Sub-quadratic Decoding of One-point Hermitian Codes

Johan Sebastian Rosenkilde Nielsen, Peter Beelen

Sub-quadratic Decoding of One-point Hermitian Codes

Johan S. R. Nielsen, Peter Beelen

Abstract

We present the first two sub-quadratic complexity decoding algorithms for one-point Hermitian codes. The first is based on a fast realisation of the Guruswami–Sudan algorithm by using state-of-the-art algorithms from computer algebra for polynomial-ring matrix minimisation. The second is a Power decoding algorithm: an extension of classical key equation decoding which gives a probabilistic decoding algorithm up to the Sudan radius. We show how the resulting key equations can be solved by the matrix minimisation algorithms from computer algebra, yielding similar asymptotic complexities.

Index Terms

Hermitian codes, AG codes, list decoding, Guruswami–Sudan, Power decoding

I. Introduction

In this article we examine fast decoding of one-point Hermitian codes beyond half the minimum distance. First we give a new algorithm for constructing the interpolation polynomial in Guruswami–Sudan decoding. Our approach is closely related to the interpolation algorithm proposed by Lee and O'Sullivan [1], where a satisfactory interpolation polynomial is found as a minimal element in a certain Gröbner basis. In [2] Beelen and Brander reformulated the interpolation problem in terms of matrices with coefficients in $\mathbb{F}_{q^2}[x]$. The advantage of this reformulation is that the interpolation problem then reduces to solving a module minimisation problem, i.e., finding a minimal weighted-degree vector in the $\mathbb{F}_{q^2}[x]$-row space of a certain explicit matrix. The Gröbner basis algorithm in this reformulation then is replaced by a weighted row reduction algorithm. Beelen and Brander [2] improved in this way the complexity of finding the interpolation polynomial given in [1] by applying Alekhnovich's row reduction algorithm [3]. For one-point Hermitian codes they obtained a decoding algorithm with quadratic complexity in the length of the code.

Instead of using Alekhnovich's row reduction algorithm, we propose to apply the row reduction algorithm by Giorgi, Jeannerod and Villard (GJV) [4]. It turns out that a straightforward application of this algorithm on the explicit matrix given in [2] does not improve complexity. However, using a different embedding than in [2] to reformulate the interpolation problem in terms of matrices with coefficients in $\mathbb{F}_{q^2}[x]$, we do find an improvement. The result is a sub-quadratic time algorithm to find the interpolation polynomial. By describing a fast way to deal with the so-called root-finding step (based on the theory of power series and the root-finding algorithm in [3]), this results in a sub-quadratic realization of the Guruswami–Sudan algorithm for one-point Hermitian codes: $O^{\sim}(n^{(2+\omega)/3}\ell^{\omega}s)$, where s and ℓ are the multiplicity and list size parameters of Guruswami–Sudan, and $\omega \leq 3$ is the exponent for matrix multiplication. Here and later, $O^{\sim}$ denotes O with $\log$-factors omitted.

Next we give a new derivation of Power decoding of one-point Hermitian codes, inspired by Gao decoding for Reed–Solomon codes [5], and show how to solve the resulting generalised key equation system in a fast way. This gives rise to a second sub-quadratic complexity decoding algorithm: $O^{\sim}(n^{(2+\omega)/3}\ell^{\omega})$, where ℓ is the "powering" parameter.

The methodology employed here applies equally well to the classical syndrome key equation of one-point Hermitian codes used in [6] for decoding up to half the minimum distance minus half the genus. Our results therefore puts that approach into a simple and well-studied computational framework yielding several algorithms with better complexity than in [6].

The article is organised as follows: In Section II, the necessary background is given on one-point Hermitian codes as well as on solving the Lagrange interpolation problem over the Hermitian function field. In Section III, module minimisation is explained, which will form the core behind the fast decoding methods described later in the article. In Section III-A, an essential ingredient is presented, namely the embedding that will be used to reformulate the interpolation step in the decoding of one-point Hermitian codes to a module minimisation problem.

In Section IV, module minimisation is applied to the interpolation step in the Guruswami–Sudan list decoding algorithm for one-point Hermitian codes and a sub-quadratic algorithm is obtained in this way. By improving existing methods to deal with the root-finding part of the Guruswami–Sudan list decoding algorithm, this leads to a complete, sub-quadratic decoding algorithm. We first give an introduction to the Guruswami–Sudan list decoding algorithm. Subsequently, in Section IV-A, the interpolation step in this algorithm is reformulated as a module minimisation problem and the techniques from Section III are applied to solve this problem in sub-quadratic time. Then in Section IV-B, the root-finding problem is discussed.

J. S. R. Nielsen is with the GRACE Project, INRIA Saclay & LIX, École Polytechnique, France (e-mail: jsrn@jsrn.dk). P. Beelen is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark (e-mail: pabe@dtu.dk). This is a preprint of work that has been published in IEEE Transactions of Information Theory. DOI: 10.1109/TIT.2015.2424415.

Another decoding algorithm is described in Section V. "Powered key equations" are given in Section V-A, while again the module minimisation techniques from Section III are applied to solve them in Section V-B, leading to a sub-quadratic "power decoding" algorithm.

We have implemented the decoding algorithms in Sage v6.4 [7] and present some simulation results in Section VI: we discuss the failure probability of either decoding method, as well as the speed of the algorithm on concrete parameters.

We finish the main part of the article with some concluding remarks in Section VII. Both in the root finding step in the Guruswami–Sudan algorithm as in an important division step in the power decoding algorithm, we need some technical machinery involving power series as well as some other technical results. These are explained in the appendices.

II. One-point Hermitian codes

Let q be some prime power, and consider the curve $\mathcal{H}$ over the field $\mathbb{F}_{q^2}$ defined by the following polynomial in X, Y:

$$\mathcal{H}(X,Y) = Y^q + Y - X^{q+1}.$$

$\mathcal{H}$ is the Hermitian curve, and it is absolutely irreducible. Let $F = \mathbb{F}_{q^2}(x,y)$ be the algebraic function field with full constant field $\mathbb{F}_{q^2}$ achieved by extending $\mathbb{F}_{q^2}(x)$ with a variable y satisfying the relation $\mathcal{H}(x,y) = 0$. For any divisor D, we denote by $\mathcal{L}(D)$ the Riemann–Roch space associated to D.

There are certain basic facts about F which we will need. They can be found in for example [8].

Proposition 1: The function field F has genus $g = \frac{1}{2}q(q-1)$ and q^3+1 rational places, which we will denote $\mathcal{P} = \{P_1, \ldots, P_{q^3}, P_\infty\}$. The place P_∞ denotes "the place at infinity" being the only rational place occurring as a pole of either x or y (in fact it is a pole of both). The place P_∞ is totally ramified in the extension $\mathbb{F}_{q^2}(x,y)/\mathbb{F}_{q^2}(x)$ of function fields and hence has ramification index q in this extension. Furthermore define

$$Я = \bigcup_{i=0}^{\infty} \mathcal{L}(iP_\infty).$$

Then $Я = \mathbb{F}_{q^2}[x,y]$.

Let $\mathcal{P}^\star = \mathcal{P} \setminus \{P_\infty\}$. By a slight abuse of notation, we can identify elements of $\mathcal{P}^\star$ with pairs $(\alpha,\beta) \in \mathbb{F}_{q^2}^2$. For any α, let $B_\alpha \subset \mathbb{F}_{q^2}$ be the set of β such that $(\alpha,\beta) \in \mathcal{P}^\star$. Then $|B_\alpha| = q$ for all α. Furthermore, we have $\mathrm{div}(x-\alpha) = \sum_{\beta \in B_\alpha} (\alpha,\beta) - qP_\infty$.

The fact that $Я = \mathbb{F}_{q^2}[x,y]$ is extremely helpful since all these functions can then be described by polynomials. For brevity, we define for any divisor D the convenient notation

$$\mathcal{L}(D + \infty P_\infty) = \bigcup_{i \in \mathbb{Z}} \mathcal{L}(D + iP_\infty).$$

Note that for instance $Я = \mathcal{L}(\infty P_\infty)$.

For a function $f \in Я$ expressed as polynomials, we can therefore reduce its y-degree to less than q using the relation $\mathcal{H}(x,y) = 0$ from which it follows that $\{x^i y^j \mid 0 \le j < q\}$ is a basis for $Я$. We will refer to this as the "standard basis" of $Я$, and usually represent its elements using this. However, for certain auxiliary calculations we will convert into other representations; the details of these calculations are given in Appendix B.

We will measure elements of $Я$ by their pole order at P_∞; when elements in $Я$ are in the standard basis, this takes on a particularly simple form:

Definition 2: Let the order function $\deg_{\mathcal{H}} : Я \mapsto \mathbb{N}_0 \cup \{-\infty\}$ be given as $\deg_{\mathcal{H}}(p) = -v_{P_\infty}(p)$ for $p \neq 0$ and $\deg_{\mathcal{H}}(0) = -\infty$, where $v_P(\cdot)$ is the valuation of a function at the place P. For a monomial $x^i y^j$, this is also given by

$$\deg_{\mathcal{H}}(x^i y^j) = \deg_{q,q+1}(x^i y^j) = qi + (q+1)j,$$

when $j < q$, and then extended to polynomials of y degree less than q by the maximal of the monomials' $\deg_{\mathcal{H}}$.

Note that all monomials $x^i y^j$ with $j < q$ have different $\deg_{\mathcal{H}}$. Therefore, $\deg_{\mathcal{H}}$ induces a term ordering $\le_{\mathcal{H}}$ on $\mathbb{F}_{q^2}[x,y]$ such that $x^{i_1}y^{j_1} \le_{\mathcal{H}} x^{i_2}y^{i_2}$ if and only if $\deg_{\mathcal{H}}(x^{i_1}y^{j_1}) \le \deg_{\mathcal{H}}(x^{i_2}y^{j_2})$. This means that we can speak of the leading monomial, $\mathrm{LM}_{\mathcal{H}}(\cdot)$, and the leading coefficient, $\mathrm{LC}_{\mathcal{H}}(\cdot)$, for elements of $Я$.

We will also need two easy technical lemmas; the first is straightforward but a proof can be found e.g. in [9, Proposition 2.2].

Lemma 3: For any non-zero $h \in F$ it holds that

$$\mathcal{L}(-\mathrm{div}(h) + \infty P_\infty) = hЯ. \tag{1}$$

Lemma 4: For any $m \in \mathbb{Z}_+$, there are at least $m - g$ distinct monomials of the form $x^i y^j$, $j < q$ such that $\deg_{\mathcal{H}}(x^i y^j) < m$.

Proof: The statement translates simply to dim $\mathcal{L}((m-1)P_\infty) \geq m-g$, which is exactly Riemann's Theorem, see e.g. [10, Theorem 1.4.17]. ∎

Let us now formally introduce the class of codes we wish to decode.

Definition 5: Let $n = q^3$ and m be an integer satisfying $2g-2 < m < n$. Then the corresponding one-point Hermitian code over $\mathbb{F}_{q^2}$ is defined as

$$\mathcal{C} = \left\{ \big(f(P_1), \ldots, f(P_n)\big) \mid f \in \mathcal{L}(mP_\infty) \right\}.$$

Note that $\mathcal{L}(mP_\infty) \subset \mathcal{R}$, so all the f we need to evalute to obtain $\mathcal{C}$ are polynomials in x and y satisfying $\deg_{\mathcal{H}} f \leq m$.

The basic parameters of these codes are completely known. First of all from [10, Theorem 2.2.2] it follows that in the context of Definition 5, $\mathcal{C}$ is an $[n,\ k,\ d]$ code where

$$k = m - g + 1 \qquad \text{and} \qquad d \geq d^\star \triangleq n - m.$$

In fact, the *exact* minimum distance is known: Stichtenoth showed that it is exactly $d^\star$ as above whenever $2g \leq m \leq n - q^2$ [8], while the remaining cases were determined by Yang and Kumar and shown to be slightly better for some values of m [11].

As a last tool before we begin, we will also need Lagrangian interpolation over the evaluation points of a considered one-point Hermitian code, i.e. given $\gamma_{\alpha,\beta} \in \mathbb{F}_{q^2}$ for every $(\alpha,\beta) \in \mathcal{P}^\star$ then find some $p \in \mathcal{R}$ such that $p(\alpha,\beta) = \gamma_{\alpha,\beta}$ for all (α,β). It is easy to see such a function must exist: for each place, the requirement specifies a linear equation in the coefficients of p seen as an element of $\mathbb{F}_{q^2}[x,y]$, so by Lemma 4 there must exist one with $\deg_{\mathcal{H}}$ less than $n+g+1$. Since it is slow to solve a linear system of equations, it is beneficial to have a closed formula though this might yield a function of slightly suboptimal $\deg_{\mathcal{H}}$. The following lemma is inspired by a similar result from [1], though the complexity analysis is new.

Lemma 6: Given $\gamma_{\alpha,\beta} \in \mathbb{F}_{q^2}$ for all $(\alpha,\beta) \in \mathcal{P}^\star$ the function

$$p = \sum_{\alpha \in \mathbb{F}_{q^2}} \prod_{\alpha' \in \mathbb{F}_{q^2} \setminus \{\alpha\}} \frac{x-\alpha'}{\alpha-\alpha'} \sum_{\beta \in B_\alpha} \left(\gamma_{\alpha,\beta} \prod_{\beta' \in B_\alpha \setminus \{\beta\}} \frac{y-\beta'}{\beta-\beta'} \right)$$

satisfies $p(\alpha,\beta) = \gamma_{\alpha,\beta}$ for $(\alpha,\beta) \in \mathcal{P}^\star$ and $\deg_{\mathcal{H}} p < n + 2g$. Furthermore, given the $\gamma_{\alpha,\beta}$ we can compute p in time $O\tilde{}(n)$.

Proof: Clearly, the given $p \in \mathcal{R}$, and first statement is easy to see. For the $\deg_{\mathcal{H}}$, clearly $\deg_x p \leq q^2-1$ and $\deg_y p \leq q-1$ and so $\deg_{\mathcal{H}}(p) \leq q(q^2-1) + (q+1)(q-1)$.

For the complexity, we use standard Divide & Conquer tricks. Denote by $L[B, \boldsymbol{\eta}](y)$ the $\mathbb{F}_{q^2}[y]$ Lagrange interpolation polynomial such that $L[B,\boldsymbol{\eta}](\beta) = \eta_\beta$ for all $\beta \in B$. Note that we have $L[B,\boldsymbol{\eta}] = \sum_{\beta \in B} \left(\eta_\beta \prod_{\beta' \in B \setminus \{\beta\}} \frac{y-\beta'}{\beta-\beta'} \right)$. Let $\tilde{\boldsymbol{\gamma}}_\alpha = \left(\gamma_{\alpha,\beta} / \prod_{\alpha' \in \mathbb{F}_{q^2} \setminus \{\alpha\}} (\alpha - \alpha') \right)_{\beta \in B_\alpha}$ for each $\alpha \in \mathbb{F}_{q^2}$. Let $A = \mathbb{F}_{q^2}$ and consider a subdivision into two disjoint sets A_1 and A_2. Then

$$\begin{aligned} p &= \sum_{\alpha \in A} \prod_{\alpha' \in \mathbb{F}_{q^2} \setminus \{\alpha\}} (x-\alpha')\ L[B_\alpha, \tilde{\boldsymbol{\gamma}}_\alpha](y) \\ &= \sum_{K=1,2} \prod_{\alpha \in A \setminus A_K} (x-\alpha) \\ &\qquad \left(\sum_{\alpha \in A_K} \prod_{\alpha' \in A_K \setminus \{\alpha\}} (x-\alpha')\ L[B_\alpha, \tilde{\boldsymbol{\gamma}}_\alpha](y) \right), \end{aligned}$$

Now the inner parenthesis is a recursive $\mathcal{R}$ Lagrange interpolation problem with half as many points. If we denote by $T(t)$ the cost of solving this problem with qt points having t different x-coordinates, we get the recursive equation for $t > 1$ that $T(t) = 2T(t/2) + qO\tilde{}(t/2)$: to collect the two recursive $\mathcal{R}$ functions we must perform $2q$ multiplications in $\mathbb{F}_{q^2}[x]$ with operands of degree at most $t/2$, followed by q sums. This has the solution $T(t) = O\tilde{}(qt) + O\tilde{}(t)T(1)$, where $T(1)$ then consists of computing a single $L[B_\alpha, \hat{\boldsymbol{\gamma}}]$ for some α and $\hat{\boldsymbol{\gamma}}$. This can be done in cost $O\tilde{}(q)$ since $|B_\alpha| = q$. The constants $\prod_{\alpha' \in \mathbb{F}_{q^2} \setminus \{\alpha\}} (\alpha - \alpha')$ for the $\tilde{\boldsymbol{\gamma}}_\alpha$ can be precomputed using Divide & Conquer methods in time $O\tilde{}(q^2)$. ∎

III. Module Minimisation

In both our algorithms, we will need to find "small" elements in certain free $\mathbb{F}_{q^2}[x]$-modules, given a basis of the module. We will solve this by representing the basis as a square $\mathbb{F}_{q^2}[x]$ matrix and then bring it to a certain standard form; the resulting matrix will still represent a basis of our module, and its rows will represent "small" elements. As a measure for being "small" we will use the quantity

$$\deg \boldsymbol{v} = \max_i \{\deg v_i\},$$

with $\boldsymbol{v} = (v_1, \ldots, v_\rho) \in \mathbb{F}_{q^2}[x]^\rho$. In this section, we will describe this process from the point where a basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\rho\}$ of an $\mathbb{F}_{q^2}[x]$-module $\mathcal{V}$ is given, in a manner completely detached from the coding theoretic setting. We will restrict ourselves to the

Complexity for computing a weak Popov form of $V \in \mathbb{F}_{q^2}[x]^{\rho\times\rho}$

Algorithm	Field operations in big-$O^\sim$
Mulders–Storjohann [14]	$\rho^2 \deg V \Delta(V)$
Alekhnovich [3]	$\rho^\omega \Delta(V)$
GJV [4] or Zhou–Labahn [15]	$\rho^\omega \deg V$

Table I
WE USE ω FOR THE EXPONENT FOR MULTIPLICATION OF $\mathbb{F}_{q^2}$ MATRICES, I.E. $\omega \leq 3$. WE ASSUME THAT $\rho < \deg V$.

case that the $\boldsymbol{v}_i$ can be represented as $\mathbb{F}_{q^2}[x]$ vectors of length ρ. Let $V \in \mathbb{F}_{q^2}[x]^{\rho\times\rho}$ be the matrix whose rows are the $\boldsymbol{v}_i$. By slight abuse of language we will sometimes also call V a basis of $\mathcal{V}$.

By "leading position", or $\text{LP}(\boldsymbol{v})$ for some $\boldsymbol{v} \in \mathbb{F}_{q^2}[x]^\rho$, we mean the right-most position i such that $\deg v_i = \deg \boldsymbol{v}$. The problem we are going to solve is the following:

Problem 7: Let $I \subseteq \{1, \ldots, \rho\}$ and let $\mathcal{V}_I$ be all vectors of $\mathcal{V}$ with leading position in I. Find then a vector $\boldsymbol{v} \in \mathcal{V}_I$ with minimal degree.

For the Guruswami–Sudan interpolation, we will set $I = \{1, \ldots, \rho\}$ and will just seek any vector of minimal degree, while for Power decoding, I will be only the first few indices.

Definition 8: A matrix $U \in \mathbb{F}_{q^2}[x]^{\rho\times\rho}$ is in *weak Popov form* if the leading position of all its rows are different.

Note that the weak Popov form is not canonical for a given matrix. The following well-known result describes why the definition is so useful:

Proposition 9: Let $U \in \mathbb{F}[x]^{\rho\times\rho}$ be a basis in weak Popov form of a module $\mathcal{V}$. Any non-zero $\boldsymbol{b} \in \mathcal{V}$ satisfies $\deg \boldsymbol{u} \leq \deg \boldsymbol{b}$, where $\boldsymbol{u}$ is the row of U with $\text{LP}(\boldsymbol{u}) = \text{LP}(\boldsymbol{b})$.

A proof can be found in e.g. [12].

Using elementary row operations, we may change V into a matrix U without changing the row space of the matrices. The matrices U and V are unimodular equivalent, that is to say that there exists $M \in \mathbb{F}_{q^2}[x]^{\rho\times\rho}$ with $\det M \in \mathbb{F}_{q^2}^*$ such that $U = MV$. Clearly then, if we can compute from V a unimodular equivalent matrix U, which is in weak Popov form, then by the above proposition we have solved our problem for any index set I. This computation is known as module minimisation, $\mathbb{F}_{q^2}[x]$-lattice basis reduction or row reduction[1]. It is well-known that the weak Popov form is also a Gröbner basis of the module for a specific monomial ordering, see e.g. [13, Section 2.1.2].

There are a number of algorithms from the literature for carrying out this computation. Principally, their running time depends on $\deg V = \max_i\{\deg \boldsymbol{v}_i\}$ where V is the input matrix. It was shown in [13, Chapter 2] how two algorithms, Mulders–Storjohann's [14] and Alekhnovich's [3], rather depend on the *orthogonality defect*:

$$\Delta(V) = \text{rowdeg}\, V - \deg\det V \leq \rho \deg V,$$

with $\text{rowdeg}\, V = \sum_{i=1}^{\rho} \deg \boldsymbol{v}_i$ and $\deg\det V$ the degree of the determinant of the matrix $V \in \mathbb{F}_{q^2}[x]^{\rho\times\rho}$. Table I summarises the complexities for module minimisation using various known algorithms. It should be noted that the two algorithms GJV [4] and Zhou–Labahn [15] compute *order bases* of $\mathbb{F}_{q^2}[x]$ matrices; it was described in [4] how to use an order basis computation to compute a row reduced form, and in [16] how to quickly compute the weak Popov form from a row reduced one. The asymptotic complexities are as reported for the entire sequence of algorithms.

A. Handling Weights

For application to the decoding algorithms we present later in the article, Problem 7 is not formulated quite general enough: rather, we will be seeking a vector of $\mathcal{V}$ whose *weighted* degree is minimal, and this weighting takes a rather general form: let $\nu \in \mathbb{Z}_+$ and $\boldsymbol{w} \in \mathbb{N}_0^\rho$, then the $(\nu, \boldsymbol{w})$-weighted degree of some $\boldsymbol{v} \in \mathbb{F}_{q^2}[x]^\rho$ is

$$\deg_{\nu,\boldsymbol{w}} \boldsymbol{v} = \max_i\{w_i + \nu \deg v_i\},$$

where v_i and w_i are the elements of $\boldsymbol{v}$ respectively $\boldsymbol{w}$. Similarly, we will consider $\text{LP}_{\nu,\boldsymbol{w}}(\boldsymbol{v}) = \max\{i \mid w_i + \nu \deg v_i = \deg_{\nu,\boldsymbol{w}} \boldsymbol{v}\}$. For decoding one-point Hermitian codes, we will be using $\nu = q$.

We will now explain how to handle such weights without changing the underlying module minimisation algorithm or incurring any serious performance penalty. We will introduce two injective mappings for matrices such that finding a weak Popov form of the image of V under either will solve the weighted minimisation problem. The first is a straightforward embedding of the weights but has two downsides: it can only be used with certain module minimisation algorithms, and those algorithms need

[1]These names sometime refer to computing a "row reduced" matrix which is a slightly weaker property than being in weak Popov form.

to be implemented in a specific manner to avoid a computational overhead. To mitigate both of these problems we derive a second embedding from the first.

First we define the following straightforward map $\Phi_{\nu,\boldsymbol{w}} : \mathbb{F}_{q^2}[x]^\rho \mapsto \mathbb{F}_{q^2}[x]^\rho$:

$$\Phi_{\nu,\boldsymbol{w}}\big((v_1,\ldots,v_\rho)\big) = \big(x^{w_1}v_1(x^\nu),\ldots,x^{w_\rho}v_\rho(x^\nu)\big).$$

We extend $\Phi_{\nu,\boldsymbol{w}}$ row-wise to $\rho\times\rho$ matrices such that the ith row of $\Phi_{\nu,\boldsymbol{w}}(V)$ is $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}_i)$, where $\boldsymbol{v}_i$ are the rows of V. Note that $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ is a free $\mathbb{F}_{q^2}[x^\nu]$-module of dimension ρ, and that any basis of it is by $\Phi^{-1}_{\nu,\boldsymbol{w}}$ sent back to a basis of $\mathcal{V}$.

Proposition 10: A vector $\boldsymbol{v}\in\mathcal{V}$ has minimal $\deg_{\nu,\boldsymbol{w}}$ if $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v})$ has minimal degree in $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$. Furthermore, $\text{LP}_{\nu,\boldsymbol{w}}(\boldsymbol{v}) = \text{LP}(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}))$.

Proof: This follows immediately since for any vector $\boldsymbol{v}=(v_1,\ldots,v_\rho)\in\mathbb{F}[x]^\rho$, then $\deg\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{v}) = \deg_{\nu,\boldsymbol{w}}\boldsymbol{v}$. ∎

In other words, we can hope to solve the weighted problem as follows: find a $\Phi_{\nu,\boldsymbol{w}}(W)$ in weak Popov form and unimodular equivalent to $\Phi_{\nu,\boldsymbol{w}}(V)$. Then the $\Phi^{-1}_{\nu,\boldsymbol{w}}$-map of the row of $\Phi_{\nu,\boldsymbol{w}}(W)$ with minimal degree and leading position in I yields the sought solution. However, a general module minimisation algorithm will consider the $\mathbb{F}_{q^2}[x]$-module spanned by $\Phi_{\nu,\boldsymbol{w}}(V)$ – and not the $\mathbb{F}_{q^2}[x^\nu]$-module – so a weak Popov form of $\Phi_{\nu,\boldsymbol{w}}(V)$ will generally not result in a matrix in $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$, and hence we cannot apply $\Phi^{-1}_{\nu,\boldsymbol{w}}$ to its rows. In the case of the Mulders–Storjohann algorithm [14] or the Alekhnovich algorithm [3], one can show that things will go well: applying either algorithm to $\Phi_{\nu,\boldsymbol{w}}(V)$ results in a weak Popov form in $\Phi_{\nu,\boldsymbol{w}}(\mathcal{V})$ [1], [2], [17]. Furthermore, if properly implemented, these algorithms will not incur a computational penalty from the $x\mapsto x^\nu$ blow-up.

To take advantage of the faster module minimisation algorithms – in a manner ensuring both correctness and speed – we introduce a second mapping which does not have the problems of $\Phi_{\nu,\boldsymbol{w}}$.

For this improved mapping, consider first the permutation π of $[1,\ldots,\rho]$ defined indirectly by the following property:

$$\begin{aligned}\pi(i)>\pi(j) \iff\ & (w_i \bmod \nu) > (w_j \bmod \nu)\\ & \vee\big((w_i \bmod \nu)=(w_j \bmod \nu)\wedge i>j\big).\end{aligned}$$

The permutation π acts on vectors of $\mathbb{F}[x]^\rho$ by permuting the positions of such vectors. Our desired mapping is now $\Psi_{\nu,\boldsymbol{w}}$:

$$\Psi_{\nu,\boldsymbol{w}}\big((v_1,\ldots,v_\rho)\big) = \pi\big((x^{\lfloor w_1/\nu\rfloor}v_1,\ldots,x^{\lfloor w_\rho/\nu\rfloor}v_\rho)\big).$$

Proposition 11: For any $\boldsymbol{v}\in\mathbb{F}[x]^\rho$ then

$$(\pi^{-1}\circ\text{LP}\circ\Psi_{\nu,\boldsymbol{w}})(\boldsymbol{v}) = (\text{LP}\circ\Phi_{\nu,\boldsymbol{w}})(\boldsymbol{v}).$$

Proof: Let v_i be the elements of $\boldsymbol{v}$, and $h=(\text{LP}\circ\Phi_{\nu,\boldsymbol{w}})(\boldsymbol{v})$. We will prove that no index but $\pi(h)$ can be $(\text{LP}\circ\Psi_{\nu,\boldsymbol{w}})(\boldsymbol{v})$. Consider first some $j>h$. By the definition of h then

$$\begin{aligned}\nu\deg v_h + w_h > \nu\deg v_j + w_j, \quad \text{i.e.}\\ \deg v_h + \lfloor w_h/\nu\rfloor + \tfrac{w_h \bmod \nu}{\nu} > \deg v_j + \lfloor w_j/\nu\rfloor + \tfrac{w_j \bmod \nu}{\nu}.\end{aligned} \tag{2}$$

So either $\deg v_h + \lfloor w_h/\nu\rfloor > \deg v_j + \lfloor w_j/\nu\rfloor$, or they are equal and $w_h \bmod \nu > w_j \bmod \nu$. In the first case, then clearly $\pi(j)$ cannot be $(\text{LP}\circ\Psi_{\nu,\boldsymbol{w}})(\boldsymbol{v})$ due to degrees. In the second case the degrees of $\Psi_{\nu,\boldsymbol{w}}(\boldsymbol{v})$ at positions $\pi(h)$ and $\pi(j)$ are tied, but we have $\pi(h)>\pi(j)$, which means that $\pi(j)$ cannot be the leading position.

Consider now some $j<h$, so we have the same inequality (2) but with $>$ replaced by $\geq$. If sharp inequality really holds, then we can continue as before, so assume instead that equality holds. That implies both $\deg v_h + \lfloor w_h/\nu\rfloor = \deg v_j + \lfloor w_j/\nu\rfloor$ and $w_h \equiv w_j \mod \nu$. So the degrees of $\Psi_{\nu,\boldsymbol{w}}(\boldsymbol{v})$ at positions $\pi(h)$ and $\pi(j)$ are tied, but then since $h>j$, we have $\pi(h)>\pi(j)$. Again, $\pi(j)$ is not the leading position. ∎

Corollary 12: For any $V\in\mathbb{F}[x]^\rho$, then $\Phi_{\nu,\boldsymbol{w}}(V)$ is in weak Popov form if and only if $\Psi_{\nu,\boldsymbol{w}}(V)$ is in weak Popov form.

The algorithm is then clear: to solve the weighted minimisation problem, simply compute a weak Popov form of $\Psi_{\nu,\boldsymbol{w}}(V)$. The row with minimal degree, and in case of a tie least LP, only among rows whose LP are in $\{\pi(i)\mid i\in I\}$ corresponds to a minimal solution, and one applies $\Psi^{-1}_{\nu,\boldsymbol{w}}$ to obtain the vector of $\mathcal{V}$. This works immediately for any module minimisation algorithm.

For calculating the resulting complexity in general, one observes that

$$\deg\Psi_{\nu,\boldsymbol{w}}(V) \leq \gamma \triangleq \deg V + \max_j(w_j/\nu).$$

The trivial bound gives $\Delta(\Psi_{\nu,\boldsymbol{w}}(V)) \leq \rho\gamma$, so in Table I, one can replace $\deg V$ with γ and $\Delta(V)$ with $\rho\gamma$ to obtain the generic complexities for solving the weighted problem.

IV. Fast Implementation of Guruswami–Sudan

We will now present a sub-quadratic realisation of the Guruswami–Sudan decoding algorithm for the one-point Hermitian codes introduced in Section II. The main contribution is demonstrating how to perform the interpolation step using the fast module minimisation techniques discussed in the previous section. This builds heavily on previous works [1], [2], and we remark further on this at the end of Section IV-A. Since the fastest previously known method for performing the root finding step was at least quadratic in n [2], we also describe how to sufficiently speed up this step in Section IV-B.

In the following sections, we will consider dealing with a particular choice of a one-point Hermitian code, and use all the introduced variables $n, k, P_i, d^\star, \mathcal{C}$, etc. from Section II. We will consider that a given codeword $\boldsymbol{c} \in \mathcal{C}$ was sent, resulting from evaluating $f \in \mathcal{L}(mP_\infty)$, and that $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ was received with some error $\boldsymbol{e}$. Further denote by $\mathcal{E}$ the set of error positions, i.e. $\mathcal{E} = \{i \mid e_i \neq 0\}$. The aim is to recover $\boldsymbol{c}$ knowing only $\boldsymbol{r}$, possibly even when $\mathrm{weight}(\boldsymbol{e}) \geq d^\star/2$.

We will be working with elements of $\mathcal{R}[z]$, i.e. the univariate polynomial ring over $\mathcal{R}$. Define for such $Q = \sum_{t=0}^{\deg_z Q} Q_t(x,y)z^t \in \mathcal{R}[z]$ the coefficient-selecting notation $Q_{[t]}$ to mean $Q_{[t]} = Q_t(x,y) \in \mathcal{R}$. We extend our degree function in a natural way to $\deg_{\mathcal{H},w}$ for any $w \in \mathbb{R}$, so that some $Q \in \mathcal{R}[z]$ has $\deg_{\mathcal{H},w} Q = \max_t\{\deg_{\mathcal{H}} Q_{[t]} + tw\}$.

Definition 13: A polynomial $Q \in \mathcal{R}[z]$ has a zero $(P, z_0) \in \mathcal{P}^\star \times \mathbb{F}_{q^2}$ with multiplicity at least s if Q can be written as $\sum_{j+h\geq s} \gamma_{j,h}\phi^j(z - z_0)^h$ for some $\gamma_{j,h} \in \mathbb{F}_{q^2}$, where ϕ is a local parameter for P.

For any place (α, β), one can choose as local parameter $\phi = x - \alpha$, which makes the above definition easy to operate with. Note though that the sum in $Q = \sum_{j+h\geq s} \gamma_{j,h}\phi^j(z - z_0)^h$ will be an infinite sum (that is to say, a power series) in general. However, to determine whether or not the multiplicity of $((\alpha, \beta), z_0)$ is at least s, one only needs to compute finitely many terms of this power series. For one-point Hermitian codes, the Guruswami–Sudan algorithm then builds on the following theorem:

Theorem 14 (Guruswami–Sudan): Let $s, \ell, \tau \in \mathbb{Z}_+$ be given. If a non-zero $Q \in \mathcal{R}[z]$ with $\deg_z Q \leq \ell$ satisfies

1) Q has a zero at (P_i, r_i) with multiplicity at least s for $i = 1, \ldots, n$,
2) $\deg_{\mathcal{H},m} Q < s(n - \tau)$

and if $|\mathcal{E}| \leq \tau$, then $Q(f) = 0$.

Note that ℓ is to be given as an a priori bound on $\deg_z Q$, but another bound is already indirectly enforced by Item 2: by this, it never makes sense to choose ℓ such that $s(n - \tau) - \ell m \leq 0$.

Remark 15: An analogous theorem holds for much more general AG codes, though $\mathcal{R}$ of course needs to be defined properly. See e.g. [18] or the expository description in [19].

One can find a satisfactory Q by solving a system of linear equations in the $\mathbb{F}_{q^2}$-coefficients for its $x^i y^j z^h$-monomials, and one can ensure that this system will have a non-zero solution by satisfying a certain expression in the parameters. The resulting equation can be analysed for determining the maximal τ and corresponding choices of s and ℓ. We are not going to perform that analysis but see e.g. [1]. Given s and ℓ, one can use the equation to compute a value $\tau_{\mathrm{GS}}(s, \ell)$ such that one can choose any $\tau \leq \tau_{\mathrm{GS}}(s, \ell)$. Furthermore, $\tau_{\mathrm{GS}}(s, \ell)$ is the greatest integer less than

$$\left(1 - \frac{s+1}{2(\ell+1)}\right) n - \frac{m}{2}\frac{\ell}{s} - \frac{g}{s} \ . \tag{3}$$

Analysing the asymptotics of this bound, one sees that there are choices of s and ℓ which allows choosing any $\tau < n - \sqrt{n(n - d^\star)}$. This function $n - \sqrt{n(n - d^\star)}$ is called the Johnson radius.

For specific parameters of the code and s and ℓ, the lower bound on $\tau_{\mathrm{GS}}(s, \ell)$ is good but not always tight; it is easy to compute the precise value of $\tau_{\mathrm{GS}}(s, \ell)$, though a closed expression is complicated. If one considers the Guruswami–Sudan as an algorithm taking s and ℓ as parameters (and the code), then $\tau_{\mathrm{GS}}(s, \ell)$ is the guaranteed number of errors that it is able to correct. It is very interesting that the algorithm will quite often succeed in correcting more errors; this was already remarked in [1]. We will get back to this in Section VI.

A. Finding Q in an Explicit Module

We will now concern ourselves with the problem of finding Q. We will assume $s \leq \ell$; with the proper analysis of choices of s and ℓ, one can show that $s > \ell$ implies $\tau < d^\star/2$.

Definition 16: Let $\mathcal{M}_{s,\ell} \subset \mathcal{R}[z]$ denote the set of all $Q \in \mathcal{R}[z]$ such that Q has a zero of multiplicity s at (P_i, r_i) for $i = 1, \ldots, n$, and $\deg_z Q \leq \ell$.

Finding a $Q \in \mathcal{R}[z]$ for satisfying the requirements of Theorem 14 is then the same as finding an element in $\mathcal{M}_{s,\ell}$ with low enough $\deg_{\mathcal{H},m}$. We will find one with minimal $\deg_{\mathcal{H},m}$ which is guaranteed to be sufficient by the choice of parameters s, ℓ, τ.

It is not hard to see that $\mathcal{M}_{s,\ell}$ is a $\mathit{Я}$-module. To proceed, we will need to give an explicit basis for $\mathcal{M}_{s,\ell}$. We will use a basis previously given in the literature [1]. We will need two functions in $\mathit{Я}$:

$$G = \prod_{i=1}^{n/q}(x-\alpha_i) = x^{q^2} - x, \tag{4}$$

$$R: \quad R(P_i) = r_i \qquad \forall i = 1,\dots,n. \tag{5}$$

The function G is known in advance and by Proposition 1, we have $\mathrm{div}(G) = \sum_{i=1}^{n} P_i - nP_\infty$.

The function R depends on the received word $\boldsymbol{r}$. Any non-zero function in $\mathit{Я}$ satisfying the interpolation constraints will do; we can either solve the linear system of equations in its coefficients, or we can use the explicit formula of Lemma 6. The desired explicit basis of $\mathcal{M}_{s,\ell}$ is the following:

Theorem 17 ([1, Proposition 7]): $\mathcal{M}_{s,\ell}$ is generated as a $\mathit{Я}$-module by the $\ell+1$ polynomials $H^{(i)} \in \mathit{Я}[z]$ given by

$$H^{(t)}(z) = G^{s-t}(z-R)^t, \qquad \text{for } 0 \le t \le s,$$
$$H^{(t)}(z) = z^{t-s}(z-R)^s, \qquad \text{for } s < t \le \ell.$$

We need to project this module, its basis and the weighted degree into $\mathbb{F}_{q^2}[x]$ in some sensible manner to be able to use the tools of Section III to find an element in $\mathcal{M}_{s,\ell}$ of minimal $\deg_{\mathcal{H}}$.

Firstly, introduce $\Upsilon : \mathit{Я} \mapsto \mathbb{F}_{q^2}[x]^q$: for any $g = \sum_{i=0}^{q-1} y^i g_i(x) \in \mathit{Я}$, then we define $\Upsilon(g) = (g_0,\dots,g_{q-1})$. As we have previously noted, any element of $\mathit{Я}$ can uniquely be written such that the y-degree is at most $q-1$. This implies that the map Υ is well-defined and a bijection. Let $\mathit{Я}[z]_\ell$ be the set of polynomials of z-degree at most ℓ; then we also introduce $\Upsilon_z : \mathit{Я}[z]_\ell \mapsto \mathbb{F}_{q^2}[x]^{(\ell+1)q}$, as for any $Q \in \mathit{Я}[z]_\ell$, then $\Upsilon_z(Q) = \big(\Upsilon(Q_{[0]}) \mid \dots \mid \Upsilon(Q_{[\ell]})\big)$. Define now $\boldsymbol{w} \in \mathbb{N}_0^{(\ell+1)q}$ as $\boldsymbol{w} = (\boldsymbol{w}_0 \mid \dots \mid \boldsymbol{w}_\ell)$, where

$$\boldsymbol{w}_t = (tm, tm+q+1, \dots, tm+(q-1)(q+1)).$$

One can then verify the following identity for any $Q \in \mathit{Я}[z]_\ell$:

$$\deg_{\mathcal{H},m}(Q) = (\deg \circ \Phi_{q,\boldsymbol{w}} \circ \Upsilon_z)(Q),$$

where $\Phi_{q,\boldsymbol{w}}$ is as in Section III.

Proposition 18: Let $A_{s,\ell} \in \mathbb{F}_{q^2}[x]^{(q(\ell+1))\times(q(\ell+1))}$ be given as

$$\left(\left(\begin{array}{c} \Upsilon_z(H^{(0)}) \\ \hline \Upsilon_z(yH^{(0)}) \\ \hline \vdots \\ \hline \Upsilon_z(y^{q-1}H^{(0)}) \end{array} \right)^\top \;\middle|\; \cdots \;\middle|\; \left(\begin{array}{c} \Upsilon_z(H^{(\ell)}) \\ \hline \Upsilon_z(yH^{(\ell)}) \\ \hline \vdots \\ \hline \Upsilon_z(y^{q-1}H^{(\ell)}) \end{array} \right)^\top \right)^\top,$$

then $\mathcal{M}_{s,\ell}$ is in bijection with the $\mathbb{F}_{q^2}[x]$ row space of $A_{s,\ell}$ through the map Υ_z. Let $\Upsilon_z(Q)$ be the vector in this row space with minimal $\Phi_{q,\boldsymbol{w}}$-weighted degree. Then Q has minimal $\deg_{\mathcal{H},m}$ in $\mathcal{M}_{s,\ell}$.

Proof: Consider some $Q(z) \in \mathcal{M}_{s,\ell}$; by Theorem 14 we can find $p_t \in \mathit{Я}$ such that $Q(z) = \sum_{t=0}^{\ell} p_t H^{(t)}(z)$. Let $p_t = \sum_{j=0}^{q-1} p_{t,j} y^j$ with $p_{t,j} \in \mathbb{F}_{q^2}[x]$, then

$$Q = \sum_{t=0}^{\ell} p_t H^{(t)} = \sum_{t=0}^{\ell}\sum_{j=0}^{q-1} p_{t,j}(y^j H^{(t)}).$$

This directly implies that

$$\Upsilon_z(Q) = \sum_{t=0}^{\ell}\sum_{j=0}^{q-1} p_{t,j}\,\Upsilon_z(y^j H^{(t)}),$$

which is to say, $\Upsilon_z(Q)$ is in the $\mathbb{F}_{q^2}[x]$ row space of $A_{s,\ell}$.

The claim on weighted degrees follow immediately from $\deg_{\mathcal{H},m} = \deg \circ \Phi_{q,\boldsymbol{w}} \circ \Upsilon_z$. ∎

By Problem 7, we can therefore find a minimal $\deg_{\mathcal{H},m}$-weighted $Q \in \mathcal{M}_{s,\ell}$ by bringing $\Psi_{q,\boldsymbol{w}}(A_{s,\ell})$ to weak Popov form. We get:

Proposition 19: In the context of Proposition 18, the worst-case complexity of finding a satisfactory Q as a minimal element in the row space of $\Psi_{q,\boldsymbol{w}}(A_{s,\ell})$ is as in Table II, for various choices of module minimisation algorithm.

Proof: We firstly need to construct $A_{s,\ell}$: we assume G^t precomputed for $t = 1,\dots,s$, and R can be computed in $O\tilde{}(n)$ according to Lemma 6. Computing R^t for $t = 1,\dots,s$, each represented in the standard basis with y-degree less than q, can

Complexity for computing Q

Algorithm	Field operations in big-$O^\sim$
Mulders–Storjohann [14]	$n^{7/3}\ell^3 s^2$
Alekhnovich [3]	$n^{(3+\omega)/3}\ell^{\omega+1}s$
GJV [4] or Zhou–Labahn [15]	$n^{(2+\omega)/3}\ell^{\omega}s$

Table II
USE OF $O^\sim$ AND ω AS IN TABLE I.

be done iteratively in $sO^\sim(q)O^\sim(sq^2) = O^\sim(s^2n)$: $R \cdot R^{t-1}$ can be computed as multiplying two degree $q-1$ polynomials in y whose coefficients are in $\mathbb{F}_{q^2}[x]$ with degree in $O(sq^2)$ by Lemma 6. We then need to use $\mathcal{H}$ to reduce the y-degree to less than q, which can be done with at most $3q$ additions of $\mathbb{F}_{q^2}[x]$-polynomials of degree at most $O(sq^2)$. To then compute the $H^{(t)}$, we need $\binom{t}{t'}G^{s-t}R^{t'}$ for $t = 0, \ldots, s$ and $0 \leq t' \leq t$ at a cost of a further $O^\sim(s^3n)$. Since the $H^{(t)}$ are then computed in the standard basis, the final construction of $A_{s,\ell}$ is simply linear in its size which is $O(\ell^2 sn)$.

By Section III-A, the complexity of bringing $\Psi_{q,\boldsymbol{w}}(A_{s,\ell})$ to weak Popov form is dominated by

$$\begin{aligned}\gamma &= \deg(A_{s,\ell}) + \max \boldsymbol{w}/q \\ &= O(sn^{2/3}) + (\ell m + (q-1)(q+1))/q.\end{aligned}$$

By the note right after Theorem 14 then $\ell m < s(n-\tau)$ so $\gamma \in O(sn^{2/3})$. The complexities then follow by noting that $A_{s,\ell}$ has $(\ell+1)q$ rows and columns, and $\Delta(\Psi_{q,\boldsymbol{w}}(A_{s,\ell})) \leq (\ell+1)q\gamma$. ∎

Remark 20: For the interpolation step of Guruswami–Sudan in decoding of algebraic geometry codes, both the Mulders–Storjohann and the Alekhnovich algorithm have been suggested, [1] respectively [2]. Note that the algorithm described in [1] is computationally equivalent with the Mulders–Storjohann algorithm though derived in terms of Gröbner bases. In both [1] and [2], the mapping $\Phi_{\nu,\boldsymbol{w}}$ was (implicitly) used together with a detailed analysis of the module minimisation algorithms to prove that the operations did not leave the $\mathbb{F}[x^q]$-module, and that the slow-down discussed in Section III-A did not occur.

The GJV and the Zhou–Labahn methods have not previously been applied for this decoding setting, and the application of $\Psi_{q,\boldsymbol{w}}$ allows us to deduce correctness and the low complexity without investigating the algorithm in detail.

Note that the GJV has previously been suggested for decoding of Reed–Solomon codes [20].

B. Fast Root finding

After having constructed $Q(z)$, we should find all $f \in \mathcal{L}(mP_\infty)$ such that $Q(f) = 0$. This can be done using Hensel lifting [19], [21], inspired by the algorithm of Roth and Ruckenstein [22] for solving the root-finding problem for Reed–Solomon codes. The complexity of these methods all have at least quadratic dependence on n, and so would be asymptotically slower than the interpolation described in the previous section.

Alekhnovich described in [3] how to use fast arithmetic to bring the method of [22] down to quasi-linear complexity in n. Using the power series idea of [19] it is easy to apply this algorithm to our root-finding problem as well. For our case, the main result can be paraphrased as follows; its proof as well as the complete root-finding algorithm is given in Appendix A.

Proposition 21: For $Q \in \mathcal{R}[z]$ satisfying the requirements of Theorem 14, then we can compute all $f \in \mathcal{L}(mP_\infty)$ such that $Q(f) = 0$ in time $O^\sim(n^{4/3}\ell^2 s)$.

We have now described how to realise the complete Guruswami–Sudan algorithm with asymptotic complexity $O^\sim(n^{(\omega+2)/3}\ell^\omega s)$. Note that the only step in the entire algorithm with this complexity is the module minimisation step; all other steps have lower order. This means that the hidden constant in the big-O notation for the leading term in our decoder must be *exactly* that of the module minimisation employed. In an implementation and for concrete parameters, one could of course still be concerned that the remaining, asymptotically lower-order terms, dominate the actual running time. We demonstrate in Section VI that this is unlikely since their running time in our implementation is very low.

V. FAST POWER DECODING

In this section we will present a decoding algorithm generalising classical syndrome decoding [23] for low-rate one-point Hermitian codes, obtained by "powering" the key equations. The technique, also known as "virtual extension to an interleaved code" was developed for Reed–Solomon codes by Schmidt et al. [24]. It has already been suggested for one-point Hermitian codes by Kampf and Li [25], [26], but no proof of the algorithm's complexity was given.

As opposed to this previous work, we will power a Gao-style key equation in place of the classical syndrome key equation. Apart from the joy of variety, this admits a succinct derivation which follows the definition of the codes as evaluations closely, and it highlights some similarities with Guruswami–Sudan decoding. Another advantage is that the sent information polynomial is evident immediately, and one does not need to find the zeroes of the error locator and do erasure decoding or similar

afterwards. For Reed–Solomon codes, this variation was suggested in [5] and proved to be behaviourally equivalent to the syndrome formulation.

We will show how to put the problem into a framework where fast algorithms for module minimisation can be directly applied, and this will yield a fast decoding algorithm with speed asymptotically comparable to that of Guruswami–Sudan. As with Guruswami–Sudan, one can set the decoding algorithm's parameters to perform minimum distance decoding, and in this case we improve upon the fastest, previously known techniques. Note that the module minimisation framework also applies to classical syndrome decoding, and is therefore the first significant speed improvement of this technique in the last 20 years, since [6].

Power decoding is not list decoding: it either gives one answer or it will fail. For Reed–Solomon codes, it might only fail when the number of errors has exceeded half the minimum distance, and statistically this has been verified to occur only very rarely. There are failure probability bounds for "powering degree" 2 and 3, but not in the general case [5], [24], [27]. For one-point Hermitian codes, the genus of the curve play a role in the decoding radius—as usual—and we will get back to the precise decoding performance in Section V-D. As for Reed–Solomon codes, we have not yet obtained a bound on the failure probability, but experiments indicate similar behaviour.

A. Key Equations

Recall that $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ was received, and denote the set of error positions by $\mathcal{E}$.

Definition 22: The *error locator* Λ is the non-zero polynomial in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$ with minimal $\deg_{\mathcal{H}}$ and $\mathrm{LC}_{\mathcal{H}}(\Lambda) = 1$.

Clearly, $\Lambda \in \mathcal{H}$ since the defining Riemann–Roch space is a subset of $\mathcal{H}$. It is easy to see that the definition is well-posed, i.e. there is exactly one element in the Riemann–Roch space satisfying the restrictions.

Lemma 23: $|\mathcal{E}| \leq \deg_{\mathcal{H}} \Lambda \leq |\mathcal{E}| + g$.

Proof: Being in $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$ specifies $|\mathcal{E}|$ homogeneous equations in the coefficients of Λ, so by Lemma 4, we will still have more coefficients than equations after requiring $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g + 1$. For the lower bound, then since $\deg(-\sum_{i\in\mathcal{E}} P_i + tP_\infty) < 0$ for $t < |\mathcal{E}|$ we must have $\mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + tP_\infty) = \{0\}$ whenever $t < |\mathcal{E}|$. Since $\Lambda \neq 0$ is in this Riemann–Roch space when $t = \deg_{\mathcal{H}} \Lambda$, then clearly $\deg_{\mathcal{H}} \Lambda \geq |\mathcal{E}|$. ∎

Recall now G and R from Equation (5), and extend the latter to "powers":

$$R^{(t)} : \; R^{(t)}(P_i) = r_i^t \quad \forall i = 1, \ldots, n, \quad t \in \mathbb{N}_0. \tag{6}$$

Again, the $R^{(t)}$ can be found by solving the emerging linear systems of equations or using the explicit formula of Lemma 6. We then immediately arrive at the powered key equations over the function field:

Theorem 24: $\Lambda R^{(t)} \equiv \Lambda f^t \mod G$ for $t \in \mathbb{N}_0$ as a congruence over $\mathcal{H}$.

Proof: We have $\Lambda R^{(t)} - \Lambda f^t = \Lambda(R^{(t)} - f^t) \in \mathcal{L}(-\sum_{i=1}^n P_i + \infty P_\infty)$, since for $i \in \mathcal{E}$ then $\Lambda(P_i) = 0$ while for $i \notin \mathcal{E}$ then $R^{(t)}(P_i) = f^t(P_i)$. Recall that $\mathrm{div}(G) = \sum_{i=1}^n P_i - nP_\infty$; therefore by Lemma 3 we must have $G \mid \Lambda(R^{(t)} - f^t)$ over $\mathcal{H}$. ∎

This means that the sought Λ is a solution to a list of key equations – but over $\mathcal{H}$. We will handle these non-linear equations similarly to how classical key equations are handled: regard the right-hand side as unknowns independent of Λ and each other, and only enforce bounds on its degree. Then seek a minimal $\deg_{\mathcal{H}}$-element $\hat{\Lambda} \in \mathcal{H}$ such that $\hat{\Lambda} R^{(t)} \bmod G$ satisfies this degree bound for each t. One then hopes that $\hat{\Lambda} = \Lambda$.

Theorem 24 provides us with infinitely many key equations, but when we are using the above strategy we are only aided by those for which the degree bound on $\hat{\Lambda} R^{(t)} \bmod G$ is not trivially satisfied; in particular, when $\deg_{\mathcal{H}}(\Lambda f^t) > \deg_{\mathcal{H}} G$ then the key equation for this t is useless. We do not know $\deg_{\mathcal{H}} \Lambda$ but we can at least disregard those equations for which $tm \geq \deg_{\mathcal{H}} G = n$. Thus, in the following, we will use equations $t = 1, \ldots, \ell$ where ℓ is chosen such that $\ell m < n$.

As in Section IV, we will project the key equations over $\mathcal{H}$ into equations over $\mathbb{F}_{q^2}[x]$ to be able to use module minimisation for finding the minimal $\hat{\Lambda}$. We will introduce a bit more notation for this: for two $a, b \in \mathcal{H}$, with vector forms $\Upsilon(a), \Upsilon(b)$ we wish to represent their product ab in vector form. With $\Upsilon(b) = (b_0, \ldots, b_{q-1})$, consider the following vector–matrix product:

$$\Upsilon(a) \begin{pmatrix} b_0 & b_1 & \ldots & b_{q-1} & & & 0 \\ & b_0 & b_1 & \ldots & b_{q-1} & & \\ & & \ddots & & & \ddots & \\ 0 & & & b_0 & b_1 & \ldots & b_{q-1} \end{pmatrix}. \tag{7}$$

The result will be a vector $(c_0, \ldots, c_{2q-2}) \in \mathbb{F}_{q^2}[x]^{2q-1}$ such that $ab = \sum_{i=0}^{2q-2} c_i(x)y^i$. Denote by $\amalg_b$ the matrix of the above form, for any $b \in \mathcal{R}$. Using the curve equation $\mathcal{H}$ to rewrite $\sum_{i=0}^{2q-2} c_i(x)y^i$ into having y-degree less than q becomes the result of the linear transformation

$$\begin{pmatrix} c_0 & c_1 & \ldots & c_{2q-2} \end{pmatrix} \left(\begin{array}{cccc} & & I_{q\times q} & \\ \hline x^{q+1} & -1 & & 0 \\ & \ddots & \ddots & \\ 0 & & x^{q+1} & -1 \\ & & & x^{q+1} \end{array} \right), \tag{8}$$

where $I_{q\times q}$ is the $q \times q$ identity matrix. Denote the matrix in the above product by Ξ. With this notation then we can write

$$\Upsilon(ab) = \Upsilon(a)\amalg_b \Xi. \tag{9}$$

Corollary 25: $\Upsilon(\Lambda) = (\Lambda_0, \ldots, \Lambda_{q-1})$ satisfies the $q\ell$ congruences over $\mathbb{F}_{q^2}[x]$:

$$\sum_{i=0}^{q-1} \Lambda_i T_{i,j} \equiv B_j \mod G, \quad j = 1, \ldots q\ell,$$

where the $B_j \in \mathbb{F}_{q^2}[x]$ satisfy

$$q \deg B_j + (q+1)((j-1) \bmod q) < \deg_{\mathcal{H}} \Lambda + m\lceil j/q \rceil + 1$$

and where $T = [T_{i,j}] \in \mathbb{F}_{q^2}[x]^{q\times q\ell}$ equals the matrix

$$\left[\amalg_{R^{(1)}}\Xi \mid \amalg_{R^{(2)}}\Xi \mid \ldots \mid \amalg_{R^{(\ell)}}\Xi\right]$$

element-wise reduced modulo G.

Proof: Theorem 24 implies for each t that there is a $p_t \in \mathcal{R}$ such that $\Lambda R^{(t)} = \Lambda f^t + p_t G$, which means

$$\Upsilon(\Lambda R^{(t)}) = \Upsilon(\Lambda f^t) + G\Upsilon(p_t).$$

Letting $\Upsilon(\Lambda f^t) = (B_{t,0}, \ldots, B_{t,q-1})$, then the above implies for $h = 0, \ldots, q-1$ that

$$\Upsilon(\Lambda)\amalg_{R^{(t)}}\Xi \equiv B_{t,h} \mod G$$

as an $\mathbb{F}_{q^2}[x]$-congruence. Furthermore, since $\deg_{\mathcal{H}}(\Lambda f^t) \le \deg_{\mathcal{H}} \Lambda + tm$, then $q \deg B_{t,h} + h(q+1) \le \deg_{\mathcal{H}} \Lambda + tm$.

Taken over all $t = 1, \ldots, \ell$ and relabelling $B_{t,h}$ appropriately, then this gives the $q\ell$ congruence equations of the corollary. ■

Note that the degree constraints on the remainders B_j depend on $\deg_{\mathcal{H}} \Lambda$, i.e. on $\max_i\{q \deg \Lambda_i + i(q+1)\}$. The above $q\ell$ equations therefore constitute a heavily generalised form of a weighted key equation. The form of the "key equation" is elsewhere often called Padé approximation, and the equations of Corollary 25 generalise both the notion of Simultaneous Padé and Hermitian Padé. This form was recently considered in [12] under the name "asymmetric 2D Padé approximation"; see also this paper for discussion on and references to other Padé-like approximants.

B. Solving the Key Equations

We will here outline the method of [12] for finding a minimal solution to the equations of Corollary 25. By "solution" we mean any $(\hat{\Lambda}_0, \ldots, \hat{\Lambda}_{q-1}) \in \mathbb{F}_{q^2}[x]^q$ such that the congruence equations of Corollary 25 are satisfied along with the degree bounds on the remainders $\hat{B}_j$. By "minimal" we will seek a solution such that $\deg_{\mathcal{H}} \hat{\Lambda}$ is minimal. The hope is then that $\Lambda = \hat{\Lambda}$; if that is not the case, we will declare a decoding failure. In Section V-D, we discuss the likelihood of this event occurring in more detail.

Consider first any vector $(\lambda_0, \ldots, \lambda_{q-1}, b_1, \ldots, b_{q\ell}) \in \mathbb{F}_{q^2}[x]^{q(\ell+1)}$ which satisfies the congruences, i.e.

$$\sum_{i=0}^{q-1} \lambda_i T_{i,j} \equiv b_j \mod G, \quad j = 1, \ldots q\ell.$$

One can quickly see that the space of all such vectors constitutes an $\mathbb{F}_{q^2}[x]$-submodule of $\mathbb{F}_{q^2}[x]^{q(\ell+1)}$. Furthermore, the rows of the following square matrix M is a basis of this submodule:

$$M = \left[\begin{array}{c|c} I_q & T \\ \hline \mathbf{0} & GI_{q\ell} \end{array}\right], \tag{10}$$

where I_m is the $m \times m$ identity matrix. "Solutions" to the equations are therefore vectors in the $\mathbb{F}_{q^2}[x]$ row-space of M such that the b_j satisfy some degree constraints which are dependent on the λ_i, and we are seeking a solution where $\Upsilon^{-1}(\lambda_0, \ldots, \lambda_{q-1})$

Complexity for solving the key equations

Algorithm	Field operations in big-$O^\sim$
Mulders–Storjohann [14]	$n^{7/3}\ell^2$
Demand–Driven [12]	$n^{7/3}\ell$
Alekhnovich [3]	$n^{(3+\omega)/3}\ell^\omega$
GJV [4] or Zhou–Labahn [15]	$n^{(2+\omega)/3}\ell^\omega$

Table III
USE OF $O^\sim$ AND ω AS IN TABLE I.

has minimal $\deg_{\mathcal{H}}$. We will handle the latter by finding appropriate weights in the sense of Section III-A, and encode the degree constraints of the b_j as a leading position-constraint on the weighted vector.

Recall the mapping $\Phi_{\nu,\boldsymbol{w}}$ of Section III-A. Let $\boldsymbol{\eta} = (0, q+1, \ldots, (q-1)(q+1))$; then for any $\lambda \in \mathcal{R}$ clearly $\Phi_{q,\boldsymbol{\eta}}(\Upsilon(\lambda)) = \deg_{\mathcal{H}} \lambda$.

Let now $\mu_j = (q+1)((j-1) \bmod q) - m\lceil j/q \rceil - 1$ for $j = 1, \ldots, \ell q$, so the degree constraints for the b_j can be written as

$$\deg b_j + \mu_j < \deg(\Phi_{q,\boldsymbol{\eta}}(\lambda_0, \ldots, \lambda_{q-1})).$$

Some of the μ_j might be negative, which the method of Section III-A cannot directly handle, so we shift all weights by $\ell m + 1$ to ensure non-negativity. Therefore letting $\bar{\eta}_i = \eta_i + \ell m + 1$ and $\bar{\mu}_j = \mu_j + \ell m + 1$, introduce $\boldsymbol{w} = (\bar{\eta}_0, \ldots, \bar{\eta}_{q-1}, \bar{\mu}_1, \ldots, \bar{\mu}_{q\ell})$ to realise that the degree constraints can now be written as

$$\text{LP}(\Phi_{q,\boldsymbol{w}}(\lambda_0, \ldots, \lambda_{q-1}, b_1, \ldots, b_{q\ell})) \leq q.$$

Therefore: a minimal solution is a minimal $\Phi_{q,\boldsymbol{w}}$-weighted vector in the row-space of M among those vectors with leading position in $\{1, \ldots, q\}$. By the results of Section III-A, we then conclude:

Proposition 26: A minimal solution $(\hat{\Lambda}_0, \ldots, \hat{\Lambda}_{q-1})$ to the equations of Corollary 25 can be found by bringing $\Psi_{q,\boldsymbol{w}}(M)$ to weak Popov form, and then extracting the row having minimal degree, and in case of a tie least leading position, only among those rows whose leading positions are in $\{\pi(1), \ldots, \pi(q)\}$.

The worst-case complexity of computing M and finding the solution is as in Table III for various choices of the module minimisation algorithm.

Proof: Only the claim on complexity needs to be discussed further. For constructing M we need to compute the sub-matrix T, i.e. for every $t = 1, \ldots, \ell$, we need to compute $\Pi_{R^{(t)}}\Xi$. Computing $R^{(1)}, \ldots R^{(\ell)}$ requires $O^\sim(\ell n)$ by Lemma 6. Due to the structure of $\Pi_{R^{(t)}}$ and Ξ, each element of the matrix product $\Pi_{R^{(t)}}\Xi$ requires at most 3 shifts and additions of the elements of $\Upsilon(R^{(t)})$, possibly followed by a modulo reduction by G, for a total of $O^\sim(n^{4/3})$ operations over $\mathbb{F}_{q^2}$. Thus M can be constructed in any of the complexities stated in Table III.

For module minimising $\Psi_{q,\boldsymbol{w}}(M)$, we should estimate $\gamma = \deg M + \max \boldsymbol{w}/q$ as well as $\Delta(\Psi_{q,\boldsymbol{w}}(M))$. For γ, we have $\deg M = \deg G = n^{2/3}$, while $\max \boldsymbol{w} \leq (q+1)(q-1) + \ell m$. As remarked after Theorem 24, we can assume $\ell m < n$, and so $\gamma \in O(n^{2/3})$.

For $\Delta(\Psi_{q,\boldsymbol{w}}(M)) = \text{rowdeg}\, \Psi_{q,\boldsymbol{w}}(M) - \deg \det \Psi_{q,\boldsymbol{w}}(M)$, we can do better than the generic bound $(\ell+1)q\gamma$: clearly the column permutation performed by $\Psi_{q,\boldsymbol{w}}$ will not affect the orthogonality defect, and so we should compute the orthogonality defect of $M\text{diag}(x^{w_1}, \ldots, x^{w_{q(\ell+1)}})$, where the w_j are the elements of $\boldsymbol{w}$. But M is upper triangular, so the determinant is simply the product of the diagonal. In the orthogonality defect, therefore only the contribution of the first q rows in the row-degree survives, yielding

$$\Delta(\Psi_{q,\boldsymbol{w}}(M)) \leq q \deg T + \sum_{i=1}^{q} w_i < q \deg G + q^3 = 2n.$$

Now the entries of Table III follow from those of Table I, except that a new entry has been added: the Demand–Driven algorithm from [12] for solving "asymmetric 2D Padé approximations". This algorithm is derived from the Mulders–Storjohann algorithm, but only applies to matrices coming from such a 2D Padé approximation. ∎

C. After Having Solved the Key Equation

We will briefly outline how one can finish decoding once a minimal solution $(\hat{\Lambda}_0, \ldots, \hat{\Lambda}_{q-1}, \hat{B}_1, \ldots, \hat{B}_{q\ell}) \in \mathbb{F}_{q^2}[x]^{q(\ell+1)}$ to the equations of Corollary 25 has been found.

Firstly, we apply Υ^{-1} block-wise to obtain $\ell+1$ elements of $\mathcal{R}$: $\hat{\Lambda}, \hat{B}_1, \ldots, \hat{B}_\ell$. Since the $\mathbb{F}_{q^2}[x]$-vector was found in the row-space of M, we know by construction that $\hat{\Lambda}R^{(t)} \equiv \hat{B}_t \mod G$ as a congruence over $\mathcal{R}$ for $t = 1, \ldots, \ell$. Therefore, if it is the case that $\Lambda = \hat{\Lambda}$, then we know by Theorem 24 that $\hat{B}_t \equiv \Lambda f^t \mod G$ as a congruence over $\mathcal{R}$ for any t. For $t = 1$, this congruence can be lifted to equivalence whenever

$$\deg_{\mathcal{H}}(\Lambda f) < \deg_{\mathcal{H}}(G), \qquad \text{i.e.} \qquad |\mathcal{E}| < n - m - g,$$

using Lemma 23. In that case, we simply need to carry out the division B_1/Λ to obtain f: we do this by representing the $\mathcal{R}$ elements as truncated power series in a local parameter at the place $(0,0)$. Conversion to and from such power series are discussed in detail in Appendix B. We choose $\phi = x$ as the local parameter, and we can convert B_1 and Λ into power series in ϕ of precision $2q^3$ in time $O(q^4)$ by Proposition 39 on page 18. Let $\Lambda' = \phi^{-\delta}\Lambda$ and $B_1' = \phi^{-\delta}B_1$ where δ is the greatest power of ϕ that divides Λ; clearly this will also divide B_1 if the correct solution has been found. Since $0 \neq \Lambda \in \mathcal{L}(|\mathcal{E}|P_\infty - \delta(0,0))$ then $\delta < |\mathcal{E}|$ which means we obtain the power series of Λ' and B_1' to at least precision q^3. Using the extended Euclidean algorithm we can calculate ${\Lambda'}^{-1} \bmod x^{q^3}$ in time $O^\sim(q^3)$, and from here $B_1'{\Lambda'}^{-1} \equiv f \mod x^{q^3}$ can be calculated in a further $O^\sim(q^3)$ computations. Finally, converting this truncated power series of f into the standard basis can be done in $O(q^4)$ according to Proposition 42 on page 19.

If we are attempting to decode beyond $n-m-g$, e.g. for extremely low-rate one-point Hermitian codes (see Proposition 30), then it seems that there is no easy way to obtain f from Λ and $(\Lambda f \bmod G)$. An alternative is to find all roots of Λ and erase those positions from $\boldsymbol{r}$, and then perform erasure decoding. We are unaware of a method for doing this in sub-quadratic time, however.

Remark 27: Note that as with the Guruswami–Sudan decoder, we now have a complete decoder which runs in $O^\sim(n^{(2+\omega)/3}\ell^\omega)$, and that the only step of the algorithm with this dominating complexity is module minimisation. Thus, again the hidden constant is *exactly* that of the module minimisation algorithm. We demonstrate in Section VI that also in practice the other steps are quite cheap to compute.

D. Decoding Performance

Power decoding is a probabilistic decoding algorithm in the sense that with non-zero probability it might fail for a given received word $\boldsymbol{r}$, i.e. produce no output. Indeed, since it can decode beyond half the minimum distance but can return only up to one codeword, this is unavoidable. However, by simulation it can be observed that the algorithm almost always works up to a very specific bound: this bound is what one could deem "the decoding radius" of Power decoding the given code.

This overall behaviour is shared by Power decoding of Reed–Solomon codes [24], but the details turn out to be more involved for one-point Hermitian codes. We will in this section characterise this behaviour as well as derive the aforementioned bound. We will repeatedly refer to various events as "likely" or "unlikely": these are based on statistical observations as well as intuition, but unfortunately we have yet no bounds for most of these probabilities. It is important future work, but judging from the simpler case of Reed–Solomon codes, where theoretical results have been obtained only for $\ell = 2, 3$ [5], [24], [27], it is also rather difficult to obtain such bounds.

For this section we will assume that the sent codeword $\boldsymbol{c}$ is uniquely the closest codeword to $\boldsymbol{r}$; indeed, if there is a different codeword closer or as close to $\boldsymbol{r}$, then it is not surprising that Power decoding with high probability fails or decodes erroneously. The following result states that when few errors occur, we are guaranteed to succeed:

Proposition 28: The vector $(\Lambda_0, \ldots, \Lambda_{q-1})$ is a minimal solution to the equations of Corollary 25 whenever $|\mathcal{E}| \leq \frac{d^\star - 1}{2} - \frac{g}{2}$.

Proof: Let $(\lambda_0, \ldots, \lambda_{q-1}, \psi_0, \ldots, \psi_{q-1})$ be a minimal solution to the equations for $\ell = 1$ while $|\mathcal{E}| \leq \frac{n-m}{2} - \frac{g}{2}$, and we will show that $\lambda_i = \gamma\Lambda_i$ for some $\gamma \in \mathbb{F}_{q^2}^*$. Since for $\ell > 1$ we impose further restrictions on the solution set, the analogous statement must then be true. Let $\lambda = \Upsilon^{-1}(\lambda_0, \ldots, \lambda_{q-1})$ and $\psi = \Upsilon^{-1}(\psi_0, \ldots, \psi_{q-1})$. By how the congruence equations and weights for Corollary 25 were derived, we immediately conclude

$$\lambda R^{(1)} \equiv \psi \mod G \tag{11}$$

and $\deg_{\mathcal{H}} \lambda + m + 1 > \deg_{\mathcal{H}} \psi$. Thus, $G \mid (\lambda R^{(1)} - \psi)$ so by Lemma 3 then $\lambda R^{(1)} - \psi \in \mathcal{L}(-\sum_{i=1}^n P_i + \infty P_\infty)$. Introduce $\hat{e} = R^{(1)} - f \in \mathcal{R}$ so $\hat{e}(P_i) = e_i$ for $i = 1, \ldots, n$. Clearly $\hat{e} \in \mathcal{L}(-\sum_{i\notin\mathcal{E}} P_i + \infty P_\infty)$, which means

$$\lambda f - \psi = (\lambda R^{(1)} - \psi) - \lambda\hat{e} \in \mathcal{L}(-\textstyle\sum_{i\notin\mathcal{E}} P_i + hP_\infty),$$

where h is an upper bound on $\deg_{\mathcal{H}}(\lambda f - \psi)$: we can choose $h = \deg_{\mathcal{H}} \lambda + m$. Now we simply want to show that if $\deg_{\mathcal{H}} \lambda \leq \deg_{\mathcal{H}} \Lambda$ then this Riemann–Roch space is $\{0\}$; for in that case $\lambda f = \psi$, so by Equation (11) then $G \mid \lambda(R^{(1)} - f)$, which means $\lambda \in \mathcal{L}(-\sum_{i\in\mathcal{E}} P_i + \infty P_\infty)$; but Λ has minimal $\deg_{\mathcal{H}}$ of non-zero elements in this Riemann–Roch space, and so $\Lambda = \gamma\lambda$ for some $\gamma \in \mathbb{F}_{q^2}$.

We have $\mathcal{L}(-\sum_{i\notin\mathcal{E}} P_i + hP_\infty) = \{0\}$ at least when the defining divisor has negative degree, and since all P_i and P_∞ are rational, this happens when $n - |\mathcal{E}| > h = \deg_{\mathcal{H}} \lambda + m$. Now $\deg_{\mathcal{H}} \lambda \leq \deg_{\mathcal{H}} \Lambda \leq |\mathcal{E}| + g$ by Lemma 23. Therefore, the divisor is negative at least when

$$n - |\mathcal{E}| > |\mathcal{E}| + g + m \qquad \iff \qquad |\mathcal{E}| < \frac{n-m-g}{2} = \frac{d^\star - g}{2}.$$

■

We have the following result for when Power decoding does not fail:

Proposition 29: If Power decoding returns an information polynomial corresponding to the codeword $\hat{\boldsymbol{c}}$, and $\boldsymbol{c}$ is the closest codeword to $\boldsymbol{r}$, then

$$0 \leq \text{weight}(\hat{\boldsymbol{c}} - \boldsymbol{r}) - \text{weight}(\boldsymbol{c} - \boldsymbol{r}) \leq g.$$

Proof: The found solution to the equations of Corollary 25 is minimal, which means that the corresponding error-locator $\hat{\Lambda}$ has minimal $\deg_{\mathcal{H}}$ amongst all solutions; in particular $\deg_{\mathcal{H}} \hat{\Lambda} \leq \deg_{\mathcal{H}} \Lambda$. Let $\hat{\mathcal{E}}$ be the error positions corresponding to $\hat{\boldsymbol{c}}$. Combining the above with Lemma 23 we get:

$$|\mathcal{E}| \leq |\hat{\mathcal{E}}| \leq \deg_{\mathcal{H}} \hat{\Lambda} \leq \deg_{\mathcal{H}} \Lambda \leq |\mathcal{E}| + g$$

and the proposition follows. ∎

Ideally, we would have hoped that when Power decoding returns a codeword, this is always the closest. Indeed, that is true for Power decoding of Reed–Solomon codes. The above states that for one-point Hermitian codes in general, a codeword slightly farther away can actually have the smaller error locator, which will then be found instead. However, simulations indicate that for random error patterns, the error locator most likely has the maximal order $|\mathcal{E}| + g$; most likely, the error locator for either codeword will satisfy this, and so the closest codeword will again have the lowest-order error locator. The probability of the errors lying such that $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g$ was shown to be $1/q$ asymptotically [28], [29].

Finally, we will discuss how many errors we should expect Power decoding to be able to cope with. Recall M of (10) on page 10 whose row space contains all $\mathbb{F}_{q^2}[x]$-vectors satisfying the congruence equations of Corollary 25. The following result puts an upper bound on the $\deg_{\mathcal{H}}$ of the λ-part of any vector in the row space of M:

Proposition 30: Let $\boldsymbol{s} = \Phi_{\nu,\boldsymbol{w}}(\lambda_0, \ldots, \lambda_{q-1}, b_1, \ldots, b_{q\ell})$ be the minimal degree vector in the row space of $\Phi_{\nu,\boldsymbol{w}}(M)$. Then

$$\deg_{\mathcal{H}} \left(\Upsilon^{-1}(\lambda_0, \ldots, \lambda_{q-1})\right) \leq \tau_{\text{Pow}}(\ell) \triangleq \tfrac{\ell}{\ell+1} n - \tfrac{1}{2}\ell m - \tfrac{\ell}{\ell+1}.$$

Proof: If $\Phi_{\nu,\boldsymbol{w}}(M')$ is a matrix unimodular equivalent with $\Phi_{\nu,\boldsymbol{w}}(M)$ and in weak Popov form, then by Proposition 9 there must be a row $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s}')$ of $\Phi_{\nu,\boldsymbol{w}}(M')$ with $\deg \Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s}') = \deg \Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s})$. We have

$$\begin{aligned} \text{rowdeg}\left(\Phi_{\nu,\boldsymbol{w}}(M')\right) &= \deg \det(\Phi_{\nu,\boldsymbol{w}}(M')) \\ &= \deg \det(\Phi_{\nu,\boldsymbol{w}}(M)) \\ &= \ell q n + \textstyle\sum_i \bar{\eta}_i + \sum_j \bar{\mu}_j, \end{aligned}$$

where $\boldsymbol{w} = (\bar{\eta}_0, \ldots, \bar{\eta}_{q-1}, \bar{\mu}_1, \ldots, \bar{\mu}_{q\ell})$ as specified in Section V-B. Inserting and simplifying, the right-hand side becomes

$$q\ell n + (\ell+1)\tbinom{q}{2} + q(\ell+1)\big(g - \tfrac{1}{2}\ell m - \tfrac{\ell}{\ell+1} + (\ell m + 1)\big). \tag{12}$$

Clearly $\deg \Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s}') \leq \frac{1}{q(\ell+1)} \text{rowdeg}\left(\Phi_{\nu,\boldsymbol{w}}(M')\right)$, but we can do slightly better due to the sparsity of the polynomials in $\Phi_{\nu,\boldsymbol{w}}(M')$: notice that for any h in $0, \ldots, q-1$, there is exactly one i such that $\eta_i \equiv h \mod q$, and there are exactly ℓ indices j such that $\mu_j \equiv h \mod q$. Also note that degree of a given row of $\Phi_{\nu,\boldsymbol{w}}(M')$ must be congruent modulo q to the weight applied at the leading position. Let $\bar{h} = (\deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s})) \bmod q)$. For any h in $0, \ldots, q-1$, since $\Phi_{\nu,\boldsymbol{w}}(M')$ is in weak Popov form, there are therefore $\ell + 1$ rows whose degree is congruent to h modulo q. For such a row $\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{m}_j)$ we therefore have

$$\deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{m}_j)) \geq \deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s})) + ((h - \bar{h}) \bmod q),$$

where the modulo representative is taken in $0, \ldots, q-1$. Summing over all rows we get

$$(\ell+1)\deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s})) + (\ell+1)\tbinom{q}{2} \leq \text{rowdeg}\, \Phi_{\nu,\boldsymbol{w}}(M').$$

Finally, by the choice of $\eta_0, \ldots, \eta_{q-1}$, we have

$$\deg_{\mathcal{H}}(\Upsilon^{-1}(\lambda_0, \ldots, \lambda_{q-1})) + \ell m + 1 \leq \deg(\Phi_{\nu,\boldsymbol{w}}(\boldsymbol{s}')).$$

Combining these inequalities gives the result. ∎

The above result therefore states that if $|\mathcal{E}| \geq \tau_{\text{Pow}}(\ell)$ then there are shorter vectors in the row space of M than $\boldsymbol{\Lambda} = (\Lambda, \Lambda f, \ldots, \Lambda f^{\ell})$. These short vectors might not have a leading position within $1, \ldots, q$ as we require from a solution, and $\boldsymbol{\Lambda}$ might still be the shortest vector satisfying this requirement. However, it seems reasonable to expect that the shortest vector with leading position within $1, \ldots, q$ usually does not have much higher degree than the unconditionally shortest vector: indeed, experiments confirm this, and Power decoding fails almost always when $|\mathcal{E}| \geq \tau_{\text{Pow}}(\ell)$. See Table IV. When it does succeed anyway, this is usually because $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g$ as previously discussed.

There is a small caveat to the above discussion: it only holds when ℓ is chosen less than or equal to the value which maximises $\tau_{\text{Pow}}(\ell)$. For $\ell \to \infty$ then $\tau_{\text{Pow}}(\ell) \to -\infty$. However, clearly having more key equations is not going to *add* solution vectors, so if, say $(\Lambda, \ldots, \Lambda f^{\ell})$ is the minimal solution choosing some ℓ, then clearly $(\Lambda, \ldots, \Lambda f^{\hat{\ell}})$ is the minimal solution when choosing any $\hat{\ell} \geq \ell$.

Assuming this choice of ℓ then whenever $|\mathcal{E}| \leq \tau_{\text{Pow}}(\ell)$, we will most likely succeed and find Λ. Unfortunately, we do not have an upper bound on the probability that we fail. However, our simulations indicate that this probability is low and

exponentially quickly decays as $|\mathcal{E}|$ fall; see Section VI. This is also the case for Power decoding of Reed–Solomon codes, where a proof of these observations is only known for $\ell = 2, 3$ [5], [24], [27].

Recall again from Section V-C that even when the key equation is solved correctly, we are only able to extract f from Λ and $(\Lambda f \bmod G)$ when $|\mathcal{E}| < n - m - g$. When $m \ll n$ it is possible that $\tau_{\text{Pow}}(\ell) > n - m - g$.

Remark 31: For $\ell = 1$, i.e. minimum distance decoding, then Proposition 30 indicates that we will probably succeed when $|\mathcal{E}| \leq \frac{n-m-1}{2} = \frac{d^\star - 1}{2}$, while Proposition 28 only promises success when $|\mathcal{E}| \leq \frac{d^\star - 1 - g}{2}$. This is an interesting, well-known caveat of "pure" key equation decoding of AG codes: we are only *assured* decoding success until $g/2$ less than $(d^\star - 1)/2$, but *almost always*, decoding will succeed all the way until $(d^\star - 1)/2$. The authors are unaware of any work investigating this classical failure probability. One can be assured of success all the way to $(d^\star - 1)/2$ using the majority voting technique of Feng et al. [30]; it is yet unclear whether this technique can be combined with the fast module minimisation and with Power decoding.

VI. Simulation Results

The proposed algorithms have been functionally implemented in Sage v6.4 [7] and can be downloaded at www.jsrn.dk/code-for-articles. The implementation includes basic manipulation of the codes and objects, the fast root-finding and all $\mathcal{R}$ conversions. It does not include either of the fast module minimisation algorithms GJV [4] or Zhou–Labahn [31], but instead accomplishes module minimisation using the simpler Mulders–Storjohann algorithm [14]. The map $\Psi_{\nu,\boldsymbol{w}}$ described in Section III-A for handling the weights efficiently has also been implemented. All parts of our implementation but the module minimisation therefore runs in the asymptotic complexities reported in this paper, though they – being high-level implementations – might not have the lowest possible hidden constant.

The implementations allow us to investigate to some degree two concerns which seem difficult to approach analytically: the failure probability of the decoders, and a breakdown of the speed of the various parts of the decoders on concrete parameters. For the latter, we can – of course and unfortunately – say little on the speed of fast module minimisation algorithms.

A. Failure Probability

We gave in Proposition 30 a bound $\tau_{\text{Pow}}(\ell)$ on how many errors we should expect Power decoding to correct, and conversely, using intuition from linear algebra, we might expect that any number of errors below this will usually be correctable. This intuition is confirmed by our simulations, which indicate that when $|\mathcal{E}| < \tau_{\text{Pow}}(\ell)$ decoding failure is unlikely, with a probability that quickly decays as $|\mathcal{E}|$ falls. Table IV summarises simulation results for two different codes. In the table, for each set of code and decoder parameters, and for each number of errors ϵ, 1000 random codewords were generated and submitted to a random error of Hamming weight exactly ϵ and attempted decoded.

It was already observed by Lee and O'Sullivan [1] that Guruswami–Sudan will usually succeed in correcting errors well beyond the guaranteed bound $\tau_{\text{GS}}(s, \ell)$ from Section IV, but they gave no description on how much beyond to expect. Observe that $\tau_{\text{Pow}}(\ell)$ is exactly $g - \ell/(\ell + 1)$ greater than the lower bound on $\tau_{\text{GS}}(1, \ell)$ given in Equation (3). As can be seen on Table IV, our simulations indicate that $\tau_{\text{GS}}(\ell) + g$ is exactly the bound one should also expect that Guruswami–Sudan will decode up to, when $s = 1$. More generally, there is also an indication that we can expect Guruswami–Sudan to succeed for at least $\tau_{\text{GS}}(s, \ell) + g/s$, but more simulations should be carried out to verify this.

For the $q = 4$ code, note how the success probability at $\tau + 1$ errors is very close to $q^{-2} = 6.25\%$. As previously discussed, this is exactly the asymptotic (for $q \to \infty$) probability that $\deg_{\mathcal{H}} \Lambda < |\mathcal{E}| + g$ [28], [29], in which case we due to Proposition 30 should expect Power decoding to succeed. The success probability seems better at $\tau + 1$ for the $q = 5$ code, where $q^{-2} = 4\%$.

For a given ℓ and $s = 1$, it is a natural question whether there is a correspondence between the cases where Power decoding fails and where Guruswami–Sudan does, for $|\mathcal{E}| \leq \tau_{\text{Pow}}(\ell)$. It surprised us that we observed no such correspondence: when Power decoding fails, Guruswami–Sudan often succeeds, and vice versa!

B. Speed

In our implementation, the running time for both decoders is completely dominated by module minimisation. Of course, one should recall that our implementations are asymptotically fast in all parts except the module minimisation, where we are using Mulders–Storjohann, so asymptotically, we should expect exactly such a dominance. However, it is still possible to get an impression on how demanding each part of the decoding algorithms is. Table VI-B shows a breakdown for the time spent on the various parts of the algorithms, using the $[343, 35, \geq 288]$ code having $q = 7, m = 55$ and $g = 21$.

The reported speeds are the median over 10 trials for each set of parameters. After module minimisation, Power decoding must perform the division of Λf with Λ as described in Section V-C, while root-finding is performed for Guruswami–Sudan. "Conversions" denote time used in converting between the representations of $\mathcal{R}$ elements, as described in Appendix B. Precomputation refers to G, and various polynomials for Lagrange interpolation as well as for $\mathcal{R}$ conversion. These simulations were run on a laptop with a Core Intel i7-4600U @ 2.1 GHz processor and 8 GB DDR3 1.6 GHz RAM.

Success probability for the $[64, 10, \geq 49]$ code, with $q = 4, m = 15$ and $g = 6$

	$\tau_{\text{GS}}(s, \ell)$	$\tau_{\text{Pow}}(\ell)$	τ	$P_s(\tau - 2)$	$P_s(\tau - 1)$	$P_s(\tau)$	$P_s(\tau + 1)$
GS $(s, \ell) = (1, 1)$	18	24	24	100%	100%	100%	6.1%
Power $\ell = 1$	—	24	24	100%	100%	100%	6.2%
GS $(s, \ell) = (1, 2)$	21	27	27	100%	100%	93.9%	6.5%
Power $\ell = 2$	—	27	27	100%	100%	94.9%	6.2%
GS $(s, \ell) = (2, 4)$	26	—	29	100%	100%	99.3%	6.5%

Success probability for the $[125, 11, \geq 105]$ code, with $q = 5, m = 20$ and $g = 10$

	$\tau_{\text{GS}}(s, \ell)$	$\tau_{\text{Pow}}(\ell)$	τ	$P_s(\tau - 2)$	$P_s(\tau - 1)$	$P_s(\tau)$	$P_s(\tau + 1)$
GS $(s, \ell) = (1, 2)$	53	62	63	100%	99.8%	96.4%	4.5%
Power $\ell = 2$	—	62	62	100%	100%	100%	7.2%
GS $(s, \ell) = (1, 3)$	54	63	64	100%	100%	96.1%	5.1%
Power $\ell = 3$	—	63	63	100%	100%	100%	8.5%

Table IV

IN THE ABOVE, $\tau = \tau_{\text{Pow}}(\ell)$ FOR POWER DECODING WHILE $\tau = \tau_{\text{GS}}(s, \ell) + g/s$ FOR GURUSWAMI–SUDAN. $P_s(t)$ DENOTES THE OBSERVED PROBABILITY OF DECODING SUCCESS WHEN EXACTLY t ERRORS IS ADDED.

Speed results for the $[343, 35, \geq 288]$ code, with $q = 7, m = 55$ and $g = 21$

	Power $\ell = 1$	Power $\ell = 2$	GS $(s, \ell) = (1, 2)$	GS $(s, \ell) = (2, 4)$
No. of errors	143	173	173	185
Module minimisation	2.25 s	5.95 s	9.36 s	177 s
Division / Root-finding	0.26 s	0.27 s	0.36 s	0.74 s
Build matrix	0.09 s	0.19 s	0.15 s	0.61 s
Conversions	0.05 s	0.05 s	0.02 s	0.06 s
Precomputation	0.01 s	0.01 s	0.01 s	0.02 s
Total time	2.6 s	6.4 s	9.9 s	178 s

We have executed our decoders with various parameters: Power decoding with $\ell = 1$ (i.e. classical key equation decoding) and with $\ell = 2$, and Guruswami–Sudan with $(s, \ell) = (1, 2)$ and $(s, \ell) = (2, 4)$. In all cases, we have run the decoder on the maximal probably decodable number of errors, as discussed in the preceding section. The received words where decoding failed were discarded from the statistics.

As mentioned, module minimisation completely dominates. Though we can not draw too final conclusions without an implementation of the asymptotically fast module minimisation algorithm GJV or Zhou–Labahn, even with this algorithm the cost of module minimisation will likely dominate the cost, for even medium sized codes such as this. In particular, as also predicted by the asymptotic analyses, the cost of conversion between the representations of $\mathcal{R}$ elements is highly unlikely to have a significant impact on the total running time.

As is known to be the case for Guruswami–Sudan decoding of Reed-Solomon codes, it seems that also in our case, the root finding is cheaper than the interpolation step. We can furthermore add that our implementation of Alekhnovich's fast root finding out-performs our implementation of the Roth–Ruckenstein root finding [22] already when the $x\deg$ of the input polynomial exceeds 100.

VII. CONCLUSION

In this paper, we have demonstrated that decoding of one-point Hermitian codes in sub-quadratic complexity is possible: we describe two decoding algorithms, both of which are able to decode beyond the classical $(d^\star - g)/2$ bound. The main ingredient was to employ recent and deep results in computer algebra for the general problem of $\mathbb{F}_{q^2}[x]$-module minimisation, combined with a new embedding of the original $\mathcal{R}$ problem from the function field.

The core of both the Guruswami–Sudan and the Power decoding algorithms seem fairly resilient to the exact function field employed. We expect in particular that the methods can be extended to one-point codes over any plane Miura-Kamira curve [32] with fairly few changes. Surprisingly, particular properties of the Hermitian curve, in particular that its equation has only few monomials, were important for attaining sub-quadratic complexity in the auxiliary computations regarding conversion to and from power series; these conversions were necessary for our solutions to the root-finding step in Guruswami–Sudan as well as the post-processing after having solved the key equations in Power decoding.

The decoding algorithms have been functionally implemented in Sage v6.4 [7] and can be downloaded at www.jsrn.dk/code-for-articles.

ACKNOWLEDGEMENTS

J. S. R. Nielsen gratefully acknowledges the support of the Digiteo foundation, project IdealCodes. Part of this work was also done while he was with Ulm University, and he gratefully acknowledges the support from the German Research Council under grant BO 867/22-1. P. Beelen gratefully acknowledges the support from The Danish Council for Independent Research (Grant No. DFF–4002-00367).

APPENDIX A
ROOT-FINDING IN $\mathcal{H}[z]$

For Guruswami–Sudan decoding of one-point Hermitian codes in Section IV, we need to efficiently find all roots of $Q \in \mathcal{H}[z]$ whose pole order at P_∞ is less than m. In [19] it was already shown how to solve this problem using the Roth–Ruckenstein algorithm [22] for finding $\mathbb{F}_{q^2}[x]$ roots of polynomials in $\mathbb{F}_{q^2}[x][z]$ by adopting a power series view. We will now show how one can instead apply Alekhnovich's Divide & Conquer variant [3] of the Roth–Ruckenstein algorithm in order to achieve a sub-quadratic complexity in n. The core is a straight-forward power series description of the algorithm of [3], though with a tighter complexity analysis, but for clarity and completeness, we show and prove the complete algorithm.

Consider the rational place $(0,0)$: a local parameter for this place is $\phi = x$. Elements of $\mathcal{H}$ have no poles at $(0,0)$, so any $h \in \mathcal{L}(mP_\infty)$ can be written as a power series in ϕ: $h = \sum_{i=0}^{\infty} h_i \phi^i \in \mathbb{F}_{q^2}[\![\phi]\!]$. Likewise, we can write $Q = \sum_{t=0}^{\ell} z^t \sum_{i=0}^{\infty} q_{t,i}\phi^i$.

Lemma 32: For any $Q \in \mathcal{H}[z]$, consider some $h \in \mathcal{L}(mP_\infty)$ satisfying

$$Q(h) \equiv 0 \mod \phi^k,$$

for some integer $k > \deg_{\mathcal{H},m}(Q)$ when $Q(h)$ is expanded into a power series in ϕ. Then $Q(h) = 0$.

Proof: If $Q(h) \neq 0$ then clearly $\deg_{\mathcal{H}}(Q(h)) \leq \deg_{\mathcal{H},m}(Q)$. Together with the congruence we conclude $Q(h) \in \mathcal{L}\big(\deg_{\mathcal{H},m}(Q)P_\infty - k(0,0)\big)$. The requirement on k ensures that this Riemann–Roch space contains only 0. ∎

The strategy is then to iteratively describe all truncated power series $h = \sum_{i=0}^{d_h} h_i\phi^i + O(\phi^{d_h+1})$ such that $Q(h) \equiv 0 \mod \phi^k$ for increasing k until $k > \deg_{\mathcal{H},m}(Q)$. From this set, those that can be extended into functions in $\mathcal{L}(mP_\infty)$ must be unconditional roots of Q. We use the power series conversion detailed in Appendix B to convert these roots into functions in the standard basis. To achieve a quasi-linear dependence on $\deg_{\mathcal{H},m}(Q)$, the iterative increments of k are structured in a divide-and-conquer tree.

Definition 33: For any non-zero $Q \in \mathbb{F}_{q^2}[\![\phi]\!][z]$, and some $k \in \mathbb{Z}_+$, by *the roots of Q of order k*, we will mean the set of $h \in \mathbb{F}_{q^2}[\![\phi]\!]$ such that $Q(h) \equiv 0 \mod \phi^k$.

The following lemma is an easy extension of [3, Lemma A.1.1], which in turn was inspired by the analysis of [22, Section 6]:

Lemma 34: Let A be the roots of Q of order k for any non-zero $Q \in \mathbb{F}_{q^2}[\![\phi]\!][z]$ and $k \in \mathbb{N}_0$, Then A can be partitioned into $\hat{\ell} \leq \deg_z(Q|_{\phi=0})$ many sets $A_1, \ldots, A_{\hat{\ell}}$ of the form $A_i = h_i + \phi^{d_i}\mathbb{F}_{q^2}[\![\phi]\!]$ for some $h_i \in \mathbb{F}_{q^2}[\phi]$ and $d_i \in \mathbb{Z}_+$.

Proof: If $Q|_{\phi=0} = 0$ then write $Q = \phi^s \grave{Q}$ where $\grave{Q}|_{\phi=0} \neq 0$. Then the roots of order k of Q are exactly the roots of order $k-s$ of $\grave{Q}$. Assume therefore that $Q|_{\phi=0} \neq 0$.

We proceed then by induction on k. For the base case $k = 1$, let $z_1, \ldots, z_{\hat{\ell}}$ be the roots of $Q|_{\phi=0} \in \mathbb{F}_{q^2}[z]$. Clearly $\hat{\ell} \leq \deg_z(Q|_{\phi=0})$, and any $h \in A$ will be of the form $h = z_i + O(\phi)$ for one of the z_i.

For the inductive case at $k > 1$, let $z_1, \ldots, z_{\hat{\ell}}$ be the roots of $Q|_{\phi=0} \in \mathbb{F}_{q^2}[z]$. As before, $\hat{\ell} \leq \deg_z(Q|_{\phi=0})$, and any $h \in A$ will be of the form $h = z_i + O(\phi)$ for one of the z_i. Furthermore, let $Q_i = \phi^{-s_i}Q(z_i + \phi z)$ where $s_i \geq 1$ is the greatest integer such that $\phi^{s_i} \mid Q(z_i + \phi z)$. It must then be the case that $h = z_i + \phi\grave{h}$ for some $\grave{h} \in A_i$, where A_i is the set of roots of Q_i of order $k - s_i$. By the induction hypothesis, A_i can be partitioned into $A_{i,1}, \ldots, A_{i,\hat{\ell}_i}$ of the appropriate form, where $\hat{\ell}_i = \deg_z(Q_i|_{\phi=0})$. We can extend each of these $\hat{\ell}_i$ sets as $A'_{i,j} = z_i + \phi A_{i,j}$ and then $h \in A'_{i,j}$ for some j. Thus clearly, A can be partitioned into $A_1, \ldots A_L$ of the appropriate form, where $L = \sum_{i=1}^{\hat{\ell}} \hat{\ell}_i$.

The lemma then follows if we can prove $L \leq \hat{\ell}$; this in turn follows by showing that $\deg_z(Q_i|_{\phi=0}) \leq m_i$ where m_i is the multiplicity of the zero z_i in $Q|_{\phi=0}$. We show that by writing $Q = (z - z_i)^{m_i}P_i + \phi\hat{Q}_i$, where $P_i \in \mathbb{F}_{q^2}[z]$ with $P_i(z_i) \neq 0$ and $\hat{Q}_i \in \mathbb{F}_{q^2}[\![\phi]\!][z]$. Then

$$\phi^{s_i}Q_i = (\phi z)^{m_i}P_i(z_i + \phi z) + \phi\hat{Q}_i(z_i + \phi z).$$

All terms on the right-hand side have ϕ-degree at least that of the z-degree, which means $\deg_z(Q_i|_{\phi=0}) \leq s_i$. But $s_i \leq m_i$ since the above right-hand side has the term $(\phi z)^{m_i}P_i(z_i)$, and this can not cancel with any term in $\phi\hat{Q}_i(z_i + \phi z)$ since these have greater ϕ-degree than z-degree. ∎

Proposition 35: Algorithm 1 is correct.

Proof: We proceed by induction on k. If $k = 1$, clearly the algorithm is correct. Now for the inductive step: each root of Q of order $k_\perp = \lceil k/2 \rceil$ will be of the form $h_i + \phi^{d_i}h'_i$ for some $h'_i \in \mathbb{F}_{q^2}[\![\phi]\!]$ for one of the iterations (h_i, d_i). This means $Q(h_i + \phi^{d_i}h'_i) \equiv 0 \mod \phi^{k_\perp}$ for any h'_i, which is only possible when $\phi^{k_\perp} \mid Q(h_i + \phi^{d_i}z)$, implying $s_i \geq k_\perp$ in Line 8 for this iteration.

Now for any h'_i, if $h_i + \phi^{d_i}h'_i$ is a root of Q of order k, then $\phi^k \mid Q(h_i + \phi^{d_i}z)|_{z=h'_i}$, i.e. $\phi^{k-s_i} \mid \hat{Q}(z)|_{z=h'_i}$, and so h'_i is a root of $\hat{Q}$ of order $k - s_i$. Again by the induction hypothesis $\{(h_{i,j}, d_{i,j})\}$ represents all such roots. Therefore, all roots of Q of order k are returned in Line 12. ∎

Algorithm 1 Roots: root-finding in $\mathbb{F}_{q^2}[\![\phi]\!][z]$

Input: $Q \in \mathbb{F}_{q^2}[\![\phi]\!][z], k \in \mathbb{Z}_+$
Output: The roots of Q of order k in disjoint sets as in Lemma 34, as a set of pairs $(h,d) \in \mathbb{F}_{q^2}[\phi] \times \mathbb{Z}_+$

$Q \leftarrow Q \bmod \phi^k$
if $k = 1$ **then**
  if $Q = 0$ **then return** $\{(0,0)\}$
  $z_1, \dots, z_{\hat\ell} \leftarrow z$-roots of $Q \in \mathbb{F}_{q^2}[z]$
  return $\{(z_i, 1)\}_{i=1}^{\hat\ell}$
else
  for $(h_i, d_i) \in \mathsf{Roots}(Q, \lceil k/2 \rceil)$ **do**
    $\hat Q \leftarrow Q(h_i + \phi^{d_i} z)/\phi^{s_i}$
      where s_i is maximal such that $\hat Q \in \mathbb{F}_{q^2}[\![\phi]\!][z]$
    $\{(h_{i,j}, d_{i,j})\}_{i=1}^{\hat\ell_i} \leftarrow \mathsf{Roots}(\hat Q, k - s_i)$
  end for
  return $\{(h_i + \phi^{d_i} h_{i,j}, d_i + d_{i,j})\}_{i,j}$
end if

Proposition 36: The complexity of Algorithm 1 is $O\tilde{}(\ell^2 k)$, where $\ell = \deg_z Q$, assuming $q^2 \in O(k)$.

Proof: Denote the complexity of the algorithm on input Q with $\deg_z Q = \ell$ and $\deg_z(Q|_{\phi=0}) = \hat\ell$ by $T_\ell(k, \hat\ell)$. Note that in none of the recursive calls can we then have $\deg_z Q > \ell$. Now, $T_\ell(1, \hat\ell) = O(\hat\ell \mathrm{P}(\hat\ell)\log(q\hat\ell))$, being the complexity of univariate root-finding using e.g. [33, Chapter 8.9], where $\mathrm{P}(n) \in O\tilde{}(n)$ denotes the complexity of multiplying two polynomials over $\mathbb{F}_{q^2}$ of degree at most n [34, Theorem 8.23].

For larger k, the main loop will have complexity

$$T_\ell(k, \hat\ell) = T_\ell(\lceil k/2 \rceil, \hat\ell) + \sum_{i=1}^{\hat\ell} \big(S_\ell(k - s_i) + T_\ell(k - s_i, m_i)\big),$$

where $S_\ell(k')$ is the cost of computing $\hat Q = Q(h_i + \phi^{d_i})$ when Q is given to precision $\phi^{k'}$, and where the m_i are as in the proof of Lemma 34. Recall that $m_1 + \ldots + m_{\hat\ell} \le \hat\ell \le \ell$.

To estimate $S_\ell(k')$, then let $Q = Q_\perp + z^{\ell'} Q_\top$, where ℓ' is the greatest power of 2 less than ℓ, and $\deg Q_\perp < \ell'$. Then

$$\begin{aligned}\phi^{s_i}\hat Q &= Q(h_i + \phi^{d_i} z) \\ &= Q_\perp(h_i + \phi^{d_i} z) + (h_i + \phi^{d_i} z)^{\ell'} Q_\top(h_i + \phi^{d_i} z).\end{aligned}$$

After precomputation of $(h_i + \phi^{d_i} z)^{2^h}$ for all $h < \log_2(\ell)$, we can compute the product of $(h_i + \phi^{d_i} z)^{\ell'}$ and $Q_\top(h_i + \phi^{d_i} z)$ in complexity $\mathrm{P}(\ell)\mathrm{P}(k')$. Thus we get

$$S_\ell(k') = 2S_{\ell/2}(k') + O(k') + \mathrm{P}(\ell)\mathrm{P}(k'),$$

which my the master theorem [35] has the solution $S_\ell(k') = O\tilde{}(\ell k')$.

Back to $T_\ell(k, \hat\ell)$, it is easy to see that the complexity is increasing at least linearly in both k and $\hat\ell$. That means that $\sum_{i=1}^{\hat\ell} T_\ell(k - s_i, m_i) \le T_\ell(k/2, \hat\ell)$, since the m_i sum to at most $\hat\ell$ and since $s_i \ge k/2$. Thus, $T_\ell(k, \hat\ell) \le \hat\ell S_\ell(k/2) + 2T_\ell(k/2, \hat\ell)$, which by the master theorem has the solution

$$T_\ell(k, \hat\ell) \in O(\hat\ell S_\ell(k)\log k + kT_\ell(1, \hat\ell)),$$

whence $T_\ell(k, \ell) \in O\tilde{}(\ell^2 k)$. ∎

Corollary 37: Given a $Q \in Я[z]$ whose coefficients are in the standard basis, we can compute all $f \in \mathcal{L}(mP_\infty)$ in the standard basis such that $Q(f) = 0$ in complexity

$$O\tilde{}(\ell^2 k + \ell k^2/q^2 + \ell q^4),$$

where $k = \deg_{\mathcal{H},m} Q$ and $\ell = \deg_z Q$.

Proof: By Lemma 32 we need to set $k = \deg_{\mathcal{H},m} Q + 1$ in Algorithm 1, and by Lemma 34 we will be returned a list of at most ℓ sets of roots. The cost of the main algorithm will therefore be $O\tilde{}(\ell^2 k)$.

Remaining is conversion of input and output. We convert Q into an element of $\mathbb{F}_{q^2}[\![\phi]\!][z]$ up to precision k using Proposition 39 for each Я-coefficient, which we can do in $O(\ell k^2/q^2)$. The root sets returned by the root-finding algorithm need to be converted back into the standard basis. Note that we do not a priori know the precision of these roots; in particular, whether each have $d_h > m$ so that unique conversion into $\mathcal{L}(mP_\infty)$ is guaranteed by Lemma 41. However, even if multiple $\mathcal{L}(mP_\infty)$-element

arise from some of the root sets, then each possible element obtained must be an unconditional root of Q by Lemma 32, and we know that there can be at most ℓ such roots in $\mathcal{R} \supset \mathcal{L}(mP_\infty)$. Thus in total, we will spend $O(\ell q^4)$ on converting the output roots, by Proposition 42. ∎

APPENDIX B
POWER SERIES CONVERSION

For both Guruswami–Sudan decoding as well as Power decoding, we need to efficiently convert $\mathcal{R}$ elements between the standard basis and truncated power series descriptions. More precisely, let ϕ denote a local parameter for the place $(0,0)$; we will in fact choose $\phi = x$. We will describe efficient algorithms to do the following: Given a sufficiently long truncated power series development of an element $f \in \mathcal{L}(mP_\infty)$ in ϕ, compute f; and given f compute its truncated power series expansion in ϕ. We will show that we can solve both of these problems reasonably efficiently.

Our usual representation of elements in $\mathcal{R}$ is an $\mathbb{F}_{q^2}$-combination of the elements in the standard basis:

$$S = \{x^i y^j \mid i \geq 0,\, 0 \leq j \leq q-1\}.$$

Let $S_m \subset S$ denote the elements of S with $\deg_{\mathcal{H}}$ at most m, i.e. S_m is a basis for $\mathcal{L}(mP_\infty)$.

We begin with showing a structural sparsity of $x^i y^j$ monomials when expressed as power series in ϕ:

Lemma 38: Let i and j be nonnegative integers. In the power series expansion of $x^i y^j$ in ϕ up to some precision $N \leq q^3$, there are at most q nonzero coefficients. If $j < q$ then for any N, there are at most N/q^2 nonzero coefficients.

Proof: First of all, note that a power series development of y in x can be obtained using $y = x^{q+1} - y^q$. Iterating this equation, one obtains that

$$y = \sum_{b=0}^{\infty} (-1)^b \phi^{(q+1)q^b}. \tag{13}$$

Every term in y^j must therefore have a ϕ-degree of the form $(q+1)\sum_{b=0}^{\infty} a_b q^b$ where the a_b are non-negative integers with $\sum_{b=0}^{\infty} a_b = j$. Let $N' = N/(q+1)$ and $r = \lfloor \log_q N' \rfloor$. In the truncation of y^j to precision N the number of terms is then at most the number of tuples $(a_0, \ldots, a_r)$ such that $\sum_{b=0}^{r} a_b q^b < N'$ and $a_0 + \ldots + a_r = j$.

For the lemma's first claim, if $N \leq q^3$ then $N' < q^2$ and $r \leq 1$. Since $a_1 > q$ implies $a_0 + a_1 q > N'$ that leaves at most q possible tuples $(j - a_1, a_1)$ for $a_1 = 0, \ldots, \min(q-1, j)$.

For the second claim, assume $j < q$, and therefore also $a_b < q$ for all b. Note that $\sum_{b=0}^{r} a_b q^b$ is then basically some number less than N' written in base q, so we are counting how many numbers less than N' have a digit sum exactly j. We can upper bound that count by counting those numbers with a digital root exactly j, which is $\lceil N'/q \rceil$ of the numbers. In total, there must be at most $\lceil N/(q+1)/q \rceil \leq N/q^2$ non-zero terms in the power series expansion of y^j up to precision N. Since $x = \phi$, the same holds for $x^i y^j$. ∎

This immediately implies that it is fast to convert elements from S_m *into* power series:

Proposition 39: Given $f \in \mathcal{L}(MP_\infty)$ described in the basis S, we can compute a power series expansion in ϕ up to precision N in complexity $O(MN/q^2)$.

Proof: f is the linear combination of at most M monomials $x^i y^j$, so the power series can be computed by scaling and summing each of these monomial's power series. The claim then follows from Lemma 38. ∎

For conversion *from* power series it turns out that a useful stepping stone is a slightly different basis than S_m:

Lemma 40: Let m be an integer at most q^3. The set

$$\hat{S}_m = \{x^i y^j \mid qi + (q+1)j \leq m,\, 0 \leq i \leq q,\, j \geq 0\}$$

is a basis for $\mathcal{L}(mP_\infty)$. Moreover, any element of $\hat{S}_m$ can be expressed as a linear combination of at most $q+1$ elements from S_m.

Proof: That $\hat{S}_m$ is a basis for $\mathcal{L}(mP_\infty)$ is clear: writing an element of $\mathcal{R} = \mathbb{F}_{q^2}[x,y]$ as a polynomial, the equation of the Hermitian curve $y^q + y = x^{q+1}$ has simply been used to reduce the x-degree (where for S, the y-degree was reduced).

Now let $x^i y^j \in \hat{S}_m$. We wish to express it as a linear combination of elements from S_m. It is sufficient to show that y^j with $0 \leq j \leq m/(q+1)$ can be expressed as such a linear combination. First of all write $j = a + bq$ for unique, nonnegative integers a and b at most $q-1$. Then we have

$$y^j = y^a \left(y^q\right)^b = y^a (x^{q+1} - y)^b. \tag{14}$$

If $a + b \leq q - 1$, this is clearly an expression of y^j as a linear combination of at most $b + 1 \leq q$ elements in S_m. If on the other hand $a + b \geq q$, we write $(x^{q+1} - y)^b = p_1 + y^{q-a} p_2$, where $\deg_y p_1 < q - a$ and $\deg_y p_2 \leq a + b - q$, and where both p_1 and $y^{q-a} p_2$ are homogeneous in the expressions x^{q+1} and y. Now

$$y^j = y^a \left(p_1 + y^{q-a} p_2\right) = y^a p_1(x,y) + (x^{q+1} - y) p_2. \tag{15}$$

Note that $\deg_y(y^a p_1) < q$, but that also $\deg_y((x^{q+1} - y)p_2) \leq a + b - q + 1 < q$. Therefore equation (15) gives the desired expression of y^j as linear combination of elements in S_m, and we need estimate only the number of elements in this combination. But both p_1 and $(x^{q+1} - y)p_2$ are homogeneous polynomials in x^{q+1} and y so the number of monomials occurring in each of them is at most their y-degree plus one. This gives a total of at most $(q - a) + (a + b - q + 2) = b + 2 \leq q + 1$ monomials. ■

The above lemma shows that we can convert any function $f \in \mathcal{L}(mP_\infty)$ expressed in the basis $\hat{S}_m$ into the basis S_m in complexity $O(mq) \subset O(q^4)$.[2]

Lemma 41: Suppose that $f \in \mathcal{L}(mP_\infty)$ with $m < q^3$, and that $m + 1$ values $a_i \in \mathbb{F}_{q^2}$ are given such that $f = \sum_{i=0}^{m} a_i \phi^i + O(\phi^{m+1})$. Then f is determined uniquely.

Proof: Consider f described in the basis $\hat{S}_m$. Note that the functions $x^i y^j \in \hat{S}_m$ have distinct order of vanishing (i.e. valuation $v(\cdot)$) at the place $(0, 0)$: indeed $v(x^i y^j) = i + j(q + 1)$, and since $0 \leq i \leq q$, these quantities will be distinct as $x^i y^j$ runs through $\hat{S}_m$. Also, for any $x^i y^j \in \hat{S}_m$ we have $v(x^i y^j) \leq m < q^3$. The coefficients a_i therefore uniquely determine a linear combination $g = \sum_{i,j} b_{i,j} x^i y^j \in \mathcal{L}(mP_\infty)$ of elements in $\hat{S}_m$ such that $v(f - g) > m$. This implies that $f - g \in \mathcal{L}(mP_\infty - (m + 1)(0, 0))$. However, since that divisor clearly has negative degree, the Riemann-Roch space must be $\{0\}$, implying $f = g$ as desired. ■

Note that when computing the linear combination $g = \sum_{i,j} b_{i,j} x^i y^j$ in the above proof, we are essentially using back-substitution: one finds $x^i y^j \in \hat{S}_m$ and $c \in \mathbb{F}_{q^2}$ such that $v(f - c x^i y^j) > v(f)$, i.e., one eliminates the lowest order term in the approximate power series development of f. Then one updates f to $f - c x^i y^j$ (as well as the corresponding truncated power series) and iterates this process till all coefficients in the truncated power series of f are eliminated. By Lemma 38 an update can be performed in $O(q)$. The total construction of g therefore can be done in $O(q^4)$. If one ends in the situation that a coefficient in the (updated) truncated power series cannot be eliminated by adding a multiple of a power series development of an element from $\hat{S}_m$, then the conclusion is that for no $f \in \mathcal{L}(mP_\infty)$ it holds that $f = \sum_{i=0}^{m} a_i \phi^i + O(\phi^{m+1})$. Otherwise, one ends up with f described in the basis $\hat{S}_m$, and one can then convert into the basis S_m. All in all, we have shown the following:

Proposition 42: Let $m < q^3$. Given a truncated power series development to precision $m + 1$ for an element f, one can determine whether or not $f \in \mathcal{L}(mP_\infty)$, and in the affirmative case express f in the basis S_m in complexity $O(q^4)$.

REFERENCES

[1] K. Lee and M. E. O'Sullivan, "List decoding of Hermitian codes using Gröbner bases," *J. Symb. Comp.*, vol. 44, no. 12, pp. 1662–1675, 2009.
[2] P. Beelen and K. Brander, "Efficient list decoding of a class of algebraic-geometry codes," *Adv. Mathematics of Comm.*, vol. 4, pp. 485–518, Nov. 2010.
[3] M. Alekhnovich, "Linear Diophantine equations over polynomials and soft decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 51, pp. 2257–2265, July 2005.
[4] P. Giorgi, C. Jeannerod, and G. Villard, "On the complexity of polynomial matrix computations," in *Proc. of ISSAC*, pp. 135–142, 2003.
[5] J. S. R. Nielsen, "Power decoding of Reed–Solomon codes revisited," in *ICMCTA*, Sept. 2014.
[6] S. Sakata, H. E. Jensen, and T. Høholdt, "Generalized Berlekamp-Massey decoding of algebraic-geometric codes up to half the Feng–Rao bound," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, pp. 1762–1768, 1995.
[7] W. A. Stein *et al.*, *Sage Mathematics Software*. http://www.sagemath.org.
[8] H. Stichtenoth, "A note on Hermitian codes over $GF(q^2)$," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1345–1348, 1988.
[9] K. Brander, *Interpolation and List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2010.
[10] H. Stichtenoth, *Algebraic Function Fields and Codes*. Springer, 2nd ed., 2009.
[11] K. Yang and P. V. Kumar, "On the true minimum distance of Hermitian codes," in *Coding Theory and Algebraic Geometry*, pp. 99–107, Springer, 1992.
[12] J. S. R. Nielsen, "Solving generalised Padé approximations over polynomial rings," in *Preprint*, Jan. 2014. Available at http://jsrn.dk/.
[13] J. S. R. Nielsen, *List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2013. Available at jsrn.dk.
[14] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *J. Symb. Comp.*, vol. 35, no. 4, pp. 377–401, 2003.
[15] W. Zhou, G. Labahn, and A. Storjohann, "Computing minimal nullspace bases," in *Proc. of ISSAC*, (New York, NY, USA), pp. 366–373, ACM, 2012.
[16] S. Sarkar and A. Storjohann, "Normalization of row reduced matrices," in *Proc. of ISSAC*, (New York, NY, USA), pp. 297–304, ACM, 2011.
[17] J. S. R. Nielsen, "Generalised multi-sequence shift-register synthesis using module minimisation," in *Proc. of IEEE ISIT*, 2013.
[18] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon codes and algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, 1999.
[19] P. Beelen and T. Høholdt, "The decoding of algebraic geometry codes," in *Advances in Algebraic Geometry Codes* (E. Martínez-Moro, ed.), vol. 5, World Scientific, 2008.
[20] H. Cohn and N. Heninger, "Ideal forms of Coppersmith's theorem and Guruswami–Sudan list decoding," *arXiv*, vol. 1008.1284, 2010.
[21] X.-W. Wu and P. H. Siegel, "Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2579–2587, 2001.
[22] R. Roth and G. Ruckenstein, "Efficient decoding of Reed–Solomon codes beyond half the minimum distance," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 246 –257, 2000.
[23] J. Justesen, K. J. Larsen, H. E. Jensen, and T. Høholdt, "Fast decoding of codes from algebraic plane curves," *IEEE Trans. Inf. Theory*, vol. 38, no. 1, pp. 111–119, 1992.
[24] G. Schmidt, V. Sidorenko, and M. Bossert, "Syndrome decoding of Reed-Solomon codes beyond half the minimum distance based on shift-register synthesis," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5245–5252, 2010.
[25] S. Kampf, "Bounds on collaborative decoding of interleaved Hermitian codes and virtual extension," *Designs, Codes and Cryptography*, pp. 1–17, 2012.
[26] S. Kampf and W. Li, "Decoding interleaved Reed–Solomon and Hermitian codes with generalized divisions," in *Proc. of SCC*, pp. 1–6, 2013.
[27] A. Zeh, A. Wachter, and M. Bossert, "Unambiguous decoding of generalized Reed–Solomon codes beyond half the minimum distance," in *Proc. of IZS*, 2012.

[2] It is, in fact, easy to show that the reverse conversion can be done with the same complexity, but we will not need that conversion.

[28] H. E. Jensen, R. R. Nielsen, and T. Høholdt, "Performance analysis of a decoding algorithm for algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1712–1717, 1999.
[29] J. P. Hansen, "Dependent rational points on curves over finite fields-Lefschetz theorems and exponential sums," in *Proc. of WCC*, pp. 297–309, 2001.
[30] G.-L. Feng, V. K. Wei, T. R. N. Rao, and K. K. Tzeng, "Simplified understanding and efficient decoding of a class of algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 40, no. 4, pp. 981–1002, 1994.
[31] W. Zhou and G. Labahn, "Efficient algorithms for order basis computation," *J. Symb. Comp.*, vol. 47, pp. 793–819, July 2012.
[32] S. Sakata, J. Justesen, Y. Madelung, H. E. Jensen, and T. Høholdt, "A fast decoding method of AG codes from Miura-Kamiya curves C_{ab} up to half the Feng-Rao bound," *Finite Fields and Their Appl.*, vol. 1, pp. 83–101, Jan. 1995.
[33] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis Of Computer Algorithms*. Addison-Wesley, 1974.
[34] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge Univ. Press, 3rd ed., 2012.
[35] T. H. Cormen, Charles E. Leierson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms*. Cambridge, Mass.: MIT Press, 2009.