Fast systematic encoding of multiplicity codes

Nicholas Coxon

FAST SYSTEMATIC ENCODING OF MULTIPLICITY CODES

NICHOLAS COXON

Abstract. We present quasi-linear time systematic encoding algorithms for multiplicity codes. The algorithms have their origins in the fast multivariate interpolation and evaluation algorithms of van der Hoeven and Schost (2013), which we generalise to address certain Hermite-type interpolation and evaluation problems. By providing fast encoding algorithms for multiplicity codes, we remove an obstruction on the road to the practical application of the private information retrieval protocol of Augot, Levy-dit-Vehel and Shikfa (2014).

1. Introduction

Multiplicity codes [17, 18] generalise the classical family of Reed–Muller codes by augmenting their construction to include the evaluations of derivatives up to a given order. They inherit the property of being locally correctable from Reed–Muller codes, allowing any specified coordinate of a codeword in a multiplicity code to be recovered with high probability after examining only a sublinear, in the dimension of the code, number of entries in a possibly corrupted version of the codeword. Restricting to Reed–Muller codes while retaining sublinear local correction also restricts the maximum attainable information rate of the codes to roughly a half. Moving to multiplicity codes allows sublinear local correction and rates approaching one [17].

A closely related notion to local correction is that of local decoding [14]. Whereas local correctability is a property of the codewords of a code, local decodability is a property of an encoding function of a code. For local decoding, one is required to recover a specified coordinate of a message after examining only a small number of coordinates in a possibly corrupted version of its encoding. It follows that a locally correctable code that is equipped with a systematic encoding function, i.e., one that embeds messages into their encodings, is also locally decodable. Augot, Levy-dit-Vehel and Ngô [2] provide a systematic encoding function for multiplicity codes by combining results of Kopparty [16] and Key, McDonough and Mavron [15]. By using their encoding function, multiplicity codes offer sublinear local decoding, while still allowing high rates.

The local decoding algorithm of multiplicity codes is randomised, with the queries to a codeword appearing uniformly distributed over its entries when viewed individually. As a result, an information-theoretically secure private information retrieval protocol may be built upon multiplicity codes by using the construction of Katz and Trevisan [14, Section 4]. Private information retrieval [9] allows a user to retrieve entries from an online database without revealing which entries are

INRIA and Laboratoire d'Informatique de l'École polytechnique, Palaiseau, France.
E-mail address: nicholas.coxon@inria.fr.
Date: April 22, 2017.

being retrieved to the database servers. Using multiplicity codes in the construction of Katz and Trevisan yields a protocol with low communication complexity, when compared to the trivial solution of downloading the entire database, since the amount of data transferred to recover a single database entry is roughly equal to amount of codeword data examined during one round of local decoding.

Augot, Levy-dit-Vehel and Shikfa [3] exploit geometric properties of multiplicity codes to improve upon the protocol obtain by the Katz–Trevisan construction, with their protocol incurring a smaller storage overhead and requiring fewer database servers. The protocol begins by systematically encoding the database as a codeword in a multiplicity code. The codeword is then distributed amongst the database servers. It follows that the encoding increases the amount of stored data by a factor equal to the inverse of the information rate of the code. Thus, the protocol favours the use of multiplicity codes over Reed–Muller codes.

For the protocol of Augot, Levy-dit-Vehel and Shikfa to be realisable for large databases, it is necessary that the initial encoding may be performed efficiently. In this paper, we show that it is possible to perform the encoding in time that is quasi-linear in the number of field elements that appear in the codeword.

1.1. **Multiplicity codes.** Let $\mathbb{F}_q$ denote the finite field with q elements. We enumerate the field as $\mathbb{F}_q = \{\alpha_0, \ldots, \alpha_{q-1}\}$ and let $[q] = \{0, 1, \ldots, q-1\}$ denote its index set. Then the elements of $\mathbb{F}_q^n$ are identified with vectors in $[q]^n$ by defining $\boldsymbol{\alpha}_{\boldsymbol{j}} = (\alpha_{j_1}, \ldots, \alpha_{j_n})$ for $\boldsymbol{j} = (j_1, \ldots, j_n) \in [q]^n$. The ring of polynomials over $\mathbb{F}_q$ in indeterminates $X_1, \ldots, X_n$ is denoted by $\mathbb{F}_q[\boldsymbol{X}] = \mathbb{F}_q[X_1, \ldots, X_n]$, and we define $\boldsymbol{X}^{\boldsymbol{i}} = X_1^{i_1} \cdots X_n^{i_n}$ for $\boldsymbol{i} = (i_1, \ldots, i_n) \in \mathbb{N}^n$.

A codeword of a multiplicity code is constructed by taking a polynomial in $\mathbb{F}_q[\boldsymbol{X}]$ and evaluating its Hasse derivatives up to a given order at all points in $\mathbb{F}_q^n$. The Hasse derivatives of a polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]$ are given by the coefficients (in $\mathbb{F}_q[\boldsymbol{X}]$) of the shifted polynomial $F(\boldsymbol{X} + \boldsymbol{Z}) \in \mathbb{F}_q[\boldsymbol{X}][\boldsymbol{Z}] = \mathbb{F}_q[\boldsymbol{X}][Z_1, \ldots, Z_n]$ for algebraically independent indeterminates $Z_1, \ldots, Z_n$ over $\mathbb{F}_q[\boldsymbol{X}]$. For $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$, the coefficient of $\boldsymbol{Z}^{\boldsymbol{s}} = Z_1^{s_1} \cdots Z_n^{s_n}$ in the shifted polynomial is called the $\boldsymbol{s}$th Hasse derivative of F, which we denote by $H(F, \boldsymbol{s})$. Accordingly, we have

$$F(\boldsymbol{X} + \boldsymbol{Z}) = \sum_{\boldsymbol{s} \in \mathbb{N}^n} H(F, \boldsymbol{s})(\boldsymbol{X}) \boldsymbol{Z}^{\boldsymbol{s}}.$$

We define the weight of a vector $\boldsymbol{i} \in \mathbb{N}^n$, denoted $|\boldsymbol{i}|$, to be the sum of its entries. Then the $\boldsymbol{s}$th Hasse derivative is said to have order $|\boldsymbol{s}|$.

The polynomials that have their derivatives evaluated in a multiplicity code are restricted by their (total) degree. Consequently, we let $\mathbb{F}_q[\boldsymbol{X}]_d$ denote the vector space of polynomials in $\mathbb{F}_q[\boldsymbol{X}]$ that have degree at most d. We index the derivatives of order less than s by the set $S_{s,n} = \{\boldsymbol{s} \in \mathbb{N}^n \mid |\boldsymbol{s}| < s\}$, and let $\sigma_{s,n}$ denote its cardinality. Then for $d, s \in \mathbb{N}$ such that $d < sq$, the multiplicity code Mult_d^s is defined to be the image of the map

$$\begin{aligned} \mathrm{ev}_d^s \;:\; \mathbb{F}_q[\boldsymbol{X}]_d &\to \left(\mathbb{F}_q^{\sigma_{s,n}}\right)^{q^n} \\ F &\mapsto \left(\left(H(F, \boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}}) \right)_{\boldsymbol{t} \in S_{s,n}} \right)_{\boldsymbol{j} \in [q]^n}. \end{aligned}$$

Thus, the multiplicity code Mult_d^s is a vector space over $\mathbb{F}_q$ of dimension $\binom{n+d}{n}$, while its minimum distance is at least $(1 - d/(sq))q^n$ [10, Lemma 8] and its information rate is $\binom{n+d}{n} / \left(\binom{n+s-1}{n} q^n \right)$.

1.2. **Systematic encoding of multiplicity codes.** Given a multiplicity code Mult_d^s, it is natural to consider encoding functions that are $\mathbb{F}_q$-linear functions from $\mathbb{F}_q^k$ onto the code, where $k = \binom{n+d}{n}$ is the code's dimension. The elements of $\mathbb{F}_q^k$ are then called the message vectors, or simply messages, of the code. Such an encoding function $\mathrm{enc} : \mathbb{F}_q^k \to \mathrm{Mult}_d^s$ is systematic if the ith entry of each message vector m, for $i \in \{1, \dots, k\}$, always appears in the encoding $\mathrm{enc}(m)$ at some fixed location. Recording these locations yields a set $\mathcal{I} \subseteq [q]^n \times S_{s,n}$ such that the map

$$\begin{aligned} \mathrm{ev}_{\mathcal{I}} \;:\; \mathbb{F}_q[\boldsymbol{X}]_d \;&\to\; \mathbb{F}_q^{\binom{n+d}{n}} \\ F \;&\mapsto\; (H(F, \boldsymbol{t})(\boldsymbol{\alpha_j}))_{(\boldsymbol{j},\boldsymbol{t})\in\mathcal{I}} \end{aligned}$$

is a bijection. Conversely, a set $\mathcal{I} \subseteq [q]^n \times S_{s,n}$ such that the map $\mathrm{ev}_{\mathcal{I}}$ is a bijection induces a systematic encoding function

$$\mathrm{ev}_d^s \circ \mathrm{ev}_{\mathcal{I}}^{-1} : \mathbb{F}_q^{\binom{n+d}{n}} \to \mathrm{Mult}_d^s.$$

Indeed, the function is systematic since the entries of a message vector each reappear in its encoding as the value of some fixed derivative. Such a set $\mathcal{I} \subseteq [q]^n \times S_{s,n}$ is called an interpolating set [16, Appendix A] or an information set [2, Definition 4] of the multiplicity code Mult_d^s.

Kopparty [16, Appendix A] provides a method of constructing information sets, and thus a construction of systematic encoding functions, for multiplicity codes. However, Kopparty does not provide explicit examples of the construction. Augot, Levy-dit-Vehel and Ngô [2] subsequently provide an explicit family of information sets by supplementing Kopparty's construction with a result of Key, McDonough and Mavron [15, Theorem 1].

Theorem 1 ([15, 16, 2])**.** *For $d, s \in \mathbb{N}$ such that $d < sq$,*

$$\mathcal{I}_{d,n} = \{(\boldsymbol{i}, \boldsymbol{s}) \in [q]^n \times \mathbb{N}^n \mid |\boldsymbol{i} + \boldsymbol{s}q| \leq d\}$$

is an information set of Mult_d^s.

We let $\mathrm{enc}_d^s = \mathrm{ev}_d^s \circ \mathrm{ev}_{\mathcal{I}_{d,n}}^{-1}$ denote the systematic encoding function of Mult_d^s provided by Theorem 1. A codeword of a multiplicity code Mult_d^s contains q^n elements of $\mathbb{F}_q^{\sigma_{s,n}}$, and thus contains $\sigma_{s,n}q^n$ field elements in total. Consequently, if the encoding function enc_d^s is to be used in the private information retrieval protocol of Augot, Levy-dit-Vehel and Shikfa [3], then it is important that the function may be evaluated in time that is close to linear in $\sigma_{s,n}q^n$. Augot, Levy-dit-Vehel and Ngô [2, Appendix] show that enc_d^s can be evaluated in $\tilde{\mathcal{O}}(\sigma_{s,n}^3 q^n + k^2)$ operations in $\mathbb{F}_q$, where $k = \binom{n+d}{n}$ and the notation $\tilde{\mathcal{O}}(\,\cdot\,)$ indicates that polylogarithmic factors are omitted from the complexity. The quadratic dependency on the dimension of the code means that their algorithm is not suitable for use in the private information retrieval context, where $k \log_2 q$ must be greater than or equal to the number of bits in the database, and q is the number of (non-colluding) servers. However, we note that the cost of evaluating enc_d^s with their algorithm can be reduced to $\tilde{\mathcal{O}}(\sigma_{s,n}^3 q^n)$ operations in $\mathbb{F}_q$ by replacing the matrix–vector products they use to perform multivariate interpolation with the quasi-linear time interpolation algorithm of van der Hoeven and Schost [23].

1.3. **Our contribution.** In Sections 3 and 4, we present two algorithms that evaluate the encoding function enc_d^s in $\mathcal{O}(\sigma_{s,n} q^n n \log^2(sq) \log\log(sq))$, or more simply $\tilde{\mathcal{O}}(\sigma_{s,n} q^n)$, operations in $\mathbb{F}_q$. The algorithm of Section 3 combines fast polynomial interpolation and evaluation algorithms to first invert the map $\mathrm{ev}_{\mathcal{I}_{d,n}}$ then evaluate ev_d^s. The algorithm of Section 4 follows a similar interpolation–evaluation approach, but aims to trade a more expensive interpolation step for a cheaper evaluation step. While the two encoding algorithms achieve the same asymptotic complexity, comparing lower order terms of their complexities suggests that they outperform each other at opposing ends of the rate spectrum, with the algorithm of Section 3 being faster for low-rate codes. Consequently, the two encoding algorithms provide complementary practical performance.

For the private information retrieval protocol of Augot, Levy-dit-Vehel and Shikfa [3] one desires to use multiplicity codes with high rates in order to obtain small storage overheads. However, storage overhead must be balanced with other aspects of the protocol when choosing parameters for the codes. The problem of parameter selection is yet to be addressed in the literature, and it remains unclear as to which rates will occur in practice. We do not address this problem here since it is out of the scope of the paper. As a result, we are prevented from determining if one of the two encoding algorithms is better suited to this application.

The interpolation and evaluation algorithms that make up the two encoding algorithms have their origins in the quasi-linear time multivariate interpolation and evaluation algorithms of van der Hoeven and Schost [23]. In Section 2, we generalise their algorithms to address certain multivariate Hermite interpolation and evaluation problems. Thus, we provide algorithms for recovering multivariate polynomials from their Hasse derivatives, as well as for the inverse problem of computing their derivatives.

The algorithms of van der Hoeven and Schost are recursive in nature, reducing each problem to multiple instances of the same problem in a single variable. Solving the univariate problems in quasi-linear time, then leads to an overall quasi-linear time algorithm. Our Hermite interpolation and evaluation algorithms similarly reduce the multivariate problems to multiple instances of the univariate problems. Applying the quasi-linear time algorithms of Chin [8] to these univariate instances then yields multivariate algorithms with quasi-linear complexity.

Conventions. We let $\mathsf{M} : \mathbb{N} \setminus \{0\} \to \mathbb{N}$ denote a function such that two univariate polynomials over $\mathbb{F}_q$ of degree less than k can be multiplied in $\mathsf{M}(k)$ operations in $\mathbb{F}_q$. For example, the algorithm of Cantor and Kaltofen [7] implies that $\mathsf{M}(k)$ may be taken to be in $\mathcal{O}(k \log k \log\log k)$. Throughout the paper, we assume that $\mathsf{M}(k)/k$ is a nondecreasing function of k.

We make frequently use of the shorthand vector notation $(f_{\boldsymbol{i}})_{\boldsymbol{i} \in I}$ for sets $I \subseteq \mathbb{N}^\ell$. So that this notation is well-defined, we order the entries $f_{\boldsymbol{i}}$ by increasing weight $|\boldsymbol{i}|$ of their multi-indices, with ties broken lexicographically. Similarly, for sets $\mathcal{I} \subseteq [q]^\ell \times \mathbb{N}^\ell$, we assume that the entries of a vector $(f_{(\boldsymbol{i},\boldsymbol{s})})_{(\boldsymbol{i},\boldsymbol{s}) \in \mathcal{I}}$ are ordered by increasing $|\boldsymbol{i} + \boldsymbol{s}q|$, with ties broken by comparing the vectors $\boldsymbol{i} + \boldsymbol{s}q$ lexicographically.

For $\boldsymbol{i} = (i_1, \ldots, i_n) \in \mathbb{Z}^n$ and $j \in \mathbb{N} \setminus \{0\}$, we define $\boldsymbol{i} \operatorname{div} j = (\lfloor i_1/j \rfloor, \ldots, \lfloor i_n/j \rfloor)$ and $\boldsymbol{i} \bmod j = \boldsymbol{i} - (\boldsymbol{i} \operatorname{div} j) j$. Similarly, for $F, G \in \mathbb{F}_q[X]$ such that $\deg G > 0$, we write $F \bmod G$ for the residue of F modulo G that has degree less than $\deg G$.

2. Multivariate Hermite interpolation and evaluation

The interpolation algorithm of van der Hoeven and Schost [23], when applied over $\mathbb{F}_q$, takes as an input a vector of field elements $(m_{\boldsymbol{j}})_{\boldsymbol{j}\in I}$ for some $I \subseteq [q]^n$, and returns the unique polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]$ that has support contained in I and satisfies $F(\boldsymbol{\alpha}_{\boldsymbol{j}}) = m_{\boldsymbol{j}}$ for $\boldsymbol{j} \in I$. Their evaluation algorithm performs the inverse computation, evaluating a polynomial with support contained in I at the points $\boldsymbol{\alpha}_{\boldsymbol{j}}$ for $\boldsymbol{j} \in I$. Both algorithms require I to be an initial segment for the partial order $\leq$ on $\mathbb{N}^n$ defined by $\boldsymbol{i} \leq \boldsymbol{j}$ if and only if $\boldsymbol{j} - \boldsymbol{i} \in \mathbb{N}^n$: a subset $I \subseteq \mathbb{N}^n$ is then an initial segment if it is nonempty and contains all $\boldsymbol{i} \in \mathbb{N}^n$ such that $\boldsymbol{i} \leq \boldsymbol{j}$ for some $\boldsymbol{j} \in I$.

For $I \subseteq \mathbb{N}^n$, let $\mathbb{F}_q[\boldsymbol{X}]_I$ denote the vector space of polynomials in $\mathbb{F}_q[\boldsymbol{X}]$ that have support contained in I. Then a key feature of the algorithms of van der Hoeven and Schost is the representation of polynomials in $\mathbb{F}_q[\boldsymbol{X}]_I$, where $I \subseteq [q]^n$ is an initial segment, with respect to a multivariate Newton basis. This basis consists of the polynomials

$$N_{\boldsymbol{i}}(\boldsymbol{X}) = N_{i_1}(X_1) \cdots N_{i_n}(X_n) \quad \text{for } \boldsymbol{i} = (i_1, \ldots, i_n) \in I,$$

where

$$N_i(X) = (X - \alpha_0) \cdots (X - \alpha_{i-1}) \quad \text{for } i \in [q]$$

are the Newton polynomials associated with the enumeration of the field. The Newton basis polynomial $N_{\boldsymbol{i}}$ vanishes at all points $\boldsymbol{\alpha}_{\boldsymbol{j}}$ with $\boldsymbol{j} \not\geq \boldsymbol{i}$, allowing van der Hoeven and Schost to address the interpolation and evaluation problems one variable at a time in a manner similar to the earlier work of Pan [21]. In doing so, they obtain algorithms for both problems that each perform $\mathcal{O}(|I| n \log^2 |I| \log\log |I|)$ field operations.

In this section, we generalise the interpolation and evaluation algorithms of van der Hoeven and Schost to address multivariate Hermite interpolation and evaluation problems. The generalised algorithms yield analogous complexities to those of the algorithms of van der Hoeven and Schost. Thus, they allow the fast recovery of polynomials from the values of their Hasse derivatives, in addition to allowing the fast evaluation of their derivatives.

2.1. Hermite interpolation and evaluation.

We generalise the interpolation and evaluation problems considered by van der Hoeven and Schost through generalising the use of the multivariate Newton basis. To allow initial segments that are not contained in $[q]^n$, we extend the definition of the Newton basis by introducing repeated roots to the basis polynomials. We define

$$N_i(X) = \prod_{j=0}^{i-1} (X - \alpha_{j \bmod q}) \quad \text{for } i \in \mathbb{N},$$

and $N_{\boldsymbol{i}}(\boldsymbol{X}) = N_{i_1}(X_1) \cdots N_{i_n}(X_n)$ for $\boldsymbol{i} = (i_1, \ldots, i_n) \in \mathbb{N}^n$. Then, for $\boldsymbol{i} \in \mathbb{N}^n$, the polynomial $N_{\boldsymbol{i}}$ may be written in the form $\sum_{\boldsymbol{k} \leq \boldsymbol{i}} n_{\boldsymbol{k}} \boldsymbol{X}^{\boldsymbol{k}}$ with coefficients $n_{\boldsymbol{k}} \in \mathbb{F}_q$ and $n_{\boldsymbol{i}} = 1$. Therefore, under the extended definition we retain the property that $\{N_{\boldsymbol{i}} \mid \boldsymbol{i} \in I\}$ is a basis of $\mathbb{F}_q[\boldsymbol{X}]_I$ when $I \subseteq \mathbb{N}^n$ is an initial segment. However, having introduced repeated roots to the basis polynomials, the vanishing property of the Newton basis now extends to include the Hasse derivatives of the basis polynomials.

Lemma 2. *For $\boldsymbol{i} \in \mathbb{N}^n$ and $(\boldsymbol{j}, \boldsymbol{t}) \in [q]^n \times \mathbb{N}^n$, we have $H(N_{\boldsymbol{i}}, \boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}}) = 0$ if $\boldsymbol{j} + \boldsymbol{t}q \not\geq \boldsymbol{i}$, and $H(N_{\boldsymbol{i}}, \boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}}) \neq 0$ if $\boldsymbol{j} + \boldsymbol{t}q = \boldsymbol{i}$.*

Proof. It is sufficient to prove the lemma for all $N_{\boldsymbol{i}+\boldsymbol{s}q}$ with $(\boldsymbol{i},\boldsymbol{s}) \in [q]^n \times \mathbb{N}^n$. Let $(\boldsymbol{i},\boldsymbol{s}),(\boldsymbol{j},\boldsymbol{t}) \in [q]^n \times \mathbb{N}^n$ with $\boldsymbol{i} = (i_1,\dots,i_n)$, $\boldsymbol{s} = (s_1,\dots,s_n)$, $\boldsymbol{j} = (j_1,\dots,j_n)$ and $\boldsymbol{t} = (t_1,\dots,t_n)$. Then, for algebraically independent indeterminates $Z_1,\dots,Z_n$ over $\mathbb{F}_q[\boldsymbol{X}]$, the definition of the Hasse derivative implies that $H(N_{\boldsymbol{i}+\boldsymbol{s}q},\boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}})$ is equal to the coefficient of $\boldsymbol{Z}^{\boldsymbol{t}} = Z_1^{t_1}\cdots Z_n^{t_n}$ in the polynomial $N_{\boldsymbol{i}+\boldsymbol{s}q}(\boldsymbol{Z}+\boldsymbol{\alpha}_{\boldsymbol{j}}) \in \mathbb{F}_q[\boldsymbol{Z}] = \mathbb{F}_q[Z_1,\dots,Z_n]$.

For $\ell \in \{1,\dots,n\}$, let $\varepsilon_\ell : \mathbb{N} \to \{0,1\}$ be the indicator function defined by $\varepsilon_\ell(k) = 1$ if and only if $k < i_\ell$. Then

$$N_{i_\ell+s_\ell q}(X_\ell) = \prod_{k\in[q]} (X_\ell - \alpha_k)^{s_\ell+\varepsilon_\ell(k)} \quad \text{for } \ell = 1,\dots,n.$$

Letting $\boldsymbol{\varepsilon} = (\varepsilon_1(j_1),\dots,\varepsilon_n(j_n))$, it follows that

$$N_{\boldsymbol{i}+\boldsymbol{s}q}(\boldsymbol{Z}+\boldsymbol{\alpha}_{\boldsymbol{j}}) = \boldsymbol{Z}^{\boldsymbol{s}+\boldsymbol{\varepsilon}} \prod_{\ell=1}^{n} \prod_{k\in[q]\setminus\{j_\ell\}} (Z_\ell + \alpha_{j_\ell} - \alpha_k)^{s_\ell+\varepsilon_\ell(k)}. \tag{1}$$

If $\boldsymbol{t} \geq \boldsymbol{s}+\boldsymbol{\varepsilon}$, then $\boldsymbol{j}+\boldsymbol{t}q \geq (\boldsymbol{j}+\boldsymbol{\varepsilon}q)+\boldsymbol{s}q \geq \boldsymbol{i}+\boldsymbol{s}q$. Therefore, if $\boldsymbol{j}+\boldsymbol{t}q \ngeq \boldsymbol{i}+\boldsymbol{s}q$, then $H(N_{\boldsymbol{i}+\boldsymbol{s}q},\boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}}) = 0$ since $\boldsymbol{t} \ngeq \boldsymbol{s}+\boldsymbol{\varepsilon}$ and $\boldsymbol{Z}^{\boldsymbol{s}+\boldsymbol{\varepsilon}}$ divides $N_{\boldsymbol{i}+\boldsymbol{s}q}(\boldsymbol{Z}+\boldsymbol{\alpha}_{\boldsymbol{j}})$ in $\mathbb{F}_q[\boldsymbol{Z}]$. If $\boldsymbol{j}+\boldsymbol{t}q = \boldsymbol{i}+\boldsymbol{s}q$, then $(\boldsymbol{j},\boldsymbol{t}) = (\boldsymbol{i},\boldsymbol{s})$ and $\boldsymbol{\varepsilon} = \boldsymbol{0}$. By substituting into (1) and computing the coefficient of $\boldsymbol{Z}^{\boldsymbol{s}} = \boldsymbol{Z}^{\boldsymbol{s}+\boldsymbol{\varepsilon}}$, we find that

$$H(N_{\boldsymbol{i}+\boldsymbol{s}q},\boldsymbol{s})(\boldsymbol{\alpha}_{\boldsymbol{i}}) = \prod_{\ell=1}^{n} \prod_{k\in[q]\setminus\{i_\ell\}} (\alpha_{i_\ell} - \alpha_k)^{s_\ell+\varepsilon_\ell(k)},$$

which is nonzero. □

In the interpolation and evaluation problems considered by van der Hoeven and Schost, the initial segment I is the support of both the polynomials and the evaluation points. In order to maintain this property when generalising these problems, we define $E(F,\boldsymbol{i}) = H(F,\boldsymbol{i} \operatorname{div} q)(\boldsymbol{\alpha}_{\boldsymbol{i} \bmod q})$ for $\boldsymbol{i} \in \mathbb{N}^n$ and $F \in \mathbb{F}_q[\boldsymbol{X}]$. Then our problem of Hermite interpolation takes a vector $(m_{\boldsymbol{j}})_{\boldsymbol{j}\in I}$ of field elements for some finite initial segment $I \subseteq \mathbb{N}^n$ and asks that we compute the polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ that satisfies $E(F,\boldsymbol{j}) = m_{\boldsymbol{j}}$ for $\boldsymbol{j} \in I$. Our Hermite evaluation problem is the inverse problem, asking for the computation of the vector $(E(F,\boldsymbol{j}))_{\boldsymbol{j}\in I}$ when given a polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$. Importantly, Lemma 2 implies that $E(N_{\boldsymbol{i}},\boldsymbol{j}) = 0$ for all $\boldsymbol{i},\boldsymbol{j} \in \mathbb{N}^n$ such that $\boldsymbol{i} \ngeq \boldsymbol{j}$, allowing us to address both problems by generalising the algorithms of van der Hoeven and Schost. Existence and uniqueness of a solution to the Hermite interpolation problem is provided by the following lemma.

Lemma 3. *Let $I \subseteq \mathbb{N}^n$ be a finite initial segment and $(m_{\boldsymbol{j}})_{\boldsymbol{j}\in I} \in \mathbb{F}_q^{|I|}$. Then there exists a unique polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ such that $E(F,\boldsymbol{j}) = m_{\boldsymbol{j}}$ for $\boldsymbol{j} \in I$.*

Proof. If $I \subseteq \mathbb{N}^n$ is a finite initial segment, then $\mathbb{F}_q[\boldsymbol{X}]_I$ and $\mathbb{F}_q^{|I|}$ are $|I|$-dimensional $\mathbb{F}_q$-vector spaces, and $I \subseteq S_{s,n}$ for $s > \max_{\boldsymbol{i}\in I}|\boldsymbol{i}|$. Therefore, it is sufficient to prove following statement: for all positive $s \in \mathbb{N}$, if $I \subseteq S_{s,n}$ is an initial segment, then the homomorphism $\operatorname{ev}_I : \mathbb{F}_q[\boldsymbol{X}]_I \to \mathbb{F}_q^{|I|}$ given by $F \mapsto (E(F,\boldsymbol{j}))_{\boldsymbol{j}\in I}$ is injective. We prove this statement by induction on s. The statement holds trivially for $s = 1$, since $\{\boldsymbol{0}\}$ is the only initial segment contained in $S_{1,n}$, and $\operatorname{ev}_{\{\boldsymbol{0}\}} : \mathbb{F}_q \to \mathbb{F}_q$ is the identity map. Therefore, suppose that the statement is true for some integer $s \geq 1$. Let $I \subseteq S_{s+1,n}$ be an initial segment, and $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ such that $E(F,\boldsymbol{j}) = 0$ for

$\boldsymbol{j} \in I$. Then to complete the proof of the lemma, it is sufficient to show that F is equal to zero.

Let $J = I \cap S_{s,n}$. Then J is an initial segment since I and $S_{s,n}$ are initial segments. Moreover, if $\boldsymbol{i} \in I \setminus J$, then its weight $|\boldsymbol{i}|$ is maximal amongst the elements of I. Consequently, if $\boldsymbol{i} \in I \setminus J$, then $\boldsymbol{j} \not\geq \boldsymbol{i}$ for $\boldsymbol{j} \in I \setminus \{\boldsymbol{i}\}$. As Hasse derivatives are linear functions and evaluation is a homomorphism, the functions $E(\,\cdot\,, \boldsymbol{j}) : \mathbb{F}_q[\boldsymbol{X}] \to \mathbb{F}_q$ for $\boldsymbol{j} \in \mathbb{N}$ are linear. Therefore, if we write $F = \sum_{\boldsymbol{i} \in I} f_{\boldsymbol{i}} N_{\boldsymbol{i}}$ such that $f_{\boldsymbol{i}} \in \mathbb{F}_q$ for all $\boldsymbol{i} \in I$, then Lemma 2 implies that

$$0 = E\left(\sum_{\boldsymbol{i} \in J} f_{\boldsymbol{i}} N_{\boldsymbol{i}}, \boldsymbol{j}\right) + \sum_{\boldsymbol{i} \in I \setminus J} f_{\boldsymbol{i}} E(N_{\boldsymbol{i}}, \boldsymbol{j}) = E\left(\sum_{\boldsymbol{i} \in J} f_{\boldsymbol{i}} N_{\boldsymbol{i}}, \boldsymbol{j}\right) \quad \text{for } \boldsymbol{j} \in J.$$

As $J \subseteq S_{s,n}$ is an initial segment, the induction hypothesis implies that $\sum_{\boldsymbol{i} \in J} f_{\boldsymbol{i}} N_{\boldsymbol{i}}$ is equal to zero. Applying Lemma 2 once again, it follows that

$$0 = \sum_{\boldsymbol{i} \in I \setminus J} f_{\boldsymbol{i}} E(N_{\boldsymbol{i}}, \boldsymbol{j}) = f_{\boldsymbol{j}} E(N_{\boldsymbol{j}}, \boldsymbol{j}) \quad \text{for } \boldsymbol{j} \in I \setminus J.$$

Moreover, the lemma states that $E(N_{\boldsymbol{j}}, \boldsymbol{j}) \neq 0$ for $\boldsymbol{j} \in I \setminus J$. Therefore, $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \setminus J$. Hence, F is equal to zero. □

Define $\kappa_n : [q]^n \times \mathbb{N}^n \to \mathbb{N}^n$ by $(\boldsymbol{i}, \boldsymbol{s}) \mapsto \boldsymbol{i} + \boldsymbol{s}q$. Then $\kappa_n(\mathcal{I}_{d,n}) = \{\boldsymbol{i} \in \mathbb{N}^n \mid |\boldsymbol{i}| \leq d\}$ is a finite initial segment for $d \in \mathbb{N}$. Moreover, for $F \in \mathbb{F}_q[\boldsymbol{X}]_d = \mathbb{F}_q[\boldsymbol{X}]_{\kappa_n(\mathcal{I}_{d,n})}$ we have $(H(F, \boldsymbol{t})(\boldsymbol{\alpha}_{\boldsymbol{j}}))_{(\boldsymbol{j}, \boldsymbol{t}) \in \mathcal{I}_{d,n}} = (E(F, \boldsymbol{j}))_{\boldsymbol{j} \in \kappa_n(\mathcal{I}_{d,n})}$. Thus, the problem of computing the polynomial that corresponds to a message vector of a multiplicity code Mult_d^s is an instance of the Hermite interpolation problem with initial segment $I = \kappa_n(\mathcal{I}_{d,n})$. Similarly, the problem of encoding a polynomial as a codeword in Mult_d^s, i.e., evaluating the map ev_d^s for some polynomial of degree at most $d < sq$, is an instance of the Hermite evaluation problem with initial segment $I = \kappa_n([q]^n \times S_{s,n}) \supset \kappa_n(\mathcal{I}_{d,n})$. In Section 3, we apply the fast algorithms developed in this section to these two instances to obtain a fast systematic encoding algorithm for low-rate codes, while in Section 4, the interpolation algorithm is applied with $I = \kappa_n([q]^n \times S_{s,n})$ as part of the encoding algorithm for higher rate codes.

As we have no need to represent polynomials with respect to the monomial basis during encoding, we only require that the output of the interpolation algorithm and input of the evaluation algorithm are written on the Newton basis. Consequently, if $I \subseteq \mathbb{N}^n$ is an initial segment, then we write $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i} \in I}$ for the vector of coefficients of $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ when written on the basis $(N_{\boldsymbol{i}})_{\boldsymbol{i} \in I}$. That is, if $F = \sum_{\boldsymbol{i} \in I} f_{\boldsymbol{i}} N_{\boldsymbol{i}}$ such that the coefficients $f_{\boldsymbol{i}} \in \mathbb{F}_q$, then $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i} \in I} = (f_{\boldsymbol{i}})_{\boldsymbol{i} \in I}$. Similarly, we write $F \dashv (\boldsymbol{X}^{\boldsymbol{i}})_{\boldsymbol{i} \in I}$ for the coefficient vector of F when written on the monomial basis. To allow us to bound the size of a finite initial segment in each of its dimensions, we extend the notation $[q]$ by defining $[s] = \{0, 1, \ldots, s-1\}$ for positive $s \in \mathbb{N}$. Using this notation, we can state the main result of this section as follows.

Theorem 4. *Let $I \subseteq \mathbb{N}^n$ be an initial segment such that $I \subseteq [s_1] \times \cdots \times [s_n]$ for positive integers $s_1, \ldots, s_n$. Then given the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j} \in I}$ for some polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$, the vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i} \in I}$ can be computed in*

$$\mathcal{O}\left(\left(\frac{\mathsf{M}(s_1) \log s_1}{s_1} + \cdots + \frac{\mathsf{M}(s_n) \log s_n}{s_n}\right)|I|\right) \tag{2}$$

operations in $\mathbb{F}_q$. Conversely, given the vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$ for some polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$, the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j}\in I}$ can be computed within the same bound on the number of operations in $\mathbb{F}_q$.

Theorem 4 directly generalises the bounds obtained by van der Hoeven and Schost [23, Propostions 2 and 3] for interpolation and evaluation. By letting $s_1, \ldots, s_n$ equal $|I|$, and taking $\mathsf{M}(k)$ to be in $\mathcal{O}(k \log k \log\log k)$, the bound (2) simplifies to $\mathcal{O}(|I| n \log^2 |I| \log\log |I|)$, matching the bound for the algorithms of van der Hoeven and Schost stated at the beginning of the section.

In other settings it may be preferable to have the output of the Hermite interpolation algorithm or the input of the Hermite evaluation algorithm represented with respect to the monomial basis. For univariate polynomials, conversion between the Newton and monomial bases can be performed in quasi-linear time by the algorithms discussed in the next section. These algorithms extend to multivariate polynomials by applying the approach of van der Hoeven and Schost [23, Section 4]. Using these algorithms, it is possible to preserve the bound (2) while having the input and output polynomials of the Hermite interpolation and evaluation algorithms given on the monomial basis.

The remainder of this section is devoted to proving Theorem 4. We begin in the next section by reviewing existing fast algorithms for solving the Hermite interpolation and evaluation problems in univariate case. Then we complete the proof of the theorem by generalising the multivariate interpolation and evaluation algorithms of van der Hoeven and Schost in Section 2.3.

2.2. **Univariate algorithms.** Hermite interpolation and evaluation for univariate polynomials can be performed in quasi-linear time with respect to the monomial basis by the algorithms of Chin [8]. In these algorithms, derivative is taken to mean the formal derivative rather than the Hasse derivative, as required here. However, by using the fact that the ith formal derivative is equal to $i!$ times the ith Hasse derivative, it is readily shown that only superficial changes to Chin's algorithms are required to allow them to work with the Hasse derivative. We note that the convolution-based algorithm of Aho, Steiglitz and Ullman [1] that is used by Chin to compute Taylor shifts of polynomials cannot be used if the characteristic of the field is not greater than their degrees. In this case, the convolution-based algorithm may be replaced by the algorithm of Olshevsky and Shokrollahi [20, Section 4.2] (see also [24, 25]), which is slower by a logarithmic factor.

Each finite initial segment in $\mathbb{N}$ is of the form $[s]$ for some positive integer s. For the Hermite interpolation and evaluation problems defined by these initial segments, applying Chin's algorithms with modifications just described provides the following complexity bounds.

Lemma 5. *Let $s \in \mathbb{N}$ be positive and $F \in \mathbb{F}_q[X]_{[s]}$. Then given $(E(F, j))_{j\in[s]}$, the vector $F \dashv (X^i)_{i\in[s]}$ can be computed in $\mathcal{O}(\mathsf{M}(s) \log s)$ operations in $\mathbb{F}_q$. Conversely, given $F \dashv (X^i)_{i\in[s]}$, the vector $(E(F, j))_{j\in[s]}$ can be computed in $\mathcal{O}(\mathsf{M}(s) \log s)$ operations in $\mathbb{F}_q$.*

Closely related alternatives to Chin's algorithms that provide the same complexity bounds are given by Olshevsky and Shokrollahi [20] and texts [22, Chapter 3], [4, Chapter 1, Section 4] and [6, Exercise 3.14]. In situations where precomputation is permitted, the asymptotic complexity of these algorithms and Chin's algorithms may be improved upon by using the techniques described by van der Hoeven [13].

Combining Lemma 5 with the following result of Gerhard [12] completes the proof of Theorem 4 for the univariate case.

Lemma 6. *Let $s \in \mathbb{N}$ be positive and $F \in \mathbb{F}_q[X]_{[s]}$. Then given $F \dashv (N_i)_{i\in[s]}$, the vector $F \dashv (X^i)_{i\in[s]}$ can be computed in $\mathcal{O}(\mathsf{M}(s)\log s)$ operations in $\mathbb{F}_q$. Conversely, given $F \dashv (X^i)_{i\in[s]}$, the vector $F \dashv (N_i)_{i\in[s]}$ can be computed in $\mathcal{O}(\mathsf{M}(s)\log s)$ operations in $\mathbb{F}_q$.*

When converting from the monomial basis to the Newton basis, the algorithm of Gerhard is improved upon in practice by the algorithm of Bostan and Schost [5].

2.3. **Multivariate algorithms.** By design, the Hermite interpolation and evaluation problems allow the algorithms of van der Hoeven and Schost to be generalised in a straightforward manner. However, we follow a slightly different course by presenting the generalised algorithms in an iterative, rather than recursive, form. This small change is used to simplify the description of modifications to the algorithms that are made in the encoding context.

We begin by introducing some geometric operations on initial segments. For $I \subseteq \mathbb{N}^n$ and $\boldsymbol{i} = (i_1, \dots, i_\ell) \in \mathbb{N}^\ell$ such that $1 \le \ell < n$, define

$$\lambda(I, \boldsymbol{i}) = \big\{(j_1, \dots, j_{n-\ell}) \in \mathbb{N}^{n-\ell} \mid (j_1, \dots, j_{n-\ell}, i_1, \dots, i_\ell) \in I\big\}$$

and

$$\rho(I, \boldsymbol{i}) = \big\{(j_1, \dots, j_{n-\ell}) \in \mathbb{N}^{n-\ell} \mid (i_1, \dots, i_\ell, j_1, \dots, j_{n-\ell}) \in I\big\}.$$

Let $\mathbf{0}_\ell$ denote the ℓ-dimensional vector of zeros. Then, given an initial segment $I \subseteq \mathbb{N}^n$ and a positive integer $\ell < n$, the set $\lambda(I, \mathbf{0}_\ell)$ is the projection of I onto the $(i_1, \dots, i_{n-\ell})$-coordinate plane, while $\rho(I, \mathbf{0}_\ell)$ is the projection of I onto the $(i_{\ell+1}, \dots, i_n)$-coordinate plane. Consequently, if $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ has coefficient vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$, then

$$F = \sum_{i_n \in \rho(I, \mathbf{0}_{n-1})} F_{i_n}(X_1, \dots, X_{n-1}) N_{i_n}(X_n) \tag{3}$$

where

$$F_{i_n}(X_1, \dots, X_{n-1}) = \sum_{(i_1, \dots, i_{n-1}) \in \lambda(I, i_n)} f_{(i_1, \dots, i_{n-1}, i_n)} N_{i_1}(X_1) \cdots N_{i_{n-1}}(X_{n-1}) \tag{4}$$

for $i_n \in \rho(I, \mathbf{0}_{n-1})$. For $F \in \mathbb{F}_q[\boldsymbol{X}]$ and $i_n \in \mathbb{N}$, we define $F_{i_n} \in \mathbb{F}_q[X_1, \dots, X_{n-1}]$ to be the polynomial given by (4) for $I = \mathbb{N}^n$ and $(f_{\boldsymbol{i}})_{\boldsymbol{i}\in\mathbb{N}^n} = F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in\mathbb{N}^n}$. Then (3) and (4) still hold whenever $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ for some initial segment $I \subseteq \mathbb{N}^n$, but the definition of F_{i_n} is now independent of I.

We base our Hermite interpolation and evaluation algorithms on the following analogue of [23, Proposition 1] for the functions $E(\,\cdot\,, \boldsymbol{j})$.

Lemma 7. *Let $I \subseteq \mathbb{N}^n$ be a finite initial segment and $F \in \mathbb{F}_q[\boldsymbol{X}]_I$. Then*

$$E(F, \boldsymbol{j}) = E\left(\sum_{i_n \in \rho(I, (j_1, \dots, j_{n-1}))} E(F_{i_n}, (j_1, \dots, j_{n-1})) N_{i_n}(X_n), j_n\right)$$

for all $\boldsymbol{j} = (j_1, \dots, j_n) \in I$.

Proof. We begin the proof by establishing a multiplicative property of the functions $E(\,\cdot\,, \boldsymbol{j})$. Let $U \in \mathbb{F}_q[X_1, \ldots, X_{n-1}]$ and $V \in \mathbb{F}_q[X_n]$. Then it follows from the definition of the Hasse derivative that $H(UV, \boldsymbol{s}) = H(U, (s_1, \ldots, s_{n-1}))H(V, s_n)$ for $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n$. As evaluation is a homomorphism, we conclude that $E(UV, \boldsymbol{j}) = E(U, (j_1, \ldots, j_{n-1}))E(V, j_n)$ for $\boldsymbol{j} = (j_1, \ldots, j_n) \in \mathbb{N}^n$.

Suppose now that $I \subseteq \mathbb{N}^n$ is a finite initial segment, $F \in \mathbb{F}_q[\boldsymbol{X}]_I$ and $\boldsymbol{j} = (j_1, \ldots, j_n) \in I$. Then (3) holds, from which it follows that

$$E(F, \boldsymbol{j}) = \sum_{i_n \in \rho(I, \mathbf{0}_{n-1})} E(F_{i_n}, (j_1, \ldots, j_{n-1}))E(N_{i_n}, j_n).$$

As $E(\,\cdot\,, j_n) : \mathbb{F}_q[X_n] \to \mathbb{F}_q$ is a linear function, the proof of the lemma will be complete if we show that $E(N_{i_n}, j_n) = 0$ for $i_n \in \rho(I, \mathbf{0}_{n-1}) \setminus \rho(I, (j_1, \ldots, j_{n-1}))$. If $i_n \in \rho(I, \mathbf{0}_{n-1})$ and $i_n \leq j_n$, then $i_n \in \rho(I, (j_1, \ldots, j_{n-1}))$ since I is an initial segment and $(j_1, \ldots, j_{n-1}, i_n) \leq (j_1, \ldots, j_{n-1}, j_n) \in I$. As a result, $i_n > j_n$ for $i_n \in \rho(I, \mathbf{0}_{n-1}) \setminus \rho(I, (j_1, \ldots, j_{n-1}))$. Hence, Lemma 2 implies that $E(N_{i_n}, j_n) = 0$ for $i_n \in \rho(I, \mathbf{0}_{n-1}) \setminus \rho(I, (j_1, \ldots, j_{n-1}))$. □

Lemma 7 sets up a natural recursive approach to the Hermite interpolation and evaluation problems by reducing each problem to a combination of univariate problems in the variable X_n, and the recovery or evaluation of the $(n-1)$-variate polynomials F_{i_n}. To allow us to instead present iterative algorithms, we must introduce some additional geometric operations on initial segments. For $n \geq 2$, $I \subseteq \mathbb{N}^n$ and $\ell \in \{1, \ldots, n\}$, we define

$$\pi_\ell(I) = \{(i_1, \ldots, i_{\ell-1}, i_{\ell+1}, \ldots, i_n) \mid (i_1, \ldots, i_n) \in I\}$$

to be the projection of I onto the $(i_1, \ldots, i_{\ell-1}, i_{\ell+1}, \ldots, i_n)$-coordinate plane. For $\boldsymbol{i} = (i_1, \ldots, i_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$, we define

$$\mu_\ell(I, \boldsymbol{i}) = \{i_\ell \in \mathbb{N} \mid (i_1, \ldots, i_{\ell-1}, i_\ell, i_{\ell+1}, \ldots, i_n) \in I\}.$$

We extend these definitions to $I \subseteq \mathbb{N}$ by defining $\pi_1(I) = \{0\}$ and $\mu_1(I, 0) = I$. When $I \subseteq \mathbb{N}^n$ is an initial segment, so too are the sets $\mu_\ell(I, \boldsymbol{i})$ for $\boldsymbol{i} \in \pi_\ell(I)$.

The multivariate Hermite evaluation and interpolation algorithms are presented in Algorithms 1 and 2, respectively. We require that the univariate algorithms they use to be in-place algorithms in the sense that inputs are overwritten by their corresponding output. In particular, the input and output specifications of the univariate algorithms should match those of their corresponding multivariate algorithm for $n = 1$. However, we do not impose restrictions on the memory usage of the algorithms, as is usual when defining the notion of "in-place", so that any univariate algorithm can be modified to fit this description. We are deliberately non-committal about the choice of univariate algorithms, since any algorithms that solve the univariate problems may be used. One may, of course, take these algorithms to be the corresponding algorithm of Chin with the modifications described in Section 2.2, including basis conversion to ensure that input and output polynomials are written on the Newton basis. In particular, it is this combination of algorithms that is used to prove Theorem 4.

We prove that Algorithm 1 is correct in Lemma 8. Combining the lemma with Lemma 3 then establishes the correctness of Algorithm 2, since the algorithm simply reverses the steps of the Algorithm 1, inverting each evaluation along the way.

Algorithm 1. Multivariate Hermite evaluation

Input: A finite initial segment $I \subseteq \mathbb{N}^n$; and the vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$ for some polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$.
Output: The vector $(f_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$ equal to $(E(F, \boldsymbol{j}))_{\boldsymbol{j}\in I}$.
1: **for** $\ell = 1, \ldots, n$ **do**
2: **for** $\boldsymbol{k} = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$ **do**
3: Call an in-place univariate Hermite evaluation algorithm on the vector $\left(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)}\right)_{i_\ell\in\mu_\ell(I,\boldsymbol{k})}$.
4: **end for**
5: **end for**

Algorithm 2. Multivariate Hermite interpolation

Input: A finite initial segment $I \subseteq \mathbb{N}^n$; and the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j}\in I} = (f_{\boldsymbol{j}})_{\boldsymbol{j}\in I}$ for some polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$.
Output: The vector $(f_{\boldsymbol{j}})_{\boldsymbol{j}\in I}$ equal to $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$.
1: **for** $\ell = n, n-1, \ldots, 1$ **do**
2: **for** $\boldsymbol{k} = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$ **do**
3: Call an in-place univariate Hermite interpolation algorithm on the vector $\left(f_{(j_1,\ldots,j_{\ell-1},j_\ell,i_{\ell+1},\ldots,i_n)}\right)_{j_\ell\in\mu_\ell(I,\boldsymbol{k})}$.
4: **end for**
5: **end for**

Lemma 8. *Algorithm 1 is correct.*

Proof. We prove the lemma by induction on n. If $n = 1$, then Algorithm 1 simply calls the univariate algorithm on the input. Accordingly, correctness holds trivially for univariate inputs. It is illustrative to consider the case $n = 2$ separately before proceeding by induction. Therefore, suppose that Algorithm 1 is called on a finite initial segment $I \subseteq \mathbb{N}^2$ and the vector $F \dashv (N_{(i_1,i_2)})_{(i_1,i_2)\in I} = (f_{(i_1,i_2)})_{(i_1,i_2)\in I}$ for some $F \in \mathbb{F}_q[X_1, X_2]_I$. Then the first iteration of the outer loop of the algorithm calls the univariate algorithm on each of the vectors

$$\left(f_{(i_1,i_2)}\right)_{i_1\in\mu_1(I,i_2)} = \left(f_{(i_1,i_2)}\right)_{i_1\in\lambda(I,i_2)} \quad \text{for } i_2 \in \pi_1(I) = \rho(I, 0).$$

Here, $(f_{(i_1,i_2)})_{i_1\in\lambda(I,i_2)}$ is equal to the coefficient vector $F_{i_2} \dashv (N_{i_1})_{i_1\in\lambda(I,i_2)}$. Therefore, after the first iteration of the outer loop has been performed, the input vector $(f_{(i_1,i_2)})_{(i_1,i_2)\in I}$ is equal to $(E(F_{i_2}, j_1))_{(j_1,i_2)\in I}$. It follows that the second iteration of the outer loop calls the univariate algorithm on each of the vectors

$$\left(f_{(j_1,i_2)}\right)_{i_2\in\mu_2(I,j_1)} = (E(F_{i_2}, j_1))_{i_2\in\rho(I,j_1)} \quad \text{for } j_1 \in \pi_2(I) = \lambda(I, 0).$$

Thus, Lemma 7 implies that after the second iteration of the outer loop has been performed, we have $(f_{(j_1,i_2)})_{i_2\in\rho(I,j_1)} = (E(F, (j_1, j_2)))_{j_2\in\rho(I,j_1)}$ for $j_1 \in \lambda(I, 0)$. As this is the last iteration of the loop, it follows that the input vector is equal to $(E(F, (j_1, j_2)))_{(j_1,j_2)\in I}$ at the end of the algorithm. Hence, Algorithm 1 is correct for the inputs I and $F \dashv (N_{(i_1,i_2)})_{(i_1,i_2)\in I}$, and the lemma holds for $n = 2$.

Suppose now that $n \geq 3$ and Algorithm 1 is correct for all inputs on $n-1$ variables. Furthermore, suppose that Algorithm 1 is called on a finite initial segment $I \subseteq \mathbb{N}^n$ and the vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$ for some polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_I$. Then the subvectors $(f_{(i_1,\ldots,i_{n-1},i_n)})_{(i_1,\ldots,i_{n-1})\in\lambda(I,i_n)}$ for $i_n \in \rho(I, \mathbf{0}_{n-1})$ are modified independently of one another during the first $n-1$ iterations of the outer loop of the algorithm. Indeed, during the first $n-1$ iterations of the outer loop, the univariate algorithm is only ever called on a subvector of one of these subvectors. For $\ell \in \{1, \ldots, n-1\}$, the family of sets

$$\{(j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_{n-1}, i_n) \mid (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_{n-1}) \in \pi_\ell(\lambda(I, i_n))\}$$

for $i_n \in \rho(I, \mathbf{0}_{n-1})$ form a partition of $\pi_\ell(I)$. Moreover, for $i_n \in \rho(I, \mathbf{0}_{n-1})$ and $\boldsymbol{k}' = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_{n-1}) \in \pi_\ell(\lambda(I, i_n))$, we have

$$\mu_\ell(I, (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_{n-1}, i_n)) = \mu_\ell(\lambda(I, i_n), \boldsymbol{k}').$$

Thus, performing the first $n-1$ iterations of the outer loop is equivalent to recursively calling the algorithm on the initial segment $\lambda(I, i_n) \subseteq \mathbb{N}^{n-1}$ and the subvector $(f_{(i_1,\ldots,i_{n-1},i_n)})_{(i_1,\ldots,i_{n-1})\in\lambda(I,i_n)}$ for each $i_n \in \rho(I, \mathbf{0}_{n-1})$. Initially, we have

$$\left(f_{(i_1,\ldots,i_{n-1},i_n)}\right)_{(i_1,\ldots,i_{n-1})\in\lambda(I,i_n)} = F_{i_n} \dashv \left(N_{(i_1,\ldots,i_{n-1})}\right)_{(i_1,\ldots,i_{n-1})\in\lambda(I,i_n)}$$

for $i_n \in \rho(I, \mathbf{0}_{n-1})$. Therefore, the induction hypothesis implies that after $n-1$ iterations of the outer loop have been performed, the input vector is equal to $(E(F_{i_n}, (j_1, \ldots, j_{n-1})))_{(j_1,\ldots,j_{n-1},i_n)\in I}$. It follows that the last iteration of the outer loop calls the univariate algorithm on each of the vectors

$$\left(f_{(j_1,\ldots,j_{n-1},i_n)}\right)_{i_n\in\mu_n(I,\boldsymbol{k})} = (E(F_{i_n}, (j_1, \ldots, j_{n-1})))_{i_n\in\rho(I,(j_1,\ldots,j_{n-1}))}$$

for $\boldsymbol{k} = (j_1, \ldots, j_{n-1}) \in \pi_n(I) = \lambda(I, 0)$. Hence, Lemma 7 implies that Algorithm 1 returns the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j}\in I}$. That is, the algorithm is correct for the inputs I and $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I}$. Thus, the lemma follows by induction. □

It is clear that the complexity of each multivariate algorithm is determined by the complexity of the corresponding univariate algorithm. We capture the nature of this dependency in the next two lemmas.

Lemma 9. *Suppose that for some function* $\mathsf{E} : \mathbb{N} \setminus \{0\} \to \mathbb{R}$ *the univariate Hermite evaluation algorithm used in Algorithm 1 performs at most* $\mathsf{E}(s)$ *operations in* $\mathbb{F}_q$ *when given the initial segment* $[s]$ *as an input, and that* $\mathsf{E}(s)/s$ *is a nondecreasing function of* s*. Then given an input such that* $I \subseteq [s_1] \times \cdots \times [s_n]$ *for positive integers* $s_1, \ldots, s_n$*, Algorithm 1 performs at most*

$$\left(\frac{\mathsf{E}(s_1)}{s_1} + \cdots + \frac{\mathsf{E}(s_n)}{s_n}\right)|I|$$

operations in $\mathbb{F}_q$*.*

Proof. If the univariate Hermite evaluation algorithm has complexity given by such a function $\mathsf{E} : \mathbb{N} \to \mathbb{R}$, then Algorithm 1 performs at most

$$\sum_{\ell=1}^{n} \sum_{\boldsymbol{k}\in\pi_\ell(I)} \mathsf{E}(|\mu_\ell(I, \boldsymbol{k})|) = \sum_{\ell=1}^{n} \sum_{\boldsymbol{k}\in\pi_\ell(I)} \frac{\mathsf{E}(|\mu_\ell(I, \boldsymbol{k})|)}{|\mu_\ell(I, \boldsymbol{k})|} |\mu_\ell(I, \boldsymbol{k})|$$

operations in $\mathbb{F}_q$. It follows that if $I \subseteq [s_1] \times \cdots \times [s_n]$ for positive integers $s_1, \ldots, s_n$, and thus $\mu_\ell(I, \boldsymbol{k}) \subseteq [s_\ell]$ for $\ell \in \{1, \ldots, n\}$ and $\boldsymbol{k} \in \pi_\ell(I)$, then the algorithm performs at most

$$\sum_{\ell=1}^{n} \frac{\mathsf{E}(s_\ell)}{s_\ell} \sum_{\boldsymbol{k} \in \pi_\ell(I)} |\mu_\ell(I, \boldsymbol{k})| = \sum_{\ell=1}^{n} \frac{\mathsf{E}(s_\ell)}{s_\ell} |I|$$

operations in $\mathbb{F}_q$. □

Lemma 10. *Suppose that for some function $\mathsf{I} : \mathbb{N} \setminus \{0\} \to \mathbb{R}$ the univariate Hermite interpolation algorithm used in Algorithm 2 performs at most $\mathsf{I}(s)$ operations in $\mathbb{F}_q$ when given the initial segment $[s]$ as an input, and that $\mathsf{I}(s)/s$ is a nondecreasing function of s. Then given an input such that $I \subseteq [s_1] \times \cdots \times [s_n]$ for positive integers $s_1, \ldots, s_n$, Algorithm 2 performs at most*

$$\left(\frac{\mathsf{I}(s_1)}{s_1} + \cdots + \frac{\mathsf{I}(s_n)}{s_n} \right) |I|$$

operations in $\mathbb{F}_q$.

We omit the proof of Lemma 10 since it uses identical arguments to those of Lemma 9. Combining the two lemmas with Lemmas 5 and 6 then completes the proof of Theorem 4. We note that Lemmas 9 and 10, and thus Theorem 4, do not account for the cost of computing the sets $\pi_\ell(I)$ and $\mu_\ell(I, \boldsymbol{k})$ during the algorithm. For the initial segments that are used in the encoding algorithms, we have simple explicit formulae that allow the sets to be computed with low complexity. The general problem is not considered here.

As for the complexities of the multivariate algorithms, their space requirements are largely determined by those of the univariate algorithms. The amount of auxiliary space used by either multivariate algorithm, i.e., storage in addition to the input array, is equal to that of the index manipulations plus the maximum amount of auxiliary space used by the corresponding univariate algorithm over all calls to it. Therefore, if the univariate algorithm is a true in-place algorithm, in the sense that it uses only $\mathcal{O}(1)$ auxiliary space, and the index manipulations also require only $\mathcal{O}(1)$ auxiliary space, then the multivariate algorithm enjoys the same auxiliary space bound.

3. Encoding algorithm for low-rate codes

In this section, we present the first of our fast systematic encoding algorithms for multiplicity codes. Although, the algorithm is suitable for multiplicity codes of all rates, we somewhat falsely refer to it as an encoding algorithm for low-rate codes since the encoding algorithm of Section 4 is faster for codes with sufficiently high rates. Recall that our goal is to efficiently evaluate the encoding function enc_d^s defined in Section 1.2. The algorithm of this section achieves this goal by using the fast Hermite interpolation and evaluation algorithms of Section 2 to successively evaluate its constituent maps $\mathrm{ev}_{\mathcal{I}_{d,n}}^{-1}$ and ev_d^s.

We use the map $\kappa_n : [q]^n \times \mathbb{N}^n \to \mathbb{N}^n$, given by $(\boldsymbol{i}, \boldsymbol{s}) \mapsto \boldsymbol{i} + \boldsymbol{s}q$, to translate the encoding problem into the language of Section 2. To this end, we let $I_{d,n}$ denote the κ_n-image of the information set $\mathcal{I}_{d,n}$ defined in Theorem 1. Then we have

$$I_{d,n} = \{\boldsymbol{i} \in \mathbb{N}^n \mid |\boldsymbol{i}| \leq d\} \quad \text{for } d \in \mathbb{N}.$$

For notational convenience, we extend this definition to $d \in \mathbb{Z}$, by defining $I_{d,n}$ to be the empty set for $d < 0$. For nonzero $s \in \mathbb{N}$, we define $C_{s,n} = \kappa_n([q]^n \times S_{s,n})$. Finally, for $d, s \in \mathbb{Z}$ such that $d < sq$ and $s > 0$, we define $R_{d,s,n} = C_{s,n} \setminus I_{d,n}$. With this notation, a message of a multiplicity code Mult_d^s is written as a vector $m = (m_{\boldsymbol{j}})_{\boldsymbol{j} \in I_{d,n}}$. Its systematic encoding is then equal to

$$\mathrm{enc}_d^s(m) = \Big((E(F, \boldsymbol{j} + \boldsymbol{t}q))_{\boldsymbol{t} \in S_{s,n}} \Big)_{\boldsymbol{j} \in [q]^n}$$

where F is the unique polynomial in $\mathbb{F}_q[\boldsymbol{X}]_d$ such that $E(F, \boldsymbol{j}) = m_{\boldsymbol{j}}$ for $\boldsymbol{j} \in I_{d,n}$. It follows that it is sufficient to consider the problem of computing the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j} \in C_{s,n}}$ when given m. In fact, we need only compute $(E(F, \boldsymbol{j}))_{\boldsymbol{j} \in R_{d,s,n}}$ since the remaining entries are present in the message m to begin with.

For $d, s \in \mathbb{N}$ such that $d < sq$, we have $\mathbb{F}_q[\boldsymbol{X}]_d = \mathbb{F}_q[\boldsymbol{X}]_{I_{d,n}} \subset \mathbb{F}_q[\boldsymbol{X}]_{C_{s,n}}$. Therefore, as noted in Section 2.1, computing the polynomial F that corresponds to a message m of the multiplicity code Mult_d^s (i.e., computing $\mathrm{ev}_{\mathcal{I}_{d,n}}^{-1}(m)$) is an instance of the Hermite interpolation problem with initial segment $I = I_{d,n}$, while computing the vector $(E(F, \boldsymbol{j}))_{\boldsymbol{j} \in C_{s,n}}$ (i.e., computing *the entries of* $\mathrm{ev}_d^s(F)$) is an instance of Hermite evaluation problem with initial segment $I = C_{s,n}$. Applying the algorithms of Section 2 to these instances of the interpolation and evaluation problems yields our first systematic encoding algorithm, presented in Algorithm 3, and the complexity bound of Theorem 11.

Algorithm 3. Systematic encoding for low-rate codes

Input: Nonnegative integers d and s such that $d < sq$; and a vector $(c_{\boldsymbol{j}})_{\boldsymbol{j} \in C_{s,n}}$ such that $m = (c_{\boldsymbol{j}})_{\boldsymbol{j} \in I_{d,n}}$ is a message of Mult_d^s, and $c_{\boldsymbol{j}} = 0$ for $\boldsymbol{j} \in R_{d,s,n}$.

Output: The vector $(c_{\boldsymbol{j}})_{\boldsymbol{j} \in C_{s,n}}$ equal to $(E(F, \boldsymbol{j}))_{\boldsymbol{j} \in C_{s,n}}$ for the polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_d$ that corresponds to the message m.

1: Call Algorithm 2 on $I_{d,n}$ and $(c_{\boldsymbol{j}})_{\boldsymbol{j} \in I_{d,n}}$.
2: Call Algorithm 1 on $C_{s,n}$ and $(c_{\boldsymbol{j}})_{\boldsymbol{j} \in C_{s,n}}$.

Theorem 11. *Given a message vector m of a multiplicity code Mult_d^s, its systematic encoding $\mathrm{enc}_d^s(m)$ can be computed in*

$$\mathcal{O}\left(\frac{\mathsf{M}(sq) \log sq}{sq} |C_{s,n}| n \right)$$

operations in $\mathbb{F}_q$.

Proof. Taking the univariate algorithms used by Algorithms 1 and 2 to be the corresponding algorithms of Chin, as modified in Section 2.2, Theorem 4 implies that Algorithm 3 performs

$$\mathcal{O}\left(\frac{\mathsf{M}(d+1) \log(d+1)}{d+1} |I_{d,n}| n + \frac{\mathsf{M}(sq) \log sq}{sq} |C_{s,n}| n \right) \tag{5}$$

operations in $\mathbb{F}_q$. As the parameters d and s satisfy the inequality $d < sq$, and thus $I_{d,n} \subset C_{s,n}$, the second term of the bound dominates. □

By taking $\mathsf{M}(k)$ to be in $\mathcal{O}(k \log k \log\log k)$, it follows from Theorem 11 that systematic encoding for Mult_d^s can be performed in $\mathcal{O}(|C_{s,n}| n \log^2(sq) \log\log(sq))$ operations in $\mathbb{F}_q$, matching the quasi-linear bound stated in the introduction. While

we have only bounded the number of field operations performed by the encoding algorithm, the cost of the index manipulations performed by Algorithms 1 and 2 during encoding is low in practice. Indeed, for $\ell \in \{1, \ldots, n\}$, we have the simple explicit formulae $\pi_\ell(I_{d,n}) = I_{d,n-1}$ and $\mu_\ell(I_{d,n}, \boldsymbol{k}) = I_{d-|\boldsymbol{k}|,1}$ for $\boldsymbol{k} \in I_{d,n-1}$. Similarly, $\pi_\ell(C_{s,n}) = C_{s,n-1}$ and $\mu_\ell(C_{s,n}, \boldsymbol{k}) = C_{s-|\boldsymbol{k} \operatorname{div} q|,1}$ for $\boldsymbol{k} \in C_{s,n-1}$.

While the encoding algorithm has quasi-optimal asymptotic complexity, it is clear that it performs more operations than is necessary. This excess is most apparent in the evaluation step of the algorithm, which recomputes the entries of original input message. We address this problem in the next section by describing some modifications to the encoding algorithm that can be used to eliminate unnecessary computations. Each modification requires modifications to be made to one or both of the univariate algorithms. As we are being non-committal about our choice of these algorithms, we only describe how the behaviour of univariate algorithms should be changed, rather than describing how to obtain the desired behaviour.

3.1. **Practical improvements.** Our first modification occurs at the interface of Algorithms 1 and 2. Suppose that, as occurs for the algorithms of Section 2.2, the univariate interpolation algorithm performs monomial to Newton basis conversion as its last step, and the univariate evaluation algorithm performs the inverse conversion as its first step. Then the conversions performed during the last iteration of the interpolation algorithm cancel with those performed during the first iteration of the evaluation algorithm. Consequently, these basis conversions can be avoided altogether, saving $\Omega(|I_{d,n}|)$ operations.

For our second improvement, we modify the evaluation step of the encoding algorithm to take advantage of the fact that the polynomial being evaluated has support contained in $I_{d,n}$, a proper, and possibly much smaller, subset of the initial segment $I = C_{s,n}$ for which we apply Algorithm 1. In the univariate case, we need only modify the algorithm to take advantage of the fact that the polynomial has degree at most d rather than at most $sq - 1$. And it is straightforward to modify the algorithm of Section 2.2 accordingly. The following lemma allows us to extend the modified univariate algorithm to the multivariate case.

Lemma 12. *If the inputs of Algorithm 1 satisfy $I \supset I_{d,n}$ and $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \setminus I_{d,n}$, for some $d \in \mathbb{N}$, then at the beginning of the ℓth iteration of the outer loop of the algorithm, for $\ell \in \{1, \ldots, n\}$, we have $f_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i} = (i_1, \ldots, i_n) \in I$ such that $i_\ell + \cdots + i_n > d$.*

Proof. We prove the lemma by induction on ℓ. The statement holds trivially for the first iteration. Therefore, suppose that at the beginning of the ℓth iteration of the outer loop, for some $1 \leq \ell < n$, we have $f_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i} = (i_1, \ldots, i_n) \in I$ such that $i_\ell + \cdots + i_n > d$. Then for all $\boldsymbol{k} = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$ such that $i_{\ell+1} + \cdots + i_n > d$, the subvector $(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)})_{i_\ell \in \mu_\ell(I,\boldsymbol{j})}$ contains all zeros and is consequently unchanged by the call to the univariate algorithm. As the sets

$$\{(j_1, \ldots, j_{\ell-1}, i_\ell, i_{\ell+1}, \ldots, i_n) \mid i_\ell \in \mu_\ell(I, \boldsymbol{k})\}$$

for $\boldsymbol{k} = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$ form a partition of I, it follows that at the beginning of the next iteration we have $f_{\boldsymbol{i}} = 0$ for all $\boldsymbol{i} = (i_1, \ldots, i_n) \in I$ such that $i_{\ell+1} + \cdots + i_n > d$. □

It follows from Lemma 12 that during the evaluation step of the encoding algorithm, the inner loop of Algorithm 1 need only be performed for $\boldsymbol{k}$ such that $i_{\ell+1}+\cdots+i_n \le d$. Moreover, for each such $\boldsymbol{k}$, the univariate algorithm evaluates a polynomial of degree at most $d-i_{\ell+1}-\cdots-i_n$. Consequently, it is straightforward to extend the modified univariate algorithm to the multivariate case. To give some indication of the number of operations saved by this modification, we observe that the number of zero entries described by Lemma 12 over all iterations is equal to

$$|R_{d,s,n}| + \sum_{\ell=2}^{n} \sum_{\boldsymbol{i}\in R_{d,s,n-\ell+1}} \left|C_{s-|\boldsymbol{i}\ \mathrm{div}\ q|,\ell-1}\right|.$$

Hence, the modification saves the most operations when the rate $|I_{d,n}|/|C_{s,n}|$ of the code is low.

For the final modification, we stop the evaluation step of the encoding algorithm from recomputing the input message, saving $\Omega(|I_{d,n}|)$ operations. These entries of the input and output are indexed by the information set $I_{d,n}$. Consequently, during the last iteration of the outer loop of Algorithm 1, where we call the univariate algorithm on $(f_{(j_1,\ldots,j_{n-1},i_n)})_{i_n\in C_{s-|\boldsymbol{k}\ \mathrm{div}\ q|,n}}$ for each $\boldsymbol{k}=(j_1,\ldots,j_{n-1})\in C_{s,n-1}$, the univariate algorithm need only return correct values in those entries with $i_n > d-j_1-\cdots-j_{n-1}$. The entries indexed by $R_{d,s,n}$ will then be correct at the end of the algorithm, while the remaining entries will contain meaningless values. Therefore, if the modification can be implement for the univariate case, then it readily extends to the multivariate case.

4. Encoding algorithm for high-rate codes

Let $m=(m_{\boldsymbol{j}})_{\boldsymbol{j}\in I_{d,n}}$ be a message vector of a multiplicity code Mult_d^s. Then Lemma 3 implies that there exists a unique polynomial $F_C\in\mathbb{F}_q[\boldsymbol{X}]_{C_{s,n}}$ such that

$$E(F_C,\boldsymbol{j}) = \begin{cases} m_{\boldsymbol{j}} & \text{if } \boldsymbol{j}\in I_{d,n},\\ 0 & \text{if } \boldsymbol{j}\in R_{d,s,n}.\end{cases}$$

Let $F_C \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}}$, and define polynomials $F_I=\sum_{\boldsymbol{i}\in I_{d,n}} f_{\boldsymbol{i}}N_{\boldsymbol{i}}$ and $F_R=-\sum_{\boldsymbol{i}\in R_{d,s,n}} f_{\boldsymbol{i}}N_{\boldsymbol{i}}$. Then $F_C=F_I-F_R$. As $I_{d,n}$ is an initial segment that is disjoint with $R_{d,s,n}$, we have $\boldsymbol{j}\not\succeq\boldsymbol{i}$ for $\boldsymbol{j}\in I_{d,n}$ and $\boldsymbol{i}\in R_{d,s,n}$. Thus, Lemma 2 implies that

$$E(F_I,\boldsymbol{j}) = E(F_C,\boldsymbol{j}) + E(F_R,\boldsymbol{j}) = \begin{cases} m_{\boldsymbol{j}} & \text{if } \boldsymbol{j}\in I_{d,n},\\ E(F_R,\boldsymbol{j}) & \text{if } \boldsymbol{j}\in R_{d,s,n}.\end{cases}$$

It follows that $F_I\in\mathbb{F}_q[\boldsymbol{X}]_{I_{d,n}}=\mathbb{F}_q[\boldsymbol{X}]_d$ is the polynomial that corresponds to the message m. Moreover, as $E(F_I,\boldsymbol{j})$ and $E(F_R,\boldsymbol{j})$ agree for $\boldsymbol{j}\in R_{d,s,n}$, the polynomial F_R may be used to compute the unknown entries of the systematic encoding $\mathrm{ev}_d^s(F_I)$ of m. In this section, we show that when the rate $|I_{d,n}|/|C_{s,n}|$ of Mult_d^s is sufficiently close to one, so that F_R has much fewer nonzero coefficients on the Newton basis than F_I, the unknown entries of the systematic encoding can be computed more efficiently by using F_R in place of F_I. When this gain is sufficient to compensate for the extra cost of computing F_R (for which we first compute F_C), when compared to that of directly computing F_I, we also gain an advantage over the encoding algorithm of Section 3.

To compute the values $E(F_R, \boldsymbol{j})$ for $\boldsymbol{j} \in R_{d,s,n}$, we use Algorithm 1 with $I = C_{s,n}$ as our starting point. Then following an approach similar to that used in Section 3.1, we eliminate unnecessary operations from the algorithm by taking advantage of the fact that $F_R \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i} \in C_{s,n}}$ has zeros in those entries with indices in $I_{d,n}$. Once again we find that the multivariate case follows readily from the univariate case. Consequently, we begin this section by considering the univariate problem.

4.1. **Univariate algorithm.** Let s be a positive integer and $F \in \mathbb{F}_q[X]$. Then the definition of the Hasse derivative implies that

$$F(X + \alpha_j) \bmod X^s = \sum_{t=0}^{s-1} H(F,t)(\alpha_j) X^t = \sum_{t=0}^{s-1} E(F, j+tq) X^t \quad \text{for } j \in [q].$$

Thus, computing the values $E(F,j)$ for $j \in C_{s,1}$ is equivalent to computing the polynomials $F(X+\alpha_j) \bmod X^s$ on the monomial basis for $j \in [q]$. Suppose now that for some nonnegative integer $d < sq-1$, the polynomial F is of the form $\sum_{i \in R_{d,s,1}} f_i N_i$ with coefficients $f_i \in \mathbb{F}_q$. Then N_{d+1} divides F, allowing us to reduce the problem of computing the values $E(F,j)$ for $j \in C_{s,1}$ to the lower degree problems of computing $E(N_{d+1}, j)$ and $E(F/N_{d+1}, j)$ for $j \in C_{s,1}$, with polynomial multiplications modulo X^s to combine the results. We improve upon this approach by replacing N_{d+1} by its largest factor that is invariant under Taylor shifts.

Let $r = \lfloor (d+1)/q \rfloor$. Then N_{rq} divides F since $rq \le d+1$, and

$$N_{rq} = \prod_{j=0}^{rq-1} (X - \alpha_{j \bmod q}) = \prod_{j=0}^{q-1} (X - \alpha_j)^r = (X^q - X)^r. \tag{6}$$

Therefore, if we let $Q = F/N_{rq} \in \mathbb{F}_q[X]$, then

$$F(X+\alpha_j) \bmod X^s = X^r \big((X^{q-1}-1)^r Q(X+\alpha_j) \bmod X^{s-r} \big) \quad \text{for } j \in [q].$$

Hence, we can compute the values $E(F,j)$ for $j \in [sq] \setminus [rq] \supseteq R_{d,s,1}$ by first computing the polynomials $Q(X+\alpha_j) \bmod X^{s-r}$ on the monomial basis for $j \in [q]$, for which we can use Chin's Hermite evaluation algorithm, then multiplying each shifted polynomial by $(X^{q-1}-1)^r \bmod X^{s-r}$. Applying this approach, we obtain Algorithm 4. We allow the input d of the algorithm to be negative, in which case $R_{d,s,1} = C_{s,1}$, in order to simplify the description of the multivariate algorithm in the next section.

Lemma 13. *Algorithm 4 performs $\mathcal{O}(\mathsf{M}((s-r)q)\log((s-r)q))$ operations in $\mathbb{F}_q$, where $r = \max(\lfloor (d+1)/q \rfloor, 0)$.*

Proof. Equation (6) implies that $N_{rq+i} = N_{rq} N_i$ for $i, r \in \mathbb{N}$. Consequently, in Line 2 of the algorithm, the coefficient vector of Q on the Newton basis can be read directly from the coefficient vector of F. As $\deg Q < (s-r)q$, Lemma 6 therefore implies that Line 2 performs $\mathcal{O}(\mathsf{M}((s-r)q)\log((s-r)q))$ operations in $\mathbb{F}_q$. Similarly, Lemma 5 implies that Line 3 performs $\mathcal{O}(\mathsf{M}((s-r)q)\log((s-r)q))$ operations in $\mathbb{F}_q$. Finally, Lines 4–6 perform q multiplications of polynomials with degree less than $s-r$, requiring at most $q\,\mathsf{M}(s-r) \le \mathsf{M}((s-r)q)$ operations in $\mathbb{F}_q$. Hence, Algorithm 4 performs $\mathcal{O}(\mathsf{M}((s-r)q)\log((s-r)q))$ operations in $\mathbb{F}_q$. □

We have included the polynomial $(X^{q-1}-1)^r \bmod X^{s-r}$ as an input to Algorithm 4 for the benefit of the multivariate algorithm of the next section, which is able to reuse these inputs for multiple calls to the algorithm. For an instance of the

Algorithm 4. Univariate Hermite evaluation over $R_{d,s,1}$

Input: Integers d and s such that $d+1 < sq$ and $s > 0$; the vector $F \dashv (N_i)_{i\in R_{d,s,1}} = (f_i)_{i\in R_{d,s,1}}$ for some $F \in \mathbb{F}_q[X]$ of the form $\sum_{i\in R_{d,s,1}} f_i N_i$; and the polynomial $(X^{q-1}-1)^r \bmod X^{s-r}$ for $r = \max(\lfloor (d+1)/q \rfloor, 0)$, written on the monomial basis.

Output: The vector $(f_i)_{i\in R_{d,s,1}}$ equal to $(E(F,j))_{j\in R_{d,s,1}}$.

1: Set $r = \max(\lfloor (d+1)/q \rfloor, 0)$.
2: Use the algorithm of Gerhard [12] to convert $Q = F/N_{rq}$ to the monomial basis.
3: Use the Hermite evaluation algorithm of Chin [8] with the modifications described in Section 2.2 to compute the coefficients of the polynomials $Q(X+\alpha_j) \bmod X^{s-r}$ for $j \in [q]$ on the monomial basis.
4: **for** $j \in [q]$ **do**
5: Compute $((X^{q-1}-1)^r \bmod X^{s-r})(Q(X+\alpha_j) \bmod X^{s-r})$ and set f_{j+tq} equal to the coefficient of X^{t-r} in the resulting polynomial for $t = \max(\lceil (d+1-j)/q \rceil, 0), \ldots, s-1$.
6: **end for**

univariate problem, this input can be computed in $\mathcal{O}(\mathsf{M}(\lceil (s-r)/(q-1) \rceil) \log r)$ operations in $\mathbb{F}_q$ by the square and multiply algorithm for exponentiation. Alternatively, one can use the binomial theorem and Lucas' lemma [19, p. 230] (see also [11]).

4.2. **Multivariate algorithm.** Recall that the polynomial F_R defined in Section 4 has support on the monomial basis that is contained in $C_{s,n}$, while its coefficient vector $F_R \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}}$ has zeros in those entries with indices in $I_{d,n}$ for some $d < sq$. The following lemma implies that if Algorithm 1 is called on $C_{s,n}$ and the coefficient vector of F_R, then the zeros in the entries indexed by $I_{d,n}$ persist throughout the algorithm.

Lemma 14. *If the inputs of Algorithm 1 satisfy $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \cap I_{d,n}$, for some $d \in \mathbb{N}$, then at the beginning of the ℓth iteration of the outer loop of the algorithm, for $\ell \in \{1, \ldots, n\}$, we have $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \cap I_{d,n}$.*

Proof. We prove the lemma by induction on ℓ. The statement holds trivially for the first iteration. Therefore, suppose that for some $d \in \mathbb{N}$ and $\ell \in \{1, \ldots, n-1\}$ we have $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \cap I_{d,n}$ at the beginning of the ℓth iteration of the outer loop. Let $\boldsymbol{k} = (j_1, \ldots, j_{\ell-1}, i_{\ell+1}, \ldots, i_n) \in \pi_\ell(I)$. Then $f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)} = 0$ for $i_\ell \le d - |\boldsymbol{k}|$, and Lemma 2 implies that

$$E\left(\sum_{i_\ell \in \mu_\ell(I,\boldsymbol{k})} f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)} N_{i_\ell}, j_\ell\right) = E\left(\sum_{i_\ell=0}^{j_\ell} f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)} N_{i_\ell}, j_\ell\right)$$

for $j_\ell \in \mu_\ell(I,\boldsymbol{k})$. Thus, the entries of $(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)})_{i_\ell\in\mu_\ell(I,\boldsymbol{k})}$ with $i_\ell \le d - |\boldsymbol{k}|$ are still zero after the univariate Hermite evaluation algorithm has been called on the vector in Line 3. Hence, $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I \cap I_{d,n}$ at the end of the ℓth iteration of the outer loop. □

Let $d, s \in \mathbb{N}$ such that $d < sq$, and $F \in \mathbb{F}_q[\boldsymbol{X}]_{C_{s,n}}$ such that its coefficient vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}}$ satisfies $f_{\boldsymbol{i}} = 0$ for $\boldsymbol{i} \in I_{d,n}$. Then it follows from

Lemma 14 that if Algorithm 1 is called on $C_{s,n}$ and $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in C_{s,n}}$, then each time Line 3 of the algorithm is executed, the vector

$$\big(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)}\big)_{i_\ell\in\mu_\ell(C_{s,n},\boldsymbol{k})} = \big(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)}\big)_{i_\ell\in C_{s-|\boldsymbol{k}\text{ div }q|,1}}$$

has zeros in those entries with $i_\ell \in I_{d-|\boldsymbol{k}|,1}$. We can take advantage of these zero entries by modifying Line 3 so that Algorithm 4 is called on the vector $(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)})_{i_\ell\in R_{d-|\boldsymbol{k}|,s-|\boldsymbol{k}\text{ div }q|,1}}$. This modification requires that Algorithm 4 is provided with the polynomial

$$(X^{q-1}-1)^r \bmod X^{s-|\boldsymbol{k}\text{ div }q|-r} \quad \text{for } r = \max(\lfloor (d-|\boldsymbol{k}|+1)/q\rfloor, 0).$$

Therefore, along with the modification to Line 3 of the algorithm, we can introduce a precomputation step to the algorithm where the polynomials

$$(X^{q-1}-1)^r \bmod X^{s-r} \quad \text{for } r = 0,\ldots,\lfloor (d+1)/q\rfloor$$

are computed on the monomial basis. Then each call to Algorithm 4 only requires that one of these polynomials be trivially reduced modulo some power of X. By making these modifications to Algorithm 1, we obtain Algorithm 5.

Algorithm 5. Multivariate Hermite evaluation over $R_{d,s,n}$

Input: Integers d and s such that $0 \le d < sq$; and the vector $F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in R_{d,s,n}} = (f_{\boldsymbol{i}})_{\boldsymbol{i}\in R_{d,s,n}}$ for some $F \in \mathbb{F}_q[\boldsymbol{X}]$ of the form $\sum_{\boldsymbol{i}\in R_{d,s,n}} f_{\boldsymbol{i}} N_{\boldsymbol{i}}$ with $n \ge 2$.
Output: The vector $(f_{\boldsymbol{i}})_{\boldsymbol{i}\in R_{d,s,n}}$ equal to $(E(F,\boldsymbol{j}))_{\boldsymbol{j}\in R_{d,s,n}}$.
1: Set $U_0 = 1$.
2: **for** $r = 1,\ldots,\lfloor (d+1)/q\rfloor$ **do**
3: Compute $U_r = (X^{q-1}-1)U_{r-1} \bmod X^{s-r}$ on the monomial basis.
4: **end for**
5: **for** $\ell = 1,\ldots,n$ **do**
6: **for** $\boldsymbol{k} = (j_1,\ldots,j_{\ell-1},i_{\ell+1},\ldots,i_n) \in C_{s,n-1}$ **do**
7: If $R_{d-|\boldsymbol{k}|,s-|\boldsymbol{k}\text{ div }q|,1}$ is nonempty (which fails to hold if and only if $d = sq-1$ and $\boldsymbol{k} = \boldsymbol{s}q$ for some $\boldsymbol{s} \in S_{s,n-1}$), then call Algorithm 4 on the integers $d' = d-|\boldsymbol{k}|$ and $s' = s-|\boldsymbol{k}\text{ div }q|$, the vector $(f_{(j_1,\ldots,j_{\ell-1},i_\ell,i_{\ell+1},\ldots,i_n)})_{i_\ell\in R_{d',s',1}}$, and the polynomial $U_r \bmod X^{s'-r}$ for $r = \max(\lfloor (d'+1)/q\rfloor, 0)$.
8: **end for**
9: **end for**

We streamline notation during the complexity analysis of Algorithm 5 by defining $\Delta_{d,s} = (s-\max(\lfloor (d+1)/q\rfloor,0))q$ for $d,s \in \mathbb{Z}$, $\mathsf{M}^*(0) = 0$ and $\mathsf{M}^*(k) = \mathsf{M}(k)\log k$ for nonzero $k \in \mathbb{N}$. Then Lemma 13 implies that Algorithm 4 performs $\mathcal{O}(\mathsf{M}^*(\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k}\text{ div }q|}))$ operations in $\mathbb{F}_q$ during Line 7 of Algorithm 5. The following lemma is used to bound the total number of operations performed by Algorithm 4 over each iteration of the main loop.

Lemma 15. *For $n \ge 2$ and $d,s \in \mathbb{Z}$ such that $d < sq$ and $s > 0$,*

$$\sum_{\boldsymbol{k}\in C_{s,n-1}} \Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k}\text{ div }q|} \le \left(1 + \frac{\max(\lfloor d/q\rfloor+1,0)}{s}\right)|R_{d,s,n}|. \tag{7}$$

Proof. We prove the lemma by induction on n. Suppose that $d, s \in \mathbb{Z}$ such that $d < sq$ and $s > 0$. Then for all integers $\ell \geq 2$, we have

$$R_{d,s,\ell} = \bigcup_{i_1 \in C_{s,1}} \big\{(i_1, i_2, \ldots, i_\ell) \mid (i_2, \ldots, i_\ell) \in R_{d-i_1, s-(i_1 \text{ div } q), \ell-1}\big\}. \tag{8}$$

It follows that

$$\begin{aligned}\sum_{k \in C_{s,1}} \Delta_{d-k, s-(k \text{ div } q)} &= \sum_{k \in C_{s,1}} \big|R_{d-k, s-(k \text{ div } q), 1}\big| + (\max(d-k+1, 0) \bmod q) \\ &= |R_{d,s,2}| + \sum_{k=0}^{d} (d-k+1 \bmod q).\end{aligned}$$

Therefore, the lemma is true for $n = 2$ since

$$\begin{aligned}\sum_{k=0}^{d} (d-k+1 \bmod q) &\leq \max\left(\left\lfloor \frac{d}{q} \right\rfloor + 1, 0\right) \sum_{k=0}^{q-1} (d-k+1 \bmod q) \\ &= \max\left(\left\lfloor \frac{d}{q} \right\rfloor + 1, 0\right) \binom{q}{2}\end{aligned}$$

and

$$|R_{d,s,2}| \geq |R_{sq-1,s,2}| = \binom{s+1}{2} q^2 - \binom{sq+1}{2} = s\binom{q}{2}.$$

Suppose now that $n \geq 3$ and that the lemma is true for all smaller values of n. Let $d, s \in \mathbb{Z}$ such that $d < sq$ and $s > 0$. Then

$$\sum_{\boldsymbol{k} \in C_{s,n-1}} \Delta_{d-|\boldsymbol{k}|, s-|\boldsymbol{k} \text{ div } q|} = \sum_{k \in C_{s,1}} \sum_{\boldsymbol{k} \in C_{s-(k \text{ div } q), n-2}} \Delta_{d-k-|\boldsymbol{k}|, s-(k \text{ div } q)-|\boldsymbol{k} \text{ div } q|}. \tag{9}$$

For $k \in C_{s,1}$, we have $d - k \leq d - (k \text{ div } q)q < (s - (k \text{ div } q))q$ and $s - (k \text{ div } q) > 0$. Thus, the induction hypothesis implies that for each $k \in C_{s,1}$, the inner sum on the right-hand side of (9) is at most

$$\left(1 + \frac{\max(\lfloor (d-k)/q \rfloor + 1, 0)}{s - \lfloor k/q \rfloor}\right) \big|R_{d-k, s-(k \text{ div } q), n-1}\big|.$$

Here, the first factor is always less than or equal to $1 + \max(\lfloor d/q \rfloor + 1, 0)/s$. Therefore, combining and substituting these upper bounds into (9) yields the inequality

$$\sum_{\boldsymbol{k} \in C_{s,n-1}} \Delta_{d-|\boldsymbol{k}|, s-|\boldsymbol{k} \text{ div } q|} \leq \left(1 + \frac{\max(\lfloor d/q \rfloor + 1, 0)}{s}\right) \sum_{k \in C_{s,1}} \big|R_{d-k, s-(k \text{ div } q), n-1}\big|.$$

Equation (8) with $\ell = n$ implies that the sum on the right-hand side of this inequality is equal to $|R_{d,s,n}|$. Hence, (7) holds and the lemma follows by induction. □

Lemma 16. *Algorithm 5 performs*

$$\mathcal{O}\left(\left(1 + \frac{\lfloor d/q \rfloor + 1}{s}\right) \frac{\mathsf{M}(\delta) \log \delta}{\delta} |R_{d,s,n}| n + \left\lceil \frac{s-1}{q-1} \right\rceil \left\lfloor \frac{d+1}{q} \right\rfloor\right) \tag{10}$$

operations in $\mathbb{F}_q$, where $\delta = sq - \max(d + 1 - n(q-1), 0)$.

Proof. Each polynomial U_r is of the form $V(X^{q-1})$ for some polynomial $V \in \mathbb{F}_q[X]$ of degree less than $\lceil (s-r)/(q-1) \rceil$. Thus, Lines 2–4 of the algorithm perform at most $\lceil (s-1)/(q-1) \rceil \lfloor (d+1)/q \rfloor$ operations in $\mathbb{F}_q$. As the polynomials U_r are written on the monomial basis, no operations in $\mathbb{F}_q$ are performed in order to

compute their residues in Line 7. Consequently, Lemma 13 implies that Line 7 performs $\mathcal{O}(\mathsf{M}^*(\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k} \operatorname{div} q|}))$ operations in $\mathbb{F}_q$. Hence, Algorithm 5 performs

$$\mathcal{O}\left(\left\lceil\frac{s-1}{q-1}\right\rceil\left\lfloor\frac{d+1}{q}\right\rfloor + n\sum_{\boldsymbol{k}\in C_{s,n-1}}\mathsf{M}^*\big(\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k} \operatorname{div} q|}\big)\right) \tag{11}$$

operations in $\mathbb{F}_q$. As $\mathsf{M}(k)/k$ is a nondecreasing function of k, so too is $\mathsf{M}^*(k)/k$. Moreover, for $\boldsymbol{k} \in C_{s,n-1}$,

$$\begin{aligned}\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k} \operatorname{div} q|} &= (s - |\boldsymbol{k} \operatorname{div} q|)q - \max(\lfloor (d - |\boldsymbol{k}| + 1)/q \rfloor q, 0)\\ &\le (s - |\boldsymbol{k} \operatorname{div} q|)q - \max(d - |\boldsymbol{k}| + 1 - (q-1), 0)\\ &\le sq - \max(d - |\boldsymbol{k} \bmod q| + 1 - (q-1), 0)\\ &\le sq - \max(d + 1 - n(q-1), 0).\end{aligned}$$

Letting $\delta = sq - \max(d + 1 - n(q-1), 0)$, it follows that

$$\sum_{\boldsymbol{k}\in C_{s,n-1}}\mathsf{M}^*\big(\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k} \operatorname{div} q|}\big) \le \frac{\mathsf{M}^*(\delta)}{\delta}\sum_{\boldsymbol{k}\in C_{s,n-1}}\Delta_{d-|\boldsymbol{k}|,s-|\boldsymbol{k} \operatorname{div} q|}.$$

Combining this inequality with (7) and (11) completes the proof. □

The factor $1 + (\lfloor d/q \rfloor + 1)/s$ of the first term of the complexity bound (10) measures the penalty that results from the complexity of Algorithm 4 being a function of $\Delta_{d,s}$ rather than a function of $|R_{d,s,1}|$. The former may be larger by a factor of $q - 1$, while the penalty incurred by Algorithm 5 is limited to a factor of 2. We have made no attempt to optimise the size of this factor, which would require strengthening the bound of Lemma 15. For $d, s \in \mathbb{N}$ such that $d < sq$, $R_{d,s,n}$ contains the vectors $\boldsymbol{j} + \boldsymbol{t}q$ for all $(\boldsymbol{j}, \boldsymbol{t}) \in [q]^n \times \mathbb{N}^n$ such that $|\boldsymbol{j}| \ge q$ and $|\boldsymbol{t}| = s - 1$. Thus, we obtain the crude lower bound

$$|R_{d,s,n}| \ge \binom{n-1+s-1}{n-1}\left(q^n - \binom{n+q-1}{n}\right).$$

From this bound, it is readily deduce that the value δ defined in Lemma 16 is $\mathcal{O}(|R_{d,s,n}|)$ for $n \ge 2$. Similarly, the second term of (10) is $\mathcal{O}(|R_{d,s,n}|)$ for $n \ge 3$. Thus, if $\mathsf{M}(k)$ is taken to be in $\mathcal{O}(k \log k \log\log k)$, then Algorithm 5 performs $\tilde{\mathcal{O}}(|R_{d,s,n}|)$ operations in $\mathbb{F}_q$ for $n \ge 3$. For $n = 2$, the second term of (10) is only guaranteed to be $\mathcal{O}(|C_{s,2}|)$, which is all that is required for fast encoding. Recall that the second term of (10) counts the cost of Lines 2–4 of Algorithm 5, which may be performed as a precomputation in many settings. With this precomputation and fast polynomial arithmetic, Algorithm 5 attains quasi-linear complexity for $n = 2$.

4.3. **Encoding algorithm.** The systematic encoding algorithm for high-rate multiplicity codes is presented in Algorithm 6. The algorithm follows the approach described in Section 4: first, the extended interpolation problem is solved in order to recover the polynomial F_C, after which F_R is deduced and used to compute the non-message entries of the encoding. Taking the univariate algorithm used by Algorithm 2 to be Chin's interpolation algorithm, as modified in Section 2.2, it follows from Theorem 4 that Line 1 of the algorithm performs

$$\mathcal{O}\left(\frac{\mathsf{M}(sq)\log sq}{sq}|C_{s,n}|n\right)$$

operations in $\mathbb{F}_q$. Line 2 then performs $|R_{d,s,n}|$ (cheap) multiplications by -1. Lines 4 and 5 perform $\mathcal{O}(\mathsf{M}(sq)\log sq)$ operations in $\mathbb{F}_q$ if $n = 1$ (see Section 4.1), while Lemma 16 bounds the number of operations performed by Line 7 if $n \geq 2$. Combining the bounds provides a second proof of Theorem 11.

Algorithm 6. Systematic encoding for high-rate codes

Input: Nonnegative integers d and s such that $d < sq$; and a vector $(c_{\boldsymbol{j}})_{\boldsymbol{j}\in C_{s,n}}$ such that $m = (c_{\boldsymbol{j}})_{\boldsymbol{j}\in I_{d,n}}$ is a message of Mult_d^s, and $c_{\boldsymbol{j}} = 0$ for $\boldsymbol{j} \in R_{d,s,n}$.

Output: The vector $(c_{\boldsymbol{j}})_{\boldsymbol{j}\in C_{s,n}}$ such that

$$(c_{\boldsymbol{i}})_{\boldsymbol{i}\in I_{d,n}} = F \dashv (N_{\boldsymbol{i}})_{\boldsymbol{i}\in I_{d,n}} \quad \text{and} \quad (c_{\boldsymbol{j}})_{\boldsymbol{j}\in R_{d,s,n}} = (E(F,\boldsymbol{j}))_{\boldsymbol{j}\in R_{d,s,n}}$$

for the polynomial $F \in \mathbb{F}_q[\boldsymbol{X}]_d$ that corresponds to the message m.

1: Call Algorithm 2 on $C_{s,n}$ and $(c_{\boldsymbol{j}})_{\boldsymbol{j}\in C_{s,n}}$.
2: Set $c_{\boldsymbol{j}}$ equal to $-c_{\boldsymbol{j}}$ for each $\boldsymbol{j} \in R_{d,s,n}$.
3: **if** $n = 1$ **then**
4: Compute $U = (X^{q-1} - 1)^r \bmod X^{s-r}$ for $r = \lfloor (d+1)/q \rfloor$.
5: Call Algorithm 4 on d, s, $(c_j)_{j\in R_{d,s,1}}$ and U.
6: **else**
7: Call Algorithm 5 on d, s and $(c_{\boldsymbol{j}})_{\boldsymbol{j}\in R_{d,s,n}}$.
8: **end if**

5. Conclusion

We presented two quasi-linear time systematic encoding algorithms for multiplicity codes which provide complimentary performance in practical settings. Of the two algorithms, the one that provides the shortest encoding time for a given set of parameters will vary with the choice of univariate algorithms. Moreover, their encoding times depend on additional factors besides the number of field operations they perform. Consequently, we cannot draw solid conclusions about the relative performance of the two algorithms in a given practical setting by comparing their stated complexities (which would also require estimating hidden constants). However, as the encoding algorithms share the same underlying subroutines, implementing both algorithms on a particular architecture should not require much more effort than implementing just one of the algorithms, and would allow for direct comparisons to be made.

In Section 3.1, we described modifications to the encoding algorithm for low-rate codes which were aimed at improving its practical performance by eliminating some unnecessary operations. Similar modifications may also be made to the encoding algorithm for high-rate codes, but we omit details here. The algorithm for high-rate codes would also benefit from improving or replacing Algorithm 4 so that Line 7 of Algorithm 5 can always be performed in $\tilde{\mathcal{O}}(\left|R_{d-|\boldsymbol{k}|,s-|\boldsymbol{k}\ \mathrm{div}\ q|,1}\right|)$ operations and without the need to provide the additional polynomial input.

The Hermite interpolation and evaluation algorithms of Section 2 may be of independent interest. The dependency of the multivariate algorithms on the univariate algorithms draws our attention to the problem of optimising the choice of univariate algorithms. In this direction, it would be interesting to develop fast univariate Hermite interpolation and evaluation algorithms that work natively on the Newton basis, which would allow us to avoid the basis conversions that are necessary when

using the algorithms discussed in Section 2.2. Encouragingly, such algorithms already exist for instances that do not involve derivatives [5, Section 5.1]. It would also be interesting to further investigate the benefits offered by the techniques of van der Hoeven [13] when using the algorithms of Section 2.2.

Acknowledgements

The author would like to thank Daniel Augot and Françoise Levy-dit-Vehel for their helpful comments on this and earlier versions of the paper, and Joris van der Hoeven for bringing reference [13] to the author's attention.

References

1. A. V. Aho, K. Steiglitz, and J. D. Ullman, *Evaluating polynomials at fixed sets of points*, SIAM J. Comput. **4** (1975), no. 4, 533–539.
2. Daniel Augot, Françoise Levy-dit-Vehel, and Cuong M. Ngô, *Information sets of multiplicity codes*, Proceedings of the 2015 IEEE International Symposium on Information Theory (ISIT), IEEE, June 2015, pp. 2401–2405.
3. Daniel Augot, Françoise Levy-dit-Vehel, and Abdullatif Shikfa, *A storage-efficient and robust private information retrieval scheme allowing few servers*, Cryptology and network security, Lecture Notes in Comput. Sci., vol. 8813, Springer, Cham, 2014, pp. 222–239.
4. Dario Bini and Victor Y. Pan, *Polynomial and matrix computations. Vol. 1*, Progress in Theoretical Computer Science, Birkhäuser Boston, Inc., Boston, MA, 1994, Fundamental algorithms.
5. Alin Bostan and Éric Schost, *Polynomial evaluation and interpolation on special sets of points*, J. Complexity **21** (2005), no. 4, 420–446.
6. Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi, *Algebraic complexity theory*, Grundlehren der Mathematischen Wissenschaften, vol. 315, Springer-Verlag, Berlin, 1997.
7. David G. Cantor and Erich Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Inform. **28** (1991), no. 7, 693–701.
8. Francis Y. Chin, *A generalized asymptotic upper bound on fast polynomial evaluation and interpolation*, SIAM J. Comput. **5** (1976), no. 4, 682–690.
9. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan, *Private information retrieval*, J. ACM **45** (1998), no. 6, 965–982.
10. Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan, *Extensions to the method of multiplicities, with applications to Kakeya sets and mergers*, SIAM J. Comput. **42** (2013), no. 6, 2305–2328.
11. N. J. Fine, *Binomial coefficients modulo a prime*, Amer. Math. Monthly **54** (1947), 589–592.
12. Jürgen Gerhard, *Modular algorithms for polynomial basis conversion and greatest factorial factorization*, Proceedings of the Seventh Rhine Workshop on Computer Algebra (T Mulders, ed.), RWCA'00, 2000, pp. 125–141.
13. Joris van der Hoeven, *Faster Chinese remaindering*, Tech. report, HAL, 2016, `http://hal.archives-ouvertes.fr/hal-01403810`.
14. Jonathan Katz and Luca Trevisan, *On the efficiency of local decoding procedures for error-correcting codes*, Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, ACM, New York, 2000, pp. 80–86.
15. J. D. Key, T. P. McDonough, and V. C. Mavron, *Information sets and partial permutation decoding for codes from finite geometries*, Finite Fields Appl. **12** (2006), no. 2, 232–247.
16. Swastik Kopparty, *List-decoding of multiplicity codes*, The Electronic Colloquium on Computational Complexity (ECCC) **TR12-044** (2012).
17. Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin, *High-rate codes with sublinear-time decoding*, STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing, ACM, New York, 2011, pp. 167–176.
18. Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin, *High-rate codes with sublinear-time decoding*, J. ACM **61** (2014), no. 5, Art. 28, 1–20.
19. Edouard Lucas, *Théorie des Fonctions Numériques Simplement Périodiques. [Continued]*, Amer. J. Math. **1** (1878), no. 3, 197–240.

20. Vadim Olshevsky and Amin Shokrollahi, *Matrix-vector product for confluent Cauchy-like matrices with application to confluent rational interpolation*, Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, ACM, New York, 2000, pp. 573–581.
21. Victor Y. Pan, *Simple multivariate polynomial multiplication*, J. Symbolic Comput. **18** (1994), no. 3, 183–186.
22. Victor Y. Pan, *Structured matrices and polynomials: unified superfast algorithms*, Birkhäuser Boston, Inc., Boston, MA; Springer-Verlag, New York, 2001.
23. Joris van der Hoeven and Éric Schost, *Multi-point evaluation in higher dimensions*, Appl. Algebra Engrg. Comm. Comput. **24** (2013), no. 1, 37–52.
24. Joachim von zur Gathen, *Functional decomposition of polynomials: the tame case*, J. Symbolic Comput. **9** (1990), no. 3, 281–299.
25. Joachim von zur Gathen and Jürgen Gerhard, *Fast algorithms for Taylor shifts and certain difference equations*, Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ACM, New York, 1997, pp. 40–47.