Computational Aspects of Algebraic Geometry Codes

Solomatov, Grigory Aleksandrovich

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Computational Aspects of Algebraic Geometry Codes

Grigory Aleksandrovich Solomatov

Kongens Lyngby 2021

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

This thesis is dedicated to an investigation of various computational problems associated with algebraic geometry (AG) codes – a diverse class of error-correcting codes among which one finds those that are the most robust to errors. Our main contribution consists of a Guruswami-Sudan style list-decoding algorithm that works for any AG code while being more efficient in this fully general setting than any previously known method. This is achieved by extending an existing technique that relies on row-reduction of polynomial matrices to solve the same problem in a more restricted context of one-point Hermitian codes.

We also consider "improved power decoding" – a completely different decoding paradigm that revolves around certain key equations. Our contribution here consists of formulating these equations for all AG codes, whereas previous work has focused on specific cases. This formulation allows us to show that power decoding likely can correct as many errors as Guruswami-Sudan decoding, which is consistent with our numerical simulations. Although our decoder uses linear algebra and is therefore computationally expensive, its conceptual framework suggests a way of making it as efficient as our main contribution.

Besides decoding, we also investigate encoding of AG codes, albeit in a less general setting, which reduces the problem to that of multi-point evaluation of bivariate polynomials. This allows us to develop algorithms that are extremely efficient for important special cases such as one-point Hermitian codes. We then consider bivariate multi-point evaluation in a setting completely divorced from AG codes, where we contribute with novel and efficient algorithms that utilize precomputation.

Summary (Danish)

Denne afhandling udforsker diverse beregningsproblemer forbundet med algebraiske geometri (AG) koder. AG koder udgør en mangfoldig klasse af fejlkorrigerende koder, hvoriblandt de mest fejlrobuste findes. Vores største bidrag består af en Guruswami-Sudan type liste-afkodningsalgoritme, der virker for enhver AG kode. Denne generelle algoritme er samtidig hurtigere en nogen tidligere kendt metode. Dette opnås ved at bygge på en eksisterende teknik, der benytter sig af række-reduktion af polynomiumsmatricer for at løse det samme problem i en begrænset kontekst af etpunkts Hermitiske koder.

Denne afhandling omhandler også "forbedret power-afkodning", hvilket er et anderledes afkodningsparadigme baseret på visse nøgleligninger. Vores bidrag her består af formuleringen af disse nøgleligninger for alle AG koder, hvorimod tidligere indsats har fokuseret på specifikke tilfælde. Denne formulering hjælper os med at demonstrere, at power-afkodning sandsynligvis kan korrigere lige så mange fejl som Guruswami-Sudan afkodning, hvilket stemmer overens med vores numeriske simuleringer. Selvom vores afkoder bruger lineær algebra og derfor er beregningsmæssigt langsomt, så indikerer dens konceptuelle ramme, at den kan konstrueres lige så hurtigt som vores hovedbidrag.

Udover afkodning undersøger vi også indkodning af AG koder. Dette gør vi i en mindre generel setting, hvilket reducerer problemet til multi-punkts evaluering af bivariate polynomier. Dette gør det muligt at udvikle algoritmer, der er ekstremt hurtige i vigtige specielle tilfælde, såsom etpunkts Hermitiske koder. Derefter undersøger vi bivariat multi-punkts evaluering i en setting, der er helt adskilt fra AG koder, hvor vi bidrager med nye og effektive algoritmer, der benytter forberegning.

Preface

The work which this thesis is based upon was done under the supervision of (formerly) Associate Professor Johan Rosenkilde and Professor Peter Beelen at the Technical University of Denmark in the period October 2018–October 2021. During this time, I have coauthored the following four papers:

[BRS20] Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast encoding of ag codes over Cab curves. *IEEE Transactions on Information Theory*, 67(3):1641–1655, 2020

[NRS20] Vincent Neiger, Johan Rosenkilde, and Grigory Solomatov. Generic bivariate multi-point evaluation, interpolation and modular composition with precomputation. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, pages 388–395, 2020

[PRS21] Sven Puchinger, Johan Rosenkilde, and Grigory Solomatov. Improved power decoding of algebraic geometry codes. *arXiv preprint arXiv:2105.00178*, 2021

[BRS21] Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast list decoding of algebraic geometry codes. *To be submitted*, 2021

Lyngby, 14-October-2021

Grigory Solomatov

Grigory Aleksandrovich Solomatov

Acknowledgments

I wish to express my sincere gratitude to my PhD advisors Johan Rosenkilde and Peter Beelen. Apart from possessing intimidating levels of expertise in a wide variety of highly technical topics, they are also both very kind and supportive, always encouraging and inspiring those around them. I would like to thank Johan for his contagious enthusiasm and relentless attitude of always being ready to tackle new challenges while insisting on the highest quality; and I wish to thank Peter for sharing some of his inexhaustible mathematical insights and curiosity as well as really useful practical wisdom. Finally, I wish to thank both of them for their patience with me as their student and for being so generous with their time.

I am grateful for having had the opportunity to collaborate with Vincent Neiger and Sven Puchinger, who are both as friendly as they are competent. Also, I would like to thank Éric Schost for letting me visit him at the University of Waterloo; it is unfortunate that I had to leave so early because of the pandemic.

I also wish to thank my colleagues at the Section for Mathematics at DTU Compute for creating such a comfortable atmosphere. Especially, I am grateful to my fellow PhD students for the many fun discussions over lunch and cappuccino breaks. In particular, I would like to thank my office mate Leonardo Landi for understanding that I was actually *letting* him win in fussball all these years.

I am thankful to my friends for exposing me to many new perspectives, and to my family for all of the support and care that they have given me. Last but not least, I thank Herdís – my better half – for making this adventure truly special.

I gratefully acknowledge the support from The Danish Council for Independent Research (DFF-FNU) for the project *Correcting on a Curve* (Grant No. 8021-00030B) on which I was hired, as well as Otto Mønsteds Fond for supporting my travel to Canada.

Contents

Chapter 1

Introduction

The ability to reliably transmit information over unreliable channels continues to grow in importance as digital communication becomes increasingly embedded into the fabric of the modern world. The ever-expanding list of applications of *error-correcting codes* includes – but is certainly not limited to – satellite and fiber-optic communication, local and distributed storage systems, network coding as well as post-quantum cryptography. In 1981, V. D. Goppa introduced the astronomically diverse class of *algebraic geometry (AG) codes*, which have since generated a considerable amount of theoretical interest as they lend themselves to deep mathematical analysis and include some of the most error-resilient codes currently known. The benefits of these codes, however, do not come without a price: the computational problems associated with using them in practice are notoriously difficult to solve efficiently, and as a consequence of this, it would be an understatement to say that their current real-world utility is rather limited.

The primary goal of this thesis is to address some of the main computational problems associated with AG codes, aiming to bring their theoretical potential closer to being realized as practical value.

1.1 Reader's guide

This thesis adheres to straightforward structure: the current chapter provides context for Chapters 2–5, all of which can be read independently from each other. There are a few cross references, but they are never essential and are always explicit.

The chapter that you are currently reading has two purposes: The first one is to unify the rest of this thesis under a single overarching "story" as well as to establish the relevant "rules of the game" when it comes to computer algebra and algebraic coding theory – an understanding of this will be expected from the reader later on. The secondary purpose is to contain those technical results and definitions that are needed in more than one of the subsequent chapters – these may be confidently skipped during a linear reading, as they are explicitly referred to when used.

In Chapter 6 we briefly summarize our results and mention potential future research. Appendix A contains an overview over often used notation.

1.2 Complexity model

Since this thesis is primarily concerned with efficient algorithms, we ought to carefully specify what we mean by "efficient". Although it would be perfectly reasonable to measure an algorithm's performance by the amount of time or energy spent while producing the output, we shall instead abide by the more common practice in computational complexity theory and count the total number of *fundamental operations* carried out during the computation. This notion of performance greatly simplifies complexity analysis as it does not depend on the physical machine on which the algorithm is being executed. Simplifying matters even further, we will only consider the *worst-case* complexity for any given input size as the workload for an algorithm can vary even across equally sized inputs.

Instead of considering exact complexity bounds, which are often laborious to obtain and impractical to compare, we will focus our attention on *asymptotic complexity*, utilizing the *big-O* and the *soft-O* notation, denoted by $\mathcal{O}(\cdot)$ and

$\tilde{\mathcal{O}}(\cdot)$ respectively. Formally, for any function $f : \mathbb{Z}_{\geqslant 0} \to \mathbb{Z}_{\geqslant 0}$

$$\mathcal{O}(f) = \left\{ g : \mathbb{Z}_{\geqslant 0} \to \mathbb{Z}_{\geqslant 0} \;\middle|\; \begin{array}{l} \exists c, m_0 \in \mathbb{R}_{>0} \text{ such that} \\ g(m) \leqslant c f(m) \;\forall m \geqslant m_0 \end{array} \right\} \text{ and}$$

$$\tilde{\mathcal{O}}(f) = \bigcup_{i=0}^{\infty} \mathcal{O}(f \log(f)^i) \; .$$

For example: $(m^2 + 2m + 3)\log(m)^3 \in \mathcal{O}(m^2 \log(m)^3) \subset \tilde{\mathcal{O}}(m^2)$. Intuitively speaking, asymptotic complexity measures how the cost of algorithms *scales* with input size. From this perspective, an algorithm that costs $10^{100}m \in \mathcal{O}(m)$ operations for input size m is more efficient than one which costs m^2, even though the former is much slower than the latter for all reasonable input sizes. Consequently, it can be argued that the hunt for asymptotically fast algorithms might sometimes yield results of little practical utility, which is definitely true[1], but in reality, such cases are exceptions rather than the rule. Before proceeding to a discussion about "fundamental operations", let us mention that we will also sometimes use Donald Knuth's *big*-Ω notation, which is defined by the equivalence: $f \in \Omega(g)$ if and only if $g \in \mathcal{O}(f)$.

Let us now specify what we earlier meant by "fundamental operations". Since the algorithms that we will consider mostly deal with algebraic objects constructed over some (finite) field $\mathbb{F}$, it is natural to express their complexity bounds in terms of the required number of *basic arithmetic operations* in $\mathbb{F}$, i.e. additions, subtractions, multiplications and divisions; as well as zero tests. Although most of these algorithms are field agnostic – and are therefore correct for all fields – this notion of *algebraic complexity* is a poor proxy for execution time on a physical machine when dealing with fields of infinite cardinality. The reason for this is simple: The unbounded bit-representation size of elements from infinite fields can make the physical cost of a single arithmetic operation arbitrarily high, e.g. due to large denominators in the case of $\mathbb{Q}$.

We conclude this section by mentioning that for certain algorithms it makes sense to conceptualize the computation as being divided into an "online phase" as well as a "precomputation phase". For an illustration of this, consider the hypothetical example where we have a database consisting of a list containing m numbers, and our task is to reply to users who wish to know whether a number of their choosing is to be found in this list. Allowing *precomputation*, i.e. sorting the list, reduces the cost of each *online* query from $\mathcal{O}(m)$ checks to $\mathcal{O}(\log m)$ checks – simply using binary search instead of linear search. As far as the user is concerned, the additional cost of precomputation then hardly matters (as long as it terminates before deployment, of course).

[1] Kedlaya and Umans' asymptotically efficient algorithm for multivariate multi-point evaluation [KU08] has resisted practical implementations according to e.g. [vdHL19], the same goes the "fastest" matrix multiplication algorithms [LG12].

1.3 Complexities of fundamental algorithms

In this section we list the computational results that form the bedrock of our algorithms. We will use them extensively throughout the thesis, often without referencing their respective sources. As detailed in Section 1.2, the cost of basic arithmetic operations in our base field $\mathbb{F}$ will be considered to be $\mathcal{O}(1)$.

Polynomials

The cost of adding two univariate plynomials in

$$\mathbb{F}[X]_{<m} := \{f \in \mathbb{F}[X] \mid \deg f < m\}$$

is clearly $\mathcal{O}(m)$ (actually, it is exactly m). We denote by $\mathsf{M}(m)$ the cost of multiplying two polynomials in $\mathbb{F}[X]_{<m}$, making the customary assumption that $\mathsf{M}(m)/m$ is a non-decreasing function. If $\mathbb{F}$ supports FFT by containing certain primitive roots of unity, then $\mathsf{M}(m)$ can be taken to be $\mathcal{O}(m \log m) \subset \widetilde{\mathcal{O}}(m)$ [vzGG12, Corollary 8.19]. In the general case, the multiplication algorithm by Shönhage and Strassen gives $\mathsf{M}(m) \in \mathcal{O}(m \log m \log\log m) \subset \widetilde{\mathcal{O}}(m)$ [vzGG12, Theorem 8.23]. Finally, to the best of our knowledge, if $\mathbb{F} = \mathbb{F}_q$ is a finite field, then the currently fastest known approach is due to [HvdH19]. The product of any two bivariate polynomials $f, g \in \mathbb{F}[X, Y]$ with $\deg_X f < m$ and $\deg_Y g < \ell$ can be computed using $\mathcal{O}(\mathsf{M}(m\ell)) \subset \widetilde{\mathcal{O}}(m\ell)$ operations in $\mathbb{F}$ [vzGG12, Corollary 8.28].

For any two polynomials $f, h \in \mathbb{F}[X]$ with $\deg f, \deg h < m$, we can compute the unique remainder $r \in \mathbb{F}[X]$ with $\deg r < \deg h$ such that $f = uh + r$ for some $u \in \mathbb{F}$ using at most $\mathcal{O}(\mathsf{M}(m))$ operations in $\mathbb{F}$ [vzGG12, Theorem 9.6].

Two important non-arithmetic operations for polynomials are *multi-point evaluation (MPE)* as well as its its inverse problem – *interpolation*. Given a polynomial $f \in \mathbb{F}[X]_{<m}$ and *evaluation points* $\alpha_1, \ldots, \alpha_m \in \mathbb{F}$, MPE computes the *evaluations* $(f(\alpha_1), \ldots, f(\alpha_m)) \in \mathbb{F}^m$. Interpolation, on the other hand, takes as input the points together with some prescribed evaluations $v_1, \ldots, v_m \in \mathbb{F}$ and computes the unique polynomial $f \in \mathbb{F}[X]_{<m}$ with $f(\alpha_j) = v_j$ for $j = 1, \ldots, m$. The cost of MPE as well as of interpolation can be taken to bee $\mathcal{O}(\mathsf{M}(m) \log m) \subset \widetilde{\mathcal{O}}(m)$ operations in $\mathbb{F}$, see [vzGG12, Corollary 10.8] and [vzGG12, Corollary 10.12] respectively.

Extension fields

Some of our algorithms might require more field elements than what can be provided by $\mathbb{F} = \mathbb{F}_q$, i.e. the finite field of cardinality q. To accommodate for this, we might need to go to an extension field $\mathbb{F}_{q^e}$ for some $e \in \mathbb{Z}_{>0}$. Constructing such an extension field costs $\mathcal{O}(e\mathsf{M}(e)\log e \log(eq)) \subset \widetilde{\mathcal{O}}(e^2 \log q)$ operations in $\mathbb{F}_q$ [vzGG12, Corollary 14.43]; however, in all of our use cases, this can be done during precomputation without affecting the online complexity. It is easy to see that one multiplication in $\mathbb{F}_{q^e}$ costs $\mathcal{O}(\mathsf{M}(e)) \subset \widetilde{\mathcal{O}}(e)$ operations in $\mathbb{F}_q$ – simply combine polynomial multiplication with remaindering.

Matrices

Following convention, we denote by ω the *matrix multiplication exponent*, i.e. some positive real number ω such that the product of any two square matrices in $\mathbb{F}^{m\times m}$ can be computed using $\widetilde{\mathcal{O}}(m^\omega)$ operations in $\mathbb{F}$. Often, ω is defined to be the smallest[2] such number, but we will consider it as a parameter that the user can choose by using an appropriate matrix multiplication algorithm. A case can be made that this view is more suitable for real-world applications as algorithms with lower values of ω tend to become less practical in spite of being asymptotically fast. It is clear that $\omega \geqslant 2$ since the input size is $\mathcal{O}(m^2)$, and that it probably never makes sense to choose $\omega > 3$ since the cost of the naive algorithm has complexity $\mathcal{O}(m^3)$.

In 1969, likely being inspired by Karatsuba's first subquadratic integer multiplication algorithm [KO64], Strassen demonstrated that a divide-and-conquer approach to matrix multiplication could achieve the exponent $\omega = \log_2 7 < 2.8074$ [Str69], shattering the commonly held belief at the time that $\omega = 3$ is optimal. This breakthrough had set in motion a collective search for even faster algorithms – Alman and Williams being the current record holders, showing that it is possible to choose $\omega < 2.37286$ [AW21]. It is conjectured that $\omega = 2$ is the optimal value, however, Ambainis, Filmus and Le Gall have firmly established that the current paradigm of matrix multiplication algorithms can never achieve $\omega < 2.3725$ [AFLG15].

[2] to make sure that such ω exists, it is then defined as the smallest value for which the cost bound $\mathcal{O}(m^{\omega+\epsilon})$ holds for every $\epsilon > 0$.

1.4 Algorithms for polynomial matrices

Multiple times throughout this thesis, we will make use of efficient algorithms for $\mathbb{F}[X]$-submodules of $\mathbb{F}[x]^m$. This section is dedicated to presenting some of the associated complexity estimates as well as parts of the conceptual framework needed to reason about them – a great resource on this topic is [Nei16]. Since the inner workings of these algorithms are not within the scope of this thesis, we will simply use them as "black boxes".

Now, computational consideration of any object – mathematical or not – requires a way of representing it, and a common way of representing submodules of $\mathbb{F}[X]^m$ is using a basis. Indeed, any such submodule is free and has rank $r \leqslant m$ [DF04, Section 12.1, Theorem 4], which means that any of its infinitely many possible bases can be associated with the rows of a full rank matrix $\boldsymbol{M} \in \mathbb{F}[X]^{r\times m}$. This level of generality, however, will not be needed anywhere in the context of this thesis, and so we will assume that $r = m$, and consequently that $\boldsymbol{M} \in \mathbb{F}[X]^{m\times m}$ is nonsingular.

It should be clear that the row space of a matrix $\boldsymbol{N} = \boldsymbol{U}\boldsymbol{M} \in \mathbb{F}[X]^{m\times m}$, i.e. the set of all of the $\mathbb{F}[X]$-linear combinations of its rows, is identical to that of $\boldsymbol{M}$ if and only if $\boldsymbol{U} \in \mathbb{F}[X]^{m\times m}$ is invertible, or equivalently if $\det \boldsymbol{U} \in \mathbb{F}\backslash\{0\}$. In such a case, it is said that $\boldsymbol{N}$ and $\boldsymbol{M}$ are *unimodularly equivalent* – they describe the same module, albeit possibly using different bases; the matrix $\boldsymbol{U}$ is then said called *unimodular*. Not all basis matrices are equally suited for computation, however, as demonstrated by the following example:

Example 1.1. Consider the matrices $\boldsymbol{M}, \boldsymbol{N} \in \mathbb{F}[X]^{2\times 2}$ defined by

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 \\ 0 & X \end{bmatrix} \quad \text{and} \quad \boldsymbol{N} = \begin{bmatrix} 1 & \sum_{i=0}^{10^{100}} X^i \\ 0 & X \end{bmatrix},$$

and note that they have identical row spaces. Indeed, $\boldsymbol{N} = \boldsymbol{U}\boldsymbol{M}$, where

$$\boldsymbol{U} = \begin{bmatrix} 1 & \sum_{i=0}^{10^{100}-1} X^i \\ 0 & 1 \end{bmatrix}$$

satisfies $\det \boldsymbol{U} = 1$, but $\boldsymbol{M}$ can be represented using only a few elements from $\mathbb{F}$, while $\boldsymbol{N}$ needs roughly 10^{100}.

In light of Example 1.1, a natural question to ask is how we can find the "best" basis matrix $\boldsymbol{M} \in \mathbb{F}[X]^{m\times m}$ for any given module $\mathcal{M} \subseteq \mathbb{F}[X]^m$, or what that would even mean. One thing is clear though, whatever the notion of "best" we

settle upon should result in the entries of $\boldsymbol{M}$ having small degrees – this motivates the following definition, which provides a generalized way of measuring degrees of row vectors:

Definition 1.2. For any polynomial vector $\boldsymbol{v} = (v_1, \ldots, v_m) \in \mathbb{F}[X]^m$ and any *shift* $\boldsymbol{s} = (s_1, \ldots, s_m) \in \mathbb{Z}^m$, we define the $\boldsymbol{s}$*-degree* of $\boldsymbol{v}$ as

$$\deg_{\boldsymbol{s}} \boldsymbol{v} = \max_i \{\deg v_i + s_i\} \ .$$

Furthermore, if $j \in \{1, \ldots, m\}$ is maximal such that $\deg v_j + s_j = \deg_{\boldsymbol{s}} \boldsymbol{v}$, then we say that v_j is the $\boldsymbol{s}$*-pivot* of $\boldsymbol{v}$, and j is its $\boldsymbol{s}$*-pivot index*. If we omit writing $\boldsymbol{s}$ in any of the above notation, then it is to be understood that $\boldsymbol{s} = \boldsymbol{0}$, i.e. we might write: *pivot*, *pivot index* and *degree*, denoting the latter by $\deg \boldsymbol{v} := \deg_{\boldsymbol{0}} \boldsymbol{v}$.

Intuitively speaking, the pivot of a polynomial vector is its rightmost entry of maximal degree, and the pivot index is the index of this entry. An analogous statement holds for the $\boldsymbol{s}$-pivot as well as the $\boldsymbol{s}$-pivot index, provided that one accounts for the shift $\boldsymbol{s}$. Sometimes, the entries of $\boldsymbol{s}$ are referred to as *weights*, and one might speak of e.g. degree or pivot index with respect to certain weights.

Having familiarized ourselves with the notion of shifted degrees of (row) vectors, we are ready to define the "best" basis matrix for any submodule of $\mathbb{F}[X]^m$, that is, with respect to given weights. As the following definition of *(shifted) Popov form* might come across as somewhat arbitrary to an unfamiliar reader, let us first quickly motivate it by revealing two of its most convenient properties: that any submodule of $\mathbb{F}[X]^m$ admits a *unique* basis matrix in Popov form, and that the degrees of the rows in this matrix are minimal – we will come back to these properties shortly.

Definition 1.3. For any shift $\boldsymbol{s} \in \mathbb{Z}^m$, a nonsingular matrix $\boldsymbol{P} \in \mathbb{F}[X]^{m \times m}$ is said to be in $\boldsymbol{s}$*-Popov form* if all of the $\boldsymbol{s}$-pivots of its rows lie on the diagonal, are monic and have degrees strictly greater then all other entries in their respective columns. Furthermore, if such a $\boldsymbol{P}$ is unimodularly equivalent to some matrix $\boldsymbol{M} \in \mathbb{F}[X]^{m \times m}$, then then $\boldsymbol{P}$ is said to be *the* $\boldsymbol{s}$*-Popov form* of $\boldsymbol{M}$.

We end this section by listing some of the results that make Popov forms extremely useful – for this, we will need a few more definitions:

Definition 1.4. For any matrix $\boldsymbol{M} \in \mathbb{F}[X]^{m \times m}$ and any shift $\boldsymbol{s} \in \mathbb{Z}^m$, we define the $\boldsymbol{s}$*-row degree* of $\boldsymbol{M}$ as the tuple $\operatorname{rdeg}_{\boldsymbol{s}}(\boldsymbol{M}) = (d_1, \ldots, d_m) \in \mathbb{Z}^m$, where d_i denotes the $\boldsymbol{s}$-degree of the i-th row of $\boldsymbol{M}$. Similarly, we define the *column degree* as $\operatorname{cdeg}(\boldsymbol{M}) = \operatorname{rdeg}(\boldsymbol{M}^\top)$; in this case we will not need shifts. Finally, for any $\boldsymbol{t} \in \mathbb{Z}_{\geqslant 0}$, we denote by $|\boldsymbol{t}|$ the sum of the entries in $\boldsymbol{t}$.

Proposition 1.5 ([Nei16, Theorem 1.11])**.** *If $\boldsymbol{P} \in \mathbb{F}[X]^{m\times m}$ is in $\boldsymbol{s}$-Popov form for some shift $\boldsymbol{s} \in \mathbb{Z}^m$, then $\boldsymbol{P}$ is $\boldsymbol{s}$-row reduced, which is to say that is satisfies all of the following equivalent assertions:*

1) $\operatorname{rdeg}_{\boldsymbol{s}}(\boldsymbol{P}) \leqslant \operatorname{rdeg}_{\boldsymbol{s}}(\boldsymbol{U}\boldsymbol{P})$ *for all unimodular matrices $\boldsymbol{U} \in \mathbb{F}[X]^{m\times m}$, where the inequality is taken entrywise after sorting in non-decreasing order;*
2) *if $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_m] \in \mathbb{F}[X]^{1\times m}$ with $\lambda_i = 1$ for some i, and $\boldsymbol{p}^{(i)} \in \mathbb{F}[X]^{1\times m}$ is the i-th row of $\boldsymbol{P}$, then $\deg_{\boldsymbol{s}}(\boldsymbol{p}^{(i)}) \leqslant \deg_{\boldsymbol{s}}(\boldsymbol{\lambda}\boldsymbol{P})$;*
3) *predictable degree property: for any $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_m] \in \mathbb{F}[X]^{1\times m}$*
$$\deg_{\boldsymbol{s}}(\boldsymbol{\lambda}\boldsymbol{P}) = \max_i\{\deg \lambda_i + \deg_{\boldsymbol{s}}(\boldsymbol{p}^{(i)})\} = \deg_{\boldsymbol{d}}(\boldsymbol{\lambda}) \ ,$$
where $\boldsymbol{d} = \operatorname{rdeg}_{\boldsymbol{s}}(\boldsymbol{P})$;
4) $|\operatorname{rdeg}_{\boldsymbol{s}}(\boldsymbol{P})| = \deg(\det(\boldsymbol{P})) + |\boldsymbol{s}|$.

Proposition 1.6 ([Nei16, Lemma 1.17])**.** *If $\boldsymbol{P} \in \mathbb{F}[X]^{m\times m}$ is in $\boldsymbol{s}$-Popov form*[3] *for some $\boldsymbol{s} \in \mathbb{Z}^m$ and $\boldsymbol{p} \in \mathbb{F}[X]^{1\times m}$ is a nonzero vector in the row space of $\boldsymbol{P}$ having $\boldsymbol{s}$-pivot index i, then $\deg_{\boldsymbol{s}} \boldsymbol{p} \geqslant \deg_{\boldsymbol{s}} \boldsymbol{p}^{(i)}$, where $\boldsymbol{p}^{(i)} \in \mathbb{F}[X]^{1\times m}$ denotes the i-th row of $\boldsymbol{P}$.*

Proposition 1.7 ([Kai80, Theorem 6.3-15][4])**.** *If $\boldsymbol{P} \in \mathbb{F}[X]^{m\times m}$ is in $\boldsymbol{s}$-Popov form for some $\boldsymbol{s} \in \mathbb{Z}^m$, then for any vector $\boldsymbol{v} \in \mathbb{F}[X]^{1\times m}$, there exist unique $\boldsymbol{q}, \boldsymbol{u} \in \mathbb{F}[X]^{1\times m}$ satisfying $\boldsymbol{v} = \boldsymbol{q}\boldsymbol{P} + \boldsymbol{u}$ and $\deg_{-\boldsymbol{\delta}}(\boldsymbol{u}) < 0$, where $\boldsymbol{\delta} = \operatorname{cdeg}(\boldsymbol{P})$.*

In the context of Proposition 1.7, we will adopt the notation $\boldsymbol{u} = \boldsymbol{v} \operatorname{rem} \boldsymbol{P}$.

Proposition 1.8 ([NV17, Theorem 1.3])**.** *There is a deterministic algorithm which for any shift $\boldsymbol{s} \in \mathbb{Z}^m$ and any nonsingular matrix $\boldsymbol{M} \in \mathbb{F}[X]^{m\times m}$ computes the $\boldsymbol{s}$-Popov form of $\boldsymbol{M}$ using $\widetilde{\mathcal{O}}(m^{\omega-1}|\operatorname{cdeg}(\boldsymbol{M})|) \subseteq \widetilde{\mathcal{O}}(m^{\omega} \deg \boldsymbol{M})$ operations in $\mathbb{F}$, where $\deg \boldsymbol{M}$ denotes the maximal degree across the entries of $\boldsymbol{M}$.*

1.5 Communication model

Before we formally introduce AG codes, let us take a moment to put them into context. In the most general form of digital communication over an unreliable channel, a *sender* and a *receiver* agree upon a *finite alphabet* Σ (prior to communication). The elements of Σ, called *symbols*, can be transmitted over a *channel*

[3] actually, $\boldsymbol{P}$ only needs to be in *weak* $\boldsymbol{s}$-Popov form, which is a weaker condition
[4] see also [Nei16, Lemma 1.24]

capable of replacing any symbol with another element of Σ due to random or adversarial noise. Of course, if every single sent symbol is randomly replaced during transmission, then communication is impossible; however, if the channel provides some guarantees on its reliability, then error-correcting codes can to some extent mitigate the noise.

A common reliability assumption on the channel is that it can corrupt at most τ symbols in every sent *block* consisting of $n > \tau$ symbols. This assumption is often used in *block codes*, where any *message* $\boldsymbol{m} \in \Sigma^k$, with $k < n$, is *encoded* by the sender using some injective map $\mathrm{Enc} : \Sigma^k \to \Sigma^n$ which is also known to the receiver. The resulting *codeword* $\boldsymbol{c} = \mathrm{Enc}(\boldsymbol{m}) \in \Sigma^n$ is then transmitted over the channel with the hope that the introduced *redundancy* will protect the encoded message against potential errors. This brings us to the first computational problem that we wish to solve efficiently:

Problem 1.9 (Encoding)**.** Given a message $\boldsymbol{m} \in \Sigma^k$ and an injective encoding map $\mathrm{Enc} : \Sigma^k \to \Sigma^n$, compute the corresponding codeword $\boldsymbol{c} = \mathrm{Enc}(\boldsymbol{m})$.

The set of all codewords, which we denote by $\mathcal{C} = \mathrm{Enc}(\Sigma^k) \subset \Sigma^n$, is known as the *code*; and Σ^n, whose elements are called *words* is commonly referred to as the *ambient space*. The number of symbols in which two words differ is called *Hamming distance*, i.e. for any two words $\boldsymbol{u} = (w_1, \ldots, w_n)$ and $\boldsymbol{u}' = (w'_1, \ldots, w'_n)$ in Σ^n the Hamming distance between $\boldsymbol{u}$ and $\boldsymbol{u}'$ is defined as

$$d(\boldsymbol{w}, \boldsymbol{w}') = |\{i \mid w_i \neq w'_i\}| \ .$$

Equipped with this notion of distance we can formally define the *unique decoding problem* for block codes:

Problem 1.10 (Unique decoding)**.** Given a received word $\boldsymbol{r} \in \mathbb{F}^n$ find the *unique* closest codeword $\boldsymbol{c} \in \mathcal{C}$ to $\boldsymbol{r}$ with respect to the Hamming distance if such a unique codeword exists.

It is hardly surprising that Problem 1.10 can't always be solved uniquely[5], since the received word might be equally close to more than one codeword. This situation being *the only* possible cause for failure, however, makes it easy to precisely quantify the necessary and sufficient for conditions success. The *minimum distance* of a code, commonly denoted by d, is defined as the smallest Hamming distance between any two distinct codewords, i.e.

$$d = \min\{d(\boldsymbol{c}, \boldsymbol{c}') \mid \boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C} \text{ such that } \boldsymbol{c} \neq \boldsymbol{c}'\} \ .$$

[5] maximum likelihood decoding of Reed-Solomon codes is known to be NP-hard [GV05]

As long as, and only if, the number of errors introduced during transmission is bounded by $\tau \leqslant \lfloor d/2 \rfloor$, then we are guaranteed that there is exactly one codeword within distance τ from the received word.

An easier problem than decoding is *unencoding*, which asks to find the message that was used to generate a given codeword. This is the inverse problem of encoding and can also be seen as decoding a received word containing no errors. There are at least two reasons to consider this problem: The first reason is that a decoder that solves Problem 1.10 by definition returns a codeword, and since in practical applications one is typically interested in the corresponding message, an unencoder is needed. Some decoders, however, obtain the message without first computing the codeword, making unencoding unnecessary. Moreover, systematic encoding, i.e. encoding that simply concatenates redundancy to the message without changing it, renders unencoding trivial; but this brings us to the second reason for why unencoding is a worthwhile problem to consider: that for some codes like Reed-Solomon codes, unencoding can be used to transform a non-systematic encoder into a systematic one. In order to refer to unencoding later, we state the problem below:

Problem 1.11 (Unencoding)**.** Given a codeword $\boldsymbol{c} \in \mathcal{C}$ find a message $\boldsymbol{m} \in \Sigma^k$ such that $\boldsymbol{c} = \mathrm{Enc}(m)$.

AG codes belong to a special class of block codes called *linear codes*, where the alphabet Σ is replaced with a finite field $\mathbb{F}$, and were any $\mathbb{F}$-linear combination of codewords (as vectors in $\mathbb{F}^n$) is required to be a codeword. Imposing this structure on the code allows us to take advantage of tools from linear algebra, since the (now linear) encoding map can be represented by a full rank matrix $\boldsymbol{G} \in \mathbb{F}^{k \times n}$, i.e. any codeword $\boldsymbol{c} \in \mathbb{F}^n$ can be written as $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$, where $\boldsymbol{m} \in \mathbb{F}^k$ is some message. This representation immediately gives rise to the naive encoding algorithm for linear codes: simply carry out the vector-matrix multiplication. In general this requires $\mathcal{O}(nk)$ operations in $\mathbb{F}$, which becomes $\mathcal{O}(n^2)$ under the customary assumption that $n/k \in \mathcal{O}(1)$, however, for special classes of matrices we can do much better. For example, if $\boldsymbol{G}$ is a *Vandermonde matrix*, i.e. for some $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$

$$\boldsymbol{G} = \begin{pmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix},$$

as is the case with *Reed-Solomon (RS) codes*, then computation of $\boldsymbol{m}\boldsymbol{G}$ can be reinterpreted as evaluation of a univariate polynomial with coefficients in $\mathbb{F}$ and degree less than k at the points $\alpha_1, \ldots, \alpha_n$. This evaluation problem is known as *univariate multi-point evaluation* (MPE) and can be solved using $\mathcal{O}(\mathsf{M}(n)\log(n)) \subset \tilde{\mathcal{O}}(n)$ operations in $\mathbb{F}$, as mentioned in Section 1.3. AG codes

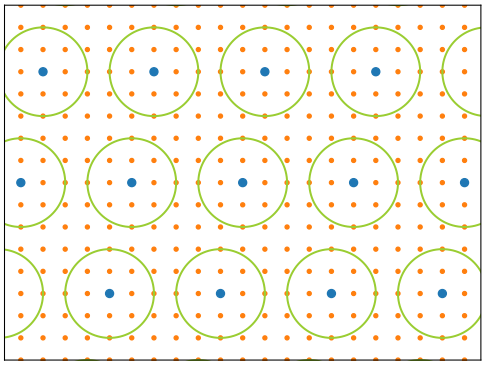

Figure 1.1: Schematic illustration of a linear code. The blue dots are the codewords, while the orange ones are possible received words containing at least one error. Euclidean distance in the image represents Hamming distance. The green circle around each codeword represents a *Hamming ball* with radius $\lfloor d/2 \rfloor$, meaning that there is no ambiguity about how to decode the words inside. In a non-linear code there would be less regularity in the codewords.

generalize RS codes by replacing $\boldsymbol{G}$ with a potentially more complicated, though still highly structured generator matrix. This generalization has the advantage of being able to describe codes with higher minimum distance than that of RS codes, however, the increased complexity in the matrix structure gives rise to more difficult computational problems.

1.6 Algebraic geometry codes

This section is dedicated to a brief introduction of AG codes; for a detailed presentation, the reader is referred to [Sti09, Chapter 2]. In the remainder of this thesis, we will denote by F some fixed function field having constant field $\mathbb{F}$, and for any divisor A of F, we denote its support by $\operatorname{supp}(A)$. The *Riemann-Roch* space associated with A will be denoted by $\mathcal{L}(A) = \{f \in F \mid (f) + A \geqslant 0\} \cup \{0\}$, where (f) is the *principal divisor* of f, and the $\mathbb{F}$-dimension of $\mathcal{L}(A)$ will be written as $\ell(A)$. Finally, for any place P of F, we let v_P denote the *valuation* at P. All of these concepts can be found in [Sti09, Chapter 1], although the font in the notation might differ slightly. Without further ado, we formally present AG codes in the following definition:

Definition 1.12. For any divisors G and $D = P_1 + \cdots + P_n$ of F, where

$P_1, \ldots, P_n$ are pairwise distinct rational places and $\operatorname{supp}(D) \cap \operatorname{supp}(G) = \varnothing$, the *algebraic geometry (AG) code* associated with D and G is defined as

$$\mathcal{C}_{\mathcal{L}}(D, G) = \{\operatorname{ev}_D(f) \in \mathbb{F}^n \mid f \in \mathcal{L}(G)\} ,$$

where $\operatorname{ev}_D(f) = (f(P_1), \ldots, f(P_n))$. We will sometimes refer to $\mathcal{L}(G)$ as the *message space*.

The ordering of the places $P_1, \ldots, P_n$ in Definition 1.12 has no impact on the interesting properties of the code such as length, dimension and minimum distance. When convenient, therefore, we will allow ourselves to index the entries of codewords using the places in $\mathcal{D} := \operatorname{supp} D$ instead of the integers $1, \ldots, n$, writing

$$\mathcal{C}_{\mathcal{L}}(D, G) = \{(f(P))_{P \in \mathcal{D}} \in \mathbb{F}^{\mathcal{D}} \mid f \in \mathcal{L}(G)\} ,$$

where $\mathbb{F}^{\mathcal{D}} \approx \mathbb{F}^n$ is the $\mathbb{F}$-vector space of all maps $\mathcal{D} \to \mathbb{F}$. This notation will only be used in Chapter 2, however.

It is well known that the AG code $\mathcal{C}_{\mathcal{L}}(D, G)$ has minimum distance at least $d^\star := n - \deg(G)$ and dimension $k = \mathfrak{l}(G) - \mathfrak{l}(G - D)$; the latter follows from the fact that the evaluation map ev_D has kernel $\mathcal{L}(G - D)$. Using [Sti09, Theorem 1.5.17], we see that $k = n$, i.e. $\mathcal{C}_{\mathcal{L}}(D, G) = \mathbb{F}^n$, whenever $\deg G \geqslant n + 2g - 1$, and because of this, we may assume that $0 \leqslant \deg(G) \leqslant n + 2g - 1$.

We conclude this section by showing how $\mathcal{C}_{\mathcal{L}}(D, G)$ can be represented using a generator matrix.

Proposition 1.13. *If $f_1, \ldots, f_k$ is an $\mathbb{F}$-basis of $\mathcal{L}(G)$ with $\deg G < n$, then*

$$\boldsymbol{G} = \begin{pmatrix} f_1(P_1) & \cdots & f_1(P_n) \\ \vdots & \ddots & \vdots \\ f_k(P_1) & \cdots & f_k(P_n) \end{pmatrix} \in \mathbb{F}^{k \times n}$$

is a generator matrix of $\mathcal{C}_{\mathcal{L}}(D, G)$.

Proof. It is clear that the row space of $\boldsymbol{G}$, i.e. the set of all of the $\mathbb{F}$-linear combinations of its rows, is contained in $\mathcal{C}_{\mathcal{L}}(D, G)$ since $f_1, \ldots, f_k \in \mathcal{L}(G)$. For the opposite inclusion, let $\boldsymbol{c} = (f(P_1), \ldots, f(P_n)) \in \mathcal{C}_{\mathcal{L}}(D, G)$, where $f \in \mathcal{L}(G)$, and observe that writing $f = \sum_{j=1}^{k} m_j f_j$, where $\boldsymbol{m} = (m_j)_{j=1}^{k} \in \mathbb{F}^k$, implies that

$$\boldsymbol{m}\boldsymbol{G} = \sum_{j=1}^{k} m_j (f_j(P_1), \ldots, f_j(P_n)) = (f(P_1), \ldots, f(P_n)) = \boldsymbol{c} .$$

The constraint $\deg G < n$ ensures that $\boldsymbol{G}$ has full rank. $\square$

1.7 Some properties of function fields

In this section, we present some definitions and facts about function fields, which will be used later in this thesis.

Definition 1.14. For any rational place P and any divisor A, let

$$\text{Я}_P(A) = \bigcup_{m=-\infty}^{\infty} \mathcal{L}(mP + A) \ .$$

Furthermore, for any function $f \in \text{Я}_P(A)$, let $\delta_A^{(P)}(f)$ be the smallest integer $m \in \mathbb{Z}$ such that $f \in \mathcal{L}(mP + A)$, i.e. $\delta_A^{(P)}(f) = -v_P(f) - v_P(A)$. We will use as convention that $\delta_A^{(P)}(0) = -\infty$. As a shorthand, we will write $\text{Я}_P = \text{Я}_P(0)$ and $\delta^{(P)}(f) = \delta_0^{(P)}(f)$.

In the context of Definition 1.14, it is not hard to see that Я_P is a ring. Moreover, this ring is intimately related with the Weierstrass semigroup at P:

Definition 1.15. The *Weierstrass semigroup* at a rational place P of F is

$$\mathcal{W}(P) = \{\delta^{(P)}(f) \mid f \in \text{Я}_P\} \ .$$

The following proposition allows us to conveniently describe Я_P using $\mathcal{W}(P)$:

Proposition 1.16 ([SH95])**.** *If P is a rational place and $x_1, \ldots, x_t \in \text{Я}_P$ are functions whose pole orders at P generate $\mathcal{W}(P)$, i.e.*

$$\langle \delta^{(P)}(x_1), \ldots, \delta^{(P)}(x_t) \rangle_{\mathbb{Z}_{\geqslant 0}} = \mathcal{W}(P) \ ,$$

then $\text{Я}_P = \mathbb{F}[x_1, \ldots, x_t]$.

Proof. It is clear that $\mathbb{F}[x_1, \ldots, x_t] \subset \text{Я}_P$. For the opposite inclusion, with the aim of reaching a contradiction, let $f \in \text{Я}_P \backslash \mathbb{F}[x_1, \ldots, x_t]$ be such that $\delta^{(P)}(f)$ is minimal, and write

$$\delta^{(P)}(f) = \sum_{i=1}^{t} n_i \delta^{(P)}(x_i) = \delta^{(P)}\left(\prod_{i=1}^{t} x_i^{n_i}\right) \ ,$$

where $n_i \in \mathbb{Z}_{\geqslant 0}$. Since $\delta^{(P)}(f) > 0$ (otherwise $f \in \mathbb{F}$), at least one of the $n_i > 0$, which guarantees existence a constant $c \in \mathbb{F}$ such that $\delta^{(P)}(h) < \delta_p(f)$, where

$$h = f - c\prod_{i=1}^{t} x_i^{n_i} \in \text{Я}_P \ ,$$

and due to the minimality of f, then necessarily $h \in \mathbb{F}[x_1, \ldots, x_t]$. However, this is a contradiction as it implies that

$$f = h + c \prod_{i=1}^{t} x_i^{n_i} \in \mathbb{F}[x_1, \ldots, x_t] \ .$$

□

We end this introductory chapter with two convenient lemmas on multi-point evaluation and interpolation in the context of function fields.

Lemma 1.17. *Let $P, E_1, \ldots, E_N$ be pairwise distinct rational places, and let $E = E_1 + \cdots + E_N$ and A be divisors with $\operatorname{supp}(E) \cap \operatorname{supp}(A) = \varnothing$. Denoting by $\operatorname{ev}_{E,A} : \mathcal{L}(A) \to \mathbb{F}^N$ the restriction of the evaluation map ev_E to $\mathcal{L}(A)$, we have the following:*

1. $\operatorname{ev}_{E,A}$ *is injective when* $\deg A < \deg E$,
2. $\operatorname{ev}_{E,A}$ *is surjective when* $\deg A \geqslant \deg E + 2g - 1$.

Proof. For the first item, simply note that the dimension of the kernel of $\operatorname{ev}_{E,A}$ is $\mathfrak{l}(A - E) = 0$, since $\deg(A - E) < 0$. For the second item, observe that the dimension of the image of $\operatorname{ev}_{E,A}$ is

$$\mathfrak{l}(A) - \mathfrak{l}(A - E) = \deg A - g + 1 - (\deg A - \deg E - g + 1) = \deg E = N \ ,$$

since $\deg A \geqslant 2g - 1$ and $\deg(A - E) \geqslant 2g - 1$, see [Sti09, Theorem 1.5.17]. □

Lemma 1.18. *If $P, E_1, \ldots, E_N$ are pairwise distinct rational places, and if A and $E = E_1 + \cdots + E_N$ are divisors with $\operatorname{supp}(E) \cap \operatorname{supp}(A) = \varnothing$, then for any $(w_1, \ldots, w_N) \in \mathbb{F}^N$ there exists a function $a \in \mathcal{R}_P(A)$ satisfying*

$$\begin{aligned} \delta_A^{(P)}(a) &\leqslant \deg E + 2g - 1 - \deg A \quad \textit{and} \\ a(E_j) &= w_j \quad \textit{for } j = 1, \ldots, N \ . \end{aligned}$$

Proof. Letting $A' = (\deg E + 2g - 1 - \deg A)P + A$ we get that

$$\deg A' \geqslant \deg E + 2g - 1 \ ,$$

which according to Lemma 1.17 implies that $\operatorname{ev}_{E,A'}$ is surjective. □

Chapter 2

Encoding and unencoding of one-point AG codes over $C_{a,b}$ curves

This chapter is based on [BRS20] and investigates encoding and unencoding of one-point AG codes defined over special plane curves called $C_{a,b}$ curves. First introduced in [Miu93, MK93], and also occurring as a special case of curves studied in [FR94, HvLP98], they have been shown to give rise to codes that can be described using bivariate polynomials over $\mathbb{F}$, making them considerably simpler to work with than general AG codes – especially from the computational point of view. Taking advantage of this, we design algorithms that efficiently solve Problem 1.9 on page 9 and Problem 1.11 on page 10, achieving quasi-linear complexity for important cases of $C_{a,b}$ curves such as the Hermitian curve and norm-trace curves. We also beat the naive algorithms for encoding and unencoding when working over curves that have many rational points, although in this case we do not always achieve quasi-linear cost.

2.1 $C_{a,b}$ curves and their codes

2.1.1 Geometry and defining polynomial

The family of $C_{a,b}$ curves consists of well behaved plane algebraic curves. In the projective setting, they have exactly one point at infinity, which, if singular is a cusp; all other points are nonsingular. In the affine setting a $C_{a,b}$ curve is defined by the set of zeroes of a bivariate polynomial $Q \in \mathbb{F}[X, Y]$ with a carefully constrained monomial support, where for any $Q = \sum_{i,j} c_{i,j} X^i Y^j \in \mathbb{F}[X, Y]$ with $c_{i,j} \in \mathbb{F}$ its monomial support is $\operatorname{supp} Q = \{X^i Y^j \mid c_{i,j} \neq 0\}$. Curve points whose coordinates belong to $\mathbb{F}$ are called *rational* (in contrast to points over extension fields). Below we define $C_{a,b}$ curves in full detail.

Definition 2.1. Let $a, b \in \mathbb{Z}_{>0}$ be such that $\gcd(a, b) = 1$. An algebraic curve is called a $C_{a,b}$ *curve* if it is defined by an equation of the form $Q(X, Y) = 0$, where $Q \in \mathbb{F}[X, Y]$ satisfies the following properties:

1) $X^b, Y^a \in \operatorname{supp} Q$,

2) $X^i Y^j \in \operatorname{supp} Q \implies ai + bj \leqslant ab$,

3) $\langle Q, \frac{\partial Q}{\partial X}, \frac{\partial Q}{\partial Y} \rangle_{\mathbb{F}[X,Y]} = \mathbb{F}[X, Y]$.

We will refer to Q as the *defining polynomial* of the curve.

Remark 2.2. As we will see later in the chapter, our algorithms are not invariant under swapping of X and Y, as their complexities typically depend only a and not b. It is therefore sensible to assume without loss of generality that $a < b$ by simply swapping the variables accordingly. Note that the case $a = b$ is only possible when $a = b = 1$, since a and b are coprime. We will disregard this degenerate case.

For any polynomial $f \in \mathbb{F}[X, Y]$ we denote by $\delta(f)$ the (a, b)-weighted degree of f, i.e. $\delta(X^i Y^j) = ai + bj$ for any $i, j \in \mathbb{Z}_{\geqslant 0}$ (which extends to the whole of $\mathbb{F}[X, Y]$ in the obvious way). The first two conditions in Definition 2.1 imply that $Q = \alpha X^b + \beta Y^a + \widehat{Q}$ with $\widehat{Q} \in \mathbb{F}[X, Y]$ having $\delta(\widehat{Q}) < ab$, which due to [HvLP98, Corollary 3.18] guarantees that Q is absolutely irreducible. The theory of Newton polygons, i.e. the convex hull of $\{(i, j) \mid X^i Y^j \in \operatorname{supp} Q\}$, can also be used to conclude this [Gao01].

Example 2.3. The polynomial

$$\begin{aligned} Q = \quad & Y^4 \\ & +4XY^3 \\ & +2Y^2 + XY^2 + 3X^2Y^2 \\ & +2XY + 3X^2Y + 5X^3Y \\ & +4 + X + 6X^2 + 5X^3 + 4X^4 + X^5 \in \mathbb{F}_7[X,Y] \end{aligned}$$

defines a $C_{a,b}$ curve with $a = 4$ and $b = 5$. It has 21 rational points, which are depicted in Figure 2.1. It was obtained by randomly sampling the space of valid $C_{a,b}$ curves more than 10^6 times, searching for curves with many points.

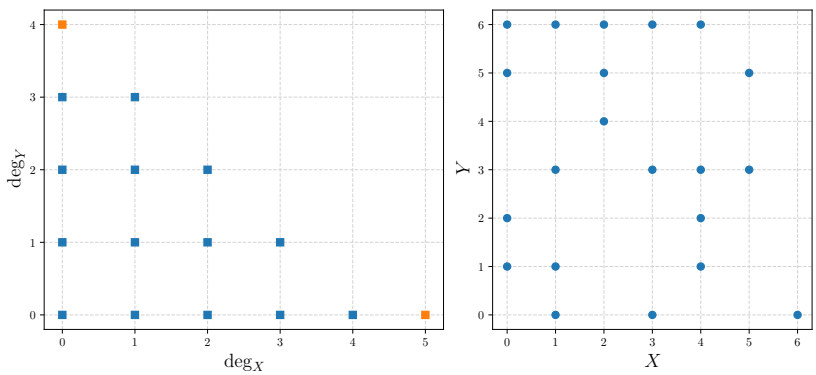

Figure 2.1: To the left: Allowed monomial support for a defining polynomial of a $C_{a,b}$ curve with $a = 4$ and $b = 5$. The monomials X^5 and Y^4 (orange) are compulsory.
To the right: The 20 finite rational points on the $C_{a,b}$ curve from Example 2.3. The underlying field is $\mathbb{F}_7$.

2.1.2 One-point codes

For any $C_{a,b}$ curve with defining polynomial Q we construct the corresponding function field $F = \mathbb{F}(x, y)$ by extending the rational function field $\mathbb{F}(x)$ with a new element y satisfying $Q(x, y) = 0$, which means that

$$\mathbb{F}(x,y) \approx \mathrm{Frac}(\mathbb{F}[X,Y]/\langle Q\rangle) = \{\frac{f}{h} \mid f, g \in \mathbb{F}[X,Y]/\langle Q\rangle \text{ with } g \neq 0\} \ .$$

Since Q is absolutely irreducible, $\mathbb{F}$ is the full constant field of F. For the remainder of this chapter we consider F to be fixed, and we list some of its well-known properties in the following proposition without giving proof, though details can be found in [Miu93, MK93, HvLP98]

Proposition 2.4. *F has genus $g = \frac{1}{2}(a-1)(b-1)$. There is exactly one rational place P_∞ such that $\delta_{P_\infty}(x), \delta_{P_\infty}(y) > 0$, and if P_∞ is such a place, then $\delta_{P_\infty}(x) = a$, $\delta_{P_\infty}(y) = b$ and $\mathcal{W}(P_\infty) = \langle a, b \rangle_{\mathbb{Z}_{\geqslant 0}}$.*

We will treat the rational place P_∞ from Proposition 2.4 as fixed, referring to it as *the place at infinity*; all the remaining rational places will be referred to as *finite*. The codes that we consider are of the form $\mathcal{C}_{\mathcal{L}}(D, mP_\infty)$, where $m \in \mathbb{Z}_{\geqslant 0}$, $D = P_1 + \cdots + P_n$ and $P_1, \ldots, P_n$ are finite rational places. Efficient algorithms for encoding and unencoding will consequently require a practical way of representing elements from the message space $\mathcal{L}(mP_\infty)$, which, as we have alluded to in the beginning of the chapter, can be done using bivariate polynomials in $\mathbb{F}[X, Y]$. We describe this representation in full detail below, starting from the allowed exponents in the monomial support of these polynomials: For the remainder of this chapter let

$$\mathcal{B}_m = \{(i, j) \in \mathbb{Z}^2_{\geqslant 0} \mid j < a \text{ and } ai + bj \leqslant m\} \ .$$

Lemma 2.5. *If $(i, j), (i', j') \in \mathcal{B}_m$ are such that $ai + bj = i'a + j'b$, then $(i, j) = (i', j')$.*

Proof. Since a and b are coprime, then $\operatorname{lcm}(a, b) = ab$. If $(i - i')a + (j - j')b = 0$, then necessarily $(i - i') \mid b$ and $(j - j') \mid a$. However, by definition of $\mathcal{B}_m$ it holds that $0 \leqslant j, j' < a$, which implies that $j = j'$, and consequently also $i = i'$. □

Proposition 2.6. *$\mathcal{M}_m := \{x^i y^j \mid (i, j) \in \mathcal{B}_m\}$ is an $\mathbb{F}$-basis of $\mathcal{L}(mP_\infty)$. Furthermore, assuming that $m < n$, if $\varphi : \mathcal{B}_m \to \{1, \ldots, k\}$ is some bijection and $\boldsymbol{G} \in \mathbb{F}^{k \times n}$, where $k = |\mathcal{B}_m| = \mathfrak{l}(mP_\infty)$, is the matrix with the $\varphi(i, j)$-th row being*

$$(x(P_1)^i y(P_1)^j, \ldots, x(P_n)^i y(P_n)^j) \in \mathbb{F}^n \quad \text{for } (i, j) \in \mathcal{B}_m \ ,$$

then $\boldsymbol{G}$ is a generator matrix for $\mathcal{C}(D, mP_\infty)$.

Proof. According to Proposition 2.4, $\mathcal{W}(P_\infty) = \langle \delta_{P_\infty}(x), \delta_{P_\infty}(y) \rangle_{\mathbb{Z}_{\geqslant 0}}$, which by Proposition 1.16 implies that $\mathcal{R}_{P_\infty} = \mathbb{F}[x, y]$, and consequently that

$$\mathcal{L}(mP_\infty) = \{f \in \mathbb{F}[x, y] \mid \delta_{P_\infty}(f) \leqslant m\} \ .$$

Since $\mathbb{F}[x, y] \approx \mathbb{F}[X, Y]/\langle Q \rangle$, where $Q = \alpha X^b + \beta Y^a + \widehat{Q} \in \mathbb{F}[X, Y]$ is the curve polynomial with $\alpha, \beta \in \mathbb{F} \backslash \{0\}$ and $\widehat{Q} \in \mathbb{F}[X, Y]$ satisfying $\delta(\widehat{Q}) < ab$, and therefore $\deg_Y \widehat{Q} < a$, it is not hard to see that

$$\mathbb{F}[x, y] = \{f \in \mathbb{F}[x, y] \mid \deg_y f < a\} \ .$$

It follows that $\mathcal{L}(mP_\infty)$ is generated by $\mathcal{M}_m$ over $\mathbb{F}$, and since Lemma 2.5 guarantees that for any distinct $x^i y^j, x^{i'} y^{j'} \in \mathcal{M}_m$ it holds that $\delta_{P_\infty}(x^i y^j) \neq \delta_{P_\infty}(x^{i'} y^{j'})$, then $\mathcal{M}_m$ is indeed an $\mathbb{F}$-basis.

The claim that $\boldsymbol{G}$ is a generator matrix is then a direct consequence of Proposition 1.13. □

The following lemma, due to [MK93, SH95], allows us to represent every message in $\mathcal{L}(mP_\infty)$ using a polynomial in $\mathbb{F}[X,Y]$, and every rational place P using a point in $\mathbb{F}^2$, thereby avoiding the computationally challenging language of function fields in our algorithms.

Lemma 2.7. *If P is a finite rational place and*

$$f = \sum_{(i,j)\in\mathcal{B}_m} m_{i,j} x^i y^j \in \mathcal{L}(mP_\infty) \quad \text{with } m_{i,j} \in \mathbb{F} \ ,$$

then $f(P) = \widehat{f}(\widehat{P})$, where $\widehat{P} = (x(P), y(P)) \in \mathbb{F}^2$ is a zero of Q and

$$\widehat{f} = \sum_{(i,j)\in\mathcal{B}_m} m_{i,j} X^i Y^j \in \mathbb{F}[X,Y] \ .$$

Armed with Lemma 2.7 we can reinterpret our setting in the following way: The message space, which used to be $\mathcal{L}(mP_\infty)$, will be replaced by

$$\begin{aligned}\mathfrak{L}(m) &:= \{ \sum_{(i,j)\in\mathcal{B}_m} m_{i,j} X^i Y^j \mid m_{i,j} \in \mathbb{F}\} \\ &= \{f \in \mathbb{F}[X,Y] \mid \deg_Y f < a \text{ and } \delta(f) < m\} \ ,\end{aligned}$$

while the code $\mathcal{C}_{\mathcal{L}}(D, mP_\infty)$, being technically unchanged, will henceforth be described as

$$\mathcal{C}_Q(\mathcal{P}, m) := \{(f(P_1), \ldots, f(P_n)) \in \mathbb{F}^n \mid f \in \mathfrak{L}(m)\} \ ,$$

where $\mathcal{P} = \{P_1, \ldots, P_n\} \subset \mathbb{F}^2$ is now a subset of the rational points on the underlying $C_{a,b}$ curve. Notice in particular that for any $f(X,Y) \in \mathfrak{L}(m)$ it holds that $f(x,y) \in \mathcal{L}(mP_\infty)$ with $\delta(f(X,Y)) = \delta_{P_\infty}(f(x,y))$. Under this interpretation, encoding and unencoding become respectively as follows:

Problem 2.8 (Encoding)**.** Given a message $f \in \mathfrak{L}(m)$ compute the codeword $\boldsymbol{c} = (f(P_1), \ldots, f(P_n)) \in \mathcal{C}(m, \mathcal{P})$.

Problem 2.9 (Unencoding)**.** Given a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}(m, \mathcal{P})$ compute the unique message $f \in \mathfrak{L}(m)$ satisfying $f(P_i) = c_i$ for $i = 1, \ldots, n$.

From the perspective of computer algebra, Problems 2.8 and 2.9 can be recognized as special cases of bivariate polynomial *multi-point evaluation* (MPE) and *interpolation* respectively. We will favor this point of view throughout the chapter, only recalling the AG context when advantageous for analysis.

2.2 Related work

2.2.1 Encoding

For the particularly simple case of RS codes, it is well known that encoding can be done with quasi-linear cost [Jus76] using fast univariate MPE. Certain AG codes have been shown to admit a space-efficient encoding algorithm using Gröbner bases and high-order automorphisms of the code instead of storing a $k \times n$ generator matrix [HLS95]; the computational cost of this approach, however, is quadratic.

It has been demonstrated that one-point Hermitian codes can be encoded with sub-quadratic complexity by interpreting them as concatenated RS codes [YB92]. The Hermitian curve, which gives rise to these codes, is a $C_{a,b}$ curve over $\mathbb{F} = \mathbb{F}_{q^2}$ with $a = q$, $b = q + 1$ and defining polynomial $Q = Y^q + Y - X^{q+1}$. Consequently, any message $f \in \mathfrak{L}(m) \subseteq \mathbb{F}[X, Y]$ can be written as $f = \sum_{i=1}^{\kappa} f_i(X)Y^i$, where $\kappa = \min\{q - 1, \lfloor m/(q + 1)\rfloor\}$ and where each $f_i \in \mathbb{F}[X]$ can be regarded as a message in an RS code, giving rise to the codeword

$$\boldsymbol{c}^{(i)} = (f_i(\alpha_1), \ldots, f_i(\alpha_{q^2})) \in \mathbb{F}^{q^2} \ ,$$

where $\{\alpha_1, \ldots, \alpha_{q^2}\} = \mathbb{F}$ is the set of all X-coordinates occurring on the Hermitian curve. There are exactly q points on the curve for every one of these X-coordinates, meaning that the evaluations of f_iY^i on all of the points can be obtained by a q-fold concatenation of $\boldsymbol{c}^{(i)}$ with itself, scaling each entry by the i-th power of the Y-coordinate of the point corresponding to that entry. Summing these concatenated (and entry-wise scaled) codewords for $i = 1, \ldots, \kappa$, the sought codeword $\boldsymbol{c} = \mathrm{Enc}(f) \in \mathcal{C} \subset \mathbb{F}^{q_3}$ of the one-point Hermitian code is obtained. Using fast RS encoding, the cost of this approach is $\widetilde{\mathcal{O}}(\kappa q q^2) \subseteq \widetilde{\mathcal{O}}(q^4) = \widetilde{\mathcal{O}}(n^{4/3})$. Though the underlying principle of our algorithm has similarities with this approach, our algorithm is a factor $\mathcal{O}(n^{1/3})$ faster for the Hermitian curve.

In [RM01] the results of [YB92] were generalized to arbitrary one-point AG codes, though the complexity is never quasi-linear.

2.2.2 Bivariate multi-point evaluation

In Section 2.1.2 we observed that encoding of one-point codes over $C_{a,b}$ curves can be formulated as Problem 2.8, which is but a variant of bivariate MPE. Since any algorithm for bivariate MPE can be applied to our encoding problem, let us mention some of the existing work on this well studied topic.

Given a point set $\mathcal{P} \subseteq \mathbb{F}^2$ and a polynomial $f \in \mathbb{F}[X, Y]$ with $\deg_X f = d_X$ and $\deg_Y f = d_Y$, bivariate MPE seeks to compute $(f(P))_{P \in \mathcal{P}} \in \mathbb{F}^n$, where $n = |\mathcal{P}|$. In the context of one-point codes over $C_{a,b}$ curves we are interested in the case where $d_X d_Y < n$ and $d_Y \ll d_X$. The former assumption is common in the literature, however, the latter is frequently replaced with $d_X \approx d_Y$, which would often result in poor performance in our use case.

Existence of a quasi-linear solution to univariate MPE continues to persuade the computer algebra community to entertain the hope for a bivariate (and multivariate) algorithm with complexity $\widetilde{\mathcal{O}}(d_X d_Y + n)$, though finding such an algorithm – devoid of restrictive assumptions on the input – remains an open problem. As in the univariate case, the naive approach computes all of the evaluations individually using a bivariate form of Horner's rule. Each evaluation costs $\mathcal{O}(d_X d_Y)$ operations in $\mathbb{F}$, amounting to a total complexity $\mathcal{O}(n d_X d_Y) \subseteq \mathcal{O}(n^2)$, assuming that $d_X d_Y < n$.

Pan

One of the first major improvements on the naive algorithm was due to Pan [Pan94], who demonstrated that quasi-linear complexity could be achieved for the special case of $\mathcal{P} = \mathcal{P}_X \times \mathcal{P}_Y$ with $\mathcal{P}_X, \mathcal{P}_Y \subseteq \mathbb{F}$, i.e. when $\mathcal{P}$ is a grid, see Figure 2.2. His algorithm is frequently refereed to as a "tensored" form of univariate MPE and in its general form works in any number of variables. The assumption that $\mathcal{P}$ is a grid can be removed by executing the algorithm on the smallest grid $\widehat{\mathcal{P}}$ containing all of $\mathcal{P}$, discarding the extraneous evaluations from the output; however, in the worst case it can happen that $|\widehat{\mathcal{P}}| = n^2$, resulting in the complexity $\widetilde{\mathcal{O}}(d_X d_Y + n^2)$, i.e. quadratic in the input size when $d_X d_Y < n$. It is by no means the case that all interesting $C_{a,b}$ curves fall into this worst case category: for the Hermitian curve $n = q^3$ and $\widehat{\mathcal{P}} = \mathbb{F}_{q^2}^2$, resulting in the cost $\widetilde{\mathcal{O}}(n^{4/3})$, as in [YB92]. Our MPE algorithm, presented in Section 2.4, essentially generalizes the bivariate form of Pan's algorithm in a way which achieves quasi-linear cost on point sets with a *semi-grid* structure; see Figure 2.2. A semi-grid is a weaker notion than a grid, so our algorithm never has worse complexity than Pan's. On the other hand, there are important families of $C_{a,b}$ curves – including

the Hermitian curve – whose points do in fact form a semi-grid, allowing us to achieve encoding with quasi-linear cost.

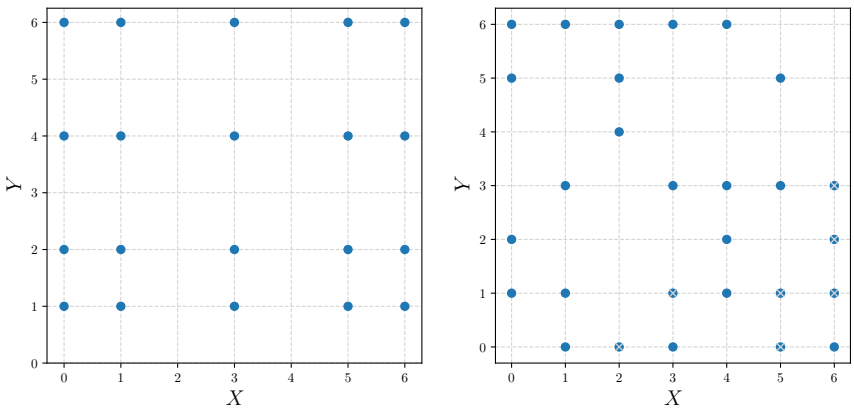

Figure 2.2: To the left: a grid $\mathcal{P} = \mathcal{P}_X \times \mathcal{P}_Y \subset \mathbb{F}_7^2$, where $\mathcal{P}_X = \{0, 1, 3, 5, 6\}$ and $\mathcal{P}_Y = \{1, 2, 4, 6\}$.
To the right: a semi-grid over $\mathbb{F}_7$. For each X-coordinate there are exactly 4 points having that X-coordinate. The points with a gray cross do *not* lie on the curve from Example 2.3. Since there are only few of these points, encoding over this curve can be done quite fast. The smallest grid that covers the curve is $\mathbb{F}_7^2$.

Nüsken & Ziegler

A completely different approach to bivariate MPE was due to Nüsken & Ziegler [NZ04]. By demonstrating that the problem can be reduced to a variant of *bivariate modular composition*, they managed to avoid restrictive assumptions on the underlying point set, albeit at the cost of increased complexity. It turns out that MPE can be solved in full generality by first considering the simpler case where $\mathcal{P} = \{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\}$ with the α_i being pairwise distinct. Letting $h = \prod_{i=1}^{n}(X - \alpha_i) \in \mathbb{F}[X]$ and $p \in \mathbb{F}[X]$ with $p(\alpha_i) = \beta_i$ for $i = 1, \ldots, n$, it is easy to see that if $r = f(X, p) \operatorname{rem} h \in \mathbb{F}[X]$, then $r(\alpha_i) = f(\alpha_i, p(\alpha_i)) = f(\alpha_i, \beta_i)$ for each i. Computing h and p costs $\widetilde{\mathcal{O}}(n)$ operations in $\mathbb{F}$ – the same goes for the evaluations $r(\alpha_1), \ldots, r(\alpha_n) \in \mathbb{F}$. Whether the modular composition problem of computing r admits a solution with quasi-linear cost remains an open problem, however, for any $\epsilon > 0$, Nüsken & Ziegler achieve

$$\mathcal{O}\big((d_X d_Y^{\omega_2/2} + n d_Y^{\omega_2/2-1})^{1+\epsilon}\big) \subset \mathcal{O}(d_X d_Y^{1.6284} + n d_Y^{0.6284}) \ ,$$

where $\omega_2 < 3.2567$ [Le 12] is such that the cost of multiplying a $t \times t$ matrix with a $t \times t^2$ matrix is $\widetilde{\mathcal{O}}(t^{\omega_2})$.

We can reduce the general problem of bivariate MPE to the aforementioned case with the α_i pairwise distinct by *shearing* the points in a degree 2 extension field $\mathbb{K}/\mathbb{F}$ using the map $(\alpha_i, \beta_i) \mapsto (\alpha_i + \theta\beta_i, \beta_i) \in \mathbb{K}^2$, where $\theta \in \mathbb{K}\backslash\mathbb{F}$. By replacing f with $\widetilde{f} := f(X - \theta Y, Y) \in \mathbb{K}[X, Y]$ we can then obtain the sought evaluations of f by evaluating $\widetilde{f}$ on the sheared points, since

$$\widetilde{f}(\alpha_i + \theta\beta_i, \beta_i) = f(\alpha_i + \theta\beta_i - \theta\beta_i, \beta_i) = f(\alpha_i, \beta_i) \ .$$

The total cost of shearing all of the points is $\mathcal{O}(n)$, and since each operation in $\mathbb{K}$ costs $\mathcal{O}(1)$ operations in $\mathbb{F}$, this strategy makes no impact on the asymptotic complexity of the algorithm except for the following issue: Generically, it will be the case that $\deg_Y \widetilde{f} = \max\{d_X, d_Y\}$, which means that the cost of computing (and storing) $\widetilde{f}$ will be high when $d_Y \ll d_X$. If, for example, $d_X \approx n^{1-\delta}$ and $d_Y \approx n^{\delta}$ for some $0 < \delta \ll 1$, then the number of coefficients in $\mathbb{K}$ needed to represent $\widetilde{f}$ is roughly $n^{2-2\delta}$, i.e. nearly quadratic in the input size. In our running example of one-point Hermitian codes, where $d_X < q^2$ and $d_Y < q$, for any $\epsilon > 0$ the shearing strategy results in the complexity

$$\mathcal{O}\big((q^2 q^{\omega_2} + q^3 q^{\omega_2 - 2})^{1+\epsilon}\big) = \mathcal{O}\big((q^{2+\omega_2})^{1+\epsilon}\big) \subset \mathcal{O}(q^{5.2567}) = \mathcal{O}(n^{1.7523}) \ ,$$

which is better than the naive approach, but not as good as Pan's algorithm with cost $\widetilde{\mathcal{O}}(n^{4/3}) \subset \mathcal{O}(n^{1.3334})$. Of course, we can always avoid the case $d_Y < d_X$ by simply swapping the variables in the input polynomial as well the coordinates of the points, preserving the sought evaluations while guaranteeing that $d_Y \geqslant d_X$. However, instead of nullifying it, this approach merely substitutes the computational penalty of shearing with another one – caused by the asymmetry of the cost bound with respect to the two variables. Doing this for one-point Hermitian codes gives complexity

$$\mathcal{O}\big((q q^{\omega_2} + q^3 q^{\omega_2 - 2})^{1+\epsilon}\big) = \mathcal{O}\big((q^{\omega_2 + 1})^{1+\epsilon}\big) = \mathcal{O}(q^{4.2567}) = \mathcal{O}(n^{1.4189})$$

for any $\epsilon > 0$, which is a notable improvement, though still inferior to Pan's algorithm.

Kedlaya and Umans

Last but not least, we mention the celebrated result [KU08] by Kedlaya and Umans, which in the bivariate[1] setting gives an MPE algorithm with complex-

[1] Actually, the algorithm by Kedlaya and Umans works with any number of variables.

ity[2]

$$\mathcal{O}\big((\max\{d_X, d_Y\}^2 + n)^{1+\epsilon}\big)$$

for any $\epsilon > 0$. In outline, the algorithm works over prime fields by lifting the data to integers, performing MPE several times modulo many small primes, and then reassembling the result using the Chinese Remainder theorem. This technique is then generalized to also work for extension fields. Although the cost of this approach is for all intents and purposes quasi-linear when $d_X \approx d_Y$, in the case of one-point Hermitian codes it is $\mathcal{O}(n^{4/3+\epsilon})$, which is ever so slightly worse than for Pan's algorithm. Another unfortunate, but not insignificant drawback is that the asymptotic complexity seems to be hiding an impractically large constant factor [vdHL19, Conclusion], making Kedlaya and Umans' algorithm less interesting for real world applications.

Remark 2.10. There is a recent bivariate MPE algorithm by van der Hoeven and Lecerf [vdHL21] which can evaluate any polynomial $f \in \mathbb{F}[X, Y]$ satisfying $\deg_X f, \deg_Y f < \sqrt{n}$ using $\mathcal{O}(\mathsf{M}(n \log n) \log(n)^3) \subset \tilde{\mathcal{O}}(n)$ operations in $\mathbb{F}$ – that is if the underlying point set is available for precomputation. No detailed comparison will be made with this results, however, we will mention that the cost of encoding one-point Hermitian codes using this algorithm becomes $\tilde{\mathcal{O}}(n^{4/3})$, as is with Pan's.

2.2.3 Bivariate interpolation

Having seen in Section 2.2.2 that encoding of one-point codes over $C_{a,b}$ curves – formulated as Problem 2.8 – can be directly solved by any algorithm for bivariate MPE, let us now turn our attention to existing work on bivariate interpolation, which can analogously be used to address Problem 2.9, i.e. unencoding. Our setting here is as follows: we are given a finite set of points $\mathcal{P} = \{P_1, \ldots, P_n\} \subseteq \mathbb{F}^2$ alongside with some prescribed evaluations $v_1, \ldots, v_n \in \mathbb{F}$, and we wish to compute a polynomial $f \in \mathbb{F}[X, Y]$ with $f(P_i) = v_i$ for $i = 1, \ldots, n$. Typically, as in our use case with unencoding, it is of interest to place restrictions on the monomial support of the sought interpolating polynomial f. At the time of writing, there is no better way of solving this interpolation problem in its full generality besides interpreting it as a $t \times n$ linear system over $\mathbb{F}$, where t is the cardinality of the allowed monomial support. Assuming that $t \leqslant n$, Gaussian elimination then yields the cost $\mathcal{O}(n^\omega) \subset \mathcal{O}(n^{2.3729})$. In the context of codes, we can instead multiply the codeword with the $n \times k$ inverted generator matrix at the improved cost $\mathcal{O}(nk) \subseteq \mathcal{O}(n^2)$. We can do notably better by observing that Problem 2.9 gives rise to a linear system with low *displacement*

[2]Their complexity model differs from ours by counting bit operations instead of field operations, but this has no impact on our use case.

rank, namely a, allowing us to use the algorithm for structured system solving by Bostan *et al.* [BJMS17] with cost $\widetilde{\mathcal{O}}(a^{\omega-1}n)$. For one-point Hermitian codes this translates to $\widetilde{\mathcal{O}}(q^{\omega-1}n) = \widetilde{\mathcal{O}}(q^{2+\omega}) \subset \mathcal{O}(n^{1.4577})$ which is inferior to the quasi-linear complexity of our algorithm; however, for general one-point codes over $C_{a,b}$ curves this is our main contender. We compare the two approaches in Section 2.5.3, and – as we will see – for most parameters of interest, our algorithm seems to be faster.

Other noteworthy interpolation techniques include Pan's [Pan94], which assumes that the underlying point set forms a grid, and its generalization by van der Hoeven and Schost [vdHS13], which works on certain structured subsets of grids. Both of these algorithms have quasi-linear cost in the input size, however, they are unable to guarantee that the returned interpolating polynomial will satisfy the monomial support requirements mandated by Problem 2.9. Fortunately, our setting allows us to address this issue in the following way: Given a polynomial $\widehat{f} \in \mathbb{F}[X,Y]$ which correctly evaluates to the prescribed interpolation values but has incorrect monomial support, we can always find the sought f in the coset $\widehat{f} + \Gamma(\mathcal{P})$ of the vanishing ideal $\Gamma(\mathcal{P})$ of $\mathcal{P}$ by reducing $\widehat{f}$ with respect to a Gröbner basis of $\Gamma(\mathcal{P})$ under a suitable monomial ordering. As we will see in Section 2.5.2, by carefully ordering these Gröbner basis elements in a sequence we can guarantee that van der Hoeven's division algorithm [vdH15] yields good complexity when applied to the output of our generalization of Pan's interpolation algorithm. This generalization, similarly to the MPE setting, weakens the requirement of $\mathcal{P}$ being a grid to being a semi-grid. The constructed polynomial can be described by a closed form expression (see Lemma 2.15). A similar expression was used for decoding for one-point Hermitian codes in [LO09], and it was shown how in [NB15] how it can be computed fast; that approach can be seen as a special case of our algorithm.

2.3 Point sets

The complexity estimates of our algorithms do not explicitly depend on the cardinality of the underlying point set. Instead, they are expressed in terms of two geometric quantities: For any point set $\mathcal{P} \subseteq \mathbb{F}^2$ we define

$$n_X = |\mathcal{X}(\mathcal{P})| \quad \text{and} \quad \nu_Y(\mathcal{P}) = \max_{\alpha \in \mathbb{F}} |\mathcal{Y}_\alpha(\mathcal{P})| \ ,$$

where $\mathcal{X}(\mathcal{P}) = \{X(P) \mid P \in \mathcal{P}\} \subseteq \mathbb{F}$ is the set of all X-coordinates in $\mathcal{P}$ and $\mathcal{Y}_\alpha(\mathcal{P}) = \{Y(P) \mid P \in \mathcal{P} \text{ with } X(P) = \alpha\} \subseteq \mathbb{F}$ is the set of all Y-coordinates of those points in $\mathcal{P}$ that have α as their X-coordinate. Whenever it is clear from the context which point set $\mathcal{P}$ we are referring to, we may simply write

$\mathcal{X}, \mathcal{Y}, n_X, \nu_Y$. Note that if all of the points in $\mathcal{P}$ lie on a $C_{a,b}$ curve with defining polynomial $Q \in \mathbb{F}[X,Y]$, then $\nu_Y(\mathcal{P}) \leqslant a$, since for any $\alpha \in \mathbb{F}$ the equation $Q(\alpha, Y) = 0$ has at most $\deg Q(\alpha, Y) \leqslant \deg_Y Q = a$ solutions in $\mathbb{F}$. We will also use the notation $n = n(\mathcal{P}) = |\mathcal{P}|$.

As we will see in the following sections, the complexity estimates in our algorithms favor point sets with $\nu_Y(\mathcal{P}) \approx a$ and $n_X(\mathcal{P})\nu_Y(\mathcal{P}) \approx |\mathcal{P}|$. The following definition captures the latter property:

Definition 2.11. A point set $\mathcal{P} \subseteq \mathbb{F}^2$ is a *semi-grid* if $|\mathcal{Y}_\alpha(\mathcal{P})| = \nu_Y(\mathcal{P})$ for each $\alpha \in \mathcal{X}(\mathcal{P})$, or equivalently if $n_X(\mathcal{P})\nu_Y(\mathcal{P}) = |\mathcal{P}|$.

It is worth noting that any non-singular $C_{a,b}$ curve whose rational points for a semi-grid is a weak Castle curve, see [MUT09].

2.4 Fast encoding using multi-point evaluation

Although we primarily seek an efficient solution to Problem 2.8, our algorithm is more naturally expressed in a slightly more general setting: For now, let $\mathcal{P} \subseteq \mathbb{F}^2$ be an arbitrary point set (not necessarily related to a $C_{a,b}$ curve) and $f \in \mathbb{F}[X,Y]$ with $\deg_X f = d_X$ and $\deg_Y f = d_Y$. Strongly inspired by Pan's MPE algorithm (see Section 2.2.2), our strategy is to write $f = \sum_{i=0}^{d_Y} f_i Y^i$, where $f_i \in \mathbb{F}[X]$, and evaluate each f_i at $\mathcal{X}(\mathcal{P})$ using fast univariate MPE, thereby allowing us to construct the polynomials $f(\alpha, Y) = \sum_{i=0}^{d_Y} f_i(\alpha) Y^i \in \mathbb{F}[Y]$ for $\alpha \in \mathcal{X}(\mathcal{P})$. Evaluating each $f(\alpha, Y)$ at $\mathcal{Y}_\alpha(\mathcal{P})$, again using fast univariate MPE, we obtain the sought evaluations. For a detailed description see the listing of Algorithm 1.

Algorithm 1 BivariateMPE$(f, \mathcal{P})$

Input:

- A polynomial $f = \sum_{i=0}^{d_Y} f_i Y^i \in \mathbb{F}[X, Y]$, where $f_i \in \mathbb{F}[X]$ with $\deg f = d_X$,
- a point set $\mathcal{P} \subset \mathbb{F}^2$ with $\mathcal{X}(\mathcal{P}) = \mathcal{X}$ and $\mathcal{Y}_\alpha(\mathcal{P}) = \mathcal{Y}_\alpha$ for each $\alpha \in \mathbb{F}$

Output:

- evaluations $(f(\alpha, \beta))_{(\alpha,\beta)\in\mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$

1: **for** $i = 0, \ldots, d_Y$ **do**
2: $\quad (f_i^{(\alpha)})_{\alpha\in\mathcal{X}} \in \mathbb{F}^{\mathcal{X}} \leftarrow (f_i(\alpha))_{\alpha\in\mathcal{X}}$ $\quad\triangleright$ using fast univariate MPE
3: **for** each $\alpha \in \mathcal{X}$ **do**
4: $\quad f^{(\alpha,Y)} \in \mathbb{F}[Y] \leftarrow \sum_{i=0}^{d_Y} f_i^{(\alpha)} Y^i$
5: $\quad (f^{(\alpha,\beta)})_{\beta\in\mathcal{Y}_\alpha} \in \mathbb{F}^{\mathcal{Y}_\alpha} \leftarrow (f^{(\alpha,Y)}(\beta))_{\beta\in\mathcal{Y}_\alpha}$ $\quad\triangleright$ using fast univariate MPE
return $(f^{(\alpha,\beta)})_{(\alpha,\beta)\in\mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$

Lemma 2.12. *Algorithm 1 is correct and costs*

$$\mathcal{O}(d_Y \mathsf{M}(d_X + n_X)\log(d_X + n_X) + n_X \mathsf{M}(d_Y + \nu_Y)\log(d_Y + \nu_Y)) \subset \tilde{\mathcal{O}}(d_X d_Y + n_X(d_Y + \nu_Y)) \ .$$

operations in $\mathbb{F}$.

Proof. For correctness, simply observe that for any $(\alpha, \beta) \in \mathcal{P}$

$$f^{(\alpha,\beta)} = f^{(\alpha,Y)}(\beta) = \sum_{i=0}^{d_Y} f_i^{(\alpha)} \beta^i = \sum_{i=0}^{d_Y} f_i(\alpha)\beta^i = f(\alpha, \beta) \ .$$

For the complexity, notice that computational work is performed only in Steps 2 and 5, while all other steps are memory management and therefore cost no operations in $\mathbb{F}$. The total cost of Step 2 over all values of $i = 0, \ldots, d_Y$ is

$$\mathcal{O}(\sum_{i=0}^{d_Y} \mathsf{M}(\deg f_i + n_X)\log(\deg f_i + n_X)) \subseteq \mathcal{O}(d_Y \mathsf{M}(d_X + n_X)\log(d_X + n_X)) \ ,$$

while the total cost of Step 5 over all $\alpha \in \mathcal{X}$ is

$$\mathcal{O}(\sum_{\alpha\in\mathcal{X}} \mathsf{M}(\deg f^{(\alpha,Y)} + |\mathcal{Y}_\alpha|)\log(\deg f^{(\alpha,Y)} + |\mathcal{Y}_\alpha|)) \subseteq \mathcal{O}(n_X \mathsf{M}(d_Y + \nu_Y)\log(d_Y + \nu_Y)) \ .$$

The total cost of the algorithm follows. □

We now specialize Lemma 2.12 to Problem 2.8.

Theorem 2.13. *Problem 2.8 can be solved using Algorithm 1 with cost*

$$\mathcal{O}(\mathsf{M}(m + an_X)\log(m + an_X)) \subset \tilde{\mathcal{O}}(m + an_X) \ .$$

Proof. For any $f \in \mathfrak{L}(m)$ it holds by definition that $ad_X + bd_Y \leqslant m$ and $d_Y < a$, where $d_X = \deg_X f$ and $d_Y = \deg_Y f$. Consequently $d_X \leqslant m/a$, and since $\nu_Y \leqslant a$, then according to Lemma 2.12 the cost of encoding becomes

$$\begin{aligned}&\mathcal{O}(a\mathsf{M}(\frac{m}{a} + n_X)\log(\frac{m}{a} + n_X) + n_X\mathsf{M}(a + \nu_Y)\log(a + \nu_Y))\\ &\subset \mathcal{O}(\mathsf{M}(m + an_X)\log(m + an_X)) \ .\end{aligned}$$

□

Note that the encoding cost, as given by Theorem 2.13, depends on the layout of the evaluation points as well as the curve parameter a. In the following corollary we bound the worst case complexity for curves satisfying some very mild assumptions. A more refined analysis for special families of $C_{a,b}$ curves will be given in Section 2.6.

Corollary 2.14 (of Theorem 2.13)**.** *If $\mathbb{F} = \mathbb{F}_q$, then any one-point code of length $n \geqslant q$ over a $C_{a,b}$ curve with genus $g \leqslant n$ can be encoded using at most*

$$\mathcal{O}(\mathsf{M}(q\sqrt{n})\log(q\sqrt{n})) \subset \tilde{\mathcal{O}}(q\sqrt{n}) \subset \tilde{\mathcal{O}}(n^{3/2})$$

operations in $\mathbb{F}$.

Proof. Assuming w.l.o.g. that $a < b$ we get that

$$n \geqslant g = \frac{1}{2}(a-1)(b-1) \geqslant \frac{1}{2}(a-1)^2 \ ,$$

hence $a \leqslant \sqrt{2n} + 1 \in \mathcal{O}(\sqrt{n})$. Since $m \leqslant n + 2g - 1 \in \mathcal{O}(n)$ (otherwise encoding is not injective) and $n_X \leqslant q$, then the claim follows from Theorem 2.13. □

2.5 Fast unencoding using interpolation

In this section we address Problem 2.9, i.e. unencoding, by first considering a slightly more general interpolation problem where we ignore the fact that the underlying points lie on a $C_{a,b}$ curve. Generally, our approach will require two separate stages:

1) interpolation with relaxed monomial support and

2) monomial support reduction.

Stage 2 dominates the cost, however, as we will see in Section 2.6.1, this step is unnecessary for some codes whose evaluation points form a semi-grid. We now proceed by examining the two stages in detail, starting with the former.

2.5.1 Interpolation with relaxed monomial support

Instead of immediately seeking a polynomial that satisfies the monomial support constraints of problem Problem 2.9, we first look for one with X-degree less than n_X and Y-degree less than ν_Y, since this allows us to obtain it from a closed form expression:

Lemma 2.15. *Given a point set $\mathcal{P} \subset \mathbb{F}^2$ and a tuple of interpolation values $\boldsymbol{c} = (c_{\alpha,\beta})_{(\alpha,\beta)\in\mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$, the polynomial*

$$R_{\mathcal{P}}^{(\boldsymbol{c})} = \sum_{\alpha\in\mathcal{X}} \prod_{\alpha'\in\mathcal{X}\setminus\{\alpha\}} \frac{X-\alpha'}{\alpha-\alpha'} \sum_{\beta\in\mathcal{Y}_\alpha} c_{\alpha,\beta} \prod_{\beta'\in\mathcal{Y}_\alpha\setminus\{\beta\}} \frac{Y-\beta'}{\beta-\beta'} \in \mathbb{F}[X,Y] \ ,$$

satisfies $R_{\mathcal{P}}^{(\boldsymbol{c})}(\alpha,\beta) = c_{\alpha,\beta}$ for every $(\alpha,\beta) \in \mathcal{P}$.

Proof. Writing

$$R_{\mathcal{P}}^{(\boldsymbol{c})} = \sum_{\alpha\in\mathcal{X}} A_\alpha \sum_{\beta\in\mathcal{Y}_\alpha} c_{\alpha,\beta} B_{\alpha,\beta} \ ,$$

where

$$A_\alpha = \prod_{\alpha'\in\mathcal{X}\setminus\{\alpha\}} \frac{X-\alpha'}{\alpha-\alpha'} \in \mathbb{F}[X] \quad \text{and} \quad B_{\alpha,\beta} = \prod_{\beta'\in\mathcal{Y}_\alpha\setminus\{\beta\}} \frac{Y-\beta'}{\beta-\beta'} \in \mathbb{F}[Y] \ ,$$

it is easy to see that for any $(\widehat{\alpha},\widehat{\beta}) \in \mathcal{P}$

$$A_\alpha(\widehat{\alpha}) = \begin{cases} 1 & \text{if } \widehat{\alpha} = \alpha \\ 0 & \text{if } \widehat{\alpha} \neq \alpha \end{cases} \quad \text{and} \quad B_{\alpha,\beta}(\widehat{\beta}) = \begin{cases} 1 & \text{if } \widehat{\beta} = \beta \\ 0 & \text{if } \widehat{\beta} \neq \beta \end{cases} \ .$$

It follows that

$$R_{\mathcal{P}}^{(\boldsymbol{c})}(\widehat{\alpha},\widehat{\beta}) = \sum_{\alpha\in\mathcal{X}} A_\alpha(\widehat{\alpha}) \sum_{\beta\in\mathcal{Y}_\alpha} c_{\alpha,\beta} B_{\alpha,\beta}(\widehat{\beta}) = A_{\widehat{\alpha}}(\widehat{\alpha}) c_{\widehat{\alpha},\widehat{\beta}} B_{\widehat{\alpha},\widehat{\beta}}(\widehat{\beta}) = c_{\widehat{\alpha},\widehat{\beta}} \ .$$

□

Our strategy for efficiently computing $R_{\mathcal{P}}^{(\boldsymbol{c})}$ from Lemma 2.15 is as follows: We first compute the polynomials $h^{(\alpha,Y)} := R_{\mathcal{P}}^{(\boldsymbol{c})}(\alpha, Y) \in \mathbb{F}[Y]$ for $\alpha \in \mathcal{X}$ using univariate interpolation. We then reinterpret the interpolation problem as being in $\mathbb{F}[Y][X]$, i.e. we seek a univariate polynomial $h(X) = R_{\mathcal{P}}^{(\boldsymbol{c})}$ with coefficients belonging to the ring $\mathbb{F}[Y]$ such that $h(\alpha) = h^{(\alpha,Y)}$ for each $\alpha \in \mathcal{X}$. For clarity and for a slight improvement in complexity (on the level of logarithms) we make the latter interpolation explicit.

Our approach will make use of *binary trees*, so us establish some notation and terminology: We model a tree $\mathcal{T}$ as a finite, non-empty set of *vertices* together with a map $\mathcal{T}^{[\cdot]}$ which sends any vertex $v \in \mathcal{T}$ to a subset $\mathcal{T}^{[v]} \subset \mathcal{T}$ containing its *children*. Clearly $\mathcal{T}^{[v]} \cap \mathcal{T}^{[v']} = \varnothing$ for any two distinct $v, v' \in \mathcal{T}$. The *root* of $\mathcal{T}$ is the unique vertex $r \in \mathcal{T}$ such that $r \notin \mathcal{T}^{[v]}$ for any $v \in \mathcal{T}$. If a vertex $v \in \mathcal{T}$ has no children, i.e. $T^{[v]} = \varnothing$, then v is a *leaf*, otherwise v is a *non-leaf*. Finally, a tree $\mathcal{T}$ is *binary* if $|\mathcal{T}^{[v]}| \leqslant 2$ for all $v \in \mathcal{T}$.

Following in the footsteps of [vzGG12, Section 10.1], we define subproduct trees:

Definition 2.16. A *partition tree* of any finite subset $S \subseteq \mathbb{F}$ is a binary tree $\mathcal{T}$ whose vertices are subsets of S such that:

- $S \in \mathcal{T}$ is the root,
- any leaf $L \in \mathcal{T}$ has $|L| = 1$, and
- any non-leaf $V \in \mathcal{T}$ is the disjoint union of its children, and
- if $V_1, V_2 \in T^{[V]}$ for some $V \in \mathcal{T}$, then $||V_1| - |V_2|| \leqslant 1$.

The corresponding *subproduct tree* $\mathcal{U} \subset \mathbb{F}[X]$ is a binary tree obtained by replacing each vertex $V \in \mathcal{T}$ with $\varphi(V) := \prod_{\alpha \in V}(X - \alpha) \in \mathbb{F}[X]$, i.e. φ is a bijection $\mathcal{T} \to \mathcal{U}$ such that $\mathcal{U}^{[\varphi(V)]} = \{\varphi(C) \mid C \in \mathcal{T}^{[V]}\}$ for any $V \in \mathcal{T}$.

Although computing a partition tree for a subset $S \subseteq \mathbb{F}$ technically does not cost any operations in $\mathbb{F}$, simply storing it requires us to hold $\mathcal{O}(|S| \log |S|)$ field elements in memory, which also bounds the computation time in practice. The following lemma enables us to compute the corresponding subproduct tree of S in roughly the same time (up to logarithmic factors).

Proposition 2.17 (Lemma 10.4 from [vzGG12])**.** *There exists an an algorithm which for any subset $S \subseteq \mathbb{F}$ and any partition tree T of $\mathcal{S}$ computes the corresponding product tree of $\mathcal{S}$ using at most $\mathcal{O}(\mathsf{M}(|\mathcal{S}|) \log(|\mathcal{S}|)) \subset \widetilde{\mathcal{O}}(|\mathcal{S}|)$ operations in $\mathbb{F}$.*

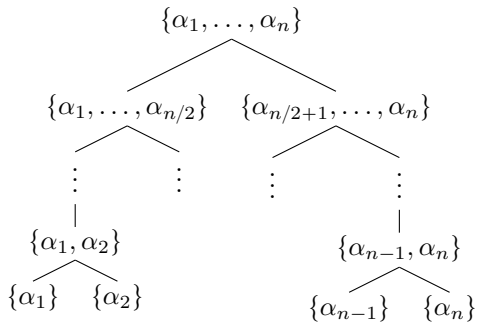

Figure 2.3: A partition tree for $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$, where $n = 2^k$.

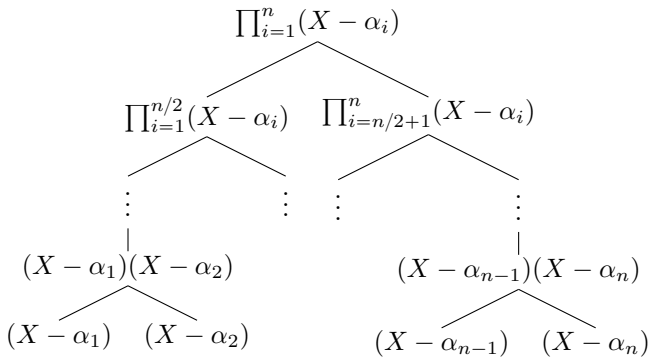

Figure 2.4: A subproduct tree for the set $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$, where $n = 2^k$.

Algorithm 2 Combine($\boldsymbol{h}$,$\mathcal{U}$)

Input:

- A subproduct tree $\mathcal{U} \subset \mathbb{F}[X]$ of some $S \subseteq \mathbb{F}$ with $|S| = 2^k$,
- polynomials $\boldsymbol{h} = (h^{(\alpha,Y)})_{\alpha \in S} \in \mathbb{F}[Y]^S$.

Output:

- the polynomial $h = \sum_{\alpha \in S} h^{(\alpha,Y)} \prod_{\alpha' \in S \setminus \{\alpha\}} (X - \alpha') \in \mathbb{F}[X, Y]$.

```
if S = {α} for some α ∈ F then
    return h_α
else
    {u_1, u_2} ← U^[u], where u ∈ U is the root
    for t = 1, 2 do
        h_t ∈ F[X, Y] ← Combine((h^(α,Y))_{α∈S_t}, U_t), where
        U_t ⊂ U are the subtrees with roots S_t and u_t respectively
    return h ∈ F[X, Y] ← u_2 h_1 + u_1 h_2
```

Lemma 2.18. *Algorithm 2 is correct, and if* $\deg h_\alpha < r$ *for all* $\alpha \in S$*, then it costs* $\mathcal{O}(r\mathsf{M}(s)\log(s)) \subset \widetilde{\mathcal{O}}(rs)$ *operations in* $\mathbb{F}$*, where* $s = |S| = 2^k$*.*

Proof. We prove correctness by induction on k. The base case $k = 0$ is clear, since then $S = \{\alpha\}$ for some $\alpha \in \mathbb{F}$ and the algorithm returns $h = h_\alpha$. For $k > 0$ we know that S is the disjoint union of S_1 and S_2, where $|S_1| = |S_2| = 2^{k-1}$. By induction hypothesis then

$$h_t = \sum_{\alpha \in S_t} h^{(\alpha,Y)} \prod_{\alpha' \in S_t \setminus \{\alpha\}} (X - \alpha') \quad \text{for } t = 1, 2 \ .$$

It follows that the output of the algorithm is

$$\begin{aligned} h &= \sum_{\alpha \in S_1} h^{(\alpha,Y)} \prod_{\alpha' \in S \setminus \{\alpha\}} (X - \alpha') + \sum_{\alpha \in S_2} h^{(\alpha,Y)} \prod_{\alpha' \in S \setminus \{\alpha\}} (X - \alpha') \\ &= \sum_{\alpha \in S} h^{(\alpha,Y)} \prod_{\alpha' \in S \setminus \{\alpha\}} (X - \alpha') \ . \end{aligned}$$

For the complexity, let $C(s)$ denote the cost of the algorithm when $|S| = s$. In each recursive step (Step 6), the algorithm solves two subproblems of half the size of the original problem, each having the cost $C(s/2)$. The computational bottleneck in combining these two solutions in Step 7 comes from carrying out the multiplications $u_2 \cdot h_1$ and $u_1 \cdot h_2$, where $\deg_X h_1, \deg_X h_2 < s/2$ and $\deg_Y h_1, \deg_Y h_2 < r$, and where both of $u_1, u_2 \in \mathbb{F}[X]$ have degree $s/2$. Each of these multiplications costs $\mathcal{O}(r\mathsf{M}(s))$ operations in $\mathbb{F}$, yielding the recurrence relation $C(s) = 2C(s/2) + \mathcal{O}(r\mathsf{M}(s))$, which has the solution $\widetilde{\mathcal{O}}(r\mathsf{M}(s)\log(s)) + \mathcal{O}(s)C(1)$. At the base case, the algorithm simply returns a univariate polynomial in Y with degree less than r, thus $C(1) \in \mathcal{O}(r)$. □

Algorithm 3 BivariateInterpolate$(\mathcal{P}, \boldsymbol{c})$

Input:

- A point set $\mathcal{P} \subseteq \mathbb{F}^2$
- a tuple of interpolation values $\boldsymbol{c} = (c_{\alpha,\beta})_{(\alpha,\beta)\in\mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$.

Output:

- the interpolating polynomial $h = R_{\mathcal{P}}^{(\boldsymbol{c})} \in \mathbb{F}[X, Y]$ from Lemma 2.15.

1: $\mathcal{U} \subset \mathbb{F}[X] \leftarrow$ a product tree of $\mathcal{X}$
2: $u \in \mathbb{F}[X] \leftarrow$ the formal derivative of the root of $\mathcal{U}$
3: $(u^{(\alpha)})_{\alpha\in\mathcal{X}} \in \mathbb{F}^{\mathcal{X}} \leftarrow (u(\alpha))_{\alpha\in\mathcal{X}}$ ▷ using fast univariate MPE
4: **for** each $\alpha \in \mathcal{X}$ **do**
5: $\quad h^{(\alpha,Y)} \in \mathbb{F}[Y] \leftarrow$ polynomial such that $h^{(\alpha,Y)}(\beta) = c_{\alpha,\beta}$ for each $\beta \in \mathcal{Y}_\alpha$ ▷ using fast univariate interpolation
6: **return** $h \in \mathbb{F}[X, Y] \leftarrow$ Combine$((h^{(\alpha,Y)}/u^{(\alpha)})_{\alpha\in\mathcal{X}}, \mathcal{U})$ ▷ Algorithm 2

Lemma 2.19. *Algorithm 3 is correct and costs*

$$\mathcal{O}(\nu_Y \mathsf{M}(n_X)\log(n_X) + n_X \mathsf{M}(\nu_Y)\log(\nu_Y)) \subset \widetilde{\mathcal{O}}(n_X \nu_Y) \ ,$$

operations in $\mathbb{F}$.

Proof. Correctness follows from Lemma 2.18, which guarantees that

$$h = \sum_{\alpha \in \mathcal{X}} \frac{h^{(\alpha,Y)}}{u^{(\alpha)}} \prod_{\alpha' \in \mathcal{X} \setminus \{\alpha\}} (X - \alpha') \ ,$$

where

$$h^{(\alpha,Y)} = \sum_{\beta \in \mathcal{Y}_\alpha} c_{\alpha,\beta} \prod_{\beta' \in \mathcal{Y}_\alpha \setminus \{\alpha\}} \frac{Y - \beta'}{\beta - \beta'}$$

(by univariate interpolation) and $u^{(\alpha)} = u(\alpha) = \prod_{\alpha' \in \mathcal{X} \setminus \{\alpha\}} (\alpha - \alpha')$, since

$$u = \sum_{\alpha \in \mathcal{X}} \prod_{\alpha' \in \mathcal{X} \setminus \{\alpha\}} (X - \alpha) \ ,$$

hence $h = R_{\mathcal{P}}^{(\boldsymbol{c})}$. It is not hard to see that $u^{(\alpha)} \neq 0$ for $\alpha \in \mathcal{X}$.

Step 1 costs $\mathcal{O}(\mathsf{M}(n_X)\log(n_X))$ according to Proposition 2.17, Step 2 costs $\mathcal{O}(n_X)$ and Step 4 costs $\mathcal{O}(\mathsf{M}(n_X)\log(n_X))$. The total cost of Step 5 over all values of $\alpha \in \mathcal{X}$ is $\mathcal{O}(n_X \mathsf{M}(\nu_Y)\log(\nu_Y))$. The cost of the algorithm as a whole follows from the fact that Step 6 costs $\mathcal{O}(\nu_Y \mathsf{M}(n_X)\log(n_X))$ due to Lemma 2.18. □

Notice that if $\mathcal{P}$ is close to being a semi-grid, i.e. $n_X \nu_Y \approx |\mathcal{P}|$, then Algorithm 3 has quasi-linear complexity. Unfortunately, this is not enough to guarantee quasi-linear unencoding as the returned polynomial $R_{\mathcal{P}}^{(\boldsymbol{c})}$ might fail to satisfy the monomial support constraints imposed by Problem 2.9, requiring us to perform additional computations. In Section 2.5.2 we explain how to efficiently reduce the support of $R_{\mathcal{P}}^{(\boldsymbol{c})}$, though the cost of doing this will generally be worse than quasi-linear. Fortunately, as we will see in Section 2.6.1, if $\mathcal{P}$ is a semi-grid satisfying certain size constraints, then this additional computational step can be omitted. This is partly due to the fact that if $\mathcal{P}$ is a semi-grid, then the $\mathbb{F}$-vector spaces $\mathcal{R} = \{h \in \mathbb{F}[X,Y] \mid \deg_X h < n_X \text{ and } \deg_Y h < \nu_Y\}$ and $\mathbb{F}^{\mathcal{P}}$ are isomorphic under the evaluation map on $\mathcal{P}$, implying that $R_{\mathcal{P}}^{(\boldsymbol{c})}$ is the *unique* polynomial in $\mathcal{R}$ that interpolates $\boldsymbol{c} \in \mathbb{F}^{\mathcal{P}}$. This is in contrast to the case when $\mathcal{P}$ in not a semi-grid, as we will discuss in the following section.

2.5.2 Reducing the monomial support

In Section 2.5.1 we have seen that for any finite point set $\mathcal{P} \subseteq \mathbb{F}^2$ and any interpolation values $\boldsymbol{c} = (c_P)_{P \in \mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$ we can compute a polynomial $R_{\mathcal{P}}^{(\boldsymbol{c})} \in \mathbb{F}[X,Y]$ with $\deg_X R_{\mathcal{P}}^{(\boldsymbol{c})} < n_X$ and $\deg_Y R_{\mathcal{P}}^{(\boldsymbol{c})} < \nu_Y$ satisfying $R_{\mathcal{P}}^{(\boldsymbol{c})}(P) = c_P$ for $P \in \mathcal{P}$, see Lemma 2.15. Now we turn our attention to transforming $R_{\mathcal{P}}^{(\boldsymbol{c})}$ into a polynomial $f \in \mathbb{F}[X,Y]$ which satisfies the constraints imposed by Problem 2.9 while preserving the evaluations on $\mathcal{P}$. More precisely, we now wish to solve the following problem:

Problem 2.20. Given the polynomial $R_{\mathcal{P}}^{(\boldsymbol{c})} \in \mathbb{F}[X,Y]$ from Lemma 2.15 find the unique polynomial $f \in \mathfrak{L}(m)$, i.e. $\deg_Y f < a$ and $\delta(f) \leqslant m$, such that $f(P) = R_{\mathcal{P}}^{(\boldsymbol{c})}(P)$ for $P \in \mathcal{P}$.

We will solve Problem 2.20 using *Gröbner bases* (see e.g. [CLO07]), so let us establish some notation: For the remainder of this section we impose the (total) monomial order $\preceq_{a,b}$ on $\mathbb{F}[X,Y]$, where

$$X^iY^j \preceq_{a,b} X^{j'}Y^{j'} \iff \begin{array}{ll} ai+bj < ai'+bj' & \text{or} \\ ai+bj = ai'+bj' & \text{and} \quad j < j' \end{array} .$$

For any polynomial $h = \sum_{(i,j)\in\mathbb{Z}^2_{\geqslant 0}} h_{i,j}X^iY^j \in \mathbb{F}[X,Y]$ with $h_{i,j} \in \mathbb{F}$ let $\operatorname{lm} h$ denote the *leading monomial* of h (with respect to $\preceq_{a,b}$), and for any monomial X^iY^j let $h^{[X^iY^j]} = h_{i,j}X^iY^j$. With this notation $h^{[\operatorname{lm} h]}$ denotes the *leading term* of h, i.e. if $\operatorname{lm} h = X^iY^j$, then $h^{[\operatorname{lm} h]} = h_{i,j}X^iY^j$.

If $\boldsymbol{g} = (g_1, \ldots, g_\gamma) \in \mathbb{F}[X,Y]^\gamma$ is a Gröbner basis of some ideal such that for $t = 1, \ldots, \gamma$ it holds that $\operatorname{lm} g_t \nmid w$ for all monomials $w \in \operatorname{supp} h$, then we say that h is *reduced* with respect to $\boldsymbol{g}$.

Lemma 2.21. *Let $f \in \mathfrak{L}(m)$ with $m < n$, and let $\boldsymbol{g} = (g_1, \ldots, g_\gamma) \in \mathbb{F}[X,Y]^\gamma$ be a Gröbner basis of the vanishing ideal $\Gamma(\mathcal{P})$ of $\mathcal{P}$. If $h \in \mathbb{F}[X,Y]$ is reduced with respect to $\boldsymbol{g}$ and satisfies $h(P) = f(P)$ for $P \in \mathcal{P}$, then $h = f$.*

Proof. If $h \neq f$, then $h \notin \mathfrak{L}(m)$, since Lemma 1.17 on page 14 guarantees that the evaluation map $r \in \mathfrak{L}(m) \mapsto (r(P))_P \in \mathbb{F}^{\mathcal{P}}$ is injective. Consequently, there exists at least one monomial $w \in \operatorname{supp} h \cap \operatorname{supp}(h-f)$, but since $0 \neq h-f \in \Gamma(\mathcal{P})$, then $h - f = \sum_{t=1}^{\gamma} p_t g_t$ with $p_t \in \mathbb{F}[X,Y]$ not all zero, which implies that $\operatorname{lm} g_u \mid w$ for some $u \in \{1, \ldots, \gamma\}$, contradicting that h is reduced with respect to $\boldsymbol{g}$. $\square$

An *extended reduction* of h with respect to $\boldsymbol{g}$ is a tuple $(p_1, \ldots, p_\gamma, f) \in \mathbb{F}[X, Y]^{\gamma+1}$ satisfying $h = f + \sum_{t=1}^{\gamma} p_t g_t$ such that f is reduced with respect to $\boldsymbol{g}$. The naive approach for computing extended reductions is presented in the listing of Algorithm 4; although it is inefficient, its inner workings are helpful for reasoning about its output, which can also be obtained using faster algorithms.

Algorithm 4 ExtendedReduce$(f, \boldsymbol{g})$

Input:

- A polynomial $h \in \mathbb{F}[X, Y]$,
- a reduced Gröbner basis $\boldsymbol{g} = (g_1, \ldots, g_\gamma) \in \mathbb{F}[X, Y]^\gamma$.

Output:

- an extended reduction $(p_1, \ldots, p_\gamma, f) \in \mathbb{F}[X, Y]^{\gamma+1}$ of h with respect to $\boldsymbol{g}$.

1: $f \in \mathbb{F}[X, Y] \leftarrow h$
2: $\boldsymbol{p} = (p_1, \ldots, p_\gamma) \in \mathbb{F}[X, Y]^\gamma \leftarrow (0, \ldots, 0)$
3: **while** f is not reduced with respect to $\boldsymbol{g}$ **do**
4: $\quad t \in \{1, \ldots, \gamma\} \leftarrow$ minimal index such that $\operatorname{lm} g_t \mid w$ for some $w \in \operatorname{supp} f$
5: $\quad r \in \mathbb{F}[X, Y] \leftarrow$ maximal monomial in $\operatorname{supp} f$ such that $\operatorname{lm} g_t \mid r$
6: $\quad w \in \mathbb{F}[X, Y] \leftarrow f^{[r]} / g_t^{[\operatorname{lm} g_t]}$
7: $\quad p_t \leftarrow p_t + w$
8: $\quad f \leftarrow f - w g_t$
9: **return** $(p_1, \ldots, p_\gamma, f) \in \mathbb{F}[X, Y]^{\gamma+1}$

Remark 2.22. Although an extended reduction is typically not unique, the output of Algorithm 4 is uniquely determined due to the minimal choice of t in Step 4.

Remark 2.23. Actually, there is no reason for Algorithm 4 to require $\boldsymbol{g}$ to be a reduced Gröbner basis; we do so merely for exposition purposes. As detailed in [vdH15], it suffices for $\boldsymbol{g}$ to be *autoreduced*, which is a weaker constraint.

Proposition 2.24 (Theorem 4 in [vdH15]). *There exists an algorithm, which we name* FastExtendedReduce*, that computes the output of Algorithm 4 using at most*

$$\mathcal{O}\Big(\sum_{t=0}^{\gamma} \mathsf{M}(\eta_X^{(t)} \eta_Y^{(t)}) \log(\eta_X^{(t)} \eta_Y^{(t)})^2\Big) \subset \tilde{\mathcal{O}}\Big(\sum_{t=0}^{\gamma} \eta_X^{(t)} \eta_Y^{(t)}\Big)$$

operations in $\mathbb{F}$, where for $W \in \{X, Y\}$

$$\eta_W^{(t)} = \begin{cases} \deg_W h & \text{for } t = 0 \\ \deg_W p_t + \deg_W g_t & \text{for } t = 1, \ldots, \gamma \end{cases}.$$

Lemma 2.25. *Let $\boldsymbol{g} = (g_1, \dots, g_\gamma) \in \mathbb{F}[X, Y]$ be a reduced Gröbner basis with*

$$\operatorname{lm} g_1 = X^u \text{ and } \operatorname{lm} g_2 = Y^v, \text{ where } g_1 \in \mathbb{F}[X] \ .$$

If $(p_1, \dots, p_\gamma, f) \in \mathbb{F}[X, Y]^{\gamma+1}$ is the output of Algorithm 4 when computing an extended reduction of some $h \in \mathbb{F}[X, Y]$ with respect to $\boldsymbol{g}$, then

$$\begin{aligned} \deg_X p_t + \deg_X g_t &\leqslant \max\{\deg_X h, 2(u-1)\} \text{ and} \\ \deg_Y p_t + \deg_Y g_t &\leqslant \max\{\deg_Y h, 2(v-1)\} \ . \end{aligned}$$

Proof. Let $f_j, w_j \in \mathbb{F}[X, Y]$ and $t_j \in \{1, \dots, \gamma\}$ denote the values of f, w and t, respectively, after the j-th iteration of the loop, setting $f_0 = h$. It is clear that

$$\begin{aligned} \deg_X f_j &\leqslant \max\{\deg_X f_{j-1}, \deg_X(w_j g_{t_j})\} \text{ and} \\ \deg_Y f_j &\leqslant \max\{\deg_Y f_{j-1}, \deg_Y(w_j g_{t_j})\} \ . \end{aligned}$$

With the aim of bounding the degrees of $w_j g_{t_j}$ we consider three cases for the value of t_j:

- If $t_j = 1$, i.e. $\deg_X f_{j-1} \geqslant u$, then
$$\deg_X(w_j g_{t_j}) = \deg_X f_{j-1} \quad \text{and} \quad \deg_Y(w_j g_{t_j}) \leqslant \deg_Y f_{j-1} \ ,$$
since $\deg_Y g_{t_j} = 0$;
- if $t_j = 2$, i.e. $\deg_X f_{j-1} < u$ and $\deg_Y f_{j-1} \geqslant v$, then
$$\deg_Y(w_j g_{t_j}) = \deg_Y f_{j-1} \quad \text{and} \quad \deg_X(w_k g_{t_j}) \leqslant 2(u-1) \ ,$$
since $\deg_X w_k \leqslant \deg_X f_k < u$ and $\deg_X g_{t_j} < u$ because $\boldsymbol{g}$ is reduced;
- if $t_j > 2$, i.e. $\deg_X f_k < u$ and $\deg_Y f_k < v$, then
$$\deg_X(w_j g_{t_j}) \leqslant 2(u-1) \quad \text{and} \quad \deg_Y(w_j g_{t_j}) \leqslant 2(v-1) \ ,$$
since $\deg_X w_j, \deg_Y g_{t_j} < u$ and $\deg_Y w_j, \deg_Y g_{t_j} < v$.

By induction it follows that for every j

$$\begin{aligned} \deg_X(w_j g_{t_j}) &\leqslant \max\{\deg_X h, 2(u-1)\} \text{ and} \\ \deg_Y(w_j g_{t_j}) &\leqslant \max\{\deg_Y h, 2(v-1)\} \ . \end{aligned}$$

The conclusion follows, since for $W \in \{X, Y\}$ and $t = 1, \dots, \gamma$ it holds that

$$\deg_W p_t + \deg_W g_t = \deg_W(p_t g_t) = \max_{j \in J_t} \deg_W(w_j g_{t_j}) \ ,$$

where $J_t = \{j \mid t_j = t\}$. □

Lemma 2.26. *If $\boldsymbol{g} = (g_1, \dots, g_\gamma) \in \mathbb{F}[X, Y]^\gamma$ is a reduced Gröbner basis, where for some $j \in \{1, \dots, \gamma\}$ it holds that $\operatorname{lm} g_j = Y^v$ with $v \in \mathbb{Z}_{\geqslant 0}$, then $\gamma \leqslant v + 1$.*

Proof. Since $\boldsymbol{g}$ is reduced, then we can relabel its entries such that

$$v = \deg_Y \operatorname{lm} g_1 > \cdots > \deg_Y \operatorname{lm} g_\gamma \geqslant 0 \ ,$$

from which follows the sought conclusion. □

Lemma 2.27. *If $\mathcal{P} \subset \mathbb{F}^2$ is a subset of the rational points on a $C_{a,b}$ curve and $\boldsymbol{g} = (g_1, \dots, g_\gamma) \in \mathbb{F}[X, Y]^\gamma$ is the reduced Gröbner basis of the vanishing ideal $\Gamma(\mathcal{P})$ of $\mathcal{P}$, then* $\mathsf{FastExtendedReduce}(R_{\mathcal{P}}^{(\boldsymbol{c})}, \boldsymbol{g})$ *from Lemma 2.15 can be computed using at most*

$$\mathcal{O}(a\mathsf{M}(an_X)\log(an_X)^2) \subset \widetilde{\mathcal{O}}(a^2 n_X)$$

operations in $\mathbb{F}$.

Proof. It is clear that if $G = \prod_{\alpha \in \mathcal{X}}(X - \alpha) \in \mathbb{F}[X]$, then $G \in \Gamma(\mathcal{P})$ with $\operatorname{lm} G = X^{n_X}$, hence we can choose g_1 such that $g_1 \in \mathbb{F}[X]$ and $\operatorname{lm} g_1 \mid X^{n_X}$ (by simply permuting the entries of $\boldsymbol{g}$). Similarly, if $Q \in \mathbb{F}[X, Y]$ is the defining polynomial of the $C_{a,b}$ curve, then $Q \in \Gamma(\mathcal{P})$ with $\operatorname{lm} Q = Y^a$, so we may pick g_2 such that $\operatorname{lm} g_2 \mid Y^a$. By Lemma 2.26, this immediately implies that $\gamma \leqslant a + 1$. If $(p_1, \dots, p_\gamma, f) \in \mathbb{F}[X, Y]^\gamma$ is the output of Algorithm 4 when computing an extended reduction of $R_{\mathcal{P}}^{(\boldsymbol{c})}$ with respect to $\boldsymbol{g}$, then Lemma 2.25 guarantees that for $t = 1, \dots, \gamma$

$$\begin{aligned} &\deg_X p_t + \deg_X g_t \leqslant 2(n_X - 1) \in \mathcal{O}(n_X) \text{ and} \\ &\deg_Y p_t + \deg_Y g_t \leqslant 2(a - 1) \in \mathcal{O}(a) \ . \end{aligned}$$

Finally, the stated complexity bound then follows from Proposition 2.24, since $\deg_X R_{\mathcal{P}}^{(\boldsymbol{c})} \leqslant n_X$ and $\deg_Y R_{\mathcal{P}}^{(\boldsymbol{c})} < a$. □

2.5.3 A fast unencoding algorithm

Algorithm 5 $\mathsf{Unencode}(\mathcal{P}, \boldsymbol{c}, \boldsymbol{g})$

Input:

- A subset $\mathcal{P} = \{P_1, \ldots, P_n\} \subseteq \mathbb{F}^2$ of the rational points on the underlying $C_{a,b}$ curve,
- a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}(m, \mathcal{P})$,
- precomputed reduced Gröbner basis $\boldsymbol{g} \in \mathbb{F}[X, Y]^\gamma$ of the vanishing ideal $\Gamma(\mathcal{P})$ of $\mathcal{P}$.

Output:

- The unique message $f \in \mathfrak{L}(m)$ such that $f(P_i) = c_i$ for $i = 1, \ldots, n$.

1: $h \in \mathbb{F}[X, Y] \leftarrow \mathsf{BivariateInterpolate}(\mathcal{P}, \boldsymbol{c})$ $\triangleright$ $h = R_{\mathcal{P}}^{(\boldsymbol{c})}$, Algorithm 3
2: $(p_1, \ldots, p_\gamma, f) \in \mathbb{F}[X, Y]^{\gamma+1} \leftarrow \mathsf{FastExtendedReduce}(h, \boldsymbol{g})$ $\triangleright$ Proposition 2.24
3: **return** f

Theorem 2.28. *Algorithm 5 is correct. Its cost is bounded by*

$$\mathcal{O}(a\mathsf{M}(an_X)\log(an_X)^2) \subset \widetilde{\mathcal{O}}(a^2 n_X)$$

operations in $\mathbb{F}$, assuming that we have precomputed the reduced Gröbner basis $\boldsymbol{g}$ of the vanishing ideal $\Gamma(\mathcal{P})$ of the points $\mathcal{P}$.

Proof. Correctness is clear, since $f(P_i) = h(P_i) = c_i$ for $i = 1, \ldots, n$, where the latter equality is due to Lemma 2.19 and the former follows from the fact that $g_j \in \Gamma(\mathcal{P})$ for $j = 1, \ldots, \gamma$. The claim that $f \in \mathfrak{L}(m)$ follows from $\boldsymbol{g}$ being a Gröbner basis with respect to $\preceq_{a,b}$.

For the complexity, note that Lemma 2.19 implies that the cost of Step 1 is

$$\mathcal{O}(a\mathsf{M}(n_X)\log(n_X) + n_X\mathsf{M}(a)\log(a)) \ .$$

The cost of Step 2 is given by Lemma 2.27 as $\mathcal{O}(a\mathsf{M}(an_X)\log(an_X)^2)$, which dominates the total cost of the algorithm. $\square$

Similarly to what we have seen in Section 2.4 with encoding, the complexity of unencoding using Algorithm 5 depends on the layout of the evaluation points $\mathcal{P}$ as well as the curve parameter a. It should be noted, however, that the cost of unencoding incurs an additional factor of a due to the potentially expensive

monomial-support-reduction step. We will see in Section 2.6.1 that this step becomes trivial for certain well behaved codes, but before we do so let us briefly examine the worst-case complexity for our encoding algorithm under very mild assumptions:

Corollary 2.29. *(of Theorem 2.28) Let $\mathcal{P} \subset \mathbb{F}^2$ be a subset of the rational points on a $C_{a,b}$ curve defined over $\mathbb{F}_q$. If $q, g \leqslant n$, where $n = |\mathcal{P}|$, then Algorithm 5 costs at most $\widetilde{\mathcal{O}}(nq) \subseteq \widetilde{\mathcal{O}}(n^2)$ operations in $\mathbb{F}$.*

Proof. Assuming w.l.o.g. that $a < b$, we use the same upper bound

$$a \leqslant \sqrt{2n} + 1 \in \mathcal{O}(\sqrt{n})$$

as in Corollary 2.14 to obtain the cost $\widetilde{\mathcal{O}}(a^2 n_X) \subseteq \widetilde{\mathcal{O}}(nq)$. □

Observe that for codes with $n \approx q$, the complexity of our encoding algorithm is no better than multiplying the codeword by the (precomputed) inverted generator matrix. In such cases it is preferable to use structured linear system solving with cost $\widetilde{\mathcal{O}}(a^{\omega-1}n) \subseteq \widetilde{\mathcal{O}}(n^{(\omega-1)/2+1}) \subset \widetilde{\mathcal{O}}(n^{1.6865})$, where we have used the best know bound $\omega < 2.3729$ by Alman and Williams [AW21]. In practice, however, due to a more manageable constant factor, it is more common to use Strassen's matrix multiplication algorithm [Str69] with $\omega < 2.8074$, resulting in the complexity $\widetilde{\mathcal{O}}(n^{1.9037})$. More generally, ignoring constant and logarithmic factors, linear system solving will beat our approach when $a^{\omega-1}n < a^2 n_X$, or equivalently

$$a^{3-\omega} > n/n_X \ . \tag{2.1}$$

Continuing use inequalities in the asymptotic sense we observe that in the context of Corollary 2.29, i.e. with $a \leqslant \sqrt{n}$, (2.1) implies that $n^{(3-\omega)/2} > n/q$, where we have used that $n_X \leqslant q$. Consequently, (2.1) can only be satisfied when $n < q^{2/(\omega-1)}$, or $n \in \widetilde{\mathcal{O}}(q^{2/(\omega-1)})$ using asymptotic notation. With the ω-bound by Alman and Williams this translates to $n \in \mathcal{O}(q^{1.4568})$, while Strassen's algorithm gives $n \in \mathcal{O}(q^{1.1065})$. It is safe to say that such codes can be considered (asymptotically) short, at least compared to the one-point Hermitian codes where $n = q^3$.

2.6 Special curves

In this section we investigate the behavior of our encoding and unencoding algorithms for special families of $C_{a,b}$ curves. As we will see, in many interesting cases we beat the cost $\mathcal{O}(n^2)$, sometimes even achieving $\widetilde{\mathcal{O}}(n)$.

2.6.1 Semi-grids

We have mentioned that our algorithms are particularly well suited for certain point sets with a semi-grid structure. Let us now explore this in detail:

Definition 2.30. A code $\mathcal{C}_Q(\mathcal{P}, m)$ over a $C_{a,b}$ curve will be called a *maximal semi-grid code* $\mathcal{P}$ is a semi-grid with $\nu_Y(\mathcal{P}) = a$, i.e. $n_X(\mathcal{P})a = n$.

Proposition 2.31. *Any maximal semi-grid code can be encoded using Algorithm 1 with cost* $\mathcal{O}(\mathsf{M}(n)\log(n)) \in \widetilde{\mathcal{O}}(n)$.

Proof. The claim is a direct consequence of Theorem 2.13, since $an_X = n$. □

The complexity of Algorithm 5 for unencoding is dominated by the monomial-support-reduction step. Fortunately, this step can be safely omitted for maximal semi-grid codes:

Lemma 2.32. *If* $\mathcal{C}_Q(\mathcal{P}, m)$ *is a maximal semi-grid code, then*

$$\boldsymbol{g} = (G, Q) \in \mathbb{F}[X, Y]^2 \ ,$$

where $G = \prod_{\alpha \in \mathcal{X}}(X - \alpha) \in \mathbb{F}[X]$*, is a Gröbner basis of the vanishing ideal* $\Gamma(\mathcal{P})$ *of* $\mathcal{P}$ *with respect to* $\preceq_{a,b}$.

Proof. It is clear that $\boldsymbol{g}$ is a Gröbner basis of $\langle G, Q\rangle_{\mathbb{F}[X,Y]} \subseteq \Gamma(\mathcal{P})$ with respect to $\preceq_{a,b}$, so it suffices to show that $\langle G, Q\rangle_{\mathbb{F}[X,Y]} = \Gamma(\mathcal{P})$, which is necessarily true, since $\deg(\langle G, Q\rangle_{\mathbb{F}[X,Y]}) = n_X a = n = \deg(\Gamma(\mathcal{P}))$, where $\deg(\mathcal{I})$ denotes the degree of any ideal $\mathcal{I} \subseteq \mathbb{F}[X, Y]$. □

Proposition 2.33. *If* $\boldsymbol{c} = (f(P))_{P \in \mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$ *is a codeword of some maximal semi-grid code* $\mathcal{C}_Q(\mathcal{P}, m)$*, where* $f \in \mathfrak{L}(m)$ *with* $m < n$*, then* $R_{\mathcal{P}}^{(\boldsymbol{c})} = f$.

Proof. By Lemma 2.15 it holds that $\deg_X R_{\mathcal{P}}^{(\boldsymbol{c})} < n_X$ and $\deg_Y R_{\mathcal{P}}^{(\boldsymbol{c})} < \nu_Y = a$. By Lemma 2.32, if $G = \prod_{\alpha \in \mathcal{X}}(X - \alpha) \in \mathbb{F}[X]$, then $\boldsymbol{g} = (G, Q) \in \mathbb{F}[X, Y]^2$ is a Gröbner basis of the vanishing ideal $\Gamma(\mathcal{P})$ of $\mathcal{P}$, and since $\operatorname{lm} Q = Y^a$ and $\operatorname{lm} G = X^{n_X}$, then $R_{\mathcal{P}}^{(\boldsymbol{c})}$ is reduced with respect to $\boldsymbol{g}$. The sought conclusion follows from Lemma 2.21, since $R_{\mathcal{P}}^{(\boldsymbol{c})}(P) = f(P)$ for $P \in \mathcal{P}$. □

Proposition 2.34. *For any maximal semi-grid code* $\mathcal{C}_Q(\mathcal{P}, m)$*, unencoding can be solved without any precomputation using no more than*

$$\mathcal{O}(a\mathsf{M}(n_X)\log(n_X) + n_X\mathsf{M}(a)\log(a)) \subset \widetilde{\mathcal{O}}(n)$$

operations in $\mathbb{F}$.

Proof. Simply consider Algorithm 5 without Step 2, i.e. skip the extended reduction. Doing so makes precomputation of the Gröbner basis $\boldsymbol{g}$ superfluous and reduces the total complexity of the algorithm to that of Algorithm 3, given by Lemma 2.19. The conclusion follows, since $n_X a = n$. □

As we now have seen in Proposition 2.31 and Proposition 2.34, we can encode and unencode maximal semi-grid codes in quasi-linear time without any precomputation. Now is therefore a good time to take a look at some concrete families of $C_{a,b}$ curves that enable construction of such codes.

Hermitian curve

The Hermitian curve is defined over $\mathbb{F}_{q^2}$ by the polynomial

$$H = Y^q + Y - X^{q+1} \in \mathbb{F}_{q^2}[X, Y] \ .$$

It can easily be checked that H defines a $C_{a,b}$ curve with $a = q$ and $b = q + 1$. A (one-point) *The Hermitian code* is simply a code whose evaluation points are solutions to the equation $H(X, Y) = 0$.

The Hermitian curve and its function field are well-known and have been studied extensively, e.g. see [Sti09, Lemma 6.4.4]. For instance, using the norm and trace maps of the extension $\mathbb{F}_{q^2}/\mathbb{F}_q$ it can be shown that for every $\alpha \in \mathbb{F}_{q^2}$ there are exactly q distinct elements $\beta \in \mathbb{F}_{q^2}$ such that $H(\alpha, \beta) = 0$, which is to say that the set $\mathcal{P}_H$ of rational points on the Hermitian curve forms a semi-grid with $\mathcal{X}(\mathcal{P}_H) = \mathbb{F}_{q^2}$ and $\nu_Y(\mathcal{P}_H) = q = a$, amounting to $n = q^3$ points in total. Moreover, any Hermitian code $\mathcal{C}_H(\mathcal{P}_H, m)$ is a maximal-semi grid code, allowing us to encode and unencode with quasi-linear cost:

Corollary 2.35 (of Proposition 2.31 and Proposition 2.34)**.** *Any Hermitian code $\mathcal{C}_H(\mathcal{P}_H, m)$ with $m < n$ can be encoded and unencoded with cost*

$$\mathcal{O}(\mathsf{M}(n)\log(n)) \quad \textit{and} \quad \mathcal{O}(q\mathsf{M}(q^2)\log(q))$$

respectively, i.e. $\widetilde{\mathcal{O}}(n)$ in both cases.

Note that the quasi-linear cost of encoding and unencoding is also preserved for subcodes with $n_X(\mathcal{P}) < q^2$ as long as we keep $\nu_Y(\mathcal{P}) = q$.

Norm-Trace and other Hermitian-like curves

The Hermitian curve belongs to a larger family of $C_{a,b}$ curves: Let r and e be positive integers satisfying $r \geqslant 2$ and $e \mid (q^r-1)/(q-1)$, and consider the $C_{a,b}$ curve over $\mathbb{F}_{q^r}$ with defining polynomial

$$N_{r,e} = Y^{q^{r-1}} + \cdots + Y^q + Y - X^e \in \mathbb{F}_{q^r}[X,Y]$$

and with $a = q^{r-1}$ and $b = e$.

We first investigate the case $e = (q^r-1)/(q-1)$, which gives rise to the *norm-trace curves* studied in [Gei03]. It is not hard to see that for $r = 2$ this simplifies to the Hermitian curve, i.e. $N_{2,e} = H$. It can be shown that that the set $\mathcal{P}_N \subset \mathbb{F}_{q^r}^2$ of rational points on the curve defined by $N_{r,e}$ forms a semi-grid with $n_X(\mathcal{P}_N) = q^r$, $\nu_Y(\mathcal{P}_N) = a$ and consequently $n = q^{2r-1}$. Corollary 2.35 generalizes directly and shows that (one-point) *norm-trace codes* can be encoded and unencoded with quasi-linear cost:

Corollary 2.36 (of Proposition 2.31 and Proposition 2.34)**.** *Any norm-trace code $\mathcal{C}_{N_{r,e}}(\mathcal{P}_N, m)$ with $m < n = q^{2r-1}$ can be encoded and unencoded with cost*

$$\mathcal{O}(\mathsf{M}(n)\log(n)) \quad \textit{and} \quad \mathcal{O}(q^{r-1}\mathsf{M}(q^r)\log(q^r))$$

respectively, i.e. $\widetilde{\mathcal{O}}(n)$ in both cases.

If $e < (q^r-1)/(q-1)$, then the equation $N_{r,e}(\alpha,\beta) = 0$ has $q^{r-1} + e(q^r - q^{r-1})$ solutions in $\mathbb{F}_{q^r}^2$, where the small term q^{r-1} corresponds to solutions with $\beta = 0$ and $\alpha^{q^{r-1}} + \cdots + \alpha = 0$. The point set $\mathcal{P}_N^{(e)} \subset \mathbb{F}_{q^r}^2$ which containing the remaining $n = e(q^r - q^{r-1})$ points forms a semi-grid with $\nu_Y(\mathcal{P}_N^{(e)}) = a = e$ and $n_X(\mathcal{P}_N^{(e)}) = q^r - q^{r-1}$, and gives rise to a maximal semi-grid code. A special case of such curves with r even and $e \mid (q^{r/2}+1)$ was considered in [MUT09].

Corollary 2.37 (of Proposition 2.31 and Proposition 2.34)**.** *Let r, e be positive integers with $r \geqslant 2$ and e a proper divisor of $(q^r-1)/(q-1)$. If $\mathcal{P}_N^{(e)} \subset \mathbb{F}_{q^r}^2$ is the set of solutions to $N_{r,e}(\alpha,\beta) = 0$, excluding those with $\alpha^{q^r} + \cdots + \alpha = 0$ and $\beta = 0$, then the code $\mathcal{C}_{N_{r,e}}(\mathcal{P}_N^{(e)}, m)$, where $m < n = e(q^r - q^{r-1})$, can be encoded and unencoded with cost*

$$\mathcal{O}(\mathsf{M}(n)\log(n)) \quad \textit{and} \quad \mathcal{O}(e\mathsf{M}(q^r)\log(q^r))$$

respectively, i.e. $\widetilde{\mathcal{O}}(n)$ in both cases.

2.6.2 Maximal curves

We have seen in Section 2.6.1 that maximal semi-grid codes can be encoded and unencoded with quasi-linear cost. In this section we turn our attention to codes over *maximal* $C_{a,b}$ curves, i.e. those that attain the *Hasse-Weil bound*:

Theorem 2.38 (Theorem 5.2.3 in [Sti09])**.** *If N is the number of rational places of an algebraic function field F over $\mathbb{F}_q$, then $N \leqslant 2gq^{1/2} + q + 1$, where g is the genus of F.*

For $C_{a,b}$ curves we know that $g = \frac{1}{2}(a-1)(b-1)$, and that there is exactly one rational place at infinity (that we don't use for evaluation). Consequently, the length n of a code $\mathcal{C}_Q(\mathcal{P}, m)$ over a $C_{a,b}$ curve is bounded by

$$n \leqslant \mathsf{HW}(a,b) := (a-1)(b-1)q^{1/2} + q \ .$$

We proceed under the assumption that $n = \mathsf{HW}(a,b)$. A well-known example of a $C_{a,b}$ curve for which this holds is the Hermitian curve, which we have already discussed in Section 2.6.1.

Lemma 2.39. *If $\mathcal{P} \subset \mathbb{F}_q^2$ is the set of all (finite) rational points on a maximal $C_{a,b}$ curve with $a < b$, then*

$$\begin{aligned} a &< \frac{n^{1/2}}{q^{1/4}} + 1 && \in \mathcal{O}\Big(\frac{n^{1/2}}{q^{1/4}}\Big) \ , \\ qa &< n^{5/4} + n && \in \mathcal{O}(n^{5/4}) \ , \\ qa^2 &< n^{3/2} + 2n^{5/4} + 2n && \in \mathcal{O}(n^{3/2}) \ . \end{aligned}$$

Proof. Since $n = \mathsf{HW}(a,b) > (a-1)(b-1)q^{1/2} > (a-1)^2 q^{1/2}$, then

$$(a-1)^2 < \frac{n}{q^{1/2}} \iff a < \frac{n^{1/2}}{q^{1/4}} + 1 \ .$$

The second bound then follows from $q \leqslant n$, since

$$qa < q\Big(\frac{n^{1/2}}{q^{1/4}} + 1\Big) = q^{3/4}n^{1/2} + q \leqslant n^{5/4} + n \ .$$

The last bound is obtained by observing that

$$\begin{aligned} qa^2 &< q\big((a-1)^2 + 2a\big) < q\Big(\frac{n}{q^{1/2}} + 2(\frac{n^{1/2}}{q^{1/4}} + 1)\Big) \\ &= q^{1/2}n + 2q^{3/4}n^1/2 + 2q = n^{3/2} + 2n^{5/4} + 2n \ . \end{aligned}$$

□

Corollary 2.40. *Any code $\mathcal{C}_Q(\mathcal{P}, m)$ with $m < n = \mathsf{HW}(a, b)$ can be, respectively, encoded and unencoded with cost*

$$\mathcal{O}(\mathsf{M}(n^{5/4}) \log(n^{5/4})) \subset \widetilde{\mathcal{O}}(n^{5/4}) \quad \textit{and}$$
$$\mathcal{O}(\mathsf{M}(n^{3/2}) \log(n^{3/2})^2) \subset \widetilde{\mathcal{O}}(n^{3/2}) \ .$$

Proof. Combining Lemma 2.39 with Theorem 2.13 and Theorem 2.28, respectively, yields the cost for encoding and unencoding, since $n_X \leqslant q$. □

Remark 2.41. It is, strictly speaking, not necessary to require $n = \mathsf{HW}(a, b)$ in order to achieve the cost bounds given by Corollary 2.40. Indeed, the asymptotic complexity remains unchanged for any family of codes where $n \geqslant c \cdot \mathsf{HW}(a, b)$ for some (global) real number $c \in (0, 1]$. More formally, if $\mathcal{Q}$ is a subset of

$$\{(\mathcal{P}, a', b') \mid \mathcal{P} \text{ is any subset of any } C_{a,b} \text{ curve with } a = a' \text{ and } b = b'\} \ ,$$

then the cost bounds given by Corollary 2.40 are still valid over $\mathcal{Q}$ as long as there exists a $c \in (0, 1]$ such that $|\mathcal{P}| \geqslant c \cdot \mathsf{HW}(a', b')$ for every $(\mathcal{P}, a', b') \in \mathcal{Q}$.

Remark 2.42. In the context of Remark 2.41, we can even allow c to depend on the given point set instead of being global over $\mathcal{Q}$ as long as we replace the cost bounds from Corollary 2.40 for encoding and unencoding with $\widetilde{\mathcal{O}}(n^{5/4}/\sqrt{c})$ and $\widetilde{\mathcal{O}}(n^{3/2}/c)$ respectively. For example, if we are working with a family of point sets where $n^2 \geqslant \mathsf{HW}(a, b)$, i.e. $c = 1/n$, then encoding costs

$$\widetilde{\mathcal{O}}(n^{5/4}\sqrt{n}) = \widetilde{\mathcal{O}}(n^{7/4}) \ ,$$

which is still better than the naive approach.

We conclude this chapter by comparing the cost of unencoding, as given by Corollary 2.40, with the alternative approach of structured system solving described in Section 2.2.3, which has the cost $\widetilde{\mathcal{O}}(a^{\omega-1}n)$. Acknowledging that a completely fair comparison is difficult, we rely on the crude bound $n \leqslant q^2$ in combination with Lemma 2.39 to deduce that

$$a \in \mathcal{O}(n^{1/2}/q^{1/4}) \subset \mathcal{O}(n^{1/2}/n^{1/8}) = \mathcal{O}(n^{3/8}) \ .$$

The cost of structured system solving can thus be bounded by $\mathcal{O}(n^{1+3/8(\omega-1)})$. Using the best known bound $\omega < 2.3729$ by Alman and Williams [AW21], this becomes $\mathcal{O}(n^{1.5149})$, which is only marginally slower than our approach with cost $\widetilde{\mathcal{O}}(n^{1.5})$. The more practical matrix multiplication algorithm by Strassen [Str69], with $\omega < 2.8074$, yields the cost $\mathcal{O}(n^{1.6778})$. Simply replacing ω by 3 results in $\mathcal{O}(n^{1.75})$.

Chapter 3

Generic bivariate algorithms

In Chapter 2 we have seen that encoding and unencoding of certain AG codes is closely related to bivariate multi-point evaluation (MPE) and interpolation respectively. We have also seen how both problems can be solved with quasi-linear cost for point sets that conform to a special geometric structure called a semi-grid. Most point sets, however, are not semi-grids, and it is easy to construct examples for which our Pan-inspired [Pan94] algorithms will have quadratic complexity. In this chapter we investigate an alternative technique for bivariate MPE and interpolation called *reshaping*, which bears more resemblance to the ideas of Nüsken and Ziegler [NZ04] discussed in Section 2.2.2.

3.1 Strategy outline

In this section we informally describe the main idea of this chapter.

Recall Nüsken and Ziegler's strategy for evaluating a polynomial $f \in \mathbb{F}[X, Y]$ on a point set $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^{n} \subset \mathbb{F}^2$ where the α_j are pairwise distinct[1]:

[1] if they are not, simply shear the points in an extension field (see Section 2.2.2)

1) compute $h = f(X, b) \text{ rem } a$, where $b \in \mathbb{F}[X]$ with $b(\alpha_j) = \beta_j$ for $j = 1, \ldots, n$ and $a = \prod_{j=1}^{n}(X - \alpha_j) \in \mathbb{F}[X]$,
2) evaluate $h \in \mathbb{F}[X]$ on $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.

Note that the *modular composition* in Step 1 constitutes the computational bottleneck of this approach; the univariate MPE in Step 1 has quasi-linear cost. It is also worth noting that, for the sake of efficient MPE, we could have used any low degree polynomial $h \in \mathbb{F}[X] \cap (f + \Gamma(\mathcal{P}))$, where $\Gamma(\mathcal{P}) \subset \mathbb{F}[X, Y]$ denotes the vanishing ideal of $\mathcal{P}$. Modular composition is merely one way of obtaining such a polynomial, and we can view it as the computation of $h = f \text{ rem } (a, y - b)$ with respect to the monomial order $\prec_{\text{lex}}$, where

$$X^i Y^j \prec_{\text{lex}} X^{i'} Y^{j'} \iff j < j' \vee (j = j' \wedge i < i') \ .$$

A natural question then arises: Can we replace a and $y - b$ in this computation with some other polynomials $g_1, \ldots, g_\gamma \in \Gamma(\mathcal{P})$ so that the remainder

$$h = f \text{ rem } (g_1, \ldots, g_\gamma) \in \mathbb{F}[X] \cap (f + \Gamma(\mathcal{P}))$$

has low degree *and* can be obtained efficiently? Fortunately, the answer to this question turns out to be: *yes, at least most of the time.* To see this, suppose that $\deg_X f \cdot \deg_Y f \approx n$ and that there exists a polynomial $g_1 = Y^{2d/3} - \breve{g}_1 \in \Gamma(\mathcal{P})$ with $\deg_Y \breve{g}_1 < d/3$, where $d = \deg_Y f + 1$ is assumed to be divisible by 3 for the sake of simplicity. Because of its special monomial support, we say that g_1 is a *reshaper* – to see why, write $f = Y^{2d/3} \widehat{f} + \breve{f}$ with $\deg_Y \breve{f} < 2d/3$, and observe that

$$f \text{ rem } g_1 = f - \widehat{f} g_1 = \breve{f} + \widehat{f}\, \breve{g}_1 \in f + \Gamma(\mathcal{P}) \ .$$

In other words, we can use g_1 to *reshape* f into a polynomial whose Y-degree is smaller by a factor of roughly $2/3$ and whose evaluations on $\mathcal{P}$ coincide with those of f. Repeating such reduction for appropriately chosen reshapers $g_2, \ldots, g_\gamma$, each time reducing the Y-degree of the current remainder by a factor of about $2/3$, we can obtain $h \in \mathbb{F}[X] \cap (f + \Gamma(\mathcal{P}))$ using only

$$\gamma \approx \log_{3/2}(\deg_Y f) \in \mathcal{O}(\log n)$$

iterations. Therefore, as long as we can ensure that each iteration has quasi-linear cost, then the total cost of this reshaping-based MPE also becomes quasi-linear.

To achieve this, however, we need the X-degrees of all of the reshapers to be sufficiently small, and unfortunately, not every point set contains such reshapers in its vanishing ideal. For example, the requirement that $g_1 \in \Gamma(\mathcal{P})$ imposes n linear constraints on the coefficients of $\breve{g}_1$, and since not all monomials u in the

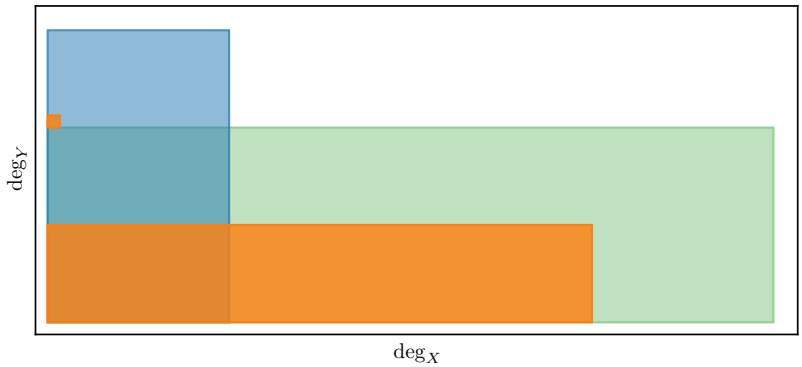

Figure 3.1: A schematic representation of one iteration of reshaping. Each colored area bounds the monomial support of a polynomial: Blue is the current remainder, orange is a reshaper and green is the subsequent remainder.

allowed monomial support are linearly independent under the evaluation map $u \mapsto (u(P))_{P\in\mathcal{P}} \in \mathbb{F}^{\mathcal{P}}$, we might therefore sometimes need to use monomials with relatively high X-degree – and this could break our quasi-linear target cost. For a *random* or *generic* point set, however, these monomials will be linearly independent with *high probability*, and since $\deg_Y \breve{g}_1 \approx d/3$, then we may expect to find g_1 with $\deg_Y \breve{g}_1 \approx 3n/d$, so that $\deg_X \breve{g}_1 \cdot \deg_Y \breve{g}_1 \approx n$. Informally, we say that a point set is *balanced* if all of the sought reshapers satisfy such degree constraints.

Example 3.1 (Unbalanced point set)**.** Suppose that we are seeking a reshaper $g = Y^{n/2} - \breve{g} \in \Gamma(\mathcal{P})$ with $\deg_Y \breve{g} \leqslant n/4$, where $\mathcal{P} = \{(\alpha_j, \alpha_j)\}_{j=1}^n$ for some pairwise distinct $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ with n divisible by 4. For efficiency reasons, we wish to have $\deg_X \breve{g} \cdot \deg_Y \breve{g} \approx n$, i.e. $\deg_X \breve{g} \approx 4$ (or some other small constant). However, noting that $\Gamma(\mathcal{P})$ is generated by $Y - X$ and $\prod_{P\in\mathcal{P}}(X - X(P))$, and letting $\mathrm{ev}_{\mathcal{P}}(u) = (u(P))_{P\in\mathcal{P}} \in \mathbb{F}_q^{\mathcal{P}}$ for any $u \in \mathbb{F}[X, Y]$, it is clear that all vectors in $\{\mathrm{ev}_{\mathcal{P}}(X^k) \mid 0 \leqslant k < n\}$ are linearly independent over $\mathbb{F}$. Consequently, the only univariate polynomial $\widetilde{g} \in \mathbb{F}[X]$ with $\deg \widetilde{g} < n$ and $\mathrm{ev}_{\mathcal{P}}(\widetilde{g}) = \mathrm{ev}_{\mathcal{P}}(Y^{n/2})$ is $\widetilde{g} = X^{n/2}$. Furthermore, since for any $i, j, k \in \{0, \ldots, n-1\}$ with $i + j < n$ it holds that $\mathrm{ev}_{\mathcal{P}}(X^iY^j) = \mathrm{ev}_{\mathcal{P}}(X^k)$ if and only if $i + j = k$, then necessarily $\operatorname{supp} \breve{g} \subset \{X^iY^j \mid j \leqslant n/4 \wedge (i + j = n/2 \vee i \geqslant n)\}$, which implies that $\deg_X \breve{g} \geqslant n/4$. But this is nowhere close to the sought $\deg_X \breve{g} \approx 4$.

Besides the requirement that the underlying point set needs to be balanced, our algorithm pays an additional price for its quasi-linear complexity: Since it is unclear how to obtain the needed reshapers within our target cost, we

have no other choice than to *precompute* them. Furthermore, the amount of precomputation needed depends on an upper bound of the Y-degree of the input polynomial. We discuss the practical implications of these two limitations below.

Precomputation and balancedness

In potential practical applications of AG codes, it is perfectly reasonable to allow the underlying point set to be available for precomputation prior to the deployment of the communication protocol – it is the computational load during the *online phase* that is most critical. Furthermore, the choice of the code naturally imposes and upper bound on the Y-degree of the message polynomial, which means that we know in advance how many reshapers we need to precompute. As outlined above, if the underlying point set is balanced, then our reshaping-based MPE is particularly well suited for this setting. Unfortunately, however, it is with respect to balancedness that we run into a potential issue: although we know that a random point set is balanced with high probability, we also know that point sets on algebraic curves are *not* random. As demonstrated by Example 3.1, some curves do indeed give rise to unbalanced points, and it is unknown how frequently this occurs. To complicate things even further, virtually all interesting AG codes contain repeating X-coordinates in their underlying point sets, which means that if reshaping-based MPE is to be used, then the points must be sheared in an extension field, and how this shearing affects balancedness is also not understood. So far, we are simply left with having to check each case individually, but since this can be done during precomputation, then we can always resort to other algorithms if the given point set turns out to be unbalanced.

Remark 3.2. As mentioned in Remark 2.10 on page 24, there is a recent bivariate MPE algorithm by van der Hoeven and Lecerf [vdHL21] which can evaluate any polynomial $f \in \mathbb{F}[X, Y]$ satisfying $\deg_X f, \deg_Y f < \sqrt{n}$ using $\mathcal{O}(\mathsf{M}(n \log n) \log(n)^3) \subset \widetilde{\mathcal{O}}(n)$ operations in $\mathbb{F}$ – not counting precomputation on the points. Their approach has similarities with ours in that it also relies on a notion genericity, however, they can ensure that it is satisfied by using a certain change of variables. It is likely that something similar can be done with our results, but this remains to be investigated. In this sense, their MPE algorithm is strictly better than ours when $\deg_X f \approx \deg_Y f$.

3.2 Reshaping

In this section we formally present the technique of reshaping in full generality. We begin by defining a single reshaper:

Definition 3.3. If $I \subseteq \mathbb{F}[X,Y]$ is an ideal and $g = Y^\eta - \breve{g} \in I$ with $\deg_Y \breve{g} < \eta$, then we say that g is an *η-reshaper* in I.

Remark 3.4. The only reason for why we have considered a general ideal I in Definition 3.3 instead of the vanishing ideal $\Gamma(\mathcal{P})$ is that this would allow us address the most general form of Nüsken and Ziegler's modular composition problem, i.e. where the modulus is not assumed to split over $\mathbb{F}$. Since this is not directly relevant for the scope of this thesis, this problem has been left out (although the generality is kept). A curious reader is referred to [NRS20].

We proceed by showing that the cost of reducing a polynomial with respect to an appropriately chosen reshaper is quasi-linear in the input size.

Lemma 3.5. *If $g = Y^\eta - \breve{g} \in I$ with $\breve{\eta} := \deg \breve{g} < \eta$ is an η-reshaper in some ideal $I \subseteq \mathbb{F}[X,Y]$, and if $f = Y^\eta \widehat{f} + \breve{f} \in \mathbb{F}[X,Y]$ with $\deg_Y f < 2\eta - \breve{\eta}$ and $\deg_Y \breve{f} < \eta$, then the polynomial $h = \breve{f} + \widehat{f}\,\breve{g} \in \mathbb{F}[X,Y]$ belongs to the coset $f + I$ and satisfies $\deg_X h \leqslant d := \deg_X f + \deg_X g$ and $\deg_Y h < \eta$. Furthermore, we can compute h using no more than $\mathcal{O}(\mathsf{M}(d\eta)) \subset \tilde{\mathcal{O}}(d\eta)$ operations in $\mathbb{F}$.*

Proof. Writing $h = \breve{f} + \widehat{f}\,\breve{g} = Y^\eta \widehat{f} + \breve{f} - \widehat{f} \cdot (Y^\eta - \breve{g}) = f - \widehat{f} g$ it is clear that $h \in f + I$. Since $\deg_Y f < 2\eta - \breve{\eta}$, then $\deg_Y(\widehat{f}\,\breve{g}) \leqslant \deg f - \eta + \breve{\eta} < \eta$, while the bound on $\deg_X h$ is trivial. The claimed cost bound is given by the cost of carrying our the product $\widehat{f} \cdot \breve{g}$ [vzGG12, Corollary 8.28]. □

As in Section 2.3, we will use the notation

$$\nu_Y(\mathcal{P}) = \max_{\alpha \in \mathbb{F}} |\mathcal{Y}_\alpha(\mathcal{P})| \ , \text{ where for any } \alpha \in \mathbb{F}$$
$$\mathcal{Y}_\alpha(\mathcal{P}) = \{Y(P) \mid P \in \mathcal{P} \text{ with } X(P) = \alpha\} \subseteq \mathbb{F} \ .$$

In order to prove existence of reshapers, we will use a folklore result which follows e.g. from [Laz85] and [Dah09, Theorem 3]:

Lemma 3.6. *If $\mathcal{P} \subseteq \mathbb{F}^2$ is a point set with $|\mathcal{P}| = n$ and $\boldsymbol{b} = (b_1, \ldots, b_s) \in \mathbb{F}[X,Y]^s$ is the reduced $\prec_{\text{lex}}$-Gröbner basis of $\Gamma(\mathcal{P})$ ordered by $\prec_{\text{lex}}$, then $b_1 \in \mathbb{F}[X]$, and b_s is Y-monic with $\deg_Y b_s = \nu_Y(\mathcal{P})$.*

Using the above, we give degree constraint under which reshapers are guaranteed to exist:

Lemma 3.7. *If $\mathcal{P} \subseteq \mathbb{F}^2$ is a point set and $\breve{\eta} \geqslant \nu_Y(\mathcal{P}) - 1$, then for any $\eta > \breve{\eta}$ there exists an η-reshaper $g = Y^\eta - \breve{g} \in \Gamma(\mathcal{P})$ with $\deg_Y \breve{g} \leqslant \breve{\eta}$.*

Proof. By Lemma 3.6, the reduced $\prec_{\text{lex}}$-Gröbner basis $\boldsymbol{b} \in \mathbb{F}[X, Y]^s$ of $\Gamma(\mathcal{P})$ contains a polynomial with $\prec_{\text{lex}}$-leading term $Y^{\nu_Y(\mathcal{P})}$. It then follows that $\deg_Y(Y^\eta \operatorname{rem} \boldsymbol{b}) < \nu_Y(\mathcal{P})$, which implies that $g = Y^\eta - (Y^\eta \operatorname{rem} \boldsymbol{b}) \in \Gamma(\mathcal{P})$ is an η-reshaper. □

As outlined in Section 3.1, we will use a sequence of reshapers in order to iteratively reduce the Y-degree of the input polynomial. Lemma 3.5 guarantees that each iteration will be efficient as long as the Y-degree of the current reshaper is sufficiently large compared to that of the current remainder. As we will see later, choosing the reshapers in accordance with the following definition ensures that this is true.

Definition 3.8. We say that $\boldsymbol{\eta} \in (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ is an (η_0, η_γ)-*degree sequence* if $\eta_{i-1} > \eta_i \geqslant \lfloor \frac{2}{3}\eta_{i-1} \rfloor$ for $i = 1, \ldots, \gamma$. Furthermore, if $I \subseteq \mathbb{F}[X, Y]$ is an ideal, and if $g_i = Y^{\eta_i} - \breve{g}_i \in I$ is an η_i-reshaper with

$$\eta_{i-1} \leqslant 2\eta_i - \breve{\eta}_i \ , \tag{3.1}$$

where $\breve{\eta}_i = \deg_Y \breve{g}_i$ for $i = 1, \ldots, \gamma$, then we say that $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in I^\gamma$ is an $\boldsymbol{\eta}$-*reshaping sequence* in I.

The following is an intermediate result used in the subsequent Lemma 3.10 in order to show that we can always find a reshaping sequence with number of elements being logarithmic in the degree of the first reshaper.

Lemma 3.9. *If $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma}$ is a degree sequence, then*

$$2\eta_i - \eta_{i-1} \geqslant \frac{\eta_i}{3} - 1 \geqslant 0$$

for $i = 1, \ldots, \gamma$.

Proof. Since $\eta_{i-1} > \eta_i \geqslant \lfloor \frac{2}{3}\eta_{i-1} \rfloor \geqslant \frac{2}{3}(\eta_{i-1} - 1)$, then

$$2\eta_i - \eta_{i-1} \geqslant \frac{4}{3}(\eta_{i-1} - 1) - \eta_{i-1} = \frac{\eta_{i-1} - 4}{3} \geqslant \frac{\eta_i}{3} - 1 > -1 \ .$$

□

Lemma 3.10. *If $\mathcal{P} \subseteq \mathbb{F}^2$ is a point set with $|\mathcal{P}| = n$, and if $a, b \in \mathbb{Z}_{>0}$ are such that $n > a > b \geqslant \nu_Y(\mathcal{P})$, then there exists an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} \in \Gamma(\mathcal{P})^\gamma$, where $\boldsymbol{\eta} \in \mathbb{Z}_{>0}^{\gamma+1}$ is an (a, b)-degree sequence with $\gamma \leqslant \log_{3/2}(a) + 1 \in \mathcal{O}(\log(a))$.*

Proof. Let $v = \nu_Y(\mathcal{P}) - 1$ and let $\boldsymbol{\eta}' = (\eta'_i) \in \mathbb{Z}_{>0}^{\gamma+1}$ be any $(a - v, b - v)$-degree sequence with $\gamma \leqslant \log_{3/2}(a - v) + 1$. To see that $\boldsymbol{\eta}'$ exists, simply choose $\eta'_i = \lfloor \frac{2}{3}\eta'_{i-1} \rfloor$ for $i = 1, \ldots, \gamma$, setting $\eta'_0 = a - v$. Now, let $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma}$ be such that $\eta_i = \eta'_i + v$ for $i = 0, \ldots, \gamma$, and observe that $\boldsymbol{\eta}$ is an (a, b)-degree sequence since

$$\eta_{i-1} > \eta_i = \eta'_i + v \geqslant \lfloor \frac{2}{3}\eta'_{i-1} \rfloor + v = \lfloor \frac{2}{3}(\eta'_{i-1} + v) + \frac{1}{3}v \rfloor = \lfloor \frac{2}{3}\eta_{i-1} + \frac{1}{3}v \rfloor \geqslant \lfloor \frac{2}{3}\eta_{i-1} \rfloor .$$

To show that there exists an $\boldsymbol{\eta}$-reshaping sequence in $\Gamma(\mathcal{P})$ it suffices to show that for $i = 1, \ldots, \gamma$ there exists an η_i-reshaper $g_i = Y^{\eta_i} - \breve{g} \in \Gamma(\mathcal{P})$ with $\deg_Y \breve{g}_i \leqslant \breve{\eta}_i$ satisfying (3.1). But indeed, choosing $\breve{\eta}_i = 2\eta_i - \eta_{i-1}$ we get from Lemma 3.9 that

$$\breve{\eta}_i = 2\eta'_i - \eta'_{i-1} + v \geqslant v = \nu_Y(\mathcal{P}) - 1 ,$$

which due to Lemma 3.7 implies that g_i exists. □

Now we are finally ready to present the reshaping algorithm.

Algorithm 6 Reshape$(f, \boldsymbol{g})$

Input:

- A polynomial $f \in \mathbb{F}[X, Y]$,
- an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in I^\gamma$, where $I \subseteq \mathbb{F}[X, Y]$ is some ideal and $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ is a degree sequence with $\deg_Y f < \eta_0$.

Output:

- a polynomial $h \in f + I$ with $\deg_Y h < \eta_\gamma$ and $\deg_X h \leqslant \deg_X f + \sum_{i=1}^{\gamma} g_i$.

```
h ← f
for i = 1, ..., γ do
    ĥ, h̆ ∈ F[X, Y] ← polynomials such that h = Y^{η_i} ĥ + h̆ with deg_Y h̆ < η_i
    h ← h̆ + ĥ ğ_i, where g_i = Y^{η_i} − ğ_i with deg_Y ğ_i < η
return h
```

Theorem 3.11. *Algorithm 6 is correct and has complexity*

$$\mathcal{O}\big(\mathsf{M}\big(\gamma \deg_X f \deg_Y f + \gamma \sum_{i=i_0}^{\gamma} \eta_i \deg_X g_i\big)\big)$$
$$\subseteq \widetilde{\mathcal{O}}(\gamma \deg_X f \deg_Y f + \gamma \sum_{i=i_0}^{\gamma} \eta_i \deg_X g_i)$$

for the smallest i_0 such that $\eta_{i_0} \leqslant \deg_Y f$.

Proof. Let $h_i, \widehat{h}_i$ and $\check{h}_i$ respectively denote the values of $h, \widehat{h}$ and $\check{h}$ at the end of iteration i, and let $\check{\eta}_i = \deg_Y \check{g}_i$. Since $f \in f + I$ and $\deg_Y f < \eta_0 \leqslant 2\eta_1 - \check{\eta}_1$, then applying Lemma 3.5 inductively guarantees that $h_i \in f + I$ with

$$\deg_X h_i \leqslant d_i := \deg_X f + \sum_{j=1}^{i} \deg_X g_j \quad \text{and} \quad \deg_Y h_i < \eta_i$$

for $i = 1, \ldots, \gamma$. No computation is performed for $i < i_0$, since then $\widehat{h}_i = 0$ and consequently $h_i = \check{h}_i$. The cost of the i-th iteration for $i \geqslant i_0$ is $\mathcal{O}(\mathsf{M}(d_i \eta_i))$, making the total cost of the algorithm

$$\begin{aligned}
\mathcal{O}\big(\sum_{i=i_0}^{\gamma} \mathsf{M}(d_i \eta_i)\big) &= \mathcal{O}\big(\sum_{i=i_0}^{\gamma} \mathsf{M}\big((\deg_X f + \sum_{j=i_0}^{i} \deg_X g_j)\eta_i\big)\big) \\
&\subseteq \mathcal{O}\big(\mathsf{M}\big(\sum_{i=i_0}^{\gamma} \eta_i \deg_X f + \sum_{i=i_0}^{\gamma} \sum_{j=i_0}^{i} \eta_i \deg_X g_j\big)\big) \\
&\subseteq \mathcal{O}\big(\mathsf{M}\big(\gamma \deg_X f \deg_Y f + \sum_{i=i_0}^{\gamma} \sum_{j=i_0}^{i} \eta_j \deg_X g_j\big)\big) \\
&\subseteq \mathcal{O}\big(\mathsf{M}\big(\gamma \deg_X f \deg_Y f + \gamma \sum_{i=i_0}^{\gamma} \eta_i \deg_X g_i\big)\big) \ ,
\end{aligned}$$

where we have used that $\eta_\gamma < \cdots < \eta_{i_0} \leqslant \deg_Y f$. □

Recall from the informal description in Section 3.1 that a point set is balanced if there exists a reshaping sequence whose elements have small X-degrees. In the following definition we make this notion of balancedness precise:

Definition 3.12. Let $\mathcal{P} \subseteq \mathbb{F}^2$ be a point set with $|\mathcal{P}| = n$ and let $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ be a degree sequence. We say that $\mathcal{P}$ is *$\boldsymbol{\eta}$-balanced* if there exists an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in \Gamma(\mathcal{P})^\gamma$ with

$$\deg_X g_i \leqslant \left\lfloor \frac{n}{2\eta_i - \eta_{i-1} + 1} \right\rfloor + 1 \quad \text{for} \quad i = 1, \ldots, \gamma \ .$$

For the remainder of this chapter, whenever we speak of a reshaping sequence of a balanced point set, we will by default assume that it satisfies the degree bounds in Definition 3.12.

The following Lemma 3.13, in combination with Lemma 3.10 and Theorem 3.11, guarantees that reshaping with respect to a balanced point set can always be done with quasi-linear cost.

Lemma 3.13. *If $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ is a degree sequence and $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in \Gamma(\mathcal{P})^{\gamma}$ is a $\boldsymbol{\eta}$-reshaping sequence for some $\boldsymbol{\eta}$-balanced point set $\mathcal{P} \subseteq \mathbb{F}^2$ with $|\mathcal{P}| = n$, then*

$$\sum_{i=0}^{\gamma} \eta_i \deg_X g_i \leqslant \gamma(3n + \eta_{i_0})$$

for any $i_0 \in \{1, \ldots, \gamma\}$.

Proof. From Lemma 3.9 it follows that if $\mathcal{P}$ is balanced, then we can choose

$$\deg_X g_i \leqslant \left\lfloor \frac{n}{2\eta_i - \eta_{i-1} + 1} \right\rfloor + 1 \leqslant \frac{n}{n_i/3} + 1 = \frac{3n}{\eta_i} + 1 \ .$$

Consequently,

$$\sum_{i=i_0}^{\gamma} \eta_i \deg_X g_i \leqslant \sum_{i=i_0}^{\gamma} \eta_i \left(\frac{3n}{\eta_i} + 1 \right) \leqslant \gamma(3n + \eta_{i_0}) \ .$$

□

3.3 Multi-point evaluation

In Section 3.2 we saw that reshaping with respect to a balanced point set can be done efficiently. Taking advantage of this, we obtain a simple algorithm for bivariate MPE on point sets with pairwise distinct X-coordinates:

Algorithm 7 $\mathsf{MPE}_{\mathsf{distinctX}}(f, \mathcal{P}, \boldsymbol{g})$

Input:

- A polynomial $f \in \mathbb{F}[X, Y]$,
- a point set $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^n \subset \mathbb{F}^2$ where the α_j are pairwise distinct,
- an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} \in \Gamma(\mathcal{P})^\gamma$ where $\boldsymbol{\eta} = (\eta_i)_{i=0}^\gamma \in \mathbb{Z}_{>0}^{\gamma+1}$ is an $(\eta_0, 1)$-degree sequence with $\deg_Y f < \eta_0$.

Output:

- Evaluations $\big(f(\alpha_1, \beta_1), \ldots, f(\alpha_n, \beta_n)\big) \in \mathbb{F}^n$.

1: $h \in \mathbb{F}[X] \leftarrow \mathsf{Reshape}(f, \boldsymbol{g})$ ▷ Algorithm 6
2: **return** $\big(h(\alpha_1), \ldots h(\alpha_n)\big) \in \mathbb{F}^n$ ▷ univariate MPE

Theorem 3.14. *Algorithm 7 is correct. If $\mathcal{P}$ is $\boldsymbol{\eta}$-balanced and $\gamma \in \mathcal{O}(\log n)$, then it costs $\widetilde{\mathcal{O}}(\deg_X f \deg_Y f + n)$ operations in $\mathbb{F}$, where $n = |\mathcal{P}|$.*

Proof. Correctness is an immediate consequence of Theorem 3.11, which guarantees that $h \in f + \Gamma(\mathcal{P})$, i.e. $h(\alpha_j) = f(\alpha_j, \beta_j)$ for $j = 1, \ldots, n$. The complexity of Step 1 is given as

$$\widetilde{\mathcal{O}}(\gamma \deg_X f \deg_Y f + \gamma \sum_{i=i_0}^{\gamma} \eta_i \deg_X g_i)$$

for the smallest i_0 with $\eta_{i_0} \leqslant \deg_Y f$. Step 2 costs $\widetilde{\mathcal{O}}(\deg_X f + \sum_{i=i_0}^{\gamma} \deg_X g_i)$, which is subsumed by Step 1. The total cost of the algorithm then follows from Lemma 3.13 and the assumption that $\gamma \in \mathcal{O}(\log n)$. □

Algorithm 7 can easily be extended to the case of nondistinct X-coordinates ($\nu_Y(\mathcal{P}) > 1$) by simply partitioning $\mathcal{P}$ into $\nu_Y(\mathcal{P})$ subsets whose X-coordinates are pairwise distinct. Although this approach incurs an additional factor $\nu_Y(\mathcal{P})$ in its complexity, this is not an issue if $\nu_Y(\mathcal{P})$ is small. For example, all point sets on elliptic curves satisfy $\nu_Y(\mathcal{P}) \leqslant 2$.

If the size of $\nu_Y(\mathcal{P})$ is significant, then we can shear the points in an extension field as proposed by Nüsken and Ziegler (see Section 2.2.2). To summarize: picking any $\theta \in \mathbb{K} \backslash \mathbb{F}$, where $\mathbb{K}$ is a degree 2 extension of $\mathbb{F}$, we can substitute the problem of evaluating $f \in \mathbb{F}[X, Y]$ on $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^n \subseteq \mathbb{F}^2$ with that of evaluating $\widetilde{f} := f(X - \theta Y, Y) \in \mathbb{K}[X, Y]$ on $\widetilde{\mathcal{P}} := \{(\alpha_j + \theta\beta_j, \beta_j)\}_{j=1}^n \subset \mathbb{K}^2$. This technique effectively reduces the general problem of bivariate MPE to the case of $\nu_Y(\mathcal{P}) = 1$, provided that we can compute $\widetilde{f}$ efficiently. Nüsken and Ziegler

achieve this with cost

$$\mathcal{O}(\mathsf{M}(d_X(d_X+d_Y))\log d_X) \subset \widetilde{\mathcal{O}}(d_X(d_X+d_Y))$$

using a univariate Taylor shift of f seen as an element in $\mathbb{K}[Y][X]$. Although this is sufficient for our target cost (assuming w.l.o.g. that $d_X \leqslant d_Y$), we present an alternative approach which is more efficient on the logarithmic level by using univariate Taylor shifts on the homogeneous components of f.

Remark 3.15. Although we don't know whether shearing preserves balancedness, as we will see in Section 3.7, a sheared version of a random point set is balanced with high probability.

Algorithm 8 $\mathsf{ShearPoly}(f, \alpha, \beta)$

Input:

- A polynomial $f = \sum_{i=0}^{d_X}\sum_{j=0}^{d_Y} f_{i,j}X^iY^j \in \mathbb{K}[X,Y]$ with $f_{i,j} \in \mathbb{K}$,
- scalars $\alpha, \beta \in \mathbb{K}$.

Output:

- $\widetilde{f} = f(\alpha X + \beta Y, Y) \in \mathbb{K}[X,Y]$.

1: **for** $t = 0, \ldots, d_X + d_Y$ **do**
2: $\quad h_t \in \mathbb{K}[Z] \leftarrow \sum_{i=\max\{0,t-d_Y\}}^{\min\{t,d_X\}} f_{i,t-i}Z^i$
3: $\quad \widetilde{h}_t \in \mathbb{K}[Z] \leftarrow h_t(\alpha Z + \beta)$
4: **return** $\widetilde{f} = \sum_{t=0}^{d_X+d_Y} \widetilde{h}_t(X/Y)Y^t$

Theorem 3.16. *Algorithm 8 correctly computes $f(\alpha X + \beta Y, Y)$ using*

$$\mathcal{O}((d_X+d_Y)\mathsf{M}(d_X)\log(d_X)) \subset \widetilde{\mathcal{O}}(d_X(d_X+d_Y))$$

operations in $\mathbb{K}$. Furthermore, the X-degree of the output polynomial is at most d_X, while its Y-degree is at most $\max\{d_X, d_Y\}$.

Proof. The degree bounds on $f(\alpha X + \beta Y, Y)$ are obvious. Observing that

$$\sum_{t=0}^{d_X+d_Y} h_t(X/Y)Y^t = \sum_{t=0}^{d_X+d_Y} \sum_{i=\max\{0,t-d_Y\}}^{\min\{t,d_X\}} f_{i,t-i}X^iY^{t-i} = \sum_{i=0}^{d_X}\sum_{j=0}^{d_Y} f_{i,j}X^iY^j = f\ ,$$

i.e. that $h_t(X/Y)Y^t \in \mathbb{F}[X,Y]$ is the homogeneous component of f having degree t, we get that

$$\widetilde{f} = \sum_{t=0}^{d_X+d_Y} \widetilde{h}_t(X/Y)Y^t = \sum_{t=0}^{d_X+d_Y} h_t(\frac{\alpha X + \beta Y}{Y})Y^t = f(\alpha X + \beta Y, Y)\ .$$

For the complexity, we note that arithmetic operations are performed only in Step 3. Using [VZG90, Fact 2.1 (iv)], we can compute the Taylor shifts

$$h_t(Z) \mapsto h_t(\alpha Z + \beta)$$

for each t using $\mathcal{O}(\mathsf{M}(d_X)\log(d_X))$ operations in $\mathbb{K}$, since $\deg h_t \leqslant d_X$. Summation over all t yields the total complexity. □

Remark 3.17. As presented, Algorithm 8 can only shear the variable X. We can, however, easily shear Y by simply swapping the variables. For the sake of being precise (but at the risk of stating the obvious) let $\sigma : f(X,Y) \mapsto f(Y,X)$ for any $f \in \mathbb{K}[X,Y]$, and observe that

$$\begin{aligned} f(X, \alpha X + \beta Y) &= \sigma(f(Y, \alpha Y + \beta X)) \\ &= \sigma\big(\mathsf{ShearPoly}(f(Y,X), \beta, \alpha)\big) \\ &= \big(\sigma \circ \mathsf{ShearPoly}(\,\cdot\,, \beta, \alpha) \circ \sigma\big)(f)\ . \end{aligned}$$

We conclude this section with an algorithm for MPE where the X-coordinates of the underlying point set are allowed to repeat.

Algorithm 9 $\mathsf{MPE}_{\mathsf{shear}}(f, \widetilde{\mathcal{P}}, \widetilde{\boldsymbol{g}})$

Input:

- A polynomial $f \in \mathbb{F}[X,Y]$,
- an element $\theta \in \mathbb{K}\backslash\mathbb{F}$, where $\mathbb{K}$ is a degree 2 extension of $\mathbb{F}$
- a sheared version $\widetilde{\mathcal{P}} = \{(\alpha_j + \theta\beta_j, \beta_j)\}_{j=1}^n \subset \mathbb{K}^2$ of a point set $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^n \subseteq \mathbb{F}^2$,
- an $\widetilde{\boldsymbol{\eta}}$-reshaping sequence $\widetilde{\boldsymbol{g}} \in \Gamma(\widetilde{\mathcal{P}})^{\widetilde{\gamma}} \subset \mathbb{K}[X,Y]^{\widetilde{\gamma}}$, where $\widetilde{\boldsymbol{\eta}} = (\widetilde{\eta}_i)_{i=0}^{\widetilde{\gamma}}$ is an $(\widetilde{\eta}_0, 1)$-degree sequence with $\max\{\deg_X f, \deg_Y f\} < \widetilde{\eta}_0$.

Output:

- Evaluations $\big(f(\alpha_1, \beta_1), \ldots, f(\alpha_n, \beta_n)\big) \in \mathbb{F}^n$.

1: $\widetilde{f} \in \mathbb{K}[X,Y] \leftarrow f(X - \theta Y, Y) = \mathsf{ShearPoly}(f, 1, -\theta)$ ▷ Algorithm 8
2: **return** $\mathsf{MPE}_{\mathsf{distinctX}}(\widetilde{f}, \widetilde{\mathcal{P}}, \widetilde{\boldsymbol{g}})$ ▷ Algorithm 7

Theorem 3.18. *Algorithm 9 is correct. If $\widetilde{\mathcal{P}}$ is $\widetilde{\boldsymbol{\eta}}$-balanced and $\widetilde{\gamma} \in \mathcal{O}(\log(n))$, then it costs $\widetilde{\mathcal{O}}(\deg_X f(\deg_X f + \deg_Y f) + n)$ operations in $\mathbb{F}$.*

Proof. For any $(\alpha_j + \theta\beta_j, \beta_j) \in \widetilde{\mathcal{P}}$ it holds that

$$\widetilde{f}(\alpha_j + \theta\beta_j, \beta_j) = f(\alpha_j + \theta\beta_j - \theta\beta_j, \beta_j) = f(\alpha_j, \beta_j)\ .$$

The rest follows from Theorem 3.16 and Theorem 3.14, i.e. correctness and complexity of $\mathsf{ShearPoly}$ and $\mathsf{MPE}_{\mathsf{distinctX}}$ respectively. □

3.4 Interpolation

In this section we use reshaping in order to address the problem of bivariate interpolation in a similar setting as for MPE, i.e. we assume that the point set $\mathcal{P} \subseteq \mathbb{F}^2$ is available for precomputation and that we receive the interpolation values as input during the online phase. If $\mathcal{P}$ is appropriately balanced, we can compute an interpolating polynomial with reasonably controlled monomial support in quasi-linear time.

In contrast to MPE, our strategy now is to first shear $\mathcal{P}$ so that all of its Y-coordinates are pairwise distinct. After computing a univariate interpolating polynomial $\widetilde{f} \in \mathbb{K}[Y]$, we reshape it into $\widetilde{h} \in \mathbb{K}[X, Y]$ with the degrees in both variables being roughly $\sqrt{n}$ (assuming it is possible). These degree constraints ensure that we can efficiently "unshear" $\widetilde{h}$ to interpolate on the original point set, and a final application of reshaping allows us to meet some targeted Y-degree.

Algorithm 10 Interpolate($\boldsymbol{v}, \,$)

Input:

- Interpolation values $\boldsymbol{v} = (v_j)_{j=1}^n \in \mathbb{F}^n$,
- an integer $d \in \mathbb{Z}_{>0}$,
- an $(\lfloor\sqrt{n}\rfloor, d)$-degree sequence $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma}$,
- an $(\widetilde{\eta}_0, \lfloor\sqrt{n}\rfloor)$-degree sequence $\widetilde{\boldsymbol{\eta}} = (\widetilde{\eta}_i)_{i=0}^{\widetilde{\gamma}}$,
- an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in \mathbb{F}[X,Y]^{\gamma}$ for a point set $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^n \subseteq \mathbb{F}^2$ with $n_X(\mathcal{P}) \leqslant d < \sqrt{n}$
- an $\widetilde{\boldsymbol{\eta}}$-reshaping sequence $\widetilde{\boldsymbol{g}} = (\widetilde{g}_i)_{i=1}^{\widetilde{\gamma}} \in \mathbb{K}[X,Y]^{\widetilde{\gamma}}$ for $\widetilde{\mathcal{P}} = \{(\alpha_j, \theta\alpha_j + \beta_j)\}_{j=1}^n \subset \mathbb{K}^2$,

Output:

- A polynomial $f \in \mathbb{F}[X,Y]$ satisfying $f(\alpha_j, \beta_j) = v_j$ for $j = 1, \ldots, n$, $\deg_Y f < d$ and $\deg_X f \leqslant \max\{\sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i, \lfloor\sqrt{n}\rfloor - 1\} + \sum_{i=1}^{\gamma} \deg_X g_i$.

1: $\widetilde{f} \in \mathbb{K}[Y] \leftarrow$ polynomial with $\deg \widetilde{f} < n$ and $\widetilde{f}(\theta\alpha_j + \beta_j) = v_j$ for $j = 1, \ldots, n$
2: $\widetilde{h} \in \mathbb{K}[X,Y] \leftarrow \mathsf{Reshape}(\widetilde{h}, \widetilde{\boldsymbol{g}})$ ▷ Algorithm 6
3: $h \in \mathbb{K}[X,Y] \leftarrow \widetilde{h}(X, \theta X + Y)$ using ShearPoly ▷ Algorithm 8 with Remark 3.17
4: $h_1, h_\theta \in \mathbb{F}[X,Y] \leftarrow$ polynomials such that $h = h_1 + \theta h_\theta$
5: **return** $f \in \mathbb{F}[X,Y] \leftarrow \mathsf{Reshape}(h_1, \boldsymbol{g})$

Theorem 3.19. *Algorithm 14 is correct and costs*

$$\widetilde{\mathcal{O}}\Big(\widetilde{\gamma}n + \gamma\big(\sqrt{n} + \sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i\big)^2 + \widetilde{\gamma}\sum_{i=1}^{\widetilde{\gamma}} \widetilde{\eta}_i \deg_X \widetilde{g}_i + \gamma\sum_{i=1}^{\gamma} \eta_i \deg_X g_i\Big)$$

operations in $\mathbb{F}$. If $\widetilde{\mathcal{P}}$ is $\widetilde{\boldsymbol{\eta}}$-balanced, $\mathcal{P}$ is $\boldsymbol{\eta}$-balanced and $\widetilde{\gamma}, \gamma \in \mathcal{O}(\log n)$, then the complexity becomes $\widetilde{\mathcal{O}}(n)$.

Proof. First note that due to Lemma 3.10, we can always choose $\gamma, \widetilde{\gamma} \in \mathcal{O}(n)$ since $d \geqslant n_X(\mathcal{P})$. For correctness, observe that all points in $\widetilde{\mathcal{P}}$ have pairwise distinct Y-coordinates, so we can indeed compute $\widetilde{f}$ using univariate Lagrange

interpolation. Viewing $\widetilde{f}$ as an element of $\mathbb{K}[X,Y]$ with $\deg_X \widetilde{f} = 0$, we get

$$\begin{aligned}
v_j &= \widetilde{f}(\alpha_j, \theta\alpha_j + \beta_j) \\
&= \widetilde{h}(\alpha_j, \theta\alpha_j + \beta_j) && \text{(Theorem 3.11)} \\
&= h(\alpha_j, \beta_j) && \text{(Theorem 3.16)} \\
&= h_1(\alpha_j, \beta_j) + \theta h_\theta(\alpha_j, \beta_j) \\
&= h_1(\alpha_j, \beta_j) && \text{(since } \alpha_j, \beta_j, v_j \in \mathbb{F} \text{ and } h_1, h_\theta \in \mathbb{F}[X,Y]) \\
&= f(\alpha_j, \beta_j)\ . && \text{(Theorem 3.11)}
\end{aligned}$$

For the degree bounds, note that $\deg_Y \widetilde{h} < \lfloor\sqrt{n}\rfloor$ and $\deg_X \widetilde{h} \leqslant \sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i$ by Theorem 3.11. Consequently, $\deg_Y h_1 \leqslant \deg_Y h \leqslant \deg_Y \widetilde{h} < \lfloor\sqrt{n}\rfloor$ and

$$\deg_X h_1 \leqslant \deg_X h \leqslant \max\{\deg_X \widetilde{h}, \deg_Y \widetilde{h}\} \leqslant \max\{\sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i, \lfloor\sqrt{n}\rfloor - 1\}\ .$$

Finally, again by Theorem 3.11, $\deg_Y f < d$ and $\deg_X f \leqslant \deg_X h_1 + \sum_{i=1}^{\gamma} \deg_X g_i$.

The total cost follows from summing the costs of the individual steps:

- Step 1: $\widetilde{\mathcal{O}}(n)$ by fast univariate interpolation,
- Step 2: $\widetilde{\mathcal{O}}(\widetilde{\gamma}n + \widetilde{\gamma}\sum_{i=1}^{\widetilde{\gamma}} \widetilde{\eta}_i \deg_X \widetilde{g}_i)$ by Theorem 3.11,
- Step 3: $\widetilde{\mathcal{O}}\big((\sqrt{n} + \sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i)^2\big)$ by Theorem 3.16,
- Step 4: memory management (no operations in $\mathbb{F}$),
- Step 5: $\widetilde{\mathcal{O}}\big(\gamma(\sqrt{n} + \sum_{i=1}^{\widetilde{\gamma}} \deg_X \widetilde{g}_i)^2 + \gamma\sum_{i=1}^{\gamma} \eta_i \deg_X g_i\big)$ by Theorem 3.11.

□

3.5 Precomputing Reshapers

The simplest way to precompute a reshaper for a point set is perhaps to solve an $r \times n$ inhomogeneous linear system, where $n = |\mathcal{P}|$ and r is the cardinality of the allowed monomial support of the reshaper. If $\mathcal{P}$ is balanced, then we can choose $r \approx n$, and so the complexity of this approach is $\mathcal{O}(n^\omega)$. In this section, we present an alternative technique which works for any zero-dimensional ideal

$I \subseteq \mathbb{F}[X,Y]$, given a $\prec_{\text{lex}}$-Gröbner basis of I. We will make extensive use of the $\mathbb{F}[X]$-module $I_d := I \cap \mathbb{F}[X,Y]_{\deg_Y < d}$, where

$$\mathbb{F}[X,Y]_{\deg_Y < d} = \{f \in \mathbb{F}[X,Y] \mid \deg_Y f < d\} \ ,$$

so we begin by establishing its relation with I as a consequence of Lazard's structure theorem on bivariate $\prec_{\text{lex}}$-Gröbner bases [Laz85].

In the following, for any $f \in \mathbb{F}[X,Y]$ we denote by $\operatorname{lt}(f) \in \mathbb{F}[X,Y]$ the *leading term* of f, and by $\operatorname{lc}_Y(f) \in \mathbb{F}[X]$ the leading Y-coefficient of f when viewed as an element of $\mathbb{F}[X][Y]$.

Corollary 3.20 (of [Laz85])**.** *If $\boldsymbol{b} = (b_0, \ldots, b_\ell) \in \mathbb{F}[X,Y]^{\ell+1}$ is a minimal $\prec_{\text{lex}}$-Gröbner basis, sorted by increasing Y-degree, then*

1) $\deg_Y b_0 < \cdots < \deg_Y b_\ell$*, and*

2) $\operatorname{lc}_Y(b_\ell) \mid \operatorname{lc}_Y(b_{\ell-1}) \mid \cdots \mid \operatorname{lc}_Y(b_0)$*.*

Lemma 3.21. *Let $\boldsymbol{b} = (b_0, \ldots, b_\ell) \in \mathbb{F}[X,Y]^{\ell+1}$ be a minimal $\prec_{\text{lex}}$-Gröbner basis for an ideal I, ordered by increasing Y-degree. If for any $d \in \mathbb{Z}_{>0}$*

$$\begin{aligned} I_d &= \{f \in I \mid \deg_Y f < d\} \ , \\ s &= \max\{t \mid \deg_Y b_k < d,\ 0 \leqslant t \leqslant \ell\} \ , \\ d_t &= \deg_Y b_t \quad \textit{for} \quad 0 \leqslant t \leqslant s \quad \textit{and} \quad d_{s+1} = d \ , \end{aligned}$$

then I_d is an $\mathbb{F}[X]$-submodule of $\mathbb{F}[X,Y]_{\deg_Y < d}$. Moreover, I_d is free with rank $d - d_0$ and admits the basis

$$B_d := \{Y^u b_t \mid 0 \leqslant t \leqslant s,\ 0 \leqslant u < d_{t+1} - d_t\} \ .$$

Proof. It is clear that I_d is an $\mathbb{F}[X]$-submodule of $\mathbb{F}[X,Y]_{\deg_Y < d}$ since I is an ideal of $\mathbb{F}[X,Y]$. Furthermore, $B_d \subset I_d$ since

$$\deg_Y(Y^u b_t) \leqslant \deg_Y(Y^{d_{s+1}-d_s} b_s) < d - d_s + d_s = d \ ,$$

and the elements of B_d have pairwise distinct Y-degrees and are therefore linearly independent over $\mathbb{F}[X]$. Also $|B_d| = d - d_0$, which implies that I_d has rank $d - d_0$ as long as B_d generates I_d. To see that it does, pick some $f \in I_d$ and note that since $f \in I$, then we can write $f = \sum_{t=0}^{\ell} q_t b_t$, where $q_t \in \mathbb{F}[X,Y]$ with $\deg_Y q_t \leqslant \deg_Y f - \deg_Y b_t$. Since $\deg_Y f < d$, then $q_{s+1} = \cdots = q_\ell = 0$. It is not hard to see that we can choose the q_t such that no term of $q_t b_t$ is divisible by $\operatorname{lt}(b_{t+1})$ for any $t < s$: in each iteration of the multivariate division algorithm, simply use the greatest index $t \in \{0, \ldots, \ell\}$ for which $\operatorname{lt}(b_t)$

divides the leading term in the current remainder. By Corollary 3.20 then $\mathrm{lc}_Y(b_{t+1}) \mid \mathrm{lc}_Y(b_t)$, and so if $\deg_Y(q_t b_t) \geqslant \deg_Y b_{t+1}$ then $\mathrm{lt}(b_{t+1}) \mid \mathrm{lt}(q_t b_t)$. Consequently, $\deg_Y q_t < \deg_Y b_{t+1} - \deg_Y b_t$, and therefore $f \in \langle b_0, \ldots, b_t \rangle_{\mathbb{F}[X]}$. $\square$

For any $d \in \mathbb{Z}_{>0}$ we define the $\mathbb{F}[X]$-module isomorphism

$$\phi_d : \begin{cases} \mathbb{F}[X,Y]_{\deg_Y < d} \to \mathbb{F}[X]^{1\times d} \\ \sum_{j=0}^{d-1} f_j(X) Y^j \mapsto [f_0, \ldots, f_{d-1}] \end{cases} .$$

If I is a zero-dimensional ideal, then in Lemma 3.21 we have that $d_0 = 0$, and so I_d has rank d. Any basis B of I_d can then be represented as the nonsingular matrix $\boldsymbol{B} \in \mathbb{F}[X]^{m\times m}$ whose rows are of the form $\phi_d(b)$ for $b \in B$. Furthermore, $\Delta(I_d) := \deg\det(\boldsymbol{B})$ does not depend on the choice of B, since all bases of I_d have the same determinant up to scalar multiplication.

In Algorithm 11 we show how to efficiently precompute reshapers – we will rely on the language of Popov forms, which can be recalled from Section 1.4 on page 6. The following result constitutes an essential ingredient:

Proposition 3.22. *([NV17]) There is an algorithm which for any $\boldsymbol{P} \in \mathbb{F}[X]^{d\times d}$ in Popov form and any $\boldsymbol{v} \in \mathbb{F}[X]^{1\times d}$ with (entrywise)*

$$\mathrm{cdeg}(\boldsymbol{v}) < \mathrm{cdeg}(\boldsymbol{P}) + (\Delta(\boldsymbol{P}), \ldots, \Delta(\boldsymbol{P}))$$

computes $\boldsymbol{v} \operatorname{rem} \boldsymbol{P}$ using $\widetilde{\mathcal{O}}(d^{\omega-1}\Delta(\boldsymbol{P}))$, provided that $d \in \mathcal{O}(\Delta(\boldsymbol{P}))$.

Algorithm 11 ComputeReshaper$(\boldsymbol{b}, \eta, d)$

Input:

- A reduced $\prec_{\mathrm{lex}}$-Gröbner basis $\boldsymbol{b} = (b_0, \ldots, b_\ell) \in \mathbb{F}[X,Y]^{\ell+1}$, ordered by increasing Y-degree with $b_0 \in \mathbb{F}[X]$, for a zero-dimensional ideal I,
- positive integers η, d with $d < \eta$.

Output:

- A polynomial $g = Y^\eta - \breve{g} \in I$ with $\deg_Y \breve{g} < d$ and $\deg_X \breve{g}$ minimal if it exists, otherwise "Fail".

1: $R \in \mathbb{F}[X,Y] \leftarrow Y^\eta \operatorname{rem} \boldsymbol{b}$
2: **if** $\deg_Y R \geqslant d$ **then return** "Fail"
3: $B_d \subset \mathbb{F}[X,Y] \leftarrow \mathbb{F}[X]$-basis of $I_d = I \cap \mathbb{F}[X,Y]_{\deg_Y < d}$ as in Lemma 3.21
4: $\boldsymbol{B} \in \mathbb{F}[X]^{m\times m} \leftarrow$ matrix whose rows are of the form $\phi_d(b)$ for $b \in B_d$
5: $\boldsymbol{P} \in \mathbb{F}[X]^{m\times m} \leftarrow$ Popov form of $\boldsymbol{B}$
6: $\breve{g} \in \mathbb{F}[X,Y] \leftarrow -\phi_d^{-1}(\phi_d(R) \operatorname{rem} \boldsymbol{P})$
7: **return** $g = Y^\eta - \breve{g} \in \mathbb{F}[X,Y]$

Theorem 3.23. *Algorithm 11 is correct, and assuming that $\eta \in \mathcal{O}(\Delta(I_d))$, it costs $\widetilde{\mathcal{O}}(d^{\omega-1}\Delta(I_d) + \eta\ell \deg_X b_0)$ operations in $\mathbb{F}$.*

Proof. Since $\boldsymbol{b}$ is a $\prec_{\text{lex}}$-Gröbner basis, then $\deg_Y R \leqslant \deg_Y \check{g} < \delta$, provided that the sought g exists. Consequently, the algorithm does not fail at Step 2 in this case.

For correctness, since $Y^\eta - R \in I$, then any satisfactory reshaper $Y^\eta - \widetilde{g} \in I$ fulfills $\widetilde{g} \in R + I_d$. Since $\boldsymbol{P}$ represents a basis of I_d, then $\check{g}$ from Step 6 belongs to $R + I_d$, and since $\boldsymbol{P}$ is in Popov form, then $\check{g}$ has minimal X-degree in $R + I_d$.

For complexity, note that computational work is done only in Steps 1, 5 and 6. Since $\boldsymbol{b}$ is reduced, then $\deg_X b_0 > \cdots > \deg_X b_\ell$, which implies that the diagonal entries in $\boldsymbol{B}$ have dominant degrees in their respective columns. Furthermore, $|\operatorname{cdeg}(\boldsymbol{B})| = \Delta(\boldsymbol{B}) = \Delta(\boldsymbol{P}) = \Delta(I_d)$. The stated cost follows, since Step 1 costs $\widetilde{\mathcal{O}}(\eta\ell \deg_X b_0)$ due to [vdH15] (see Proposition 2.24), step 5 costs $\widetilde{\mathcal{O}}(d^{\omega-1}|\operatorname{cdeg} \boldsymbol{B}|)$ by Proposition 1.8 on page 8, and Step 6 costs $\widetilde{\mathcal{O}}(d^{\omega-1}\Delta(\boldsymbol{P}))$ by Proposition 3.22 because $\deg_X R < \deg_X b_0 < \Delta(\boldsymbol{P})$. □

3.6 Precomputing reduced $\prec_{\text{lex}}$-Gröbner basis

Since Algorithm 11 requires as input the reduced $\prec_{\text{lex}}$-Gröbner basis of $\Gamma(\mathcal{P})$, let us now consider the problem of computing it. Our strategy will be to first compute the *Hermite basis* of the $\mathbb{F}[X]$-submodule $\Gamma_d(\mathcal{P}) := \Gamma(\mathcal{P}) \cap \mathbb{F}[X,Y]_{\deg_Y < d}$, which by Lemma 3.6 and Lemma 3.21 is free and of rank d. This basis is unique, and its corresponding matrix $\boldsymbol{H} \in \mathbb{F}[X]^{d\times d}$ is lower triangular with each diagonal entry being monic and strictly dominating the degrees in its column.

Lemma 3.24. *If $\mathcal{P} \subseteq \mathbb{F}^2$ and $d > \nu_Y(\mathcal{P})$, then $\Gamma(\mathcal{P}) = \langle \Gamma_d(\mathcal{P}) \rangle_{\mathbb{F}[X,Y]}$ and $\Delta(\Gamma_d(\mathcal{P})) = |\mathcal{P}|$.*

Proof. Lemma 3.6 guarantees that all of the elements in the reduced $\prec_{\text{lex}}$-Gröbner basis of $\Gamma(\mathcal{P})$ have Y-degree at most $\nu_Y(\mathcal{P})$, which proves the first claim. It follows that $\mathbb{F}[X,Y]/\Gamma(\mathcal{P})$ and $\mathbb{F}[X,Y]_{\deg_Y<d}/\Gamma_d(\mathcal{P})$ are isomorphic as $\mathbb{F}[X]$-modules. It is a well known property of zero-dimensional varieties that the $\mathbb{F}$-dimension of the former – and thus also the latter – equals $|\mathcal{P}|$, and by [NV17, Lemma 2.3] this dimension is $\Delta(\Gamma_d(\mathcal{P}))$. □

Proposition 3.25. *There is an algorithm which for any $\mathcal{P} \subseteq \mathbb{F}^2$ with $|\mathcal{P}| = n$ computes the reduced $\prec_{\text{lex}}$-Gröbner basis of $\Gamma(\mathcal{P})$ using $\widetilde{\mathcal{O}}(d^{\omega-1}n)$ operations in $\mathbb{F}$, where $d = \nu_Y(\mathcal{P}) + 1$.*

Proof. The Hermite basis $\boldsymbol{H} \in \mathbb{F}[X]^{d\times d}$ of $\Gamma_d(\mathcal{P})$ can be computed with cost $\widetilde{\mathcal{O}}(d^{\omega-1}n)$ using a special case of [JNSV16, Theorem 1.5], where the shift

$$\boldsymbol{s} = (0, n, \ldots, (d-1)n) \in \mathbb{Z}^d$$

ensures that the $\boldsymbol{s}$-Popov basis $\boldsymbol{P}$ equals $\boldsymbol{H}$.

Letting $\boldsymbol{b} = (b_1, \ldots, b_d) \in \mathbb{F}[X,Y]^d$ with $b_j = \phi_d^{-1}(\boldsymbol{H}_j)$, where $\boldsymbol{H}_j \in \mathbb{F}[X]^{1\times d}$ denotes the j-th row of $\boldsymbol{H}$, note that $\Gamma(\mathcal{P}) = \langle b_1, \ldots, b_d\rangle_{\mathbb{F}[X,Y]}$ by Lemma 3.24, and since $\boldsymbol{H}$ is lower triangular, then $\boldsymbol{b}$ is a $\prec_{\text{lex}}$-Gröbner basis, albeit not necessarily a minimal one. To achieve minimality, simply exclude all elements b from $\boldsymbol{b}$ for which there exists another basis element b' whose leading term divides that of b [CLO15, Lemma 3, Chapter 2 §7]. Obtaining this new basis from $\boldsymbol{H}$ cost no operations in $\mathbb{F}$, and since $\boldsymbol{H}$ is in Hermite form, then it is not hard to see that this basis is necessarily reduced. □

Corollary 3.26. *Given a point set $\mathcal{P} \subseteq \mathbb{F}^2$ with $|\mathcal{P}| = n$ and a degree sequence $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ with $\log_{3/2}(\eta_0) + 1 \leqslant \gamma \in \log(n)$ and $\nu_Y(\mathcal{P}) < \eta_\gamma \leqslant n$, we can compute an $\boldsymbol{\eta}$-reshaping sequence $\boldsymbol{g} = (g_i)_{i=1}^{\gamma} \in \Gamma(\mathcal{P})^\gamma$ with the smallest possible X-degrees using $\widetilde{\mathcal{O}}(\eta_0^{\omega-1}n + \eta_0 dn)$ operations in $\mathbb{F}$, where $d = \nu_Y(\mathcal{P}) + 1$. Consequently, we can also determine whether $\mathcal{P}$ is $\boldsymbol{\eta}$-balanced within the same cost.*

Proof. Note first that the sought reshaping sequence exists due to Lemma 3.10. Using Proposition 3.25, we can compute a reduced $\prec_{\text{lex}}$-Gröbner basis $\boldsymbol{b}$ of $\Gamma(\mathcal{P})$ within the cost $\widetilde{\mathcal{O}}(d^{\omega-1}n) \subseteq \widetilde{\mathcal{O}}(\eta_0^{\omega-1}n)$. By Theorem 3.23, computing

$$g_i = \mathsf{ComputeReshaper}(\boldsymbol{b}, \eta_i, 2\eta_i - \eta_{i-1} + 1)$$

costs $\widetilde{\mathcal{O}}(\eta_0^{\omega-1}n + \eta_0 dn)$ for each i because $\Delta(\Gamma_{d'}(\mathcal{P})) = n$ for any $d' > \nu_Y(\mathcal{P})$ by Lemma 3.24. The total cost follows from the assumption that $\gamma \in \mathcal{O}(\log n)$. □

3.7 Balancedness

In Section 3.5 we have seen how to compute reshapers of minimal X-degree. The aim for this section is to show that for a random point set these X-degrees are with high probability small enough so that our algorithms are able to achieve quasi-linear complexity in the online phase. The following lemma forms the foundation of our probabilistic results:

Lemma 3.27 (DeMillo-Lipton-Schwartz-Zippel [DL78, Sch80, Zip79])**.** *Suppose that $f \in \mathbb{F}[X_1, \ldots, X_d]$ is non-zero of total degree d and $S \subseteq \mathbb{F}$. If $\alpha_1, \ldots, \alpha_d \in S$*

are chosen independently and uniformly at random, then the probability that $f(\alpha_1, \dots, \alpha_d) = 0$ *is at most* $d/|S|$.

For any matrix $\boldsymbol{A} \in \mathbb{F}[Y_1, \dots, Y_n]^{m \times m'}$ and any vector $\boldsymbol{v} = (v_1, \dots, v_n) \in \mathbb{F}^n$ let $\boldsymbol{A}^{(\boldsymbol{v})} \in \mathbb{F}^{m \times m'}$ denote the matrix obtained by substituting each Y_i in $\boldsymbol{A}$ with v_i. Furthermore, we denote by $\operatorname{rank}(\boldsymbol{A})$ the rank of $\boldsymbol{A}$ when considered over the field of fractions $\mathbb{F}(Y_1, \dots, Y_n)$.

Lemma 3.28. *If* $\alpha_1, \dots, \alpha_n \in \mathbb{F}$ *are pairwise distinct,* $s \in \mathbb{Z}_{>0}$ *and*

$$\boldsymbol{A}_s = [\boldsymbol{V}_s | \boldsymbol{D}\boldsymbol{V}_s | \dots | \boldsymbol{D}^{d-1}\boldsymbol{V}_s] \in \mathbb{F}[Y_1, \dots, Y_n]^{n \times ds} ,$$

where $\boldsymbol{V}_s = [\alpha_i^{j-1}]_{1 \leqslant j \leqslant s}^{1 \leqslant i \leqslant n} \in \mathbb{F}^{n \times s}$ *and* $\boldsymbol{D} \in \mathbb{F}[Y_1, \dots, Y_n]^{n \times n}$ *is the diagonal matrix with entries* $(Y_1, \dots, Y_n)$*, then* $\operatorname{rank}(\boldsymbol{A}_s) = \min\{n, ds\}$.

Proof. Letting $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}^n$, note that $\boldsymbol{A}_s^{(\boldsymbol{\alpha})} = [\alpha_i^{j-1}]_{1 \leqslant j \leqslant ms}^{1 \leqslant i \leqslant n} \in \mathbb{F}^{n \times ds}$ is a Vandermonde matrix. Since the α_i are pairwise distinct, then $\boldsymbol{A}_s^{(\boldsymbol{\alpha})}$. Consequently, the same holds for $\boldsymbol{A}_s$. □

In the context of Lemma 3.28, note that the columns of $\boldsymbol{A}_s$ naturally correspond to monomials $X^i Y^j$ with $i < s$ and $j < d$. To be more precise, if $\mathcal{P} = \{(\alpha_i, \beta_i)\}_{i=1}^n \subset \mathbb{F}^2$ with α_i pairwise distinct, then the $\mathbb{F}$-coefficients of any polynomial in $\Gamma(\mathcal{P})$ form a vector in the right kernel of $\boldsymbol{A}_s^{(\boldsymbol{\beta})} \in \mathbb{F}^{n \times ds}$, where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n) \in \mathbb{F}^n$.

In the following lemma we determine the exact row degrees of the Popov basis $\boldsymbol{P} \in \mathbb{F}[X]^{d \times d}$ of $\phi_d(\Gamma_d(\mathcal{P})) \subset \mathbb{F}[X]^{1 \times d}$ for a random point set $\mathcal{P}$, where $\phi_d : \mathbb{F}[X, Y]_{\deg_Y < d} \to \mathbb{F}[X]^{1 \times d}$ is the $\mathbb{F}[X]$-module isomorphism from Section 3.5 and $\Gamma_d(\mathcal{P}) = \Gamma(\mathcal{P}) \cap \mathbb{F}[X, Y]_{\deg_Y < d}$ is the $\mathbb{F}[X]$-module from Section 3.6.

Lemma 3.29. *Let* $\alpha_1, \dots, \alpha_n \in \mathbb{F}$ *be pairwise distinct, let* $\mathcal{T} \subseteq \mathbb{F}$ *be a finite subset, and let* $\lambda : \mathbb{F}^n \to \mathbb{F}^n$ *be an affine map. Additionally, let* $\tau_1, \dots, \tau_n \in \mathcal{T}$ *chosen independently and uniformly at random and let* $\mathcal{P} = \{(\alpha_i, \beta_i)\}_{i=1}^n \subset \mathbb{F}^2$ *be such that* $(\beta_1, \dots, \beta_n) = \lambda(\tau_1, \dots, \tau_n)$*. If* $d \in \mathbb{Z}_{>0}$ *with* $\nu_Y(\mathcal{P}) < d \leqslant n$ *and* $u, r \in \mathbb{Z}_{\geqslant 0}$ *with* $r < m$ *are such that* $n = um + r$*, then with probability at least* $1 - 2nd/|\mathcal{T}|$ *the Popov basis* $\boldsymbol{P} \in \mathbb{F}[X]^{d \times d}$ *of* $\phi_d(\Gamma_d(\mathcal{P}))$ *contains exactly* $d - r$ *rows of degree* u *and* r *rows of degree* $u + 1$*. In particular,* $\deg \boldsymbol{P} \leqslant u + 1$.

Proof. Let $p_1, \dots, p_d \in \mathbb{F}[X, Y]$ be the polynomials given by the rows of $\boldsymbol{P}$ through ϕ_d^{-1}. Due to Lemma 3.6, we know that $\Delta(\boldsymbol{P}) = n = \sum_{i=1}^d \deg_X p_i$.

For any $s \in \mathbb{Z}_{>0}$ let $\boldsymbol{A}_s \in \mathbb{F}[Y_1, \ldots, Y_n]^{n \times ds}$ with $\operatorname{rank}(\boldsymbol{A}_s) = \min\{n, ds\}$ be as in Lemma 3.28, and consider $\boldsymbol{A}_s^{(\boldsymbol{\beta})} \in \mathbb{F}^{n \times ds}$ with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n) \in \mathbb{F}^n$. Choosing $s = u$, if $\deg_X p_i < u$ for some i, then the $\mathbb{F}$-coefficients of p_i form a vector in the right kernel of $\boldsymbol{A}_u^{(\boldsymbol{\beta})}$ (as mentioned above this lemma). Consequently,

$$\operatorname{rank}(\boldsymbol{A}_u^{(\boldsymbol{\beta})}) < \operatorname{rank}(\boldsymbol{A}_u) = du \leqslant n \ ,$$

so letting $M \in \mathbb{F}[Y_1, \ldots, Y_n]$ be a non-zero $du \times du$ minor of $\boldsymbol{A}_u$ we get that

$$M(\beta_1, \ldots, \beta_n) = M(\lambda(\tau_1, \ldots, \tau_n)) = 0.$$

Since M has degree at most $d-1$ in each variable, then the total degree of M is strictly less than nd, and since λ is affine, then the total degree of $M(\lambda(Z_1, \ldots, Z_n)) \in \mathbb{F}[Z_1, \ldots, Z_n]$ satisfies the same upper bound. Therefore, by Lemma 3.27, the probability that $M(\lambda(\tau_1, \ldots, \tau_n)) = 0$ is at most $nd/|\mathcal{T}|$.

Now, assume that all rows of $\boldsymbol{P}$ have degree at least u. Then for each i such that $\deg_X p_i = u$, the $\mathbb{F}$-coefficients of p_i form a vector in the right kernel of $\boldsymbol{A}_{u+1}^{(\boldsymbol{\beta})} \in \mathbb{F}^{n \times d(u+1)}$. By Lemma 3.28, then $\boldsymbol{A}_{u+1}$ has a right kernel over $\mathbb{F}(Y_1, \ldots, Y_n)$ of dimension $d(u+1) - n = d - r$. Since the rows of $\boldsymbol{P}$ are linearly independent over $\mathbb{F}[X]$, and thus also over $\mathbb{F}$, then at most $d-r$ rows of $\boldsymbol{P}$ have degree u whenever $\operatorname{rank}(\boldsymbol{A}_{u+1}^{(\boldsymbol{\beta})}) = \operatorname{rank}(\boldsymbol{A}_{u+1})$. Therefore, considering as before a non-zero $n \times n$ minor $N \in \mathbb{F}[Y_1, \ldots, Y_n]$ of $\boldsymbol{A}_{u+1}$, we conclude that N has total degree less than nd, and that the probability that $N(\beta_1, \ldots, \beta_N) = N(\lambda(\tau_1, \ldots, \tau_n)) = 0$ is at most $nd/|\mathcal{T}|$.

Hence, with probability at least $1 - 2nu/|T|$, $\boldsymbol{P}$ has all rows of degree at least u and j rows of degree exactly u with $j \leqslant d - t$. It follows that each of the remaining $d - j$ rows has degree at least $u + 1$, and the sum of these degrees is

$$n - ju = du + r - jd = (d - j)u + t \leqslant (d - j)(u + 1) \ .$$

But then each of them has degree exactly $u + 1$. □

When using Algorithm 11 to compute a reshaper $g = Y^\eta - \breve{g} \in \Gamma(\mathcal{P})$ with $\deg_Y \breve{g} < d$, we are guaranteed that $\deg_X \breve{g} \leqslant \deg_X \boldsymbol{P}$, where $\boldsymbol{P}$ is the Popov basis of $\Gamma_d(\mathcal{P})$. Lemma 3.29 states that in the generic case we can expect that $\deg \boldsymbol{P} \leqslant \lfloor n/d \rfloor + 1$, which matches the Definition 3.12 if we set $d = 2\eta_i - \eta_{i-1} + 1$ in a degree sequence $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$.

Corollary 3.30. *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ be distinct, $\mathcal{T} \subseteq \mathbb{F}$ a finite subset, and let $\lambda : \mathbb{F}^n \to \mathbb{F}^n$ be an affine map. For $\tau_1, \ldots, \tau_n \in \mathcal{T}$ chosen independently and uniformly at random, let $\mathcal{P} = \{(\alpha_j, \beta_j)\}_{j=1}^n \subset \mathbb{F}^2$ be such that $(\beta_1, \ldots, \beta_n) = \lambda(\tau_1, \ldots, \tau_n)$. If $\boldsymbol{\eta} = (\eta_i)_{i=0}^{\gamma} \in \mathbb{Z}_{>0}^{\gamma+1}$ is a degree sequence with $\eta_0 \leqslant n$, then $\mathcal{P}$ is $\boldsymbol{\eta}$-balanced with probability at least $1 - n^2\gamma/|\mathcal{T}|$.*

Corollary 3.30 directly applies to both MPE and interpolation algorithms on random point sets with unique X-coordinates. In the case of interpolation, we shear the points so that their Y-coordinates become pairwise distinct. Although balancedness is not inherited a priori by the sheared point set, the probability of being balanced is preserved, since shearing acts as an affine transformation on the Y-coordinates. We conclude this chapter by summarizing our probabilistic results:

Corollary 3.31. *Let $d, n \in \mathbb{Z}_{>0}$ with $d \leqslant n$ and assume that $\mathbb{F} = \mathbb{F}_q$. If $\mathcal{P} = \{(\alpha_i, \beta_i)\}_{i=1}^n \subseteq \mathbb{F}_q^2$ is chosen uniformly at random among the point sets with cardinality n, then with probability of at least*

$$\Big(1 - \frac{n^2}{q}\Big)\Big(1 - \frac{3n^2(\log_{3/2}(n) + 1)}{q}\Big)$$

the following two problems can be solved with cost online $\tilde{\mathcal{O}}(n)$:

1) *Given a polynomial $f \in \mathbb{F}[X, Y]$ with $\deg_X f < n/d$ and $\deg_Y f < d$, compute $f(\alpha_1, \beta_1), \ldots, f(\alpha_n, \beta_n) \in \mathbb{F}$.*
2) *Given interpolation values $v_1, \ldots, v_n \in \mathbb{F}$, compute an $f \in \mathbb{F}[X, Y]$ satisfying $f(\alpha_j, \beta_j) = v_j$ for $j = 1, \ldots, n$ as well as $\deg_Y f < d$ and $\deg_X f \leqslant cn$ for some constant c dependent only on n and d.*

Proof. It is clear that there are $\binom{q^2}{n}$ possible point sets in $\mathbb{F}_q^2$ of cardinality n. It is also not hard to see that $\binom{q}{n} q^n$ of them have pairwise distinct X-coordinates: first choose n-distinct X-coordinates and then count the possible Y-coordinates for each choice. It follows that the probability that a random point set $\mathcal{P}$ has pairwise distinct X-coordinates is

$$\begin{aligned}\frac{\left(q^n \frac{q!}{n!(q-n)!}\right)}{\left(\frac{q^2!}{n!(q^2-n)!}\right)} &> \frac{q^n}{q^{2n}} \cdot q(q-1)\cdots(q-(n-1)) \\ &= \prod_{j=0}^{n-1}\Big(1 - \frac{j}{q}\Big) > \Big(1 - \frac{n}{q}\Big)^n \geqslant 1 - \frac{n^2}{q}\ ,\end{aligned}$$

where the latter follows from Bernoulli's inequality. This explains the first factor in the claimed probability bound. For the last factor, recall from Lemma 3.10 that we can always choose $\gamma \leqslant \log_{3/2}(n) + 1$. Therefore, for each of the three balancedness requirements that are jointly posed by 1) and 2), the probability that $\mathcal{P}$ is unbalanced is at most $\frac{n^2(\log_{3/2}(n)+1)}{q}$, as a direct consequence of Corollary 3.30. $\square$

Chapter 4

Partial unique decoding

This chapter is dedicated to decoding of AG codes beyond half the minimum distance. In particular, we will investigate the error-correcting capabilities of *power decoding* – a technique for *partial unique decoding*, meaning that we seek the unique closest codeword to the received word, but that we also allow the decoder to declare *failure*, since unique decoding is not always possible. This contrasts with the setting of *list decoding* (see Chapter 5), where we are tasked with returning a list containing *all* of the viable candidate-codewords.

4.1 Related work

Power decoding was originally proposed in [SSB06] and [SSB10] as a method for decoding RS codes beyond half the minimum distance. Being comparable with Sudan's list decoder [Sud97] in its decoding radius and asymptotic complexity, many of its advantages are to be found in more practical considerations: Power decoding is simpler in that it only requires a solution for a single shift-register type problem; this is in contrast to Sudan's approach, which has an interpolation step as well as a root-finding step. Moreover, this root-finding step – although asymptotically cheaper than interpolation [NRS17] – still carries with it the cost of taking up significant circuit area in hardware-implementations

[AKS11], and in the more general context of interleaved codes it actually becomes the computational bottleneck [CH13], making power decoding asymptotically faster [PR17].

At the heart of power decoding lies a set of highly non-linear "key equations", whose solution yields a so called *error locator*. In the case of RS codes, the error locator is a univariate polynomial over the base field, and as the name suggests, it contains information about where the errors are located in the received word. Depending on the particular choice of key equations, the error positions can be encoded either in the coefficients of the error locator, as was originally proposed in [SSB06], or in its roots, as was implied in [Gao03] and detailed in [Nie14]. In this chapter we consider the latter. In our setting of general AG codes, however, the error locator is not necessarily a univariate polynomial. Instead, it is an element in the function field, and it is required to vanish (possibly with multiplicity) at the places that correspond to the error positions. Following in the footsteps of [PR17] and [PRB19], we also replace the notion of "degree", which plays an important role in key equations for RS codes, with pole order at a place that we denote by P_∞. This way of measuring "size" of function field elements was originally proposed in [PSP92].

Although power decoding differs conceptually from Sudan's decoder, the two approaches share a surprising amount of common features. Besides the mutual resemblance in decoding radius and asymptotic complexity, they have also seen a similar historical development. Only about a year after its conception, Sudan's approach gave rise to the celebrated Guruswami-Sudan (GS) decoder [GS98], which by introducing a *multiplicity parameter* greatly enhanced its predecessor's decoding radius, achieving the so called *Johnson radius*. Furthermore, this new result was applicable not only to RS codes, but also arbitrary AG codes due to [SW99]. Although the maturation process of power decoding has been more incremental than that of Sudan's decoder, its overall trajectory has been nearly identical: First, the so called *improved power decoding* was introduced for RS codes in [Ros18], achieving the Johnson radius – curiously also by the means of introducing a multiplicity parameter. This result was soon shown to also be applicable to one-point Hermitian codes [PRB19], and then the original (non-improved) power decoding was generalized to work for arbitrary AG codes [CP20]. The final step of adapting improved power decoding to this general setting was made in [PRS21], which this chapter is based on.

4.2 Contributions

Having provided some historical context for our contributions to the development of power decoding, let us now discuss their precise scope. In this chapter we:

- formulate improved power decoding in the language of function fields, making it applicable to arbitrary AG codes,
- give an explicit construction of a linear system that can be used to solve the *key equations*, and
- derive the standard *educated guess* for the decoding radius and verify it using Monte-Carlo simulations.

It is also worth mentioning that the decoding radius of our method perfectly coincides with that of [Ros18] and [PRB19] for RS codes and one-point Hermitian codes respectively, and that by achieving the Johnson radius in the context of arbitrary AG codes, we finally close the generality gap with the GS decoder. It should be emphasized, however, that power decoding and GS operate under different notions of decoding radius, making a direct comparison tricky: In contrast to GS, where decoding radius is defined as the greatest number of errors for which decoding is guaranteed to succeed, the decoding radius of power decoding is defined – somewhat counterintuitively – as the greatest number of errors that does not trigger a certain condition that is sufficient for decoding failure. This means that it *is* possible for power decoding to fail even below its decoding radius, however, as evidenced by our simulations, this happens rarely for random errors. We give no theoretic bounds on this failure probability, as they have proven to be difficult to obtain even in the case of RS codes [Ros18], except for a few small choices of decoding parameters [SSB10, Nie14].

Another limitation of our contribution that we ought to be transparent about is that computational complexity is *not* taken into consideration. Our decoder is formulated (and implemented) in terms of linear algebra, which is undeniably inefficient. That being said, this formulation is particularly well suited for our primary aim: to show that improved power decoding achieves the Johnson radius for AG codes. Although obtaining an efficient algorithm remains an open problem, it is likely within reach: There are good reasons to believe that the linear algebra part that constitutes the computational bottleneck of our decoder can be replaced with more sophisticated techniques such as efficient *simultaneous Hermite Padé approximations* [RS19], following in the footsteps of [Ros18] and [PBR17] for RS codes and one-point Hermitian codes respectively. Indeed, we will use this extensively in Chapter 5, where efficiency is within the scope.

4.3 Constructing the key equations

In this section we formally introduce the *key equations* for the code

$$\mathcal{C}_{\mathcal{L}}(D,G) = \{\big(f(P_1),\dots,f(P_n)\big) \in \mathbb{F}^n \mid f \in \mathcal{L}(G)\} \ ,$$

where $P_1,\dots,P_n$ are rational places, and G as well as $D = P_1 + \cdots + P_n$ are divisors with $\operatorname{supp} D \cap \operatorname{supp} G = \varnothing$. We begin by defining the terms involved. For every divisor A we will use the shorthand notation $\text{Я}(A) = \text{Я}_{P_\infty}(A)$, where $\text{Я}_P(A)$ is as in Definition 1.14 on page 13. Similarly, for any function $a \in \text{Я}(A)$, we will write $\delta_A(a) = \delta_A^{(P_\infty)}(a)$.

Definition 4.1. If $\boldsymbol{r} = (r_1,\dots,r_n) \in \mathbb{F}^n$ is the received word, and $R \in \text{Я}(G)$ with $R(P_i) = r_i$ for $i = 1,\dots,n$, then we say that R is an $\boldsymbol{r}$*-interpolator*.

Lemma 4.2. *For any received word $\boldsymbol{r} \in \mathbb{F}^n$, there always exists an $\boldsymbol{r}$-interpolator $R \in \text{Я}(G)$ with $\delta_G(R) \leqslant n - \deg G + 2g - 1$.*

Definition 4.3. Let $\boldsymbol{c} \in \mathbb{F}^n$ and $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ be the sent codeword and the received word respectively, where $\boldsymbol{e} = (e_1,\dots,e_n) \in \mathbb{F}^n$ is some error vector. If $E = \sum_{i\in\mathcal{E}} P_i$, where $\mathcal{E} = \{i \mid e_i \neq 0\}$ and $0 \neq \Lambda_s \in \text{Я}(-sE) \subseteq \text{Я}$ for some $s \in \mathbb{Z}_{>0}$, then we say that Λ_s is an *error locator* with multiplicity s.

In the context of Definition 4.3, note that $\delta(\Lambda_s) = \delta_{-sE}(\Lambda_s)$ since $v_{P_\infty}(E) = 0$.

Lemma 4.4. *In the context of Definition 4.3, there exists an error locator $\Lambda_s \in \text{Я}$ with multiplicity s satisfying $\delta(\Lambda_s) \leqslant s|\mathcal{E}| + g$.*

Proof. By the Riemann-Roch theorem

$$\ell\big((s|\mathcal{E}| + g)P_\infty - sE\big) \geqslant (s|\mathcal{E}| + g - s|\mathcal{E}|) - g + 1 = 1 \ ,$$

which guarantees that

$$\{\Lambda_s \in \text{Я}(-sE) \mid \delta_{-sE}(\Lambda_s) \leqslant s|\mathcal{E}| + g\} = \mathcal{L}\big((s|\mathcal{E}| + g)P_\infty - sE\big) \neq \{0\} \ .$$

□

For the remainder of this chapter, fix two parameters $\rho, \lambda_s, \in \mathbb{Z}_{\geqslant 0}$. Letting $\tau \geqslant |\mathcal{E}|$ be the greatest number of errors that we hope to correct, we will typically set $\rho = n - \deg G + 2g - 1$ and $\lambda_s = s\tau + g$ in accordance with Definition 4.1 and Definition 4.3 respectively, so that we can rely on existence

of an $\boldsymbol{r}$-interpolator R and an error locator Λ_s with $\delta_G(R) \leqslant \rho$ and $\delta(\Lambda_s) \leqslant \lambda_s$. For the – now fixed – values of ρ and λ_s, we also fix the divisors

$$\begin{aligned} V_t &= \lambda_s P_\infty + tG \ , \\ Q_t &= \lambda_s P_\infty + t(G + \rho P_\infty) \text{ and} \\ W_j &= \lambda_s P_\infty + j(G + \rho P_\infty - D) \end{aligned}$$

for $t = 1, \ldots, \ell$ and $j = 0, \ldots, s-1$. The following purely technical lemma relates these divisors with R, Λ_s and the sent message, and it will be instrumental in proving the validity of the key equations.

Lemma 4.5. *If $f \in \mathcal{L}(G)$ is the sent message, $R \in \text{Я}(G)$ is an $\boldsymbol{r}$-interpolator with $\delta_G(R) \leqslant \rho$, and $\Lambda_s \in \text{Я}$ is an error locator having multiplicity s and $\delta(\Lambda_s) \leqslant \lambda_s$, then for $t = 1, \ldots, \ell$ and $j = 0, \ldots, s-1$ it holds that*

$$\Lambda_s f^t \in \mathcal{L}(V_t) \quad \textit{and} \quad \Lambda_s (f - R)^j R^{t-j} \in \begin{cases} \mathcal{L}\big(W_j + (t-j)(G + \rho P_\infty)\big) & \textit{if } j < s \\ \mathcal{L}(Q_t - sD) & \textit{if } j \geqslant s \end{cases} .$$

Proof. Since $\delta_{tG}(f^t) = t\delta_G(f) = 0$, then

$$\delta_{tG}(\Lambda_s f^t) = \delta(\Lambda_s) + \delta_{tG}(f^t) \leqslant \lambda_s \ ,$$

which proves the first claim. For the second claim, note that

$$f - R \in \mathcal{L}\big(G + \rho P_\infty - (D - E)\big) \ ,$$

which implies that for $j < s$

$$\begin{aligned} \Lambda_s (f - R)^j &\in \mathcal{L}\big((\lambda_s P_\infty - sE) + j(G + \rho P_\infty - D + E)\big) \\ &= \mathcal{L}\big(\lambda_s P_\infty + j(G + \rho P_\infty - D) - (s - j)E\big) \\ &\subseteq \mathcal{L}(W_j) \ , \end{aligned}$$

and that for $j \geqslant s$

$$\begin{aligned} \Lambda_s (f - R)^j R^{t-j} &\in \mathcal{L}\big((\lambda_s P_\infty - sE) + j(G + \rho P_\infty - D + E) + (t - j)(\rho P_\infty + G)\big) \\ &= \mathcal{L}\big(\lambda_s P_\infty - sE + t(G + \rho P_\infty) - j(D - E)\big) \\ &\subseteq \mathcal{L}\big(\lambda_s P_\infty - sE + t(G + \rho P_\infty) - s(D - E)\big) \qquad (*) \\ &= \mathcal{L}\big(\lambda_s P_\infty + t(G + \rho P_\infty) - sD\big) \\ &= \mathcal{L}(Q_t - sD) \ , \end{aligned}$$

where $(*)$ is a consequence of $D - E$ being effective. □

Finally, we are ready to introduce the key equations:

Theorem 4.6 (The key equations)**.** *If $f \in \mathcal{L}(G)$ is the sent message, $R \in \mathcal{L}(\rho P_\infty + G)$ is an $\boldsymbol{r}$-interpolator and $\Lambda_s \in \mathcal{L}(\lambda P_\infty)$ is an error locator with multiplicity s, then for $t = 1, \ldots, \ell$*

$$\Lambda_s f^t - \sum_{j=0}^{\min\{t,s-1\}} \binom{t}{j} \Lambda_s (f-R)^j R^{t-j} \in \begin{cases} \{0\} & \text{if } 1 \leqslant t \leqslant s-1 \\ \mathcal{L}(Q_t - sD) & \text{if } s \leqslant t \leqslant \ell \end{cases} . \quad (4.1)$$

Proof. Observing that

$$\Lambda_s f^t = \Lambda_s \big((f-R)+R\big)^t = \sum_{j=0}^{t} \binom{t}{j} \Lambda_s (f-R)^j R^{t-j}$$

immediately proves the claim for $1 \leqslant t \leqslant s-1$. If $s \leqslant t \leqslant \ell$, then it follows from Lemma 4.5 that

$$\begin{aligned} \Lambda_s f^t - \sum_{j=0}^{\min\{t,s-1\}} \binom{t}{j} \Lambda_s (f-R)^j R^{t-j} &= \sum_{j=s}^{t} \binom{t}{j} \Lambda_s (f-R)^j R^{t-j} \\ &\in \mathcal{L}(Q_t - sD) . \end{aligned}$$

□

In the context of Theorem 4.6, if we are given the received word $\boldsymbol{r}$, then we can easily compute an $\boldsymbol{r}$-interpolator $R \in \mathcal{L}(\rho P_\infty + G)$ e.g. by using linear algebra (see Section 4.4). The unknowns in the key equations (4.1) are therefore f and Λ_s. If these equations admit a unique solution (up to $\mathbb{F}$-scaling of Λ_s), then decoding simply reduces to finding it – however, it is not immediately clear how to accomplish this directly. In Section 4.4, we use the language of function fields to formulate the usual trick of *linearization*, which will often allow us to successfully deal with this otherwise highly nonlinear problem.

4.4 Solving the key equations

In this section we explain how the key equations (4.1) can be solved by strategically "forgetting" some of their original algebraic structure in such a way as to make them linear over $\mathbb{F}$. This trick incurs the price of potentially "polluting" the solution space with new elements that could get in the way of us finding the hopefully unique solution of the original problem, however, previous work on power decoding as well as our simulations (see Section 4.6) indicate that this happens with small probability. Without further ado, we present the linearized problem below.

Problem 4.7. Given an $\boldsymbol{r}$-interpolator $R \in \mathcal{L}(\rho P_\infty + G)$, find $\phi_t \in \mathcal{L}(V_t)$ for $t = 1, \ldots, \ell$ and $\psi_j \in \mathcal{L}(W_j)$ for $j = 0, \ldots, s-1$ (not all zero) such that

$$\phi_t - \sum_{j=0}^{\min\{t,s-1\}} \binom{t}{j} \psi_j R^{t-j} \in \begin{cases} \{0\} & \text{if } 1 \leqslant t \leqslant s-1 \\ \mathcal{L}(Q_t - sD) & \text{if } s \leqslant t \leqslant \ell \end{cases} . \tag{4.2}$$

It is not hard to see the resemblance between (4.1) and (4.2) if we "squint our eyes" a little. Indeed, it is a direct consequence of Theorem 4.6 that any solution f, Λ_s of (4.1) gives rise to a solution of (4.2) of the form

$$\phi_t = \Lambda_s f^t \quad \text{and} \quad \psi_j = \Lambda_s (f - R)^j \tag{4.3}$$

for $t = 1, \ldots, \ell$ and $j = 0, \ldots, s-1$. Of course, the converse does not always hold, which means that a solution of the latter might fail to be a solution of the former. Foreshadowing our discussion in Section 4.5, it is exactly this phenomenon that defines the decoding radius of power decoding.

Our decoding strategy thus becomes as follows: Use linear algebra to find an $\mathbb{F}$-basis of the solution space of Problem 4.7. If this basis only has one element, then this element is necessarily of the form (4.2) (up to $\mathbb{F}$-scaling). The message f that corresponds to the closest codeword $\boldsymbol{c}$ of the received word $\boldsymbol{r}$ can thus be obtained by e.g the division

$$\frac{\phi_1}{\psi_0} = \frac{\Lambda_s f^t}{\Lambda_s} = f \ . \tag{4.4}$$

Having outlined our plan of action, we now proceed to explaining how to translate our AG setting into raw linear algebra over $\mathbb{F}$. To be more precise, we give an explicit construction of a matrix $\boldsymbol{U}$ whose right kernel describes the solution space of Problem 4.7. Besides aiding implementation, having a description of this matrix – in particular its dimensions – will allow us to derive the decoding radius of our decoder.

Constructing the linear system

We begin by establishing some notation: For any divisor A, let $\boldsymbol{\beta}_A \in \mathcal{L}(A)^{\mathfrak{l}(A)}$ denote a vector whose entries form *any* $\mathbb{F}$-basis of $\mathcal{L}(A)$ in such a way as to ensure that for any divisor B it holds that

$$\mathcal{L}(A) = \mathcal{L}(B) \iff \boldsymbol{\beta}_A = \boldsymbol{\beta}_B \ .$$

In other words, the map $A \mapsto \boldsymbol{\beta}_A$ should associate to every Riemann-Roch space some "canonical" $\mathbb{F}$-basis. Continuing, for any $h \in \mathcal{L}(A)$ let $h_A \in \mathbb{F}^{\mathfrak{l}(A)}$ denote the *unique* vector for which $\boldsymbol{\beta}_A \cdot h_A = h$, i.e. writing $\boldsymbol{\beta}_A = (a_1, \ldots, a_{\mathfrak{l}(A)})$ and $h = \sum_{i=1}^{\mathfrak{l}(A)} c_i a_i$, where $c_i \in \mathbb{F}$, we get that $h_A = (c_1, \ldots, c_{\mathfrak{l}(A)})$. Finally, for any $p \in \mathcal{L}(B-A)$, let $p_{B,A} \in \mathbb{F}^{\mathfrak{l}(B)\times\mathfrak{l}(A)}$ denote the matrix whose i-th column is given by the vector $(pa_i)_B$. The following simple lemma will be useful for proving the correctness of our linear system, as it allows us to express products of function field elements using linear algebra:

Lemma 4.8. *If A and B are divisors, then for any $h \in \mathcal{L}(A)$ and $p \in \mathcal{L}(B-A)$*

$$\boldsymbol{\beta}_B p_{B,A} h_A = ph \in \mathcal{L}(B) \ .$$

Proof. Writing $\boldsymbol{\beta}_A = (a_1, \ldots, a_{\mathfrak{l}(A)})$ and noting that $pa_i \in \mathcal{L}(B)$ for $i = 1, \ldots, \mathfrak{l}(A)$, it becomes clear that

$$\boldsymbol{\beta}_B p_{B,A} = \boldsymbol{\beta}_B [(pa_i)_B]_{i=1}^{\mathfrak{l}(A)} = p[\boldsymbol{\beta}_B \cdot (a_i)_B]_{i=1}^{\mathfrak{l}(A)} = p[a_i]_{i=1}^{\mathfrak{l}(A)} = p\boldsymbol{\beta}_A \ ,$$

where the $(pa_i)_B$ are treated as column vectors. The conclusion follows, since

$$\boldsymbol{\beta}_B p_{B,A} h_A = p\boldsymbol{\beta}_A \cdot h_A = ph \ .$$

□

As Problem 4.7 requires us to have an $\boldsymbol{r}$-interpolator $R \in \mathcal{L}(H)$, where $\boldsymbol{r} \in \mathbb{F}^n$ is the received word and $H = \rho P_\infty + G$, let us briefly mention that it can be obtained as $R = \boldsymbol{\beta}_H \cdot \boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{F}^{\mathfrak{l}(H)}$ is any solution to the linear system $\boldsymbol{M}\boldsymbol{v} = \boldsymbol{r}$ with $\boldsymbol{M} \in \mathbb{F}^{n\times\mathfrak{l}(H)}$ being the matrix whose i-th column is $[b_i(P_1) \cdots b_i(P_n)]^\top$ and $\boldsymbol{\beta}_H = (b_1, \ldots, b_{\mathfrak{l}(H)})$.

The following lemma and its subsequent Corollary 4.10 give an explicit construction of the aforementioned matrix $\boldsymbol{U}$ whose right kernel fully describes the solution space of Problem 4.7.

Lemma 4.9. *Let $\phi_r \in \mathcal{L}(V_r)$ and $\psi_j \in \mathcal{L}(W_j)$ for $r = 1, \ldots, \ell$ and $j = 0, \ldots, s-1$ be functions, and define the vector*

$$\boldsymbol{u} = \big((\phi_1)_{V_1} | \ldots | (\phi_\ell)_{V_\ell} | (\psi_0)_{W_0} | \ldots | (\psi_{s-1})_{W_{s-1}}\big) \in \mathbb{F}^\nu \ ,$$

where $\nu = \sum_{r=1}^{\ell} \mathfrak{l}(V_r) + \sum_{j=0}^{s-1} \mathfrak{l}(W_j)$. Furthermore, for $t = 1, \ldots, \ell$ let

$$\boldsymbol{U}_t = [\boldsymbol{V}_{t,1} | \cdots | \boldsymbol{V}_{t,\ell} | \boldsymbol{W}_{t,0} | \cdots | \boldsymbol{W}_{t,s-1}] \in \mathbb{F}^{\mathfrak{l}(Q_t)\times\nu}$$

be the matrix where

$$\boldsymbol{V}_{t,r} = \begin{cases} 1_{Q_t,V_r} & \text{if } t = r \\ 0_{Q_t,V_r} & \text{if } t \neq r \end{cases} \quad \text{and} \quad \boldsymbol{W}_{t,j} = \begin{cases} -\binom{t}{j}(R^{t-j})_{Q_t,W_j} & \text{if } 1 \leqslant t \leqslant s-1 \\ 0_{Q_t,W_j} & \text{if } j > t \end{cases} \ .$$

Then the ϕ_r and ψ_j constitute a solution to Problem 4.7 if and only if

$$\boldsymbol{\beta}_{Q_t}\boldsymbol{U}_t\boldsymbol{u} \in \begin{cases} \{0\} & \text{if } 1 \leqslant t \leqslant s-1 \\ \mathcal{L}(Q_t - sD) & \text{if } s \leqslant t \leqslant \ell \end{cases} .$$

Proof. The claim follows immediately from the identity

$$\begin{aligned} \boldsymbol{\beta}_{Q_t}\boldsymbol{U}_t\boldsymbol{u} &= \boldsymbol{\beta}_{Q_t}\sum_{r=1}^{\ell}\boldsymbol{V}_{t,r}(\phi_t)_{V_t} + \boldsymbol{\beta}_{Q_t}\sum_{j=0}^{s-1}\boldsymbol{W}_{t,j}(\psi_j)_{W_j} \\ &= \boldsymbol{\beta}_{Q_t}1_{Q_t,V_t}(\phi_t)_{V_t} - \boldsymbol{\beta}_{Q_t}\sum_{j=0}^{\min\{t,s-1\}}\binom{t}{j}(R^{t-j})_{Q_t,W_j}(\psi_j)_{W_j} \\ &= \phi_t - \sum_{j=0}^{\min\{t,s-1\}}\binom{t}{j}\psi_t R^{t-j} , \end{aligned}$$

where the last equality is due to Lemma 4.8. □

Corollary 4.10. *In the context of Lemma 4.9, if*

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{K}_1\boldsymbol{U}_1 \\ \hline \vdots \\ \hline \boldsymbol{K}_\ell\boldsymbol{U}_\ell \end{bmatrix}$$

is the matrix where

$$\boldsymbol{K}_t = \begin{cases} 1_{Q_t,Q_t} & \text{if } 1 \leqslant t \leqslant s-1 \\ \text{left kernel matrix of } 1_{Q_t,Q_t-sD} & \text{if } s \leqslant t \leqslant \ell \end{cases} ,$$

then the ϕ_r and ψ_j constitute a solution to Problem 4.7 if and only if $\boldsymbol{U}\boldsymbol{u} = \boldsymbol{0}$.

Proof. Due to Lemma 4.9, it suffices to show that for $s \leqslant t \leqslant \ell$

$$\boldsymbol{\beta}_{Q_t}\boldsymbol{U}_t\boldsymbol{u} \in \mathcal{L}(Q_t - sD) \iff \boldsymbol{K}_t\boldsymbol{U}_t\boldsymbol{u} = \boldsymbol{0} .$$

But this follows immediately from the observation that

$$\boldsymbol{\beta}_{Q_t}\boldsymbol{U}_t\boldsymbol{u} \in \mathcal{L}(Q_t - sD) \iff \boldsymbol{U}_t\boldsymbol{u} = 1_{Q_t,Q_t-sD}(\boldsymbol{\beta}_{Q_t}\boldsymbol{U}_t\boldsymbol{u})_{Q_t-sD} .$$

□

For clarity, we the present the outlined decoder in Algorithm 12.

Algorithm 12 LinearAlgebraDecoder$(\boldsymbol{r}, \ell, s, D, G)$

Input:

- The received word $\boldsymbol{r} \in \mathbb{F}^n$,
- decoding parameters $\ell, s \in \mathbb{Z}_{>0}$ with $s \leqslant \ell$,
- divisors D and G for the code $\mathcal{C}_{\mathcal{L}}(D, G)$.

Output:

- "Fail" or $f \in \mathcal{L}(G)$ such that $\boldsymbol{c} = (f(P_1), \ldots, f(P_n)) \in \mathcal{C}_{\mathcal{L}}(D, G)$ is the closest codeword to $\boldsymbol{r}$ (in Hamming distance).

1: $\rho \leftarrow n - \deg G + 2g - 1$
2: **for** $\lambda_s = 0, \ldots, sn + g$ **do**
3: $\quad \boldsymbol{U} \in \mathbb{F}^{\epsilon \times \nu} \leftarrow$ the matrix from Corollary 4.10
4: $\quad \boldsymbol{V} \in \mathbb{F}^{\nu \times \kappa} \leftarrow$ the right kernel matrix of $\boldsymbol{U}$
5: $\quad$ **if** $\kappa = 1$ **then**
6: $\quad\quad \boldsymbol{u} = \big((\phi_1)_{V_1} | \ldots | (\phi_\ell)_{V_\ell} | (\psi_0)_{W_0} | \ldots | (\psi_{s-1})_{W_{s-1}}\big) \in \mathbb{F}^\nu \leftarrow$ the unique column of $\boldsymbol{V}$
7: $\quad\quad f \leftarrow \frac{\phi_1}{\psi_0}$
8: $\quad\quad$ **if** $f \in \mathcal{L}(G)$ **then return** f
9: $\quad\quad$ **else return** "Fail"
10: $\quad$ **else if** $\kappa \geqslant 2$ **then**
11: $\quad\quad$ **return** "Fail"

Remark 4.11. A computationally minded reader will notice that the for-loop in line 2 of Algorithm 12 introduces a seemingly unnecessary factor of $s|\mathcal{E}| + g$ into the complexity. We comment on this despite our repeated disclaimer that efficiency falls outside this chapter's scope. The ultimate purpose of the loop is to solve the key equations (4.1) for an error locator Λ_s with minimal $\delta(\Lambda_s)$. Not considering more sophisticated techniques than linear algebra, we can easily replace the aforementioned factor $s|\mathcal{E}| + g$ with $\log(sn + g)$ simply by swapping the for-loop with a binary search. Moreover, we can even get rid of this factor altogether by fixing $\lambda_s = s\tau + g$, where $\tau \geqslant |\mathcal{E}|$ bounds the number of errors (see Lemma 4.4). At first sight, doing this potentially breaks solution-uniqueness of (4.1), since it allows for multiple linearly independent error locators to exist when $|\mathcal{E}| < \tau$. In practice, however, this is not a problem as long as τ does not exceed the decoding radius, since the corresponding messages for all of these error locators will be identical – at least with high probability; our simulations rely on this in order to improve performance. The reason for why we stick with the for-loop in Algorithm 12, however, is that this formulation makes it easier to reason about the decoding radius.

Remark 4.12. When it comes to solving the linearized key equations (4.2)

efficiently, then it is likely possible to take advantage of *simultaneous Hermite-Padé approximations*; see Theorem 5.25 on page 102.

Now that we have translated Problem 4.7 into raw linear algebra, we are ready to turn our attention to the subtle topic of decoding radius.

4.5 Decoding radius

This section is dedicated to a theoretical investigation of the error-correction capabilities of our decoder. As we will see, however, theory will only get us so far, as it is not even clear how to define decoding radius when we are attempting *unique decoding beyond half the minimum distance*. Indeed, this is an oxymoron, which is why we are forced to settle with *partial* unique decoding, allowing declaration of failure in cases where unique decoding is impossible. But doing this implies that the intuitive notion of decoding radius, i.e. the greatest number of errors for which the decoder is guaranteed to succeed, is necessarily bounded by half the minimum distance, and this fails to capture some of the practically interesting features of actual decoding performance. Indeed, all experiments indicate that for random errors, decoding beyond half the minimum distance succeeds with very high probability – up to a certain limit, of course. Above this limit, the apparent probability of success does not slowly taper off, but instead, it abruptly drops to zero (or almost zero, depending on the formulation of the decoder). From a practical perspective, therefore, it would have made a lot of sense to define the decoding radius by this very drop, but unfortunately, the underlying probability distribution remains too poorly understood for that.[1]

At this point, it might seem that the only way to estimate the decoding performance of power decoding is by means of numerical simulation, however, this is not true in practice. An *educated guess* – one that is yet to be experimentally refuted – can be obtained from a fairly straightforward investigation of the following *sufficient condition for decoding failure*:

$$\nu > \epsilon + 1 \ , \tag{4.5}$$

where ν is the number of variables in the underlying linear system and ϵ is the number of equations. Indeed, when (4.5) is satisfied, then the system is guaranteed to have at least two linearly independent solutions, and this prompts the decoder to declare failure. It ought to be emphasized, however, that the decoding radius – being nothing but an educated guess derived from (4.5) – provides no theoretical guarantees for decoding success whatsoever; and yet, due to its

[1] except for partial results in the case of RS codes [SSB10, Nie14]

repeated experimental success [SSB06, SSB10, Ros18, PBR17, PRB19, PRS21], it has undeniable practical significance. Having hopefully shed some light on the subtle nature of decoding radius, let us without further ado proceed to an investigation of (4.5) from a technical point of view.

Throughout this section, we will treat the matrix $\boldsymbol{U}$ from Corollary 4.10 – the one that defines our linear system – as fixed, and we will denote its row and column dimensions by ϵ and ν respectively, so that $\boldsymbol{U} \in \mathbb{F}^{\epsilon \times \nu}$. In the following lemma, we obtain estimates on ϵ and ν by considering the dimensions of the Riemann-Roch spaces involved in Problem 4.7. For ease of notation, we introduce the shorthand $\gamma = \deg G$.

Lemma 4.13. *It holds that*

$$\nu \geqslant (\ell + s)(\lambda_s - g + 1) + \tfrac{\ell(\ell+1)}{2}\gamma + \tfrac{s(s-1)}{2}(\gamma + \rho - n) \ .$$

Furthermore, if $\gamma \geqslant 2g - 2$, then

$$\epsilon \leqslant (s-1)(\lambda_s - g + 1) + \tfrac{s(s-1)}{2}(\gamma + \rho) + (\ell - s + 1)sn \ .$$

Proof. Recall from Section 4.3 the divisors

$$\begin{aligned} V_t &= \lambda_s P_\infty + tG \ , \\ Q_t &= \lambda_s P_\infty + t(G + \rho P_\infty) \text{ and} \\ W_j &= \lambda_s P_\infty + j(G + \rho P_\infty - D) \end{aligned}$$

for $t = 1, \ldots, \ell$ and $j = 0, \ldots, s-1$, and note that by the Riemann-Roch theorem

$$\begin{aligned} \mathfrak{l}(V_t) &\geqslant \lambda_s + t - g + 1 \ , \\ \mathfrak{l}(W_j) &\geqslant \lambda + j(\gamma + \rho - n) - g + 1 \quad \text{and} \\ \mathfrak{l}(Q_t - sD) &\geqslant \lambda + t(\gamma + \rho) - sn - g + 1 \ . \end{aligned}$$

Note also that if $\gamma \geqslant 2g - 2$, then by [Sti09, Theorem 1.5.17] we have that

$$\mathfrak{l}(Q_t) = \lambda_s + t(\gamma + \rho) - g + 1 \ .$$

It follows that

$$\begin{aligned} \nu &= \sum_{t=1}^{\ell} \mathfrak{l}(V_t) + \sum_{j=0}^{s-1} \mathfrak{l}(W_j) \\ &\geqslant \sum_{t=1}^{\ell} (\lambda + t\gamma - g + 1) + \sum_{j=0}^{s-1} (\lambda_s + j(\gamma + \rho - n) - g + 1) \\ &= \ell(\lambda_s - g + 1) + \tfrac{\ell(\ell+1)}{2}\gamma + s(\lambda_s - g + 1) + \tfrac{(s-1)s}{2}(\gamma + \rho - n) \ . \end{aligned}$$

Furthermore, the rank-nullity theorem implies that the matrix $\boldsymbol{K}_t$ from Corollary 4.10 has exactly $\mathfrak{l}(Q_t) - \mathfrak{l}(Q_t - sD)$ rows, from which it follows that

$$\begin{aligned}\epsilon &= \sum_{t=1}^{s-1} \mathfrak{l}(Q_t) + \sum_{t=s}^{\ell} \big(\mathfrak{l}(Q_t) - \mathfrak{l}(Q_t - sD)\big) \\ &\leqslant \sum_{t=1}^{s-1} (\lambda_s + t(\gamma+\rho) - g + 1) + \sum_{t=s}^{\ell} sn \\ &= (s-1)(\lambda_s - g + 1) + \tfrac{(s-1)s}{2}(\gamma+\rho) + (\ell - s + 1)sn \ .\end{aligned}$$

□

It is now straightforward to obtain a sufficient condition for decoding failure:

Lemma 4.14. *If* $\lambda_s > \frac{2\ell+1-s}{2(\ell+1)}sn - \frac{\ell}{2}\gamma - \frac{\ell}{\ell+1} + g$, *then* $\nu > \epsilon + 1$.

Proof. Lemma 4.13 implies that $\nu > \epsilon + 1$ is necessitated by

$$\begin{aligned}&(\ell+s)(\lambda_s - g + 1) + \tfrac{\ell(\ell+1)}{2}\gamma + \tfrac{s(s-1)}{2}(\gamma + \rho - n) \\ >&(s-1)(\lambda_s - g + 1) + \tfrac{s(s-1)}{2}(\gamma+\rho) + (\ell - s + 1)sn + 1 \ ,\end{aligned}$$

which is equivalent to

$$\begin{aligned}(\ell+1)(\lambda_s - g + 1) &> (\ell - s + 1)sn + \tfrac{s(s-1)}{2}n - \tfrac{\ell(\ell+1)}{2}\gamma + 1 \ , \\ (\ell+1)(\lambda_s - g + 1) &> \tfrac{2\ell+1-s}{2}sn - \tfrac{\ell(\ell+1)}{2}\gamma + 1 \ , \\ \lambda_s &> \tfrac{2\ell+1-s}{2(\ell+1)}sn - \tfrac{\ell}{2}\gamma - \tfrac{\ell}{\ell+1} + g \ .\end{aligned}$$

□

Prompted by Lemma 4.14, we now take a "leap of faith" in defining the decoding radius as

$$\tau_{\mathsf{max}}(\ell, s) = \left\lfloor \tfrac{2\ell+1-s}{2(\ell+1)}n - \tfrac{\ell}{2s}\gamma - \tfrac{\ell}{s(\ell+1)} \right\rfloor \ , \tag{4.6}$$

with the underlying reasoning being as follows: According to Lemma 4.4, in order for us to be able to correct up to τ errors, we should set $\lambda_s \geqslant s\tau + g$; otherwise, it could happen that no error locator $\Lambda_s \in \mathcal{L}(\lambda_s P_\infty)$ exists, which would result in the key equations (4.1) having no appropriate solutions. Our particular choice of τ_{max} is, therefore, the greatest number of errors for which:

1) the key equations necessarily have at least one solution, while

2) the sufficient condition for decoding failure from Lemma 4.14 is not met.

Although the conditions above are necessary for successful decoding, they are not a priori sufficient. Notwithstanding, as we will see in Section 4.6, the decoding radius from (4.6) is in perfect agreement with our simulation results. Moreover, it coincides with that of [Ros18] and [PRB19] for RS codes and one-point Hermitian codes respectively. We conclude this section with a result that shows how to choose the decoding parameters ℓ and s in order to reach the Johnson radius (asymptotically).

Theorem 4.15 (Theorem 5, [PRB19])**.** *If $(\ell_i, s_i)_{i\in\mathbb{Z}_{>0}}$ is the sequence where $\ell_i = i$ and $s_i = \lfloor\sqrt{\frac{\gamma}{n}}i\rfloor + 1$, then $\tau_{\mathsf{max}}(\ell_i, s_i) = n\left(1 - \sqrt{\frac{\gamma}{n}} - \mathcal{O}(\frac{1}{i})\right)$ as $i \to \infty$.*

Proof. Simply observe that

$$\begin{aligned}\frac{\tau_{\mathsf{max}}(\ell_i,s_i)+1}{n} &> \frac{2\ell_i+1-s_i}{2(\ell_i+1)} - \frac{\ell_i}{2s_i}\frac{\gamma}{n} - \frac{\ell_i}{s_i n(\ell_i+1)}\\ &= \frac{2(\ell_i+1)-(s_i+1)}{2(\ell_i+1)} - \frac{\ell_i}{2s_i}\frac{\gamma}{n} - \frac{\ell_i}{s_i n(\ell_i+1)}\\ &> 1 - \underbrace{\frac{(s_i+1)}{2(\ell_i+1)}}_{\mathcal{O}(\frac{1}{i})} - \underbrace{\frac{\ell_i}{2s_i}\frac{\gamma}{n}}_{<\sqrt{\frac{\gamma}{n}}} - \underbrace{\frac{\ell_i}{s_i n(\ell_i+1)}}_{\mathcal{O}(\frac{1}{i})}.\end{aligned}$$

where we have assumed that $\gamma < n$. □

4.6 Simulation results

In this section we present results of Monte-Carlo simulations that were conducted in order to verify the decoding radius as hypothesized in (4.6). The exact nature of each iteration in these simulations was as follows:

1 fix parameters $\ell, s, \tau \in \mathbb{Z}_{>0}$, and set $\rho = n - \deg G + 2g - 1$ and $\lambda_s = s\tau + g$;

2 pick a random message $f \in \mathcal{L}(G)$ and compute its corresponding codeword $\boldsymbol{c} = (f(P_1), \ldots, f(P_n)) \in \mathbb{F}^n$;

3 generate a random error $\boldsymbol{e} \in \mathbb{F}^n$ of Hamming weight exactly τ and compute the received word $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$;

4 construct the matrix $\boldsymbol{U} \in \mathbb{F}^{\epsilon\times\nu}$ from Corollary 4.10 and compute its right kernel matrix $\boldsymbol{V} \in \mathbb{F}^{\nu\times\kappa}$;

5 recover the candidate message $\widehat{f}$ from the first column of $\boldsymbol{V}$ using (4.4);

6 if $\widehat{f} = f$ declare *success*; otherwise, declare *failure*.

Remark 4.16. The observant reader might complain that the failure criterion in the simulations described above differs from what was used in the derivation of decoding radius – indeed, we have previously stated that decoding failure is to be declared when the right kernel of $\boldsymbol{U}$ fails to have dimension one. This discrepancy is justified in Remark 4.11.

In our simulations we considered a few one-point and two-point codes over three well-known function fields:

- the Hermitian function field, which is defined over $\mathbb{F}_{q^2}$ by the equation $H_q : y^q + y = x^{q+1}$, has genus $g = \frac{1}{2}q(q-1)$ and $q^3 + 1$ rational places;
- the Suzuki function field, which is defined over $\mathbb{F}_{q^4}$ by the equation $S_q : y^q + y = x^{q_0}(x^2 + x)$, where $q = 2q_0^2 > 2$, has genus $g = q_0(q-1)$ and $q^2 + 1$ rational places; and
- the function field from [GS96, Lemma 3.2], which is defined over $\mathbb{F}_{q^2}$ by the equation $T_q : y^q + y = \frac{x^q}{x^{q-1}+1}$, has genus $g = (q-1)^2$ and $q^3 - q^2 + 2q$ rational places.

The simulation results are presented in Table 4.1.

Table 4.1: Simulation results

Curve	$\lvert\mathbb{F}\rvert$	γ	n	k	d^*	ℓ	s	τ	OFR	$N \geqslant$
H_4	4^2	15	64	10	49	4	2	29^+	0.00	10^2
								30	1.00	10^2
H_4	4^2	$10+5$	63	10	48	4	2	28^+	0.00	10^2
								29	1.00	10^2
H_5	5^2	55	125	46	70	5	2	36^+	0.00	10^2
								37	1.00	10^2
H_5	5^2	$30+25$	124	46	69	3	2	35^+	0.00	10^2
								36	0.94	10^2
S_1	2^4	12	24	12	12	2	2	5^+	0.00	10^4
								6	> 0.99	10^4
S_1	2^4	$6+6$	23	12	11	2	2	5^+	0.00	10^4
								6	1.00	10^4
S_1	2^4	4	24	4	20	6	2	12^+	$< 6.98 \cdot 10^{-4}$	10^3
								13	1.00	10^3
S_1	2^4	$2+2$	23	4	19	6	2	11^+	0.00	10^3
								12	> 0.99	10^3
T_4	4^2	15	55	7	40	4	2	23^+	$< 1.74 \cdot 10^{-3}$	10^3
								24	1.00	10^3

Code parameters γ, n, k, d^*. $\gamma = a + b$ means that $G = aP + bP'$ for some rational places P and P'. Decoder parameters ℓ, s. Number of errors τ, $^+$ means that $\tau = \tau_{\max}$. Observed failure rate OFR. Each simulation was repeated at least N times.

Chapter 5

List decoding

In this chapter we present an efficient algorithm for *list decoding* of AG codes. Similarly to partial unique decoding (see Chapter 4), list decoding aims to correct errors beyond half the minimum distance; however, as the name suggests, instead of declaring failure when unique decoding is impossible, we are tasked with returning a list containing all viable candidate-codewords. Following the style of the Guruswami-Sudan decoder [GS98], we solve this problem for *all* AG codes, and by utilizing contemporary algorithms for univariate polynomial matrices, as well as for univariate polynomials with power series coefficients, we achieve the best known complexity in this fully general setting.

5.1 Related work

The first non-trivial list decoder for RS codes was introduced by Sudan in [Sud97] and has since been interpreted as an extension of Welch and Berlekamp's key equations [WB86] – viewing them as an interpolation problem with certain degree constraints. Shokrollahi and Wasserman [SW99] generalized Sudan's technique to work for all AG codes, setting up the stage for the celebrated Guruswami-Sudan decoder [GS98], which achieved a greater decoding radius in the same general setting by introducing a multiplicity parameter. A dominant

paradigm in list decoding of AG codes was thus firmly established, revolving around an *interpolation step* as well as a *root-finding step*; and much future research would focus on making these two steps computationally efficient. In the next two paragraphs, about interpolation and root-finding respectively, we outline the "shortest path" from the emergence of this research to the present moment, omitting but some of the most essential works upon which the contributions in this chapter are based. For a more comprehensive historical overview of Guruswami-Sudan list-decoding, the reader is referred to [LO08, BB10] – and especially [Nie13].

Several authors, including [NH00, OF02, McE03, Ale05, FG05], formulated the interpolation step as a problem of finding a minimal polynomial, with respect to a weighted monomial order, in a certain vanishing ideal. Prompted by this, Lee and O'Sullivan developed a technique for obtaining such a polynomial from a Gröbner basis (of $\mathbb{F}[x]$-modules), that was itself computed starting from a particular generating set – first for RS codes [LO08], and then for one-point Hermitian codes [LO09]. The complexity of this strategy was further improved by Beelen and Brander in [BB10] by utilizing Alekhnovich's algorithm for row reduction of polynomial matrices [Ale05]. Furthermore, their decoder was applicable to the wider family of one-point codes over $C_{a,b}$ curves (see Chapter 2), making it more general. Specializing back to one-point Hermitian codes, Rosenkilde and Beelen [NB15] sped up this approach even more by delegating the row-reduction phase to the algorithm by Giorgi, Jeannerod and Villard [GJV03], which is more efficient than the one by Alekhnovich. Doing this required additional improvements to keep up with the new target complexity, including efficient computation of the initial $\mathbb{F}[x]$-basis, as well as a way of handling fractional weights. The result was the first list-decoder of one-point Hermitian codes having sub-quadratic complexity in the code length. The goal of this chapter is to generalize the tools from [NB15] to be applicable to all AG codes, which we accomplish by relying on the conceptual framework from [LBAO14]. Before shifting our attention to the root-finding step, we ought to mention the multivariate interpolation algorithm by Chowdhury, Jeannerod, Neiger, Schost and Villard [CJN+15] – it was the first to enable the currently best complexity in the special case of RS codes.

Some of the earliest root-finding algorithms for Guruswami-Sudan list-decoding include Roth and Ruckenstein's [RR00] as well as Gao and Shokrollahi's [GS00]. Alekhnovich described in [Ale05] an efficient approach for computing the $\mathbb{F}[\![x]\!]$-roots modulo x^β of a polynomial $Q \in \mathbb{F}[\![x]\!][z]$; its complexity was shown in [NB15] to be $\tilde{\mathcal{O}}(\beta^2\ell)$ operations in $\mathbb{F}$, where ℓ is the z-degree of Q. Another technique by Berthomieu, Lecerf and Quintin [BLQ13] achieved the cost $\tilde{\mathcal{O}}(\beta\ell^2)$. In this chapter, we rely on the algorithm by Neiger, Rosenkilde and Schost [NRS17], whose complexity of $\tilde{\mathcal{O}}(\beta\ell)$ operations is provably quasi-optimal.

We now proceed to a detailed discussion about the setting which is considered in this chapter. In particular, we show how the decoding problem in its most general form can be simplified by extending the base field $\mathbb{F} = \mathbb{F}_q$.

5.2 Setting

In this chapter we consider list decoding in the most general setting, i.e. for the code

$$\mathcal{C}_{\mathcal{L}}(D, G) = \{\mathrm{ev}_D(f) \mid f \in \mathcal{L}(G)\} \subseteq \mathbb{F}^n \ ,$$

where D and G are divisors with $\operatorname{supp} D \cap \operatorname{supp} G = \varnothing$, $D = P_1 + \cdots + P_n$ for some rational places $P_1, \ldots, P_n$, and $\mathrm{ev}_D(f) = (f(P_1), \ldots, f(P_n))$. Throughout the chapter we will keep using the notation $\mathrm{ev}_E(h)$ to denote evaluation of any function $h \in F$ with respect to any divisor $E = E_1 + \cdots + E_N$, where $E_1, \ldots, E_N$ are places that are not poles of h.

Notwithstanding that we are committed to full generality, we will permit ourselves to make a few simplifying yet non-restrictive assumptions on the function field F as well as the code $\mathcal{C}_{\mathcal{L}}(D, G)$ – incurring but a marginal penalty in computational complexity. As in Chapter 4, for example, we will require a rational place P_∞ of F, distinct from $P_1, \ldots, P_n$, and some of our later algorithms will require additional rational places as well. I principle, there is no guarantee that our original setting will be able to provide us with these, however, by simply extending our constant field – say from $\mathbb{F}_q$ to $\mathbb{F}_{q^e}$ for some e – we can artificially increase our supply of rational places as needed. Abiding by the notational conventions in [Sti09], we will denote by $F\mathbb{F}_{q^e}$ the function field obtained from F by extending the constant field to $\mathbb{F}_{q^e}$.

As far as decoding is concerned, the code $\mathcal{C}_{\mathcal{L}}(D, G)$ is in a trivial way an $\mathbb{F}_q$-linear subcode of the code obtained from $F\mathbb{F}_{q^e}$ using the divisors $\mathrm{Con}(D)$ and $\mathrm{Con}(G)$, where Con denotes the *conorm* with respect to $F\mathbb{F}_{q^e}/F$ [Sti09, Definition 3.1.8]. Consequently, any list-decoding algorithm for the code $\mathcal{C}_{\mathcal{L}}(\mathrm{Con}(D), \mathrm{Con}(G))$ can be repurposed for our original code $\mathcal{C}_{\mathcal{L}}(D, G)$ by simply discarding the returned codewords that contain entries from $\mathbb{F}_{q^e} \backslash \mathbb{F}_q$. The only potential issue that we ought to be careful about is that a single arithmetic operation in $\mathbb{F}_{q^e}$ costs $\tilde{\mathcal{O}}(e)$ operations in $\mathbb{F}_q$ [CK91], which means that e should preferably be small for complexity reasons. The following results will be helpful for bounding the extension degree needed to obtain all of the rational places that we will need.

Lemma 5.1. *Let F be a function field over $\mathbb{F}_q$ of genus g, and denote by N_e the number of rational places of the function field $F\mathbb{F}_{q^e}$ over $\mathbb{F}_{q^e}$. If $N, e \in \mathbb{Z}_{>0}$ are such that $e \geqslant 2 \log_q \max\{N, 2g+1\}$, then $N_e > N$.*

Proof. The Hasse-Weil bound $|(q^e + 1) - N_e| \leqslant 2q^{e/2}g$ implies that

$$\log_q N_e > \log_q(q^e - 2q^{e/2}g) = e/2 + \log_q(q^{e/2} - 2g) \geqslant e/2 \geqslant \log_q N \ .$$

□

Lemma 5.1 assures us that if we require a certain number of rational places, then an extension degree that is logarithmic in that number will always suffice in obtaining them, which is good news for the complexity. In addition to simply having many rational places, however, we will also need one of them to satisfy a specific property:

Lemma 5.2. *Let F be a function field over $\mathbb{F}$ having genus g. If P_∞ is a rational place and μ is the smallest positive element in its Weierstrass semigroup, then any set containing at least $3g + 1$ rational places distinct from P_∞ contains a place P_0 with a local parameter in $\mathcal{L}(\mu P_\infty)$.*

Proof. Let $x \in \mathcal{L}(\mu P_\infty)$ be a function satisfying $v_{P_\infty}(x) = -\mu$. First, we claim that the extension $F/\mathbb{F}(x)$ is separable. Assuming, for the sake of contradiction, that it is not, then by the general theory of inseparable extensions we can find an intermediate field E such that $E/\mathbb{F}(x)$ is separable and F/E is purely inseparable. By [Sti09, Proposition 3.10.2], therefore,

$$\mathbb{F}(x) \subseteq F^p := \{f^p \mid f \in F\} \ ,$$

where p denotes the characteristic, implying that $x = y^p$ for some $y \in F$. Consequently, since x has a pole at P_∞, then so does y, albeit of order μ/p. But this contradicts the minimality of μ, proving that $F/\mathbb{F}(x)$ is separable.

The Hurwitz genus formula (see e.g. [Sti09, Corollary 3.4.14]) thus applies to this extension, so we proceed by using it to estimate the genus g. Letting $Q_\infty = P_\infty \cap \mathbb{F}(x)$, we observe that since $v_{P_\infty}(x) = -\mu$ and $v_{Q_\infty}(x) = -1$, then $e(P_\infty|Q_\infty) = \mu$. Now, assuming that we have $N > 3g$ rational places distinct from P_∞, say $P_1, \ldots, P_N$, let $Q_i = P_i \cap \mathbb{F}(x)$ for $i = 1, \ldots, N$, and observe that

$$v_{P_i}(x - x(P_i)) = e(P_i|Q_i)v_{Q_i}(x - x(P_i)) = e(P_i|Q_i) \ .$$

Suppose, again for the sake of contradiction, that $e(P_i|Q_i) \geqslant 2$ for every i. Since $\mu = [F : \mathbb{F}(x)]$ by [Sti09, Theorem 1.4.11], then the Hurwitz genus formula combined with the estimate $d(P_i|Q_i) \geqslant e(P_i|Q_i) - 1$, where d denotes the different

exponent, implies that

$$\begin{aligned} 2g-2 &\geqslant -2[F:\mathbb{F}(x)] + d(P_\infty|Q_\infty) + \sum_{i=1}^{N} d(P_i|Q_i) \\ &\geqslant -2\mu + (\mu-1) + N(2-1) \\ &= N-\mu-1 \\ &\geqslant N-g-2\ , \end{aligned}$$

the last inequality being a consequence of $\mu \leqslant g+1$. But then $N \leqslant 3g$, which contradicts our earlier assumption. We conclude that $v_{P_i}(x-x(P_i)) = 1$ for at least one value of i, meaning that we can pick $P_0 = P_i$, since it has the local parameter $x - x(P_i) \in \mathcal{L}(\mu P_\infty)$. □

To motivate Lemma 5.2, observe that by extending our constant field in accordance with Lemma 5.1, we can "create" sufficiently many new rational places to ensure existence of a function $x \in F$ satisfying the following two properties:

1) $x \in \mathcal{L}(\mu P_\infty)$, and
2) x is a local parameter of some rational place P_0 not in $\operatorname{supp} G$.

As we will see in Section 5.3, Property 1 allows us to impose an $\mathbb{F}[x]$-module structure on the interpolation step of Guruswami-Sudan. In Section 5.5.6, we will use Property 2 to solve the root-finding step by representing the coefficients of the polynomial at hand using power series in $\mathbb{F}[\![x]\!]$. Letting x be the generating element in both representations allows for efficient conversion between them – and so existence of an x having these two properties is very helpful.

The final assumption that we will make about our setting is that the divisor G is effective. In the following lemma we show that this assumption is also nonrestrictive – at least as long as our code is not *degenerate*: we call a code degenerate if there is an index $i \in \{1,\dots,n\}$ such that every single codeword is zero in the i-th entry.

In the following lemma, recall that two codes $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}^n$ are said to be *monomially equivalent* if there exist nonzero $\gamma_1,\dots,\gamma_n \in \mathbb{F}$ and a permutation π on $\{1,\dots,n\}$ such that $(c_1,\dots,c_n) \in \mathcal{C} \iff (\gamma_1 c_{\pi(1)},\dots,\gamma_n c_{\pi(n)}) \in \mathcal{C}'$; we will not need the permutation.

Lemma 5.3. *If F is a function field, and D, G are divisors as before, then either the code $\mathcal{C}_\mathcal{L}(D,G)$ is degenerate, or $\mathcal{C}_\mathcal{L}(D,\operatorname{Con}(G))$ (of which it is a subcode over $\mathbb{F}_q$) is monomially equivalent over $\mathbb{F}_{q^e}$, where $e \geqslant 1 + \lceil \log_q(n) \rceil$, to a code $\mathcal{C}_\mathcal{L}(D,G')$, where G' is an effective divisor of $F\mathbb{F}_{q^e}$ of degree $\deg G$.*

Proof. Consider the extension $\mathbb{F}_{q^e}/\mathbb{F}_q$, and let $\mathcal{C}_i = \{(c_1, \ldots, c_n) \in \mathcal{C} \mid c_i = 0\}$, where $\mathcal{C} = \mathcal{C}_{\mathcal{L}}(D, \mathrm{Con}(G))$. Clearly, if $\mathcal{C}_{\mathcal{L}}(D, G)$ is nondegenerate, then so is $\mathcal{C}$. In this case, $\mathcal{C} \neq \mathcal{C}_i$ for all i. If every codeword in $\mathcal{C}$ has at least one zero coordinate, then $\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{C}_i$, which implies that $(q^e)^k \leqslant n(q^e)^{k-1}$, where $k = \dim \mathcal{C}$. It follows that $q^e \leqslant n$, implying the contradiction that $e \leqslant \log_q(n)$. Consequently, $\mathcal{C}$ contains a codeword of full Hamming weight n, say $\boldsymbol{c} = \mathrm{ev}_D(\widetilde{f})$ for some $\widetilde{f} \in \mathcal{L}(\mathrm{Con}(G))$. Since by construction $\widetilde{f}(P_i) \neq 0$ for all i, we see that the codes $\mathcal{C}$ and $\mathcal{C}_{\mathcal{L}}(D, \mathrm{Con}(G) + (\widetilde{f}))$ are monomially equivalent under the map $(c_1, \ldots, c_n) \mapsto (\widetilde{f}(P_1)c_1, \ldots, \widetilde{f}(P_n)c_n)$. Moreover, the divisor $G' = \mathrm{Con}(G) + (\widetilde{f})$ is effective and has support disjoint from D. □

From the perspective of error-correction, degenerate codes can be safely disregarded; if the i-th entry of every codeword is zero, then correcting errors in that position is trivial. Moreover, since this entry carries no information, then one might instead consider the punctured code where it has been removed. Doing this will neither change the code's dimension nor its minimum distance, which means that decoding of any degenerate code can be reduced to decoding of a nondegenerate one.

In the context of Lemma 5.3, note that since $\mathcal{C}_{\mathcal{L}}(D, \mathrm{Con}(G))$ and $\mathcal{C}_{\mathcal{L}}(D, G')$ are monomially equivalent, any list-decoding algorithm for $\mathcal{C}_{\mathcal{L}}(D, G')$ immediately gives a one for $\mathcal{C}_{\mathcal{L}}(D, \mathrm{Con}(G))$ as well as for its $\mathbb{F}_q$-subcode $\mathcal{C}_{\mathcal{L}}(D, G)$. Of course, this incurs the additional complexity of dividing and multiplying codeword entries with the column multipliers $\widetilde{f}(P_i)$, however, for each codeword this only costs $\mathcal{O}(n)$ operations in $\mathbb{F}_q$, i.e. $\widetilde{\mathcal{O}}(ne)$ operations in $\mathbb{F}_q$. Moreover, as we are about to see, the extension degree e can be chosen small enough not to have any impact on the total complexity – at least as far as the $\widetilde{\mathcal{O}}$-notation is concerned.

Below we summarize our simplifying (non-restrictive) assumptions and fix some associated notation that will be used henceforth in the chapter.

1) We assume that G is an effective divisor satisfying $0 \leqslant \deg G \leqslant n + 2g - 1$ (see Section 1.6 on page 11).

2) We assume that apart from the rational places in the support of the divisor $D = P_1 + \cdots + P_n$, the function field F has at least one more rational place P_∞, which may or may not be in $\operatorname{supp} G$.

3) There exists a rational place P_0 of F having a local parameter $x \in \mathcal{L}(\mu P_\infty)$, where μ is the smallest positive element of the Weierstrass semigroup at P_∞. The place P_0 may be in $\operatorname{supp} D$ but not in $\operatorname{supp} G$.

4) We assume that we generally have access to as many rational places not in $\operatorname{supp} G$ as we need, provided that we account for the cost penalty associated with extending the constant field using Lemma 5.1 (see **??**).

Let us now assess the size of the extension degree e needed to satisfy the first three items in the above list – the fourth item will be considered separately. Although one can likely do better, for the sake of simplicity we pick $e = e_1 e_2 e_3$, where

$$\begin{aligned} e_1 &= 1 + \lceil \log_q n \rceil \ , \\ e_2 &= \lceil 2 \log_{q^{e_1}} \max\{n+1, 2g+1\} \rceil \ , \text{ and} \\ e_3 &= \lceil 2 \log_{q^{e_1 e_2}} (5g + 1 + n) \rceil \ . \end{aligned}$$

Too see why this choice works, observe that extending $\mathbb{F}_q$ to $\mathbb{F}_{q^{e_1}}$ takes care of the first item due to Lemma 5.3, further extending $\mathbb{F}_{q^{e_1}}$ to $\mathbb{F}_{q^{e_1 e_2}}$ covers the second item by Lemma 5.1, and finally, the last item follows from Lemma 5.2 by extending $\mathbb{F}_{q^{e_1 e_2}}$ to $\mathbb{F}_{q^{e_1 e_2 e_3}}$. Using the identity $\log_{q^t}(z) = \log_q(z)/t$, this can be summarized as

$$\begin{aligned} e = e_1 e_2 e_3 &\leqslant 2 \log_q (5g + 1 + n) + e_1 e_2 \\ &\leqslant 2 \log_q (5g + 1 + n) + 2 \log_q \max\{n+1, 2g+1\} + e_1 \\ &\leqslant 2 \log_q (5g + 1 + n) + 2 \log_q \max\{n+1, 2g+1\} + \log_q(n) + 2 \ . \end{aligned}$$

In other words, the the cost of satisfying the first three simplifying assumptions is merely a factor of $\mathcal{O}(\log(n+g))$, which is does not break our target complexity.

When it comes to the last assumption, estimating the number rational places that we need gets quite technical, albeit conceptually simple: it essentially boils down to ensuring existence of invertible multi-point evaluation maps on certain Riemann-Roch spaces, which is useful for efficient multiplication of function field elements. It can be verified throughout the chapter that for decoding parameters $s, \ell \in \mathbb{Z}_{>0}$, where s is the multiplicity and ℓ is the designed list size (see Section 5.4), requiring an additional

$$\begin{aligned} m := \deg G + \max\{&(\ell+1)\deg G + 4g + (s+1)n, \\ &\deg G + (\ell+3)(2g-1) + (s+1)n + 2 + \mu\} \in \mathcal{O}(\ell(n+g)) \ , \end{aligned}$$

rational places will always suffice; however, justifying this bound here is impractical. The important thing to note is that we can ensure the existence of these places by extending our constant field once again, this time with extension degree $\lceil 2 \log_{q^e}(\max\{m, 2g+1\}) \rceil = \lceil 2 \log_{q^e} m \rceil$ – relying on Lemma 5.1 as before. Consequently, the computational penalty of this last assumption is an

additional factor of $\mathcal{O}(\log(\ell(n+g)))$, which also does not interfere with our target complexity.

In the remainder of this chapter, therefore, we will simply assume that the constant field $\mathbb{F} = \mathbb{F}_q$ is large enough to satisfy all of the simplifying assumptions mentioned in this section.

Remark 5.4. The extension degree estimates given in this section are but crude upper bounds – they are chosen in a way that preserves the total complexity in the $\widetilde{\mathcal{O}}$-sense without being optimized any further than that. The hidden factors, therefore – both constant and logarithmic – can be easily improved upon, however, doing so falls outside of this chapter's scope.

5.3 Representations of function field elements

Our decoder consists of three main parts, each requiring its own way of representing function field elements:

1) Following the ideas in [BB10], we approach the interpolation step by first constructing a basis of the associated $\mathbb{F}[x]$-module. Doing this requires efficient multiplication of function field elements, which we accomplish by representing them using evaluations at many rational places and multiplying them "pointwise".

2) In order to obtain a *reduced basis* of the aforementioned $\mathbb{F}[x]$-module, we will switch the representation to one that is isomorphic to $\mathbb{F}[x]^\mu$. This will be detailed shortly.

3) Lastly, we address the root-finding step using the results in [NRS17]. For this, we will represent certain functions using their power series expansions in the local parameter x of P_0.

We now proceed with a detailed elaboration of the second out of the three representations mentioned above. As we did in Chapter 4 on page 67, for every divisor A we will use the shorthand notation $\text{Я}(A) = \text{Я}_{P_\infty}(A)$, where $\text{Я}_P(A)$ is as in Definition 1.14, and for any function $a \in \text{Я}(A)$, we will write $\delta_A(a) = \delta_A^{(P_\infty)}(a)$. It is clear that Я is a ring and $\text{Я}(A)$ is a Я-module. Moreover, Я is an $\mathbb{F}[x]$-module and so is $\text{Я}(A)$, which means that we can represent any $a \in \text{Я}(A)$ using an $\mathbb{F}[x]$-basis of $\text{Я}(A)$ (see Lemma 5.6). Modules of the form $\text{Я}(A)$ are essentially already considered for decoding in [KP95] as well as in [BH08, LBAO14, NB15].

Following [LBAO14], we will use a special $\mathbb{F}[x]$-basis of $\mathcal{R}(A)$, which they call an *Apéry system*:

Definition 5.5. For any divisor A let $y_i^{(A)} \in \mathcal{A}_i$ be such that $\delta_A(y_i^{(A)}) \leqslant \delta_A(a)$ for all $a \in \mathcal{A}_i$, where $i = 0, \ldots, \mu - 1$ and

$$\mathcal{A}_i = \{a \in \mathcal{R}(A) \mid \delta_A(a) \equiv i \mod \mu\} .$$

We also define $y_i = y_i^{(0)}$.

Lemma 5.6. *For any divisor A it holds that*

1) $y_0^{(A)}, \ldots, y_{\mu-1}^{(A)}$ is an $\mathbb{F}[x]$-basis of $\mathcal{R}(A)$ and

2) $-\deg A \leqslant \delta_A(y_i^{(A)}) \leqslant 2g - 1 - \deg(A) + \mu$ for $i = 0, \ldots, \mu - 1$.

Proof. The first statement is from [LBAO14]. For the convenience of the reader we give a proof. From the strict triangle inequality for v_{P_∞}, it is clear that the elements $y_0^{(A)}, \ldots, y_{\mu-1}^{(A)}$ are linearly independent over $\mathbb{F}[x]$. Also, it is clear that $\mathcal{Y} \subseteq \mathcal{R}(A)$, where $\mathcal{Y} = \langle y_0^{(A)}, \ldots, y_{\mu-1}^{(A)} \rangle_{\mathbb{F}[x]}$. If $\mathcal{Y} \neq \mathcal{R}(A)$, then there would exist $a \in \mathcal{R}(A) \backslash \mathcal{Y}$, such that $\delta_A(a) > -\infty$ is minimal. Writing $\delta_A(a) = m\mu + r$ and $\delta_A(y_r^{(A)}) = m'\mu + r$, where $m, m', r \in \mathbb{Z}$ with $0 \leqslant r < \mu$, note that $m' \leqslant m$ by definition of $y_r^{(A)}$. Since

$$\delta_A(x^{m-m'} y_r^{(A)}) = \delta(x^{m-m'}) + \delta_A(y_r^{(A)}) = (m - m')\mu + (m'\mu + r) = \delta_A(a) ,$$

then there exists a constant $\beta \in \mathbb{F}$ such that $\delta_A(c) < \delta_A(a)$, where

$$c = a - \beta x^{m-m'} y_r^{(A)} \in \mathcal{R}(A) .$$

The minimality of $\delta_A(a)$ guarantees that $c \in \mathcal{Y}$, however, this would imply that $a = c + \beta x^{m-m'} y_r^{(A)} \in \mathcal{Y}$. Hence $\mathcal{Y} = \mathcal{R}(A)$ after all.

In the second statement, the lower bound simply follows from the fact that $y_i^{(A)} \in \mathcal{L}(\delta_A(y_i^{(A)})P_\infty + A) \neq \{0\}$. For the upper bound it is sufficient to show that for every integer $m > 2g - 1 - \deg(A)$ there exists an $a \in \mathcal{R}(A)$ with $\delta_A(a) = m$. But indeed, if $m > 2g - 1 - \deg(A)$, then $\deg(mP_\infty + A) > 2g - 1$, and so [Sti09, Theorem 1.5.17] implies that $\mathcal{L}(mP_\infty + A) \neq \mathcal{L}\big((m-1)P_\infty + A\big)$, which concludes the proof. □

Remark 5.7. According to Lemma 5.6, any function $a \in \mathcal{R}(A)$ can be uniquely represented as $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)}$, where $a_i \in \mathbb{F}[x]$. For clarity, we highlight that it is only the coefficients a_i that need to be stored in memory when implementing

our algorithms; i.e. in practice, a would be represented as $(a_0, \ldots, a_{\mu-1}) \in \mathbb{F}[x]^\mu$. When it comes to the basis functions $y_i^{(A)}$, we only have to know their evaluations at certain rational places, as well as their power series expansions at P_0 – this information is used to convert between the different representations of a.

We end this section with a result that bounds the "size" of function field elements when represented in terms of the $\mathbb{F}[x]$-basis from Definition 5.5. This will be useful in complexity estimates.

Lemma 5.8. *If $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)} \in \text{Я}(A)$, where $a_i \in \mathbb{F}[x]$ and A is a divisor, then*

$$\deg a_i \leqslant \frac{1}{\mu}(\delta_A(a) - \delta_A(y_i^{(A)})) \leqslant \frac{1}{\mu}(\delta_A(a) + \deg A) \ .$$

Proof. Simply observe that for $i = 0, \ldots, \mu - 1$ it holds that

$$\delta_A(a) = \max_j \delta_A(a_j y_j^{(A)}) \geqslant \delta(a_i) + \delta_A(y_i^{(A)}) \geqslant \delta(a_i) - \deg A \ ,$$

where the equality follows from the strict triangle inequality for v_{P_∞}, and second inequality is given by Lemma 5.6. But then

$$\deg a_i = \delta(a_i)/\mu \leqslant \frac{1}{\mu}(\delta_A(a) - \delta_A(y_i^{(A)})) \leqslant \frac{1}{\mu}(\delta_A(a) + \deg A) \ .$$

□

5.4 Guruswami-Sudan decoding

In this section, we formulate the Guruswami-Sudan list-decoding algorithm [GS99] in terms of Я-modules. For the remainder of this chapter, fix the decoding parameters $s, \ell \in \mathbb{Z}_{>0}$ such that $s \leqslant \ell$, where ℓ is the designed list size and s is the multiplicity parameter. The corresponding list-decoding radius will be denoted by τ.

Definition 5.9. Let P be a rational place of F, $r \in \mathbb{F}$ and $Q \in F[z]$. We will say that "Q has a root of multiplicity s at (P, r)" if for any local parameter ϕ of P, there exist $c_{u,v} \in \mathbb{F}$ such that

$$Q = \sum_{\substack{u,v \geqslant 0 \\ u+v \geqslant s}} c_{u,v} \phi^u (z - r)^v$$

with $c_{u,s-u} \neq 0$ for at least one $0 \leqslant u \leqslant s$.

The following lemma will be useful for proving the main theorem of Guruswami-Sudan decoding in our context.

Lemma 5.10. *If $Q \in F[z]$ has a root of multiplicity s at (P, r) and $f \in F$ is such that $f(P) = r$, then $v_P(Q(f)) \geqslant s$.*

Proof. Writing

$$Q(f) = \sum_{\substack{u,v \geqslant 0 \\ u+v \geqslant s}} c_{u,v} \phi^u (f-r)^v \ ,$$

where ϕ is any local parameter of P and $c_{u,v} \in \mathbb{F}$, the triangle inequality gives

$$v_P(Q(f)) \geqslant \min_{\substack{u,v \geqslant 0 \\ u+v \geqslant s}} \big(v_P(\phi^u) + v_P((f-r)^v) \big) \geqslant \min_{\substack{u,v \geqslant 0 \\ u+v \geqslant s}} (u+v) \geqslant s \ .$$

□

We are now ready for the main decoding theorem, which formally introduces the aforementioned steps of interpolation and root-finding. It will be convenient to fix some more notation: For any $Q = \sum_{t=0}^{\ell} z^t Q^{(t)}$ with $Q^{(t)} \in \text{Я}(-tG)$, we define $\delta_G(Q) = \max_t \delta_{-tG}(Q^{(t)})$, and for a given received word $\boldsymbol{r} = (r_1, \ldots, r_n) \in \mathbb{F}^n$, we will consider the set

$$\begin{aligned} \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) = \Big\{ Q = \sum_{t=0}^{\ell} z^t Q^{(t)} \in F[z] \ \Big| \ & Q^{(t)} \in \text{Я}(-tG) \text{ and} \\ & Q \text{ has a root of multiplicity at least } s \text{ at } (P_j, r_j) \\ & \text{for } j = 1, \ldots, n \Big\} \ . \end{aligned} \tag{5.1}$$

Theorem 5.11 (Special case of Guruswami–Sudan [GS99])**.** *Let $\boldsymbol{r}$ be a received word and $Q \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ with $\delta_G(Q) < s(n-\tau)$. If $f \in \mathcal{L}(G)$ is such that $d(\boldsymbol{r}, \mathrm{ev}_D(f)) \leqslant \tau$, where d denotes the Hamming distance, then $Q(f) = 0$.*

Proof. Since $f^t \in \mathcal{L}(tG)$ and $Q^{(t)} \in \text{Я}(-tG)$, then $f^t Q^{(t)} \in \text{Я}$, and consequently $Q(f) \in \text{Я}$. Furthermore, since $\delta_{tG}(f^t) \leqslant 0$, then by the triangle inequality

$$\delta(Q(f)) \leqslant \max_t \delta_{-tG}(Q^{(t)}) = \delta_G(Q) \ .$$

Letting $\mathcal{E} = \{ j \mid r_j \neq f(P_j) \}$, note that the cardinality of $\mathcal{E}$ is at most τ, and since $f(P_j) = r_j$ for $j \notin \mathcal{E}$, then it follows from Lemma 5.10 that $Q(f) \in \text{Я}(-T)$, where $T = s \sum_{j \notin \mathcal{E}} P_j$. But $\delta_G(Q) < s(n-\tau) \leqslant \deg T$, therefore

$$Q(f) \in \mathcal{L}\big(\delta_G(Q) P_\infty - T\big) = \{0\} \ .$$

□

In the context of Theorem 5.11, the interpolation step consists of finding a nonzero polynomial $Q \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) \subset F[z]$ with $\delta_G(Q) < s(n-\tau)$, while the root-finding step consists of finding all $f \in \mathcal{L}(G)$ satisfying $Q(f) = 0$. Before we sketch our approach for solving these two steps in Section 5.4.2, let us first understand the $\mathbb{F}[x]$-module structure of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ in the next section.

5.4.1 Module structure of interpolation

It is easily seen that the set $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ from (5.1) is a module over the ring Я. This point of view will aid us in describing a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over $\mathbb{F}[x]$, which is the ultimate goal of this section. We begin with an investigation of the Я-module structure of $\text{Я}(A)$ for any divisor A, where we will take advantage of the fact that Я is a Dedekind domain and $\text{Я}(A)$ is one if its fractional ideals [NX01, Section 1.2]. It is well known that any such fractional ideal can be generated by at most two elements [FTT91, Corollary 2 to Theorem 4], however, in our setting we also care about the "size" of these elements for complexity reasons.

Lemma 5.12. *For any divisor A it holds that $\text{Я}(A) = \langle a_1, a_2 \rangle_{\text{Я}}$ for some functions $a_1, a_2 \in \text{Я}(A)$ satisfying*

$$\delta_A(a_1) \leqslant 2g - 1 - \deg(A) + \mathfrak{a} \quad \textit{and} \quad \delta_A(a_2) \leqslant 4g - 2 - \deg(A) + \mathfrak{a} \ ,$$

where $\mathfrak{a} = \sum_{i=1}^{t} \deg Q_i$ and $A = \sum_{i=1}^{t} n_i Q_i$ for some places $Q_1, \ldots, Q_t$.

Proof. Prime ideals of Я correspond exactly to places distinct from P_∞. Therefore, from the proof of in [FTT91, Corollary 2 to Theorem 4], we see that two elements $a_1, a_2 \in \text{Я}(A)$ generate $\text{Я}(A)$ as Я-module if and only if for all places $Q_i \in \operatorname{supp}(A)$ distinct from P_∞, we have $\min\{v_{Q_i}(a_1), v_{Q_i}(a_2)\} = -n_i$ and for any other place $Q \neq P_\infty$ of F we have $\min\{v_Q(a_1), v_Q(a_2)\} = 0$. We proceed by constructing such two elements:

Letting $m_1 = 2g - 1 - \deg(A) + \mathfrak{a}$, for $j = 1, \ldots, t$ pick any

$$a_1^{(j)} \in \mathcal{L}(A - \sum_{i \neq j} Q_i + m_1 P_\infty) \backslash \mathcal{L}(A - \sum_i Q_i + m_1 P_\infty) \ .$$

To see that such $a_1^{(j)}$ exist, use the Riemann-Roch theorem to conclude that

$$\mathfrak{l}(A - \sum_{i \neq j} Q_i + m_1 P_\infty) > \mathfrak{l}(A - \sum_i Q_i + m_1 P_\infty) \ .$$

Choosing $a_1 = \sum_{j=1}^{t} a_1^{(j)}$, we see that $v_{Q_i}(a_1) = -n_i$ for $j = 1, \ldots, t$, while $v_Q(a_1) \geqslant 0$ for any other place Q distinct from P_∞. Moreover, $\delta_A(a_1) \leqslant m_1$ since $a_1 \in \mathcal{L}(A + m_1 P_\infty)$.

Now, denote by $Q_{t+1}, \ldots, Q_{t+s}$ the zeroes of a_1 not in $\operatorname{supp}(A) \cup \{P_\infty\}$, and observe that since $a_1 \in \mathcal{L}(A+m_1P_\infty)$, then $\sum_{i=t+1}^{t+s} \deg Q_i \leqslant \deg A + m_1$. Letting $m_2 = 2g-1+m_1 = 4g-2-\deg(A)+\mathfrak{a}$, similarly to the above, we can construct an $a_2 \in \mathcal{L}(A + m_2 P_\infty)$ with $v_{Q_i}(a_2) = 0$ for $i = t+1, \ldots, t+s$, and by this construction $\delta_A(a_2) \leqslant m_2$.

To verify that a_1 and a_2 are satisfactory, observe that $v_{Q_i}(a_2) \geqslant -n_i$ and $v_{Q_i}(a_1) = -n_i$ for $i = 1, \ldots, t$. Letting $Q \notin \operatorname{supp} A \cup \{P_\infty\}$, observe that if Q is a zero of a_1, then $\min\{v_Q(a_1), v_Q(a_2)\} = 0$ since $v_Q(a_2) \geqslant 0$. If, on the other hand, Q is a zero of a_1, then $v_Q(a_2) = 0$, so that also in this case $\min\{v_Q(a_1), v_Q(a_2)\} = 0$. The sought conclusion then follows. □

We proceed by decomposing $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ into simpler Я-modules, which will allow us to describe a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ over Я using Lemma 5.12.

In the remainder of this chapter, we will make extensive use of the divisors

$$G_u^{(s)} = -uG - \max\{0, s-u\}D \quad \text{for } u = 0, \ldots, \ell \ , \tag{5.2}$$

and when s is clear from context, then we may simply write $G_u = G_u^{(s)}$.

Theorem 5.13. *If $\boldsymbol{r} = (r_1, \ldots, r_n)$ is a received word and $R \in Я(G)$ is such that $R(P_j) = r_j$ for $j = 1, \ldots, n$, then*

$$\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G) = \bigoplus_{u=0}^{\ell} (z-R)^u Я(G_u) \ .$$

Proof. Since $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ is a Я-module, then in order to prove the inclusion

$$\bigoplus_{u=0}^{\ell} (z-R)^u Я(G_u) \subseteq \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G) \ ,$$

it suffices to show that $(z-R)^u Я(G_u) \subseteq \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ for $u = 0, \ldots, \ell$. But indeed, every element in $(z-R)^u Я(G_u)$ has a root of multiplicity at least s at (P_j, r_j) because $(z-R)^u$ has one of multiplicity u and every $h \in Я(G_u)$ satisfies $v_{P_j}(h) \geqslant \max\{0, s-u\}$. Furthermore, since $R \in Я(G)$, then

$$(z-R)^u Я(G_u) = \Big(\sum_{t=0}^{u} z^t \binom{u}{t} (-R)^{u-t} \Big) Я(G_u) \subseteq \bigoplus_{t=0}^{\ell} z^t Я(-tG) \ ,$$

which proves the sought inclusion.

Now, with the aim of proving the reverse inclusion

$$\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) \subseteq \bigoplus_{u=0}^{\ell}(z-R)^u \text{Я}(G_u) \ ,$$

let $Q = \sum_{t=0}^{\ell} z^t Q^{(t)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ and rewrite

$$\begin{aligned} Q(z) &= Q((z-R)+R) \\ &= \sum_{t=0}^{\ell}((z-R)+R)^t Q^{(t)} \\ &= \sum_{t=0}^{\ell}\sum_{u=0}^{t}\binom{t}{u}(z-R)^u R^{t-u} Q^{(t)} \\ &= \sum_{u=0}^{\ell}\sum_{t=u}^{\ell}\binom{t}{u}(z-R)^u R^{t-u} Q^{(t)} = \sum_{u=0}^{\ell}(z-R)^u \widetilde{Q}^{(u)} \ , \end{aligned}$$

where $\widetilde{Q}^{(u)} = \sum_{t=u}^{\ell}\binom{t}{u}R^{t-u}Q^{(t)}$. It now suffices to show that $\widetilde{Q}^{(u)} \in \text{Я}(G_u)$ for each u, which we prove by induction on s. From Lemma 5.10, it follows that $\widetilde{Q}^{(0)} = Q(R) \in \text{Я}(G_0)$, and since $R \in \text{Я}(G)$, then clearly $\widetilde{Q}^{(u)} \in \text{Я}(-uG)$. Consequently, in the case of $s = 1$, where $\text{Я}(-uG) = \text{Я}(G_u)$ for $u > 0$, we obtain the sought conclusion that $\widetilde{Q}^{(u)} \in \text{Я}(G_u)$.

Proceeding to the induction step, where we assume that $s > 1$, observe that it follows from $\widetilde{Q}^{(0)} \in \text{Я}(G_0) \subseteq \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ that

$$\begin{aligned} \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) \ni Q - \widetilde{Q}^{(0)} &= \sum_{u=0}^{\ell}(z-R)^u \widetilde{Q}^{(u)} - \widetilde{Q}^{(0)} \\ &= \sum_{u=1}^{\ell}(z-R)^u \widetilde{Q}^{(u)} = (z-R)\sum_{u=0}^{\ell-1}(z-R)^u \widetilde{Q}^{(u+1)} \ . \end{aligned}$$

Since $z - R$ has a root of multiplicity one at (P_j, r_j) for all j, we see that $\sum_{u=0}^{\ell-1}(z-R)^u \widetilde{Q}^{(u+1)}$ has a root of multiplicity at least $s-1$, and therefore

$$\sum_{u=0}^{\ell-1}(z-R)^u \widetilde{Q}^{(u+1)} \in \mathcal{M}_{s-1,\ell}^{(\boldsymbol{r})}(D,G) \ .$$

The induction hypothesis for $s-1$ thus implies that $\widetilde{Q}^{(u+1)} \in \text{Я}(G_u^{(s-1)})$ for $u = 0, \ldots, \ell-1$, from which it follows that $\widetilde{Q}^{(u+1)} \in \text{Я}(G_{u+1}^{(s)})$, since we also know that $\widetilde{Q}^{(u+1)} \in \text{Я}\big(-(u+1)G\big)$. In combination with the previously established $\widetilde{Q}^{(0)} \in \text{Я}(G_0^{(s)})$, this concludes the proof. $\square$

Remark 5.14. In the context of Theorem 5.13, note that we have already seen R in Definition 4.1.

Armed with Lemma 5.12 and Theorem 5.13, it is straightforward to show that $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ can be generated over Я by fairly "small" elements:

Corollary 5.15. *There exist* $g_1^{(u)}, g_2^{(u)} \in \text{Я}(G_u)$ *for* $u = 0, \dots, \ell$ *with*

$$\delta_{G_u}(g_1^{(u)}) \leqslant 2g - 1 + (u+1)\deg G + \max\{0, s-u+1\}n \quad \textit{and}$$
$$\delta_{G_u}(g_2^{(u)}) \leqslant 4g - 2 + (u+1)\deg G + \max\{0, s-u+1\}n$$

such that

$$\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) = \langle B_v^{(u)} \mid u \in \{0,\dots,\ell\}, v \in \{1,2\}\rangle_{\text{Я}} \ , \quad \textit{where}$$
$$B_v^{(u)} = (z-R)^u g_v^{(u)} = \sum_{t=0}^{u} \binom{u}{t} z^t (-R)^{u-t} g_v^{(u)} \in \bigoplus_{t=0}^{\ell} z^t \text{Я}(-tG) \ .$$

Proof. To obtain the stated upper bounds on $\delta_{G_u}(g_1^{(u)})$ and $\delta_{G_u}(g_2^{(u)})$ from Lemma 5.12, note that

$$\sum_{Q \in \operatorname{supp} G} \deg Q \leqslant \deg G$$

since G is an effective divisor. The bounds then follow from

$$\sum_{Q \in \operatorname{supp} G_u} \deg Q \leqslant \begin{cases} \deg G + n & \text{if } u < s \\ \deg G & \text{if } u \geqslant s \end{cases} \ .$$

The claim that the $B_v^{(u)}$ generate $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over Я is a direct consequence of Theorem 5.13 combined with Lemma 5.12. □

Remark 5.16. The proof of Corollary 5.15 actually implies that for $u = s$ the stated upper bounds for $\delta_{G_u}(g_1^{(u)})$ and $\delta_{G_u}(g_2^{(u)})$ can be improved by n.

In the context of Corollary 5.15, it is straightforward to obtain a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over $\mathbb{F}[x]$.

Corollary 5.17. *It holds that*

$$\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) = \langle y_i B_v^{(u)} \mid i \in \{0,\dots,\mu-1\}, u \in \{0,\dots,\ell\}, v \in \{1,2\}\rangle_{\mathbb{F}[x]} \ .$$

Remark 5.18. Since $G_u = -uG$ for $s < u \leqslant \ell$, a minor modification of the proof of Theorem 5.13 results in a different decomposition:

$$\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) = \bigoplus_{u=0}^{\ell} \begin{cases} (z-R)^u \text{Я}(G_u) & \text{if } 0 \leqslant u \leqslant s \\ (z-R)^s z^{u-s} \text{Я}(G_u) & \text{if } s < u \leqslant \ell \end{cases}.$$

Consequently, an alternative Я-generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ is given by

$$\widetilde{B}_v^{(u)} = \begin{cases} (z-R)^u g_v^{(u)} = B_v^{(u)} & \text{if } 0 \leqslant u \leqslant s \\ (z-R)^s z^{t-s} g_v^{(u)} & \text{if } s < u \leqslant \ell \end{cases}$$

for $v = 1, 2$. The corresponding $\mathbb{F}[x]$-generating set $\{y_i \widetilde{B}_v^{(u)}\}$ is cheaper to compute when $s < \ell$, and is likely to improve the total complexity by a factor of s/ℓ. However, due to the approaching deadline for this thesis, this will not be pursued any further.

Having understood the module structure of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$, we proceed by sketching our decoding algorithm.

5.4.2 Strategy outline

Before delving into the computational details, let us present an overview over the main steps in our approach to Guruswami-Sudan list-decoding. Given an AG code $\mathcal{C}_{\mathcal{L}}(D,G) \subseteq \mathbb{F}^n$ satisfying the non-restrictive assumption described in Section 5.2, decoding parameters $s, \ell \in \mathbb{Z}_{>0}$ with $s \leqslant \ell$ and a received word $\boldsymbol{r} \in \mathbb{F}^n$, our strategy will be as follows:

Interpolation step:

1) Compute a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over Я (see Section 5.5.3).
2) Compute a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over $\mathbb{F}[x]$ (see Section 5.5.4).
3) Use fast row reduction over $\mathbb{F}[x]$ to find a nonzero $Q \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ satisfying $\delta_G(Q) < s(n-\tau)$ (Section 5.5.5).

Root-finding step:

1) Compute the power series representation $\widehat{Q} \in \mathbb{F}[\![x]\!][z]$ of Q by expanding its coefficients at P_0.

2) Compute the roots of $\widehat{Q}$ over $\mathbb{F}[\![x]\!]$.

3) Convert these roots to $\mathcal{L}(G)$, discarding any "spurious" solutions.

As we will see in Section 5.5.6, all of these steps can be solved using $\widetilde{\mathcal{O}}(\mu^{\omega-1}\ell^{\omega+1}(n+g))$ operations in $\mathbb{F}$. Moreover, by choosing the slightly more complicated generating set from Remark 5.18, this complexity can likely be improved to $\widetilde{\mathcal{O}}(\mu^{\omega-1}s\ell^{\omega}(n+g))$.

Now that we have charted out our course, we are ready to present the algorithmic content of this chapter.

5.5 Algorithms

In this section, we describe in full detail all of the intermediate algorithms that we will use in our realization of the Guruswami-Sudan list decoder – the actual decoder is presented separately in Section 5.6.

Our first milestone will be to compute the generating set $\{B_v^{(u)}\}_{v=1,2}^{u=0,\dots,\ell}$ of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over $\mathcal{R}$ as given by Corollary 5.15, which will require us to efficiently multiply function field elements. As outlined in Section 5.3, we will do this "pointwise", which motivates the multi-point evaluation and interpolation algorithms that are presented in Section 5.5.1 and Section 5.5.2 respectively.

5.5.1 Multi-Point Evaluation

We now consider the following problem: given a function $a \in \mathcal{R}(A)$ for any divisor A, compute $\mathrm{ev}_E(a) = (a(E_1),\dots,a(E_N)) \in \mathbb{F}^N$ where $E = E_1 + \cdots + E_N$ is a divisor made up of rational places $E_1,\dots,E_N$ not in $\mathrm{supp}(A) \cup \{P_\infty\}$. We will be encountering this relationship between two divisors so often, that it will be convenient to use shorthand notation for it:

Definition 5.19. If A and $E = E_1 + \cdots + E_N$ are divisors, where $E_1,\dots,E_N$ are distinct rational places not in $\mathrm{supp}(A) \cup \{P_\infty\}$, then we will write

$$E = E_1 + \cdots + E_N \in \mathbb{D}(A)\ .$$

Without futher ado, we present Algorithm 13 for multi-point evaluation.

Algorithm 13 Evaluate$(a, E, A, \boldsymbol{x}, \boldsymbol{y})$

Input:

- Divisors A and $E = E_1 + \cdots + E_N \in \mathbb{D}(A)$,
- a function $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)} \in \text{Я}(A)$, where $a_i \in \mathbb{F}[x]$,
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\ldots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\ldots,N}^{i=0,\ldots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$.

Output:

- Evaluations $\mathrm{ev}_E(a) \in \mathbb{F}^N$.

1: **for** $i = 0, \ldots, \mu - 1$ **do**
2: $\quad (a_{i,1}, \ldots, a_{i,N}) \in \mathbb{F}^N \leftarrow (a_i(x_1), \ldots, a_i(x_N))$ ▷ Univariate MPE
3: **return** $\sum_{i=0}^{\mu-1} (a_{i,1} y_{i,1}, \ldots, a_{i,N} y_{i,N}) \in \mathbb{F}^N$

Lemma 5.20. *Algorithm 13 is correct and costs* $\widetilde{\mathcal{O}}(\mu N + \delta_A(a) + \deg A)$ *operations in* $\mathbb{F}$.

Proof. Correctness simply follows from the fact that for $j = 1, \ldots, N$

$$\sum_{i=0}^{\mu-1} a_{i,j} y_{i,j} = \sum_{i=0}^{\mu-1} a_i(x(E_j)) y_i^{(A)}(E_j) = \sum_{i=0}^{\mu-1} (a_i y_i^{(A)})(E_j) = a(E_j) \ .$$

For complexity, notice that the total cost of the for-loop on Line 1 amounts to that of evaluating each of the univariate polynomials $a_0, \ldots, a_{\mu-1} \in \mathbb{F}[x]$ on N points. According to Lemma 5.8,

$$\deg a_i \leqslant \frac{1}{\mu}(\delta_A(a) + \deg A) \quad \text{for } i = 0, \ldots, \mu - 1 \ ,$$

hence the total cost of the for-loop is bounded by

$$\widetilde{\mathcal{O}}(\mu(N + \max_i \deg a_i)) \subseteq \widetilde{\mathcal{O}}(\mu N + \delta_A(a) + \deg A) \ .$$

Line 3 costs $\mathcal{O}(\mu N)$, which is subsumed by the cost of the for-loop. □

5.5.2 Interpolation

Let us now turn our attention to interpolation – the inverse problem of multipoint evaluation. To attain an efficient solution, we will partition the evaluation points in a particular way:

Definition 5.21. Let $E_1, \ldots, E_N \neq P_\infty$ be pairwise distinct rational places, and let $E = E_1 + \cdots + E_N$. If $U_1, \ldots, U_\mu$ are effective divisors satisfying

1) $E = U_1 + \cdots + U_\mu$,
2) $\operatorname{supp} U_i \cap \operatorname{supp} U_k = \varnothing$ when $i \neq k$,
3) $|\deg U_i - \deg U_k| \leqslant 1$ for all i, k,
4) for any $E_j, E_k \in \operatorname{supp} U_i$ it holds that $x(E_j) = x(E_k) \iff E_j = E_k$,

then we will say that $U_1, \ldots, U_\mu$ is an *x-partition* of E.

In order to prove that an x-partition always exists, we will use the following:

Lemma 5.22. *If $\mathcal{P}$ is a set of places satisfying $x(P) = x(P')$ for all $P, P' \in \mathcal{P}$, then $|\mathcal{P}| \leqslant \mu$.*

Proof. If $\alpha = x(P)$ for every $P \in \mathcal{P}$, then $0 \neq x - \alpha \in \mathcal{L}(\mu P_\infty - \sum_{P \in \mathcal{P}} P)$. But if $|\mathcal{P}| > \mu$, then the above Riemann-Roch space has dimension zero. □

The proof of the following lemma gives an easy way of constructing an x-partition for any appropriate divisor E:

Lemma 5.23. *There exists an x-partition of any divisor $E = E_1 + \cdots E_N$, where $E_1, \ldots, E_N \neq P_\infty$ are pairwise distinct rational places.*

Proof. We use induction on N. The base case $N = 0$ is trivial, so consider the induction step: Assuming that $U_1, \ldots, U_\mu$ is an x-partition of $E - E_N$, let

$$U \in \{U_i\}_{i=1}^{\mu} \text{ and } V \in \{U_i \mid x(E_j) \neq x(E_N) \text{ for all } E_j \in \operatorname{supp} U_i,\ i = 1, \ldots, \mu\}$$

have minimal degrees in their respective sets, the latter of which is non-empty due to Lemma 5.22. If $\deg U = \deg V$, then an x-partition of E can be obtained by replacing V with $V + E_N$. On the other hand, if $\deg U = \deg V - 1$, then U contains a place P_U with $x(P_U) = x(E_N)$, and V contains a place P_V with $x(P_V) \neq x(E_j)$ for all $E_j \in U$. In this case, an x-partition of E can be obtained by replacing U with $U + E_N - P_U + P_V$, and V with $V - P_V + P_U$. □

Another ingredient that we will use in our interpolation algorithm is the homogeneous case of *simultaneous Hermite-Padé approximations*, which is captured in the following definition:

Definition 5.24. For any matrix $\boldsymbol{A} \in \mathbb{F}[x]^{\phi\times\theta}$ with columns $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\theta$ and any vector $\boldsymbol{u} = (u_1, \ldots, u_\theta) \in \mathbb{F}[x]^\theta$, we define the $\mathbb{F}[x]$-module

$$\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A}) = \{\boldsymbol{v} \in \mathbb{F}[x]^\phi \mid \boldsymbol{v} \cdot \boldsymbol{A}_k \equiv 0 \pmod{u_k} \text{ for } k = 1, \ldots, \theta\} .$$

In the context of Definition 5.24, we are interested in being able to compute a shifted Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$. Fortunately, this problem has been studied extensively in the literature [JNSV16, JNSV17, RS21]. The following is a direct adaptation of [RS21, Theorem 1.7].

Theorem 5.25 ([RS21, Theorem 1.7])**.** *If $\phi, \theta \in \mathbb{Z}_{>0}$ are integers such that $\phi \geqslant \theta$, then there exists an algorithm which for any $\boldsymbol{A} \in \mathbb{F}[x]^{\phi\times\theta}$, $\boldsymbol{u} \in (\mathbb{F}[x]\backslash\{0\})^\theta$ and $\boldsymbol{d} = (d_1, \ldots, d_\phi) \in \mathbb{Z}^\phi_{\geqslant 0}$ can compute a matrix $\boldsymbol{V} \in \mathbb{F}[x]^{\phi\times\phi}$ in $(-\boldsymbol{d})$-Popov form, whose rows form an $\mathbb{F}[x]$-basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$. Furthermore, if there exists a vector $\boldsymbol{v} = (v_1, \ldots, v_\phi) \in \mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$ satisfying the degree constraints $\deg v_t < d_t$ for $t = 1, \ldots, \phi$, then at least one row of $\boldsymbol{V}$ will also satisfy these constraints. The complexity of such an algorithm can be taken to be $\widetilde{\mathcal{O}}(\phi^{\omega-1}\theta d)$ operations in $\mathbb{F}$, where $d = \max_t d_t + \max_k \deg u_k$.*

In some of our use cases of Theorem 5.25, we will need to allow the shifts $d_1, \ldots, d_\phi$ to be non-integer – in particular, they will belong to $\frac{1}{\mu}\mathbb{Z} \subset \mathbb{Q}$. Fortunately, this has already been solved in [NB15] by simply permuting the columns in a very specific way:

Theorem 5.26 (Reformulation of Corollary 12 in [NB15])**.** *Let $\boldsymbol{V} \in \mathbb{F}[x]^{\gamma\times\phi}$ and $\boldsymbol{d} = (d_1/\mu, \ldots, d_\phi/\mu) \in (\frac{1}{\mu}\mathbb{Z})^\phi \subset \mathbb{Q}^\phi$, where $d_1, \ldots, d_\phi \in \mathbb{Z}$. If π is the permutation on $\{1, \ldots, \phi\}$ defined by*

$$\pi(i) > \pi(j) \iff \begin{array}{c} (d_i \operatorname{rem} \mu) > (d_j \operatorname{rem} \mu) \\ \textit{or} \\ (d_i \operatorname{rem} \mu) = (d_j \operatorname{rem} \mu) \textit{ and } i > j \end{array} ,$$

and $\Psi : \mathbb{F}[x]^\phi \to \mathbb{F}[x]^\phi$ is the map

$$(v_1, \ldots, v_\phi) \mapsto (x^{\lfloor d_{\pi(1)}/\mu \rfloor} v_{\pi(1)}, \ldots, x^{\lfloor d_{\pi(\phi)}/\mu \rfloor} v_{\pi(\phi)}) ,$$

then $\boldsymbol{V}$ is in $\boldsymbol{d}$-Popov form if and only if $\Psi(\boldsymbol{V})$ is in (unshifted) Popov form, where $\Psi(\boldsymbol{V}) \in \mathbb{F}[x]^{\gamma\times\phi}$ is the matrix created by applying Ψ to each row of $\boldsymbol{V}$.

Using the permutation defined in Theorem 5.26, we immediately obtain a stronger version of Theorem 5.25:

Corollary 5.27. *In the context of Theorem 5.25, we can allow $\boldsymbol{d} \in (\frac{1}{\mu}\mathbb{Z})^\phi$ and still find the sought matrix $\boldsymbol{V} \in \mathbb{F}[x]^{\phi\times\phi}$ in complexity $\widetilde{\mathcal{O}}(\phi^{\omega-1}\theta d)$ operations in $\mathbb{F}$, where $d = \max_t |d_t| + \max_k \deg u_k$.*

Proof. Write $\boldsymbol{d} = (\widetilde{d}_1/\mu, \dots, \widetilde{d}_\phi/\mu)$ with $\widetilde{d}_t \in \mathbb{Z}$, and notice that Theorem 5.26 implies that $\boldsymbol{V} \in \mathbb{F}[x]^{\phi\times\phi}$ is in $(-\boldsymbol{d})$-Popov form if and only if $\widetilde{\boldsymbol{V}} \in \mathbb{F}[x]^{\phi\times\phi}$ is in $(-\widetilde{\boldsymbol{d}})$-Popov form, where

$$\widetilde{\boldsymbol{d}} = (\lfloor \widetilde{d}_{\pi(1)}/\mu \rfloor, \dots, \lfloor \widetilde{d}_{\pi(\phi)}/\mu \rfloor) \in \mathbb{Z}^\phi \ ,$$

and $\widetilde{\boldsymbol{V}}$ is matrix obtained from $\boldsymbol{V}$ by permuting its columns using π from Theorem 5.26. By Theorem 5.25, for any matrix $\boldsymbol{A} \in \mathbb{F}[x]^{\phi\times\theta}$, we can compute the basis $\widetilde{\boldsymbol{V}} \in \mathbb{F}[x]^{\phi\times\phi}$ of $\mathcal{H}_{\boldsymbol{u}}(\widetilde{\boldsymbol{A}})$ in $(-\widetilde{\boldsymbol{d}})$-Popov form, where $\widetilde{\boldsymbol{A}} \in \mathbb{F}[x]^{\phi\times\theta}$ is obtained by permuting the rows of $\boldsymbol{A}$ by π, as long as the entries of $\widetilde{\boldsymbol{d}}$ are non-negative. By simply adding the constant $\max_t \lceil |\widetilde{d}_t|/\mu \rfloor$ to all coordinates of $\widetilde{\boldsymbol{d}}$, we can ensure that this is true without breaking the target complexity. Finally, it is trivial to obtain $\boldsymbol{V}$ from $\widetilde{\boldsymbol{V}}$ by applying π^{-1} to its columns. □

We proceed by reducing our interpolation problem to that of simultaneous Hermite-Padé approximations, which will allow us to solve it using Corollary 5.27.

Lemma 5.28. *Given divisors A and $E = E_1 + \cdots + E_N \in \mathbb{D}(A)$, interpolation values $(w_1, \dots, w_N) \in \mathbb{F}^N$ and an x-partition $U_1, \dots, U_\mu$ of E, consider a row-vector $\boldsymbol{T} = [T_k] \in \mathbb{F}[x]^{1\times\mu}$ and a matrix $\boldsymbol{S} = [S_{i,k}] \in \mathbb{F}[x]^{\mu\times\mu}$, where*

$$T_k(x(E_j)) = -w_j \quad \textit{and} \quad S_{i,k}(x(E_j)) = y_i^{(A)}(E_j) \quad \textit{for all} \quad E_j \in \operatorname{supp}(U_k) \ .$$

If $\boldsymbol{d} = (d_0, \dots, d_{\mu-1}, 0) \in (\frac{1}{\mu}\mathbb{Z})^{\mu+1}$ with $d_i = \frac{1}{\mu}(\deg E + 2g - \deg A - \delta_A(y_i^{(A)}))$, and $\boldsymbol{u} = (u_1, \dots, u_\mu) \in \mathbb{F}[x]^\mu$ with $u_k = \prod_{E_j \in \operatorname{supp}(U_k)} (x - x(E_j))$, then in the $(-\boldsymbol{d})$-Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$, where

$$\boldsymbol{A} = \left[\frac{\boldsymbol{S}}{\boldsymbol{T}}\right] \in \mathbb{F}^{(\mu+1)\times\mu} \ ,$$

there exists a vector $\boldsymbol{a} = (a_0, \dots, a_{\mu-1}, 1) \in \mathbb{F}[x]^{\mu+1}$ with $\deg a_i < d_i$. Moreover, if $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)}$, then $\delta_A(a) \leqslant \deg E + 2g - 1 - \deg A$ and $a(E_j) = w_j$ for $j = 1, \dots, N$.

Proof. According to Lemma 1.18, there exists a function $b \in \text{Я}(A)$ with

$$\delta_A(b) \leqslant \deg E + 2g - 1 - \deg A$$

such that $b(E_j) = w_j$ for $j = 1, \dots, N$. If we write $b = \sum_{i=0}^{\mu-1} b_i y_i^{(A)}$, where $b_i \in \mathbb{F}[x]$, then it follows from Lemma 5.8 that

$$\deg b_i \leqslant \frac{1}{\mu}(\delta_A(a) - \delta_A(y_i^{(A)})) = \frac{1}{\mu}(\deg E + 2g - 1 - \deg A - \delta_A(y_i^{(A)})) < d_i \ .$$

We claim that $\boldsymbol{b} := (b_0, \ldots, b_{\mu-1}, 1) \in \mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$. To see this let

$$c_k = \sum_{i=0}^{\mu-1} b_i S_{i,k} + T_k \in \mathbb{F}[x] \quad \text{for } k = 1, \ldots, \mu \ ,$$

and observe that for any $E_j \in U_k$ it holds that

$$c_k(x(E_j)) = \sum_{i=0}^{\mu-1} b_i(x(E_j)) y_i^{(A)}(E_j) - w_j = b(E_j) - w_j = 0 \ .$$

This implies that $\boldsymbol{b}\boldsymbol{A}_k = c_k \equiv 0 \pmod{u_k}$, where $\boldsymbol{A}_k \in \mathbb{F}[x]^{(\mu+1)\times 1}$ denotes the k-th column of $\boldsymbol{A}$. But then indeed, $\boldsymbol{b} \in \mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$ by definition.

Note that the $(-\boldsymbol{d})$-leading position of $\boldsymbol{b}$ is the last one, which implies that the $(-\boldsymbol{d})$-Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$ will contain a vector $\boldsymbol{a} = (a_0, \ldots, a_{\mu-1}, a_\mu)$ with the same leading position – in particular $a_\mu \neq 0$. Since $\boldsymbol{a}$ has minimal $(-\boldsymbol{d})$-degree among all vectors in $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$ whose leading position is the last one, we conclude that $\boldsymbol{a}$ satisfies the same degree constraints as $\boldsymbol{b}$.

To conclude the proof, observe that

$$\begin{aligned}\delta_A(a) &= \max_i(\delta(a_i) + \delta_A(y_i^{(A)})) = \max_i(\mu \deg a_i + \delta_A(y_i^{(A)})) \\ &< \max_i(\mu d_i + \delta_A(y_i^{(A)})) = \deg E + 2g - \deg A \ ,\end{aligned}$$

and that for any $E_j \in U_k$, where $k = 1, \ldots, \mu$, it holds that

$$\begin{aligned}a(E_j) - w_j &= \sum_{i=0}^{\mu-1} a_i(x(E_j)) y_i^{(A)}(E_j) - w_j \\ &= \sum_{i=0}^{\mu-1} a_i(x(E_j)) S_{i,k}(x(E_j)) + T_k(x(E_j)) \\ &= (\boldsymbol{a}\boldsymbol{A}_k)(x(E_j)) = 0 \ ,\end{aligned}$$

since $\boldsymbol{a} \in \mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$. Consequently, $a(E_j) = w_j$ for $j = 1, \ldots, N$. □

It is straightforward to extract an interpolation algorithm from Lemma 5.28 – we present it in Algorithm 14.

Algorithm 14 Interpolate$(\boldsymbol{w}, E, A, \boldsymbol{x}, \boldsymbol{y})$

Input:

- Divisors A and $E = E_1 + \cdots + E_N \in \mathbb{D}(A)$,
- interpolation values $\boldsymbol{w} = (w_1, \ldots, w_N) \in \mathbb{F}^N$,
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\ldots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\ldots,N}^{i=0,\ldots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$.

Output:

- $a \in \text{Я}(A)$ satisfying $a(E_j) = w_j$ for $j = 1, \ldots, N$ and $\delta_A(a) \leqslant \deg E + 2g - 1 - \deg A$.

1: $U_1, \ldots, U_\mu \leftarrow$ an x-partition of E
2: $\boldsymbol{S} = [S_{i,k}] \in \mathbb{F}[x]^{\mu\times\mu} \leftarrow$ matrix with $S_{i,k}(x_j) = y_{i,j}$ for all $E_j \in U_k$
3: $\boldsymbol{T} = [T_k] \in \mathbb{F}[x]^{\mu} \leftarrow$ row vector with $T_k(x_j) = -w_j$ for all $E_j \in U_k$
4: $\boldsymbol{u} = (u_1, \ldots, u_\mu) \in \mathbb{F}[x]^{\mu} \leftarrow$ vector with $u_k = \prod_{E_j \in U_k}(x - x_j)$
5: $\boldsymbol{d} = (d_0, \ldots, d_{\mu-1}, 0) \in (\frac{1}{\mu}\mathbb{Z})^{\mu+1} \leftarrow$ vector with
$d_i = \frac{1}{\mu}(\deg E + 2g - \deg A - \delta_A(y_i^{(A)}))$
6: $\boldsymbol{P} \in \mathbb{F}[x]^{(\mu+1)\times(\mu+1)} \leftarrow (-\boldsymbol{d})$-Popov basis matrix of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$,
where $\boldsymbol{A} = \left[\dfrac{\boldsymbol{S}}{\boldsymbol{T}}\right] \in \mathbb{F}^{(\mu+1)\times\mu}$
7: $\boldsymbol{a} = (a_0, \ldots, a_{\mu-1}, 1) \in \mathbb{F}[x]^{\mu+1} \leftarrow$ a row of $\boldsymbol{P}$ with $\deg a_i < d_i$ for all i ▷ last row
8: **return** $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)}$

Proposition 5.29. *Algorithm 14 is correct and has complexity* $\tilde{\mathcal{O}}(\mu^{\omega-1}(N+g))$.

Proof. Correctness follows directly from Lemma 5.28. For complexity, note that for all i and k, we have $\deg u_k = |U_k| \leqslant \lceil N/\mu \rceil$, and we can choose

$$\deg S_{i,k}, \deg T_k < \lceil N/\mu \rceil \ .$$

Step 2 costs $\tilde{\mathcal{O}}(\mu^2 N/\mu) = \tilde{\mathcal{O}}(\mu N)$, while Steps 3 and 4 cost $\tilde{\mathcal{O}}(\mu N/\mu) = \tilde{\mathcal{O}}(N)$. The bottleneck lies in Step 6, which according to Corollary 5.27 costs

$$\tilde{\mathcal{O}}(\mu^{\omega-1}\mu(\max_i d_i + \max_k \deg u_k)) \subseteq \tilde{\mathcal{O}}(\mu^{\omega}(\frac{\deg E + 2g}{\mu} + \frac{N}{\mu})) = \tilde{\mathcal{O}}(\mu^{\omega-1}(N+g)) \ .$$

Here we used that $d_i \leqslant (\deg E + 2g)/\mu$, since $\deg A + \delta_A(y_i^{(A)}) \geqslant 0$ by Lemma 5.6. □

Before we proceed to the next section, where we will use both interpolation and multi-point evaluation in order to compute a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ over

Я, let us make a final comment on the output of Algorithm 14. We know from Lemma 5.28 that the returned $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)} = \mathsf{Interpolate}(\boldsymbol{w}, E, A, \boldsymbol{x}, \boldsymbol{y})$ satisfies $\delta_A(a) \leqslant \deg E + 2g - 1 - \deg A$, however, more can be said: Although this bound is sharp in the sense that it is the best that one can expect in general, in specific cases it can be beaten. A convenient property of Algorithm 14 is that its output a is guaranteed to have the smallest possible value of $\delta_A(a)$ among all possible interpolating functions, which is shown in the following lemma.

Lemma 5.30. *In the context of Algorithm 14, the output $a \in Я(A)$ satisfies $\delta_A(a) \leqslant \delta_A(b)$ for all functions $b \in Я(A)$ with $b(E_j) = w_j$ for $j = 1, \ldots, N$.*

Proof. Consider the map φ which sends any function $b = \sum_{i=0}^{\mu-1} b_i y_i^{(A)} \in Я(A)$ to the vector $(b_0, \ldots, b_{\mu-1}) \in \mathbb{F}[x]^\mu$, and observe that if $b(E_j) = w_j$ for all j, then $\varphi(a - b)$ is in the row space of the matrix $\widetilde{\boldsymbol{P}} \in \mathbb{F}[x]^{\mu\times\mu}$ obtained from the first μ rows and columns of $\boldsymbol{P}$. It is clear that $\widetilde{\boldsymbol{P}}$ is in $(-\widetilde{\boldsymbol{d}})$-Popov form, where $\widetilde{\boldsymbol{d}} = (d_0, \ldots, d_{\mu-1})$, and that each entry in $\varphi(a)$ has degree strictly smaller than the maximal degree of the corresponding column in $\widetilde{\boldsymbol{P}}$: otherwise $\boldsymbol{P}$ would not be in $(-\boldsymbol{d})$-Popov form. But then, if $\varphi(b)$ satisfies the same degree constraints as $\varphi(a)$, so does $\varphi(a - b)$, and it follows from Proposition 1.6 on page 8 that $\varphi(a - b) = 0$, implying that $a = b$ (see also [Kai80, Theorem 6.3-15] or [Nei16, Lemma 1.24]). □

5.5.3 Computing a Я-generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$

In this section we show how to efficiently compute the generating set $\{B_v^{(u)}\}$ of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ over Я, as given by Corollary 5.15. As we have mentioned previously, we will carry out the required multiplications of function field elements pointwise – relying on Algorithm 13 and Algorithm 14 for multi-point evaluation and interpolation respectively. The following lemma essentially tells us how many evaluation points we should use in order to guarantee that no information is lost in the process.

Lemma 5.31. *Let $a \in Я(A)$ and $b \in Я(B)$, where A and B are divisors, and let $E = E_1 + \cdots + E_N \in \mathbb{D}(A) \cap \mathbb{D}(B)$. If $c \in Я(A + B)$ satisfies $c(E_j) = a(E_j)b(E_j)$ for $j = 1, \ldots, N$ and $\delta_{A+B}(c) < N - \deg(A + B)$, then $c = ab$.*

Proof. Note that $c \in \mathcal{L}(C)$, where $C = \delta_{A+B}(c)P_\infty + A + B$. Using the notation as well as the claim of Lemma 1.17, we get that $\mathrm{ev}_{E,C}$ is injective. The sought conclusion follows from $\mathrm{ev}_{E,C}(c) = \mathrm{ev}_{E,C}(ab)$. □

As can be seen in Algorithm 15, it is now straightforward to compute the sought generators of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over Я.

Algorithm 15 $\mathsf{Generators}_{\text{Я}}(\boldsymbol{r}, D, G, E, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{g})$

Input:

- Received word $\boldsymbol{r} \in \mathbb{F}^n$,
- divisors D and G for the code $\mathcal{C}_{\mathcal{L}}(D,G)$,
- a divisor $E = E_1 + \cdots + E_N \in \mathbb{D}(G)$ with $N \geqslant (\ell+1)\deg G + 4g + (s+1)n$,
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\ldots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\ldots,N}^{i=0,\ldots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{g} = (g_{v,j}^{(u)})_{v=1,2,\ j=1,\ldots,N}^{u=0,\ldots,\ell}$, where $g_{v,j}^{(u)} = g_v^{(u)}(E_j) \in \mathbb{F}$, $\langle g_1^{(u)}, g_2^{(u)} \rangle_{\text{Я}} = \text{Я}(G_u)$ and $\delta_{G_u}(g_v^{(u)}) \leqslant 4g - 1 + (u+1)\deg(G) + (s+1)n$, as in Corollary 5.15.

Output:

- $(B_v^{(u)})_{v=1,2}^{u=0,\ldots,\ell}$ such that $\langle B_v^{(u)} \rangle_{\text{Я}} = \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$.

$R \in \text{Я}(G) \leftarrow \mathsf{Interpolate}(\boldsymbol{r}, D, G, \boldsymbol{x}, \boldsymbol{y})$ ▷ Algorithm 14
$(\rho_j^{(0)})_{j=1}^N \in \mathbb{F}^N \leftarrow (1, \ldots, 1)$
$(\rho_j^{(1)})_{j=1}^N \in \mathbb{F}^N \leftarrow \mathsf{Evaluate}(-R, E, G, \boldsymbol{x}, \boldsymbol{y})$ ▷ Algorithm 13
for $u = 2, \ldots, \ell$ **do**
  $(\rho_j^{(u)})_{j=1}^N \in \mathbb{F}^N \leftarrow (\rho_j^{(1)} \rho_j^{(u-1)})_{j=1}^N$
for $u = 0, \ldots, \ell$, $v = 1, 2$ and $t = 0, \ldots, u$ **do**
  $\boldsymbol{c}_{v,t}^{(u)} \in \mathbb{F}^N \leftarrow (\rho_j^{(u-t)} g_{v,j}^{(u)})_{j=1,\ldots,N}$
  $c_{v,t}^{(u)} \in \text{Я}(-tG) \leftarrow \mathsf{Interpolate}(\boldsymbol{c}_{v,t}^{(u)}, E, -tG, \boldsymbol{x}, \boldsymbol{y})$
for $u = 0, \ldots, \ell$ and $v = 1, 2$ **do**
  $B_v^{(u)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G) \leftarrow \sum_{t=0}^{u} \binom{u}{t} z^t c_{v,t}^{(u)}$
return $(B_v^{(u)})_{v=1,2}^{u=0,\ldots,\ell}$

Proposition 5.32. *Algorithm 15 is correct and costs* $\tilde{\mathcal{O}}(\mu^{\omega-1}\ell^3(n+g))$ *operations in* $\mathbb{F}$.

Proof. Note that $\delta_G(R) \leqslant n + 2g - 1 - \deg G$ due to Proposition 5.29, which

together with the given upper bound for $\delta_{G_u}(g_v^{(u)})$ implies that

$$\begin{aligned}
\delta_{-tG}(R^{u-t}g_v^{(u)}) &= \delta_{(u-t)G+G_u}(R^{u-t}g_v^{(u)}) \\
&\leqslant (u-t)(n+2g-1-\deg G)+4g-1+(u+1)\deg G+(s+1)n \\
&= (t+1)\deg G+(u-t+2)(2g-1)+1+(s+1)n \qquad (5.3)\\
&= (t+1)(\deg G-2g+1)+(u+3)(2g-1)+1+(s+1)n \\
&\leqslant (\ell+1)(\deg G-2g+1)+(\ell+3)(2g-1)+1+(s+1)n \\
&= (\ell+1)\deg G+2(2g-1)+1+(s+1)n \\
&< (\ell+1)\deg G+4g+(s+1)n\ .
\end{aligned}$$

Lemma 5.30 then implies that $\mathsf{Interpolate}(\boldsymbol{c}_{v,t}^{(u)}, E, -tG)$ will output a function $c_{v,t}^{(u)} \in \text{Я}(G)$ satisfying $\delta_{-tG}(c_{v,t}^{(u)}) < (\ell+1)\deg G+4g+(s+1)n$.

To complete the correctness proof, we consider Lemma 5.31 for the divisors $A = (u-t)G$, $B = -uG$ and the functions $a = (-R)^{u-t}$, $b = g_v^{(u)}$, and $c = c_{v,t}^{(u)}$. By construction, it is clear that for all $E_j \in \operatorname{supp} E$ we have that $c_{v,t}^{(u)}(E_j) = (-R)^{u-t}(E_j)g_v^{(u)}(E_j)$. Moreover, $\deg E \geqslant (\ell+1)\deg G+4g+(s+1)n$, whence $\delta_{-tG}(c_{v,t}^{(u)}) < \deg E \leqslant \deg E - \deg(-tG)$. Consequently, Lemma 5.31 implies that $c_{v,t}^{(u)} = (-R)^{u-t}g_v^{(u)}$.

The complexity of the algorithm is dominated by the for-loop in Lines 6–8. The $\mathcal{O}(\ell^2)$ calls of $\mathsf{Interpolate}(\boldsymbol{c}_{v,t}^{(u)}, E, -tG)$ cost $\widetilde{\mathcal{O}}(\mu^{\omega-1}\ell(n+g))$ operations each, amounting to the total cost of $\widetilde{\mathcal{O}}(\mu^{\omega-1}\ell^3(n+g))$. □

Remark 5.33. The generating set $\{\widetilde{B}_v^{(u)}\}$ from Remark 5.18 can be computed slightly faster. Indeed, in these generators, the needed powers $(-R)^u$ have the range $u = 0, \ldots, s$, so the for-loop in Lines 6–8 performs $\mathcal{O}(s\ell)$ calls to $\mathsf{Interpolate}(\boldsymbol{c}_{v,t}^{(u)}, E, -tG)$, resulting in the total cost of $\widetilde{\mathcal{O}}(\mu^{\omega-1}s\ell^2(n+g))$.

5.5.4 Computing an $\mathbb{F}[x]$-generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$

In the previous section, we saw how to efficiently compute the generating set $\{B_v^{(u)}\}$ of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over Я, as in Corollary 5.15. Following the strategy outlined in Section 5.4.2, the next logical step is to compute the set of products $\{y_i B_v^{(u)}\}$, which generates $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ over $\mathbb{F}[x]$ according to Corollary 5.17. Naturally, we now consider the following problem: given a function $a \in \text{Я}(A)$ for some divisor A, compute $y_0 a, \ldots, y_{\mu-1}a \in \text{Я}(A)$. Indeed, any algorithm for solving this can be used to obtain the sought $\mathbb{F}[x]$-basis of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$

by simply calling it on the z-coefficients of the $B_v^{(u)}$. We could in principle compute the products $y_i a$ individually, relying on pointwise multiplication as we did in Algorithm 15; however, this would break our target complexity because computing each $y_i B_v^{(u)}$ this way would cost $\tilde{\mathcal{O}}(\mu^{\omega-1}\ell^2(n+g))$ operations, and we need to compute $2\mu(\ell+1)$ such terms in total.

In this section we present a more efficient approach which computes all of the $y_0 a, \ldots, y_{\mu-1} a$ simultaneously by taking advantage of the fact that they form an $\mathbb{F}[x]$-basis of $\langle a \rangle_{\text{Я}}$. Since there is an obvious $\mathbb{F}[x]$-isomorphism between the modules $\langle a \rangle_{\text{Я}}$ and $\langle z - a \rangle_{\text{Я}}$, where z is an indeterminate, obtaining a basis of the former is easily reduced to obtaining one of the latter. As we will see, $\langle z - a \rangle_{\text{Я}}$ turns out to be closely related to a particular instance of simultaneous Hermite-Padé approximations, which will allow us to compute the sought $\mathbb{F}[x]$-basis using Theorem 5.25 as well as Corollary 5.27. In order to explain this precisely, we proceed with a few technical results.

If $H(z) \in F[z]$, $\alpha \in \mathbb{F}$ and P is a rational place that is not a pole of any of the coefficients of $H(z)$, then we will denote by $H(P, \alpha) \in \mathbb{F}$ the evaluation of $H(\alpha) \in F$ at P. Furthermore, for any divisors A and $E = E_1 + \cdots + E_N \in \mathbb{D}(A)$, and any function $a \in \text{Я}(A)$, we define the $\mathbb{F}[x]$-module

$$\mathcal{N}_{A,E}(a) = \{H = H_0 + H_1 z \in \text{Я}(A) \oplus z\text{Я} \mid H(P, a(P)) = 0 \text{ for all } P \in \operatorname{supp}(E)\} .$$

To avoid repetition, in the next few lemmas we will treat A, E and a as fixed.

It is clear that $\langle z - a \rangle_{\text{Я}}$ is a submodule of $\mathcal{N}_{A,E}(a)$. The following result shows that it is a particularly well-behaved one:

Lemma 5.34. *If $H = H_0 + zH_1 \in \mathcal{N}_{A,E}(a)$ satisfies*

$$\max\{\delta_A(H_0), \delta(H_1) + \delta_A(a)\} < \deg E - \deg A ,$$

then $H(a) = 0$, i.e. $H \in \langle z - a \rangle_{\text{Я}}$.

Proof. Since $H \in \mathcal{N}_{A,E}(a)$, then $H(a) \in \text{Я}(A)$. Moreover, by definition of δ_A, $H(a) \in \mathcal{L}(\delta_A(H(a))P_\infty + A)$, and since $H(a)(E_j) = 0$ for $j = 1, \ldots, N$, then $H(a) \in \mathcal{L}(\delta_A(H(a))P_\infty + A - E)$ because $E \in \mathbb{D}(A)$. But we also know that

$$\delta_A(H(a)) \leqslant \max\{\delta_A(H_0), \delta(H_1) + \delta_A(a)\} < \deg E - \deg A ,$$

which implies that the aforementioned Riemann-Roch space is trivial. □

Lemma 5.34 essentially tells us that "small" elements in $\mathcal{N}_{A,E}(a)$ necessarily also belong to $\langle z - a \rangle_{\text{Я}}$. In particular, this means that we might be able to recognize

the sought $\mathbb{F}[x]$-basis of $\langle z-a\rangle_{\text{Я}}$ as a subset of some appropriately "reduced" basis of $\mathcal{N}_{A,E}(a)$. In the following lemma we relate $\mathcal{N}_{A,E}(a)$ to simultaneous Hermite-Padé approximations, which will allow us to obtain such a reduced basis using Theorem 5.25 and Corollary 5.27.

Lemma 5.35. *Let $U_1,\dots,U_\mu$ be an x-partition of E, and consider matrices $\boldsymbol{S}=[S_{i,k}],\boldsymbol{T}=[T_{i,k}]$ in $\mathbb{F}[x]^{\mu\times\mu}$ satisfying*

$$S_{i,k}(x(E_j))=y_i^{(A)}(E_j)\quad\text{and}\quad T_{i,k}(x(E_j))=a(E_j)y_i(E_j)\quad\text{for } E_j\in U_k\ .$$

If $\boldsymbol{u}=(u_1,\dots,u_\mu)\in\mathbb{F}[x]^\mu$, where $u_k=\prod_{E_j\in\operatorname{supp}U_k}(x-x(E_j))$, then the map

$$\psi:\sum_{i=0}^{\mu-1}(s_iy_i^{(A)}+t_izy_i)\mapsto(s_0,\dots,s_{\mu-1},t_0,\dots,t_{\mu-1})$$

is an $\mathbb{F}[x]$-isomorphism between $\mathcal{N}_{A,E}(a)$ and $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$, where

$$\boldsymbol{A}=\left[\frac{\boldsymbol{S}}{\boldsymbol{T}}\right]\in\mathbb{F}^{2\mu\times\mu}\ .$$

Proof. Clearly ψ is an $\mathbb{F}[x]$-isomorphism between $\text{Я}(A)\oplus z\text{Я}$ and $\mathbb{F}[x]^{2\mu}$, so it suffices to show that for any $H\in\text{Я}(A)\oplus z\text{Я}$ it holds that $H\in\mathcal{N}_{A,E}(a)$ if and only if $\psi(H)\in\mathcal{H}_{\boldsymbol{q}}(\boldsymbol{A})$, or in other words that $H(E_j,a(E_j))=0$ for all $E_j\in\operatorname{supp}U_k$, $k=1,\dots,\mu$ if and only if $\psi(H)\cdot\boldsymbol{A}_k\equiv 0 \mod u_k$, $k=1,\dots,\mu$, where $\boldsymbol{A}_k$ denotes the k-th column of $\boldsymbol{A}$. But this is necessarily true, since for every $E_j\in U_k$ the following identity holds, where $\alpha=x(E_j)$:

$$\begin{aligned}H(E_j,a(E_j))&=\sum_{i=1}^{\mu}\big(s_i(\alpha)y_i^{(A)}(E_j)+t_i(\alpha)a(E_j)y_i(E_j)\big)\\&=\sum_{i=1}^{\mu}\big(s_i(\alpha)S_{i,k}(\alpha)+t_i(\alpha)T_{i,k}(\alpha)\big)=(\psi(H)\cdot\boldsymbol{A}_k)(\alpha)\ .\end{aligned}$$

□

In the context of Lemma 5.35, since $\mathcal{N}_{A,E}(a)$ is isomorphic to $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$, then $\langle z-a\rangle_{\text{Я}}$ is isomorphic to some submodule of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$. In the following lemma, we show that the sought $\mathbb{F}[x]$-basis of $\langle z-a\rangle_{\text{Я}}$ can be extracted from a certain shifted Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$.

Lemma 5.36. *In the context of Lemma 5.35, if $\boldsymbol{P}\in\mathbb{F}[x]^{2\mu\times2\mu}$ is the $\boldsymbol{d}$-Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})=\psi(\mathcal{N}_{A,E}(a))$, where $\deg E\geqslant 2g+\mu+\delta_A(a)+\deg A$ and*

$$\boldsymbol{d}=\frac{1}{\mu}\big(\delta_A(y_0^{(A)}),\dots,\delta_A(y_{\mu-1}^{(A)}),\delta(y_0)+\delta_A(a),\dots,\delta(y_{\mu-1})+\delta_A(a)\big)\in(\tfrac{1}{\mu}\mathbb{Z})^{2\mu}\ ,$$

then exactly μ rows of $\boldsymbol{P}$ have $\boldsymbol{d}$-degree less than $\frac{1}{\mu}(\deg E - \deg A)$. Furthermore, if $\widetilde{\boldsymbol{P}} \in \mathbb{F}[x]^{\mu\times 2\mu}$ is the submatrix of $\boldsymbol{P}$ consisting of these rows, then the k-th row of $\widetilde{\boldsymbol{P}}$ is $\psi(Y_k)$ for $k = 1, \ldots, \mu$, where

$$Y_k = -ay_{k-1} + zy_{k-1} \in \langle z - a\rangle_{\text{Я}} \subset \mathcal{N}_{A,E}(a) \ .$$

Consequently, if $\widetilde{\boldsymbol{P}} = [\widetilde{\boldsymbol{P}}_1 | \widetilde{\boldsymbol{P}}_2]$, where $\widetilde{\boldsymbol{P}}_1, \widetilde{\boldsymbol{P}}_2 \in \mathbb{F}[x]^{\mu\times\mu}$, then

$$ay_{k-1} = -\sum_{i=0}^{\mu-1} p_{k,i} y_i^{(A)} \ ,$$

where $(p_{k,0}, \ldots, p_{k,\mu-1})$ denotes the k-th row of $\widetilde{\boldsymbol{P}}_1$.

Proof. For any $H = H_0 + zH_1 \in \mathcal{N}_{A,E}(a)$, where $H_0 = \sum_{i=0}^{\mu-1} s_i y_i^{(A)} \in \text{Я}(A)$ and $H_1 = \sum_{i=0}^{\mu-1} t_i y_i \in \text{Я}$ with $s_i, t_i \in \mathbb{F}[x]$, it holds that

$$\begin{aligned}\deg_{\boldsymbol{d}} \psi(H) &= \max\Big\{ \max_i \big\{ \deg s_i + \frac{\delta_A(y_i^{(A)})}{\mu} \big\}, \max_i \big\{ \deg t_i + \frac{\delta(y_i) + \delta_A(a)}{\mu} \big\} \Big\} \\ &= \frac{1}{\mu} \max\{\delta_A(H_0), \delta(H_1) + \delta_A(a)\} \ .\end{aligned}$$

It then follows from Lemma 5.34 that

$$\deg_{\boldsymbol{d}} \psi(H) < \frac{1}{\mu}(\deg E - \deg A) \implies H \in \langle z - a\rangle_{\text{Я}} \ ,$$

which shows that at most μ rows of $\boldsymbol{P}$ can have $\boldsymbol{d}$-degree strictly less than $\frac{1}{\mu}(\deg E - \deg A)$ because $\langle z - a\rangle_{\text{Я}}$ has rank μ as an $\mathbb{F}[x]$-module. On the other hand, since $Y_1, \ldots, Y_\mu$ are linearly independent over $\mathbb{F}[x]$, and since

$$\deg_{\boldsymbol{d}} \psi(Y_k) = \frac{1}{\mu}(\delta(y_{k-1}) + \delta_A(a)) < \frac{1}{\mu}(\delta_A(a) + 2g + \mu) \leqslant \frac{1}{\mu}(\deg E - \deg A)$$

for $k = 1, \ldots, \mu$, where the strict inequality is due to Lemma 5.6, then at least μ rows of $\boldsymbol{P}$ have $\boldsymbol{d}$-degree less than $\frac{1}{\mu}(\deg E - \deg A)$ because $\boldsymbol{P}$ is $\boldsymbol{d}$-row reduced, which proves the first claim of the lemma.

For the second claim, it is sufficient to show that the $\boldsymbol{d}$-pivot index of $\psi(Y_k)$ is $\mu + k$, since this would imply that the matrix whose rows are $\psi(Y_k)$ is in $\boldsymbol{d}$-Popov form. To see this, write $Y_k = -\sum_{i=0}^{\mu-1} w_i y_i^{(A)} + zy_{k-1}$, where $w_i \in \mathbb{F}[x]$, and note that $Y_k(a) = 0$ implies that

$$\max_i \delta_A(w_i y_i^{(A)}) = \delta_A(\sum_{i=1}^{\mu} w_i y_i^{(A)}) = \delta_A(ay_{k-1}) = \delta(y_{k-1}) + \delta_A(a) \ .$$

Consequently, $\deg_{\boldsymbol{d}} \psi(Y_k) = \frac{1}{\mu}(\delta(y_{k-1}) + \delta_A(a))$, which shows that $\mu + k$ is indeed the $\boldsymbol{d}$-pivot index of $\psi(Y_k)$. $\square$

Lemma 5.36 immediately gives rise to Algorithm 16 as a way of computing the products $ay_0, \ldots, ay_{\mu-1} \in \text{Я}(A)$ efficiently.

Algorithm 16 BasisProducts$(a, E, A, \boldsymbol{x}, \boldsymbol{y})$

Input:

- A divisor A,
- a function $a \in \text{Я}(A)$,
- a divisor $E = E_1 + \cdots + E_N \in \mathbb{D}(A)$ with $\deg E \geqslant \deg A + \delta_A(a) + 2g + \mu$,
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\ldots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\ldots,N}^{i=0,\ldots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$.

Output:

- Products $(ay_0, \cdots, ay_{\mu-1})$, where each $ay_i \in \text{Я}(A)$.

1: **if** $a = 0$ **then**
2: **return** $(0, \ldots, 0)$
3: $U_1, \ldots, U_\mu \leftarrow$ an x-partition of E
4: $\boldsymbol{S} = [S_{i,k}] \in \mathbb{F}[x]^{\mu \times \mu} \leftarrow$ matrix with $S_{i,k}(x_j) = y_{i,j}$ for $E_j \in U_k$
5: $\boldsymbol{T} = [T_{i,k}] \in \mathbb{F}[x]^{\mu \times \mu} \leftarrow$ matrix with $T_{i,k}(x_j) = a(E_j) y_{i,j}$ for $E_j \in U_k$
6: $\boldsymbol{u} = (u_1, \ldots, u_\mu) \in \mathbb{F}[x]^\mu \leftarrow$ vector with $u_k = \prod_{E_j \in U_k} (x - x(E_j))$
7: $\boldsymbol{d} \in \mathbb{Q}^{2\mu} \leftarrow \frac{1}{\mu}\big(\delta_A(y_0^{(A)}), \ldots, \delta_A(y_{\mu-1}^{(A)}), \delta(y_0) + \delta_A(a), \ldots, \delta(y_{\mu-1}) + \delta_A(a)\big)$
8: $\boldsymbol{P} \in \mathbb{F}[x]^{2\mu \times 2\mu} \leftarrow \boldsymbol{d}$-Popov basis of $\mathcal{H}_{\boldsymbol{u}}(\boldsymbol{A})$, where $\boldsymbol{A} = \left[\begin{array}{c} \boldsymbol{S} \\ \hline \boldsymbol{T} \end{array}\right] \in \mathbb{F}[x]^{2\mu \times \mu}$
9: $[\widetilde{\boldsymbol{P}}_1 | \widetilde{\boldsymbol{P}}_2] \in \mathbb{F}[x]^{\mu \times 2\mu} \leftarrow$ the submatrix of $\boldsymbol{P}$ consisting of all rows with $\boldsymbol{d}$-degree less than $\frac{1}{\mu}(\deg E - \deg A)$, where $\widetilde{\boldsymbol{P}}_1, \widetilde{\boldsymbol{P}}_2 \in \mathbb{F}[x]^{\mu \times \mu}$
10: **for** $k = 1, \ldots, \mu$ **do**
11: $(p_{k,0}, \ldots, p_{k,\mu-1}) \in \mathbb{F}[x]^\mu \leftarrow k$-th row of $\boldsymbol{P}_1$
12: $a_k \in \text{Я}(A) \leftarrow -\sum_{i=0}^{\mu-1} p_{k,i} y_i^{(A)}$
13: **return** $(a_1, \ldots, a_\mu)$

Lemma 5.37. *Algorithm 16 is correct and costs* $\widetilde{\mathcal{O}}(\mu^{\omega-1}(N + |\deg A|))$ *operations in* $\mathbb{F}$.

Proof. Correctness is given by Lemma 5.36. For complexity, simply note that the computational bottleneck lies in Step 8, in which case $\delta_A(a) \geqslant -\deg A$ because a is nonzero and $a \in \mathcal{L}(\delta_A(a)P_\infty + A)$. By assumption, we have that $N = \deg E \geqslant \deg A + \delta_A(a) + 2g + \mu$, hence by Lemma 5.6

$$\begin{aligned} -\deg A \leqslant \delta_A(y_i^{(A)}) &\leqslant 2g - 1 - \deg A + \mu \\ &< \deg E - 2\deg A - \delta_A(a) \leqslant \deg E - \deg A\ . \end{aligned}$$

Since $\deg u_k \leqslant N/\mu$ for $k = 1, \ldots, \mu$, then the total complexity of the algorithm is given by Corollary 5.27 as

$$\widetilde{\mathcal{O}}\big(\mu^{\omega-1} \max\{|\deg E|, |\deg E - \deg A|, |\deg A|\}\big) \subseteq \widetilde{\mathcal{O}}(\mu^{\omega-1}(N + |\deg A|))$$

operations in $\mathbb{F}$. □

With Algorithm 16 at our disposal, we can compute an $\mathbb{F}[x]$-basis of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ within our target complexity. The details are presented in Algorithm 17.

Algorithm 17 $\mathsf{Generators}_{\mathbb{F}[x]}(\boldsymbol{r}, D, G, E, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{g})$

Input:

- Received word $\boldsymbol{r} \in \mathbb{F}^n$,
- divisors D and G for the code $\mathcal{C}_{\mathcal{L}}(D, G)$,
- a divisor $E = E_1 + \cdots + E_N \in \mathbb{D}(G)$ with $N \geqslant \max\{\deg G + (\ell+3)(2g-1) + (s+1)n + 2 + \mu,$ $(\ell+1)\deg G + 4g + (s+1)n\}$,
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\ldots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\ldots,N}^{i=0,\ldots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{g} = (g_{v,j}^{(u)})_{v=1,2,\ j=1,\ldots,N}^{u=0,\ldots,\ell}$, where $g_{v,j}^{(u)} = g_v^{(u)}(E_j) \in \mathbb{F}$, $\langle g_1^{(u)}, g_2^{(u)} \rangle_{\text{Я}} = \text{Я}(G_u)$ and $\delta_{G_u}(g_v^{(u)}) \leqslant 4g - 1 + (u+1)\deg(G) + (s+1)n$, as in Corollary 5.15.

Output:

- $(y_i B_v^{(u)})_{i=0,\ldots,\mu-1,\ v=1,2}^{u=0,\ldots,\ell}$, where the $B_v^{(u)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ are as in Corollary 5.15, i.e. $\langle y_i B_v^{(u)} \rangle_{\mathbb{F}[x]} = \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$.

1: $(B_v^{(u)})_{v=1,2}^{u=0,\ldots,\ell} \leftarrow \mathsf{Generators}_{\text{Я}}(\boldsymbol{r}, D, G, E, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{g})$ ▷ Algorithm 15
2: **for** $u = 0, \ldots, \ell$, $v = 1, 2$ and $t = 0, \ldots, u$ **do**
3: $\quad b_{v,t}^{(u)} \in \text{Я}(-tG) \leftarrow$ the z^t-coefficient of $B_v^{(u)}$
4: $\quad (y_i b_{v,t}^{(u)})_{i=0,\ldots,\mu-1} \leftarrow \mathsf{BasisProducts}(b_{v,t}^{(u)}, E, -tG, \boldsymbol{x}, \boldsymbol{y})$ ▷ Algorithm 16
5: **for** $u = 0, \ldots, \ell$, $v = 1, 2$ and $i = 0, \ldots, \mu - 1$ **do**
6: $\quad B_{v,i}^{(u)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G) \leftarrow \sum_{t=0}^{u} z^t y_i b_{v,t}^{(u)}$
7: **return** $(B_{v,i}^{(u)})_{v=1,2,\ i=0,\ldots,\mu-1}^{u=0,\ldots,\ell}$

Proposition 5.38. *Algorithm 17 is correct and costs* $\widetilde{\mathcal{O}}(\mu^{\omega-1}\ell^3(n+g))$ *operations in* $\mathbb{F}$.

Proof. Correctness follows immediately from Corollary 5.17 and Lemma 5.37 once we show that the calls $\mathsf{BasisProducts}(b_{v,t}^{(u)}, E, -tG, \boldsymbol{x}, \boldsymbol{y})$ in Line 4 are valid. In particular, we need to verify that

$$N \geqslant \deg(-tG) + \delta_{-tG}(b_{v,t}^{(u)}) + 2g + \mu \tag{5.4}$$

for all appropriate values of u, v and t. Using the notation from Corollary 5.15 and Algorithm 15, we know that $b_{v,t}^{(u)} = \binom{u}{t}(-R)^{u-t} g_v^{(u)}$, hence by (5.3)

$$\delta_{-tG}(b_{r,v}^{(u)}) \leqslant (t+1)\deg G + (u-t+2)(2g-1) + (s+1)n + 1 \ . \tag{5.5}$$

The sought bound (5.4) on N then follows from

$$-t\deg G + \delta_{-tG}(b_{v,t}^{(u)}) \leqslant \deg G + (\ell+2)(2g-1) + (s+1)n + 1 \ .$$

For the complexity, we note that Line 1 costs $\widetilde{\mathcal{O}}(\mu^{\omega-1}\ell^3(n+g))$ operations by Proposition 5.32, while each call $\mathsf{BasisProducts}(b_{v,t}^{(u)}, E, -tG, \boldsymbol{x}, \boldsymbol{y})$ in Line 4 costs $\widetilde{\mathcal{O}}(\mu^{\omega-1}(N + |\deg(-tG)|)) \subseteq \widetilde{\mathcal{O}}(\mu^{\omega-1}\ell(n+g))$ operations by Lemma 5.37. Since the for-loop in Line 2 has $\mathcal{O}(\ell^2)$ iterations, the stated complexity follows – the rest of the algorithm is memory management and is therefore "free". □

Remark 5.39. In the context of Remark 5.18, computing the generating set $\{y_i \widetilde{B}_v^{(u)}\}$ over $\mathbb{F}[x]$ of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ can be done using $\widetilde{\mathcal{O}}(\mu^{\omega-1}s\ell^2(n+g))$ operations in $\mathbb{F}$, since in that case only $\mathcal{O}(s\ell)$ coefficients of the $\widetilde{B}_v^{(u)}$ are nonzero.

5.5.5 Solving the interpolation step of Guruswami-Sudan

In the previous section we saw how to efficiently compute a generating set of $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ over $\mathbb{F}[x]$. As we are about to see in the current section, this allows us to find a nonzero $Q \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ whose $\delta_G(Q)$ is minimal. The interpolation step of Guruswami-Sudan list-decoding is thereby be complete if we discover that $\delta_G(Q) < s(n-\tau)$. If on the other hand $\delta_G(Q) \geqslant s(n-\tau)$, then we can be sure that all codewords necessarily have Hamming distance strictly greater than τ from the received word, in which case we simply declare decoding failure. In the following lemma, we construct a matrix $\boldsymbol{M}_{s,\ell} \in \mathbb{F}[x]$ whose shifted Popov from contains a row from which the sought Q can be recovered.

Lemma 5.40. *For any divisor A and any function $a = \sum_{i=0}^{\mu-1} a_i y_i^{(A)} \in \text{Я}(A)$, where $a_i \in \mathbb{F}[x]$, let $\curlyvee^{(A)}(a) = (a_0, \ldots, a_{\mu-1}) \in \mathbb{F}[x]^\mu$. Furthermore, for any polynomial $Q = \sum_{t=0}^{\ell} z^t Q^{(t)} \in \bigoplus_{t=0}^{\ell} z^t \text{Я}(-tG)$ let*

$$\curlyvee_G(Q) = \left(\curlyvee^{(0)}(Q_0) \mid \curlyvee^{(-G)}(Q_1) \mid \cdots \mid \curlyvee^{(-\ell G)}(Q_\ell) \right) \in \mathbb{F}[x]^{\mu(\ell+1)} \ ,$$

and consider the matrix

$$\boldsymbol{M}_{s,\ell} = \left[\begin{array}{c} \boldsymbol{M}_{s,\ell}^{(1)} \\ \hline \boldsymbol{M}_{s,\ell}^{(2)} \end{array}\right] \in \mathbb{F}[x]^{2\mu(\ell+1)\times\mu(\ell+1)} ,$$

where for $v = 1, 2$ we define

$$\boldsymbol{M}_{s,\ell}^{(v)} = \left(\left[\begin{array}{c} \curlyvee_G(y_0 B_v^{(0)}) \\ \hline \vdots \\ \hline \curlyvee_G(y_{\mu-1} B_v^{(0)}) \end{array}\right]^\top \middle| \cdots \middle| \left[\begin{array}{c} \curlyvee_G(y_0 B_v^{(\ell)}) \\ \hline \vdots \\ \hline \curlyvee_G(y_{\mu-1} B_v^{(\ell)}) \end{array}\right]^\top \right)^\top ,$$

and where the $B_v^{(u)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ are as in Corollary 5.15. It then holds that $\curlyvee_G$ is an $\mathbb{F}[x]$-isomorphism between $\mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ and the row space of $\boldsymbol{M}_{s,\ell}$, and that for any Q as before, $\delta_G(Q) = \mu \deg_{\boldsymbol{d}} \curlyvee_G(Q)$, where

$$\begin{aligned} \boldsymbol{d} &= (\boldsymbol{d}^{(0)} | \cdots | \boldsymbol{d}^{(\ell)}) \in (\tfrac{1}{\mu}\mathbb{Z})^{\mu(\ell+1)} \quad \textit{with} \\ \boldsymbol{d}^{(t)} &= \frac{1}{\mu}\big(\delta_{-tG}(y_0^{(-tG)}), \ldots, \delta_{-tG}(y_{\mu-1}^{(-tG)})\big) \in (\tfrac{1}{\mu}\mathbb{Z})^{\mu} \quad \textit{for } t = 0, \ldots, \ell , \end{aligned}$$

recalling from Section 5.4 that $\delta_G(Q) := \max_t \delta_{-tG}(Q^{(t)})$.

Proof. Corollary 5.17 immediately implies that $\curlyvee_G$ is the claimed $\mathbb{F}[x]$-isomorphism. Writing $Q^{(t)} = \sum_{i=0}^{\mu-1} Q_i^{(t)} y_i^{(-tG)}$ for $t = 0, \ldots, \ell$, where $Q_i^{(t)} \in \mathbb{F}[x]$, gives

$$\begin{aligned} \delta_G(Q) &= \max_t\{\delta_{-tG}(Q^{(t)})\} = \max_{t,i}\{\delta_{-tG}(Q_i^{(t)} y_i^{(-tG)})\} \\ &= \max_{t,i}\{\delta(Q_i^{(t)}) + \delta_{-tG}(y_i^{(-tG)})\} = \max_{t,i}\{\mu \deg Q_i^{(t)} + \delta_{-tG}(y_i^{(-tG)})\} \\ &= \mu \deg_{\boldsymbol{d}} \curlyvee_G(Q) . \end{aligned}$$

□

Remark 5.41. In the context of Lemma 5.40, notice the neat similarity between the fact that $\delta_G(Q) = \mu \deg_{\boldsymbol{d}} \curlyvee_G(Q)$ and that $\delta(h) = \mu \deg h$ for any $h \in \mathbb{F}[x]$.

In the context of Lemma 5.40, it should be clear that if the $\boldsymbol{d}$-Popov form of $\boldsymbol{M}_{s,\ell}$ contains a row $\boldsymbol{p}$ satisfying $\mu \deg_{\boldsymbol{d}} \boldsymbol{p} < s(n - \tau)$, then we can obtain the sought Q as $\curlyvee_G^{-1}(\boldsymbol{p})$. Conversely, if no such $\boldsymbol{p}$ exists, then the same goes for Q. Recall from Proposition 1.8 on page 8 that computing a shifted Popov form of any full rank matrix $\boldsymbol{M} \in \mathbb{F}[x]^{m\times m}$ can be done with cost $\tilde{\mathcal{O}}(m^\omega \deg(\boldsymbol{M}))$. Of course, we can not use these results directly, as our matrix $\boldsymbol{M}_{s,\ell} \in \mathbb{F}[x]^{2\mu(\ell+1)\times\mu(\ell+1)}$ is not square. However, we can easily reduce our setting to the square case

using the results from [ZL13]: for any matrix $\boldsymbol{N} \in \mathbb{F}[x]^{r \times m}$, where $r \geqslant m$, we can compute a row basis matrix $\boldsymbol{B}$ in (unshifted) Popov form in complexity $\tilde{\mathcal{O}}(rm^{\omega-1} \deg(\boldsymbol{N}))$.

For us this means that we can compute a matrix $\boldsymbol{B}_{s,\ell} \in \mathbb{F}[x]^{\mu(\ell+1)\times\mu(\ell+1)}$ in (unshifted) Popov form having the same row space as $\boldsymbol{M}_{s,\ell}$ using no more than $\tilde{\mathcal{O}}(\mu^{\omega}\ell^{\omega} \deg \boldsymbol{M}_{s,\ell})$ operations in $\mathbb{F}$. Furthermore, we can bound

$$\begin{aligned}\deg \boldsymbol{M}_{s,\ell} &\leqslant \frac{1}{\mu} \max_{u,v,i,t} \{-t \deg G + \delta(y_i) + \delta_{-tG}(b_{v,t}^{(u)})\} \\ &\leqslant \max_{u,t} \frac{6g - 2 + \mu + (u-t)(n+2g-1) + (s+1)n + \deg G}{\mu} \\ &\in \mathcal{O}(\mu^{-1}\ell(n+g)) \ , \end{aligned} \tag{5.6}$$

where the first inequality is due to Lemma 5.8, while the second follows from combining Lemma 5.6 with (5.5). Consequently, $\boldsymbol{B}_{s,\ell}$ can be computed within our target complexity $\tilde{\mathcal{O}}(\mu^{\omega-1}\ell^{\omega+1}(n+g))$, and since $\boldsymbol{B}_{s,\ell}$ satisfies the same degree bound as $\boldsymbol{M}_{s,\ell}$ in (5.6), its $\boldsymbol{d}$-Popov form can be obtained with the same cost.

Remark 5.42. Once again, we point out that the complexity can be improved using the alternative generating set from Remark 5.18. In (5.6), the expression $u-t$ corresponds to the exponent of R in the expression $\binom{u}{t}(-R)^{u-t}g_v^{(u)}$, which is the z^t-coefficient of $B_v^{(u)}$. Since the exponent of R in any coefficient of $\widetilde{B}_v^{(u)}$ never exceeds s, we obtain the improved complexity $\tilde{\mathcal{O}}(\mu^{\omega-1}s\ell^{\omega}(n+g))$.

At this point, we have addressed all of the parts in the interpolation step of Guruswami-Sudan list-decoding, as outlined in Section 5.4.2. Before we combine them into a single algorithm in Section 5.6, let us consider the root-finding step in the following section.

5.5.6 Root-finding

In this section, we consider the final computational ingredient that we will need for Guruswami-Sudan list-decoding: given a polynomial $Q(z) \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$, compute the set[1] $L = \{f \in \mathcal{L}(G) \mid Q(f) = 0\}$ of all roots of Q. As foreshadowed in Section 5.4.2, we accomplish this by changing the representation of Q from

[1]Some readers might object to our choice of using a set instead of a list – we are doing *list* decoding after all. Although it hardly matters in the context of this thesis, a case can be made that set is the more appropriate data structure here, as we do not care about the order of its members nor about repetitions.

$\bigoplus_{t=0}^{\ell} z^t \mathcal{H}(-tG)$ to $\bigoplus_{t=0}^{\ell} z^t \mathbb{F}[\![x]\!]$, which will allow us to use the root-finding algorithm from [NRS17]. It is not hard to see that the latter representation can be obtained from the former by simply considering the P_0-adic power series expansions in x of the z-coefficients of Q – recall from Section 5.2 that a rational place P_0 having x as a local parameter can be assumed to exist as long as we are willing to pay the price of additional logarithmic factors in the total complexity. Let us now briefly summarize the results from [NRS17], adapting them to our setting.

Considering the aforementioned rational place P_0 as fixed, for any function $h \in F$, we denote by $\widehat{h} \in x^{v_{P_0}(h)}\mathbb{F}[\![x]\!]$ the P_0-adic power series expansion of h in x, letting $\widehat{0} = 0$. Furthermore, for any polynomial $Q = \sum_t z^t Q^{(t)} \in F[z]$, we write $\widehat{Q} = \sum_t z^t \widehat{Q}^{(t)}$. The following definition is from [NRS17], and it describes the output of their root-finding algorithm:

Definition 5.43. If $\widehat{Q} \in \mathbb{F}[\![x]\!][z]$ and $\beta \in \mathbb{Z}_{\geqslant 0}$, then a *basic root set* of $\widehat{Q}$ to precision β is a set $\{(\widehat{f}_r, \alpha_r)\}_{r=1}^{m} \subset \mathbb{F}[x] \times \mathbb{Z}_{\geqslant 0}$ with $m \leqslant \deg \widehat{Q}$ such that

1) $\widehat{Q}(\widehat{f}_r + x^{\alpha_r} z) \equiv 0 \pmod{x^\beta}$ for $r = 1, \ldots, m$, and
2) $\widehat{Q}(\widehat{f}) \equiv 0 \pmod{x^\beta} \iff \widehat{f} \in \bigcup_{r=1}^{m} (\widehat{f}_r + x^{\alpha_r}\mathbb{F}[\![x]\!])$ for every $\widehat{f} \in \mathbb{F}[\![x]\!]$.

Example 5.44. An illuminating example taken straight from [NRS17] is when $\widehat{Q} = z^2 + z \in \mathbb{F}_2[\![x]\!][z]$ and $\beta = 1$. Here, it is crucial to understand that $\{(0,0)\}$ is *not* a basic root set of $\widehat{Q}$ to precision 1 even though every $\widehat{f} \in \mathbb{F}_2[\![x]\!]$ satisfies $\widehat{Q}(\widehat{f}) \equiv 0 \pmod{x}$; the reason for this is that the first restriction in Definition 5.43 is not satisfied. Instead, a basic root set is given by $\{(0,1),(1,1)\}$.

Our algorithm for computing the sought $\mathcal{L}(G)$-roots of $Q \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D,G)$ will fundamentally rely on the following result:

Theorem 5.45 ([NRS17, Theorem 1.2])**.** *There is an algorithm which for any $\widehat{Q} \in \mathbb{F}[\![x]\!][z]$ and any precision $\beta \in \mathbb{Z}_{\geqslant 0}$ computes a basic root set of $\widehat{Q}$ to precision β using $\widetilde{\mathcal{O}}(\ell\beta)$ deterministic operations in $\mathbb{F}$, together with an extra $\widetilde{\mathcal{O}}(\mathsf{R}_{\mathbb{F}}(\ell)\beta)$ operations, where $\mathsf{R}_{\mathbb{F}}(\ell)$ is the cost of finding all $\mathbb{F}$-roots of a degree ℓ polynomial in $\mathbb{F}[z]$. Here, we can choose to use a Las Vegas algorithm with $\mathsf{R}_{\mathbb{F}}(\ell) \in \widetilde{\mathcal{O}}(\ell)$, e.g. [vzGG12, Corollary 14.16], or a deterministic one from [Sho91] with $\mathsf{R}_{\mathbb{F}}(\ell) \in \widetilde{\mathcal{O}}(\ell\kappa^2\sqrt{p})$, where $|\mathbb{F}| = p^\kappa$ for some prime p.*

In order to use Theorem 5.45 in our setting, we will need to figure out:

1) how to choose the precision β,

2) how to convert $Q \in \bigoplus_{t=0}^{\ell} z^t \text{Я}(-tG)$ to $\widehat{Q} \in \bigoplus_{t=0}^{\ell} z^t \mathbb{F}[\![x]\!]$ and

3) how to obtain the sought roots $f \in \mathcal{L}(G)$ of Q from a basic root set of $\widehat{Q}$.

The second item in the above list is the simplest – writing $Q = \sum_{t=0}^{\ell} z^t Q^{(t)}$ with $Q^{(t)} = \sum_{i=0}^{\mu-1} Q_i^{(t)} y_i^{(-tG)}$, where $Q_i^{(t)} \in \mathbb{F}[x]$, we can compute $\widehat{Q} = \sum_{t=0}^{\ell} z^t \widehat{Q}^{(t)}$ by simply relying on the identity $\widehat{Q}^{(t)} = \sum_{i=0}^{\mu-1} Q_i^{(t)} \widehat{y}_i^{(-tG)}$. Assuming that we have precomputed the $\widehat{y}_i^{(-tG)} \in \mathbb{F}[\![x]\!]$ to sufficiently high precision, this is just basic arithmetic in $\mathbb{F}[x]$.

When it comes to the choice of the precision β, then there are two restrictions that ought to be considered. The first one comes from making sure that we don't return "spurious" roots, i.e. those $f \in \mathcal{L}(G)$ such that $\widehat{Q}(\widehat{f}) \equiv 0 \pmod{x^\beta}$ while $Q(f) \neq 0$. As we are about to see in the following lemma, this issue is easily avoided by choosing $\beta > \delta_G(Q)$.

Lemma 5.46. *Let $Q(z) = \sum_{t=0}^{\ell} z^t Q^{(t)}$ with $Q^{(t)} \in \text{Я}(-tG)$, and let $f \in \mathcal{L}(G)$. If $\beta > \delta_G(Q)$ and $\widehat{Q}(\widehat{f}) \equiv 0 \pmod{x^\beta}$, then $Q(f) = 0$.*

Proof. Notice that since $f^t Q^{(t)} \in \text{Я}$ for all t, then $Q(f) \in \text{Я}$. Furthermore, since

$$\delta(f^t Q^{(t)}) = \delta_{tG}(f^t) + \delta_{-tG}(Q^{(t)}) \leqslant \delta_{-tG}(Q^{(t)}) \leqslant \delta_G(Q) \ ,$$

where the first inequality is due to $f \in \mathcal{L}(G)$, then $\delta(Q(f)) \leqslant \delta_G(Q)$. Combining this with the assumption that $\widehat{Q}(\widehat{f}) = \widehat{Q(f)} \equiv 0 \pmod{x^\beta}$, we may conclude that $Q(f) \in \mathcal{L}(\delta_G(Q)P_\infty - \beta P_0)$, and if $\beta > \delta_G(Q)$, then this Riemann-Roch space is trivial. □

Remark 5.47. Actually, in our context of list decoding, we could dispense with Lemma 5.46 altogether: Theorem 5.11 merely states that every codeword within radius τ from the received word $\boldsymbol{r}$ corresponds to an $\mathcal{L}(G)$-root of Q. The converse, however, is not guaranteed to be true – there could hypothetically exist roots of Q whose associated codewords are far away from $\boldsymbol{r}$. To detect and discard such roots, we simply encode them and check their Hamming distance to $\boldsymbol{r}$; and in the process of doing so we would automatically get rid of any aforementioned spurious roots of Q. Moreover, these additional checks would have no impact on our asymptotic complexity bound. In spite of this, we keep Lemma 5.46 for the sake of having a root-finding algorithm that is provably correct even in isolation from list decoding.

The second restriction on the precision β is posed by the task of converting the truncated power series roots of $\widehat{Q}$ back to $\mathcal{L}(G)$. Indeed, a basic root set

$\{(\widehat{f}_r, \alpha_r)\}_{r=1}^m$ describes each root $\widehat{f}_r \in \mathbb{F}[x]$ of $\widehat{Q}$ only to precision α_r, and if this α_r is too small, then there could exist two[2] distinct functions $h_1, h_2 \in \mathcal{L}(G)$ satisfying $\widehat{h}_1 \equiv \widehat{h}_2 \equiv \widehat{f}_r \pmod{x^{\alpha_r}}$. In Lemma 5.49, we will see how we can indirectly control α_r by increasing β; but first, let us show that conversion from truncated power series to $\mathcal{L}(G)$ is guaranteed to be unambiguous as long as $\alpha_r > \deg G$.

Lemma 5.48. *If $\widehat{h} \in \mathbb{F}[x]$ and $\alpha > \deg G$, then $|\mathcal{L}(G) \cap (\widehat{h} + x^\alpha \mathbb{F}[\![x]\!])| \leqslant 1$.*

Proof. If $h_1, h_2 \in \mathcal{L}(G) \cap (\widehat{h} + x^\alpha \mathbb{F}[\![x]\!])$, then $\widehat{h}_1 \equiv \widehat{h}_2 \equiv \widehat{h} \pmod{x^\alpha}$. But then it follows that $h_1 - h_2 \in \mathcal{L}(G - \alpha P_0) = \{0\}$. □

As promised, we now proceed by showing that the α_r from Definition 5.43 can be made arbitrarily large by choosing the precision β appropriately.

Lemma 5.49. *If $Q(z) = \sum_{t=0}^{\ell} z^t Q^{(t)} \neq 0$ with $Q^{(t)} \in \mathcal{R}(-tG)$, and if $f \in \mathcal{R}(G)$ satisfies $\widehat{Q}(\widehat{f} + x^\alpha z) \equiv 0 \pmod{x^\beta}$ for some $\alpha \in \mathbb{Z}$, then*

$$\alpha \geqslant \frac{1}{\ell}(\beta - \delta_G(Q)) - \delta_G(f) \ .$$

Proof. We begin by defining

$$T = Q(z+f) = \sum_{t=0}^{\ell} (z+f)^t Q^{(t)} = \sum_{t=0}^{\ell} \sum_{u=0}^{t} \binom{t}{u} z^u f^{t-u} Q^{(t)} = \sum_{u=0}^{\ell} z^u T_u \ ,$$

where $T_u = \sum_{t=u}^{\ell} \binom{t}{u} f^{t-u} Q^{(t)}$. Since $f^{t-u} \in \mathcal{R}((t-u)G)$ and $Q^{(t)} \in \mathcal{R}(-tG)$, then $T_u \in \mathcal{R}(-uG)$. Furthermore, $x^{\alpha u}\widehat{T}_u \equiv 0 \pmod{x^\beta}$ for all u because

$$\widehat{Q}(\widehat{f} + x^\alpha z) = \widehat{T}(x^\alpha z) = \sum_{u=0}^{\ell} z^u x^{\alpha u} \widehat{T}_u \equiv 0 \pmod{x^\beta} \ .$$

Letting $r \in \{0, \ldots, \ell\}$ be such that $v_{P_0}(T_r) < \infty$ is maximal, observe that

$$\alpha\ell + v_{P_0}(T_r) \geqslant \alpha r + v_{P_0}(T_r) = v_{P_0}(x^{\alpha r} T_r) \geqslant \beta \ ,$$

which implies that $\alpha \geqslant \frac{1}{\ell}(\beta - v_{P_0}(T_r))$. Finally, noting that

$$0 \neq T_r \in \mathcal{L}(\delta_{-rG}(T_r) P_\infty - rG - v_{P_0}(T_r) P_0) \ ,$$

[2] Actually, if there exist distinct functions $h_1, h_2 \in \mathcal{L}(G)$ with $\widehat{h}_1 \equiv \widehat{h}_2 \pmod{x^{\alpha_r}}$, then $h_1 - h_2 \in \mathcal{L}(G - \alpha_r P_0) \neq \{0\}$. Consequently, there are exactly $q^{\ell(G - \alpha_r P_0)}$ possibilities.

then the sought conclusion follows from

$$\begin{aligned} v_{P_0}(T_r) &\leqslant \delta_{-rG}(T_r) - r\deg G \leqslant \delta_{-rG}(T_r) \\ &= \delta_{-rG}\big(\sum_{t=r}^{\ell}\binom{t}{r} f^{t-r}Q^{(t)}\big) \\ &\leqslant \max_t\{\delta_{-rG}(f^{t-r}Q^{(t)})\} \\ &\leqslant \max_t\{(t-r)\delta_G(f) + \delta_{-tG}(Q^{(t)})\} \\ &= \ell\delta_G(f) + \delta_G(Q)\ . \end{aligned}$$

□

Combining Lemma 5.49 and Lemma 5.48, we obtain the final restriction

$$\beta \geqslant 2\ell\deg G + s(n-\tau)\ ,$$

which ensures that unambiguous conversion from the truncated power series roots of $\widehat{Q}$ to $\mathcal{L}(G)$ is always possible. Indeed, this bound follows immediately from the fact that $\delta_G(f) \leqslant \deg G$ for all $f \in \mathcal{L}(G)$ and the assumption that $\delta_G(Q) < s(n-\tau)$. Knowing that such conversion is possible, however, is not enough – we also need to know how to actually carry it out. In the following simple lemma, we show how this can be done using simultaneous Hermite-Padé approximations, which we have already familiarized ourselves with in Section 5.5.2.

Lemma 5.50. *If $f \in \mathcal{L}(G)$ and $\sum_{i=0}^{\mu-1} f_i\widehat{y}_i^{(G)} \equiv \widehat{f} \pmod{x^\alpha}$ for some $f_i \in \mathbb{F}[x]$ with $\deg f_i \leqslant -\frac{1}{\mu}\delta_G(y_i^{(G)})$ and $\alpha > \deg G$, then $\sum_{i=0}^{\mu-1} f_i y_i^{(G)} = f$.*

Proof. Since $\delta(f_i) = \mu\deg f_i$, then $\delta_G(f_i y_i^{(G)}) \leqslant \mu\deg f_i + \delta_G(y_i^{(G)}) \leqslant 0$. But then $\sum_{i=0}^{\mu-1} f_i y_i^{(G)} \in \mathcal{L}(G)$, and the conclusion follows from Lemma 5.48. □

Using the notation from Definition 5.24 in the context of Lemma 5.50, we see that $(f_0,\ldots,f_{\mu-1},1) \in \mathcal{H}_{x^\alpha}(\boldsymbol{A})$, where $\boldsymbol{A} = [\widehat{y}_0^{(G)},\cdots,\widehat{y}_{\mu-1}^{(G)},-\widehat{f}] \in \mathbb{F}[x]^{1\times(\mu+1)}$. Recovering $f \in \mathcal{L}(G)$ from $\widehat{f}\operatorname{rem} x^\alpha$ thus translates to finding a polynomial vector $\boldsymbol{f} \in \mathcal{H}_{x^\alpha}(\boldsymbol{A})$ whose rightmost entry is 1 and $\deg_{\boldsymbol{d}}\boldsymbol{f} = 0$, where

$$\boldsymbol{d} = \frac{1}{\mu}(\delta_G(y_0^{(G)}),\ldots,\delta_G(y_{\mu-1}^{(G)}),0) \in (\tfrac{1}{\mu}\mathbb{Z})^{\mu+1}\ .$$

But this is easily accomplished by relying on Theorem 5.25 and Corollary 5.27. We conclude this section by presenting our root-finding approach in its entirety in Algorithm 18.

Algorithm 18 FindRoots$(D, G, Q, \widehat{\boldsymbol{y}})$

Input:

- Divisors D and G for the code $\mathcal{C}_{\mathcal{L}}(D, G)$,
- a nonzero $Q = \sum_{t=0}^{\ell} z^t Q^{(t)} \in \mathcal{M}_{s,\ell}^{(\boldsymbol{r})}(D, G)$ with $\delta_G(Q) < s(n-\tau)$, where $Q^{(t)} = \sum_{i=0}^{\mu-1} Q_i^{(t)} y_i^{(-tG)}$ for some $Q_i^{(t)} \in \mathbb{F}[x]$,
- $\widehat{\boldsymbol{y}} = (\widehat{y}_i^{(-tG)})_{i=0,\dots,\mu-1}^{t=0,\dots,\ell}$ with $\widehat{y}_i^{(-tG)} \in \mathbb{F}[x]$ such that $v_{P_0}(y_i^{(-tG)} - \widehat{y}_i^{(-tG)}) \geqslant \beta := 2\ell \deg G + s(n-\tau)$.

Output:

- $L = \{f \in \mathcal{L}(G) \mid Q(f) = 0\}$ with $|L| \leqslant \ell$.

1: $\widehat{Q}^{(t)} \in \mathbb{F}[x] \leftarrow \sum_{i=0}^{\mu-1} Q_i^{(t)} \widehat{y}_i^{(-tG)}$ for $t = 0, \dots, \ell$
2: $\widehat{Q} \in \mathbb{F}[x] \leftarrow \sum_{t=0}^{\ell} z^t \widehat{Q}^{(t)}$
3: $\widehat{L} \subset \mathbb{F}[x] \leftarrow$ all polynomials from a basic root set of $\widehat{Q}$ to precision β
4: $L \leftarrow \varnothing$
5: $\boldsymbol{d} \in (\frac{1}{\mu}\mathbb{Z})^{\mu+1} \leftarrow \frac{1}{\mu}(\delta_G(y_0^{(G)}), \dots, \delta_G(y_{\mu-1}^{(G)}), 0)$
6: $\alpha \in \mathbb{Z}_{>0} \leftarrow \deg G + 1$
7: **for** $\widehat{f} \in \widehat{L}$ **do**
8: $\quad \boldsymbol{F} \in \mathbb{F}^{(\mu+1)\times(\mu+1)} \leftarrow \boldsymbol{d}$-Popov basis of $\mathcal{H}_{x^\alpha}([\widehat{y}_0^{(G)}, \dots, \widehat{y}_{\mu-1}^{(G)}, -\widehat{f}])$
9: $\quad$ **if** $\boldsymbol{F}$ contains a row $\boldsymbol{f} = (f_0, \dots, f_{\mu-1}, 1)$ with $\deg_d \boldsymbol{f} = 0$ **then**
10: $\quad\quad L \leftarrow L \cup \{\sum_{i=0}^{\mu-1} f_i y_i^{(G)}\}$
11: **return** L

Proposition 5.51. *Algorithm 18 is correct and costs $\widetilde{\mathcal{O}}(\ell^2 \mu^{\omega-1}(n+g))$ operations in $\mathbb{F}$.*

Proof. For correctness, our goal is to prove that $L = K$, where L is the output of the algorithm and $K = \{f \in \mathcal{L}(G) \mid Q(f) = 0\}$. If $\{(\widehat{f}_r, \alpha_r)\}_{r=1}^m \subset \mathbb{F}[x] \times \mathbb{Z}_{\geqslant 0}$ denotes the basic root set used in Line 3, i.e. $\widehat{L} = \{\widehat{f}_r\}_{r=1}^m$, then it is clear that $K \subseteq \bigcup_{r=1}^m K_r$, where $K_r = \mathcal{L}(G) \cap (\widehat{f}_r + x^{\alpha_r}\mathbb{F}[\![x]\!])$ and $m \leqslant \ell$. Since $\delta_G(Q) < s(n-\tau)$, $\delta_G(\widehat{h}_r) \leqslant \deg G$ and $\beta = 2\ell \deg G + s(n-\tau)$, then Lemma 5.49 guarantees that $\alpha_r \geqslant \frac{1}{\ell}\big(\beta - \delta_G(Q)\big) - \delta_G(\widehat{h}) \geqslant \deg G + 1$, hence $|K_r| \leqslant 1$ by Lemma 5.48. Combining this with the fact that each non-empty K_r necessarily contains an $\mathcal{L}(G)$-root of Q, as implied by Lemma 5.46 because $\beta > \delta_G(Q)$, we may conclude that $K = \bigcup_{r=1}^m K_r$. But due to Lemma 5.50

$$\bigcup_{r=1}^{m} K_r = L = \Big\{ \sum_{i=0}^{\mu-1} f_i^{(r)} y_i^{(G)} \mid \sum_{i=0}^{\mu-1} f_i^{(r)} \widehat{y}_i^{(G)} \equiv \widehat{f}_r \pmod{x^\alpha}, \\ \deg f_i^{(r)} \leqslant -\frac{1}{\mu}\delta_G(y_i^{(G)}),\ r = 1, \dots, m \Big\} .$$

For the complexity, computing the $(\ell+1)\mu$ products $Q_i^{(t)}\widehat{y}_i^{(-tG)}$ in Line 1 costs $\widetilde{\mathcal{O}}(\ell\mu\beta) \subseteq \widetilde{\mathcal{O}}(\mu\ell^2(n+g))$. The basic root set of $\widehat{Q}$ in Line 3 can be computed with cost $\widetilde{\mathcal{O}}(\beta \deg_z(\widehat{Q})) \subseteq \widetilde{\mathcal{O}}(\ell^2(n+g))$ due to [NRS17] (see Theorem 5.45). Finally, the total cost of computing the $\boldsymbol{d}$-Popov bases in line 8 across all of the $\mathcal{O}(\ell)$ iterations in the surrounding for-loop is $\widetilde{\mathcal{O}}(\ell\mu^{\omega-1}(n+g))$ by Corollary 5.27. The claimed complexity of the algorithm follows. □

5.6 A complete decoding algorithm

We conclude this chapter by combining all of the intermediate algorithms from Section 5.5 into an efficient realization of Guruswami-Sudan list-decoding for the code $\mathcal{C}_{\mathcal{L}}(D, G)$. We begin by listing all of the data that we have assumed to be precomputed:

- a divisor $E = E_1 + \cdots + E_N \in \mathbb{D}(G)$ satisfying $N \geqslant \max\{\deg G + (\ell+3)(2g-1) + (s+1)n + 2 + \mu, (\ell+1)\deg G + 4g + (s+1)n\}$
- evaluations $\boldsymbol{x} = (x_j)_{j=1,\dots,N}$, where $x_j = x(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{y} = (y_{i,j})_{j=1,\dots,N}^{i=0,\dots,\mu-1}$, where $y_{i,j} = y_i^{(A)}(E_j) \in \mathbb{F}$,
- evaluations $\boldsymbol{g} = (g_{v,j}^{(u)})_{v=1,2,\ j=1,\dots,N}^{u=0,\dots,\ell}$, where $g_{v,j}^{(u)} = g_v^{(u)}(E_j) \in \mathbb{F}$ with $\langle g_1^{(u)}, g_2^{(u)} \rangle_{Я} = Я(G_u)$ and $\delta_{G_u}(g_v^{(u)}) \leqslant 4g - 1 + (u+1)\deg(G) + (s+1)n$, as in Corollary 5.15,
- $\widehat{\boldsymbol{y}} = (\widehat{y}_i^{(-tG)})_{i=0,\dots,\mu-1}^{t=0,\dots,\ell}$ with $\widehat{y}_i^{(-tG)} \in \mathbb{F}[x]$ such that $v_{P_0}(y_i^{(-tG)} - \widehat{y}_i^{(-tG)}) \geqslant \beta := 2\ell \deg G + s(n-\tau)$.

Without further ado, we present the main result of this chapter in Algorithm 19.

Algorithm 19 $\mathsf{Decode}(\boldsymbol{r}, s, \ell, D, G)$

Input:

- Received word $\boldsymbol{r} \in \mathbb{F}^n$,
- divisors D and G for the code $\mathcal{C}_{\mathcal{L}}(D, G)$,
- decoding parameters $s, \ell \in \mathbb{Z}_{>0}$ with $s \leqslant \ell$,
- corresponding list-decoding radius $\tau \in \mathbb{Z}_{>0}$,

Output:

- $L = \{f \in \mathcal{L}(G) \mid d(\boldsymbol{r}, \boldsymbol{c}) \leqslant \tau\}$ or FAIL

$(B_{v,i}^{(u)})_{v=1,2,\ i=0,\dots,\mu-1}^{u=0,\dots,\ell} \leftarrow \mathsf{Generators}_{\mathbb{F}[x]}(\boldsymbol{r}, D, G, E, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{g})$
$\boldsymbol{M}_{s,\ell} \in \mathbb{F}[x]^{2\mu(\ell+1)\times\mu(\ell+1)} \leftarrow$ matrix based on the $B_{v,i}^{(u)}$ as in Lemma 5.40
$\boldsymbol{B}_{s,\ell} \in \mathbb{F}[x]^{\mu(\ell+1)\times\mu(\ell+1)} \leftarrow$ basis matrix in (unshifted) Popov form of $\boldsymbol{M}_{s,\ell}$
$\boldsymbol{d} \in (\frac{1}{\mu}\mathbb{Z})^{\mu(\ell+1)} \leftarrow (\boldsymbol{d}^{(0)}|\cdots|\boldsymbol{d}^{(\ell)})$ with $\boldsymbol{d}^{(t)} = \frac{1}{\mu}(\delta_{-tG}(y_i^{(-tG)}))_{i=0}^{\mu-1} \in (\frac{1}{\mu}\mathbb{Z})^{\mu}$
$\boldsymbol{V}_{s,\ell} \in \mathbb{F}[x]^{\mu(\ell+1)\times\mu(\ell+1)} \leftarrow \boldsymbol{d}$-Popov form of $\boldsymbol{B}_{s,\ell}$
$\boldsymbol{Q} = \big((Q_i^{(0)})_{i=0}^{\mu}|\dots|(Q_i^{(\ell)})_{i=0}^{\mu}\big) \in \mathbb{F}[x]^{\mu(\ell+1)} \leftarrow \deg_{\boldsymbol{d}}$-minimal row of $\boldsymbol{V}_{s,\ell}$
if $\deg_{\boldsymbol{d}} \boldsymbol{Q} \geqslant s(n-\tau)$ **then**
    return FAIL
$Q \in \bigoplus_{t=0}^{\ell} z^t \text{Я}(-tG) \leftarrow \sum_{t=0}^{\ell} z^t \sum_{i=0}^{\mu-1} Q_i^{(t)} y_i^{(-tG)}$
$L \leftarrow \mathsf{FindRoots}(D, G, Q, \widehat{\boldsymbol{y}})$
for $f \in L$ **do**
    $\boldsymbol{c} \in \mathbb{F}^n \leftarrow \mathsf{Evaluate}(f, D, G, \boldsymbol{x}, \boldsymbol{y})$
    if $d(\boldsymbol{r}, \boldsymbol{c}) > \tau$ **then** $L \leftarrow L \backslash \{f\}$
return L

Remark 5.52. Note that it is trivial to modify Algorithm 19 to return the codewords $\boldsymbol{c} \in \mathcal{C}_{\mathcal{L}}(D, G)$ within radius τ from the received word – this is in contrast to returning the underlying messages $f \in \mathcal{L}(G)$ in the usual spirit of Guruswami-Sudan list-decoding.

Combining all of the results from Section 5.5, we arrive at the following:

Theorem 5.53. *Algorithm 19 is correct and costs* $\widetilde{\mathcal{O}}(s\ell^{\omega}\mu^{\omega-1}(n+g))$ *operations in* $\mathbb{F}$.

In the remainder of this chapter, we consider the performance of Algorithm 19 for special cases of codes, comparing with previously known results when possible.

5.6.1 Examples

Example 5.54. AG codes obtained from the rational function field $\mathbb{F}(x)$ are known as *generalized Reed-Solomon (GRS) codes*. In this case $g = 0$ and $\mu = 1$, which specializes the complexity of Algorithm 19 to $\widetilde{\mathcal{O}}(s\ell^{\omega}n)$ operations in $\mathbb{F}$. The same complexity is achieved for families of function fields having fixed small genus, e.g. those arising from elliptic curves. The best known complexity for Guruswami-Sudan list-decoding of GRS codes is $\widetilde{\mathcal{O}}(s^2\ell^{\omega-1}n)$ [CJN+15].

Example 5.55. By definition, any maximal function field F over $\mathbb{F} = \mathbb{F}_q$ attains the Hasse-Weil bound – it has exactly $N_1 = q + 1 + 2g\sqrt{q}$ rational places, where q is necessarily a square. If F is such a function field, then any place P of $F\overline{\mathbb{F}}$, where $\overline{\mathbb{F}}$ denotes the algebraic closure of $\mathbb{F}$, necessarily contains a positive element no larger than $\sqrt{q}$ in its Weierstrass semigroup [HKT08, Theorem 10.6], i.e. we are guaranteed that $\mu \leqslant \sqrt{q}$ in the complexity of Algorithm 19. Furthermore, it is well known that all maximal function fields satisfy $g \leqslant \sqrt{q}(\sqrt{q}-1)/2 \in \mathcal{O}(q)$, which implies that any code of length $n \in \Omega(q)$ over such a function field can be decoded using no more that $\widetilde{\mathcal{O}}(s\ell^{\omega}q^{(\omega-1)/2}n) \subseteq \widetilde{\mathcal{O}}(s\ell^{\omega}n^{(\omega+1)/2})$ operations in $\mathbb{F}$.

We obtain even better results for long codes over specific maximal function fields:

Example 5.56. In the case of Hermitian[3] function field $F = \mathbb{F}_{q^2}(x_1, x_2)$, where $x_2^q + x_2 = x_1^{q+1}$, we have $N_1 = q^3 + 1$ rational places and genus $g = q(q-1)/2$. The usual choice of P_∞ in one-point codes of F gives $\mu = q$. Consequently, we can decode any such code of length $n \in \Omega(q^3)$ using

$$\widetilde{\mathcal{O}}(s\ell^{\omega}q^{\omega-1}q^3) = \widetilde{\mathcal{O}}(s\ell^{\omega}q^{\omega+2}) = \widetilde{\mathcal{O}}(s\ell^{\omega}n^{(\omega+2)/3})$$

operations in $\mathbb{F}$. For $n = q^3$, our approach specializes to the one from [NB15].

Example 5.57. The Giulietti-Korchmaros function field $\mathbb{F}_{q^6}(x_1, x_2, x_3)$ from [GK09], where $x_2^q + x_2 = x_1^{q+1}$ and $x_3^{q^2-q+1} = x_1^{q^2} - x_1$, is also maximal – it has $\mu \leqslant q^3$, $g = (q^5 - 2q^3 + q^2)/2$ and $N_1 = q^8 - q^6 + q^5 + 1$. In this case, we can decode any code of lenth $n \in \Omega(q^8)$ with cost $\widetilde{\mathcal{O}}(s\ell^{\omega}n^{(3\omega+5)/8})$.

Example 5.58. The Suzuki[4] function field $F = \mathbb{F}_q(x_1, x_2)$, where $q = 2^{2e+1}$ is an odd power of two and $x_2^q + x_2 = x_1^{2^e}(x_1^q + x_1)$, has genus $g = 2^e(q-1)$ and $N_1 = q^2 + 1$ rational places. Although it is not maximal in the sense of the Hasse-Weil bound, no other function field with the same genus and constant field can surpass its number of rational places [Ser, Section 5.4]. From [BMZ21], it immediately follows that the Weierstrass semigroup of any place P contains

[3] already seen in Section 2.6.1 on page 40
[4] already seen in Section 4.6 on page 80

a positive element no greater than q, i.e. $\mu \leqslant q$. This means that for any code over F of length $n \in \Omega(q^2)$, the complexity of Algorithm 19 specializes to $\widetilde{\mathcal{O}}(s\ell^{\omega} n^{(\omega+1)/2})$.

Example 5.59. Let F be a function field over $\mathbb{F}$ corresponding to a $C_{a,b}$ curve; in this example, we will use the notation from Section 2.1. One-point codes of the form $\mathcal{C}_{\mathcal{L}}(D, mP_{\infty})$ over F were decoded in [BB10] with complexity $\widetilde{\mathcal{O}}(\ell^5 a^3(n+g))$ under the additional assumption that $D - nP_{\infty}$ is a principal divisor. It should be noted that here P_{∞} refers to a very specific rational place, which among other things implies that $a = \mu$. Using Algorithm 19, we can decode these codes at the cost of $\widetilde{\mathcal{O}}(s\ell^{\omega} a^{\omega-1}(n+g))$ operations in $\mathbb{F}$, which is notably better – and we need not assume that $G = mP_{\infty}$ and that $D - nP_{\infty}$ is principal.

In conclusion, our algorithm is faster than any other general list-decoding algorithm for AG codes of the form $\mathcal{C}_{\mathcal{L}}(D, G)$. Moreover, except for GRS codes, it is at least as fast as any specialized decoding algorithm.

Chapter 6

Conclusion

In this thesis we have considered various computational problems associated with algebraic geometry codes. In Chapter 2, we investigated encoding and unencoding of codes over $C_{a,b}$ curves and arrived at algorithms that have quasi-linear complexity for codes whose underlying points admit a semi-grid structure. In Chapter 3, we attempted to let go of the semi-grid assumption – this resulted in algorithms that require precomputation but are quasi-linear for generic point sets. In Chapter 4, we investigated the error-correcting capability of improved power decoding for general AG codes – we did this both theoretically and experimentally, and we concluded that it is comparable to that of the Guruswami-Sudan list decoder. Finally, in Chapter 5 we developed an efficient variant of the Guruswami-Sudan decoder in the fully general setting of all AG codes.

In the future, it would be interesting to develop the results from Chapter 3 further – removing the reliance on genericity. Furthermore, the algorithms could be generalized to higher dimensions, and the precomputation could be sped up. When it comes to Chapter 4, then it is likely that the proposed power decoder can be made efficient using simultaneous Hermite-Padé approximations. Finally, the complexity of our list decoder in Chapter 5 can likely be improved by a factor s/ℓ if we use a slightly different generating set in the interpolation step.

Appendix A

Notations

Complexity:	
$\mathcal{O}(\cdot), \widetilde{\mathcal{O}}(\cdot), \Omega(\cdot)$	Asymptotic complexity, Section 1.2.
$\mathsf{M}(\cdot)$	Multiplication of univariate polynomials, Section 1.3.
Common objects:	
$\mathbb{F}$	A field.
$\mathbb{F}_q$	A finite field with q elements.
$\mathbb{F}[X]_{<m}$	Polynomials over $\mathbb{F}$ of degree less than m.
$\mathbb{Z}, \mathbb{Q}, \mathbb{R}$	Integers, rationals and reals respectively.
$\mathbb{Z}_{>0}, \mathbb{Z}_{\geqslant 0}$	Positive and non-negative integers respectively.
$\mathbb{F}[x]$-modules (Section 1.4):	
$\deg_{\boldsymbol{s}}$	Shifted degree of a polynomial vector.
$\mathrm{rdeg}_{\boldsymbol{s}}, \mathrm{cdeg}_{\boldsymbol{s}}$	Shifted row and column degree of a polynomial matrix.
Algebraic geometry (Section 1.4):	
F	Function field.
$v_P(\cdot)$	Valuation.
$\mathcal{L}(\cdot)$	Riemann-Roch space.
$\mathfrak{l}(\cdot)$	dimension of Riemann-Roch space.
$\mathcal{Я}_P(A)$	$\bigcup_{m=-\infty}^{\infty} \mathcal{L}(mP + A)$.
$\delta_A^{(P)}(f)$	$-v_P(f) - v_P(A)$.
$\mathcal{C}_{\mathcal{L}}(D, G)$	Algebraic geometry code.

Bibliography

[AFLG15] Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: Limitations of the coppersmith-winograd method. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 585–593, New York, NY, USA, 2015. Association for Computing Machinery.

[AKS11] A. Ahmed, R. Koetter, and N. R. Shanbhag. VLSI Architectures for Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Transactions on Information Theory*, 57(2):648–667, February 2011.

[Ale05] M. Alekhnovich. Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed–Solomon Codes. *IEEE Transactions on Information Theory*, 51(7):2257–2265, July 2005.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[BB10] Peter Beelen and Kristian Brander. Efficient list decoding of a class of algebraic-geometry codes. *Advances in Mathematics of Communications*, 4(4):485–518, November 2010.

[BH08] P. Beelen and Tom Høholdt. The Decoding of Algebraic Geometry Codes. In E. Martínez-Moro, editor, *Advances in Algebraic Geometry Codes*, volume 5. World Scientific Publishing Company, 2008.

[BJMS17] A. Bostan, C.-P. Jeannerod, C. Mouilleron, and E. Schost. On matrices with displacement structure: Generalized operators and faster

algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(3):733–775, 2017.

[BLQ13] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin. Polynomial root finding over local rings and application to error correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 24(6):413–443, July 2013.

[BMZ21] Daniele Bartoli, Maria Montanucci, and Giovanni Zini. Weierstrass semigroups at every point of the Suzuki curve. *Acta Arith.*, 197(1):1–20, 2021.

[BRS20] Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast encoding of ag codes over Cab curves. *IEEE Transactions on Information Theory*, 67(3):1641–1655, 2020.

[BRS21] Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast list decoding of algebraic geometry codes. *To be submitted*, 2021.

[CH13] Henry Cohn and Nadia Heninger. Approximate Common Divisors via Lattices. *The Open Book Series*, 1(1):271–293, 2013.

[CJN+15] M.F.I. Chowdhury, C.-P. Jeannerod, V. Neiger, E. Schost, and G. Villard. Faster Algorithms for Multivariate Interpolation With Multiplicities and Simultaneous Polynomial Approximations. *IEEE Transactions on Information Theory*, 61(5):2370–2387, May 2015.

[CK91] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, July 1991.

[CLO07] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra (Undergraduate Texts in Mathematics)*. Springer, 3rd edition edition, 2007.

[CLO15] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 4th edition, 2015.

[CP20] Alain Couvreur and Isabella Panaccione. Power error locating pairs. *Designs, Codes and Cryptography*, 88(8):1561–1593, 2020.

[Dah09] Xavier Dahan. Size of coefficients of lexicographical groöbner bases: the zero-dimensional, radical and bivariate case. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 119–126, 2009.

[DF04] David S. Dummit and Richard M. Foote. Abstract Algebra, John Wiley & Sons. 2004.

[DL78] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[FG05] J Farr and Shuhong Gao. Grobner bases, pade approximation, and decoding of linear codes. *Contemporary Mathematics*, 381:3, 2005.

[FR94] Gui-Liang Feng and Thammavarapu Rao. Simple approach for construction of algebraic-geometric codes from affine plane curves. *Information Theory, IEEE Transactions on*, 40:1003 – 1012, 08 1994.

[FTT91] Albrecht Fröhlich, Martin J Taylor, and Martin J Taylor. *Algebraic number theory.* Number 27. Cambridge University Press, 1991.

[Gao01] Shuhong Gao. Absolute irreducibility of polynomials via newton polytopes. *Journal of Algebra*, 237(2):501 – 520, 2001.

[Gao03] Shuhong Gao. A New Algorithm for Decoding Reed-Solomon Codes. In *Communications, Information and Network Security*, number 712 in The Springer International Series in Engineering and Computer Science, pages 55–68. Springer, January 2003.

[Gei03] Olav Geil. On codes from norm–trace curves. *Finite Fields and Their Applications*, 9(3):351 – 371, 2003.

[GJV03] P. Giorgi, C.P. Jeannerod, and G. Villard. On the Complexity of Polynomial Matrix Computations. In *International Symposium on Symbolic and Algebraic Computation*, pages 135–142, 2003.

[GK09] Massimo Giulietti and Gábor Korchmáros. A new family of maximal curves over a finite field. *Math. Ann.*, 343(1):229–245, 2009.

[GS96] Arnaldo Garcia and Henning Stichtenoth. On the asymptotic behaviour of some towers of function fields over finite fields. *Journal of number theory*, 61(2):248–273, 1996.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved Decoding of Reed–Solomon and Algebraic-Geometric Codes. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 28–37, 1998.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved Decoding of Reed–Solomon Codes and Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[GS00] Shuhong Gao and M Amin Shokrollahi. Computing roots of polynomials over function fields of curves. In *Coding Theory and Cryptography*, pages 214–228. Springer, 2000.

[GV05] V. Guruswami and A. Vardy. Maximum-likelihood decoding of reed-solomon codes is np-hard. *IEEE Transactions on Information Theory*, 51(7):2249–2256, 2005.

[HKT08] J. W. P. Hirschfeld, G. Korchmáros, and F. Torres. *Algebraic curves over a finite field*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2008.

[HLS95] C. Heegard, J. Little, and K. Saints. Systematic encoding via Grobner bases for a class of algebraic-geometric Goppa codes. *IEEE Transactions on Information Theory*, 41(6):1752–1761, 1995.

[HvdH19] David Harvey and Joris van der Hoeven. Faster polynomial multiplication over finite fields using cyclotomic coefficient rings. *Journal of Complexity*, 54:101404, 2019.

[HvLP98] Tom Høholdt, Jacobus H van Lint, and Ruud Pellikaan. Algebraic Geometry Codes. *Handbook of Coding Theory*, 1(Part 1):871–961, 1998.

[JNSV16] Claude-Pierre Jeannerod, Vincent Neiger, Éric Schost, and Gilles Villard. Fast Computation of Minimal Interpolation Bases in Popov Form for Arbitrary Shifts. In *International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 295–302, New York, NY, USA, 2016. ACM.

[JNSV17] Claude-Pierre Jeannerod, Vincent Neiger, Éric Schost, and Gilles Villard. Computing minimal interpolation bases. *Journal of Symbolic Computation*, 83:272–314, November 2017.

[Jus76] J. Justesen. On the complexity of decoding Reed-Solomon codes (Corresp.). *IEEE Transactions on Information Theory*, 22(2):237–238, March 1976.

[Kai80] T Kailath. *Linear Systems*. Prentice-Hall, 1980.

[KO64] A. Karatsuba and Y. Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. *Proceedings of the USSR Academy of Sciences*, 145:293–294, 1964.

[KP95] Christoph Kirfel and Ruud Pellikaan. The minimum distance of codes in an array coming from telescopic semigroups. volume 41, pages 1720–1732. 1995. Special issue on algebraic geometry codes.

[KU08] K. S. Kedlaya and C. Umans. Fast modular composition in any characteristic. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–155, Oct 2008.

[Laz85] Daniel Lazard. Ideal bases and primary decomposition: case of two variables. *Journal of Symbolic Computation*, 1(3):261–270, 1985.

[LBAO14] Kwankyu Lee, M. Bras-Amoros, and M.E. O'Sullivan. Unique Decoding of General AG Codes. *IEEE Transactions on Information Theory*, 60(4):2038–2053, April 2014.

[Le 12] F. Le Gall. Faster algorithms for rectangular matrix multiplication. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 514–523, Oct 2012.

[LG12] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *2012 IEEE 53rd annual symposium on foundations of computer science*, pages 514–523. IEEE, 2012.

[LO08] Kwankyu Lee and Michael E. O'Sullivan. List Decoding of Reed–Solomon Codes from a Gröbner Basis Perspective. *Journal of Symbolic Computation*, 43(9):645 – 658, 2008.

[LO09] Kwankyu Lee and Michael E. O'Sullivan. List decoding of Hermitian codes using Gröbner bases. *Journal of Symbolic Computation*, 44(12):1662–1675, 2009.

[McE03] R.J. McEliece. The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes. *IPN progress report*, pages 42–153, 2003.

[Miu93] Shinji Miura. Algebraic geometric codes on certain plane curves. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 76(12):1–13, January 1993.

[MK93] S. Miura and N. Kamiya. Geometric-goppa codes on some maximal curves and their minimum distance. *Proceedings of 1993 IEEE Information Theory Workshop*, pages 85–86, 06 1993.

[MUT09] Carlos Munuera, A Ulveda, and Fernando Torres. Castle curves and codes. *Advances in Mathematics of Communications*, 3, 11 2009.

[NB15] J.S.R. Nielsen and P. Beelen. Sub-Quadratic Decoding of One-Point Hermitian Codes. *IEEE Transactions on Information Theory*, 61(6):3225–3240, June 2015.

[Nei16] Neiger Vincent. *Bases of relations in one or several variables: fast algorithms and applications*. PhD Thesis, ENS Lyon, November 2016.

[NH00] R Refslund Nielsen and Tom Høholdt. Decoding reed-solomon codes beyond half the minimum distance. In *Coding Theory, Cryptography and Related Areas*, pages 221–236. Springer, 2000.

[Nie13] Johan Sebastian Rosenkilde Nielsen. List decoding of algebraic codes. 2013.

[Nie14] Johan S. R. Nielsen. Power Decoding of Reed–Solomon Codes Revisited. In *International Castle Meeting on Coding Theory and Applications*, September 2014.

[NRS17] Vincent Neiger, Johan Rosenkilde, and Éric Schost. Fast Computation of the Roots of Polynomials Over the Ring of Power Series. In *International Symposium on Symbolic and Algebraic Computation*, July 2017.

[NRS20] Vincent Neiger, Johan Rosenkilde, and Grigory Solomatov. Generic bivariate multi-point evaluation, interpolation and modular composition with precomputation. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, pages 388–395, 2020.

[NV17] Vincent Neiger and Thi Xuan Vu. Computing Canonical Bases of Modules of Univariate Relations. In *International Symposium on Symbolic and Algebraic Computation*, page 8, July 2017.

[NX01] Harald Niederreiter and Chaoping Xing. *Rational points on curves over finite fields: theory and applications*, volume 285 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2001.

[NZ04] Michael Nüsken and Martin Ziegler. Fast multipoint evaluation of bivariate polynomials. In Susanne Albers and Tomasz Radzik, editors, *Algorithms – ESA 2004*, pages 544–555, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[OF02] Henry O'Keeffe and Patrick Fitzpatrick. Gröbner basis solutions of constrained interpolation problems. *Linear algebra and its applications*, 351:533–551, 2002.

[Pan94] Victor Y. Pan. Simple Multivariate Polynomial Multiplication. *Journal of Symbolic Computation*, 18(3):183–186, September 1994.

[PBR17] Sven Puchinger, Irene Bouw, and Johan Rosenkilde né Nielsen. Improved Power Decoding of One-Point Hermitian Codes. In *International Workshop on Coding and Cryptography*, 2017. arXiv:1703.07982.

[PR17] Sven Puchinger and Johan Rosenkilde né Nielsen. Decoding of Interleaved Reed-Solomon Codes Using Improved Power Decoding. *IEEE International Symposium on Information Theory*, 2017.

[PRB19] Sven Puchinger, Johan Rosenkilde, and Irene Bouw. Improved Power Decoding of Interleaved One-Point Hermitian Codes. *Designs, Codes and Cryptography*, 87(2-3):589–607, 2019.

[PRS21] Sven Puchinger, Johan Rosenkilde, and Grigory Solomatov. Improved power decoding of algebraic geometry codes. *arXiv preprint arXiv:2105.00178*, 2021.

[PSP92] Sidney C Porter, B-Z Shen, and Ruud Pellikaan. Decoding geometric goppa codes using an extra place. *IEEE transactions on information theory*, 38(6):1663–1676, 1992.

[RM01] K. Sakaniwa R. Matsumoto, M. Oishi. Fast encoding of algebraic geometry codes. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(10):2514–2517, 2001.

[Ros18] Johan Rosenkilde. Power decoding Reed-Solomon codes up to the Johnson radius. *Advances in Mathematics of Communications*, 12(1):81, 2018.

[RR00] R.M. Roth and G. Ruckenstein. Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. *IEEE Transactions on Information Theory*, 46(1):246 –257, 2000.

[RS19] Johan Rosenkilde and Arne Storjohann. Algorithms for simultaneous Hermite–Padé approximations. *Journal of Symbolic Computation*, In press, October 2019.

[RS21] Johan Rosenkilde and Arne Storjohann. Algorithms for simultaneous hermite–padé approximations. *Journal of Symbolic Computation*, 102:279 – 303, 2021.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Ser] Jean-Pierre Serre. *Rational points on curves over finite fields*, volume 18 of *Documents Mathématiques (Paris)*.

[SH95] K. Saints and C. Heegard. Algebraic-geometric codes and multidimensional cyclic codes: a unified theory and algorithms for decoding using grobner bases. *IEEE Transactions on Information Theory*, 41(6):1733–1751, 1995.

[Sho91] Victor Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, pages 14–21, 1991.

[SSB06] G. Schmidt, V. Sidorenko, and M. Bossert. Decoding Reed-Solomon Codes Beyond Half the Minimum Distance Using Shift-Register Synthesis. In *IEEE International Symposium on Information Theory*, pages 459–463, 2006.

[SSB10] G. Schmidt, V.R. Sidorenko, and M. Bossert. Syndrome Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis. *IEEE Transactions on Information Theory*, 56(10):5245–5252, 2010.

[Sti09] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer, 2nd edition, 2009.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[Sud97] Madhu Sudan. Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. *Journal of Complexity*, 13(1):180–193, 1997.

[SW99] Mohammad Amin Shokrollahi and Hal Wasserman. List Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 45(2):432–437, 1999.

[vdH15] Joris van der Hoeven. On the complexity of multivariate polynomial division. In *Special Sessions in Applications of Computer Algebra*, pages 447–458. Springer, 2015.

[vdHL19] Joris van der Hoeven and Grégoire Lecerf. Fast multivariate multi-point evaluation revisited. *Journal of Complexity*, April 2019.

[vdHL21] Joris van der Hoeven and Grégoire Lecerf. Fast amortized multi-point evaluation. *Journal of Complexity*, page 101574, 2021.

[vdHS13] Joris van der Hoeven and Eric Schost. Multi-point evaluation in higher dimensions. *Applicable Algebra in Engineering, Communication and Computing*, 24(1):37–52, 2013.

[VZG90] Joachim Von Zur Gathen. Functional decomposition ofpolynomials: the tame case. *Journal of Symbolic Computation*, 9(3):281–299, 1990.

[vzGG12] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2012.

[WB86] Lloyd R Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, December 1986. US Patent 4,633,470.

[YB92] T. Yaghoobian and I. F. Blake. Hermitian codes as generalized reed-solomon codes. *Designs, Codes and Cryptography*, 2:5–17, 1992.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings EUROSAM'79*, pages 216–226, 1979.

[ZL13] Wei Zhou and George Labahn. Computing Column Bases of Polynomial Matrices. In *International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 379–386, New York, NY, USA, 2013. ACM.