)\lceil n/k \rceil}$, we will use $Q(w^{(l)})$ to denote the evaluation at Q.

Now for each code place P, do the following. Let P' denote the code place obtained by restricting P to $F_{(i-1)\lceil n/k \rceil}$. Let P'' denote the code place of $F_{\lceil n/k \rceil}$ which maps x_0 to $P(x_{(i-1)\lceil n/k \rceil})$, x_1 to

$P(x_{(i-1)\lceil n/k\rceil+1})$, etc. It is easy to check that P'' is indeed a code place. Now output

$$\sum_{l=0}^{q^{\lceil n/k\rceil}-1} P'(w^{(l)})P''(g_l) \tag{5}$$

as the evaluation at P, using the stored values of $P'(w^{(l)})$ and $P''(g_l)$.

Finally, extend the definition of Construct-Level-i to $i = k$, $v \in \mathbb{F}_{q^2}^{q^n}$ in essentially the same way, noting that we can use the evaluations computed for $F_{\lceil n/k\rceil}$ to get evaluations for $F_{n-(k-1)\lceil n/k\rceil}$.

It is easy to see that Construct-Level-i is correct, by induction on i. For the runtime, we claim inductively that on input (i, w) with $i \in [1, k-1]$, Construct-Level-i requires $\le iq^{(i+1)\lceil n/k\rceil}(q^2-q)$ multiplications and $\le iq^{(i+1)\lceil n/k\rceil}(q^2-q)$ additions over $\mathbb{F}_{q^2}$. (Note that Construct-Level-i does no other types of operations over $\mathbb{F}_{q^2}$, and all other steps take time bounded by a constant times the number of field operations.)

The base case $i = 0$ is trivial. Now assume the claim is true for i, and let us prove it for $i+1$. The $q^{\lceil n/k\rceil}$ recursive calls to Construct-Level-i on $(i-1, w^{(l)})$ require $\le iq^{(i+2)\lceil n/k\rceil}(q^2-q)$ multiplications over $\mathbb{F}_{q^2}$, and likewise for additions. For each code place P, computing the sum (5) requires $q^{\lceil n/k\rceil}$ multiplications and $q^{\lceil n/k\rceil}-1$ additions. There are $q^{(i+1)\lceil n/k\rceil}(q^2-q)$ code places, so computing the sum for all code places requires $q^{(i+2)\lceil n/k\rceil}(q^2-q)$ multiplications and at most that number of additions. Hence in total, we require $\le (i+1)q^{(i+2)\lceil n/k\rceil}(q^2-q)$ each of multiplications and additions, proving the claim.

Modifying the above argument slightly in the case $i = k$, we find that running Construct-Level-i on (k, v), $v \in \mathbb{F}_{q^2}^{q^n}$, takes $\le kq^{n+\lceil n/k\rceil}(q^2-q)$ each of multiplications and additions over $\mathbb{F}_{q^2}$.

Finally, for general $v \in \mathbb{F}_{q^2}^{q^n(q^2-q-kq-k+1)+1}$, we can compute $\mathrm{ev}(\psi(v))$ with at most $q^2-(k+1)q$ calls to Construct-Level-i on $(k, (-))$, using the fact that $f_{sq^n+t} = x_0^s f_t$ for $s \ge 0$, $t \in [0, q^n-1]$. This requires $\le kq^{n+\lceil n/k\rceil}(q^2-q)(q^2-(k+1)q) \le kq^{n+\lceil n/k\rceil+4}$ each of multiplications and additions over $\mathbb{F}_{q^2}$. Thus in total (taking into account the runtime of Shum's algorithm as well), we can encode $v \in \mathbb{F}_{q^2}^{q^n(q^2-q-kq-k+1)+1}$ using at most

$$(\lceil n/k\rceil)^3 q^{3\lceil n/k\rceil+4} + kq^{n+\lceil n/k\rceil+4}$$

multiplications and divisions over $\mathbb{F}_{q^2}$, plus a comparable number of additions over $\mathbb{F}_{q^2}$ and a comparable number of other machine operations.

3.2.5 Discussion

We have established a tradeoff between runtime and code quality, controlled by the parameter k. As k increases, the runtime approaches an asymptote while the code quality decreases linearly, so perhaps the most useful value is $k = 2$. In this case we get codes beating the Gilbert-Varshamov bound when $q \ge 19$, with encoding requiring

$$\begin{aligned}
&\le (\lceil n/2\rceil)^3 q^{3\lceil n/2\rceil+4} + 2q^{n+\lceil n/2\rceil+4} \\
&\le \left((n/2+1)^3+2\right) q^{3n/2+7} \\
&= O(n^3 q^{3n/2+7})
\end{aligned}$$

multiplications and divisions over $\mathbb{F}_{q^2}$. This is about a square-root improvement over Shum's algorithm.

I believe that the codes we construct constitute subcodes of the k-fold tensor product of copies of the Garcia-Stichtenoth code for $F_{\lceil n/k \rceil}$. We get much better code parameters from the Riemann-Roch theorem than we do from this tensor product code formulation.

Better algorithms for computing actually regular functions translate to better runtimes in this algorithm for most k. In particular, to eliminate the extra $q^{\lceil n/k \rceil + O(1)}$ above linear time, we would need a way to construct regular functions that are somehow nicer than those given by Shum's algorithm, so that we could compute the encoding map for regular functions in nearly linear time (currently we only get quadratic time).

3.3 Mostly Regular Functions when $q = 2$ in time $O(n)$

In this section, we aim to construct elements in the n-th level of the Garcia-Stichtenoth tower over $q = 2$ of all weights in the range $[1, q^n - 1]$, which are regular outside of $S_{0,n-2}^{(n)}$, with poles at $S_i^{(n)}$ ($i \in [1, n-2]$) bounded by those of $x_n x_{n-1} \dots x_{i+2}$, and with poles at $S_0^{(n)}$ bounded by those of $x_n x_{n-1} \dots x_1$.

For a set $S \subset [1, n]$, we will use $\mathrm{elem}(S)$ to denote the element that we construct with weight $\sum_{j \in S} 2^{n-j}$. In other words, $\mathrm{elem}(S)$ will be the "corrected" form of $\prod_{j \in S} x_j$. For $i \in \mathbb{Z}$, we let $S[i] := \{i + j \mid j \in S\}$ denote the *shift* of S by i. We use the same notation for elements of F_n, so that, e.g., $x_j[i] = x_{i+j}$; by our theory of shifting, this is well-defined.

Now define $\mathrm{elem}(S)$ inductively by

$$\mathrm{elem}(S) = \begin{cases} 1 & \text{if } S = \emptyset \\ x_1 & \text{if } S = \{1\} \\ \mathrm{elem}(S[-1])[1] + \sum_{T \subsetneq S} \mathrm{elem}(T[-1])[1] \frac{x_1}{x_0} & \text{if } S \neq \emptyset, \{1\} \text{ and } 1 \notin S \\ \mathrm{elem}((S \setminus \{1\})[-1])[1] x_1 + \sum_{T \subsetneq (S \setminus \{1\})} \mathrm{elem}(T[-1])[1] \frac{x_1}{x_0} & \text{if } S \neq \emptyset, \{1\} \text{ and } 1 \in S. \end{cases}$$

The motivation for this element is as follows. We know that $u_j := x_j + \frac{x_{j-1}}{x_{j-2}} + \dots + \frac{x_1}{x_0}$ is regular outside $P_\infty^{(n)}$ and $S_{0,j-2}^{(n)}$, with poles at $S_{1,j-2}^{(n)}$ bounded by those of x_j, poles at $S_0^{(n)}$ bounded by those of x_1, and the weight of x_j. Hence

$$x_1^{\chi(1 \in S)} \prod_{j \in (S \setminus \{1\})} u_j$$

has the desired pole orders except at $S_0^{(n)}$. (Here we define $\chi(1 \in S)$ to be 1 if $1 \in S$ and 0 otherwise.) Now writing $u_j = v_j + \frac{x_1}{x_0}$, the above element equals

$$x_1^{\chi(1 \in S)} \prod_{j \in (S \setminus \{1\})} \left(v_j + \frac{x_1}{x_0} \right),$$

which we can expand as

$$\sum_{T\subset(S\setminus\{1\})}\left(\prod_{j\in T}v_j\right)\frac{x_1^{\chi(1\in S)+\sharp(S\setminus T)}}{x_0^{\sharp(S\setminus T)}}.$$

Now we see where the bad poles at $S_0^{(n)}$ come from: high powers of x_1. It would be convenient if we could just rewrite those high powers down to 1, by analogy with the $n=2,3$ cases for general q. But this might introduce bad poles at the other places unless we have some kind of nice inductive guarantee on the coefficient

$$\left(\prod_{j\in T}v_j\right)\frac{1}{x_0^{\sharp(S\setminus T)}}.$$

By analogy with the $n=3$ case for general q, such a nice inductive guarantee would result if we replaced each such coefficient with the "good" element of the same weight, except coming from one level lower and shifted upwards, and simultaneously reduced the exponent of x_0 down to $<q$. In other words, we wish that the above sum equalled

$$\mathrm{elem}((S\setminus\{1\})[-1])[1]x_1^{\chi(1\in S)}+\sum_{T\subsetneq(S\setminus\{1\})}\mathrm{elem}(T[-1])[1]\frac{x_1^{\chi(1\in S)+\sharp(S\setminus T)}}{x_0},$$

where elem denotes our recursively defined "good" element function. It is not *a priori* clear that this element should still be regular at $S_1^{(n)}$. But if we hope that it is, then after replacing each exponent of x_1 with 1, we should get an element of the desired form, including good poles at $S_0^{(n)}$. In this way, we come to the definition for $\mathrm{elem}(S)$ given above.

Returning to the proof, for $S\subset[1,n]$, we claim the following:

- $\mathrm{elem}(S)$ is regular at all places outside of $P_\infty^{(n)}$ and $S_{0,n+1}^{(n)}$.
- For $i\in[0,n+1]$, the pole orders of $\mathrm{elem}(S)$ at $S_i^{(n)}$ are bounded by those of $\prod_{j\in S\cap[i+2,n]}x_j$. In particular, $\mathrm{elem}(S)$ is regular at $S_{n-1,n+1}^{(n)}$.
- $\mathrm{elem}(S)$ has weight equal to that of $\prod_{j\in S}x_j$.

All of these claims follow easily by induction and our theory of shifting, except for the claim that the pole orders of $\mathrm{elem}(S)$ at $S_1^{(n)}$ are bounded by those of $\prod_{j\in S\cap[3,n]}x_j$.

It remains to prove the bound on the pole orders at $S_1^{(n)}$, under the assumption that all of the above claims hold inductively. In the base case $S=\emptyset$, this is obvious. Now assume $S\neq\emptyset$. We split into several cases.

Case $1,2\notin S$: By assumption, $S[-1]$ and all $T[-1]$, $\emptyset\neq T\subsetneq S$, fall into the third case in the definition of $\mathrm{elem}(-)$. Additionally, the definition of $\mathrm{elem}(\emptyset)$ agrees with that given by the third

case. Applying that definition to all elem(−)'s appearing in the definition of elem(S), we get

$$\begin{aligned}
\operatorname{elem}(S) &= \operatorname{elem}(S[-1])[1] + \sum_{T \subsetneq S} \operatorname{elem}(T[-1])[1]\frac{x_1}{x_0} \\
&= \left(\operatorname{elem}(S[-2])[1] + \sum_{T' \subsetneq S[-1]} \operatorname{elem}(T'[-1])[1]\frac{x_1}{x_0}\right)[1] \\
&\quad + \sum_{T \subsetneq S} \left(\operatorname{elem}(T[-2])[1] + \sum_{U' \subsetneq T[-1]} \operatorname{elem}(U'[-1])[1]\frac{x_1}{x_0}\right)[1]\frac{x_1}{x_0} \\
&= \operatorname{elem}(S[-2])[2] + \sum_{T \subsetneq S} \operatorname{elem}(T[-2])[2]\frac{x_2}{x_1} \\
&\quad + \sum_{T \subsetneq S} \operatorname{elem}(T[-2])[2]\frac{x_1}{x_0} + \sum_{T \subsetneq S}\sum_{U \subsetneq T} \operatorname{elem}(U[-2])[2]\frac{x_2}{x_0} \\
&= \operatorname{elem}(S[-2])[2] + \sum_{T \subsetneq S} \operatorname{elem}(T[-2])[2]\left(\frac{x_2}{x_1} + \frac{x_1}{x_0}\right),
\end{aligned}$$

noting that for each $U \subsetneq S$, we have

$$\sharp\{T \mid U \subsetneq T \subsetneq S\} = \sharp\{A \subset (S \setminus U) \mid A \neq \emptyset, S \setminus U\} = 2^{\sharp(S\setminus U)} - 2 \equiv 0 \pmod 2,$$

hence the double-sum is killed by the characteristic. Now the claim follows from our inductive bounds on the weights of elem($S[-2]$) and the elem($T[-2]$) (which become bounds on the pole order at $S_1^{(n)}$ by our theory of shifting), together with the fact that $\frac{x_2}{x_1} + \frac{x_1}{x_0}$ is regular at $S_1^{(n)}$ (see Theorem 28(ii)[5]).

Case $1 \notin S, 2 \in S$**:** Write $S = R \sqcup \{2\}$. Note that R may equal $\emptyset$. By definition,

$$\begin{aligned}
\operatorname{elem}(S) = \operatorname{elem}(R \sqcup \{2\}) &= \operatorname{elem}(R[-1] \sqcup \{1\})[1] + \operatorname{elem}(R[-1])[1]\frac{x_1}{x_0} \\
&\quad + \sum_{T \subsetneq R} \operatorname{elem}(T[-1] \sqcup \{1\})[1]\frac{x_1}{x_0} + \sum_{T \subsetneq R} \operatorname{elem}(T[-1])[1]\frac{x_1}{x_0}.
\end{aligned}$$

By assumption, all elem(−)'s appearing in the first and third terms fall into the fourth case in the definition of elem, and all elem(−)'s appearing in the second and fourth terms fall into the third case. (This is true even when elem($\emptyset$) or elem($\{1\}$) appears, since elem($\emptyset$) agrees with the third case of the definition of elem, and elem($\{1\}$) agrees with the fourth case of the definition of

elem.) Now we apply the appropriate definitions to all elem(−)'s appearing above:

$$\begin{aligned}
\operatorname{elem}(R \sqcup \{2\}) &= \left(\operatorname{elem}(R[-2])[1]x_1 + \sum_{T' \subsetneq R[-1]} \operatorname{elem}(T'[-1])[1]\frac{x_1}{x_0}\right)[1] \\
&\quad + \left(\operatorname{elem}(R[-2])[1] + \sum_{T' \subsetneq R[-1]} \operatorname{elem}(T'[-1])[1]\frac{x_1}{x_0}\right)[1]\frac{x_1}{x_0} \\
&\quad + \sum_{T \subsetneq R}\left(\operatorname{elem}(T[-2])[1]x_1 + \sum_{U' \subsetneq T[-1]} \operatorname{elem}(U'[-1])[1]\frac{x_1}{x_0}\right)[1]\frac{x_1}{x_0} \\
&\quad + \sum_{T \subsetneq R}\left(\operatorname{elem}(T[-2])[1] + \sum_{U' \subsetneq T[-1]} \operatorname{elem}(U'[-1])[1]\frac{x_1}{x_0}\right)[1]\frac{x_1}{x_0} \\
&= \operatorname{elem}(R[-2])[2]x_2 + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2]\frac{x_2}{x_1} \\
&\quad + \operatorname{elem}(R[-2])[2]\frac{x_1}{x_0} + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2]\frac{x_2}{x_0} \\
&\quad + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2]\frac{x_2x_1}{x_0} + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2]\frac{x_1}{x_0} \\
&= \operatorname{elem}(R[-2])[2]\left(x_2 + \frac{x_1}{x_0}\right) + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2]\left(\left(\frac{x_2}{x_1} + \frac{x_1}{x_0}\right) + (x_1+1)\frac{x_2}{x_0}\right).
\end{aligned}$$

Here we have used the characteristic to kill the double-sums. Now again the claim follows from our inductive bounds on the weights of elem($R[-2]$) and the elem($T[-2]$), together with the fact that $x_2 + \frac{x_1}{x_0}$, $\frac{x_2}{x_1} + \frac{x_1}{x_0}$, and $(x_1+1)\frac{x_2}{x_0}$ are regular at $S_1^{(n)}$ (see Theorem 28(ii)[5]).

Case $1 \in S$: Let $S = R \sqcup \{1\}$. By definition and the characteristic,

$$\operatorname{elem}(S) = \operatorname{elem}(R) + \operatorname{elem}(R[-1])[1](x_1+1).$$

(Note that this equation holds even if $R = \emptyset$.) By the inductive hypothesis, elem$(R[-1])(x_0+1)$ is regular at $S_0^{(n-1)}$, so elem$(R[-1])[1](x_1+1)$ is regular at $S_1^{(n)}$ by our theory of shifting. Hence the poles of elem(S) at $S_1^{(n)}$ are bounded by those of elem(R), so we have reduced to the cases above.

A recursive algorithm for elem(S) using the definition would be very slow — $2^{O(n^2)}$ using plain recursion, $O(n2^{n-1})$ using our standard tricks to compute it for all S with the same maximum simultaneously. The fact that expanding the definition recursively turns elem(S) from a sum of $O(2^{n-1})$ terms into a sum of $O(2^{n-2})$ terms in one step suggests that we should be able to get an $O(n)$ time algorithm by somehow repeatedly expanding it recursively and cancelling most terms. We now show that we can indeed get an $O(n)$ time algorithm, by using last section's calculations to get a much shorter recursive form for elem(S).

Let $S \neq \emptyset$. We first derive short recursive forms for elem(S), which vary depending on whether 1 and 2 are in S.

Case $1, 2 \notin S$: From the last section, we have

$$\operatorname{elem}(S) = \operatorname{elem}(S[-2])[2] + \sum_{T \subsetneq S} \operatorname{elem}(T[-2])[2] \left(\frac{x_2}{x_1} + \frac{x_1}{x_0} \right). \tag{6}$$

Meanwhile, by definition,

$$\operatorname{elem}(S[-1])[1] = \operatorname{elem}(S[-2])[2] + \sum_{T \subsetneq S} \operatorname{elem}(T[-2])[2] \frac{x_2}{x_1}. \tag{7}$$

Solving for the sum in Equation (7) and substituting into Equation (6), we get

$$\operatorname{elem}(S) = \operatorname{elem}(S[-2])[2] + (\operatorname{elem}(S[-1])[1] + \operatorname{elem}(S[-2])[2]) \left(1 + \frac{x_1^2}{x_0 x_2} \right).$$

Case $1 \notin S, 2 \in S$**:** Write $S = R \sqcup \{2\}$. From the last section, we have

$$\operatorname{elem}(S) = \operatorname{elem}(R[-2])[2] \left(x_2 + \frac{x_1}{x_0} \right) + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2] \left(\left(\frac{x_2}{x_1} + \frac{x_1}{x_0} \right) + (x_1 + 1) \frac{x_2}{x_0} \right). \tag{8}$$

Meanwhile, by definition,

$$\operatorname{elem}(R[-1])[1] = \operatorname{elem}(R[-2])[2] + \sum_{T \subsetneq R} \operatorname{elem}(T[-2])[2] \frac{x_2}{x_1}. \tag{9}$$

Solving for the sum in Equation (9) and substituting into Equation (8), we get

$$\begin{aligned} \operatorname{elem}(S) = \operatorname{elem}(R[-2])[2] &\left(x_2 + \frac{x_1}{x_0} \right) \\ &+ (\operatorname{elem}(R[-1])[1] + \operatorname{elem}(R[-2])[2]) \left(1 + \frac{x_1^2}{x_0 x_2} + (x_1 + 1) \frac{x_1}{x_0} \right). \end{aligned}$$

Case $1 \in S$: As noted in the last section,

$$\operatorname{elem}(S) = \operatorname{elem}(R) + \operatorname{elem}(R[-1])[1](x_1 + 1).$$

Now we can compute $\operatorname{elem}(S)$ in $O(n)$ time (where $n = \max S$) with the above formulas using a dynamic programming algorithm: first compute $\operatorname{elem}(S[-(n-1)] \cap \{1\})[n-1]$, then $\operatorname{elem}(S[-(n-2)] \cap \{2\})[n-2]$ and $\operatorname{elem}(S[-(n-2)] \cap [1,2])[n-2]$, etc., computing each $\operatorname{elem}(S[-(n-i)] \cap [2,i])[n-i]$ and $\operatorname{elem}(S[-(n-i)] \cap [1,i])[n-i]$ in order until we have computed $\operatorname{elem}(S) = \operatorname{elem}(S \cap [1,n])$.

3.4 Iterated Corrections

3.4.1 Corrections

The goal of this section is to prove the following theorem:

Theorem 3.4.1. *Given $t \in [1, n-1]$. Given a monomial of the form $x_{t+1}^{e_{t+1}} x_t^{e_t} x_{t-1}^{e_{t-1}}$ with all $e_i \in [-(q-1), q-1]$. If $e_{t+1} - e_{t-1} \geq q$, additionally assume $e_t \leq 0$. Then there exists a sum*

$$\sum_\alpha c_\alpha \frac{x_{t+1}^{f_\alpha}}{x_{t-1}^{h_\alpha}} x_t^{g_\alpha},$$

where all $c_\alpha \in \mathbb{F}_p$ (the prime field of $\mathbb{F}_{q^2}$), $f_\alpha, h_\alpha \in [0, q-1]$, $g_\alpha \in [1, q-1]$, and no $\frac{x_{t+1}^{f_\alpha}}{x_{t-1}^{h_\alpha}} x_t^{g_\alpha}$ equals $x_{t+1}^{e_{t+1}} x_t^{e_t} x_{t-1}^{e_{t-1}}$, such that

$$x_{t+1}^{e_{t+1}} x_t^{e_t} x_{t-1}^{e_{t-1}} + \sum_\alpha c_\alpha \frac{x_{t+1}^{f_\alpha}}{x_{t-1}^{h_\alpha}} x_t^{g_\alpha}$$

is regular at $S_t^{(n)}$. The c_α, f_α, g_α, and h_α can be computed in time $O(q^2)$. If $e_{t+1} - e_{t-1} < q$, each term in the sum has the same pole order at $S_t^{(n)}$ as the original monomial; else each term has the same pole order as $x_{t+1}^{e_{t+1}} x_{t-1}^{q+e_{t-1}}$. If $e_{t-1} \neq 0$, then the sum is actually $O(x_{t-1})$ at $S_t^{(n)}$, not just regular.

For ease of notation, we will prove the theorem for $t = 1$; the general case follows by shifting. We cover several cases.

Suppose $e_2 - e_0 \geq q$. For any $P \in S_1^{(n)}$, letting α_0 correspond to P under the notation in Theorems 2(i), 28(i), or 36(i) in Shum's thesis,[5] we have

$$x_1 = -\alpha_0 + x_0^q + O(x_0^{2q-1})$$

(Theorems 2(ii), 28(ii), and 36(ii)[5]). Since $\alpha_0^{q-1} + 1 = 0$,

$$x_1^{q-1} + 1 = x_0^q + O(x_0^{2q-1}).$$

Then recalling that $e_1 \leq 0$, we have

$$x_2^{e_2} x_1^{e_1} x_0^{e_0} + x_2^{e_2} x_1^{e_1+q-1} x_0^{e_0} = x_2^{e_2} x_1^{e_1} x_0^{e_0} (1 + x_1^{q-1}) = x_2^{e_2} x_1^{e_1} x_0^{e_0+q} + O(x_0)$$

at P. Thus to fix $x_2^{e_2} x_1^{e_1} x_0^{e_0}$, we can add $x_2^{e_2} x_1^{e_1+q-1} x_0^{e_0}$, then add the correction terms for $x_2^{e_2} x_1^{e_1} x_0^{e_0+q}$ (note that $e_0 + q \in [1, q-1]$).

For the rest of this section, we consider the case $e_2 - e_0 \leq q - 1$. When $e_0 = 0$, we can use

$$\sum_{i=1}^{e_2} \binom{e_2}{i} (-1)^{\sigma(e_1+2i)} x_2^{e_2-i} \frac{x_1^{\rho(e_1+2i)}}{x_0^i}$$

as usual, where $\rho : \mathbb{Z} \to [1, q-1]$ is defined by $\rho(m) \equiv m \pmod{q-1}$ and $\sigma : \mathbb{Z} \to \mathbb{Z}$ is defined by $\sigma(m) = (m - \rho(m))/(q-1)$.

Lemma 3.4.2. *Let $b > 0$ and $a \geq 0$. Then at any $P \in S_1^{(n)}$,*

$$(x_2 x_0)^b x_2^a - g_{a,b} = O(x_0^{q-a}),$$

where $g_{a,b}$ is defined by

$$g_{0,b} = (-1)^{\sigma(2b)} x_1^{\rho(2b)}$$
$$g_{a,b} = x_2 g_{a-1,b} + \binom{a+b-1}{b-1} (-1)^b \sum_{i=0}^{a} \binom{a}{i} (-1)^{\sigma(2b+2i)} x_2^{a-i} \frac{x_1^{\rho(2b+2i)}}{x_0^i}.$$

Proof. Let $P \in S_1^{(n)}$ correspond to the solution α_0, α_1, with notation as in Theorems 2(i), 28(i), or 36(i) in Shum's thesis.[5] We will first prove the stronger statement that at P,

$$(x_2x_0)^b x_2^a = f_{a,b} + \sum_{i=1}^{b} \binom{a+b}{b-i} (-1)^{b-i} x_0^i \alpha_1^{a+i} \alpha_0^{2(b-i)} + O(x_0^{q-a}),$$

where $f_{a,b}$ is defined by

$$\begin{aligned} f_{0,b} &= (-1)^b \alpha_0^{2b} \\ f_{a,b} &= x_2 f_{a-1,b} + \binom{a+b-1}{b-1} (-1)^b \alpha_0^{2b} \alpha_1^a. \end{aligned}$$

Induct on a. In the base case $a = 0$, this is just the binomial theorem applied to

$$(x_2x_0)^b = (-\alpha_0^2 + \alpha_1 x_0 + O(x_0^q))^b$$

(Theorems 2(ii), 28(ii), and 36(ii),[5] noting that $\alpha_0 = 1$ when $q = 2$). Now to prove the statement for a, multiply the equation from the $a-1$ case by x_2:

$$\begin{aligned} (x_2x_0)^b x_2^a &= x_2 f_{a-1,b} + \sum_{i=1}^{b} \binom{a+b-1}{b-i} (-1)^{b-i} (x_2x_0) x_0^{i-1} \alpha_1^{a-1+i} \alpha_0^{2(b-i)} + O(x_0^{q-a}) \\ &= x_2 f_{a-1,b} + \sum_{i=1}^{b} \binom{a+b-1}{b-i} (-1)^{b-i} \left(-\alpha_0^2 + \alpha_1 x_0 + O(x_0^q)\right) x_0^{i-1} \alpha_1^{a-1+i} \alpha_0^{2(b-i)} + O(x_0^{q-a}) \\ &= x_2 f_{a-1,b} + \binom{a+b-1}{b-1} (-1)^b x_0^i \alpha_1^{a+i} \alpha_0^{2b} \\ &\quad + \sum_{i=1}^{b-1} \left(\binom{a+b-1}{b-i} + \binom{a+b-1}{b-i-1} \right) (-1)^{b-i} x_0^i \alpha_1^{a+i} \alpha_0^{2(b-i)} + x_0^b \alpha_1^{a+b} + O(x_0^{q-a}) \\ &= f_{a,b} + \sum_{i=1}^{b} \binom{a+b}{b-i} (-1)^{b-i} x_0^i \alpha_1^{a+i} \alpha_0^{2(b-i)} + O(x_0^{q-a}). \end{aligned}$$

This completes the induction.

To prove the lemma from the above statement, it suffices to prove that $g_{a,b} = f_{a,b} + O(x_0^{q-a})$ for all a, b. Recall that

$$\begin{aligned} x_1 &= -\alpha_0 + O(x_0^q) \\ x_2 + \frac{\alpha_0^2}{x_0} &= \alpha_1 + O(x_0^{q-1}) \end{aligned}$$

at P (Theorems 2(ii), 28(ii), and 36(ii),[5] noting that $\alpha_0 = 1$ when $q = 2$), hence

$$\begin{aligned} x_1^s &= (-\alpha_0)^s + O(x_0^q) \\ \left(x_2 + \frac{\alpha_0^2}{x_0}\right)^s &= \alpha_1^s + O(x_0^{q-1}) \end{aligned}$$

for all $s \geq 0$. Using the equation $\alpha_0^q + \alpha_0 = 0$, we find that

$$x_1^{\rho(s)} = (-1)^{\sigma(s)} (-\alpha_0)^s + O(x_0^q).$$

Now it is easy to see that $g_{a,b} = f_{a,b} + O(x_0^{q-a})$ by induction on a. □

Lemma 3.4.3. *Let $b > 0$ and $a \geq 0$. Then at any $P \in S_1^{(n)}$,*

$$\left((x_2x_0)^{-1}\right)^b (x_0^{-1})^a - h_{a,b} = O(x_0^{q-a}),$$

where $h_{a,b}$ is defined by

$$h_{0,b} = (-1)^{\sigma(-2b)} x_1^{\rho(-2b)}$$
$$h_{a,b} = x_0^{-1} h_{a-1,b} + \binom{a+b-1}{b-1}(-1)^b \sum_{i=0}^{a} \binom{a}{i} (-1)^{\sigma(-2b-2i))} x_2^i \frac{x_1^{\rho(-2b-2i)}}{x_0^{b-i}}.$$

Proof. This can be proved in the same way as Lemma 3.4.2. Alternatively, we can use the following argument. Let $\phi : F_2 \to F_2$ denote the order-2 isomorphism described on p. 59 of Shum's thesis[5] (see also p. 2237 in Shum et. al.[3]). Recall that ϕ maps $S_1^{(2)}$ to itself. Then by ϕ applied to the conclusion of Lemma 3.4.2, at any $P \in S_1^{(n)}$,

$$\left((x_2x_0)^{-1}\right)^b (x_0^{-1})^a - \phi(g_{a,b}) = O(x_0^{q-a}).$$

Here $\phi(g_{a,b})$ satisfies the relations

$$\phi(g_{0,b}) = (-1)^{\sigma(2b)} x_1^{-\rho(2b)}$$
$$\phi(g_{a,b}) = x_0^{-1}\phi(g_{a-1,b}) + \binom{a+b-1}{b-1}(-1)^b \sum_{i=0}^{a} \binom{a}{i} (-1)^{\sigma(2b+2i)} x_2^i \frac{x_1^{-\rho(2b+2i)}}{x_0^{b-i}}.$$

Now using the fact that $x_1^{q-1} + 1 = O(x_0^q)$ at any $P \in S_1^{(n)}$ together with the identity $\sigma(m) + \sigma(-\rho(m)) \equiv \sigma(-m) \pmod 2$, it is easy to show that $h_{a,b} = \phi(g_{a,b}) + O(x_0^{q-a})$, proving the claim. □

Lemma 3.4.4. *Let $b > 0$ and $a \geq 0$. Then at any $P \in S_1^{(n)}$,*

$$\left((x_2x_0)^{-1}\right)^b x_2^a - j_{a,b} = O(x_0^{q-a}),$$

where $j_{a,b}$ is defined by

$$j_{0,b} = (-1)^{b+\sigma(-2b)} x_1^{\rho(-2b)}$$
$$j_{a,b} = x_2 j_{a-1,b} + \binom{b}{a}(-1)^{a+b} \sum_{i=0}^{a} \binom{a}{i} (-1)^{\sigma(-2b+2i)} x_2^{a-i} \frac{x_1^{\rho(-2b+2i)}}{x_0^i}.$$

Proof. Let $P \in S_1^{(n)}$ correspond to the solution α_0, α_1, with notation as in Theorems 2(i), 28(i), or 36(i) in Shum's thesis.[5] We will first prove the stronger statement that at P,

$$\left((x_2x_0)^{-1}\right)^b x_2^a = k_{a,b} + (-1)^{a+b} \sum_{i=1}^{q-1-a} \binom{b+i-1}{a+i} \alpha_0^{-2b-2i} \alpha_1^{a+i} x_0^i + O(x_0^{q-a}),$$

where $k_{a,b}$ is defined by

$$k_{0,b} = (-1)^b \alpha_0^{-2b}$$
$$k_{a,b} = x_2 k_{a-1,b} + \binom{b}{a}(-1)^{a+b} \alpha_0^{-2b} \alpha_1^a.$$

Induct on a. In the base case $a = 0$, note that by

$$x_2 x_0 = -\alpha_0^2 + \alpha_1 x_0 + O(x_0^q),$$

we have

$$-\alpha_0^{-2} x_2 x_0 = 1 - \alpha_0^{-2}\alpha_1 x_0 + O(x_0^q),$$

hence

$$-\alpha_0^2 (x_2 x_0)^{-1} = \sum_{i=0}^{q-1} \alpha_0^{-2i} \alpha_1^i x_0^i + O(x_0^q).$$

Then using stars and bars, it is easy to see that

$$\left((x_2 x_0)^{-1}\right)^b = (-1)^b \sum_{i=0}^{q-1} \binom{b+i-1}{i} \alpha_0^{-2b-2i} \alpha_1^i x_0^i + O(x_0^q),$$

proving the base case.

Now to prove the statement for a, multiply the equation from the $a-1$ case by x_2:

$$\begin{aligned}
\left((x_2 x_0)^{-1}\right)^b x_2^a &= x_2 k_{a-1,b} + (-1)^{a-1+b} \sum_{i=1}^{q-a} \binom{b+i-1}{a-1+i} \alpha_0^{-2b-2i} \alpha_1^{a-1+i} (-\alpha_0^2 + \alpha_1 x_0 + O(x_0^q)) x_0^{i-1} \\
&\quad + O(x_0^{q-a}) \\
&= x_2 k_{a-1,b} + \binom{b}{a} (-1)^{a+b} \alpha_0^{-2b} \alpha_1^a \\
&\quad + (-1)^{a-1+b} \sum_{i=1}^{q-a-1} \left(\binom{b+i-1}{a-1+i} - \binom{b+i}{a+i} \right) \alpha_0^{-2b-2i} \alpha_1^{a+i} x_0^i + O(x_0^{q-a}) \\
&= k_{a,b} + (-1)^{a+b} \sum_{i=1}^{q-1-a} \binom{b+i-1}{a+i} \alpha_0^{-2b-2i} \alpha_1^{a+i} x_0^i + O(x_0^{q-a}).
\end{aligned}$$

This completes the induction.

As in the proof of Lemma 3.4.2, it is easy to check that $j_{a,b} = k_{a,b} + O(x_0^{q-a})$, proving the lemma. □

Using these lemmas, it is easy to verify the theorem.

3.4.2 F_3

Define a series of mutually recursive functions as follows. Assuming the recursion terminates, it is easy to prove the following inductively:

- $\mathrm{elem}(x_3^{r_3} x_2^{r_2} x_1^{r_1})$ is such that $\pi_0 x_0 \mathrm{elem}(x_3^{r_3} x_2^{r_2} x_1^{r_1})$ is regular, and $\mathrm{elem}(x_3^{r_3} x_2^{r_2} x_1^{r_1}) - x_3^{r_3} x_2^{r_2} x_1^{r_1}$ has negative weight.
- $\mathrm{fix}i(m)$ is such that $m + \mathrm{fix}i(m)$ is regular at $S_i^{(3)}$.

In the below definitions, we assume that all $r_i \geq 0$ unless otherwise specified. When we do a sum across α, we mean to take the corresponding sum from the previous section for the place begin fixed.

$$\text{elem}(x_3^{r_3}x_2^{r_2}x_1^{r_1}) = x_3^{r_3}x_2^{r_2}x_1^{r_1} + \text{fix2}(x_3^{r_3}x_2^{r_2}x_1^{r_1}) + \text{fix1}(x_3^{r_3}x_2^{r_2}x_1^{r_1})$$

$$\text{fix2}(x_3^{r_3}x_2^{r_2}x_1^{r_1}) = \sum_\alpha c_\alpha \left(x_3^{f_\alpha} \frac{x_2^{g_\alpha}}{x_1^{h_\alpha}} + \text{fix1}(x_3^{f_\alpha} \frac{x_2^{g_\alpha}}{x_1^{h_\alpha}}) \right)$$

$$\text{fix1}(x_3^{r_3}x_2^{r_2}x_1^{r_1}) = \sum_\alpha c_\alpha \left(x_3^{r_3}x_2^{f_\alpha} \frac{x_1^{g_\alpha}}{x_0^{h_\alpha}} + \text{fix2}(x_3^{r_3}x_2^{f_\alpha} \frac{x_1^{g_\alpha}}{x_0^{h_\alpha}}) \right)$$

$$\text{fix1}(x_3^{r_3} \frac{x_2^{r_2}}{x_1^{r_1}}), r_3 + r_1 \leq q-1, = x_3^{r_3}x_2^{r_2}x_1^{q-1-r_1} + \text{elem}(x_3^{r_3}x_2^{r_2}x_1^{q-1-r_1})$$

$$\text{fix2}(x_3^{r_3}x_2^{r_2} \frac{x_1^{r_1}}{x_0^{r_0}}), r_1 > 0, r_0 > 0, = \sum_\alpha c_\alpha \left(x_3^{f_\alpha}x_2^{g_\alpha} \frac{1}{x_1^{h_\alpha}x_0^{r_0}} + \text{fix1}(x_3^{f_\alpha}x_2^{g_\alpha} \frac{1}{x_1^{h_\alpha}x_0^{r_0}}) \right)$$

$$\text{fix1}(x_3^{r_3}x_2^{r_2} \frac{1}{x_1^{r_1}x_0^{r_0}}), r_3 + r_1 \leq q-2, r_0 > 0, = x_3^{r_3}x_2^{r_2}x_1^{q-1-r_1} \frac{1}{x_0^{r_0}} + \text{fix1}(x_3^{r_3}x_2^{r_2} \frac{1}{x_1^{r_1}} x_0^{q-r_0})$$

$$\text{fix1}(x_3^{r_3}x_2^{r_2} \frac{1}{x_1^{r_1}} x_0^{r_0}), r_0 > 0, = \sum_\alpha c_\alpha \left(x_3^{r_3}x_2^{f_\alpha} \frac{x_1^{g_\alpha}}{x_0^{h_\alpha}} + \text{fix2}(x_3^{r_3}x_2^{f_\alpha} \frac{x_1^{g_\alpha}}{x_0^{h_\alpha}}) \right)$$

It remains to show that this is total recursive. Note that we never need to call fix2 on a term which has x_1 in the denominator, and we never need to call fix1 on a term which has x_0 in the denominator (except for the second last line, where we only add one term and then a call to fix1 on a term without x_0 in the denominator). Also, in the definition of each fix2, we only make recursive calls to terms with a lower power of x_3, which cannot be increased by calls to fix1. Thus in recursively going back and forth between fix1 and fix2, we monotonically decrease the power of x_3 to 0. After that, one call to fix1 finishes the recursion.

Using this recursion, we can encode Garcia-Stichtenoth codes over F_3 as follows. Given $v \in \mathbb{F}_{q^2}^{q^3}$. Start with $f = g = \sum_{r=1}^{q^3} v_r x_3^{r_3} x_2^{r_2}$, where the r_i are given by $r = r_3 + qr_2 + q^2 r_1$, $r_i \in [0, q-1]$, and where v_r is the coefficient of the r-th basis vector in v. Now loop through terms in order by their power of r_3 and "apply" fix2 and fix1 to them, i.e., add their non-recursive terms to f and add their recursive terms to g, labelled with whether those terms need to be fixed at 1 or 2. Proceed in this way until all un-fixed terms in f are taken care of; then $\pi_0 x_0 g$ will be regular with a power series expansion at ∞ which corresponds to v in a natural way.

The resulting regular functions are nice because they are a sum of the monomials $x_3^{r_3}x_2^{r_2}x_1^{r_1}x_0^{r_0}$ with all $r_i \in [-(q-1), q-1]$. Using multipoint evaluation on each x_i and using the special structure of the code places, we can evaluate in nearly $O(q^5)$ time at $O(q^5)$ code places: once we specify values for $x_0, \ldots, x_2$, there are q possible values for x_3, so we are just evaluation a degree $2(q-1)$ polynomial in x_3 at q values, which can be done in almost linear time; we have to do this $O(q^4)$ times, since there are that many possibilities for $x_0, \ldots, x_2$ (note that we start with $q^2 - q$ values for x_0); etc.

To generalize to higher n, there are a few problems:

- Fixing one term may generate fix requests for multiple other poles, so there is a more complicated recursion than the simple 1-2-1-2 etc. in this case.

- Fixing, e.g., x_4x_3 at $S_2^{(n)}$ requires more work than the above corrections.
- Sometimes two fixes for the same term are independent in the sense that they involve disjoint sets of variables. In this case, if we try doing one fix at a time on a term that needs fixing at both places, we will loop back and forth between fixing those poles. To avoid this, we can correct both of them simulateneously using the term (for a monomial m)

$$(\text{fix } m \text{ at first place})(\text{fix } m \text{ at second place}) - m.$$

3.5 A New, But Slower, Algorithm for Regular Functions

For $i \geq 0$, define ψ_i by $\psi_0 = g_0$, and

$$\psi_i = g_i \psi_{i-1}^q$$

for $i > 0$. We can expand this as $\psi_i = \pi_i \pi_{i-1}^{q-1} \cdots \pi_0^{q^i - q^{i-1} - \cdots - 1}$. Also, we claim by induction on i that $\psi_i - 1$ has a zero of at least the same order as x_i^{q-1} at all places in $S_{i+1,n+1}^{(n)}$. The base case $i = 0$ is trivial. For the inductive step, write

$$\begin{aligned} \psi_i - 1 &= (1 + x_i^{q-1})\left(1 + (\psi_{i-1} - 1)^q\right) - 1 \\ &= (1 + x_i^{q-1})(\psi_{i-1} - 1)^q + x_i^{q-1}. \end{aligned}$$

The claim holds for the first term by the inductive hypothesis, and it is trivial for the second term, so the claim holds for $\psi_i - 1$ by the triangle inequality.

Let $f \in F_n$. Ordinarily when taking the power series of f at $P_\infty^{(n)}$, we would use some uniformizer, such as x_n^{-1}, and expand f as

$$\sum_{i=-m}^{\infty} c_i (x_n^{-1})^i$$

for some $c_i \in \mathbb{F}_{q^2}$. However, these power series have the undesirable property that elements of F_{n-1} may have non-zero c_i when $q \nmid i$, even though such an element can be expanded (in F_{n-1}) as a power series with uniformizer x_{n-1}^{-1}, which has valuation q at $P_\infty^{(n)}$. To get around this issue, we define the *modified power series* of f at $P_\infty^{(n)}$ to be the unique sum

$$\sum_{e_0=0}^{\infty} \sum_{e_1=0}^{q-1} \cdots \sum_{e_n=0}^{q-1} c_{e_0,e_1,\ldots,e_n} x_0^{e_0} x_1^{e_1} \cdots x_n^{e_n},$$

with all $c_{e_0,e_1,\ldots,e_n} \in \mathbb{F}_{q^2}$ and almost all of them 0, such that

$$f = \sum_{e_0=0}^{\infty} \sum_{e_1=0}^{q-1} \cdots \sum_{e_n=0}^{q-1} c_{e_0,e_1,\ldots,e_n} x_0^{e_0} x_1^{e_1} \cdots x_n^{e_n} + O(1)$$

at $P_\infty^{(n)}$.

We now begin the construction. First, we want to see how to "fix" x_n at $S_{\lceil n/2 \rceil, n-1}^{(n)}$. As a first attempt, we could use

$$x_n + \frac{x_{n-1}^{\rho(2)}}{x_{n-2}} + \cdots + \frac{x_{\lceil n/2 \rceil}^{\rho(2)}}{x_{2\lceil n/2 \rceil - n}},$$

since we know by the work of Shum that each $x_n + \frac{x_{n-t}^{\rho(2)}}{x_{n-2t}}$ is regular at $S_{n-t}^{(n)}$. However, $\frac{x_{n-t}^{\rho(2)}}{x_{n-2t}}$ introduces undesired poles at $S_{n-2t,n-t-1}^{(n)}$. To fix this, note from the expansion of ψ_{n-t-1} as a product of π_i's that $\psi_{n-t-1}\frac{x_{n-t}^{\rho(2)}}{x_{n-2t}}$ is regular at $S_{n-2t,n-t-1}^{(n)}$: indeed, one π_i is sufficient to cancel $\frac{1}{x_{n-2t}}$'s pole at $S_i^{(n)}$ for $i \in [n-2t+1, n-t-2]$, the leading π_{n-t-1} cancels $x_{n-t}^{\rho(2)}$'s poles at $S_{n-2t+1,n-t-2}^{(n)}$, and the leading π_{n-t-1} cancels the total pole at $S_{n-t-1}^{(n)}$. Also, from the claim in the first paragraph,

$$(\psi_{n-t-1} - 1)\frac{x_{n-t}^{\rho(2)}}{x_{n-2t}}$$

is regular at $S_{n-t}^{(n)}$.

Hence

$$x_n + \psi_{n-2}\frac{x_{n-1}^{\rho(2)}}{x_{n-2}} + \cdots + \psi_{n-\lceil n/2\rceil-1}\frac{x_{\lceil n/2\rceil}^{\rho(2)}}{x_{2\lceil n/2\rceil-n}}$$

is regular except for poles at $S_{0,\lceil n/2\rceil-1}^{(n)}$, and it easy to see that the poles at those places come only from x_n, hence have order 1.

Thus given $r = e_n + qs$, to get a regular element of weight $q^{n+1} + r$, we start with

$$\left(x_n + \psi_{n-2}\frac{x_{n-1}^{\rho(2)}}{x_{n-2}} + \cdots + \psi_{n-\lceil n/2\rceil-1}\frac{x_{\lceil n/2\rceil}^{\rho(2)}}{x_{2\lceil n/2\rceil-n}}\right)^{e_n} \mathrm{elem}(s, n-1),$$

which (by induction on n) is regular except for poles of order at most $q^{n-2t}-1$ at $S_t^{(n)}$, $0 \le t < n/2$.

However, the weight is bad, since the weight of the correction may be more than that of $x_n^{e_n}\mathrm{elem}(s, n-1)$. We can fix this by pole cancelling using elem's that we have already constructed, as follows. Note that the correction is a sum of terms of the form $x_n^i y$ with $i \in [0, e_n - 1]$, $y \in f_{n-1}$. The modified power series of $x_n^i y$ in F_n equals x_n^i times the modified power series of y in F_{n-1}; in particular, every power of x_n is i. If the highest-weight term in this modified power series has weight at least as big as that of $x_n^{e_n}\mathrm{elem}(s, n-1)$, then we can cancel that term by adding some $\mathbb{F}_{q^2}$-multiple of an almost regular element z with the same weight, which we have already constructed inductively because $i < e_n$. The resulting sum $x_n^i y + z$ may no longer have a modified power series equal to x_n^i time a modified power series coming from F_{n-1}, but we can assume inductively that z's modified power series contains only terms with x_n^j for $j \in [0, i]$. Thus the modified power series of $x_n^i y + z$ contians only terms with x_n^j for $j \in [0, i]$. Hence we can cancel the next-highest weight term in the same way, etc., continuing until we have a sum of terms whose overall weight is less than that of $x_n^{e_n}\mathrm{elem}(s, n-1)$. (We could even get it to be less than 0.)

Then assuming by induction that $\pi_0 x_0 \mathrm{elem}(a + qb, i)$ is regular for all $\mathrm{elem}(a + qb, i)$ that we have already constructed, we see that $\pi_0 x_0 \mathrm{elem}(r, n)$ is regular of weight $q^{n+1} + r$.

However, although this algorithm appears more "explicit" than the one in Shum et. al.,[3] its runtime is probably slower. Indeed, to do the weight cancellation described above, we need to know the correction term's power series or modifed power series at $P_\infty^{(n)}$. We do not yet know how to compute modified power series; using results in Shum et. al.,[3] we can compute ordinary power series at $P_\infty^{(n)}$, but computing such a series to precision m requires $O(m^2)$ multiplications in $\mathbb{F}_{q^2}$. Since ψ_i may have weight up to approximately $(q^n)^2$, we need to know its power series to

that precision, which takes time approximately $O((q^n)^4)$ to compute, compared to $(q^n)^3\text{poly}(q,n)$ for the algorithm in Shum et. al.[3]

3.5.1 Functions With Small Poles

In this section, we modify the above approach to find functions which are not quite regular, but avoid the q^{4n} bottleneck of the previous section. However, these functions are not regular enough to give good sequences of codes.

For $i \geq 0$, define τ_i by

$$\tau_i = 1 + x_i^{q-1} + x_i^{q-1}x_{i-1}^{q-1} + \cdots + x_i^{q-1}\cdots x_0^{q-1}.$$

It is easy to check that τ_i has zero of the same order as x_{i-1}^{q-1} at $S_i^{(n)}$, is regular at $S_{i+1,n+1}^{(n)}$, and has a pole of the same order as $x_{t+2}^{q-1}x_{t+3}^{q-1}\cdots x_i^{q-1}$ at $S_t^{(n)}$ for $t \in [0, i-1]$. Also, $\tau_i - 1$ has a zero of the same order as x_i^{q-1} at $S_{i+1}^{(n)}$.

Given $n \geq 2$ and $r \geq 0$, we wish to construct a function $\text{elem}(r,n)$ of weight r in F_n, regular except for poles at $S_t^{(n)}$ $(1 \leq t \leq n-2)$ of degree at most $(n-t-1)q^{\max\{t,n-t\}-1}$ and a pole at $S_0^{(n)}$ of degree at most $(q^n-1)+(n-1)q^{n-1}$. It will follow that $\pi_0 f_r^{(n)}$ has weight $q^{n+1}-q^n+r$ and is regular except for poles in $S_{0,n-2}^{(n)}$, and the sum of the maximum possible weighted pole orders at these places is at most $\binom{n}{2}q^n$. (To get asympotically good sequences of codes, we would need the total pole order to be $O(q^{n+1})$.)

Let $r = e_n + qs$. Inductively assume we have constructed all $\text{elem}(a+qb,n)$ for $0 \leq a < e_n$, $0 \leq b$. (In the base case $a = 0$, we set $\text{elem}(qb,n) := \text{elem}(b,n-1)$. In the further base case $n = 1$, we have $\text{elem}(a+qb,1) = x_0^b x_1^a$.)

Start with $x_n^{e_n}\text{elem}(s,n-1)$. We wish to fix the bad pole at $S_{n-1}^{(n)}$. To do this, consider

$$\left(\sum_{i=0}^{e_n}\binom{e_n}{i}x_n^{e_n-i}\frac{x_{n-1}^{\rho(2i)}}{x_{n-2}^i}\right)\text{elem}(s,n-1),$$

where $\rho : \mathbb{Z} \to [1, q-1]$ is such that $\rho(m) \equiv m \pmod{q-1}$. Think of this element as

$$x_n^{e_n}\text{elem}(s,n-1) + \left(\sum_{i=1}^{e_n}\binom{e_n}{i}x_n^{e_n-i}\frac{x_{n-1}^{\rho(2i)}}{x_{n-2}^i}\right)\text{elem}(s,n-1),$$

i.e., the non-0 part of the sum is a "correction" to $x_n^{e_n}\text{elem}(s,n-1)$ designed to make it regular at $S_{n-1}^{(n)}$. The sum is regular at $S_{n,n+1}^{(n)}$ and $S_{0,n-3}^{(n)}$, but it has the problem that it might not be regular at $S_{n-2}^{(n)}$, due to the $x_n^{e_n-i}x_{n-1}^{\rho(2i)}$. To fix this, multiply the correction by τ_{n-2}, i.e., consider the element

$$x_n^{e_n}\text{elem}(s,n-1) + \tau_{n-2}\left(\sum_{i=1}^{e_n}\binom{e_n}{i}x_n^{e_n-i}\frac{x_{n-1}^{\rho(2i)}}{x_{n-2}^i}\right)\text{elem}(s,n-1)$$

By the above remarks about τ_{n-2}, this new correction term has poles at $S_t^{(n)}$, $t \in [0, n-3]$, bounded by those of $x_{t+2}^{q-1} x_{t+3}^{q-1} \cdots x_{n-2}^{q-1}$, and it is regular at $S_{n,n+1}^{(n)}$. Also,

$$(\tau_{n-2} - 1) \left(\sum_{i=1}^{e_n} \binom{e_n}{i} x_n^{e_n - i} \frac{x_{n-1}^{\rho(2i)}}{x_{n-2}^i} \right) \mathrm{elem}(s, n-1)$$

is regular at $S_{n-1}^{(n)}$. Hence the above corrected element has the desired bounds on pole orders in $S_{0,n+1}^{(n)}$. Also, it is easy to see that it is regular at all places besides $P_\infty^{(n)}$ and $S_{0,n+1}^{(n)}$, since any such place assigns a non-zero value to all x_i.

As in the previous section, the weight is bad, we can correct it inductively. We still need to compute the power series of the above element at $P_\infty^{(n)}$, but now this only takes approximately $O((q^n)^2)$ time, since τ_i has weight approximately q^n. However, we have not yet computed the overall runtime required to compute our functions.

4 Acknowledgments

I would foremost like to thank Dr. Anand Narayanan for mentoring me throughout this project, especially for continuing to do so even after he moved to France. I would also like to thank Prof. Chris Umans for useful discussions and advice during the summer, as well as for funding this project through NSF grant #CCF-1423544 and his Simons Foundation Investigator grant. Additionally, I thank the Caltech SFP Office for organizing the SURF program. Finally, I would like to thank Dr. and Mrs. Anthony Skjellum for supporting the SURF program through a generous donation; my SURF is named in honor of Dr. Skjellum's parents.

References

[1] Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248 – 273, 1996.

[2] Gil Cohen and Amnon Ta-Shma. Pseudorandom generators for low degree polynomials from algebraic geometry codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:155, 2013.

[3] K. W. Shum, I. Aleshnikov, P. V. Kumar, H. Stichtenoth, and V. Deolalikar. A low-complexity algorithm for the construction of algebraic-geometric codes better than the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 47(6):2225–2241, Sep 2001.

[4] Ruud Pellikaan. On the missing functions of a pyramid of curves. In *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing*, pages 33–40, Urbana, IL, USA, 1997. University of Illinois at Urbana-Champaign.

[5] Kenneth Shum. *A Low-Complexity Construction of Algebraic Geometric Codes Better Than the Gilbert-Varshamov Bound*. PhD thesis, University of Southern California, 12 2000.

[6] Ruud Pellikaan, Henning Stichtenoth, and Fernando Torres. Weierstrass semigroups in an asymptotically good tower of function fields. *Finite Fields and Their Applications*, 4(4):381 – 392, 1998.