INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAX101

Thèse de doctorat

On decoding algorithms for algebraic geometry codes beyond half the minimum distance

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École Polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le 03/12/2021, par

Isabella Panaccione

Composition du Jury :

Jean-Pierre TILLICH Directeur de recherche, Inria Paris	Président
Delphine BOUCHER Maîtresse de conférence, Université de Rennes 1, IRMAR	Rapporteuse
Gilles ZÉMOR Professeur, Université de Bordeaux, IMB	Rapporteur
Peter BEELEN Professeur, Technical University of Denmark	Examinateur
Eleonora GUERRINI Maîtresse de conférence, Université de Montpellier, LIRMM	Examinatrice
Emmanuel HALLOUIN Maître de conférence, Université Toulouse Jean Jaurès, IMT	Examinateur
Alain COUVREUR Directeur de recherche, Inria Saclay	Directeur de thèse
Daniel AUGOT Directeur de recherche, Inria Saclay	Co-directeur de thèse
Vincent NEIGER Maître de conférence, Sorbonne Université, LIP6	Invité

Remerciements

Il s'est écoulé près de trois ans depuis que je suis arrivé chez Inria et je sens que ce temps a été suffisant pour me changer profondément. C'est seulement maintenant en écrivant ces mots que je me rends compte de pouvoir être fière d'avoir su affronter trois grands obstacles : une nouvelle vie dans un nouveau pays, une nouvelle langue, un nouveau travail que j'ai pu porter jusqu'à la fin. Je tire de tout cela une grande satisfaction, mais il faut aussi dire que cela n'aurait pu être possible sans aide.

Je commence par remercier Delphine Boucher et Gilles Zémor d'avoir accepté d'être les rapporteurs de mon manuscrit de thèse et d'avoir su m'apporter des observations et des conseils suite à une lecture très approfondie de mon travail. De plus, je tiens à remercier doublement Delphine, car c'est grâce à elle que je me suis penchée sur la théorie des codes et que j'ai ensuite eu la chance de travailler ici.

Je remercie aussi les autres membres du Jury d'avoir accepté de participer à ma soutenance aujourd'hui: Jean-Pierre Tillich, Emmanuel Hallouin, Peter Beelen et Eleonora Guerrini. Vincent, merci beaucoup pour ta présence, mais aussi pour ta patience et tes explications de calcul formel pendant la semaine passée au sein de notre équipe. Emmanuel merci aussi pour tes précieux conseils et remarques. Ça a été un plaisir pour moi de discuter avec toi.

Lorsque l'on fait une thèse, il arrive souvent de ne pas avoir un directeur de thèse présent et d'être laissé seul devant le grand défi de la recherche. Mon cas a été complètement différent. Daniel, je voudrais te remercier pour ton suivi et pour tes précieux conseils. Ton soutien a été aussi important pendant la période hors covid que durant les mois difficiles du confinement. Alain, je me rends compte que les mots qui suivent ne suffisent pas pour exprimer ma gratitude à cent pour cent. Ton soutien pendant ces trois années a été indispensable, tu m'as expliqué tellement de sujets, aidée, écoutée. Même dans les moments de travail intense, tu as toujours trouvé le temps pour suivre mes progrès et pour donner une direction à mes efforts. Merci enfin pour le confort que tu as su me donner dans les instants compliqués et pour m'avoir poussée lorsque c'était nécessaire.

Je voudrais remercier aussi toute l'équipe de Grace dont j'ai pu connaitre le cœur mais aussi les personnes que l'on a vues arriver et puis partir au cours de ces années. Merci pour nos discussions, pauses, groupes de travail, déjeuners. En particulier je voudrais remercier Ben, François et Olivier qui ont assisté à cet exposé plusieurs fois maintenant. Merci pour votre patience et pour vos conseils : ils ont été essentiels. *Grazie Alessandro, Gianira e Martino per i vostri interventi, riflessioni e la vostra passione per la pizza. Martino, grazie ancora per l'incoraggiamento e l'aiuto che mi hai sempre dato. Mi dispiace di non aver ancora un algoritmo per i tuoi codici!*

Je dois un des plus grands merci à Sarah. Merci pour avoir parcouru ce chemin de plus que trois ans avec moi et d'en avoir partagé les moments les plus difficiles, comme les plus joyeux. Merci d'avoir été beaucoup plus que juste "ma collègue de bureau". C'est en grande partie grâce à toi que j'en suis ici aujourd'hui.

Un grand merci aux "filles". Jade, ton amitié et ton honnêté m'ont beaucoup fait grandir. Ta passion et ton envie de savoir ont été contagieuses: je n'arrive plus a compter les fois où une simple discussion avec toi m'a ouvert un monde. *Ilaria, grazie per il tuo affetto, il tuo incoraggiamento, per la tua capacità di ascoltare e per le nostre chiacchierate filosofiche in macchina (per non parlare dei nostri discorsi incredibili sul power decoding!). Grazie, per avermi spinta a credere nelle mie capacità. Spero di averti aiutata*

almeno un po' in questo senso anche io. Elena, grazie per la tua gentilezza e per la tua positività. Grazie soprattutto per i tuoi consigli, che arrivano sempre per magia al momento giusto.

Je pense aussi aux autres petits oisillons (passés et présents) de Grace. En particulier je tiens à remercier Angelo pour sa gentillesse infinie: tu nous manques beaucoup dans l'équipe ! Merci aussi à Maxime pour les petites promenades parisiennes et à Antonin pour ses sourires et les petits "coucou" du matin au laboratoire qui font toujours du bien en début de journée ! *Grazie Giuseppe per le chiacchierate di questi mesi e per la vita sociale che hai riportato nell'équipe. Posso dire che ci hai risvegliati dal lungo sonno del lockdown.* Merci Maxime pour ton envie de discuter et tes pensées qui sont toujours inspirants. Enfin courage Mathilde pour la dernière ligne droite, on y est arrivé !

Chère Agustina, merci pour ton sourire contagieux et ton esprit toujours positif et bienveillant. Tes mots rassurants dans les moments de panique ont fait plus de bien de ce que tu penses.

De la même manière, je pense très fort à tous ceux qui, pendant ces trois ans et même plus que ça, ont joué un rôle déterminant en dehors du laboratoire.

Marie, beaucoup de fois nous avons partagé les mêmes inquiètudes et la même envie de bien faire. En parlant avec toi, j'ai beaucoup compris de moi même et j'ai pris plus de confiance en mes capacités. Merci pour ces moments éclairants et pour ton amitié ! *Grazie Teresa per la ventata di Sud che hai portato cosí all'improvviso. Sei la sola che mi fa uscire il dialetto che non ho mai parlato.*

Fabio e Dons voi siete nei miei ringraziamenti da anni e anni e non è un caso. Nonostante siamo lontani, vi sento nella vita di tutti i giorni, sempre pronti a dare una mano o, quando serve, a dirmi che forse potrei migliorare qualcosa. Siete il mio rifugio a portata di tasca. Caro Felice, anche se ci sono momenti in cui ci sentiamo di più ed altri di meno, le chiacchierate con te fanno sempre bene. Sono quelli i momenti in cui riesco a respirare aria di casa qui a Parigi.

Talla, Anna e Barbara mi sento orgogliosa di noi. Pian piano stiamo crescendo, nonostante i nostri problemi, più o meno grandi, più o meno presenti. Mi sento di dire che il nostro rapporto ha un ruolo determinante in tutto questo. Ed è un rapporto che, straordinariamente, sento alimentarsi sempre più.

Cara Sara, ma quanti messaggi vocali? Quanto hai dovuto sopportarmi? Mi sembrava proprio di parlare con te, core a core, senza filtri. "Ancora un po' di tempo e poi come per magia la preoccupazione svanirà e quello che resterà sarà solo soddisfazione". Che dire, avevi ragione come sempre. Il tempo delle cimici è finito ! Grazie per la tua pazienza e il grande affetto che non manca mai.

Un abbraccio enorme anche ai miei cari "invasati del 92": Irene, Stefano il Rosso, Agnese, Carlo, Stefano, Maurizio, Ivan e Sara. Penso a voi molto spesso. A volte, come all'improvviso, vi ritrovo in alcuni miei comportamenti, decisioni. Cosa farebbe la saggia Irene al posto mio? E Agnese, cercherebbe un diagramma commutativo? La nostalgia di Pisa si fa sentire anche dopo tutto questo tempo.

Ci sono persone che a volte ti aiutano enormemente, ma in punta di piedi. Fra queste, ci sei tu Simo, che hai il potere di farmi sorridere anche quando proprio non ne ho voglia. Il tuo affetto è un toccasana. Cara zia Francesca, non so quante volte mi hai provato che la tua pazienza è infinita. I tuoi lunghi discorsi alla "you can do it!" sono tuttora impressi nella mia mente, da quelli del liceo a quelli che mi hai fatto in questi tre anni e mi motivano tuttora.

E ora non posso non ringraziare i miei Panaccetti. Grazie per aver percorso questa strada insieme a me, per le lunghe videochiamate, per i sorrisi e i tanti "forza" di questi anni. Oggi sono felice di questo traguardo, perché sento che non l'ho tagliato da sola, ma insieme a voi.

Pierre, c'est enfin toi que je veux remercier du fond de mon coeur. Tu m'as dit mille fois que je suis courageuse, que ce n'est pas facile ce que je fais. J'ai enfin réussi à voir mes efforts avec tes yeux et à en être fière. Merci pour ton sourire et pour tes mots qui ont le pouvoir de tout changer.
Merci pour ton immense amour.

Introduction

This thesis deals with algebraic geometry (AG) codes and their decoding. Given a finite field $\mathbb{F}_q$, a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ and a vector $\boldsymbol{y} \in \mathbb{F}_q^n$, decoding $\boldsymbol{y}$ means finding, if exists, an element of $\mathcal{C}$ whose *Hamming distance* from $\boldsymbol{y}$ is less than a certain threshold t. In other words, it means finding a solution to the following problem.

Definition 0.0.1 (Bounded decoding problem)**.** Suppose $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a code and $t \in \mathbb{N}$. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find if exists $\boldsymbol{c} \in \mathcal{C}$ such that

$$\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t.$$

This problem though, has been proven to be NP-complete [BMvT78]. We focus then on particular thresholds t and codes $\mathcal{C}$, for which the problem can be solved. It is well-known that if t is less than half the minimum distance of the code, then there exists at most one solution to the bounded decoding problem. On the other hand, if t exceeds this bound, the uniqueness of the solution is no longer guaranteed. One can then be satisfied with one solution or be rather interested in having the list of all possible solutions, that is, in solving the following problem:

Definition 0.0.2 (List decoding problem)**.** Suppose $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a code and $t \in \mathbb{N}$. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find the set

$$\{\boldsymbol{c} \in \mathcal{C} \mid \mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t\}.$$

We then focus on *bounded decoding algorithms* and *list decoding algorithms*, solving respectively the bounded decoding problem and the list decoding problem for some t. For values of t exceeding half the minimum distance, one can exhibit cases where no solution is the closest one, but there exist two solutions or more at the same distance from $\boldsymbol{y}$. We refer to such cases to as *worst cases*. Given such t and $\boldsymbol{y}$, while bounded decoding algorithms fail, list decoding algorithms return a list containing all solutions. In particular, given this list of solutions one can recover all the closest ones simply computing the distance from $\boldsymbol{y}$ of every element in the list. However, to do so in reasonable time, the number of solutions needs to be controlled some way. A breakthrough in this sense was given by Johnson in 1962 when he provided, for any linear code, a bound for t able to control the size of the list of solutions [Joh62]. In particular, for any t less than this bound, the list decoding problem admits a list of solutions whose size is polynomial in the length of the code n. A class of codes which dispose from both bounded decoding algorithms and list decoding algorithms correcting amount of errors beyond half the minimum distance, is the one of Reed–Solomon codes ([Sud97, GS99, SSB10]). This important amount of algorithms for Reed–Solomon codes is due to their strong structure. They can indeed be seen as vectors of evaluations of bounded degree polynomials in n distinct elements of $\mathbb{F}_q$. Using this representation, their minimum distance is easy to evaluate. One can in particular prove that they are MDS.

Algebraic geometry codes, which extend the class of Reed–Solomon codes, were first introduced by Goppa in 1981 [Gop81]. Given a curve $\mathcal{X}$ and n distinct rational points $\{P_1, \ldots, P_n\} \subseteq \mathcal{X}$, an algebraic geometry code consists in vectors of evaluations of specific functions at the P_i's. These codes gave a breakthrough in coding theory when Tsafsman, Vlădut and Zink proved that Gilbert Varshamov bound could be exceeded when some specific curves where considered [TVZ82]. Furthermore, algebraic geometry codes interested the cryptography scene as well. In particular, since the use of Reed–Solomon codes for McEliece scheme had been proven unsecure by Sidelnikov and Shestakov [SS92], new guarantees of reliability were sought in this larger class of codes.

Unique decoding for algebraic geometry codes

Thanks to their strong underlying algebraic structure, several bounded decoding algorithms were designed to decode algebraic geometry codes. In 1989 Justesen, Larsen, Jensen, Havemose and Høholdt proposed one of the first algorithms for a specific class of algebraic geometry codes [JLJ+89] achieving the correction capacity of $\frac{\mathrm{d}^*-1-g}{2}$, where g is the genus of the curve and d^* is the Goppa designed distance of the code. This algorithm, also called *basic algorithm* in the literature, was then generalized to arbitrary curves by Skorobatov and Vlădut [SV90]. A few years later, Pellikaan and independently Koetter, gave a formulation without algebraic geometry using simply the language of codes. This new interpretation, takes the name "Error Correcting Pairs" (ECP) algorithm and represents a great innovation, since it applies to every code having a certain structure which is described only in terms of component-wise products of codes. The decoding radius of this algorithm depends on the code to which it is applied. For Reed-Solomon codes, it reaches half the minimum distance, which is the threshold for the solution to be unique. For AG codes, the algorithm almost always manages to decode a quantity of errors equal to half the designed distance. However, the algorithm being equivalent, for algebraic geometry codes, to the basic algorithm, its success is only guaranteed for a quantity of errors less than $\frac{\mathrm{d}^*-1-g}{2}$. Several attempts were then made to erase this genus-proportional penalty. In their work, Skorobatov and Vlădut [SV90] improved the decoding radius in some cases. Their result was in turn improved by Duursma [Duu93] who generalised it to all algebraic geometry codes and attained the decoding radius $\frac{\mathrm{d}^*-1}{2}-\sigma$, where σ is the *Clifford defect*. The problem though was not complitely solved, as for instance for plane curves we have in average $\sigma=\frac{g}{4}$. This last algorithm is also referred to as the *modified algorithm*. In parallel to the basic algorithm and with the same correction capacity, we have the algorithm proposed by Porter in his thesis [Por88]. Porter's idea mainly consists in solving a key equation, using a generalization of Euclide's algorithm for functions on curves. Though, the price of this generalisation lies in strong restrictions on the codes and the curves, which entail the correctness of the algorithm only for a small class of codes. In [Ehr92], Ehrhard generalized this algorithm to all curves by solving the key equation of Porter's algorithm with simple linear algebra operations and proved this algorithm to be equivalent to the basic algorithm for a divisor F with no evaluation points in its support. The correctness of the algorithm was proved independently as well by Porter, Shen and Pellikaan in [PSP92], where in addition they succeeded in pushing the decoding radius up to the one of the modified algorithm.

The proof of the existence of an algorithm able to correct up to $\frac{\mathrm{d}^*-1}{2}$ has been given by Pellikaan in [Pel89]. The argument proposed by Pellikaan is based on counting tools exploiting the *Zeta function* and proves the existence of a particular divisor F, such that the basic algorithm, run on F, can correct $\frac{\mathrm{d}^*-1}{2}$ errors. Though, this result only ensures the existence of the divisor F and even if it was extended to almost all curves (Vlădut [Vl0]), no practical procedures to find this specific F were proposed at that time. Finally in [Ehr93], Ehrhard succeeded in finding an algorithm able to construct such a divisor F and hence to achieve the correction capacity $\frac{\mathrm{d}^*-1}{2}$. In the same year Feng and Rao proposed the so called *majority vote for unknown syndromes* [FR93], which also corrects $\frac{\mathrm{d}^*-1}{2}$ errors, but works only for the so-called *one-point* codes. It was then extended to arbitrary divisors in [Pel93, KP95]. We point out that, although the version of the basic algorithm given by Pellikaan and Kötter [Pel92, Köt92] is simpler and versatile because of the use of the language of codes, all the progressive steps aimed at erasing the genus penalty from its decoding radius (in particular, [Ehr93]) were possible thanks to a return to the language of functions and divisors.

Beyond half the designed distance

Starting from the late nineties, several decoding algorithms with correction capacity exceeding half the minimum distance, were designed for Reed–Solomon codes. The first one, a list decoder depending on a parameter ℓ, was proposed by Sudan [Sud97]. The decoding radius t_{max} of the algorithm depends on this parameter, that is, $t_{max}=t_{max}(\ell)$ and, most importantly, Sudan proved that if $t\leq t_{max}(\ell)$, then there exist at most ℓ solutions to the list decoding problem with parameter t.
More than ten years later Schmidt, Sidorenko and Bossert provided a bounded decoding algorithm, also depending on a parameter ℓ, inspired on the decoding of interleaved Reed–Solomon codes [SSB10] and that, later on, took the name of *power decoding* [RnN15]. Despite the different kind of outputs (for

Sudan algorithm, it is a list, while for the power decoding it is a codeword or failure), the two algorithms present almost the same decoding radius (they differ at most by one). Furthermore, in 2003, McEliece proved that for Reed–Solomon codes, if t is less than the Johnson bound, the size of the list of solutions is most of the time one [McE03a]. Therefore, for these codes, bounded decoders rarely fail and, hence, are most of the time reliable just as list decoders. The two algorithms are constructed by exploiting the strong structure of Reed–Solomon codes. Algebraic geometry codes being similarly structured, it has been possible to extend both Sudan's and the power decoding algorithms to them ([SW99, PRS21]). As for the error correcting pairs algorithm, the transition from Reed–Solomon to algebraic geometry codes, produces a genus penalty in the decoding radius of Sudan's algorithm, that is $\frac{\ell g}{\ell+1}$. On the other hand, this penalty does not affect the decoding radius of the power decoding algorithm, which then results to be larger than Sudan's decoding radius. Finally in [GS99], it is given a list decoder extending Sudan algorithm and called Guruswami–Sudan algorithm. Its correction capacity attains the Johnson bound, but at the cost of a potentially high complexity.

Our contributions

Our contributions concern mainly bounded and list decoding algorithms for values of t larger than half the minimum distance of the code.

The PELP algorithm. Although the error correcting pairs algorithm [Pel92, Köt92] does not provide an optimal decoding radius for algebraic geometry codes, it is still an algorithm of interest, since it can extend to other linear codes as, for instance, some cyclic codes (see [DK94]). However, Pellikaan proved in [Pel92] that its decoding radius cannot exceed half the minimum distance of the code. We introduce then a bounded decoding algorithm, we call "Power Error Locating Pairs" (PELP) algorithm, which extends the ECP algorithm to correct superior amounts of errors, preserving at the same time a certain versatility. Indeed, as the ECP, this algorithm can be run on every code disposing from a structure consisting in a pair of codes $(\mathcal{A}, \mathcal{B})$ fulfilling specific properties with respect to the component wise product of codes. This structure though, is slightly different from the one used for the error correcting pair algorithm, since, besides demanding two supplementary hypotheses, it avoids the requirement $\mathrm{d}(\mathcal{B}^{\perp}) > t$. Thanks to this choice of properties then, one can increase the decoding radius of the algorithm. As for the ECP algorithm, the decoding radius of PELP depends on the code the algorithm is applied to. For Reed–Solomon codes, one can prove that the algorithm is equivalent to the power decoding. In particular then, it can correct the same amount of errors and fails in the same cases. For algebraic geometry codes the equivalence to the power decoding appears empirically in the decoding radius and the failure cases. Finally, it is possible to apply the PELP algorithm also to specific cyclic codes, for which the decoding radius exceeds half the Roos bound. Another good property of the PELP algorithm, besides its versatility, is the exploitation of a pair of codes related to $\mathcal{C}$ and not of the code $\mathcal{C}$ itself. This feature can be then applied in cryptography to build an attack for McEliece encryption scheme applied to algebraic geometry codes with amounts of errors beyond half the designed distance of the code. This attack is based on the one proposed by [CMCP17] for amounts of errors under half the designed distance, which uses instead error correcting pairs. Given a divisor G on a curve $\mathcal{X}$, a sequence $\mathcal{P}$ of rational points of $\mathcal{X}$ and a generator matrix of the code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, it is possible to create a sequence of specific pairs of codes which, if sufficiently long, will provide a power error locating pair for $\mathcal{C}$.

An existence result. As seen before, for algebraic geometry codes, there exists a gap between the decoding radius of the power decoding algorithm and the one of Sudan's algorithm. This gap is mainly given by a term proportional to the genus of the curve g. The second main result presented in this thesis is the proof of the existence of a list decoding algorithm for AG codes, recovering at most ℓ solutions and with no genus penalty in its decoding radius. That is, given a divisor G satisfying certain hypotheses and the code $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) \subseteq \mathbb{F}_q^n$, this decoding radius is given by

$$t_{max}(\ell) = \frac{2\ell n - \ell(\ell+1)\deg G - 2}{2(\ell+1)}.$$

This result is inspired to the proof proposed by Pellikaan in [Pel89] for the existence of a bounded decoding algorithm with correction capacity achieving half the designed distance of the code. Given a curve $\mathcal{X}$

and denoting $\mathbb{D}_{g-1}$ the set of effective divisors of degree $g-1$ and by $J(\mathcal{X})$ the Jacobian of $\mathcal{X}$, we show that an algorithm returning at most ℓ solutions exists whenever $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell}$. Moreover we show that, if $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell+1}$, there exists an algorithm finding the entire list of solutions of the list decoding problem. In particular, we prove that there are at most ℓ codewords whose distance from $\boldsymbol{y}$ is bounded by $t_{max}(\ell)$, which, in some cases, provides a better bound on the number of solutions, than the one of Guruswami–Sudan algorithm.

A list decoding algorithm. We finally propose a list decoding algorithm with a decoding radius equal to $t_{max}(\ell)$. Although the proof of its correctness is incomplete, several empiric experiences indicate that it is a good candidate to fulfill the properties exposed in the previous existence result. This algorithm is obtained as a fusion and extension of characterizing tools from Sudan's algorithm in Ehrhard's one. In particular, given an algebraic geometry code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, a divisor F, a parameter ℓ, we consider an instance of the list decoding problem $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\mathrm{w}(\boldsymbol{e}) \leq t_{max}(\ell)$ and $\boldsymbol{c} \in \mathcal{C}$, that is, $\boldsymbol{c}$ is the evaluation at the points of $\mathcal{P}$ of a function $f_{\boldsymbol{c}} \in L(G)$. We introduce two spaces: the one of *Good Sudan polynomials* $GS(F)$ and the one of *All Sudan polynomials* $AS(F)$. The space $GS(F)$ consists in all polynomials of degree ℓ fulfilling some particular properties and vanishing at the function $f_{\boldsymbol{c}}$. Hence, this is the space we want to compute and which, *a priori*, is unknown and cannot be computed efficiently. On the other hand the space $AS(F)$ is a space containing $GS(F)$ and which can be computed using simple linear algebra. The idea is to adapt the divisor F to have the equality $AS(F) = GS(F)$. Indeed, if this equality holds, one can return a list of solutions found among the roots of a nonzero element of $AS(F)$. This adaptation process works as follows: one denotes $F_0 = F$ and inductively $F_{j+1} = F_j - P_{i_j}$, where $P_{i_j} \in \mathcal{P}$ *narrows the gap* between $AS(F_j)$ and $GS(F_j)$, that is, P_{i_j} verifies

$$\dim AS(F_j - P_{i_j}) - \dim GS(F_j - P_{i_j}) < \dim AS(F_j) - \dim GS(F_j).$$

We raise then two particular issues:

- we prove that $\dim AS(F_0) - \dim GS(F_0) \leq t$, that is, the algorithm needs to construct at most t successive divisors $F_1, \ldots, F_t$ to have $AS(F_j) = GS(F_j)$ for some j. However, after t steps of the algorithm one could have $GS(F_j) = \{0\}$;
- since *a priori* one cannot compute $GS(F)$, another issue is to determine whether a point narrows the gap or not between $AS(F)$ and $GS(F)$.

We succeeded in characterizing the points narrowing the gap when $\deg(F_j + G) \geq t + 2g$ and in some other cases. Though, the hypothesis $\deg(F_j + G) \geq t + 2g$ is not realizable for any $j = 1, \ldots, t$ and, hence, theoretically, we are not able to decide for every point whether it narrows the gap or not. However, we show that experimentally the algorithm finds easily points which clearly narrow the gap, terminates in less than t steps and finds the list of all solutions.

Outline of the thesis

In the first chapter, we introduce the notion of linear code, together with some main properties and tools in coding theory. We focus in particular on two decoding problems and on the reason why we decode the way we do. We then present the class of Reed–Solomon codes and the one of algebraic geometry codes, the latter class containing the former. We also provide some algebraic geometry basic tools which will be useful in the entire thesis. We then describe some known decoding algorithms for both Reed–Solomon and algebraic geometry codes, among them, the Error Correcting Pairs algorithm. In Chapter 2, we present our first result, which is the Power Error Locating Pairs algorithm. In particular, the equivalence of this algorithm and the power decoding algorithm for Reed–Solomon codes is proved. It is possible to detect the equivalence with the power decoding also for algebraic geometry codes, thanks to the tests conducted with good starting parameters. For the results of these tests, see Table 2.1. In Chapter 3 we first present the version of the basic algorithm for algebraic geometry codes of Porter. After that, we present the existence result of Pellikaan and the algorithm of Ehrhard. In particular, we give a table of results of tests made on Ehrhard's algorithm, the aim being to show that in practice, the gradual adaptation process for the divisor F can work for almost every point $P \in \mathcal{P}$. Moreover, all these three algorithms have been translated using the language of functions, which is closer to a more modern point

of view and makes the results more intuitive. We then present our result of existence of a list decoding algorithm with a genus penalty free decoding radius $t_{max}(\ell)$. We prove in particular that the size of the list of solutions for the list decoding problem with parameter $t_{max}(\ell)$, is bounded by ℓ. Finally we present a proposition for a list decoding algorithm with decoding radius $t_{max}(\ell)$. We focus in particular on the quantities $\dim AS(F) - \dim AS(F-P)$ and $\dim GS(F) - \dim GS(F-P)$, in order to characterize the points P such that

$$\dim AS(F_j - P_{i_j}) - \dim GS(F_j - P_{i_j}) < \dim AS(F_j) - \dim GS(F_j).$$

A table of tests, showing that the algorithm works, concludes this chapter. Finally, the reader can find a conclusion pointing out the future prospects for these works.

Résumé

Cette thèse porte sur les codes géométriques et leur décodage. Étant donné un corps fini $\mathbb{F}_q$, un code $\mathcal{C} \subseteq \mathbb{F}_q^n$ et un vecteur $\boldsymbol{y} \in \mathbb{F}_q$, "decoder" $\boldsymbol{y}$ signifie trouver, s'il existe, un élément de $\mathcal{C}$ dont la *distance de Hamming* de $\boldsymbol{y}$ est assez petite. Plus précisément, le décodage consiste à résoudre le problème suivant.

Definition 0.0.3 (Problème de décodage borné)**.** Soient $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code et $t \in \mathbb{N}$. Étant donné $\boldsymbol{y} \in \mathbb{F}_q^n$, trouver, s'il existe, $\boldsymbol{c} \in \mathcal{C}$ tel que

$$\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t.$$

Cépendant, ce problème a été prouvé NP-complet [BMvT78]. Nous allons ainsi travailler avec des seuils t et des codes $\mathcal{C}$ pour lesquelles ce problème peut être résolu. La valeur de t peut donner des informations par rapport au nombre de solutions de ce problème. En particulier, si t est inférieur à la moitié de la distance minimale du code, alors il existe au maximum une solution. Par contre, si t dépasse ce seuil, cette propriété d'unicité n'est plus garantie. Dans ce cas, on peut vouloir trouver une seule solution ou plutôt les chercher toutes. Faire cela, est equivalent à résoudre le problème suivant:

Definition 0.0.4 (Problème de décodage en liste)**.** Soient $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code et $t \in \mathbb{N}$. Étant donné $\boldsymbol{y} \in \mathbb{F}_q^n$, trouver l'ensemble

$$\{\boldsymbol{c} \in \mathcal{C} \mid \mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t\}.$$

Nous allons ainsi travailler avec des *algorithmes de decodage borné* et des *algorithmes de decodage en liste*, qui résolvent respectivement le problèmes de décodage borné et de décodage en liste pour certains t. Lorsque t est inférieur à la moitié de la distance minimale ainsi, ces deux types d'algorithmes sont equivalents et retrouvent le mot de code le plus proche de $\boldsymbol{y}$. Toutefois, lorsque t dépasse la moitié de la distance minimale, on peut présenter des cas où il n'y a pas une seule solution la plus proche, mais il y en a deux où plus qui sont à la même distance de $\boldsymbol{y}$. On appelle ces cas *pire cas*. Étant donnés de tels t et $\boldsymbol{y}$, un algorithme de type décodage borné pourrait échouer, alors que un algorithme de décodage en liste donne en sortie une liste contenant toute solution. En particulier, une fois la liste calculée, on peut répérer les solutions les plus proches en calculant la distance de chaque élément de liste de $\boldsymbol{y}$. Toutefois, pour pouvoir faire cela en temps raisonnable, il faut avoir une liste de taille modérée. Un résultat en cette direction a été trouvé par Johnson en 1962. En effet il introduit, pour tout code lineaire, une borne pour t qui, si respectée, assure une taille de liste des solutions polynomiale en la longueur du code n. Une classe de codes équipée avec des algorithmes de décodage borné et en liste, est celle des codes de Reed–Solomon ([Sud97, GS99]). La grande quantité d'algorithmes conçus pour ces codes est dûe en particulier à leur forte structure algébrique. En effet, ils peuvent être décrits comme espaces de vecteurs d'évaluation de polynômes de degré borné en des éléments distincts de $\mathbb{F}_q$. De plus, en utilisant cette représentation, leur distance minimale est facile à calculer. En particulier, on peut prouver que ces codes sont MDS.

La classe des codes géométriques, qui élargit celle des codes de Reed–Solomon, a été introduite premièrement par Goppa en 1981 [Gop81]. Étant donné une courbe $\mathcal{X}$ et n points rationnels distincts $\{P_1, \ldots, P_n\} \subseteq \mathcal{X}$, un code géométrique est constitué de vecteurs d'evaluations de fonctions spécifiques en les P_i. Dès leur découverte, les codes géométriques on suscité un vif intérêt dans la communauté de théorie des codes. En effet seulement après une année, Tsasman, Vlăduţ and Zink ont prouvé que la borne de Gilbert Varshamov pouvait être dépassée en choisissant des courbes specifiques [TVZ82]. De plus, les codes géométriques ont été considérés pour des applications cryptographiques [JM96]. En effet, comme l'utilisation des codes de Reed–Solomon pour le schéma de McEliece avait été prouvé pas fiable par Sidelnikov et Shestakov [SS92], de nouvelles garanties de fiabilité ont été cherchées dans cette plus large classe de codes.

Décodage unique pour les codes géométriques

La structure algébrique sous-jacente de ces codes a permis de concevoir plusieurs algorithmes de décodage. Un premier algorithme pour les codes provenant de courbes planes est proposé en 1989 par Justesen, Larsen, Jensen, Havemose et Hoholdt [JLJ+89] avec le rayon de décodage $\frac{d^*-1-g}{2}$, où g est le genre de la courbe et d^* est la distance construite de Goppa du code. Cet algorithme, appelé *basic algorithm* dans la litterature, est en suite généralisé à toute courbe par Skorobatov et Vlădut [SV90]. Après quelques années, Pellikaan et, independemment, Kötter, en donnent une formulation sans géométrie algébrique utilisant simplement le langage des codes. Cette nouvelle interprétation prend le nom d'algorithme *Error Correcting Pairs* (ECP) et représente une percée en théorie des codes, car l'algorithme s'applique à tout code muni d'une certaine structure qui se décrit uniquement en termes de produits coordonnées par coordonnées de codes. Le rayon de décodage de cet algorithme dépend du code auquel il est appliqué. Pour les codes de Reed-Solomon, il atteint la moitié de la distance minimale, seuil d'unicité de la solution. Pour les codes géométriques, l'algorithme arrive à décoder presque toujours une quantité d'erreurs égale à la moitié de la distance construite. Toutefois, étant équivalent au *basic algorithm* pour les codes géométriques, le bon fonctionnement de l'algorithme n'est garanti que pour une quantité d'erreurs inférieure à $\frac{d^*-1-g}{2}$. Plusieurs tentatives ont ensuite été menées pour effacer cette penalité dûe au genre. Dans leur travail, Skorobatov et Vlădut [SV90] améliorent le rayon de décodage dans certains cas. Leur résultat est à son tour amélioré par Duursma [Duu93], qui l'étend à tout code géométrique et pousse le rayon de décodage à $\frac{d^*-1}{2} - \sigma$, où σ est le *défaut de Clifford*. Le problème pourtant, n'est pas complétement résolu, comme par exemple pour les courbes planes on a typiquement $\sigma = \frac{g}{2}$. Ce dernier algorithme est aussi appelé *modified algorithm* dans la littérature. En parallèle au *basic algorithm* et avec le même rayon de décodage, on trouve l'algorithme proposé par Porter dans sa thèse [Por88]. L'idée de Porter consiste à résoudre une équation clé en utilisant une généralisation de l'algorithme d'Euclide pour des fonctions sur une courbe. Cependant, le prix de cette généralisation est de fortes restrictions sur le code et la courbe, ce qui assure le bon fonctionnement de l'algorithme seulement pour une petite classe de codes. Dans [Ehr92], Ehrhard étend cet algorithme à toute courbe en résolvant l'equation clé de l'algorithme de Porter avec de simples outils d'algèbre linéaire. De plus, il prouve que cet algorithme est equivalent au *basic algorithm*, si le diviseur choisi F n'a pas de points d'évaluation dans son support. Le bon fonctionnement de l'algorithme est prouvé indépendemment par Porter, Shen et Pellikaan en [PSP92], où en plus, ils arrivent à pousser le rayon de décodage jusqu'à atteindre celui du *modified algorithm*.

La preuve de l'existence d'un algorithme de décodage capable de corriger $\frac{d^*-1}{2}$ erreurs a été donnée par Pellikaan en [Pel89]. Cette preuve est basée sur des arguments de comptage utilisant la *fonction Zeta* et montre qu'il existe un diviseur spécifique F, tel que le *basic algorithm* appliqué à un tel F a un rayon de décodage égal à la moitié de la distance construite. Toutefois, ce résultat garantit seulement l'existence d'un tel diviseur F et, même s'il a été étendu presque à toute courbe ([Vl0]), une procédure pour construire ce F n'a pas été immédiatement trouvée. Ehrhardt [Ehr93] fournit une algorithme permettant de calculer un tel F et ainsi de corriger jusqu'à $\frac{d^*-1}{2}$ erreurs. Son idée consiste à construire graduellement un bon diviseur F, puis lui appliquer le *basic algorithm*. Dans la même année, Feng et Rao proposent un algorithme appelé *majority voting for unknown syndromes* [FR93], qui corrige aussi $\frac{d^*-1}{2}$ erreurs, mais seulement pour des codes dont le diviseur est supporté par un seul point. Cet algorithme est après étendu à tout diviseur en [Pel93, KP95]. Nous voulons signaler que, bien que la version du *basic algorithm* donnée par Pellikaan et Kötter [Pel92, Köt92] soit plus simple et polyvalente grâce à l'utilisation du langage des codes, toutes les étapes progressives visant à éliminer la pénalité du genre du rayon de décodage, ont été possibles grâce à un retour au langage des fonctions et des diviseurs.

Au délà de la distance construite

À partir de la fin des années 90, plusieurs algorithmes de décodage avec rayon de décodage au délà de la moitié de la distance minimale, ont été conçus pour les codes de Reed–Solomon. Le premier, un algorithme de décodage en liste dépendant d'un paramètre ℓ et avec rayon de décodage $t_{max}(\ell)$, est proposé par Sudan [Sud97]. Une conséquence de ce résultat est que pour tout $t \leq t_{max}(\ell)$, il existe au maximum ℓ solutions pour le problème de décodage avec paramètre t.
Quelques années après, Schmidt, Sidorenko and Bossert proposent un algorithme de décodage borné,

dependant de même d'un paramètre ℓ, et inspiré par un algorithme de décodage des codes de Reed–Solomon entrelacés [SSB10]. Cet algorithme prend en suite le nom de *power decoding* [RnN15]. Malgré les types différents de sorties (pour l'algorithme de Sudan, c'est une liste, alors que pour le *power decoding* c'est un mot de code ou l'échec), les deux algorithmes ont à peu près le même rayon de décodage (ils différent au maximum de un). De plus, en 2003, McEliece prove que pour les codes de Reed–Solomon, si t est inférieur à la borne de Johnson, alors la taille de la liste des solutions presque toujours égale à un [McE03b]. On a alors que pour ces codes, les algorithmes de décodage borné échouent très rarement et, ainsi, ils sont presque aussi fiables que les algorithmes de décodage en liste. Les deux algorithmes sont construits en se basant sur la forte structure algébrique des codes de Reed–Solomon. Comme les codes géométriques sont construits de façon similaire, il a été possible d'étendre ces deux algorithmes à cette classe. Comme pour l'algorithme des *error correcting pairs*, la transition des codes de Reed–Solomon aux codes géométriques, crée une penalité proportionnelle au genre dans de rayon de décodage de l'algorithme de Sudan. En revanche, cette penalité n'affecte pas le rayon de décodage de l'algorithme du *power decoding*, qui ainsi resulte supérieur à celui de Sudan. Enfin, dans [GS99], il est présenté un algorithme de décodage en liste qui étend celui de Sudan et qui est appelé algorithme de Guruswami-Sudan. Son rayon de décodage atteint la borne de Johnson, mais le prix de cette amélioration est une potentielle augmentation de la complexité.

Nos contributions

Nos contributions portent en particulier sur des algorithmes de décodage borné et en liste pour des valeurs de t supérieur à la moitié de la distance minimale.

L'algorithme PELP. Bien que l'algorithme des *error correcting pairs* (ECP) [Pel92, Köt92] ne donne pas un rayon de décodage optimal pour les codes géométriques, il reste un algorithme particulièrement intéressant, notamment grâce à sa polyvalence. En revanche, Pellikaan en [Pel92] montre que le rayon de décodage de cet algorithme ne peut pas dépasser la moitié de la distance minimale du code. Nous proposons ainsi un algorithme de décodage borné, que nous appelons algorithme des *Power Error Locating Pairs* (PELP), qui étend l'algorithme des ECP en corrigeant un plus grand nombre d'erreurs et garde au même temps une certaine polyvalence. En effet, comme l'algorithme des ECP, cet algorithme peut être appliqué à tout code muni d'une structure qui se décrit uniquement en termes de produit coordonnées par coordonnées de codes. Cette structure est légèrement différente de celle de ECP comme, au delà de demander des hypothèses supplémentaires, elle ne demande pas la propriété $\mathrm{d}(\mathcal{B}^{\perp}) > t$. Ainsi, grâce à ce choix de propriétés, il est possible de pousser le rayon de décodage au delà de la moitié de la distance minimale. Comme pour l'algorithme des ECP, ce rayon de décodage dépend du code auquel il est appliqué. Pour les codes de Reed–Solomon, on peut prouver que cet algorithme est equivalent au *power decoding*. En particuler, il peut corriger le même nombre d'erreurs et échoue dans les même cas. Pour les codes géométriques l'equivalence avec le *power decoding* apparaît uniquement d'un point de vue empirique. Finalement, on peut appliquer l'algorithme des PELP aussi à des codes cycliques spécifiques. Pour ces codes, le rayon de décodage de l'algorithme dépasse la moitié de la borne de Roos.
Une autre bonne propriété de l'algorithme des PELP, au delà de sa polyvalence, c'est qu'il consiste à utiliser un couple de codes liés à $\mathcal{C}$ plutôt que le code $\mathcal{C}$ lui-même. Cette propriété peut être appliquée en cryptographie pour construire une attaque pour le schéma de McEliece pour les codes géométriques avec un nombre d'erreurs supérieur à la moitié de la distance construite. Cette attaque étend celle proposée par [CMCP17]. Étant donnée une matrice génératrice pour le code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, on peut créer une suite de couples de codes spécifiques. Si elle est suffisamment longue, cette suite a la propriété qu'un de ses termes est un *power error locating pair* pour $\mathcal{C}$.

Une preuve d'existence. On a vu que, étant donné un code géométrique $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, les rayons de decodage des algorithmes de Sudan et du *power decoding* ne sont pas equivalents. Cette différence est donnée principalement par un terme proportionnel au genre de la courbe g. Le deuxième résultat principal de cette thèse est la preuve de l'existence d'un algorithme de décodage en liste pour les codes géométriques, capable de trouver ℓ solutions et avec un rayon de décodage privé de cette pénalité dûe au

genre. En particulier, étant donné un code $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) \subseteq \mathbb{F}_q^n$, ce rayon de décodage est donné par

$$t_{max}(\ell) = \frac{2\ell n - \ell(\ell+1)\deg G - 2}{2(\ell+1)}.$$

Ce résultat est inspiré par la preuve donnée par Pellikaan en [Pel89] pour l'existence d'un algorithme de décodage borné avec rayon de décodage égal à la moitié de la distance construite. Étant donnée une courbe $\mathcal{X}$, soit $\mathbb{D}_{g-1}$ l'ensemble des diviseurs effectifs de degree $g-1$ et soit $J(\mathcal{X})$ la Jacobienne de $\mathcal{X}$. Nous montrons que, si $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell}$, alors il existe un algorithme trouvant ℓ solutions. De plus, nous montrons que, si $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell+1}$, alors il existe un algorithme qui trouve la liste complète des solutions ou, en d'autres termes, qu'il y a au maximum ℓ mots de code à une distance de $\boldsymbol{y}$ inférieure à $t_{max}(\ell)$. Ce résultat, donne une borne plus précise sur le nombre de solutions, que celle donnée par l'algorithme de Guruswami–Sudan.

Un algorithme de décodage en liste. Enfin, nous proposons un algorithme de décodage en liste, avec un rayon de décodage égal à $t_{max}(\ell)$. Bien que la preuve de son bon fonctionnement ne soit pas complète, plusieurs tests indiquent que c'est un bon candidat pour être un algorithme vérifiant les propriétés annoncées par le précédent résultat d'existence. Cet algorithme est obtenu par la fusion et l'extension des outils de l'algorithme de Sudan, dans l'algorithme de Ehrhard. En particulier, étant donné un code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, un diviseur F et un paramètre ℓ, nous considerons une instance du problème de décodage en liste $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, où $\mathrm{w}(\boldsymbol{e}) \leq t_{max}(\ell)$ et $\boldsymbol{c} \in \mathcal{C}$. On a ainsi $\boldsymbol{c} = \mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}})$ avec $f_{\boldsymbol{c}} \in L(G)$. Nous introduisons deux espaces: celui des *Bons polynômes de Sudan* $GS(F)$ et celui de *Tous les polynômes de Sudan* $AS(F)$. L'espace $GS(F)$ contient tous les polynômes de degré ℓ qui vérifient des propriétés particulières et s'annulent en $f_{\boldsymbol{c}}$. Ainsi, $GS(F)$ c'est l'espace que l'on cherche à calculer et qui, *a priori* n'est pas facile à trouver efficacement. En revanche, l'espace $AS(F)$ contient $GS(F)$ et il peut être calculé en utilisant de simples outils d'algèbre linéaire. L'idée est d'adapter le diviseur F pour garantir l'égalité $AS(F) = GS(F)$. En effet, si cette égalité est vérifiée, la liste complète des solutions au problème de décodage est contenue dans l'ensemble des racines de tout élément non nul de $AS(F)$. Ce processus d'adaptation fonctionne de la manière suivante: on note $F_0 = F$ et par récurrence $F_{j+1} = F_j - P_{i_j}$, où $P_{i_j} \in \mathcal{P}$ *réduit l'écart* entre $AS(F_j)$ et $BS(F_j)$ ou, en d'autres termes:

$$\dim AS(F_j - P_{i_j}) - \dim BS(F_j - P_{i_j}) < \dim AS(F_j) - \dim GS(F_j).$$

Nous soulevons les problèmatiques suivantes:

- nous prouvons que $\dim AS(F_0) - \dim GS(F_0) \leq t$. C'est à dire que l'algorithme a besoin de construire au maximum t diviseurs $F_1, \ldots, F_t$ pour avoir $AS(F_j) = GS(F_j)$ pour quelque j. Pourtant, après t étapes de l'algorithme, on pourrait avoir $GS(F_j) = \{0\}$;
- comme *a priori* on ne peut pas calculer $GS(F)$, un autre problème est celui de déterminer quand un point réduit l'écart ou pas entre $AS(F)$ et $GS(F)$.

Nous arrivons à caractériser les points qui réduisent l'écart lorsque $\deg(F_j + G) \geq t + 2g$ et dans quelques autres cas. Toutefois, l'hypothèse $\deg(F_j + G) \geq t + 2g$ n'est pas réalisable pour tout $j = 1, \ldots, t$, ainsi, théoriquement nous ne sommes pas capable de décider pour chaque point s'il réduit l'écart ou pas. En revanche, nous montrons que experimentalement l'algorithme trouve facilement des points qui clairement serrent le gap. De plus nous montrons que dans nos tests, l'algorithme termine en moins de t étapes et trouve la liste de toutes les solutions.

Contents

1 Codes and decoding algorithms **17**
1.1 Channels . . . 17
1.1.1 The binary symmetric channel . . . 17
1.1.2 The q-ary symmetric channel . . . 17
1.2 Coding . . . 18
1.3 Decoding . . . 19
1.4 Some properties and constructions on codes . . . 21
1.4.1 Star product of words and codes . . . 22
1.4.2 A Kneser–like theorem . . . 22
1.5 Reed–Solomon codes . . . 22
1.6 Basic algorithms for Reed–Solomon codes . . . 23
1.6.1 Unique decoding : Welch-Berlekamp algorithm . . . 24
1.6.2 Unique decoding: the Error Correcting Pairs algorithm . . . 25
1.6.3 Beyond the unique decoding bound: Sudan's algorithm . . . 29
1.6.4 Beyond the unique decoding bound: the power decoding algorithm . . . 31
1.7 Algebraic geometry codes . . . 33
1.7.1 Algebraic geometry prerequisites . . . 33
1.7.2 Riemann–Roch Theorem . . . 38
1.7.3 Algebraic geometry codes . . . 39
1.8 Basic algorithms for algebraic geometry codes . . . 40
1.8.1 Unique decoding : the basic algorithm . . . 42
1.8.2 Beyond half the designed distance : power decoding algorithm . . . 42
1.8.3 Considerations on the decoding radius of Sudan's algorithm . . . 44
1.9 Other algebraic codes . . . 45

2 The Power Error Locating Pairs algorithm **47**
2.1 Power error locating pairs algorithm . . . 47
2.1.1 The case $\ell = 2$ . . . 48
2.1.2 The general case: $\ell \geq 2$ . . . 51
2.1.3 Complexity . . . 52
2.2 ℓ–PELP algorithm for Reed–Solomon codes . . . 52
2.2.1 The space M and the key equations of power decoding . . . 53
2.2.2 Equivalence of the two algorithms for Reed–Solomon codes with $\ell = 2$ . . . 54
2.3 PELP algorithm for algebraic geometry codes . . . 54
2.3.1 Context . . . 55
2.3.2 Decoding Radius . . . 55
2.3.3 Comparison with decoding radii of other algorithms for algebraic geometry codes . . . 56
2.3.4 Cryptanalytic application . . . 57
2.4 PELP algorithm for cyclic codes . . . 57
2.4.1 Roos bound . . . 58
2.4.2 ℓ–PELP algorithm and Roos bound . . . 59
2.4.3 Comparison with Roos bound . . . 61

3 An Ehrhard-like list decoding algorithm for algebraic geometry codes with no genus penalty **63**
3.1 Porter's algorithm 63
3.1.1 The key equation 66
3.2 Erasing the genus penalty: unique decoding 66
3.2.1 Some prerequisites 67
3.2.2 Pellikaan's result: the existence of an algorithm with no genus penalty 67
3.2.3 Ehrhard's algorithm 69
3.3 Erasing the genus penalty: beyond the unique decoding bound 73
3.3.1 Context 73
3.3.2 A reformulation of the spaces $S(F)$ and $L(F - D_{\boldsymbol{e}})$ 74
3.3.3 The existence of a list decoding algorithm with no genus penalty 77
3.3.4 A new algorithm for $\ell = 2$ 80
3.3.5 Some empirical considerations 86

Chapter 1

Codes and decoding algorithms

In this first chapter we introduce error correcting codes and the decoding problems we are going to face along this thesis. In particular we want to show why we decode in a certain way and what we expect it to be doable or not.

Most of the times the communication between two systems is modeled with a noisy channel. In particular, if messages consist in strings of elements of a certain alphabet $\mathbb{F}$, the noise of the channel may affect some characters of these messages during the transmission, with the effect that the message at the output of the channel is different from the message at the input.

1.1 Channels

All channels we will talk about are *memoryless*, which means that if the noise causes an error on a digit of our message, this error will be independent from the rest of the transmitted digits. In particular, the main models of channel we will work on, are the *binary symmetric channel* and the *q-ary symmetric channel*.

1.1.1 The binary symmetric channel

In this channel model we suppose that the digits which can be transmitted across the channel can only be 0 or 1. To a binary symmetric channel it is associated a parameter $0 \leq p \leq 1$ which is the probability that the channel flips a digit. In other words, if $Out(x)$ is the output of the channel when the digit $x \in \{0, 1\}$ is sent, then

$$Out(0) = \begin{cases} 1 \text{ with probability } p \\ 0 \text{ with probability } (1-p) \end{cases} \qquad Out(1) = \begin{cases} 0 \text{ with probability } p \\ 1 \text{ with probability } (1-p), \end{cases}$$

from which the word *symmetric*.

1.1.2 The q-ary symmetric channel

Binary symmetric channels can be seen as channels which transmit elements of $\mathbb{F}_2$ and so, they can be generalized to q-ary symmetric channels in this way: let us suppose that the channel can only transmit the elements of $\mathbb{F}_q$. Then, this channel is a q-ary symmetric channel if there exists a parameter $0 \leq p \leq 1$ such that for any $x \in \mathbb{F}_q$ we have

$$Out(x) = \begin{cases} y \neq x \text{ with probability } \frac{p}{q-1} \\ x \text{ with probability } (1-p). \end{cases}$$

1.2 Coding

A way to face the problem of the channel noise is to add *redundancy* to the messages, which is exactly one of the uses of error correcting codes. Indeed, let us suppose that for a certain $k \in \mathbb{N}$, $\mathbb{F}_q^k$ is the set of all possible messages. A way to add some redundancy is to associate to every message, an element of $\mathbb{F}_q^n$ with $n \geq k$. That is, we need to define a so called *encoding function*, which is an injective map φ

$$\varphi : \begin{cases} \mathbb{F}_q^k & \longrightarrow & \mathbb{F}_q^n \\ \boldsymbol{m} & \longmapsto & \varphi(\boldsymbol{m}). \end{cases}$$

If φ is linear, then $\mathrm{Im}(\varphi)$ is a subspace of $\mathbb{F}_q^n$ with dimension over $\mathbb{F}_q$ equal to k.

Definition 1.2.1. Given a finite field $\mathbb{F}_q$ and an integer n, a *linear code* of *length* n over $\mathbb{F}_q$ is a subspace $\mathcal{C}$ of $\mathbb{F}_q^n$. The elements of $\mathcal{C}$ are called *codewords* and its dimension over $\mathbb{F}_q$ is called the *dimension* of $\mathcal{C}$.

Given a linear code $\mathcal{C}$, we will denote its dimension by $\dim \mathcal{C}$ or k if there is no ambiguity on the code. It is also possible to define nonlinear codes, but all along this thesis we will assume all codes we will talk about to be linear. We now introduce one last parameter of a code, which is its minimum distance. To do so, we first have to introduce the Hamming distance.

Definition 1.2.2. Given two vectors $\boldsymbol{a} = (a_1, \ldots, a_n), \boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{F}_q^n$, their *Hamming distance* is

$$\mathrm{d}(\boldsymbol{a}, \boldsymbol{b}) \stackrel{\text{def}}{=} \#\{i \in \{1, \ldots, n\} \mid a_i \neq b_i\}.$$

The *weight* of $\boldsymbol{a}$ is defined as $\mathrm{w}(\boldsymbol{a}) \stackrel{\text{def}}{=} \mathrm{d}(\boldsymbol{a}, \boldsymbol{0})$. Finally, the *support* of a vector $\boldsymbol{a} \in \mathbb{F}_q^n$ is the subset $\mathrm{supp}(\boldsymbol{a}) \subseteq \{1, \ldots, n\}$ of indexes i satisfying $a_i \neq 0$.

Definition 1.2.3. Given a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, the *minimum distance* of $\mathcal{C}$ is

$$\mathrm{d}(\mathcal{C}) \stackrel{\text{def}}{=} \min_{\substack{\boldsymbol{a}, \boldsymbol{b} \in \mathcal{C} \\ \boldsymbol{a} \neq \boldsymbol{b}}} \mathrm{d}(\boldsymbol{a}, \boldsymbol{b}).$$

In the rest of the paper we will write sometimes d instead of $\mathrm{d}(\mathcal{C})$ when there is no ambiguity on the code. We recall that if the code $\mathcal{C}$ is linear, which is the case for the codes considered in this thesis, we have

$$\mathrm{d}(\mathcal{C}) = \min_{\boldsymbol{a} \in \mathcal{C} \setminus \{\boldsymbol{0}\}} \mathrm{w}(\boldsymbol{a}).$$

We point out that, given a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, its minimum distance controls the Hamming distance between the codewords of $\mathcal{C}$. This means that if the channel flips a few digits of a codeword and the minimum distance is large, then the message in output is not luckily to belong to the code. In particular, the receiver will be aware that this message does not correspond to the input and that some errors occurred. On the other hand, one may want also to contain the redundancy $n - k(\mathcal{C})$, for instance, for memory reasons. Ideally then a good code should have both large dimension and minimum distance with respect to n, but these two properties cannot be satisfied at the same time, as expressed by the following bound.

Proposition 1.2.4 (Singleton bound)**.** *For any code $\mathcal{C} \subseteq \mathbb{F}_q^n$, we have* $\mathrm{d}(\mathcal{C}) \leq n - k(\mathcal{C}) + 1$.

A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is said *Maximum Distance Separable* (MDS) if its parameters satisfy

$$\mathrm{d}(\mathcal{C}) = n - k(\mathcal{C}) + 1,$$

that is, if its minimum distance is the largest possible with respect to the dimension $k(\mathcal{C})$.

1.3 Decoding

If it is easy to have an intuition about how coding works, it is not that elementary to imagine what *decoding* could mean after all. Let us suppose we want to send a message $\boldsymbol{m}$ through the channel. Then, first we compute the codeword of $\mathcal{C}$ associated to $\boldsymbol{m}$, that is $\boldsymbol{c} \stackrel{\text{def}}{=} \varphi(\boldsymbol{m})$, and we send $\boldsymbol{c}$. At the output of the channel we will get a vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_q^n$, where $\boldsymbol{e} \in \mathbb{F}_q^n$ is the so called *error vector* added by the noise of the channel. The aim of a *decoder* or *decoding algorithm* is then to recover $\boldsymbol{m}$. Note that since φ is injective, the knowledge of $\boldsymbol{m}$ implies the knowledge of $\boldsymbol{c}$ and *vice versa*. In this thesis, we will hence consider decoders which recover $\boldsymbol{c}$ instead. It is impossible to dispose of a decoder which is able to recover $\boldsymbol{c}$ for any error vector $\boldsymbol{e}$, hence usually the following notion is the used one.

Definition 1.3.1. Given a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, a *decoder* for $\mathcal{C}$ is a map $\psi : \mathbb{F}_q^n \to \mathcal{C} \cup \{?\}$ such that $\psi(\boldsymbol{c}) = \boldsymbol{c}$ for any $\boldsymbol{c} \in \mathcal{C}$. A *decoding algorithm* for a code $\mathcal{C}$ is an algorithm which implements a decoder for $\mathcal{C}$.

The symbol "?" describes the case where the decoder fails in recovering any codeword from the vector $\boldsymbol{y}$. In what follows we will focus on decoding algorithms responding to particular decoding problems. Two of the most intuitive decoding problems are the maximum–likelihood decoding problem and the nearest–codeword decoding problem.

Definition 1.3.2 (Maximum–likelihood decoding problem)**.** Given a code $\mathcal{C}$ and a q-ary symmetric channel qSC, for any $\boldsymbol{y} \in \mathbb{F}_q^n$, find $\boldsymbol{c} \in \mathcal{C}$ maximizing the conditional probability

$$\mathbb{P}_{\boldsymbol{e} \sim qSC}(\boldsymbol{y} \text{ received } \mid \boldsymbol{c} \text{ sent}). \tag{1.1}$$

Definition 1.3.3 (Nearest–codeword decoding problem)**.** Given a code $\mathcal{C}$, for any $\boldsymbol{y} \in \mathbb{F}_q^n$, find the codeword which is the closest to $\boldsymbol{y}$ with respect to the Hamming distance, that is, a $\boldsymbol{c} \in \mathcal{C}$ which minimizes $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c})$.

Remark 1.3.4. Note that the ambiguous instances of these problems can be treated and adjusted. Indeed if there are two codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ maximizing (1.1) or minimizing $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c})$, one can decide to choose one of them depending on some preference criteria.

Remark 1.3.5. There exists a decoding algorithm which solves both the problems, which is the brute force algorithm: it consists in computing the conditional probability in (1.1) or the quantity $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c})$ for any $\boldsymbol{c} \in \mathcal{C}$ and looking respectively for the maximum or the minimum of these values. Of course the complexity of this algorithm is of $O(|\mathcal{C}|) = O(q^k)$.

Let us now focus on these two problems. On the one hand we look for the codeword which, with the highest probability, is the input of the received vector $\boldsymbol{y}$. On the other hand, in the nearest–codeword decoding problem, we look for the codeword which is the closest to $\boldsymbol{y}$. The closest codeword though is not necessarily the most probable input, especially if the channel is very noisy. However under the right circumstances these two problems are equivalent.

Theorem 1.3.6. *Suppose the messages coded by $\mathcal{C}$ are transmitted through a q–ary symmetric channel with probability parameter $p < 1 - 1/q$. Then the maximum–likelihood decoding problem and the nearest–codeword decoding problem have the same solution.*

Proof. For $q = 2$, see [Rot06, Example 1.6]. It is then easy to generalize the proof to every prime power. □

From now on we will suppose we work with q–ary symmetric channels with probability parameter fulfilling the hypothesis in Theorem 1.3.6. Hence, given a vector $\boldsymbol{y} \in \mathbb{F}_q^n$, our purpose will be to describe algorithms able to find the closest codeword to $\boldsymbol{y}$. It is well-known that this problem is NP–complete [BMvT78], hence in order to build efficient algorithms, we consider the following subproblem for specific choices of parameters.

Definition 1.3.7 (Bounded decoding problem)**.** Suppose $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a code and $t \in \mathbb{N}$. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find if exists, $\boldsymbol{c} \in \mathcal{C}$ such that

$$\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t.$$

Definition 1.3.8. In the following, all algorithms solving the bounded decoding problem will be referred to as *bounded decoding algorithms*. For a decoding algorithm solving the bounded decoding problem, the largest possible t such that the algorithm works is called its *decoding radius*.

Now, depending on the value of t in the bounded decoding problem, it is possible to have some information about the number of solutions one could have.

Theorem 1.3.9. *Let $\mathcal{C}$ be a code and d its minimum distance. Then, if $t \leq \frac{\mathrm{d}-1}{2}$, for any $\boldsymbol{y} \in \mathbb{F}_q^n$ there exists at most one solution to the bounded decoding problem with parameter t.*

This theorem tells us that, whenever we have a vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $\mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}-1}{2}$, then $\boldsymbol{c}$ is the only solution to the bounded decoding problem with $t = \frac{\mathrm{d}-1}{2}$. Though, given a generic code that does not mean that we dispose of an efficient algorithm to find this solution. We will present later in this chapter a class of codes for which several efficient decoding algorithms do exist for such a t. Later on we will refer to the amount $\frac{d-1}{2}$ as to the *unique decoding bound*.

Definition 1.3.10. Given a vector $\boldsymbol{x} \in \mathbb{F}_q^n$, the Hamming ball of radius $\rho \geq 0$ centered in $\boldsymbol{x}$ is the subset $\mathcal{B}(\boldsymbol{x}, \rho) \subseteq \mathbb{F}_q^n$ defined as

$$\mathcal{B}(\boldsymbol{x}, \rho) \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathbb{F}_q^n \mid \mathrm{d}(\boldsymbol{x}, \boldsymbol{a}) \leq \rho\}.$$

A consequence of Theorem 1.3.9 is that, given a code $\mathcal{C}$, the Hamming balls of radius $\rho = \frac{d-1}{2}$ centered in the elements of $\mathcal{C}$ do not intersect. Moreover, note that

$$\#\mathcal{B}(\boldsymbol{c}, \rho) = \#\mathcal{B}(\boldsymbol{0}, \rho)$$

for any $\boldsymbol{c} \in \mathcal{C}$. We get then the following result.

Theorem 1.3.11 (Sphere Packing bound)**.** *Given a code of length n, dimension k and minimum distance d over $\mathbb{F}_q$, then*

$$q^k \cdot \#\mathcal{B}\left(\boldsymbol{0}, \frac{\mathrm{d}-1}{2}\right) \leq q^n$$

If the equality holds for a code $\mathcal{C}$, then every vector of $\mathbb{F}_q^n$ is contained in a Hamming ball centered in a codeword $\boldsymbol{c}$ and, hence, can be decoded with no ambiguity with $\boldsymbol{c}$. In this case the code is said to be *perfect*. Perfect codes are anyway really rare and most of the times there exist several vectors of $\mathbb{F}_q^n$ which are farther than $\frac{\mathrm{d}-1}{2}$ from the code. Let us then consider the bounded decoding problem with $t > \frac{\mathrm{d}-1}{2}$. We know that the uniqueness of the solution is no longer guaranteed. In particular, suppose we have a decoding algorithm D^{ML} which solves the maximum–likelihood decoding problem. Let $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{c}$ is the sent message and suppose there exists $\boldsymbol{c}_1 \in \mathcal{C}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}_1) \leq \mathrm{w}(\boldsymbol{e})$. In this situation the algorithm, gives in return $\boldsymbol{c}_1$, hence we have a failure.

Definition 1.3.12. Let us consider a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, a q-ary symmetric channel qSC and the maximum–likelihood decoder D^{ML}. The failure probability for D^{ML} is defined as

$$\mathbb{P}_{fail}(\mathcal{C}, D^{ML}) \stackrel{\text{def}}{=} \mathbb{P}_{\boldsymbol{e} \sim qSC, \boldsymbol{c} \sim \mathcal{C}}(D^{ML}(\boldsymbol{c} + \boldsymbol{e}) \neq \boldsymbol{c}).$$

The following result, which is the first part of Shannon theorem, states the existence of codes for which the maximum–likelihood decoder's failure probability is small and hence theoretically it could be possible to decode with no ambiguity independently from the size of the error. First we need the notion of *entropy function*.

Definition 1.3.13. The entropy function for a q–ary symmetric channel is defined as

$$H_q(p) \stackrel{\text{def}}{=} p \log_q(q-1) - p \log_q p - (1-p) \log_q(1-p).$$

Theorem 1.3.14 (Shannon Theorem-first part)**.** *Let $0 < p < 1 - 1/q$ and $0 < \epsilon < 1 - 1/q - p$. Then there exists $\delta > 0$ such that for any large enough n, there is a code $\mathcal{C}_n$ of length n and dimension $k = n\lfloor 1 - H_q(p) - \epsilon \rfloor$ for which the maximum likelihood decoder D^{ML} fulfills*

$$\mathbb{P}_{fail}(\mathcal{C}, D^{ML}) \leq q^{-\delta n}.$$

From the proof of this result [Rot06, Theorem 4.17, Corollary 4.18], it is possible to show something even stronger. Indeed, not only there exist codes for which the maximum–likelihood decoder failure probability is really low, but actually almost all codes fulfill this property. Hence we know that for almost all codes, finding the closest codeword $\boldsymbol{c}$ to $\boldsymbol{y}$, even if $\mathrm{d}(\boldsymbol{c},\boldsymbol{y}) > \frac{\mathrm{d}-1}{2}$, corresponds to recovering the original message with high probability. Hence if on the one hand, under the unique decoding bound we are sure we do not have ambiguity on the solution to the nearest–codeword decoding problem, on the other, we know that theoretically it is possible to correct amounts of errors larger than the unique decoding bound at the price of really few failure cases. In this thesis we will then focus, on the one hand, on decoding algorithms which solve the bounded decoding problem for values of t under and beyond the unique decoding bound. On the other, we will also present some methods treating the rare failure cases of the maximum likelihood decoder. To do so, we will focus on decoding algorithms solving the following problem, also called the *list decoding problem*.

Definition 1.3.15 (list decoding problem)**.** Suppose $\mathcal{C} \subseteq \mathbb{F}_q^n$ is a code and $t \in \mathbb{N}$. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find the set

$$\{\boldsymbol{c} \in \mathcal{C} \mid \mathrm{d}(\boldsymbol{y},\boldsymbol{c}) \leq t\}.$$

Definition 1.3.16. All algorithms solving the list decoding problem, will be referred to as *list decoding algorithms*. As for the bounded decoding algorithms, the decoding radius of a list decoding algorithm is the maximum value of t for which the algorithm works.

All along this thesis we will work under the following assumption.

Assumption 1. In the following, given a code $\mathcal{C}$ and a positive integer t, when considering the bounded or the list decoding problem, we always suppose that the *received vector* $\boldsymbol{y} \in \mathbb{F}_q^n$ is of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c} \in \mathcal{C}$ and $\mathrm{w}(\boldsymbol{e}) = t$. Equivalently, we always suppose that the bounded decoding problem has at least one solution.

1.4 Some properties and constructions on codes

We recall in this section some codes properties we will need in the following chapters. First of all, since we consider only linear codes, we can represent them in ways which are more compact than an enumeration of their codewords. Let us consider a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with dimension k.

Notation. Let $\mathcal{M}_{n,m}(\mathbb{F}_q)$ be the space of the matrices with entries in $\mathbb{F}_q$ having n rows and m columns. Any vector $\boldsymbol{a} \in \mathbb{F}_q^n$ will be seen as row vector.

Definition 1.4.1. A *generator matrix* for $\mathcal{C}$ is a matrix $G \in \mathcal{M}_{k,n}(\mathbb{F}_q)$ such that

$$\mathcal{C} = \{\boldsymbol{m}G \mid \boldsymbol{m} \in \mathbb{F}_q^k\}.$$

Definition 1.4.2. A *parity check matrix* of $\mathcal{C}$ is a matrix $H \in \mathcal{M}_{n-k,n}(\mathbb{F}_q)$ such that

$$\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n \mid H\boldsymbol{c}^t = \boldsymbol{0}\}.$$

Given a linear code $\mathcal{C}$, if one disposes of a parity check matrix of $\mathcal{C}$, then it is possible to get some information about the minimum distance of the code.

Proposition 1.4.3. *Given a linear code $\mathcal{C}$, let d and H be respectively its minimum distance and a parity check matrix. If $H^{j_1}, \ldots, H^{j_m}$ are m columns of H with $m < \mathrm{d}$, then these columns are linearly independent. Conversely, if for any choice of m columns $H^{j_1}, \ldots, H^{j_m}$, these columns are are linearly independent, then $\mathrm{d} > m$.*

Definition 1.4.4. Given a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, its *dual* is a code $\mathcal{C}^\perp \subseteq \mathbb{F}_q^n$ defined as

$$\mathcal{C}^\perp \stackrel{\mathrm{def}}{=} \{\boldsymbol{a} \in \mathbb{F}_q^n \mid \langle \boldsymbol{a}, \boldsymbol{c} \rangle = 0 \ \forall \boldsymbol{c} \in \mathcal{C}\},$$

where $\langle \cdot, \cdot \rangle$ is the canonical inner product in $\mathbb{F}_q^n$ defined, for any $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{c} = (c_1, \ldots, c_n)$ in $\mathbb{F}_q^n$, by $\langle \boldsymbol{a}, \boldsymbol{c} \rangle = \sum_{i=1}^n a_i c_i$.

One can prove that $\dim \mathcal{C}^\perp = n - \dim \mathcal{C}$ and that, given a parity check matrix H for $\mathcal{C}$, H is a generator matrix for $\mathcal{C}^\perp$.

1.4.1 Star product of words and codes

The space $\mathbb{F}_q^n$ is a product of n fields and hence has a natural structure of ring. We denote by $*$ the component wise product of vectors

$$(a_1, \dots, a_n) * (b_1, \dots, b_n) \stackrel{\text{def}}{=} (a_1 b_1, \dots, a_n b_n).$$

Given a vector $\boldsymbol{a} \in \mathbb{F}_q^n$, the i–th power $\boldsymbol{a}^i$ of $\boldsymbol{a}$ is defined as $\boldsymbol{a}^i \stackrel{\text{def}}{=} (a_1^i, \dots, a_n^i)$.

Remark 1.4.5. This product should not be confused with the canonical inner product in $\mathbb{F}_q^n$. Note that these two operations are related by the following adjunction property

$$\langle \boldsymbol{a} * \boldsymbol{b}, \boldsymbol{c} \rangle = \langle \boldsymbol{a}, \boldsymbol{b} * \boldsymbol{c} \rangle = \langle \boldsymbol{a} * \boldsymbol{c}, \boldsymbol{b} \rangle. \tag{1.2}$$

In particular $\boldsymbol{a} * \boldsymbol{b} \in \{\boldsymbol{c}\}^\perp \iff \boldsymbol{a} * \boldsymbol{c} \in \{\boldsymbol{b}\}^\perp \iff \boldsymbol{c} * \boldsymbol{b} \in \{\boldsymbol{a}\}^\perp$.

Definition 1.4.6. Given two codes $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$, the star product $\mathcal{A} * \mathcal{B}$ is the code spanned by all the products $\boldsymbol{a} * \boldsymbol{b}$ for $\boldsymbol{a} \in \mathcal{A}$ and $\boldsymbol{b} \in \mathcal{B}$. If $\mathcal{A} = \mathcal{B}$, the product is denoted by $\mathcal{A}^2$. We define

$$\mathcal{A}^0 \stackrel{\text{def}}{=} \text{span}_{\mathbb{F}_q}\{(1, \dots, 1)\}, \qquad \mathcal{A}^i \stackrel{\text{def}}{=} \mathcal{A} * \mathcal{A}^{i-1} \quad \forall i \geq 1.$$

1.4.2 A Kneser–like theorem

We conclude this section with a result which will be useful in the sequel and can be regarded as a star product counterpart of the famous Kneser Theorem in additive combinatorics (see [TV06, Theorem 5.5]). We first have to introduce a notion.

Definition 1.4.7. Given a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, the *stabilizer* of $\mathcal{C}$ is defined as

$$\text{Stab}(\mathcal{C}) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{F}_q^n \mid \boldsymbol{x} * \mathcal{C} \subseteq \mathcal{C}\}.$$

Theorem 1.4.8. *Let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$ be two codes. Then,*

$$\dim(\mathcal{A} * \mathcal{B}) \geq \dim \mathcal{A} + \dim \mathcal{B} - \dim \text{Stab}(\mathcal{A} * \mathcal{B}).$$

Proof. See [MZ15, Theorem 18] or [BL17, Theorem 4.1]. □

For any code $\mathcal{C}$, the stabilizer of $\mathcal{C}$ has dimension at least 1 since it contains the span of the vector $(1, \dots, 1)$. On the other hand, it has been proved in [KS80, Theorem 1.2] that a code $\mathcal{C}$ has a stabilizer of dimension > 1 if and only if it is *degenerated*, i.e. if and only if either it is a direct sum of subcodes with disjoint supports or any generator matrix of $\mathcal{C}$ has a zero column. This leads to the following analog of Cauchy Davenport Theorem ([TV06, Theorem 5.4]).

Corollary 1.4.9. *Let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$ be two codes such that $A * B$ is non degenerated, then*

$$\dim \mathcal{A} * \mathcal{B} \geq \dim \mathcal{A} + \dim \mathcal{B} - 1.$$

1.5 Reed–Solomon codes

The space of polynomials with coefficients in $\mathbb{F}_q$ and degree less than k is denoted by $\mathbb{F}_q[X]_{<k}$. Given an integer $n \geq k$ and a vector $\boldsymbol{x} \in \mathbb{F}_q^n$ whose entries are distinct, the *Reed–Solomon code* of length n and dimension k is the image of the space $\mathbb{F}_q[X]_{<k}$ by the map

$$\text{ev}_{\boldsymbol{x}} : \left\{ \begin{array}{ccc} \mathbb{F}_q[X] & \longrightarrow & \mathbb{F}_q^n \\ f & \longmapsto & (f(x_1), \dots, f(x_n)). \end{array} \right. \tag{1.3}$$

This code is denoted by $\mathbf{RS}_q[\boldsymbol{x}, k]$ or $\mathbf{RS}_q[k]$ when there is no ambiguity on the vector $\boldsymbol{x}$. That is:

$$\mathbf{RS}_q[k] \stackrel{\text{def}}{=} \{(f(x_1), \dots, f(x_n)) \mid f \in \mathbb{F}_q[X]_{<k}\} = \text{ev}_{\boldsymbol{x}}\left(\mathbb{F}_q[X]_{<k}\right).$$

One can actually consider a larger class of codes called *generalized Reed–Solomon codes* and defined as:

$$\mathbf{GRS}_q[\boldsymbol{x}, \boldsymbol{y}, k] \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbb{F}_q[X]_{<k}\},$$

where $\boldsymbol{y} \in (\mathbb{F}_q^\times)^n$. Such a code has length n, dimension k and minimum distance $\mathrm{d} = n - k + 1$. In this thesis, for the sake of simplicity, we focus on the case of Reed–Solomon codes ($\boldsymbol{y} = (1, \ldots, 1)$). Actually, the results of the present thesis for Reed–Solomon codes, extend straightforwardly to generalized Reed–Solomon codes at the cost of slightly more technical proofs. Moreover we will suppose to work with $n = q$, i.e. the so–called *full–support* Reed–Solomon codes. This context is much more comfortable for duality since we can assert that

$$\mathbf{RS}_q[k]^\perp = \mathbf{RS}_q[n - k].$$

In the general case, the above statement remains true by replacing Reed–Solomon codes by generalized Reed–Solomon codes with a specific choice of $\boldsymbol{y}$. Indeed, we have $\mathbf{GRS}_q[\boldsymbol{x}, \boldsymbol{y}, k]^\perp = \mathbf{GRS}_q[\boldsymbol{x}, \boldsymbol{y}', n - k]$, where

$$\boldsymbol{y}' = -\frac{1}{y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} (x_j - x_i)}.$$

See for instance [Rot06, Problem 5.7].

Finally, although the Schur product of generic codes has large dimension, the Schur product of Reed–Solomon codes remains relatively small.

Proposition 1.5.1 (Star product of Reed–Solomon codes)**.** *Let $\boldsymbol{x} \in \mathbb{F}_q^n$ be a vector with distinct entries and k, k' be positive integers. If $k + k' - 1 > n$, then $\mathbf{RS}_q[\boldsymbol{x}, k] * \mathbf{RS}_q[\boldsymbol{x}, k'] = \mathbb{F}_q^n$. Otherwise,*

$$\mathbf{RS}_q[\boldsymbol{x}, k] * \mathbf{RS}_q[\boldsymbol{x}, k'] = \mathbf{RS}_q[\boldsymbol{x}, k + k' - 1].$$

1.6 Basic algorithms for Reed–Solomon codes

It is well-known that several decoding algorithms have been designed for Reed–Solomon codes [WB83, Sud97, GS99, RnN15]. Thanks to their strong algebraic structure, depending on the algorithm, we are able to solve the bounded decoding problem up to half the minimum distance and beyond. We have seen that, for values of t exceeding the unique decoding bound, the uniqueness of the solution is no longer guaranteed. In particular, there could be two codewords at the same distance from the received vector $\boldsymbol{y}$. In this case, a bounded decoding algorithm (see Definition 1.3.8) fails, while a list decoding algorithm (see Definition 1.3.16), returns the list of all possible solutions. Naturally, the intention being to have an algorithm performing in a reasonable time, one wants the length of the list of solutions to be polynomial in the length of the code n. In this scenario, *Johnson's bound* gives a treshold in coding theory, since for any t smaller than this amount, the size of the list is proven to be polynomial in n [Joh62]. This bound is given by the following result.

Theorem 1.6.1 (Johnson's bound)**.** *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a code with minimum distance d and let $\boldsymbol{y} \in \mathbb{F}_q^n$. Given $\delta \stackrel{def}{=} \mathrm{d}/n$, we define the Johnson's bound as*

$$\rho \stackrel{def}{=} \left(1 - \frac{1}{q}\right)\left(1 - \sqrt{1 - \frac{q\delta}{q-1}}\right), \tag{1.4}$$

We have

$$\#\{\boldsymbol{c} \in \mathcal{C} \mid \mathrm{d}(\boldsymbol{c}, \boldsymbol{y}) \leq n\rho\} \leq qn\,\mathrm{d}\,.$$

In particular, if we do an asymptotic analysis of (1.4) for Reed–Solomon codes, that is, we consider a sequence of Reed–Solomon codes with length tending to infinity and with constant ratio $R = k/n$, then Johnson's bound becomes:

$$\rho = 1 - \sqrt{R}. \tag{1.5}$$

In this section, we recall some decoding algorithms for Reed–Solomon codes. The reader can find in Figure 1.1 a representation of this partial state of art,

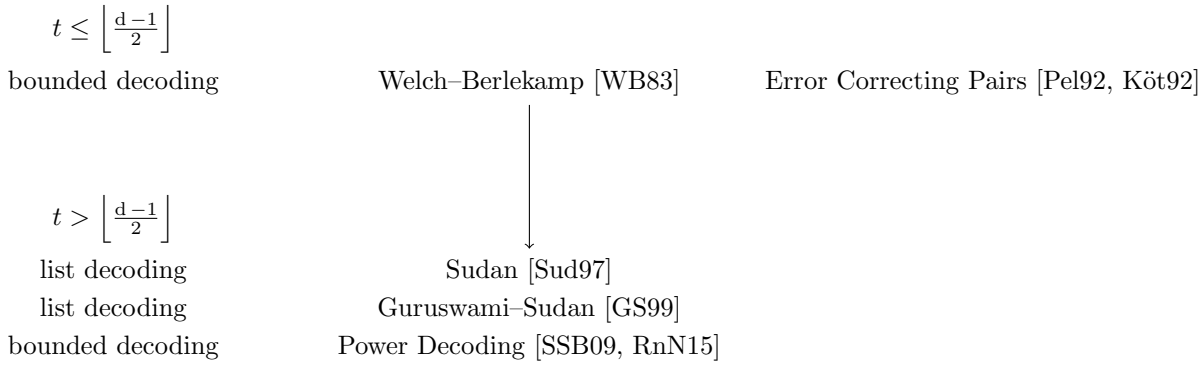

Figure 1.1: Some decoding algorithms for Reed–Solomon codes

with particular distinction between the algorithms which can correct up to the unique decoding bound and the ones which can correct more. Also, the reader can find on the left the kind of the problem the algorithm answers to, which can be the bounded decoding problem (see Definition 1.3.7) or the list decoding problem (see Definition 1.3.15). For unique decoding, that is, for $t \leq \frac{d-1}{2}$, we discuss about Welch–Berlekamp algorithm and the Error Correcting Pairs algorithm (ECP). The Error Correcting Pairs algorithm is not actually a specific algorithm for Reed–Solomon codes, but can be run on a larger class of codes. We will see that, as shown in Figure 1.1, applied to Reed–Solomon codes it can correct up to the unique decoding bound. On the other hand, Welch–Berlekamp algorithm, besides using intuitive and simple objects, can be extended to algorithms correcting amounts of errors beyond the unique decoding bound. Among these extensions we can find the power decoding algorithm, which answers to the bounded decoding problem and, hence, returns only one solution, and Sudan's algorithm, which is a list decoder. Sudan's algorithm can be in turn extended to Guruswami–Sudan algorithm, which attains the Johnson bound. In this section though, we will not describe this algorithm, but we will mainly focus on the power decoding algorithm and Sudan's algorithm. We recall that, whenever we discuss a decoding problem we suppose Assumption 1 to be satisfied, i.e. we suppose that the decoding problem has at least one solution. Hence, we can write

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}, \tag{1.6}$$

for some $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ with $\mathrm{w}(\boldsymbol{e}) = t$. Note that since $\mathcal{C} = \mathbf{RS}_q[k]$, the codeword $\boldsymbol{c}$ can be written as the evaluation of a polynomial $f(x)$ with $\deg(f) < k$. We recall also that the vector $\boldsymbol{e}$ is referred to as the *error vector* and we define

$$I_{\boldsymbol{e}} \stackrel{\text{def}}{=} \mathrm{supp}(\boldsymbol{e}) = \{i \in \{1, \dots, n\} \mid e_i \neq 0\}.$$

Hence, we have $t = \mathrm{w}(\boldsymbol{e}) = |I_{\boldsymbol{e}}|$.

1.6.1 Unique decoding : Welch-Berlekamp algorithm

Welch-Berlekamp algorithm [WB83] boils down to a linear system based on n key equations. The decoding radius is given by a sufficient condition for the correctness of the algorithm. In other words, if t is smaller than the decoding radius of the algorithm and there exists a solution, then the algorithm will find it.

Definition 1.6.2. Given $\boldsymbol{y}, \boldsymbol{e}, f$ and $I_{\boldsymbol{e}}$ as above, we define

- the *error locator polynomial* as $\Lambda(X) \stackrel{\text{def}}{=} \prod_{i \in I_{\boldsymbol{e}}}(X - x_i)$;
- $N(X) \stackrel{\text{def}}{=} \Lambda(X) f(X)$.

Hence, for any $i \in \{1, \dots, n\}$, the polynomials Λ and N verify

$$\Lambda(x_i) y_i = N(x_i). \tag{1.7}$$

The aim of the algorithm is then to solve the following

Key Problem 1. Find a pair of polynomials (λ, ν) such that $\deg(\lambda) \leq t$, $\deg(\nu) \leq t + k - 1$ and

$$\forall i \in \{1, \dots n\}, \quad \lambda(x_i) y_i = \nu(x_i). \tag{S_{WB}}$$

Remark 1.6.3. Actually, the degrees of Λ and N are related by

$$\deg(N) \leq \deg(\Lambda) + k - 1.$$

With this constraint, the problem can be solved using Berlekamp–Massey algorithm, which is just a reformulation of the Euclidean algorithm [HJ99]. However, in this thesis we consider the simplified constraints of Key Problem 1, in order to have linear constraints which make the analysis of the decoding radius easier. By the following lemma it will be clear that making this choice has no consequence on the decoding radius of Welch–Berlekamp algorithm.

The system (S_{WB}) is linear and has n equations in $2t+k+1$ unknowns. We know that the pair $(\Lambda, \Lambda f)$ is in its solutions space. The following result proves that, for certain values of t, actually it is not necessary to find exactly that solution to solve the decoding problem.

Lemma 1.6.4. *Let $t \leq \frac{\mathrm{d}-1}{2}$. If (λ, ν) is a nonzero solution of (S_{WB}), then $\lambda \neq 0$ and $f = \frac{\nu}{\lambda}$.*

For the proof we refer for instance to [JH04, Theorem 4.2.2]. We can finally write the algorithm (see Algorithm 1). Its correctness is entailed by Lemma 1.6.4 whenever $t \leq \frac{\mathrm{d}-1}{2}$, that is, the decoding radius of Welch-Berlekamp algorithm is $t = \lfloor \frac{\mathrm{d}-1}{2} \rfloor$.

Algorithm 1 Welch-Berlekamp Algorithm

Inputs: $\boldsymbol{y} \in \mathbb{F}_q^n$ as in (1.6), $k \leq n, t = \mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}-1}{2}$.
Output: $f \in \mathbb{F}_q[X]_{<k}$ such that $\mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(f), \boldsymbol{y}) = t$.

1: $(\lambda, \nu) \leftarrow$ arbitrary nonzero element in the solutions space of (S_{WB})
2: **return** $f = \frac{\nu}{\lambda}$.

1.6.2 Unique decoding: the Error Correcting Pairs algorithm

The Error Correcting Pairs (ECP) algorithm has been designed by Pellikaan [Pel92] and independently by Kötter [Köt92] in 1992. Its formalism gives an abstract description of a decoding algorithm originally arranged for algebraic geometry codes (which is the *basic algorithm* [JLJ+89, SV90], also described in §3.1) and whose description required notions of algebraic geometry. In their works, Pellikaan and Kötter, simplified the tools needed in the original decoding algorithm and made the algorithm applicable to any linear code benefiting from a certain elementary structure called *error correcting pair* and defined in Definition 1.6.5 below. Given a code $\mathcal{C}$ and a received vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c} \in \mathcal{C}$ and $\mathrm{w}(\boldsymbol{e}) \leq t$ for some positive integer t, the ECP algorithm consists in two steps:

(1) find $J \subseteq \{1, \dots, n\}$ such that $J \supseteq I_{\boldsymbol{e}}$, where $I_{\boldsymbol{e}}$ denotes the support of $\boldsymbol{e}$;

(2) recover the nonzero entries of $\boldsymbol{e}$.

As said before, these steps can be solved if the code has a t-error correcting pair where $t = \mathrm{w}(\boldsymbol{e})$ is small enough.

Definition 1.6.5. Given a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$, a pair $(\mathcal{A}, \mathcal{B})$ of linear codes, with $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$ is called *t-error correcting pair* for $\mathcal{C}$ if

(ECP1) $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^{\perp}$;

(ECP2) $\dim(\mathcal{A}) > t$;

(ECP3) $\mathrm{d}(\mathcal{B}^{\perp}) > t$;

(ECP4) $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$.

Remark 1.6.6. One can observe that, thanks to Remark 1.4.5,

$$\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^{\perp} \iff \mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^{\perp}.$$

Since this notion does not look very intuitive, an example of error correcting pair for Reed–Solomon codes and an interpretation of the various hypotheses above in light of this example are given at the end of this subsection. For now, we want to explain more precisely how the ECP algorithm works. To do so, we will need the following notations.

Definition 1.6.7. Given $J = \{j_1, \dots, j_s\} \subset \{1, \dots, n\}$ and $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_q^n$, we denote by $\boldsymbol{x}_J$ the projection of $\boldsymbol{x}$ on the coordinates in J and by $Z(\boldsymbol{x})$ the complement of the support of $\boldsymbol{x}$ in $\{1, \dots, n\}$. Namely,

$$\boldsymbol{x}_J \stackrel{\text{def}}{=} (x_{j_1}, \dots, x_{j_s}) \qquad \text{and} \qquad Z(\boldsymbol{x}) \stackrel{\text{def}}{=} \{i \in \{1, \dots, n\} \mid x_i = 0\}$$

Moreover, for $A \subseteq \mathbb{F}_q^n$, we define

(*i*) $A_J \stackrel{\text{def}}{=} \{\boldsymbol{a}_J \mid \boldsymbol{a} \in A\} \subseteq \mathbb{F}_q^{|J|}$ (*puncturing*);

(*ii*) $A(J) \stackrel{\text{def}}{=} \{\boldsymbol{a} \in A \mid \boldsymbol{a}_J = \boldsymbol{0}\} \subseteq \mathbb{F}_q^n$ (*shortening*);

(*iii*) $Z(A) \stackrel{\text{def}}{=} \{i \in \{1, \dots, n\} \mid a_i = 0 \ \ \forall \boldsymbol{a} \in A\}$ (*complementary of the support*).

Remark 1.6.8. In (*ii*) we made an abuse of language by using the term *shortening* which classically refers to the operation

$$\{\boldsymbol{a}_{\{1,\dots,n\}\setminus J} \mid \boldsymbol{a}_J = \boldsymbol{0}\}.$$

In comparison to the usual definition we do not remove the prescribed zero positions. We do so because we need to compute some star products (see §1.4.1) and for this sake involved vectors should have the same length.

Remark 1.6.9. Classically in the literature, *puncturing* a code at a subset J means deleting the entries whose index is in J. Here our notation does the opposite operation by keeping only the positions whose indexes are in J and removing all the other ones.

We can now illustrate the two steps of the error correcting pairs algorithm.

First step of the error correcting pair algorithm In Step (1) of the ECP algorithm, we wish to locate the error positions, hence our aim is to find a set J which contains $I_{\boldsymbol{e}}$ and hopefully is not too large. An idea could be, given a code $\mathcal{A}$, to find the space

$$\{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{e} = \boldsymbol{0}\}, \tag{1.8}$$

which is made by all elements of $\mathcal{A}$ whose components indexed by $I_{\boldsymbol{e}}$ are zero, that is, the elements of this space *locate* the error positions. In particular we will refer to the nonzero elements of this space as to *locators* of $\boldsymbol{e}$. It is easy to see that the space in (1.8) is equal to $\mathcal{A}(I_{\boldsymbol{e}})$ and that in order to locate the error support we need to have $\mathcal{A}(I_{\boldsymbol{e}}) \neq \{\boldsymbol{0}\}$, that is, we need to have at least one locator. The following elementary result states that (ECP2) is a sufficient condition to have so.

Proposition 1.6.10. *If* $\dim(\mathcal{A}) > t$, *then* $\mathcal{A}(I_{\boldsymbol{e}}) \neq \{0\}$.

Though, since we do not know $I_{\boldsymbol{e}}$, *a priori* we do not have any information about $\mathcal{A}(I_{\boldsymbol{e}})$. That is why a new vector space is introduced:

$$M_1 \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^{\perp}\}. \tag{1.9}$$

The key of the algorithm is in the following result.

Theorem 1.6.11. *Let* $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, $I_{\boldsymbol{e}} = \mathrm{supp}(\boldsymbol{e})$ *and* M_1 *as above. If* $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^{\perp}$, *then*

(*i*) $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M_1 \subseteq \mathcal{A}$;

(*ii*) *if* $\mathrm{d}(\mathcal{B}^{\perp}) > t$, *then* $\mathcal{A}(I_{\boldsymbol{e}}) = M_1$.

Proof. The proof is deeply explained in [Pel92]. Though, we will report it here, for the sake of self-containedness and to shed light on Chapter 2. First, we prove the inclusions of (i). By the definition of M_1, we have $M_1 \subseteq \mathcal{A}$. Now, let $\boldsymbol{a} \in \mathcal{A}(I_{\boldsymbol{e}})$. For all $\boldsymbol{b} \in \mathcal{B}$, we get

$$\begin{aligned} \langle \boldsymbol{a} * \boldsymbol{y}, \boldsymbol{b} \rangle &= \langle \boldsymbol{a} * \boldsymbol{c}, \boldsymbol{b} \rangle + \langle \boldsymbol{a} * \boldsymbol{e}, \boldsymbol{b} \rangle && (1.10) \\ &= \langle \boldsymbol{a} * \boldsymbol{b}, \boldsymbol{c} \rangle + \langle \boldsymbol{a} * \boldsymbol{e}, \boldsymbol{b} \rangle && (1.11) \\ &= 0. && (1.12) \end{aligned}$$

In (1.10) and (1.11), we used the properties of the star product (see Remark 1.4.5) and the bilinearity of the standard inner product. Finally, (1.12) holds since $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^{\perp}$ and the supports of $\boldsymbol{a}$ and $\boldsymbol{e}$ are complementary. We now prove (ii). Given $\boldsymbol{a} \in M_1$, we have

$$\boldsymbol{a} * \boldsymbol{y} = \boldsymbol{a} * \boldsymbol{c} + \boldsymbol{a} * \boldsymbol{e}$$

Now, since $\boldsymbol{a} \in M_1$ and by the hypothesis $\mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^{\perp}$, we have $\boldsymbol{a} * \boldsymbol{e} = \boldsymbol{a} * \boldsymbol{y} - \boldsymbol{a} * \boldsymbol{c} \in \mathcal{B}^{\perp}$. Though, $\mathrm{w}(\boldsymbol{a} * \boldsymbol{e}) \leq \mathrm{w}(\boldsymbol{e}) \leq t$, while $\mathrm{d}(\mathcal{B}^{\perp}) > t$. Hence $\boldsymbol{a} * \boldsymbol{e} = \boldsymbol{0}$, that is, $\boldsymbol{a} \in \mathcal{A}(I_{\boldsymbol{e}})$. □

Therefore, if the pair $(\mathcal{A}, \mathcal{B})$ fulfills (ECP1-3) in Definition 1.6.5, then $Z(M_1)$ is non trivial and contains $I_{\boldsymbol{e}}$ (see Definition 1.6.7). Therefore, Step (1) of the algorithm consists in computing $J = Z(M_1)$.

Second step of the error correcting pair algorithm Step (2) consists simply in the resolution of a linear system depending on J and the syndrome of $\boldsymbol{y}$. First, some notation is needed.

Notation. Given $H \in \mathcal{M}_{n,m}(\mathbb{F}_q)$ and $J \subseteq \{1, \ldots, m\}$, we denote by H_J the submatrix of H whose columns are those with index $j \in J$.

Suppose we have computed $J \supseteq I_{\boldsymbol{e}}$ in Step (1) of the algorithm. Consider a full rank–parity check matrix H for $\mathcal{C}$. The vector $\boldsymbol{e}_J$ satisfies $H_J \cdot \boldsymbol{e}_J^T = H \cdot \boldsymbol{y}^T$. We want then to recover $\boldsymbol{e}_J$ by solving the linear system

$$H_J \cdot \boldsymbol{u}^T = H \cdot \boldsymbol{y}^T. \tag{1.13}$$

Now, *a priori*, the solution may not be unique. Though, condition (ECP4) in Definition 1.6.5 yields the following result.

Lemma 1.6.12. *If* $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$, $\dim \mathcal{A} > t$ *and* $J = Z(M_1)$, *then* $|J| < \mathrm{d}(\mathcal{C})$.

Proof. By Proposition 1.6.10, there exists $\boldsymbol{a} \in \mathcal{A}(I_{\boldsymbol{e}}) \setminus \{0\}$. Now, since $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$, we get

$$|J| = |Z(M_1)| \leq |Z(\boldsymbol{a})| = n - \mathrm{w}(\boldsymbol{a}) \leq n - \mathrm{d}(\mathcal{A}) < \mathrm{d}(\mathcal{C}).$$

□

Theorem 1.6.13. *Given* $\boldsymbol{y} \in \mathbb{F}_q^n$ *and* $J \subseteq \{1, \ldots, n\}$ *with* $t = |J| < \mathrm{d}(\mathcal{C})$, *then there exists at most one solution for (1.13).*

Proof. Observe that by Proposition 1.4.3, since $|J| < \mathrm{d}(\mathcal{C})$, the columns of the matrix H_J are linearly independent, hence the system in (1.13) has at most one solution. □

This theorem, together with Lemma 1.6.12, entails that if J contains the support of the error, then $\boldsymbol{e}_J$ is the unique solution to system (1.13). Thus, the second step of the algorithm consists in finding $\boldsymbol{e}_J$ by solving system (1.13) and recovering $\boldsymbol{e}$ from $\boldsymbol{e}_J$ imposing $e_i = 0$ for all $i \notin J$.

Remark 1.6.14. The problem in Step (2), once the set J is known, is related to the so-called *erasure decoding problem*. This problem is described as follows: given s coordinates of a codeword $\boldsymbol{c} \in \mathcal{C}$ together with their positions, recover the entire codeword $\boldsymbol{c}$. In particular, given an instance of this problem with $s \geq n - \mathrm{d} + 1$, it is possible to fill the erasures with random elements of $\mathbb{F}_q$ and, hence, to recover the entire $\boldsymbol{c}$ with the method described for Step (2).

The entire algorithm is described in Algorithm 2.
The correctness of the algorithm is proved in [Pel92]. It is straightforward to see that the algorithm returns a unique solution and that a sufficient condition for the algorithm to correct any possible pattern of t errors, is the existence of an error correcting pair with parameter t. This consideration, leads to the following result.

Algorithm 2 Error correcting pairs algorithm

Inputs: $\mathcal{C}$ linear code, $\boldsymbol{y} \in \mathbb{F}_q^n$ as in (1.6), $t = \mathrm{w}(\boldsymbol{e})$, $(\mathcal{A}, \mathcal{B})$ a t-error correcting pair for $\mathcal{C}$.
Output: $\boldsymbol{c} \in \mathcal{C}$ such that $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ for some $\boldsymbol{e} \in \mathbb{F}_q^n$ with $\mathrm{w}(\boldsymbol{e}) \leq t$.

1: compute $M_1 = \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^\perp\}$ (linear system);
2: $J \leftarrow Z(M_1)$;
3: $\boldsymbol{e}_J \leftarrow$ solution of (1.13);
4: recover $\boldsymbol{e}$ from $\boldsymbol{e}_J$;
5: **return** $\boldsymbol{c} = \boldsymbol{y} - \boldsymbol{e}$;

Corollary 1.6.15 ([Pel92, Corollary 2.15])**.** *If a linear code $\mathcal{C}$ has a t-error correcting pair, then*

$$t \leq \left\lfloor \frac{\mathrm{d}(\mathcal{C}) - 1}{2} \right\rfloor.$$

Error correcting pairs for Reed–Solomon codes

For an arbitrary code, there is no reason that an error correcting pair exists. Indeed, the existence of an ECP for a given code relies on the existence of a pair $(\mathcal{A}, \mathcal{B})$ of codes, both having a sufficiently large dimension and satisfying $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp$, which is actually a very restrictive condition. Among the codes for which an ECP exists, there are Reed–Solomon codes. Indeed, given $\mathcal{C} = \mathbf{RS}_q[k]$, consider the pair $(\mathcal{A}, \mathcal{B})$:

$$\mathcal{A} = \mathbf{RS}_q[t+1], \quad \mathcal{B}^\perp = \mathbf{RS}_q[t+k]. \tag{1.14}$$

Recall that thanks to Proposition 1.5.1, we have $\mathcal{A} * \mathcal{C} = \mathbf{RS}_q[t+k]$.

Lemma 1.6.16. *Given $\mathcal{B}$ as above, we have*

$$\mathrm{d}(\mathcal{B}^\perp) > t \iff t \leq \frac{\mathrm{d}(\mathcal{C}) - 1}{2}. \tag{1.15}$$

Proof. We have $\mathrm{d}(\mathcal{B}^\perp) = n - t - k + 1 > t \iff t \leq \frac{\mathrm{d}(\mathcal{C})-1}{2}$. □

Proposition 1.6.17. *The pair $(\mathcal{A}, \mathcal{B})$ of (1.14) is a t-error correcting pair for $\mathcal{C}$ if and only if*

$$t \leq \frac{\mathrm{d}(\mathcal{C}) - 1}{2}. \tag{1.16}$$

Proof. We have to prove that (1.16) is a necessary and sufficient condition for (ECP1–4) in Definition 1.6.5 to hold. First of all, by Lemma 1.6.16 we have (ECP3). Moreover $\dim \mathcal{A} = t + 1 > t$ by definition of $\mathcal{A}$ and this gives (ECP2). By Proposition 1.5.1, as seen above, the codes $\mathcal{A}, \mathcal{B}, \mathcal{C}$ verify $\mathcal{A} * \mathcal{C} = \mathcal{B}^\perp$, then by Remark 1.4.5 we obtain $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp$. Finally, it is easy to see that $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n \iff t < \mathrm{d}(\mathcal{C})$. Hence if $t \leq \left\lfloor \frac{\mathrm{d}(\mathcal{C})-1}{2} \right\rfloor$, (ECP1–4) hold and conversely. □

Remark 1.6.18. In Chapter 2, we are going to work with structures which are slightly different from error correcting pairs, that is, we will still require (ECP1, 2) and (ECP4) to hold, together with other conditions. Note that, given $\mathcal{A}$ and $\mathcal{B}$ as in (1.14) Conditions (ECP1, 2) and (ECP4) hold if and only if $t < \mathrm{d}(\mathcal{C})$.

ECP and Welch–Berlekamp key equations The example of Reed–Solomon also permits to understand the rationale behind EPC's in light of Welch–Berlekamp algorithm. Indeed, we now show that the choice of M_1 we made in the ECP algorithm, if one looks at the key equations of Welch–Berlekamp algorithm, appears to be really natural. Let us consider $\mathcal{C} = \mathbf{RS}_q[k]$ and the pair $(\mathcal{A}, \mathcal{B})$ we defined in (1.14). We can write (1.7) for any $i \in \{1, \ldots, n\}$ using the star product in this way

$$(\Lambda(x_1), \ldots, \Lambda(x_n)) * \boldsymbol{y} = (N(x_1), \ldots, N(x_n)).$$

From that, we can deduce

- $(N(x_1), \ldots, N(x_n)) \in \mathbf{RS}_q[t+k] = \mathcal{B}^\perp$;

- $(\Lambda(x_1), \ldots, \Lambda(x_n)) \in \mathbf{RS}_q[t+1](I_{\boldsymbol{e}}) = \mathcal{A}(I_{\boldsymbol{e}})$;
- Moreover $(\Lambda(x_1), \ldots, \Lambda(x_n)) \in \underbrace{\{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^\perp\}}_{M_1}$.

In other words, the vector $(\Lambda(x_1), \ldots, \Lambda(x_n))$ belongs to the space $\mathcal{A}(I_{\boldsymbol{e}})$ we are looking for in the ECP algorithm. Moreover it fulfills a property which characterizes a space $M_1 \supseteq \mathcal{A}(I_{\boldsymbol{e}})$, that is exactly the space we define in the ECP algorithm and that turns to be equal to $\mathcal{A}(I_{\boldsymbol{e}})$ under certain conditions.

The roles of the properties (ECP1–4) We give here a brief resume of the role of the properties defining an error correcting pairs. Morally, one looks for a code $\mathcal{A}$ which is large enough and, at the same time, such that the product $\mathcal{A} * \mathcal{C}$ is not too large:

- we first want to compute a nonzero locator for $\boldsymbol{e}$ in $\mathcal{A}$. The condition $\dim \mathcal{A} > t$ guarantees that such a nonzero locator exists;
- we introduce a space M_1. If $\mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^\perp$, then M_1 contains the space of locators;
- if, moreover, $\mathrm{d}(\mathcal{B}^\perp) > t$, then M_1 is equal to the space of locators and we can compute $J \supseteq I_{\boldsymbol{e}}$;
- if $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$ and $\dim(\mathcal{A}) > t$, then J is not too large and we can find the values of $\boldsymbol{e}$ by solving a linear system.

It is actually possible to simplify the structure of error correcting pair:

Definition 1.6.19 (Error corrector)**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^n$. A linear code $\mathcal{A}$ is an *t-error corrector* for $\mathcal{C}$ if

- $\dim(\mathcal{A}) > t$;
- $\mathrm{d}(\mathcal{A} * \mathcal{C}) > t$;
- $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$.

Observe that, at the price of some small changes, the error correcting pairs algorithm can be run on this structure as well. In particular one can compute, instead of M_1, the space $\tilde{M}_1$ defined as follows

$$\tilde{M}_1 \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{A} * \mathcal{C}\}.$$

However, in order to provide an error corrector $\mathcal{A}$, one should be able to compute or bound the minimum distance of the code $\mathcal{A} * \mathcal{C}$, which is not always easy.

Remark 1.6.20. The example of ECP given above for Reed–Solomon codes is actually an example of error corrector, as $\mathcal{A} * \mathcal{C} = \mathcal{B}^\perp$.

1.6.3 Beyond the unique decoding bound: Sudan's algorithm

Sudan's algorithm is a list decoding algorithm, that is, given a positive integer t and a vector $\boldsymbol{y} \in \mathbb{F}_q^n$, it returns the list of any possible codeword $\boldsymbol{c}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq t$. It is then a "worst case" algorithm, as it manages also the cases where several codewords have the same distance from $\boldsymbol{y}$ or when the original message is not the closest one to $\boldsymbol{y}$. Also, it provides an upper bound for the size of the list of solutions. Finally, it can be seen as an extension of Welch-Berlekamp algorithm. In this section we present briefly all these properties, but the interested reader can find more details in [Sud97]. First, let us write the key equations of Welch-Berlekamp algorithm in an equivalent form. We define

$$Q(X, Y) \stackrel{\text{def}}{=} \Lambda(X)Y - N(X),$$

where Λ and N are the polynomials defined in Welch-Berlekamp algorithm and satisfying (1.7). The polynomial Q fulfills $Q(x_i, y_i) = 0$ for all $i \in \{1, \ldots, n\}$. Hence, if $t = \left\lfloor \frac{\mathrm{d}-1}{2} \right\rfloor$, the problem faced in Welch-Berlekamp algorithm is equivalent to the following *interpolation* problem.

Key Problem 2. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find $Q(X, Y) = Q_0(X) + Q_1(X)Y$ such that

(*i*) $Q(x_i, y_i) = 0$ for all $i = 1, \ldots, n$;

(*ii*) $\deg(Q_j) < n - t - j(k-1)$ for $j = 0, 1$.

Remark 1.6.21. Since for $t \leq \left\lfloor \frac{\mathrm{d}-1}{2} \right\rfloor$ Key Problems 1 and 2 are equivalent, the solution spaces of the corresponding linear systems are equal.

If this problem admits a nonzero solution $Q(X, Y)$, then by Remark 1.6.21, we can easily recover f as

$$f = -\frac{Q_0}{Q_1},$$

that is, as the root of the polynomial Q with respect to Y. Now, Sudan's idea consists in defining a higher degree polynomial $Q(X, Y) = Q_0(X) + Q_1(X)Y + \cdots + Q_\ell(X)Y^\ell$ and in generalizing Key Problem 2 as follows:

Key Problem 3. Given $\boldsymbol{y} \in \mathbb{F}_q^n$, find $Q(X, Y) = Q_0(X) + \cdots + Q_\ell(X)Y^\ell$ such that

(i') $Q(x_i, y_i) = 0$ for all $i = 1, \ldots, n$;

(ii') $\deg(Q_j) < n - t - j(k-1)$ for $j = 0, \ldots, \ell$,

where t is not necessarily smaller than the unique decoding bound.

As for Key Problem 1, this problem reduces to solve a linear system S_{Sud} whose unknowns are the coefficients of the polynomial Q and the equations are given by (i') and (ii'). Once we find such a polynomial $Q(X, Y) \neq 0$, we would like to recover our solutions, but this time we no longer have a link between the shape of Q and the error locator polynomial Λ. Though the following result shows that all solutions show up among the roots of Q with respect to the variable Y.

Lemma 1.6.22. *Let Q fulfill (i') and (ii') for some t. If $f(X) \in \mathbb{F}_q[X]_{<k}$ is such that $\mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(f), \boldsymbol{y}) \leq t$, then $(Y - f(X))|Q(X, Y)$.*

For the proof, see [Sud97]. The algorithm consists then in two main parts (see Algorithm 3): an interpolation problem and a root finding problem. In the root finding step it is not necessary to compute the complete factorization of Q, but Gao-Shokrollahi algorithm [GS00] can be used instead and that can be done in $O(n^2\ell^3)$ operations in $\mathbb{F}_q$. The main cost will be then that of the interpolation step consisting in solving the linear system S_{Sud}, that is $O(n^\omega \ell)$ operations using Gaussian elimination. Note that the interpolation step can be solved efficiently performing fast linear algebra (see Kötter's algorithm [McE03b] or, for further improvements, [Nie14, CH15]).

Remark 1.6.23. A part from giving an algorithm finding all possible solutions, Lemma 1.6.22 permits to bound the length of the list of solutions. That is, if we apply the algorithm with a certain ℓ, we know that the length of this list will be at most ℓ.

Algorithm 3 Sudan's Algorithm

Inputs: $\boldsymbol{y} \in \mathbb{F}_q^n$, $k \leq n$, t
Output: $[g \in \mathbb{F}_q[X]_{<k} \mid \mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(g), \boldsymbol{y}) \leq t]$

1: $Q \leftarrow$ a nonzero solution of S_{Sud}
2: $L \leftarrow [g(X)$ such that $(Y - g(X))|Q(X, Y)]$
3: **return** $[g(X) \in L$ such that $\deg(g) < k$ and $\mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(g), \boldsymbol{y}) \leq t]$

Decoding Radius

First, let us observe that Key Problem 3 gives a constraint on t. Indeed, from (ii'), we need to have

$$t < n - \ell(k-1). \tag{1.17}$$

Moreover, Lemma 1.6.22 implies that a sufficient condition for the algorithm to work, is the existence of a nonzero solution Q for the system S_{Sud}. Hence a sufficient condition for that, is to have more unknowns than equations. By this condition we find

$$t \leq \frac{2n\ell - k\ell(\ell+1) + \ell(\ell+1) - 2}{2(\ell+1)}. \tag{1.18}$$

Normally the bound in (1.18) is less than the bound in (1.17), hence it is referred to as the decoding radius of Sudan's algorithm. One can notice that by (1.17), $\ell \leq \frac{n-t}{k-1}$, hence if we consider a sequence of Reed–Solomon codes with constant ratio $R = k/n$, for n which tends to infinity, we get

$$\lim_{n\to\infty} \frac{t}{n} \lessapprox 1 - \sqrt{2R},$$

which is still far from Johnson bound 1.5. Unlike Sudan's algorithm, Guruswami–Sudan algorithm finally gets to attain this bound by forcing Sudan's polynomial to vanish at the points (x_i, y_i) with a larger multiplicity (for more details, see [GS99]).

1.6.4 Beyond the unique decoding bound: the power decoding algorithm

This bounded decoding algorithm was at first introduced by Sidorenko, Schmidt and Bossert in [SSB09]. The terminology *power decoding*, though, was introduced only later by Rosenkilde in its revisited version of the algorithm [RnN15]. The power decoding is inspired from a decoding algorithm for interleaved Reed–Solomon codes. Given the vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ we want to correct, it consists in considering several powers (with respect to the star product) of $\boldsymbol{y}$, that is $\boldsymbol{y}, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^\ell$, in order to have more relations to work on (see § 1.4.1 for the definition of $\boldsymbol{y}^i$). We can define

$$\boldsymbol{e}^{(i)} = \boldsymbol{y}^i - \boldsymbol{c}^i \quad \forall i = 1, \ldots, \ell,$$

where we get clearly $\boldsymbol{e}^{(1)} = \boldsymbol{e}$. One can see then $\boldsymbol{y}^i$ as a perturbation of a codeword

$$\boldsymbol{c}^i \in \mathcal{C}^i = \mathbf{RS}_q[ik - i + 1]$$

by the error vector $\boldsymbol{e}^{(i)}$. Hence for any $i \in \{1, \ldots, \ell\}$, $\boldsymbol{y}^i$ is an instance of a bounded decoding problem with respect to $\mathcal{C}^i$. In addition, we have the following elementary result which is the key of power decoding.

Proposition 1.6.24. *We have* $I_{\boldsymbol{e}^{(i)}} \subseteq I_{\boldsymbol{e}^{(1)}}$.

It asserts that on all the $\boldsymbol{y}^i$'s the errors are localized at the same positions. More precisely, error positions on $\boldsymbol{y}^i$ are error positions on $\boldsymbol{y}$. Hence, we are in the error model of interleaved codes: the equations

$$\boldsymbol{y}^i = \boldsymbol{c}^i + \boldsymbol{e}^{(i)} \quad i = 1, \ldots, \ell$$

can be regarded as a decoding problem for the interleaving of ℓ codewords with errors at most at t positions. Therefore, errors can be decoded simultaneously using the same error locator polynomial. We consider then the error locator polynomial $\Lambda(X) = \prod_{i \in I_{\boldsymbol{e}}} (X - x_i)$ as for Welch-Berlekamp algorithm and the polynomials

$$N_i \stackrel{\text{def}}{=} \Lambda f^i \quad \forall i = 1, \ldots \ell.$$

Thanks to Proposition 1.6.24, it is possible to write the key equations

$$\begin{cases} \Lambda(x_i) y_i = N_1(x_i) & \forall i \in \{1, \ldots, n\} \\ \quad \ldots \quad \ldots & \quad \ldots \\ \Lambda(x_i) y_i^\ell = N_\ell(x_i) & \forall i \in \{1, \ldots, n\} \end{cases} \tag{1.19}$$

Consequently, the power decoding algorithm consists in solving the following problem.

Key Problem 4. Given $\boldsymbol{y} \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$, find $(\lambda, \nu_1, \ldots, \nu_\ell)$ which fulfills

$$\begin{cases} \lambda(x_i)y_i = \nu_1(x_i), & \forall i \in \{1, \ldots, n\} \\ \quad \ldots \quad \ldots & \quad \ldots \\ \lambda(x_i)y_i^\ell = \nu_\ell(x_i), & \forall i \in \{1, \ldots, n\}. \end{cases} \tag{S_{Po}}$$

with $\deg(\lambda) \leq t$ and $\deg(\nu_j) \leq t + j(k-1)$ for $j \in \{1, \ldots, \ell\}$.

Remark 1.6.25. Key Problem 4 is slightly different from the problem faced in the original paper describing Power Decoding ([SSB10]). We used a *key equation* formulation of the problem instead of the *syndrome* one. The two formulations are equivalent if proper bounds on polynomials' degrees are taken. In particular, one should look for $(\lambda, \nu_1, \ldots, \nu_\ell)$ such that $\deg(\nu_j) \leq \deg(\lambda) + j(k-1)$ for all $j \in \{1, \ldots, \ell\}$ (see [RnN15]). However, similarly as for Welch-Berlekmap algorithm, we consider ℓ weaker constraints which allow to reduce the problem to a linear system to solve. The price is that our Key Problem could get more failure cases than problem in [SSB10]. However, we observed experimentally that these cases are really rare.

The vector $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ is a solution of the linear system (S_{Po}). Though, at the moment there is no guaranteed method to recover it among all the solutions. We only know that, if g is a polynomial such that $\deg(g) < k$ and $\mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(g), \boldsymbol{y}) \leq t$, then there exists an error locator polynomial Γ of the error with respect to g, such that the vector

$$(\Gamma, \Gamma g, \ldots, \Gamma g^\ell)$$

is solution of (S_{Po}). Among all solutions like that, we want to pick the one that gives the closest codeword, that is the one such that $\deg(\Gamma)$ is minimal (see pt.1 in Algorithm 4).

Algorithm 4 Power decoding algorithm with $\ell \geq 2$

Inputs: $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_q^n$ with $\boldsymbol{c} \in \mathcal{C}$, $t = \mathrm{w}(\boldsymbol{e}), k \leq n$
Output: $g \in \mathbb{F}_q[X]_{<k}$ such that $\mathrm{ev}_{\boldsymbol{x}}(g) = \boldsymbol{c}$ or "?".

1: $(\lambda, \nu_1, \ldots, \nu_\ell) \leftarrow$ a nonzero solution of (S_{Po}) with λ of smallest possible degree.
2: **if** $\left(\lambda | \nu_i \text{ **and** } \left(\frac{\nu_1}{\lambda}\right)^i = \frac{\nu_i}{\lambda} \quad \forall i = 1, \ldots, \ell\right)$ **then** $g \stackrel{\text{def}}{=} \frac{\nu_1}{\lambda}$
3: **if** $\mathrm{d}(\mathrm{ev}_{\boldsymbol{x}}(g), \boldsymbol{y}) = t$ **and** $\deg(g) < k$, **then**
4: **return** g.
5: **return** "?"

Remark 1.6.26. The ℓn equations we obtain in Key Problem 4, consist in the key equations for all the $\boldsymbol{y}^i$'s, that is ℓ simultaneous decoding problems. The important aspect is that these two decoding problems share the error locator polynomial Λ. Hence, by adding $(\ell-1)n$ relations, we only add $\sum_{i=2}^{\ell} \deg(N_i) + \ell - 1$ unknowns instead of $\sum_{i=2}^{\ell} \deg(N_i) + (\ell-1)t + 2(\ell-1)$.

Remark 1.6.27. To compute the decoding radius of the Power Decoding algorithm we look for a condition on the size of the system (S_{Po}). Note that the algorithm gives one solution or none, hence there cannot be a sufficient condition for the correctness of the algorithm as soon as $t > \frac{\mathrm{d}-1}{2}$. For this reason, we look for a **necessary condition** for the system (S_{Po}) to have a solution space of dimension 1.

Remark 1.6.28. We recall that we assume to know the amount of errors t. Now, suppose the closest codeword to $\boldsymbol{y}$ is not $\boldsymbol{c}$, but a certain $\tilde{\boldsymbol{c}} \neq \boldsymbol{c}$, i.e. $\mathrm{d}(\tilde{\boldsymbol{c}}, \boldsymbol{y}) < t$. Then, if the algorithm recovers a polynomial g with $\deg g < k$, because of the method used to choose a solution of the system in Algorithm 4, one has $\mathrm{ev}_{\boldsymbol{x}}(g) = \tilde{\boldsymbol{c}}$. This case is considered as a failure case as, although the algorithm recovers the closest codeword to $\boldsymbol{y}$, it does not find the original codeword $\boldsymbol{c}$.

Decoding radius

Under Assumption 1, we know that $(\Lambda, \Lambda f, \ldots, \Lambda f^\ell)$ is a solution for (S_{Po}) and hence, the algorithm returns $\boldsymbol{c}$ if and only if the solution space of (S_{Po}) has dimension one.
Let us define the polynomial

$$\pi(X) \stackrel{\text{def}}{=} \prod_{i=1}^{n} (X - x_i)$$

and consider the bounds in Key Problem 4 on the degrees of the ν_i's. If $t+\ell(k-1) \geq n$, then $(0, 0, \ldots, 0, \pi)$ would be a solution to (S_{Po}) and the dimension of the solution space would be at least 2. Hence we want

$$t + \ell(k-1) < n \tag{1.20}$$

However, bound (1.20) is actually much larger than the decoding radius. We look then for a stricter bound on t. Another necessary condition to have a solution space of dimension one for (S_{Po}), is:

$$\#unknowns \leq \#equations + 1,$$

which gives the decoding radius

$$t \leq \frac{2n\ell - k\ell(\ell+1) + \ell(\ell-1)}{2(\ell+1)}. \tag{1.21}$$

Remark 1.6.29. This analysis consists in giving a necessary condition for the algorithm to work. We recall that the algorithm works whenever it returns exactly the original codeword $\boldsymbol{c}$ and not necessarily the closest codeword to $\boldsymbol{y}$ (see Remark 1.6.28).

1.7 Algebraic geometry codes

Algebraic geometry codes can be seen as a generalization of Reed–Solomon codes. Indeed, if a Reed–Solomon code is the space of the evaluations of polynomials of bounded degree on a set of elements of $\mathbb{F}_q$, an algebraic geometry code is composed by the evaluations of rational functions with bounded zeros and poles, on a set of points of a projective curve. In order to understand the notion of algebraic geometry code and to be able to work with it, we will need a small algebraic geometry introduction.

1.7.1 Algebraic geometry prerequisites

In this subsection we will give some basic elements and notions of algebraic geometry. We will briefly present the results without writing the proofs. The reader will be able to find further details on algebraic geometry and function fields in [TVN07, Sti09, Ful69]. In this thesis we will work exclusively with finite fields, hence in this subsection we will just give the following introductory notions for $\mathbb{F}_q$ with q a prime power and its algebraic closure $\overline{\mathbb{F}}_q$. The first notion we need, is the one of projective space.

Definition 1.7.1. Given a field $\mathbb{F}_q$, for any $n \geq 1$ the *n-dimensional projective space* is defined as

$$\mathbb{P}^n(\mathbb{F}_q) \stackrel{\text{def}}{=} (\mathbb{F}_q^{n+1} \setminus \{\mathbf{0}\})/_{\sim}$$

where, given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^{n+1}$ we have $\boldsymbol{a} \sim \boldsymbol{b} \iff \boldsymbol{a} = \lambda \boldsymbol{b}$ for some $\lambda \in \mathbb{F}_q$. The elements of $\mathbb{P}^n(\mathbb{F}_q)$ are called *points* of $\mathbb{P}^n(\mathbb{F}_q)$. For any $\boldsymbol{a} \in \mathbb{F}_q^{n+1}$, we denote by $[\boldsymbol{a}]$ the point associated to $\boldsymbol{a}$ in $\mathbb{P}^n(\mathbb{F}_q)$.

Definition 1.7.2. An homogeneous polynomial $p \in \mathbb{F}_q[X_0, \ldots, X_n]$ is a polynomial such that all monomials of p have the same degree.

One can note that, given an homogeneous polynomial $f \in \mathbb{F}_q[X_0, \ldots, X_n]$, the evaluation of f on a point of $\mathbb{P}^n(\overline{\mathbb{F}}_q)$ is not well defined, as this evaluation changes with respect to the chosen representative of the class. Though, we have that if $f(\boldsymbol{a}) = 0$ for a certain $\boldsymbol{a} \in \overline{\mathbb{F}}_q^{n+1}$, then $f(\lambda \boldsymbol{a}) = 0$ for any $\lambda \in \overline{\mathbb{F}}_q$. Hence the following notion is consistent.

Definition 1.7.3. Suppose $f_1, \ldots, f_h$ are homogeneous polynomials in $\mathbb{F}_q[X_0, \ldots, X_n]$. We can define the *projective algebraic set* associated to $\{f_1, \ldots, f_h\}$ as

$$V(\{f_1, \ldots, f_h\}) \stackrel{\text{def}}{=} \{P \in \mathbb{P}^n(\overline{\mathbb{F}}_q) \mid f_i(P) = 0 \ \ \forall i = 1, \ldots, h\}.$$

A projective algebraic set $\mathcal{V} \subseteq \mathbb{P}^n(\overline{\mathbb{F}}_q)$ is said to be *irreducible* in $\mathbb{F}_q$ if it is not empty and if, for any pair $\mathcal{V}_1$ and $\mathcal{V}_2$ of projective algebraic sets in $\mathbb{F}_q$ such that

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \text{ and } \mathcal{V}_1, \mathcal{V}_2 \neq \emptyset,$$

we have $\mathcal{V}_1 \subseteq \mathcal{V}_2$ or $\mathcal{V}_2 \subseteq \mathcal{V}_1$. This notion clearly depends on the field we are considering, as for instance the projective algebraic set could be reducible in an extension of $\mathbb{F}_q$. If this set is irreducible in $\overline{\mathbb{F}}_q$, it is called *absolutely irreducible*.

Definition 1.7.4. A *projective variety* over $\mathbb{F}_q$ is a projective algebraic set which is irreducible over $\mathbb{F}_q$.

Definition 1.7.5. Given a subset $A \subseteq \mathbb{P}^n(\overline{\mathbb{F}}_q)$, the ideal associated to A in $\mathbb{F}_q$ is

$$\mathcal{I}(A) \stackrel{\text{def}}{=} \text{span}_{\mathbb{F}_q}\{f \in \mathbb{F}_q[X_0, \ldots, X_n] \mid f \text{ homogeneous } \wedge f(P) = 0 \ \ \forall P \in A\}.$$

Remark 1.7.6. Note that among the elements of $\mathcal{I}(A)$ we have also polynomials which are not homogeneous. For any of these polynomials, though, the property $f(P) = 0$ still makes sense, as it can be proven that for any representative $\boldsymbol{a} \in \overline{\mathbb{F}}_q^{n+1}$ of P, we have $f(\boldsymbol{a}) = 0$.

Proposition 1.7.7. *Let $\mathcal{V} \subseteq \mathbb{P}^n(\overline{\mathbb{F}}_q)$ be a projective algebraic set. Then $\mathcal{V}$ is irreducible in $\mathbb{F}_q$ if and only if $\mathcal{I}(\mathcal{V})$ is prime.*

Example 1.7.8. Let us consider the polynomial $F(X, Y, Z) = X^3 - Y^3 - Z^3 \in \mathbb{F}_q[X, Y, Z]$ and the projective algebraic set $\mathcal{X} = V(F)$. We want to prove that $\mathcal{X}$ is a variety in $\mathbb{F}_q$ and in all its algebraic extensions. We recall that, given a ring A and an ideal $I \subseteq A$, the radical of I is defined as

$$\sqrt{I} \stackrel{\text{def}}{=} \{a \in A \mid a^i \in I \text{ for some } i \in \mathbb{N}\}.$$

Since $\overline{\mathbb{F}}_q$ is algebrically closed and F is irreducible (for instance by Eisenstein criterion), if we compute the ideal associated to $\mathcal{X}$ in $\overline{\mathbb{F}}_q$, we have by the Nullstellensatz theorem

$$\mathcal{I}(V(F)) = \sqrt{(F)} = (F),$$

which is prime. Hence by Proposition 1.7.7, $\mathcal{X}$ is absolutely irreducible, that is, $\mathcal{X}$ is a variety in $\overline{\mathbb{F}}_q$.

It is clear that for any $f \in \mathbb{F}_q[X_0, \ldots, X_n]$ and for any element $\boldsymbol{a} \in \overline{\mathbb{F}}_q^{n+1}$ such that $[\boldsymbol{a}]$ belongs to a projective variety $\mathcal{V}$, we have

$$f(\boldsymbol{a}) = f(\boldsymbol{a}) + g(\boldsymbol{a}) \quad \forall g \in \mathcal{I}(\mathcal{V}).$$

Hence we can define the following object.

Definition 1.7.9. Given a projective variety $\mathcal{V}$, its *coordinate ring* is $\mathbb{F}_q[\mathcal{V}] \stackrel{\text{def}}{=} \mathbb{F}_q[X_0, \ldots, X_n]/\mathcal{I}(\mathcal{V})$.

Since $\mathcal{V}$ is irreducible, by Proposition 1.7.7, $\mathcal{I}(\mathcal{V})$ is a prime ideal and the coordinate ring of $\mathcal{V}$ is a integral domain. It is then possible to consider its quotient field $\text{Frac}(\mathbb{F}_q[\mathcal{V}])$. Note that, given $\frac{f_1}{h_1}$ and $\frac{f_2}{h_2}$ in $\text{Frac}(\mathbb{F}_q[\mathcal{V}])$, we have $\frac{f_1}{h_1} = \frac{f_2}{h_2}$ if and only if $f_1 h_2 - f_2 h_1 \in \mathcal{I}(\mathcal{V})$. Now again, since we want to evaluate at points of $\mathbb{P}^n$ and not at elements of $\mathbb{F}^{n+1}$, we consider the following subfield of $\text{Frac}(\mathbb{F}_q[\mathcal{V}])$.

Definition 1.7.10. The function field of a projective variety $\mathcal{V}$ is defined as

$$\mathbb{F}_q(\mathcal{V}) \stackrel{\text{def}}{=} \left\{ \frac{F}{H} \in \text{Frac}(\mathbb{F}_q[\mathcal{V}]) \mid F, H \text{ homogeneous polynomials with } \deg F = \deg H \right\}.$$

Remark 1.7.11. It is easy to verify that $\mathbb{F}_q \subseteq \mathbb{F}_q(\mathcal{V})$. Moreover the evaluation of the elements of $\mathbb{F}_q(\mathcal{V})$ at points of $\mathbb{P}^n(\overline{\mathbb{F}}_q)$ is well defined.

Definition 1.7.12. The *dimension* of a projective variety $\mathcal{V} \subseteq \mathbb{P}^n(\overline{\mathbb{F}}_q)$, is the transcendence degree of $\mathbb{F}_q(\mathcal{V})$ over $\mathbb{F}_q$.

In this thesis we will work only on specific projective varieties, that is, on projective curves.

Definition 1.7.13. A *projective curve* $\mathcal{X}$ is a projective variety of dimension one. A projective curve $\mathcal{X}$ is said to be *plane* if $\mathcal{X} \subseteq \mathbb{P}^2(\overline{\mathbb{F}}_q)$.

Example 1.7.14. We show that the variety $\mathcal{X}$ presented in Example 1.7.8 is a projective curve. To compute the dimension of $\mathcal{X}$, we consider its function field $\mathbb{F}_q(\mathcal{X})$ and we look for the greatest number of algebrically independent elements over $\mathbb{F}_q$. We show that for any pair of elements $f, g \in \mathbb{F}_q(\mathcal{X})$ there exists a polynomial $p \in \mathbb{F}_q[X_1, X_2]$ such that $p(f, g) = 0$ in $\mathbb{F}_q(\mathcal{X})$:

- $\frac{X}{Z}$ and $\frac{X}{Y}$ are algebrically dependent by the polynomial $p(X_1, X_2) = X_1^3 + X_2^3 - X_1^3 X_2^3$;
- $\frac{X}{Z}$ and $\frac{Z}{Y}$ are algebrically dependent by the polynomial $p(X_1, X_2) = X_2^3 - X_1^3 X_2^3 + 1$;
- $\frac{X}{Z}$ and $\frac{Z}{X}$ are algebrically dependent by the polynomial $p(X_1, X_2) = X_1 X_2 - 1$;
- $\frac{X}{Z}$ and $\frac{Y}{Z}$ are algebrically dependent by the polynomial $p(X_1, X_2) = X_1^3 - X_2^3 - 1$.

More generally, one can prove that all other pairs of elements of $\mathbb{F}_q(\mathcal{X})$ are algebrically dependent over $\mathbb{F}_q$, that is, the transcendence degree of $\mathbb{F}_q(\mathcal{X})$ over $\mathbb{F}_q$ is 1 and $\mathcal{X}$ is a curve.

Example 1.7.15. Let us consider in $\mathbb{P}^3(\overline{\mathbb{F}}_q)$ the projective algebraic set $\mathcal{X} = V(\{F_1, F_2\})$, where

$$\begin{aligned} F_1(X, Y, Z, T) &= X^3 - Y^3 - Z^3 \\ F_2(X, Y, Z, T) &= X - T. \end{aligned}$$

Now, since $F_2 \in I(\mathcal{X})$, we have that $\frac{T}{F} = \frac{X}{F}$, for any polynomial F with degree 1. Hence all the relations presented in Example 1.7.14 can be used here as well to prove that $\mathbb{F}_q(\mathcal{X})$ has transcendence degree 1 on $\mathbb{F}_q$ and hence that $\mathcal{X}$ is a non-plane curve.

In the rest of this subsection, for the sake of simplicity, we will focus on plane projective curves. Though we precise that it is possible to adapt the following notions also to non-plane curves.

Definition 1.7.16. Let $\mathcal{X} = V(F)$ with $F \in \mathbb{F}_q[X, Y, Z]$ be a plane projective curve and $P \in \mathcal{X}$. Then P is called *singular* if $\frac{\partial F}{\partial X}(P) = \frac{\partial F}{\partial Y}(P) = \frac{\partial F}{\partial Z}(P) = 0$. Furthermore, a curve $\mathcal{X}$ is said to be *nonsingular* or *smooth* if P is nonsingular for any $P \in \mathcal{X}$.

Definition 1.7.17. Let $\mathcal{X}$ be a projective curve and let P be a point of $\mathcal{X}$. A rational function $f \in \mathbb{F}_q(\mathcal{X})$ is *regular* in P if there exist A, $B \in \mathbb{F}_q[X_0, \ldots, X_n]$ homogeneous such that $\deg(A) = \deg(B)$, $B(P) \neq 0$ and $f = \frac{A}{B}$.

Definition 1.7.18. Given a projective curve $\mathcal{X}$ and a point $P \in \mathcal{X}$, the *local ring* of P is defined as

$$\mathcal{O}_P \stackrel{\text{def}}{=} \{f \in \mathbb{F}_q(\mathcal{X}) \mid f \text{ is regular in } P\}.$$

Proposition 1.7.19. *Given a projective curve $\mathcal{X}$ and a point $P \in \mathcal{X}$, the local ring $\mathcal{O}_P$ is indeed local, that is, there exists a unique maximal ideal $\mathfrak{M}_P$ in $\mathcal{O}_P$. In particular this ideal is the following:*

$$\mathfrak{M}_P \stackrel{def}{=} \{f \in \mathcal{O}_P \mid f(P) = 0\}.$$

Definition 1.7.20 (Valuation ring)**.** A *valuation ring* of $\mathbb{F}_q(\mathcal{X})$ is a subring $O \subseteq \mathbb{F}_q(\mathcal{X})$ such that for any $x \in \mathbb{F}_q(\mathcal{X})$ we have

$$x \in O \vee x^{-1} \in O.$$

Proposition 1.7.21. *Let O be a valuation ring of $\mathbb{F}_q(\mathcal{X})$. Then the following statements hold*

1) O is a local ring, indeed its only maximal ideal is $\mathfrak{M} \stackrel{def}{=} O \setminus O^\times$,

2) for any $x \in \mathbb{F}_q(\mathcal{X})$, $x \in \mathfrak{M} \iff x^{-1} \notin O$,

3) the ideal $\mathfrak{M}$ is principal.

Proof. See [Sti09, Theorem 1.1.5, Theorem 1.1.6]. □

Proposition 1.7.22. *Given a projective curve $\mathcal{X}$ and a non singular point $P \in \mathcal{X}$, the local ring $\mathcal{O}_P$ is a valuation ring of $\mathbb{F}_q(\mathcal{X})$.*

Definition 1.7.23. Given a projective curve $\mathcal{X}$, a *place* P of $\mathcal{X}$ is the maximal ideal of a valuation ring O of $\mathbb{F}_q(\mathcal{X})$. Moreover, by Proposition 1.7.21, P is principal, that is there exists $t \in P$ such that $P = tO$. The function t is called *local parameter* of P.

Given the finite field $\mathbb{F}_q$, we consider the Frobenius morphism applied on $\mathbb{P}^2(\overline{\mathbb{F}}_q)$:

$$\begin{cases} \mathbb{P}^2(\mathbb{F}_q) & \longrightarrow & \mathbb{P}^2(\mathbb{F}_q) \\ [x_1, x_2, x_3] & \longmapsto & [x_1^q, x_2^q, x_3^q]\,. \end{cases}$$

Given a point $Q \in \mathcal{X}$, its orbit with respect to the Frobenius morphism is called a *closed point*. We denote the set of closed points of $\mathcal{X}$ by $C(\mathcal{X})$ and, for any $P \in C(\mathcal{X})$, we define $\deg P \stackrel{\text{def}}{=} \#P$. It is possible to prove that there exists a one-to-one correspondence between the places of $\mathbb{F}_q(\mathcal{X})$ and the closed points of $\mathcal{X}$. In particular, a point corresponds to a place if and only if it lives in $\mathbb{P}^2(\mathbb{F}_q)$, that is, $\deg P = 1$.

Definition 1.7.24. Given a projective curve $\mathcal{X}$ on $\mathbb{F}_q$, a point P of $\mathcal{X}$ is said to be *rational* if $P \in \mathbb{P}^2(\mathbb{F}_q)$. The set of rational points is denoted by $\mathcal{X}(\mathbb{F}_q)$.

If we consider the variety on the field $\overline{\mathbb{F}}_q$, then every point is rational. Hence, every point describes a place and *vice versa*. However, later on we will work with a finite field $\mathbb{F}_q$ and we will use the words "place" and "closed point" interchangeably.

Proposition 1.7.25. *If t is a local parameter for P maximal ideal of a valuation ring O, then for any $f \in \mathbb{F}_q(\mathcal{X})$ there exist $a \in \mathbb{Z}$ and $u \in O^\times$ such that $f = ut^a$.*

Given a place P of $\mathcal{X}$ and a function $f \in \mathbb{F}_q(\mathcal{X})$ we know that $\mathcal{O}_P$ is a valuation ring and that there exists a local parameter t which generates $\mathfrak{M}_P$. By Proposition 1.7.25, $f = ut^a$ with $u \in {O_P}^\times$ and $a \in \mathbb{Z}$. Then, it is possible to define the *valuation* at P of f as $v_P(f) \stackrel{\text{def}}{=} a$.

Definition 1.7.26. The place P is a *zero* for f if $v_P(f) > 0$, while it is a *pole* for f if $v_P(f) < 0$. We will use the convention $v_P(0) = \infty$ for any place P.

Proposition 1.7.27. *The described valuation fulfills the following properties:*

- $v_P(fg) = v_P(f) + v_P(g)$;
- $v_P(f+g) \geq \min\{v_P(f), v_P(g)\}$;
- $\exists f \in \mathbb{F}_q(\mathcal{X})$ *such that* $v_p(f) = 1$;
- $v_P(\lambda) = 0$ *for any* $\lambda \in \mathbb{F}_q$.

Definition 1.7.28. Let $\mathcal{X}$ be a curve. The divisor group $\mathrm{Div}(\mathcal{X})$ over $\mathcal{X}$ is the free abelian group generated by the places of $\mathcal{X}$. In particular a *divisor* G is an element of $\mathrm{Div}(\mathcal{X})$, that is, it can be written as

$$G = \sum_{P \in C(\mathcal{X})} n_P P, \tag{1.22}$$

where there is only a finite number of nonzero coefficients n_P. Given the divisor G as in (1.22), its *support* and *degree* are defined respectively as

$$\mathrm{supp}(G) \stackrel{\text{def}}{=} \{P \in C(\mathcal{X}) \mid n_P \neq 0\} \qquad \deg G \stackrel{\text{def}}{=} \sum_{P \in C(\mathcal{X})} n_P \deg P,$$

where we recall that $\deg P_i = \#P$.

If $n_P \geq 0$ for any $P \in \mathrm{supp}(G)$, then the divisor G is said to be *effective* and we write $G \geq 0$. This notion permits to define a partial order over the group $\mathrm{Div}(\mathcal{X})$. Indeed, given two divisors $A = \sum n_{A,P} P$ and $B = \sum n_{B,P} P$ over $\mathcal{X}$, we say that $A \geq B$ if $n_{A,P} \geq n_{P,B}$ for any place P. Since this is not a total order, the objects $\max\{A, B\}$ and $\min\{A, B\}$ are not well defined. We extend this notion as follows.

Definition 1.7.29. Let $\mathcal{X}$ be a curve. Given two divisors $A = \sum n_{A,P} P$ and $B = \sum n_{B,P} P$ on $\mathcal{X}$, we define

$$\max\{A, B\} = \sum_{P \in C(\mathcal{X})} \max\{n_{A,P}, n_{B,P}\} P \qquad \min\{A, B\} = \sum_{P \in C(\mathcal{X})} \min\{n_{A,P}, n_{B,P}\} P$$

Remark 1.7.30. Note that if $A \geq B$, then $\max\{A, B\} = A$ and $\min\{A, B\} = B$.

Given a place P over $\mathcal{X}$, we have seen that it is possible to define the valuation of the non-zero elements of the function field at P. It is possible to do so also for the divisors.

Definition 1.7.31. Given a divisor G as in (1.22) and a place P of $\mathcal{X}$, the *valuation* of G over P is

$$v_P(G) \stackrel{\text{def}}{=} n_P.$$

We now see how these two notions of valuation are related. First we show that every function $f \in \mathbb{F}_q(\mathcal{X})$ induces a divisor.

Proposition 1.7.32. *Any non-zero function $f \in \mathbb{F}_q(\mathcal{X})$ admits a finite number of zeros and poles.*

By Proposition 1.7.32, given a non-zero $f \in \mathbb{F}_q(\mathcal{X})$, we have that the set $\{P \in \mathcal{X} \mid v_P(f) > 0 \vee v_P(f) < 0\}$ is finite. It is possible then to define the divisor

$$(f) \stackrel{\text{def}}{=} \sum_{P \in C(\mathcal{X})} v_P(f) P. \tag{1.23}$$

Hence one can observe that, by construction, the valuation of the divisor (f) in P, is equal to the valuation of the function f at P.

Definition 1.7.33. A divisor A is *principal* if there exists a function $f \in \mathbb{F}_q(\mathcal{X})$ such that $A = (f)$.

Note that, given two principal divisors $A = (f)$ and $B = (h)$, by Proposition 1.7.27 we get $A + B = (fh)$ and $-A = (\frac{1}{f})$. Therefore the set of principal divisors is a subgroup of $\mathrm{Div}(\mathcal{X})$ and will be denoted in what follows by $\mathrm{Princ}(\mathcal{X})$.

Proposition 1.7.34. *Any non-zero function $f \in \mathbb{F}_q(\mathcal{X})$ has the same number of zeros and poles counted with multiplicity. In particular, if G is a principal divisor, then $\deg G = 0$.*

Now we can finally introduce Riemann–Roch spaces, whose notion uses importantly the one of divisor induced by a function and is, in turn, essential to define algebraic geometry codes. From now on, by the word "curve" we denote an absolutely irreducible smooth curve.

Definition 1.7.35. Let $\mathcal{X}$ be a curve over $\mathbb{F}_q$. Given a divisor G on $\mathcal{X}$, the Riemann–Roch space of G is defined as

$$L(G) \stackrel{\text{def}}{=} \{f \in \mathbb{F}_q(\mathcal{X})^\times \mid (f) \geq -G\} \cup \{0\}.$$

We denote by $\ell(G)$ the dimension of $L(G)$ over $\mathbb{F}_q$.

Proposition 1.7.36. *Let $\mathcal{X}$ be a curve over $\mathbb{F}_q$ and A, B be divisors over $\mathcal{X}$. Then the following properties hold:*

- *if $\deg A < 0$, then $L(A) = \{0\}$;*
- *if $g \in L(A)$ and $f \in L(B)$, then $gf \in L(A + B)$;*
- *$L(A) \cap L(B) = L(\min\{A, B\})$;*
- *$L(A) + L(B) \subseteq L(\max\{A, B\})$,*

where we recall that the notions of $\max\{A, B\}$ and $\min\{A, B\}$ are given in Definition 1.7.29.

Definition 1.7.37. Let A, B be divisors over a curve $\mathcal{X}$. We say that A and B are *linearly equivalent* if there exists $G \in \mathrm{Princ}(\mathcal{X})$ such that $A = B + G$.

Proposition 1.7.38. *Let A, B be two linearly equivalent divisors over $\mathcal{X}$. Then $L(A)$ and $L(B)$ are isomorphic over $\mathbb{F}_q$.*

1.7.2 Riemann–Roch Theorem

Riemann–Roch theorem is the most important tool to determine the dimension of a Riemann–Roch space. In order to present this result though, we need to present some more objects relative to a curve $\mathcal{X}$, as for instance its *differential forms* and *genus*. For these notions we will follow [HvLP].

Definition 1.7.39. Given a curve $\mathcal{X}$ over $\mathbb{F}_q$, a *derivation* is a $\mathbb{F}_q$-linear map $D : \mathbb{F}_q(\mathcal{X}) \longrightarrow \mathbb{F}_q(\mathcal{X})$ satisfying for any $f, g \in \mathbb{F}_q(\mathcal{X})$ the Leibenitz property

$$D(fg) = fD(g) + gD(f).$$

The set of derivations is denoted by $\mathrm{Der}(\mathcal{X})$ and it is easy to see that it is a $\mathbb{F}_q(\mathcal{X})$-vector space.

Definition 1.7.40. A *rational differential form* or *differential* is an element of $\mathrm{Der}(\mathcal{X})^\vee$, that is, a $\mathbb{F}_q(\mathcal{X})$-linear map

$$\omega : \mathrm{Der}(\mathcal{X}) \longrightarrow \mathbb{F}_q(\mathcal{X}).$$

The set of differentials is denoted by $\Omega(\mathcal{X})$.

Example 1.7.41. Given a function $f \in \mathbb{F}_q(\mathcal{X})$, we can associate to f the map df defined as

$$\begin{aligned} df : \mathrm{Der}(\mathcal{X}) &\longrightarrow \mathbb{F}_q(\mathcal{X}) \\ D &\longmapsto D(f). \end{aligned}$$

It is easy to see that df is $\mathbb{F}_q(\mathcal{X})$-linear and hence, is a differential.

One may wonder if every element of $\Omega(\mathcal{X})$ is of the form gdf for some $g, f \in \mathbb{F}_q(\mathcal{X})$. The following result gives an affirmative answer.

Proposition 1.7.42. *The set $\Omega(\mathcal{X})$ is a $\mathbb{F}_q(\mathcal{X})$-vector space and its dimension is 1. In particular given $f \in \mathbb{F}_q(\mathcal{X})$ such that $df \neq 0$, for every element $\omega \in \Omega(\mathcal{X})$, there exists $g \in \mathbb{F}_q(\mathcal{X})$ such that $\omega = gdf$.*

Again, as we did for the rational functions and the divisor, it is possible to define a *valuation* also for differentials. Let us consider $\omega \in \Omega(\mathcal{X})$ and P a place of $\mathcal{X}$. We know that there exists $t_P \in \mathbb{F}_q(\mathcal{X})$ a local parameter for P. Furthermore, by Proposition 1.7.42, there exists $g \in \mathbb{F}_q(\mathcal{X})$ such that $\omega = gdt_P$.

Definition 1.7.43. Given $\omega \in \Omega(\mathcal{X})$ and P a place of $\mathcal{X}$, let $t_P \in \mathbb{F}_q(\mathcal{X})$ be a local parameter for P and $g \in \mathbb{F}_q(\mathcal{X})$ a rational function such that $\omega = gdt_P$. Then the *valuation* of ω in P is defined as

$$v_P(\omega) = v_P(g).$$

It is possible to prove that this valuation is well defined and does not depend on the local parameter. Therefore it is possible, given a differential ω, to construct the divisor

$$(\omega) \stackrel{\text{def}}{=} \sum_{P \in C(\mathcal{X})} v_P(\omega) P.$$

Definition 1.7.44. A divisor G is called *canonical* if there exists $\omega \in \Omega(\mathcal{X})$ such that $G = (\omega)$.

Proposition 1.7.45. *All canonical divisors are linearly equivalent.*

Definition 1.7.46. Let $\mathcal{X}$ be a curve over $\mathbb{F}_q$ and let W be a canonical divisor over $\mathcal{X}$. The *genus* of $\mathcal{X}$ is

$$g \stackrel{\text{def}}{=} \ell(W).$$

Observe that, by Proposition 1.7.45, together with Proposition 1.7.38, the genus is well defined. Moreover, for smooth plane curve it is possible to give a simple way to compute it.

Proposition 1.7.47 (Plücker formula)**.** *Let $\mathcal{X} = V(F)$ be a smooth plane curve and let $d = \deg F$. Then, the genus of $\mathcal{X}$ is*

$$g = \frac{(d-1)(d-2)}{2}.$$

Theorem 1.7.48 (Riemann–Roch Theorem)**.** *Let $\mathcal{X}$ be a curve and G a divisor over $\mathcal{X}$. For any W canonical divisor over $\mathcal{X}$, we have*

$$\ell(G) = \deg G - g + 1 + \ell(W - G).$$

The first consequence to Riemann–Roch Theorem is the following result.

Corollary 1.7.49. *Given $\mathcal{X}$ a curve and W a canonical divisor over $\mathcal{X}$, then* $\deg W = 2g - 2$.

Furthermore, it it possible to prove the following properties.

Proposition 1.7.50. *Let A, B be two divisors with $B \geq 0$. Then*

(i) $\ell(A - B) \geq \ell(A) - \deg B$;

(ii) $\ell(A - B) \leq \max\{0, \ell(A) - \ell(B) + 1\}$;

(iii) $\ell(A) \leq \max\{0, \deg A + 1\}$;

(iv) $\ell(A) \geq \deg A - g + 1$;

(v) if $\deg A > 2g - 2$, *then* $\ell(A) = \deg A - g + 1$.

For the proof of (i) see [Sti09, Lemma 1.4.8]. For (ii), one can observe that if $\ell(A - B) = 0$ or $\ell(B) = 0$, then the inequality is clearly true, while the proof for $\ell(A - B), \ell(B) > 0$ can be found in [Sti09, Lemma 1.6.14].

Corollary 1.7.51. *Given a divisor F and a rational point P, if* $\deg F \geq 2g$, *then* $L(F - P) \neq L(F)$.

Finally we will need the following result.

Theorem 1.7.52 (Clifford's Theorem)**.** *For all divisors A with* $0 \leq \deg A \leq 2g - 2$ *holds*

$$\ell(A) \leq 1 + \frac{1}{2} \deg A.$$

Proof. See [TVN07, Sti09]. □

1.7.3 Algebraic geometry codes

In what follows, by *curve* we always mean a smooth absolutely irreducible projective curve defined over $\mathbb{F}_q$. We will focus on the notion of algebraic geometry codes as spaces of evaluations of rational functions and we avoid the representation using the differentials and the residues. The interested reader, can find more details in [TVN07, Sti09].

Definition 1.7.53. Given such a curve $\mathcal{X}$, a divisor G on $\mathcal{X}$ and a sequence $\mathcal{P} = (P_1, \ldots, P_n)$ of rational points of $\mathcal{X}$ avoiding the support of G, one can define the code

$$\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) \stackrel{\text{def}}{=} \{(f(P_1), \ldots, f(P_n)) \mid f \in L(G)\},$$

where $L(G)$ denotes the Riemann–Roch space associated to G (see Definition 1.7.35).

Proposition 1.7.54 (Parameters of an algebraic geometry code)**.** *Let $\mathcal{X}$ be a curve over $\mathbb{F}_q$ and $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ be an algebraic geometry code. If* $\deg G < n$, *then*

$$\dim \mathcal{C} = \ell(G) \geq \deg G + 1 - g, \qquad \mathrm{d} \geq n - \deg G$$

where g is the genus of the curve and d is the minimum distance of the code. Moreover, if $\deg G > 2g - 2$, *we have* $\dim \mathcal{C} = \deg G - g + 1$.

Definition 1.7.55. The *Goppa designed distance* of the code $\mathcal{C}$ is defined as

$$\mathrm{d}^* \stackrel{\text{def}}{=} n - \deg G.$$

Duality and Schur product for algebraic geometry codes

As for Reed–Solomon codes, also algebraic geometry codes behave well with respect to the duality and the Schur product. First, we need some notation. Let us consider a curve $\mathcal{X}$, a differential $\omega \in \Omega(\mathcal{X})$ and a place P of $\mathcal{X}$. We know that if t_P is a local parameter for P, there exists $f \in \mathbb{F}_q(\mathcal{X})$ such that $\omega = f dt_P$. Furthermore, it is possible to expand f in a Laurent series in t_P:

$$f = \sum_{i \in \mathbb{Z}} a_i t_P^i.$$

The term a_{-1} is called *residue* of ω at P and is denoted by $\mathrm{Res}_P(\omega)$.

Proposition 1.7.56. *Let $\mathcal{P} = \{P_1, \dots, P_n\} \subseteq \mathcal{X}$. Then there exists a differential $\omega \in \Omega(\mathcal{X})$ such that*

$$v_{P_i}(\omega) = -1 \quad \textit{and} \quad \mathrm{Res}_\omega(P_i) = 1 \quad \forall i = 1, \dots n,$$

Let us consider a differential ω as in Proposition 1.7.56 (we know it exists). We can define the two divisors

$$W \stackrel{\text{def}}{=} (\omega) \qquad D \stackrel{\text{def}}{=} \sum_{P \in \mathcal{P}} P. \tag{1.24}$$

Proposition 1.7.57. *Let $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ and W, D be as above. The dual code of $\mathcal{C}$ is*

$$\mathcal{C}^\perp = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, W + D - G).$$

Proof. See [HvLP]. □

Proposition 1.7.58 (Star product of AG codes)**.** *Let $\mathcal{X}$ be a curve of genus g, G, G' be two divisors of $\mathcal{X}$ and $\mathcal{P} = (P_1, \dots, P_n)$ be a sequence of rational points of $\mathcal{X}$ avoiding the support of G and G'. The following statements hold:*

- *we have $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) * \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G') \subseteq \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G + G')$;*
- *if $\deg G \geq 2g$ and $\deg G' \geq 2g + 1$, then $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) * \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G') = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G + G')$.*

Proof. This is a consequence of [Mum70, Theorem 6]. For instance, see [CMCP17, Corollary 9]. □

Remark 1.7.59. In this chapter we presented two class of codes, which are the Reed–Solomon and the algebraic geometry codes. Actually, it is easy to see that the first class is included in the second one, that is, Reed–Solomon codes are particular algebraic geometry codes. Indeed, let us consider $\mathcal{X} \stackrel{\text{def}}{=} \mathbb{P}^1(\mathbb{F}_q)$, $\mathcal{P} = \{P_1, \dots, P_n\}$ where $P_i = [x_i, 1]$ and for every $i \neq j$, then $x_i \neq x_j$. Finally let $P_\infty \stackrel{\text{def}}{=} [1, 0]$ and $G \stackrel{\text{def}}{=} kP_\infty$. By the Riemann-Roch Theorem, since the genus g of $\mathcal{X}$ is zero, we have $\ell(G) = k + 1$. In particular we have

$$L(G) = \mathrm{span}_{\mathbb{F}_q} \left\{ 1, \frac{X}{Y}, \dots, \frac{X^k}{Y^k} \right\}.$$

Now, evaluating $\frac{X^h}{Y^h}$ in $[x_i, 1]$ is equivalent to evaluating X in x_i. Hence given $\boldsymbol{x} = (x_1, \dots, x_n)$, we have

$$\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) = \mathbf{RS}_q[\boldsymbol{x}, k + 1].$$

1.8 Basic algorithms for algebraic geometry codes

In this section we give a partial state of art of the decoding algorithms designed for algebraic geometry codes. Before though, we want to stress that although for Reed–Solomon codes we know exactly the value of the minimum distance, for algebraic geometry codes we do not dispose of a formula to compute it and, generally, it is not easy to find it. We only have some lower bounds for it, as the *designed distance* of the code d^* (see Definition 1.7.55). The decoding radius of the following algorithms will be then given with respect to d^* and not d. The first decoding algorithm for algebraic geometry codes, also called *basic algorithm*, was proposed by Justesen, Larsen, Jensen, Havemose and Høholdt in 1989 ([JLJ+89]).

In parallel Porter [Por88], gave an equivalent algorithm using some *key equations*. The decoding radius of these algorithms is $\frac{\mathrm{d}^*-1-g}{2}$, where g is the genus of the curve, which leaves room to improvement, as we know that unique decoding is theoretically possible whenever the number of errors is less than $\frac{\mathrm{d}-1}{2}$. Hence, from that moment, much effort has been aimed at erasing the term in g by adapting the basic algorithm accordingly. In this spirit, the same year, Pellikaan proved the existence of an algorithm able to correct up to $\frac{\mathrm{d}^*-1}{2}$ errors, but with no practical procedure to provide it ([Pel89]). At last, in 1993, a constructive procedure to achieve the correction capacity $\frac{\mathrm{d}^*-1}{2}$ was finally found by Ehrhard ([Ehr93]), whose algorithm consists in selecting a "good" divisor F with a progressive adaptation process and in applying the basic algorithm on it. Finally, it is important to mention that in the same year Feng and Rao proposed another method, called *majority vote* for *unknown syndromes* and able to correct the same amount of errors ([FR93]).

As for Reed–Solomon codes, for algebraic geometry codes, it is possible to design decoding algorithms

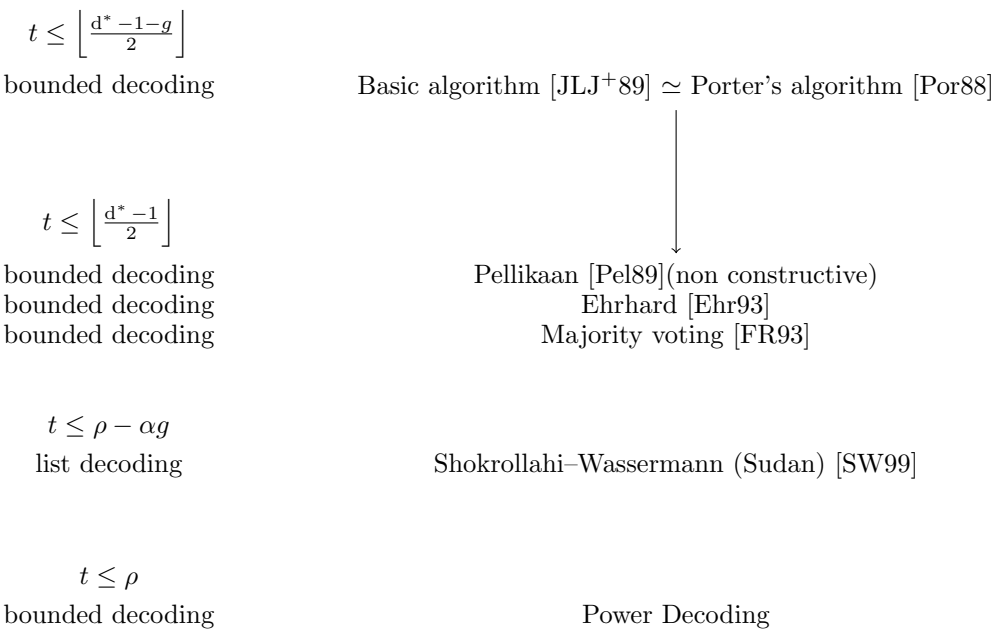

Figure 1.2: Some decoding algorithms for algebraic geometry codes

to correct amount of errors even larger than half the minimum distance. Again, we distinguish the algorithms in *bounded decoders* and *list decoders*. Since Reed–Solomon codes are particular algebraic geometry codes (see Remark 1.7.59), these algorithms come as natural generalizations of the ones for Reed–Solomon codes. Among them, we have the power decoding algorithm which, we recall, is a bounded decoder. Interestingly, in the transition from Reed–Solomon codes to algebraic geometry codes, its decoding radius does not get any penalty in the genus of the curve g. On the other hand, Shokrollahi and Wassermann generalized Sudan's algorithm to algebraic geometry codes [SW99], getting this time though a decoding radius with a penalty in g. That means that for amounts of errors between $\frac{\mathrm{d}^*-1-g}{2}$ and $\frac{\mathrm{d}^*-1}{2}$, Sudan's algorithm cannot be run to solve the list decoding problem and even though Guruswami–Sudan algorithm can be as well extended to algebraic geometry codes, we precise that this algorithm is quite expensive in terms of complexity. The reader can find in Figure 1.2 this partial state of art for algebraic geometry codes. We denote by α a real positive parameter and by ρ the decoding radius of the power decoding algorithm, which is independent from the genus g and, under some circumstances, is larger than half the minimum distance of the code.

In the following, we will present some of these decoding algorithms for algebraic geometry codes. All along we will consider a code $\mathcal{C} \stackrel{\text{def}}{=} \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ with $\deg G < n$. Furthermore, we assume to have a received vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $t = \mathrm{w}(\boldsymbol{e})$ and we recall that we denote the support of the error vector by

$I_{\boldsymbol{e}}$. Finally, we recall that $D = \sum_{P \in \mathcal{P}}$ and we define the divisor

$$D_{\boldsymbol{e}} = \sum_{i \in I_{\boldsymbol{e}}} P_i.$$

1.8.1 Unique decoding : the basic algorithm

In this section, we will focus mainly on the basic algorithm, which is the first decoding algorithm designed for algebraic geometry codes. The basic algorithm mainly consists, like many others algorithms which followed, in locating the error positions $\mathrm{supp}(\boldsymbol{e})$. To do so, the strategy is to introduce a divisor F with $\mathrm{supp}(F) \cap \mathrm{supp}(D) = \emptyset$ and recover someway the space $L(F - D_{\boldsymbol{e}})$. Indeed, any nonzero function f belonging to this space *locates* the error positions, as

$$e_i \neq 0 \implies f(P_i) = 0.$$

For that, a nonzero element of $L(F - D_{\boldsymbol{e}})$ takes the name of *locator function* for $\boldsymbol{e}$. However, in 1992 the technical language of functions and Riemann–Roch spaces was translated, by Pellikaan and Kötter, in the simple language of codes and linear algebra. This process gave a new algorithm, called *error correcting pairs algorithm* (see §1.6.2) which is possible to apply not only to algebraic geometry codes but to every code with a special structure. Of course, for algebraic geometry codes it behaves just like the basic algorithm. Here we will then show the point of view of the error correcting pairs algorithm but the interested reader, can find the original point of view of functions and Riemann–Roch spaces in [JLJ+89, HP95]. We recall that in order to apply the error correcting pairs algorithm to our code $\mathcal{C}$, we need to exhibit a t-error correcting pair for $\mathcal{C}$. Here we propose the pair $(\mathcal{A}, \mathcal{B})$, where

$$\mathcal{A} \stackrel{\text{def}}{=} \mathcal{C}_L(\mathcal{X}, \mathcal{P}, F) \qquad \mathcal{B}^\perp \stackrel{\text{def}}{=} \mathcal{C}_L(\mathcal{X}, \mathcal{P}, F + G),$$

$\deg F = t + g$ and $\deg(F + G) < n$.

Proposition 1.8.1. *The pair $(\mathcal{A}, \mathcal{B})$ is a t-error correcting pair for $\mathcal{C}$ if*

$$t \leq \frac{\mathrm{d}^* - 1 - g}{2}.$$

Proof. First, we recall that by Remark 1.6.6, $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp \iff \mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^\perp$. Now, since $L(F)L(G)$ is included in $L(F + G)$, we have

$$\mathcal{A} * \mathcal{C} \subseteq \mathcal{C}_L(\mathcal{X}, \mathcal{P}, F + G) = \mathcal{B}^\perp$$

and (ECP1) holds. Furthermore by Riemann–Roch theorem, together with the hypothesis $\deg F = t + g$, we get (ECP2) for any t. For (ECP3) we know that since $\deg(F + G) < n$, we have a lower bound for the minimum distance of $\mathcal{B}^\perp$, that is

$$\mathrm{d}(\mathcal{B}^\perp) > n - \deg G - t - g \geq n - \deg G - \frac{n - \deg G - g - 1}{2} - g > t.$$

Finally, we have $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n - \deg F + n - \deg G > n$, since we supposed $\deg(F + G) < n$. Therefore $(\mathcal{A}, \mathcal{B})$ is a t-error correcting pair for $\mathcal{C}$. □

1.8.2 Beyond half the designed distance : power decoding algorithm

Power decoding extends naturally from Reed–Solomon codes to algebraic geometry codes. However, until few years ago, its use for decoding AG codes in the literature concerned mainly one–point codes from the Hermitian curve (see [NB15, PRB19]). In particular, for these specific curves it was proven that the decoding radius of the algorithm achieves Johnson's bound. Recently, during the development of this manuscript, [PRS21] generalized this result to all algebraic curves. However, to avoid digressions which are far from our purposes, in this section we just give a brief presentation of the basic version of the power decoding for algebraic geometry codes, together with an analysis of its decoding radius. Let $\mathcal{C} = C_L(\mathcal{X}, \mathcal{P}, G)$ with $2g - 2 < \deg G < n$ and $\mathbf{y} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n$ the vector we want to correct, where $\mathbf{c} \in \mathcal{C}$, that is

$$c = \mathrm{ev}_{\mathcal{P}}(f) \text{ with } f \in L(G).$$

By Assumption 1, we have $\mathrm{w}(\mathbf{e}) = t$, where we assume $t \geq \frac{\mathrm{d}^* - 1}{2}$. The algorithm is really similar to the version for Reed–Solomon codes: let us suppose to have $\Lambda \in \mathbb{F}_q(\mathcal{X})$ such that $\Lambda(P_i) = 0$ for all $i \in I_{\boldsymbol{e}}$. Then, given $\ell \in \mathbb{N}$ we get

$$\Lambda(P_i) y_i^j = \Lambda(P_i) f^j(P_i) \quad \forall\, i = 1, \dots, n, \quad j = 1, \dots \ell. \tag{1.25}$$

We would like then to find Λ as above. It is easy to see that, given a divisor F, such a Λ exists in $L(F) \setminus \{0\}$ whenever $\mathrm{supp}(F) \cap \mathcal{P} = \emptyset$ and $\deg F \geq t + g$. Let us consider then a divisor F satisfying these properties and let $\Lambda \in L(F)$ such that $\Lambda(P_i) = 0$ for any $i \in I_{\boldsymbol{e}}$. It is possible to see (Λ, f) as a solution of the following problem.

Key Problem 5. Find (λ, φ) with $\lambda \in L(F)$ and $\varphi \in L(G)$, such that

$$\lambda(P_i) y_i^j = \lambda(P_i) \varphi^j(P_i) \quad \forall\, i = 1, \dots, n, \quad j = 1, \dots \ell. \tag{1.26}$$

This is a system of $n\ell$ equations whose unknowns are the coordinates of λ and φ in the basis of respectively $L(F)$ and $L(G)$. System (1.26) is not linear in the unknowns though, hence we linearize it by considering a new function $\nu_j \coloneqq \lambda \varphi^j$ for any equation. For all $j \in \{1, \dots, \ell\}$, we get

$$\nu_j \in L(F) L(jG) \subseteq L(F + jG).$$

We get then the following problem.

Key Problem 6. Given $\mathbf{y} \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$, look for $\lambda, \nu_1, \dots, \nu_\ell \in \mathbb{F}_q^n(\mathcal{X})$ such that

- $\lambda \in L(F)$ with $\deg(F) = t + g$;
- $\nu_j \in L(F + jG)$ for all $j = 1, \dots, \ell$;
- $\lambda(x_i) y_i = \nu_j(x_i)$ for all $i = 1, \dots, n$ and $j = 1, \dots, \ell$.

Therefore, even in this case, the power decoding algorithm consists in solving a linear system. Then a nonzero solution $(\lambda, \nu_1, \dots, \nu_\ell)$ such that λ has the minimum number of zeros among the points of $\mathcal{P}$, is picked. To conclude, this solution is tested to see if it gives a codeword which is at distance t from $\boldsymbol{y}$.

Remark 1.8.2. One can notice that, since algebraic geometry codes behave well with respect to the Schur product, in the linearization process, the number of unknowns does not explode.

Decoding Radius. As for Reed–Solomon codes, since we know there exists a solution to the linear system $(\Lambda, \Lambda f, \dots, \Lambda f^\ell)$ we would like the dimension of the solution space of to be at most one. A necessary condition for that, is

$$\#unknowns \leq \#equations + 1. \tag{1.27}$$

The number of equations is $n\ell$. For the number of unknowns, we need to know the dimension of the spaces $L(F + jG)$ for all $j = 1, \dots, \ell$. The bounds we have set in the hypothesis give

$$\dim(L(F + jG)) = t - g + j \deg(G) + 1.$$

Hence by condition (1.27) we get the following decoding radius

$$t \leq \frac{2n\ell - \ell(\ell + 1) \deg(G)}{2(\ell + 1)} - \frac{\ell}{\ell + 1}, \tag{1.28}$$

which indeed corresponds to the empirical result obtained in [NB15].

Remark 1.8.3. Thanks to the choice $\deg F = t + g$, we get a decoding radius which does not depend on the genus of the curve g. Despite this positive aspect, the obtained decoding radius is neither a sufficient, nor a necessary condition to find the solution $\boldsymbol{c}$. To see that, we compare this case with the one for Reed–Solomon codes. For Reed–Solomon codes, let $(\Lambda, \Lambda f_{\boldsymbol{c}}, \Lambda f_{\boldsymbol{c}}^2)$ be the solution we want to find. If $t = t_{max} + 1$, there exists (Γ, ν_1, ν_2) which is independent from $(\Lambda, \Lambda f_{\boldsymbol{c}}, \Lambda f_{\boldsymbol{c}}^2)$. It is easy to prove that such a solution with a Γ vanishing in more points than Λ cannot exists (we would have $\deg \Gamma > t_{max} + 1$, while

by construction, $\deg \Gamma \leq t_{max}+1$). Hence Λ cannot be the function with the smallest degree among the solutions. For algebraic geometry codes, we do not have the same nice properties. We have $\ell(F) = t_{max}+2$ and again, since $t > t_{max}$, there exists a solution (Γ, ν_1, ν_2) independent from $(\Lambda, \Lambda f_{\boldsymbol{c}}, \Lambda f_{\boldsymbol{c}}^2)$ and such that $\Gamma \in L(F)$. Since we do not have $\deg F - t_{max} - 2 > 2g-2$, then $t_{max}+2$ conditions of the form $\Gamma(P) = 0$ are not necessarily independent on $L(F)$, hence a nonzero Γ could vanish in more points than Λ. Hence, even if $t > t_{max}$, it is possible for Λ to be the nonzero function in the space of solutions with the minimum number of zeros among the points in $\mathcal{P}$.

1.8.3 Considerations on the decoding radius of Sudan's algorithm

The idea presented in this section has been suggested by Peter Beelen in a e-mail correspondence and shows how to get an improved decoding radius for Sudan's algorithm with respect to the one of Shokrollahi–Wassermann for algebraic geometry codes. This improvement mainly consists in analyzing the parameters of the linear system to get Sudan polynomial Q. Given a curve of genus g, we consider a code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, with $2g-2 < \deg G < n$, and the received vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $\mathrm{w}(\boldsymbol{e}) = t$. Let F be a divisor with $\deg F = n - t - 1$.

Original problem (Sudan): given $\ell \geq 1$, find a polynomial $Q(\boldsymbol{x}, y) = Q_0(\boldsymbol{x}) + Q_1(\boldsymbol{x})y + \cdots + Q_\ell(\boldsymbol{x})y^\ell$ such that

(*i*) $Q_i(\boldsymbol{x}) \in L(F - iG)$ for all $i = 0, \ldots, \ell$;

(*ii*) $Q(P_j, y_j) = 0$ for all $j = 1, \ldots, n$.

This problem can be solved with a linear system of n equations in $\sum_{i=0}^{\ell} \ell(F - iG)$ unknowns. Hence the system has nonzero solutions if

$$t \leq \frac{2n\ell - \ell(\ell+1)\deg(G) - 2}{2(\ell+1)} - g. \tag{1.29}$$

Now, we want to show that this decoding radius can be actually optimized. To do so, let us consider a Sudan polynomial Q, that is, a polynomial verifying *(i-ii)*. We have the following result.

Lemma 1.8.4. *Let $f \in L(G)$ such that $\mathrm{d}(\mathrm{ev}_{\mathcal{P}}(f), \boldsymbol{y}) \leq t$. Then $Q(\boldsymbol{x}, f(\boldsymbol{x})) = 0$.*

Proof. The proof is analog to the one for Reed–Solomon codes (see [BH08, Lemma 2.32]). □

Lemma 1.8.5. *Let Q be a polynomial satisfying (i-ii) and let $f_{\boldsymbol{c}} \in L(G)$ such that $\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}}) = \boldsymbol{c}$. Then Q can be written as*

$$Q(\boldsymbol{x}, y) = (y - f_{\boldsymbol{c}}(\boldsymbol{x}))(\tilde{Q}_0(\boldsymbol{x}) + \tilde{Q}_1(\boldsymbol{x})y + \cdots + \tilde{Q}_{\ell-1}(\boldsymbol{x})y^{\ell-1}), \tag{1.30}$$

where $\tilde{Q}_i(\boldsymbol{x}) \in L(F - (i+1)G)$ for $i = 0, \ldots, \ell-1$.

Proof. Since $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) = t$, by Lemma 1.8.4, we have $Q(f_{\boldsymbol{c}}) = 0$, that is, Q can be written as in (1.30). To conclude, we prove that $\tilde{Q}_i(\boldsymbol{x}) \in L(F - (i+1)G)$ for $i = 0, \ldots, \ell-1$. It is clear that

$$\tilde{Q}_{\ell-1}(\boldsymbol{x}) = Q_\ell(\boldsymbol{x}) \in L(F - \ell G).$$

Now, if $\tilde{Q}_{i+1}(\boldsymbol{x}) \in L(F - (i+2)G)$, then $\tilde{Q}_i(\boldsymbol{x}) \in L(F - (i+1)G)$. Indeed, we have

$$\tilde{Q}_i(\boldsymbol{x}) = f_{\boldsymbol{c}}(\boldsymbol{x})\tilde{Q}_{i+1}(\boldsymbol{x}) + Q_i(\boldsymbol{x})$$

and, since $f_{\boldsymbol{c}} \in L(G)$, we conclude. □

Remark 1.8.6. One can easily see that, if Q verifies *(i-ii)*, and the $\tilde{Q}_i$'s are as in (1.30), then we have for any $j \in I_{\boldsymbol{e}}$

$$\tilde{Q}_0(P_j) + \tilde{Q}_1(P_j)y_j + \cdots + \tilde{Q}_{\ell-1}(P_j)y_j^{\ell-1} = 0.$$

Let us consider then the space S of polynomials satisfying *(i-ii)* and the space

$$\tilde{S} \stackrel{\text{def}}{=} \{\tilde{Q} = \tilde{Q}_0 + \tilde{Q}_1 y + \cdots + \tilde{Q}_{\ell-1}y^{\ell-1} \mid \tilde{Q}_i \in L(F-(i+1)G)\ \forall i = 0, \ldots, \ell-1 \ \wedge \ \tilde{Q}(P_j, y_j) = 0\ \forall j \in I_{\boldsymbol{e}}\}.$$

Lemma 1.8.7. *There exists a one-to-one correspondence between S and $\tilde{S}$*

$$\begin{array}{ccc} S & \longleftrightarrow & \tilde{S} \\ Q & \longmapsto & \frac{Q}{(y-f_{\boldsymbol{c}})} \\ \tilde{Q}(y-f_{\boldsymbol{c}}) & \longmapsto & \tilde{Q} \end{array}$$

Thanks to lemma 1.8.7, we have that a nonzero Sudan polynomial exists if and only if $\tilde{S} \neq \{0\}$. The space $\tilde{S}$ can be in turn seen as the solution space of the following linear system:

$$\tilde{Q}_0(P_i) + \tilde{Q}_1(P_i)y_i + \cdots + \tilde{Q}_{\ell-1}(P_i)y_i^{\ell-1} = 0 \quad \forall i \in \mathrm{supp}(D_{\boldsymbol{e}}).$$

This is a system of t equations in $\sum_{i=1}^{\ell} L(F+iG)$ unknowns. Hence $\tilde{S}$ is nonzero if the number of unknowns is larger than the number of equations, which gives

$$t \leq \frac{2n\ell - \ell(\ell+1)\deg(G) - 2}{2(\ell+1)} - \frac{\ell g}{\ell+1}. \tag{1.31}$$

1.9 Other algebraic codes

Several other algebraic constructions of codes can be deduced from Reed–Solomon and algebraic geometry codes. For instance, alternant codes and BCH codes are subfield subcodes of generalized Reed–Solomon codes, and hence they inherit the decoding algorithms proposed in this chapter. Anyway the class of cyclic codes, which contains the one of BCH, takes some distance from Reed–Solomon codes. We will give more details about this class in Chapter 2 and show that it is possible to provide a bounded decoding algorithm to correct beyond half the minimum distance, to others than BCH codes.

Chapter 2

The Power Error Locating Pairs algorithm

2.1 Power error locating pairs algorithm

We have seen in Chapter 1 that several decoding algorithms exist for Reed–Solomon codes and, more in general, algebraic geometry codes for amounts of errors even larger than half the minimum distance. In particular the error correcting pairs algorithm (see § 1.6.2) reformulates both Welch–Berlekamp and the basic algorithms in a language using only codes and their parameters. Thanks to these tools, ECP provides a unique decoding algorithm for every code which disposes of an error correcting pair. It is then natural to wonder whether is possible to reformulate in the same spirit an algorithm with a larger correction capacity. That is, if it is possible to fill the gap with the question mark in Figure 2.1:

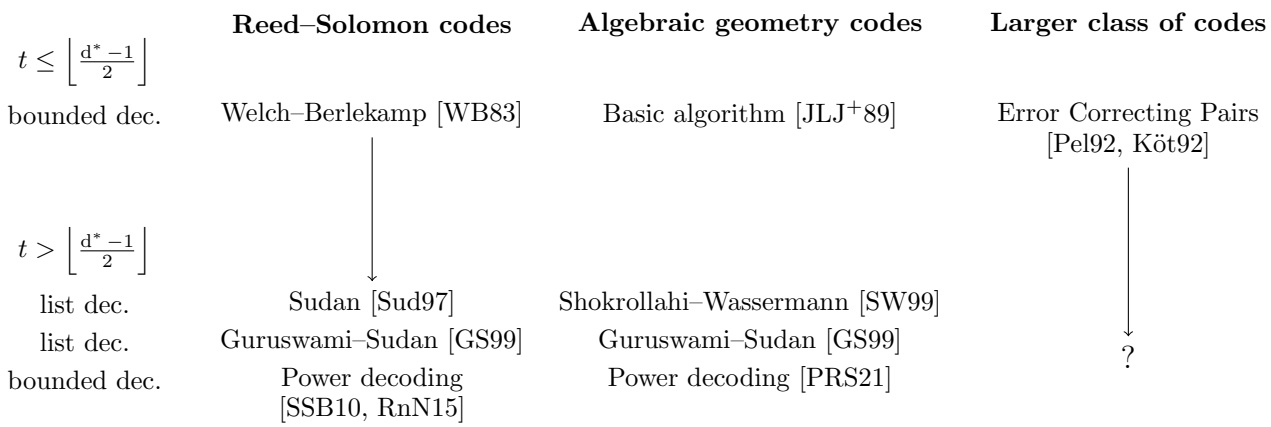

Figure 2.1: Extensions of decoding algorithms.

The answer is affirmative: the algorithm is a bounded decoder (see Definition 1.3.7) and takes the name of *Power Error Locating Pairs* algorithm (PELP). This chapter relates the results of a published paper and is part of our contribution. As for the error correcting pairs algorithm (see 1.6.2), we first give a generic description of the algorithm and later, some examples of its application. In order to generalize the ECP algorithm to correct more errors, we introduce a new parameter ℓ we call *power* and define a slightly different structure from error correcting pairs. We first describe that structure and the algorithm for $\ell = 2$ and then explain how to generalize it to $\ell \geq 2$. In the entire chapter we will work under Assumption 1, that is, we will suppose that there exist $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ such that $\mathrm{w}(\boldsymbol{e}) = t$ and $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.

2.1.1 The case $\ell = 2$

In Pellikaan's paper [Pel92], a structure called *error locating pairs* is already defined. It is a pair of codes $(\mathcal{A}, \mathcal{B})$ which satisfy (ECP1, 2) and (ECP4). In particular it is shown that, without changing anything in the algorithm, with such a structure it is possible to correct errors if the support of the error vector $I_{\boldsymbol{e}}$ is an *independent t-set of error position* with respect to the code $\mathcal{B}$, that is, if

$$(\mathcal{A} * \boldsymbol{e}) \cap \mathcal{B}^{\perp} = \{0\}.$$

In this chapter, in order to correct beyond half the designed distance of the code, we do not consider particular error supports, but we rather choose to work with a more particular structure than error locating pairs and change the first step of the algorithm.

Definition 2.1.1 (2–Power error locating pairs)**.** Given a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$, a pair of linear codes $(\mathcal{A}, \mathcal{B})$ with $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$ is a 2*–power t–error locating pair* for $\mathcal{C}$ if

(2–PELP1) $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^{\perp}$;

(2–PELP2) $\dim(\mathcal{A}) > t$;

(2–PELP3) $\mathrm{d}(\mathcal{A}^{\perp}) > t$;

(2–PELP4) $\dim(\mathcal{B}) + \dim(\mathcal{B}^{\perp} * \mathcal{C})^{\perp} \geq t$;

(2–PELP5) $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$.

With respect to the definition of error correcting pairs, we removed (ECP 3) which is too restrictive to correct errors beyond half the minimum distance (see the following Remark 2.1.2). Instead, in the same spirit as the power decoding, we look for a necessary condition for the algorithm to succeed. In this context, under Condition (2–PELP3), Condition (2–PELP4) provides this necessary condition together with the key tool for the analysis of the decoding radius of our algorithm. We recall that $\mathcal{A}$ represents the space of locators for $\boldsymbol{e}$ and that Condition (2–PELP2) guarantees the existence of at least a nonzero locator.

Remark 2.1.2. In the transition between ECP algorithm and 2–PELP algorithm, it is very important to get rid of the property $\mathrm{d}(\mathcal{B}^{\perp}) > t$. Indeed, since we want $\mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^{\perp}$, if we had $\mathrm{d}(\mathcal{B}^{\perp}) > t$, then, assuming that $\mathcal{A} * \mathcal{C}$ is non degenerate (see § 1.4.2) we would get

$$t + \dim(\mathcal{C}) \leq \dim(\mathcal{A}) + \dim(\mathcal{C}) - 1 \leq \dim(\mathcal{A} * \mathcal{C}) \leq \dim(\mathcal{B}^{\perp}) \leq n - t,$$

where the second inequality is due to Corollary 1.4.9 and the third, to the Singleton bound (see Proposition 1.2.4). This entails that $t \leq \frac{n - \dim(\mathcal{C})}{2}$. Then, for instance for Reed–Solomon codes, the condition $\mathrm{d}(\mathcal{B}^{\perp}) > t$ would hinder an improvement of the decoding radius of the algorithm (see §1.6.2).

Let us consider a code $\mathcal{C}$, a word $\boldsymbol{y} \in \mathbb{F}_q^n$ such that $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $\boldsymbol{c} \in \mathcal{C}$ and $t = \mathrm{w}(\boldsymbol{e})$ and let $(\mathcal{A}, \mathcal{B})$ be a 2–power t–error locating pair for $\mathcal{C}$.

Definition 2.1.3. As in the ECP algorithm, let $M_1 = \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^{\perp}\}$. We then define the spaces

$$M_2 \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y}^2 \in \mathcal{B}^{\perp} * \mathcal{C}\}$$

and

$$M \stackrel{\text{def}}{=} M_1 \cap M_2.$$

Notation. As in the description of the power decoding algorithm, since we work with $\boldsymbol{y}$ and $\boldsymbol{y}^2$, we indicate with $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(2)}$ the vectors such that respectively $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}^{(1)}$ and $\boldsymbol{y}^2 = \boldsymbol{c}^2 + \boldsymbol{e}^{(2)}$.

Algorithm 5 2–Power error locating pairs algorithm

Inputs: $\mathcal{C}$ linear code, $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_q^n$, $t = \mathrm{w}(\boldsymbol{e})$ and $(\mathcal{A}, \mathcal{B})$ a 2–power t–error locating pair for $\mathcal{C}$
Output: a codeword $\tilde{\boldsymbol{c}}$ such that $\mathrm{d}(\tilde{\boldsymbol{c}}, \boldsymbol{y}) = t$ or "?".

compute $M = M_1 \cap M_2$ by solving a linear system
$J \leftarrow Z(M)$
if system (1.13) does not have any nonzero solution **then**
    return "?"
$\boldsymbol{e}_J^{(1)} \leftarrow$ nonzero solution of (1.13)
recover $\boldsymbol{e}^{(1)}$ from $\boldsymbol{e}_J^{(1)}$
if $\mathrm{w}(\boldsymbol{e}^{(1)}) = t$ **then**
    return $\tilde{\boldsymbol{c}} = \boldsymbol{y} - \boldsymbol{e}^{(1)}$
return "?"

As shown in Algorithm 5, the only change from the error correcting pair algorithm consists in computing a new set M, which is smaller than the set M_1 considered in the basic algorithm. The reason why we do so, is that since we are now working with a 2–power error locating pair, we are no longer asking for $\mathrm{d}(\mathcal{B}^\perp) > t$. Hence, without this property the equality

$$\mathcal{A}(I_{\boldsymbol{e}}) = M_1$$

is not entailed and only the inclusion $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M_1$ still holds. That is, M_1 could be too large with respect to $\mathcal{A}(I_{\boldsymbol{e}})$ and in this case the algorithm would fail. Indeed, if there was $\boldsymbol{a} \in M_1 \setminus \mathcal{A}(I_{\boldsymbol{e}})$, that is $\boldsymbol{a}_{I_{\boldsymbol{e}}} \neq 0$, we would have $J = Z(M_1) \not\supseteq I_{\boldsymbol{e}}$.

Proposition 2.1.4. *If $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp$, then the set M fulfills $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M \subseteq M_1 \subseteq \mathcal{A}$.*

Proof. We have to prove that $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M$, that is $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M_1, M_2$. First, we already know that $\mathcal{A}(I_{\boldsymbol{e}}) \subseteq M_1$ by Theorem 1.6.11(1). About M_2, we adapt the proof of Theorem 1.6.11(1). First, note that we have

$$\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp \iff \mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^\perp \implies \mathcal{A} * \mathcal{C}^2 \subseteq \mathcal{B}^\perp * \mathcal{C}. \tag{2.1}$$

Given $\boldsymbol{a} \in \mathcal{A}(I_{\boldsymbol{e}})$, for any $\boldsymbol{v} \in (\mathcal{B}^\perp * \mathcal{C})^\perp$ we obtain

$$\begin{aligned} \langle \boldsymbol{a} * \boldsymbol{y}^2, \boldsymbol{v} \rangle &= \langle \boldsymbol{a} * \boldsymbol{c}^2, \boldsymbol{v} \rangle + \langle \boldsymbol{a} * \boldsymbol{e}^{(2)}, \boldsymbol{v} \rangle & (2.2) \\ &= 0. & (2.3) \end{aligned}$$

In (2.2) we used the bilinearity, while (2.3) holds because of (2.1) and the fact that the supports of $\boldsymbol{e}^{(2)}$ and $\boldsymbol{a}$ are disjoint (see Proposition 1.6.24). □

Thanks to this proposition, one can see that if we work with a 2–PELP, there are more chances to have $\mathcal{A}(I_{\boldsymbol{e}}) = M$ than $\mathcal{A}(I_{\boldsymbol{e}}) = M_1$, that is why in 2–PELP algorithm, we look for M instead of M_1. Furthermore, we get the following result.

Theorem 2.1.5. *Let $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $\boldsymbol{c} \in \mathcal{C}$ and $t = \mathrm{w}(\boldsymbol{e})$. Suppose a pair of codes $(\mathcal{A}, \mathcal{B})$ verifies Conditions (2–PELP1, 2, 5) with respect to t. Then Algorithm 5 returns $\boldsymbol{c}$ if and only if $\mathcal{A}(I_{\boldsymbol{e}}) = M$.*

Proof. Suppose the algorithm returned $\boldsymbol{c}$. This entails that we computed a set $J = Z(M)$ which contains $I_{\boldsymbol{e}}$ and hence $M = M(I_{\boldsymbol{e}})$. Therefore, from Proposition 2.1.4, we get $M = \mathcal{A}(I_{\boldsymbol{e}})$. Conversely, if $M = \mathcal{A}(I_{\boldsymbol{e}})$, then $J = Z(M) \supseteq I_{\boldsymbol{e}}$. Next, using Conditions (2–PELP2) and (2–PELP5), one has $|J| < \mathrm{d}(\mathcal{C})$ (see the proof of Lemma 1.6.12). Hence, given a parity check matrix H, the system

$$H_J \cdot \boldsymbol{u}^T = H \cdot \boldsymbol{y}^T$$

admits a unique solution $\boldsymbol{e}_J$. Filling the other entries of the solution with zeros, the algorithm finds $\boldsymbol{e}$ and returns $\boldsymbol{c} = \boldsymbol{y} - \boldsymbol{e}$. □

We now show how the properties (2–PELP3) and (2–PELP4) are involved in 2–PELP algorithm. To do so, we want to focus on necessary conditions for the algorithm to return $\boldsymbol{c}$. By Theorem 2.1.5, this is equivalent to look for a necessary condition to have $\mathcal{A}(I_{\boldsymbol{e}}) = M$, which is the point of the following statement and explains the rationale behind Condition (2–PELP4).

Theorem 2.1.6. *Suppose Conditions (2–PELP1) is verified. If $\mathcal{A}(I_{\boldsymbol{e}}) = M$ and $\mathrm{d}(\mathcal{A}^\perp) > t$, then*

$$\dim(\mathcal{B}) + \dim(\mathcal{B}^\perp * \mathcal{C})^\perp \geq t.$$

Before proving Theorem 2.1.6, we need to present some other results.

Lemma 2.1.7. *If Condition (2–PELP1) holds, the following statements are equivalent:*

(i) $M = \mathcal{A}(I_{\boldsymbol{e}})$;

(ii) $M(I_{\boldsymbol{e}}) = M$;

(iii) $M_{I_{\boldsymbol{e}}} = \{0\}$.

Proof. First, notice that $\mathcal{A}(I_{\boldsymbol{e}}) = M(I_{\boldsymbol{e}})$ by Proposition 2.1.4. Hence, *(i)* and *(ii)* are equivalent. We now prove *(ii)* $\iff$ *(iii)*. If $M(I_{\boldsymbol{e}}) = M$, the projection on $I_{\boldsymbol{e}}$ of every element of M, is the zero vector and conversely. □

Thanks to this lemma, if we find a necessary condition for $M_{I_{\boldsymbol{e}}} = \{0\}$, then we find a necessary condition for $M = \mathcal{A}(I_{\boldsymbol{e}})$ and conversely. For this reason, we now study the object $M_{I_{\boldsymbol{e}}}$.

Theorem 2.1.8. *If Condition (2–PELP1) holds, we have $M_{I_{\boldsymbol{e}}} = \mathcal{A}_{I_{\boldsymbol{e}}} \cap (\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}} \cap \left(\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp\right)^\perp_{I_{\boldsymbol{e}}}$.*

Remark 2.1.9. Note that the notation $(\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}}$ and $\left(\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp\right)^\perp_{I_{\boldsymbol{e}}}$ could be considered as ambiguous since the operation of puncturing and that consisting in taking the dual code do not commute. Actually, in this situation, there is no ambiguity since the supports of $\boldsymbol{e}^{(1)} * B$ and $\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp$ are contained in $I_{\boldsymbol{e}}$.

Proof of Theorem 2.1.8. Let us characterize the elements of $\mathcal{A}$ that are in M_1 and M_2. Let $\boldsymbol{a} \in \mathcal{A}$,

$$\begin{aligned}
\boldsymbol{a} \in M_1 &\iff \langle \boldsymbol{a} * \boldsymbol{y}, \boldsymbol{b} \rangle = 0 \quad \forall \boldsymbol{b} \in \mathcal{B} \\
&\iff \langle \boldsymbol{a}, \boldsymbol{e}^{(1)} * \boldsymbol{b} \rangle = 0 \quad \forall \boldsymbol{b} \in \mathcal{B} \\
&\iff \boldsymbol{a}_{I_{\boldsymbol{e}}} \in (\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}}.
\end{aligned}$$

For the second equivalence we used the property $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp$ which is equivalent to $\mathcal{A} * \mathcal{C} \subseteq \mathcal{B}^\perp$ (see Remark 1.6.6) while in the third, we used $\mathrm{supp}(\boldsymbol{e}^{(1)}) = I_{\boldsymbol{e}}$. In the same way, it is possible to prove that given $\boldsymbol{a} \in \mathcal{A}$,

$$\boldsymbol{a} \in M_2 \iff \boldsymbol{a}_{I_{\boldsymbol{e}}} \in (\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp)^\perp_{I_{\boldsymbol{e}}}.$$

The result comes now easily. □

It is actually possible to simplify the form of $M_{I_{\boldsymbol{e}}}$.

Lemma 2.1.10. *Let $\mathcal{A} \subseteq \mathbb{F}_q^n$ be a linear code and $t \in \mathbb{N}$. The following facts are equivalent:*

- $\mathrm{d}(\mathcal{A}^\perp) > t$;
- $\mathcal{A}_J = \mathbb{F}_q^t$ *for any* $J \subseteq \{1, \ldots, n\}$ *with* $|J| = t$.

Corollary 2.1.11. *Let $(\mathcal{A}, \mathcal{B})$ be a pair of codes satisfying Condition (2–PELP1). If $\mathrm{d}(\mathcal{A}^\perp) > t$, then*

$$M_{I_{\boldsymbol{e}}} = (\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}} \cap (\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp)^\perp_{I_{\boldsymbol{e}}}.$$

It is now possible to prove Theorem 2.1.6.

Proof Theorem 2.1.6. Looking for a necessary condition to have $M_{I_{\boldsymbol{e}}} = \{0\}$, we get

$$\begin{aligned}
M_{I_{\boldsymbol{e}}} = \{0\} &\iff (\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}} \cap (\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp)^\perp_{I_{\boldsymbol{e}}} = \{0\} && (2.4) \\
&\implies \dim((\boldsymbol{e}^{(1)} * \mathcal{B})^\perp_{I_{\boldsymbol{e}}}) + \dim((\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp)^\perp_{I_{\boldsymbol{e}}}) \leq t && (2.5) \\
&\implies \dim(\boldsymbol{e}^{(1)} * \mathcal{B})_{I_{\boldsymbol{e}}} + \dim(\boldsymbol{e}^{(2)} * (\mathcal{B}^\perp * \mathcal{C})^\perp)_{I_{\boldsymbol{e}}} \geq t && (2.6) \\
&\implies \dim(\mathcal{B}) + \dim((\mathcal{B}^\perp * \mathcal{C})^\perp) \geq t. && (2.7)
\end{aligned}$$

□

2.1.2 The general case: $\ell \geq 2$

It is possible to generalize the algorithm for $\ell \geq 2$. First we generalize the structure we use.

Definition 2.1.12 (ℓ–Power error locating pairs)**.** Given a linear code $\mathcal{C} \subseteq \mathbb{F}_q^n$, a pair of linear codes $(\mathcal{A}, \mathcal{B})$ with $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}_q^n$ is an *ℓ–power t–error locating pair* for $\mathcal{C}$ if

(ℓ–PELP1) $\mathcal{A} * \mathcal{B} \subseteq \mathcal{C}^\perp$;

(ℓ–PELP2) $\dim(\mathcal{A}) > t$;

(ℓ–PELP3) $\mathrm{d}(\mathcal{A}^\perp) > t$;

(ℓ–PELP4) $\dim(\mathcal{B}) + \sum_{i=2}^{\ell} \dim(\mathcal{B}^\perp * \mathcal{C}^{i-1})^\perp \geq t$;

(ℓ–PELP5) $\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) > n$;.

A generalization of the notion of M is needed too, that is, for a generic power ℓ, M will be the intersection of ℓ spaces.

Definition 2.1.13. As in the ECP algorithm, let $M_1 = \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^\perp\}$. For any $i = 2, \ldots, \ell$, we define the space

$$M_i \stackrel{\text{def}}{=} \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y}^i \in \mathcal{B}^\perp * \mathcal{C}^{i-1}\}.$$

Finally we define

$$M \stackrel{\text{def}}{=} \bigcap_{i=1}^{\ell} M_i. \tag{2.8}$$

Then, the algorithm for a general ℓ is the same as Algorithm 5 changing only the definition of M by that given in (2.8). Let us look for a necessary condition for this generalized algorithm to return $\boldsymbol{c}$. It can be proven that Theorem 2.1.5 can be adapted to the generalized notion (2.8) of M. The following theorem gives the necessary condition we look for.

Theorem 2.1.14. *If $\mathcal{A}(I_{\boldsymbol{e}}) = M$, then* $\dim(\mathcal{B}) + \sum_{i=2}^{\ell} \dim(\mathcal{B}^\perp * \mathcal{C}^{i-1})^\perp \geq t$.

Again, in order to prove this Theorem, we study the condition $M_{I_{\boldsymbol{e}}} = \{0\}$, since it is equivalent to $\mathcal{A}(I_{\boldsymbol{e}}) = M$ by Lemma 2.1.7.

Theorem 2.1.15. *We have*

$$M_{I_{\boldsymbol{e}}} = \bigcap_{i=1}^{\ell} \left(\boldsymbol{e}^{(i)} * (\mathcal{B}^\perp * \mathcal{C}^{i-1})^\perp\right)_{I_{\boldsymbol{e}}}^\perp,$$

where $\boldsymbol{e}^{(i)} = \sum_{h=1}^{i} \binom{i}{h} \boldsymbol{c}^{i-h} * \boldsymbol{e}^h$ is such that $\boldsymbol{y}^i = \boldsymbol{c}^i + \boldsymbol{e}^{(i)}$ for all $i = 1, \ldots, \ell$.

To prove this result, it is possible to adapt the proof of Theorem 2.1.8 and observe that it still holds $\mathcal{A}_{I_{\boldsymbol{e}}} = \mathbb{F}_q^t$. We will use the following remark.

Remark 2.1.16. Given a vector space V with $\dim(V) = t$ and $A_1, \ldots, A_n \subseteq V$, we have

$$\dim\Big(\bigcap_{i=1}^{n} A_i^\perp\Big) \geq t - \sum \dim(A_i);$$

in addition, it is easy to see that $\dim((\boldsymbol{e}^{(1)} * B)_{I_{\boldsymbol{e}}}) \leq \dim(B)$ and

$$\forall i \in \{2, \ldots, \ell\}, \quad \dim((\boldsymbol{e}^{(i)} * (B^\perp * C^{i-1})^\perp)_{I_{\boldsymbol{e}}}) \leq \dim((B^\perp * C^{i-1})^\perp).$$

Now, it is possible to prove Theorem 2.1.14.

Proof Theorem 2.1.14. It holds

$$M_{I_{\boldsymbol{e}}} = \{0\} \iff (\boldsymbol{e}^{(1)} * \mathcal{B})^{\perp}_{I_{\boldsymbol{e}}} \cap \bigcap_{i=2}^{\ell} \left(\boldsymbol{e}^{(i)} * (\mathcal{B}^{\perp} * \mathcal{C}^{i-1})^{\perp}\right)^{\perp}_{I_{\boldsymbol{e}}} = \{0\}$$

$$\implies \dim((\boldsymbol{e}^{(1)} * \mathcal{B})^{\perp}_{I_{\boldsymbol{e}}}) + \dim\left(\bigcap_{i=2}^{\ell} (\boldsymbol{e}^{(i)} * (\mathcal{B}^{\perp} * \mathcal{C}^{i-1})^{\perp})^{\perp}_{I_{\boldsymbol{e}}}\right) \leq t. \tag{2.9}$$

Now, thanks to Remark 2.1.16, one can easily see that (2.9) implies

$$\dim(\mathcal{B}) + \sum_{i=2}^{\ell} \dim(\mathcal{B}^{\perp} * \mathcal{C}^{i-1})^{\perp} \geq t.$$

□

2.1.3 Complexity

To conclude this section, let us discuss the complexity of the algorithm. We denote by ω the exponent of the complexity of matrix multiplications. First, recall that the computation of the star product of two codes of length n costs $O(n^{\omega+1})$ arithmetic operations in $\mathbb{F}_q$ using a deterministic algorithm and $O(n^{\omega})$ using a probabilistic algorithm (see for instance [COT17, § VI.A and D]).

The evaluation of the complexity of the power error locating pairs algorithm should be divided in two parts:

- the *pre-computation phase*, that should be done once for good and is independent from the error and the corrupted codeword;
- the *online phase*, which depends on the corrupted codeword.

The precomputation phase

This phase consists essentially in computing generator matrices for the codes $(\mathcal{B}^{\perp} * \mathcal{C}^{i-1})^{\perp}$ for $i \in \{1, \ldots, \ell\}$. Each new calculation consists in the computation of a $*$–product and a dual. This yields an overall cost of $O(\ell n^{\omega})$ operations in $\mathbb{F}_q$ using a probabilistic algorithm and $O(\ell n^{\omega+1})$ operations using a deterministic one.

The online phase

- The computation of each space M_i boils down to the resolution of a linear system with $\dim \mathcal{A}$ variables and $\dim (\mathcal{B}^{\perp} * \mathcal{C}^{i-1})^{\perp}$ equations. Hence, a cost of $O(n^{\omega})$ operations in $\mathbb{F}_q$.
- The computation of M consists in the calculation of $\ell - 1$ intersections of spaces. Since the cost of the calculation of an intersection is $O(n^{\omega})$ operations, the cost of the computation of M from the knowledge of the M_i's is in $O(\ell n^{\omega})$

In summary, the overall complexity of the online phase is in $O(\ell n^{\omega})$ operations in $\mathbb{F}_q$.

Remark 2.1.17. Note that the previous complexity analysis is purely generic and does not take into account that codes with an error locating pair such as Reed–Solomon code may be described by structured matrices permitting to perform fast linear algebra.

2.2 ℓ–PELP algorithm for Reed–Solomon codes

We now give some applications of the ℓ–PELP algorithm, starting with Reed–Solomon codes. For these codes, the algorithm is much more intuitive. Indeed, as for the error correcting pairs algorithm, it is possible to deduce the PELP algorithm from a former decoding algorithm for Reed–Solomon codes: the

power decoding. Let us consider the code $\mathcal{C} = \mathbf{RS}_q[k]$ and the pair $(\mathcal{A}, \mathcal{B})$, where $\mathcal{A} = \mathbf{RS}_q[t+1]$ and $\mathcal{B}^\perp = \mathbf{RS}_q[t+k]$. We look for the values of t for which $(\mathcal{A}, \mathcal{B})$ is an ℓ–power t–error locating pair for $\mathcal{C}$. One can see that, since we no longer ask for $\mathrm{d}(\mathcal{B}^\perp) > t$, by Lemma 1.6.16, t can be larger than $\frac{\mathrm{d}(\mathcal{C})-1}{2}$. About the conditions to fulfill, we already have seen that properties (ℓ–PELP1, 2, 5) hold for any $t < \mathrm{d}(\mathcal{C})$ (see Proposition 1.6.17). Let us find the values of t which verify

(ℓ–PELP3) $\mathrm{d}(\mathcal{A}^\perp) > t$;

(ℓ–PELP4) $\dim(\mathcal{B}) + \sum_{i=2}^{\ell} \dim(\mathcal{B}^\perp * \mathcal{C}^{i-1})^\perp \geq t$.

Property (ℓ–PELP3) holds for any t since Reed–Solomon codes are MDS and $\mathcal{A}$ has dimension $t+1$. Let us now focus on property (ℓ–PELP4). By Proposition 1.5.1, we know that $\mathcal{B}^\perp * \mathcal{C}^{i-1} = \mathbf{RS}_q[t+i(k-1)+1]$ and all these codes are not equal to $\mathbb{F}_q^n$ as soon as

$$t < n - \ell(k-1) - 1. \tag{2.10}$$

If (2.10) is satisfied, then the bound in property (ℓ–PELP4) yields

$$t \leq \frac{2n\ell - k\ell(\ell+1) + \ell(\ell-1)}{2(\ell+1)}, \tag{2.11}$$

which is the decoding radius for the power decoding algorithm for a general ℓ (see (1.21)).

Remark 2.2.1. Note that (2.11) came in power decoding as a necessary condition to have a unique solution for a linear system. Here instead, it comes up as a necessary condition for an intersection of some vector spaces to be $\{\mathbf{0}\}$.

2.2.1 The space M and the key equations of power decoding

In § 1.6.2, we have seen that it is possible to relate the definition of M_1 with the key equations of Welch–Berlekamp algorithm. One can do the same with the definition of M in the power error locating pairs algorithm and the key equations of the ℓ–power decoding algorithm. Here, we only consider the case $\ell = 2$, since it is easy to generalize the idea for a larger ℓ. It is possible to write (1.19) in this way

$$\begin{cases} \mathrm{ev}_{\boldsymbol{x}}(\Lambda) * \boldsymbol{y} &= \mathrm{ev}_{\boldsymbol{x}}(N_1) \\ \mathrm{ev}_{\boldsymbol{x}}(\Lambda) * \boldsymbol{y}^2 &= \mathrm{ev}_{\boldsymbol{x}}(N_2), \end{cases}$$

where $\mathrm{ev}_{\boldsymbol{x}}$ is the evaluation map introduced in (1.3). Hence, we can deduce

- $\mathrm{ev}_{\boldsymbol{x}}(N_1) \in \mathbf{RS}_q[t+k] = \mathcal{B}^\perp$;
- $\mathrm{ev}_{\boldsymbol{x}}(N_2) \in \mathbf{RS}_q[t+2k-1] = \mathcal{B}^\perp * C$;
- $\mathrm{ev}_{\boldsymbol{x}}(\Lambda) \in \mathbf{RS}_q[t+1](I_{\boldsymbol{e}}) = \mathcal{A}(I_{\boldsymbol{e}})$;
- $\mathrm{ev}_{\boldsymbol{x}}(\Lambda) \in M$, where M is the set defined in the 2–power error locating pairs algorithm. Indeed we recall that $M = M_1 \cap M_2$, where

$$\begin{aligned} M_1 &= \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^\perp\}, \\ M_2 &= \{\boldsymbol{a} \in \mathcal{A} \mid \boldsymbol{a} * \boldsymbol{y}^2 \in \mathcal{B}^\perp * \mathcal{C}\}. \end{aligned}$$

In other words, in the power decoding algorithm one works with polynomials, while in the power error locating pairs algorithm one works with their evaluations.

2.2.2 Equivalence of the two algorithms for Reed–Solomon codes with $\ell = 2$

Thanks to the link presented in the previous subsection, it is possible to find an isomorphism between the solution space of power decoding and the space M. For the sake of simplicity, we prove this result in the case $\ell = 2$. The proof in the general case is easy to deduce at the cost of heavier notation.

Theorem 2.2.2. *Let $\boldsymbol{y} = \mathbb{F}_q^n$ and t a positive integer and suppose we run both the power decoding algorithm and the power error locating pairs algorithm with the same t and $\ell = 2$. Denote by Sol the solution space of the linear system in Problem 4 for $\ell = 2$. Then the linear map*

$$\varphi \stackrel{def}{=} \begin{cases} Sol & \longrightarrow & M \\ (\lambda, \nu_1, \nu_2) & \longmapsto & \mathrm{ev}_{\boldsymbol{x}}(\lambda). \end{cases} \tag{2.12}$$

is bijective.

Proof. First, let us show that φ is well defined. Let (λ, ν_1, ν_2) belong to Sol. Then, it holds

$$\begin{cases} \lambda(x_i) y_i = \nu_1(x_i) & i = 1, \ldots, n \\ \lambda(x_i) y_i^2 = \nu_2(x_i) & i = 1, \ldots, n. \end{cases} \tag{2.13}$$

As we have seen in § 2.2, these two conditions are equivalent to the statement

$$\mathrm{ev}_{\boldsymbol{x}}(\lambda) \in M_1 \cap M_2 = M.$$

Conversely, given $\boldsymbol{a} \in M$, there exists $\lambda \in \mathbb{F}_q[X]$ with $\deg(\lambda) < t + 1$ such that $\mathrm{ev}_{\boldsymbol{x}}(\lambda) = \boldsymbol{a}$. Moreover, since $\boldsymbol{a} \in M$, we have

$$\boldsymbol{a} * \boldsymbol{y} \in \mathcal{B}^{\perp} = \mathbf{RS}_q[t + k], \quad \boldsymbol{a} * \boldsymbol{y}^2 \in \mathcal{B}^{\perp} * \mathcal{C} = \mathbf{RS}_q[t + 2k - 1].$$

Thus, there exist $\nu_1, \nu_2 \in \mathbb{F}_q[X]$ with $\deg(\nu_1) < t + k$, $\deg(\nu_2) < t + 2k - 1$ and

$$\mathrm{ev}_{\boldsymbol{x}}(\nu_1) = \boldsymbol{a} * \boldsymbol{y}, \quad \mathrm{ev}_{\boldsymbol{x}}(\nu_2) = \boldsymbol{a} * \boldsymbol{y}^2. \tag{2.14}$$

We can then define another map

$$\psi \stackrel{\mathrm{def}}{=} \begin{cases} M & \longrightarrow & Sol \\ \boldsymbol{a} & \longmapsto & (\lambda, \nu_1, \nu_2) \end{cases}, \tag{2.15}$$

where λ, ν_1 and ν_2 are the polynomials associated to $\boldsymbol{a}$ as before. It is easy to prove that, under condition[1]

$$t < n - 2(k - 1) \tag{2.16}$$

it holds $\varphi \circ \psi = Id_M$ and $\psi \circ \varphi = Id_{Sol}$. □

In summary, for Reed–Solomon codes, power decoding and power error locating pairs algorithms are equivalent. In particular they succeed or fail for the same instances.

2.3 PELP algorithm for algebraic geometry codes

As said previously, the power error locating pairs algorithm can be run on any code with a PELP. We have seen that Reed–Solomon codes belong to this class of codes. In the sequel, we show that algebraic geometry codes also belong to it. Similarly to the case of Reed–Solomon codes, this algorithm can be compared with the power decoding algorithm. In particular, we show that the analysis of the power decoding provides a decoding radius which is equivalent to the one of the power error locating pairs algorithm.

[1] That is again bound (1.20).

2.3.1 Context

Let $\mathcal{X}$ be a smooth absolutely irreducible projective curve of genus g over $\mathbb{F}_q$. Let G be a divisor on $\mathcal{X}$ and $\mathcal{P} = (P_1, \ldots, P_n)$ be an ordered n–tuple of pairwise distinct rational points of $\mathcal{X}$ avoiding the support of G. We denote by k and d respectively the dimension and the minimum distance of the code $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$. Moreover, we denote by D the divisor $P_1 + \cdots + P_n$ and by W the divisor (ω) where $\omega \in \Omega(\mathcal{X})$ is a differential form such that $v_{P_i}(\omega) = -1$ and $res_{P_i}(\omega) = 1$ for any $i \in \{1, \ldots, n\}$ (see Proposition 1.7.56). We now introduce an extra divisor F on $\mathcal{X}$ and the pair $(\mathcal{A}, \mathcal{B})$ with

$$\mathcal{A} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, F) \quad \mathcal{B} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, D + W - F - G). \tag{2.17}$$

This pair of codes is our candidate to be a power error locating pair for $\mathcal{C}$. We analyze the case $\ell = 2$ for simplicity (it is easy to generalize to arbitrary $\ell \geq 2$).

2.3.2 Decoding Radius

In order to find the decoding radius of the 2–power error locating pairs algorithm for algebraic geometry codes, we follow the same path as for Reed–Solomon codes. That is, we look for the pairs $(\mathcal{A}, \mathcal{B})$ as in (2.17) which satisfy properties (2–PELP1–4) in Definition 2.1.1. One can easily see that $\mathcal{B}^\perp * \mathcal{C}$ is included in $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, F + 2G)$. Thus, we get

$$(\mathcal{B}^\perp * \mathcal{C})^\perp \supseteq \mathcal{C}_L(\mathcal{X}, \mathcal{P}, D + W - F - 2G). \tag{2.18}$$

We have then the following result.

Proposition 2.3.1. *Let* $\deg F = t + 2g$ *and* $2g - 2 < \deg G$ *with* $\ell(F + 2G) < n$. *Then* $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ *admits a* 2*–PELP as in* (2.17) *if*

$$t \leq \frac{2n - 3\deg G - 2}{3} - \frac{2}{3}g. \tag{2.19}$$

In this case, bound (2.19) gives the decoding radius of the 2*–PELP algorithm.*

Proof. Condition (2–PELP1) is obviously satisfied by the codes $\mathcal{A}, \mathcal{B}$ defined in (2.17). Moreover, since $\deg(F) = t + 2g$, we get by Riemann–Roch theorem $\dim(\mathcal{A}) > t$ and $\mathrm{d}(\mathcal{A}^\perp) > t$ i.e. Property (2–PELP2-3). The hypothesis $\ell(F + 2G) < n$ implies $\deg(F + G) < n$. Hence,

$$\mathrm{d}(\mathcal{A}) + \mathrm{d}(\mathcal{C}) \geq 2n - \deg F - \deg G > n,$$

that is, Property (2–PELP5) is verified. Finally one notes that, by the hypotheses $\ell(F + 2G) < n$ and $\deg F = t + 2g$, using (2.18), we have

$$\begin{aligned} \dim(\mathcal{B}) + \dim((\mathcal{B}^\perp * \mathcal{C})^\perp) &\geq 2n - 2\deg F - 3\deg G + 2g - 2 \\ &= 2n - 2t - 2g - 3\deg G - 2. \end{aligned}$$

Hence, if t verifies (2.19), Property (2–PELP4) holds. □

Remark 2.3.2. Note that, under the condition $\deg F = t + 2g$, the hypothesis $\ell(F + 2G) < n$ is verified for any $t \leq \frac{2n - 3\deg G - 2 - 2g}{3}$, whenever $\deg G < \frac{n - g - 1}{3}$. Since $\dim \mathcal{C} = \deg G - g + 1$, we get

$$\dim \mathcal{C} \leq \frac{n - 4g + 2}{3}.$$

The decoding radius can be computed for arbitrary values of ℓ. Indeed, imposing $t \leq n - \ell \deg(G) - g - 2$, we get

$$t \leq \frac{2n\ell - \ell(\ell + 1)\deg G - 2\ell}{2(\ell + 1)} - \frac{\ell g}{\ell + 1}. \tag{DR(g)}$$

2.3.3 Comparison with decoding radii of other algorithms for algebraic geometry codes

We can now compare this decoding radius with the decoding radii of Sudan algorithm and the power decoding algorithm for algebraic geometry codes. We have (see § 1.8.3 and § 1.8.2):

$$\begin{aligned}
&\textbf{Sudan} && t \leq \frac{2n\ell - \ell(\ell+1)\deg G - 2}{2(\ell+1)} - \frac{\ell g}{\ell+1} \\
&\textbf{Power decoding} && t \leq \frac{2n\ell - \ell(\ell+1)\deg G}{2(\ell+1)} - \frac{\ell}{\ell+1} \\
&\textbf{PELP algorithm} && t \leq \frac{2n\ell - \ell(\ell+1)\deg G - 2\ell}{2(\ell+1)} - \frac{\ell g}{\ell+1}.
\end{aligned}$$

It may seem that unlike for Reed–Solomon codes, for algebraic geometry codes, the power decoding algorithm and the power error locating pairs algorithm no longer have the same decoding radius, but the first one is larger. Though we point out that, while the decoding radius of the power decoding is computed for $\deg F = t + g$, for the one of the power error locating pairs we used $\deg F = t + 2g \implies \mathrm{d}(\mathcal{A}^{\perp}) > t$. Actually experimentally we have seen that it is possible to run the power error locating pairs algorithm with $\deg F = t + g$. For this value of the degree of F, Condition (ℓ-PELP5) holds whenever

$$t \leq \frac{2n\ell - \ell(\ell+1)\deg G}{2(\ell+1)} - \frac{\ell}{\ell+1}. \tag{DR}$$

Although in this case this bound does not represent a necessary condition for the algorithm to work, empirically it turns out to be an accurate decoding radius for the algorithm (see 2.1). Its correction capacity equals then the one of the power decoding algorithm and, hence, results larger than Sudan's.

Tests on the accuracy of the decoding radius

In this paragraph we expose some of the empirical experiences conducted on the power error locating pairs algorithm. The results of these tests are reported in Table 2.1 and are relative to the following two curves:

1. $X^6 - ZY^5 - YZ^5 = 0$ in $\mathbb{F}_{5^2}$;
2. $X^6 + Y^6 + XZ^5 = 0$ in $\mathbb{F}_{7^3}$.

Moreover, we point out that we do not consider only one-point codes, but also codes depending on a divisor G with distinct places in its support. We denote by DR(g) the decoding radius obtained with $\deg F = t + 2g$ and by DR the one obtained with $\deg F = t + g$. We underline the value of t whenever t is larger than the expected decoding radius. In the last column of the table, we indicate whether the algorithm has been able to recover $\boldsymbol{c}$ or not.

q	$\mathcal{X}$	g	n	$\deg G$	$\deg F$	$\frac{\mathrm{d}^*-1}{2}$	DR(g)	DR	t	finds $\boldsymbol{c}$
5^2	1	10	104	$2g$	$t+2g$	41	42	48	42	yes
5^2	1	10	104	$2g$	$t+2g$	41	42	48	$\underline{43}$	no
5^2	1	10	104	$2g$	$t+g$	41	42	48	48	yes
5^2	1	10	104	$2g$	$t+g$	41	42	48	$\underline{49}$	no
7^3	2	10	120	$2g-1$	$t+2g$	50	53	60	53	yes
7^3	2	10	120	$2g-1$	$t+2g$	50	53	60	$\underline{54}$	no
7^3	2	10	120	$2g-1$	$t+g$	50	53	60	60	yes
7^3	2	10	120	$2g-1$	$t+g$	50	53	60	$\underline{61}$	no

Table 2.1: PELP algorithm for algebraic geometry codes.

2.3.4 Cryptanalytic application

One application of the power error locating pairs algorithm can be found in cryptography, as this algorithm provides an attack for McEliece encryption scheme [McE78] with algebraic geometry codes. This scheme, introduced in the seventies, uses profoundly the coding and decoding processes exposed in Chapter 1. In particular, let us consider a family of triples $(\mathcal{C}_i, D_i, t_{max,i})$, where for any i, $\mathcal{C}_i$ is a linear code and D_i is a decoding algorithm for $\mathcal{C}$ with decoding radius equal to $t_{max,i}$. McEliece scheme can be described as follows:

Key generation: A triple $(\mathcal{C}, D, t_{max})$ if chosen. Then, given a generator matrix G for $\mathcal{C}$, the *public key* and *private key* are set respectively as

$$\mathcal{K}_{pub} = (G, t_{max}) \qquad \mathcal{K}_{priv} = D.$$

Encryption: Given n such that $\mathcal{C} \subseteq \mathbb{F}_q^n$, Alice encrypts the message $\boldsymbol{m}$ as $\boldsymbol{y} = \boldsymbol{m}G + \boldsymbol{e}$, where $\boldsymbol{e}$ is a random vector in $\mathbb{F}_q^n$ and $\mathrm{w}(\boldsymbol{e}) \leq t_{max}$.

Decryption: Bob recovers $\boldsymbol{m}$ using the decoding algorithm D.

In the last forty years, many attempts for instanciating McEliece encryption scheme [McE78] using algebraic codes have been proposed in the literature. The use of generalized Reed–Solomon codes is known to be unsecure since Sidelnikov and Shestakov's attack [SS92] permitting to recover the whole structure of such a code from the very knowledge of a generator matrix. This attack has been extended to algebraic geometry codes from curves of genus 1 and 2 [Min07, FM08]. For general algebraic geometry codes, an attack has been given [CMCP17] that permits to recover an error correcting pair or an *error correcting array* from the knowledge of a generating matrix. This attack does not permits to recover the curve, the divisor and the evaluation points but is enough to break the system as soon as the decoder corrects at most half the designed distance.

In a nutshell, this attack of [CMCP17] consists in computing some filtered sequences of codes from the knowledge of a generator matrix of $(\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G))^{\perp}$. Namely, the codes computed are of the form $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, iP)$ and $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G - iP)$ for a given rational point P and for any integer i. For i large enough, the pair $(\mathcal{C}_L(\mathcal{X}, \mathcal{P}, iP), \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G - iP))$ yields an error correcting pair.

Suppose now that McEliece scheme is instantiated with an algebraic geometry code and whose decryption step requires to correct beyond half the designed distance by using Sudan's or power decoding algorithm. *Stricto sensu*, such a scheme is out of reach by the attack [CMCP17]. However, the very same approach permits to design a power error locating pair. Then, Algorithm 5 can be run without requiring any further knowledge on the curve and the divisor. This yields an interesting application of this abstract formulation of decoding beyond half the minimum distance. Note that no such cryptographic proposal exists in the literature but [ZZ18], which is unfortunately out of reach of power error locating pairs since it requires the use of a Guruswami–Sudan like decoder yielding a decoding radius close to Johnson bound.

2.4 PELP algorithm for cyclic codes

In this section, we give an application of the PELP algorithm for some cyclic codes. In 1994, Duursma and Kötter showed in [DK94] that an ECP algorithm can correct up to half the BCH bound and, in particular cases, also half the Roos bound (see Theorem 2.4.9 for a definition and [Roo83] for details).

First, we recall the main notions and fix some notation (for more details see [DK94]). Let us consider a field $\mathbb{F}_q$ and an integer n with $\gcd(n, q) = 1$. Given a vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n$, we denote by $c(X)$ the image of $\boldsymbol{c}$ by the following linear map:

$$\begin{cases} \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q[X]/(X^n - 1) \\ (c_0, \ldots, c_{n-1}) & \longmapsto & \sum_{i=0}^{n-1} c_i X^i. \end{cases}$$

It is well-known that cyclic codes of length n over $\mathbb{F}_q$ are in correspondence with the factors of the polynomial $X^n - 1$. In particular, given $g(X)|X^n - 1$ in $\mathbb{F}_q[X]$, the cyclic code $\mathcal{C}_g$ associated to g is

$$\mathcal{C}_g \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathbb{F}_q^n \text{ such that } g(X)|c(X)\}.$$

In the same way, the roots of g determine in a unique way the code $\mathcal{C}_g$. Hence, let us consider an extension $\mathbb{F} \supseteq \mathbb{F}_q$ such that $\mathbb{F}$ contains the n–th roots of unity and let γ be a primitive n–th root of unity.

Definition 2.4.1. Given $R = \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$, we define the $m \times n$ matrix

$$M(R) \stackrel{\text{def}}{=} \begin{pmatrix} 1 & \gamma^{i_1} & \cdots & \gamma^{i_1(n-1)} \\ 1 & \gamma^{i_2} & \cdots & \gamma^{i_2(n-1)} \\ \vdots & \vdots & & \vdots \\ 1 & \gamma^{i_m} & \cdots & \gamma^{i_m(n-1)} \end{pmatrix}.$$

To any subset $R \subseteq \{1, \ldots, n\}$, one can then associate two cyclic codes.

Definition 2.4.2. R is called *defining set* for the code $\mathcal{C}$ if

$$\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}_q^n \mid M(R)\boldsymbol{c}^T = 0\}. \tag{2.20}$$

Remark 2.4.3. One can see that if $\mathcal{C}$ is defined as in (2.20), then $\mathcal{C}$ is a cyclic code. Indeed, we have $\mathcal{C} = \mathcal{C}_g$, where $g = \text{lcm}\{m_i(x) \mid i \in R\}$ and m_i is the minimal polynomial of γ^i on $\mathbb{F}_q$. Note that different defining sets can define the same cyclic code $\mathcal{C}$. By applying several times Frobenius morphism to the set $\{\gamma^i \mid i \in R\}$, one can find the maximal defining set, also called *complete*. In the sequel we will treat a general situation, where a defining set will not necessarily be complete.

Remark 2.4.4. Note that, if R is a defining set for a code $\mathcal{C}$, then $\mathcal{C} = \tilde{\mathcal{C}} \cap \mathbb{F}_q^n$, where $\tilde{\mathcal{C}} \subseteq \mathbb{F}^n$ is a cyclic code with parity check matrix $M(R)$. If we denote by d_R the minimum distance of the code $\tilde{\mathcal{C}}$, we get $\mathrm{d}(\mathcal{C}) \geq \mathrm{d}_R$.

Definition 2.4.5. R is called *generating set* for the code $\mathcal{C}$ if

$$\mathcal{C} = \{\boldsymbol{a}M(R) \mid \boldsymbol{a} \in \mathbb{F}^m\}. \tag{2.21}$$

We stress that if R is a generating set for a code $\mathcal{C}$, then $\mathcal{C}$ is a code with coefficients in the larger alphabet $\mathbb{F}$ and has generating matrix $M(R)$. In particular, $\dim_{\mathbb{F}}(\mathcal{C}) = |R|$.

Remark 2.4.6. Note that a code $\mathcal{C}$ as in (2.21) is a cyclic code. Indeed $\mathcal{C}$ is the dual code of the cyclic code $\mathcal{D} \subseteq \mathbb{F}^n$ with defining set R and it is well-known that the dual of a cyclic code is cyclic itself.

2.4.1 Roos bound

There are cases where it is possible to bound the minimum distance of a cyclic code. Apart from the BCH bound, another and more general bound has been given by Roos ([Roo83]).

Definition 2.4.7. Given $R \subseteq \{1, \ldots, n\}$, denote by $\overline{R}$ the smallest set made of consecutive indexes modulo n that contains R. Moreover, if S is another subset of $\{1, \ldots, n\}$, we can define the sum set

$$S + R \stackrel{\text{def}}{=} \{s + r \mod n \mid s \in S, r \in R\}.$$

Finally, given $a < n$, we define the set $aR \stackrel{\text{def}}{=} \{ar \mod n \mid r \in R\}$.

It is possible to relate the star product of two cyclic codes to the sum of their generating sets.

Proposition 2.4.8. *Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{W}$ be three cyclic codes with generating sets respectively S, R and $S + R$. Then,*

$$\mathcal{A} * \mathcal{B} = \mathcal{W}.$$

Proof. First, note that for any $j \in S + R$ we get by Definition 2.4.7

$$j = s + r \mod n$$

for some $s \in S$ and $r \in R$. Hence,

$$(1, \gamma^j, \ldots, \gamma^{j(n-1)}) = (1, \gamma^s, \ldots, \gamma^{s(n-1)}) * (1, \gamma^r, \ldots, \gamma^{r(n-1)}). \tag{2.22}$$

Now, it is easy to see that the set of generators of $\mathcal{A} * \mathcal{B}$

$$G \stackrel{\text{def}}{=} \{(1, \gamma^{s+r}, \ldots, \gamma^{(s+r)(n-1)}) \mid s \in S, r \in R\}$$

is equal to the set composed by the rows of the matrix $M(S + R)$ (see Definition 2.4.1). Since, by Definition 2.4.5, this is a generator matrix for the code $\mathcal{W}$, we get that G is a set of generators for both $\mathcal{A} * \mathcal{B}$ and $\mathcal{W}$, hence $\mathcal{A} * \mathcal{B} = \mathcal{W}$. □

Given a defining set R for a code $\mathcal{C}$, let d_R be the minimum distance of $\tilde{\mathcal{C}} \subseteq \mathbb{F}^n$ (we recall that $\mathcal{C} = \tilde{\mathcal{C}} \cap \mathbb{F}_q^n$).

Theorem 2.4.9 (Roos bound)**.** *Let $R, S \subseteq \{1, \ldots, n\}$ such that $|\overline{S}| \leq |S| + \mathrm{d}_R - 2$. Then,*

$$\mathrm{d}_{R+S} \geq |S| + \mathrm{d}_R - 1.$$

Proof. See [Roo83]. □

Remark 2.4.10. Under the hypotheses of Theorem 2.4.9, if $\mathcal{C}$ is the cyclic code with defining set $R + S$, since $\mathrm{d}(\mathcal{C}) \geq \mathrm{d}_{R+S}$, then $\mathrm{d}(\mathcal{C}) \geq |S| + \mathrm{d}_R - 1$ as well.

Remark 2.4.11. One can note that in the same hypotheses as Theorem 2.4.9, the proof given in [Roo83] can be adapted to prove that

$$\mathrm{d}_{aR+bS} \geq |S| + \mathrm{d}_R - 1$$

for any $a, b < n$ with $\gcd(a, n) = \gcd(b, n) = 1$.

2.4.2 ℓ–PELP algorithm and Roos bound

We now focus on cyclic codes with defining set $R + S$ with R and S satisfying the hypotheses of Roos bound (Theorem 2.4.9). Actually, we will work with the code in $\mathbb{F}^n$ for the sake of simplicity.

Theorem 2.4.12. *Let $a, b < n$ with $\gcd(a, n) = \gcd(b, n) = 1$ and let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{F}^n$ be cyclic codes with generating sets respectively aS and bR, where*

$$|\overline{S}| \leq |S| + \mathrm{d}_R - 2, \qquad |S| > t, \qquad \mathrm{d}_S > t.$$

Let $\tilde{\mathcal{C}} \subseteq \mathbb{F}^n$ be the cyclic code with parity check matrix $M(aS + bR)$ and $k = \dim(\tilde{\mathcal{C}})$. Let us suppose that

(i) for any $i \in \{1, \ldots, \ell - 1\}$ we have $\mathcal{B}^\perp * \tilde{\mathcal{C}}^i \subsetneq \mathbb{F}_q^n$;

(ii) any nonzero cyclic subcode of $\mathcal{B}$ is non degenerated (see §1.4.2).

Then $(\mathcal{A}, \mathcal{B})$ is an ℓ–power t–error locating pair for the code $\tilde{\mathcal{C}}$ with

$$t \leq \ell n - \Big[\frac{\ell(\ell+1)}{2}(k-1) + \ell(|S| + \delta) + \sum_{i=1}^{\ell-1} \gamma_i\Big], \tag{2.23}$$

where δ and $\gamma_1, \ldots, \gamma_{\ell-1}$ fulfill

$$\begin{aligned} n - k &= |S| + |R| - 1 + \delta \\ \dim(\mathcal{B}) &= \dim((\mathcal{B}^\perp * \tilde{\mathcal{C}}^i)^\perp) + i \dim(\tilde{\mathcal{C}}) - i + \gamma_i \quad \forall i = 1, \ldots, \ell - 1. \end{aligned} \tag{2.24}$$

Remark 2.4.13. Note that if $(\mathcal{A}, \mathcal{B})$ is an ℓ–power t–error locating pair for $\tilde{\mathcal{C}}$, then it is an ℓ–power t–error locating pair for the cyclic code $\mathcal{C}$ with defining set $aS + bR$ as well. Actually, providing a decoding structure for $\tilde{\mathcal{C}}$ in order to have one for $\mathcal{C}$, is a standard procedure for cyclic codes (see for instance [DK94]). In particular, that is why if $\tilde{\mathcal{C}}$ is a Reed–Solomon code, the optimized choice of PELP for $\tilde{\mathcal{C}}$ with $|S| = t + 1$, will give the decoding radius found in § 2.2.

Remark 2.4.14. Condition (ii) on $\mathcal{B}$ can be reformulated as follows: for any non empty subset U of R, there does not exist $i \in \mathbb{Z}/n\mathbb{Z}$ such that $U + i \equiv U \mod n$.

Before proving Theorem 2.4.12, we need the two following lemmas on the notion of *degenerated* codes (see § 1.4.2).

Lemma 2.4.15. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a degenerated code. Then for any code $\mathcal{D} \subseteq \mathbb{F}_q^n$, the code $\mathcal{C} * \mathcal{D}$ is degenerated too.*

Proof. It suffices to observe that $\mathrm{Stab}(\mathcal{C}) \subseteq \mathrm{Stab}(\mathcal{C} * \mathcal{D})$. □

Lemma 2.4.16. *A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is degenerated if and only if $\mathcal{C}^\perp$ is degenerated.*

Proof. Using the adjunction property (1.2) of the star product, one proves that $\mathrm{Stab}(\mathcal{C}) = \mathrm{Stab}(\mathcal{C}^\perp)$. □

Proof of Theorem 2.4.12. We treat the case $a = b = 1$, the general case being an easy generalization. We have by hypothesis $\dim(\mathcal{A}) = |S| > t$. Next, from Proposition 2.4.8, $\mathcal{A} * \mathcal{B} = \tilde{\mathcal{C}}^\perp$. Furthermore, we have $\mathrm{d}(\mathcal{A}^\perp) = \mathrm{d}_S > t$. Hence, properties ($\ell$–PELP1), ($\ell$–PELP2) and ($\ell$–PELP3) are satisfied.
Now, let us focus on property (ℓ–PELP5). We have that $\mathcal{A}$ is contained in the code with generating set $\overline{S}$, whose distance is $n - |\overline{S}| + 1$ (note that it is a generalized Reed–Solomon code). Hence, we get $\mathrm{d}(\mathcal{A}) \geq n - |\overline{S}| + 1$, which, together with Roos bound, gives

$$\mathrm{d}(\mathcal{A}) + \mathrm{d}(\tilde{\mathcal{C}}) \geq n - |\overline{S}| + |S| + \mathrm{d}_R \geq n + 2 > n.$$

We finally prove Property (ℓ–PELP4). Set $\mathcal{Z}_i \stackrel{\text{def}}{=} (\mathcal{B}^\perp * \tilde{\mathcal{C}}^i)^\perp$. Then,

$$\mathcal{Z}_i \perp \mathcal{B}^\perp * \tilde{\mathcal{C}}^i \iff \mathcal{Z}_i * \tilde{\mathcal{C}}^i \subseteq \mathcal{B}. \tag{2.25}$$

From Condition (ii) on $\mathcal{B}$, the code $\mathcal{Z}_i * \tilde{\mathcal{C}}$ is non degenerated. This last observation, together with inclusion (2.25) and Corollary 1.4.9 yield

$$\dim \mathcal{Z}_i + \dim \tilde{\mathcal{C}}^i - 1 + \gamma_i' = \dim \mathcal{B} \tag{2.26}$$

for some nonnegative integer γ_i'. Thus, using (2.26), we get

$$\begin{aligned}(\ell\text{–PELP4}):\ \dim(\mathcal{B}) + \sum_{i=1}^{\ell-1} \dim(\mathcal{B}^\perp * \tilde{\mathcal{C}}^i)^\perp \geq t \quad &\iff \quad \ell \dim(\mathcal{B}) - \frac{\ell(\ell-1)}{2}(\dim(\tilde{\mathcal{C}}) - 1) - \sum_{i=1}^{\ell-1} \gamma_i' \geq t \\ &\iff \quad \ell|R| - \frac{\ell(\ell-1)}{2}(k-1) - \sum_{i=1}^{\ell-1} \gamma_i' \geq t \end{aligned} \tag{2.27}$$

Next, since $\mathcal{Z}_i * \tilde{\mathcal{C}}^i$ is non degenerated, from Lemmas 2.4.15 and 2.4.16, the code $\tilde{\mathcal{C}}^\perp$ is non degenerated too. Therefore, since $\mathcal{A} * \mathcal{B} = \tilde{\mathcal{C}}^\perp$, using Corollary 1.4.9 again, we know that there exists $\delta \geq 0$ such that

$$|S| + |R| - 1 + \delta = n - k. \tag{2.28}$$

Hence, by (2.27) and (2.28), property (ℓ–PELP4) is equivalent to

$$t \leq \ell n - \Big[\frac{\ell(\ell+1)}{2}(k-1) + \ell(|S| + \delta) + \sum_{i=1}^{\ell-1} \gamma_i\Big].$$

□

Remark 2.4.17. Note that δ and γ do not depend only on the choice of R and S but also on the parameters a, b. Hence, in particular, the decoding radius depends as well on a, b.

2.4.3 Comparison with Roos bound

We now would like to compare the obtained decoding radius with the Roos bound. To do so, we consider a particular case of cyclic code. Let $R, S \subseteq \{1, \ldots, n\}$ such that $R = \{0, \ldots, r-1\} \cup \{r+u\}$, $S = \{0, \ldots, s-1\} \cup \{s+u\}$ with $|S| > t$ and $|\overline{S}| \leq |S| + \mathrm{d}_R - 2$. Let us denote by $t_{\mathcal{A},\mathcal{B}}$ the decoding radius (2.23) for $\ell = 2$ and by d_{Roos} the amount $|S| + \mathrm{d}_R - 1$.

The reader can find in Table 2.2 the results of several application of the power error locating pairs algorithm on codes with defining set $R + S$, where R and S are constructed as above. In particular, this construction permits to consider codes which are not BCH. We stress that the value of t equals in the first 4 lines of the table the decoding radius $t_{\mathcal{A},\mathcal{B}}$, while in the test of the last line, it exceeds it.

q	n	R	S	$\frac{\mathrm{d}_{Roos}-1}{2}$	$\frac{\mathrm{d}(\tilde{\mathcal{C}})-1}{2}$	t	recovers $\boldsymbol{c}$
2^8	51	$\{0, \ldots, 13\} \cup \{20\}$	$\{0 \ldots, 29\} \cup \{35\}$	21	25	30	yes
2^8	51	$\{0, \ldots, 13\} \cup \{20\}$	$\{0 \ldots, 27\} \cup \{34\}$	20	24	27	yes
2^{14}	43	$\{0, \ldots, 11\} \cup \{16\}$	$\{0 \ldots, 24\} \cup \{29\}$	18	20	24	yes
2^{12}	91	$\{0, \ldots, 26\} \cup \{32\}$	$\{0 \ldots, 39\} \cup \{45\}$	32	36	37	yes
2^{12}	91	$\{0, \ldots, 26\} \cup \{32\}$	$\{0 \ldots, 39\} \cup \{45\}$	32	36	38	no

Table 2.2: PELP algorithm for some cyclic codes.

We point out that the decoding radius $t_{\mathcal{A},\mathcal{B}}$ seems optimal. Indeed, the algorithm succeeds in recovering $\boldsymbol{c}$ whenever t is bounded by the decoding radius and it fails when it exceeds it (see the last line in Table 2.2). Note that, not only t exceeds half the Roos bound, but it can be larger than half the minimum distance of $\tilde{\mathcal{C}}$ itself.

Chapter 3

An Ehrhard-like list decoding algorithm for algebraic geometry codes with no genus penalty

In this chapter we will focus on the decoding problem for algebraic geometry codes and on the particular task of designing an algorithm with no genus penalty affecting its correction capacity. Pellikaan found a result of existence of such an algorithm [Pel89] for unique decoding and for a large enough ground field. Later on, Ehrhard finally found a constructive way of providing it [Ehr93], by generalizing Porter's algorithm [Por88, Ehr92]. Our results, presented in §3.3, follow the same path, but for decoding radii beyond the unique decoding bound: for a given parameter ℓ, we first prove the existence of a Sudan-like list decoding algorithm able to find ℓ solutions and with decoding radius equal to

$$t_{max}(\ell) = \frac{2\ell n - \ell(\ell+1)\deg G - 2}{2(\ell+1)}.$$

In particular, unlike Guruswami–Sudan algorithm, this algorithm does not need to use multiplicities to attain this recoding radius. In the last part of this chapter we provide then, an algorithm whose decoding radius equals empirically $t_{max}(\ell)$ for $\ell = 2$ and which recovers a list of solutions. For the sake of clarity, we first describe the results of Pellikaan and Ehrhard in §3.2. For that, we need to describe Porter's algorithm.

3.1 Porter's algorithm

In this subsection we will describe Porter's algorithm which, we recall, is equivalent to the basic algorithm for algebraic geometry codes (see [Ehr92, §3]). In particular, since it will be useful for what follows, we present the version of Porter's algorithm proposed by Ehrhard in [Ehr92]. We point out that in the original papers of Porter and Ehrhard, such as in most of the papers of that period, the used language is the one of differential forms. Indeed the approaches were mainly oriented to syndrome decoding, which would lead to consider more comfortable ways to describe the dual of the code, instead of the code itself. Here, we will instead use a more modern and intuitive point of view consisting in decoding functional codes.

Let us consider a code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) \subseteq \mathbb{F}_q^n$, where $g - 1 \leq \deg(G) < n$ and $\mathrm{supp}(G) \cap \mathcal{P} = \emptyset$. We recall that by Assumption 1 the received vector is of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{c} = \mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}})$ with $f_{\boldsymbol{c}} \in L(G)$ and $t = \mathrm{w}(\boldsymbol{e})$. In particular, since this is an algorithm for unique decoding, we suppose $t \leq \frac{\mathrm{d}-1}{2}$.

Foundations and purpose of the algorithm First, we would like to express all vectors in $\mathbb{F}_q^n$ as vectors of evaluations of certain functions. In order to do that, we introduce a divisor G' such that $\mathrm{supp}(G') \cap \mathcal{P} = \emptyset$, $G' \geq G$ and $\ell(W + D - G') = 0$, where W has been defined in (1.24). We get then the

inclusion

$$L(G) \subset L(G').$$

By using the hypothesis $\ell(W + D - G') = 0$, one can prove that $\mathrm{ev}_{\mathcal{P}} : L(G') \to \mathbb{F}_q^n$ is surjective. Then, since $\mathrm{ev}_{\mathcal{P}|L(G)} : L(G) \to \mathbb{F}_q^n$ is injective, there exists a vector space V such that $L(G) \subset V \subset L(G')$ and $\mathrm{ev}_{\mathcal{P}|V} : V \to \mathbb{F}_q^n$ is an isomorphism (see [Ehr92]). Hence, we get the following diagram:

$$\begin{array}{ccccc} L(G) & \hookrightarrow & V & \hookrightarrow & L(G') \\ \wr\downarrow \mathrm{ev}_{\mathcal{P}} & & \wr\downarrow \mathrm{ev}_{\mathcal{P}} & & \\ \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) & \hookrightarrow & \mathbb{F}_q^n & & \end{array}$$

We can now see the received vector $\boldsymbol{y}$ and the error vector $\boldsymbol{e}$ as vectors of the evaluation of two functions $f_{\boldsymbol{e}}, f_{\boldsymbol{y}} \in L(G')$. We denote by $D_{\boldsymbol{e}}$ the divisor such that $0 \leq D_{\boldsymbol{e}} \leq D$ and $P_i \in \mathrm{supp}(D_{\boldsymbol{e}})$ if and only if $i \in \mathrm{supp}(\boldsymbol{e})$ (i.e. $e_i \neq 0$). The aim of Porter's algorithm follows the one expressed in §1.8.1, that is to introduce an additional divisor F and try to compute the space

$$L(F - D_{\boldsymbol{e}}). \tag{3.1}$$

Indeed we recall that the space $L(F - D_{\boldsymbol{e}})$ is composed by all functions in $L(F)$ locating in some way the error positions. In particular, if $\mathrm{supp}(F) \cap \mathrm{supp}(D_{\boldsymbol{e}}) = \emptyset$, then $L(F - D_{\boldsymbol{e}})$ is composed by all functions in $L(F)$ which vanish at $\{P_i \mid i \in \mathrm{supp}(\boldsymbol{e})\}$. However, we will see soon that this last hypothesis on the support of F is not necessary for this specific algorithm to work and that, by adding a simple assumption on the degree of F, the very knowledge of an arbitrary nonzero $f \in L(F - D_{\boldsymbol{e}})$ makes possible to recover $f_{\boldsymbol{e}}$. First let us consider $\lambda \in L(F - D_{\boldsymbol{e}})$. We have

$$\lambda f_{\boldsymbol{y}} = \lambda f_{\boldsymbol{c}} + \lambda f_{\boldsymbol{e}},$$

where $\lambda f_{\boldsymbol{c}} \in L(F + G)$ and $\lambda f_{\boldsymbol{e}} \in L(F + G' - D)$. One can note that, if it is possible to isolate the second part, that is $\lambda f_{\boldsymbol{e}}$, then, by dividing by λ, we can recover $f_{\boldsymbol{e}}$. To do so, we want to add an assumption in order to have uniqueness of the decomposition of $\lambda f_{\boldsymbol{y}}$. Hence, we now introduce the following map

$$\begin{array}{ccc} L(F) & \longrightarrow & L(F + G') \\ \lambda & \longrightarrow & \lambda f_{\boldsymbol{y}}. \end{array}$$

Let us analyze the space $L(F + G')$. One can note that, by the hypothesis $G' \geq G$, we get $L(F + G)$, $L(F + G' - D) \subseteq L(F + G')$. Moreover, since $\mathrm{supp}(G') \cap \mathcal{P} = \emptyset$, we have

$$L(F + G) \cap L(F + G' - D) = L(F + G - D).$$

The assumption we need is then the following one.

Assumption 2. We assume $\deg(G + F) < n$.

Now, thanks to Assumption 2, we get $L(F+G-D) = \{0\}$ and, for a certain vector space $Z_1 \subset L(F+G')$,

$$L(F + G') = L(F + G) \oplus L(F + G' - D) \oplus Z_1. \tag{3.2}$$

Theorem 3.1.1. *The very knowledge of an arbitrary $\lambda \in L(F - D_{\boldsymbol{e}}) \setminus \{0\}$ makes possible to recover $f_{\boldsymbol{c}}$.*

Proof. Let us denote by π_0 the projection $L(F + G') \to L(F + G)$ with respect to the decomposition in (3.2). As said before, for any $\lambda \in L(F - D_{\boldsymbol{e}})$ we have $\lambda f_{\boldsymbol{c}} \in L(F + G)$ and $\lambda f_{\boldsymbol{e}} \in L(F + G' - D)$, that is

$$\lambda f_{\boldsymbol{y}} \in L(F + G) \oplus L(F + G' - D). \tag{3.3}$$

In particular, since the decomposition is unique, the equality $\lambda f_{\boldsymbol{c}} = \pi_0(\lambda f_{\boldsymbol{y}})$ holds. Therefore, if $\lambda \neq 0$, we can easily recover $f_{\boldsymbol{c}} = \frac{\pi_0(\lambda f_{\boldsymbol{y}})}{\lambda}$. □

Remark 3.1.2. By Theorem 3.1.1 we need to have $L(F-D_{\boldsymbol{e}}) \neq \{0\}$. In particular, since by Riemann–Roch theorem $\ell(F - D_{\boldsymbol{e}}) \geq \deg F - t - g + 1$, we decide to consider F such that

$$t + g \leq \deg F < n - \deg G. \tag{3.4}$$

Remark 3.1.3. One can note that Theorem 3.1.1 is an analog result for algebraic geometry codes of Lemma 1.6.4 for Welch-Berlekamp algorithm. Knowing that the error correcting pairs algorithm extends Welch-Berlekamp algorithm to algebraic geometry codes, a more detailed comparison with the error correcting pairs algorithm is given in Remark 3.1.9.

Remark 3.1.4. Note that Theorem 3.1.1 holds whenever λ belongs to a space different from $\{0\}$ of the form

$$L(F - D_{\boldsymbol{e}'}),$$

where $\mathrm{supp}(\boldsymbol{e}') \supseteq \mathrm{supp}(\boldsymbol{e})$. However, in order to have $L(F - D_{\boldsymbol{e}'}) \neq \{0\}$, the support of $\boldsymbol{e}'$ needs not to be too large. Indeed, let us consider the designed distance $\mathrm{d}^* = n - \deg G$ of the code and suppose $\mathrm{w}(\boldsymbol{e}') \geq \mathrm{d}^*$. By Assumption 2, we have

$$\deg(F - D_{\boldsymbol{e}'}) = \deg F - \mathrm{w}(\boldsymbol{e}') \leq \deg F - n + \deg G < 0.$$

Hence, by Proposition 1.7.36, $L(F-D_{\boldsymbol{e}'}) = \{0\}$. Anyway, in the rest of this section we will try to recover exactly $L(F - D_{\boldsymbol{e}})$.

The algorithm The problem now is to find a way to compute $L(F-D_{\boldsymbol{e}})$, without knowing the support of the error vector. Porter's idea, reformulated by Ehrhard in [Ehr93], is to consider the space

$$S(F) \stackrel{\text{def}}{=} \{\lambda \in L(F) \mid \lambda f_{\boldsymbol{y}} \in L(F + G) \oplus L(F + G' - D)\}, \tag{3.5}$$

which fulfills the inclusion $L(F - D_{\boldsymbol{e}}) \subseteq S(F)$ (see (3.3)), and to impose a condition on the divisor F to guarantee the equality $L(F - D_{\boldsymbol{e}}) = S(F)$. Indeed, once we have this equality, it is possible to compute $S(F)$, and hence $L(F - D_{\boldsymbol{e}})$, just by solving a linear system (see Remark 3.1.8). After that, by Theorem 3.1.1 we only need to pick a nonzero element in $S(F)$ and find $f_{\boldsymbol{e}}$. A sufficient condition for this equality to hold is given by the following results.

Proposition 3.1.5. *Let π_1 be the projection $L(F+G') \to L(F+G'-D)$ with respect to the decomposition in* (3.2)*. There is an exact sequence of vector spaces*

$$0 \longrightarrow L(F - D_{\boldsymbol{e}}) \xrightarrow{\ i\ } S(F) \xrightarrow{\ \Phi\ } L(G + F - D + D_{\boldsymbol{e}})$$

where for any $\lambda \in S(F)$, $\Phi(\lambda) = \lambda f_{\boldsymbol{e}} - \pi_1(\lambda f_{\boldsymbol{y}})$.

Proof. The proof can be found in [Ehr93], but we report it here for the sake of completeness. First we show that the map Φ is well-defined. Let $\lambda \in S(F)$. We know that $\lambda f_{\boldsymbol{y}} = \alpha + \beta$ where $\alpha \in L(F + G)$ and $\beta = \pi_1(\lambda f_{\boldsymbol{y}}) \in L(F + G' - D)$. On the other hand, we have $\lambda f_{\boldsymbol{y}} = \lambda f_{\boldsymbol{c}} + \lambda f_{\boldsymbol{e}}$. Thus, we get

$$\Phi(\lambda) = \lambda f_{\boldsymbol{e}} - \pi_1(\lambda f_{\boldsymbol{y}}) = \alpha - \lambda f_{\boldsymbol{c}}. \tag{3.6}$$

Hence, looking at the divisors induced by these functions, we obtain

$$\begin{aligned} (\alpha - \lambda f_{\boldsymbol{c}}) &\geq -F - G, \\ (\lambda f_{\boldsymbol{e}} - \pi_1(\lambda f_{\boldsymbol{y}})) &\geq \min\{-F - G' + D - D_{\boldsymbol{e}}, -F - G' + D\} = -F - G' + D - D_{\boldsymbol{e}}. \end{aligned}$$

Hence, by (3.6), since $G' \geq G$, we get

$$(\Phi(\lambda)) \geq \max\{-F - G' + D - D_{\boldsymbol{e}}, -F - G\} = -F - G + D - D_{\boldsymbol{e}},$$

that is, $\Phi(\lambda) \in L(F + G - D + D_{\boldsymbol{e}})$. Now, it is clear that i is injective and one can easily verify that Φ is linear. We focus then on the exactness of the sequence in $S(F)$. We have seen that if $\lambda \in L(F - D_{\boldsymbol{e}})$, then $\lambda f_{\boldsymbol{e}} \in L(F + G' - D)$, hence $i(L(F - D_{\boldsymbol{e}})) \subseteq \ker(\Phi)$. Conversely, suppose $\Phi(\lambda) = 0$, that is $\lambda f_{\boldsymbol{e}} = \pi_1(\lambda f_{\boldsymbol{y}}) \in L(F + G' - D)$. Then, for any $P \in \mathrm{supp}(D_{\boldsymbol{e}})$, since $f_{\boldsymbol{e}}(P) \neq 0$, we get

$$v_P(\lambda) = v_P(\lambda) + v_P(f_{\boldsymbol{e}}) = v_P(\lambda f_{\boldsymbol{e}}) = v_P(\pi_1(\lambda f_{\boldsymbol{y}})) \geq -v_P(F) + 1,$$

that is, $\lambda \in L(F - D_{\boldsymbol{e}})$. □

Corollary 3.1.6 ([Ehr92] Proposition 1)**.** *If* $\deg F + t < \mathrm{d}^*$*, then* $L(F - D_{\boldsymbol{e}}) = S(F)$*.*

Proof. Since $\deg F + t < \mathrm{d}^*$, we get $L(F+G-D+D_{\boldsymbol{e}}) = \{0\}$. Hence the inclusion i in the exact sequence is surjective. □

From Corollary 3.1.6, we can deduce the algorithm and its decoding radius.

Algorithm 6 Porter's algorithm - unique decoding

Inputs: $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_q^n$ where $\boldsymbol{c} \in C$ and $\mathrm{w}(\boldsymbol{e}) \leq \frac{d^*-1-g}{2}$, $t = \mathrm{w}(\boldsymbol{e})$, the projection π_0 with respect to the decomposition (3.2) $\pi_0 : L(F + G') \to L(F + G)$.
Output: $f_{\boldsymbol{c}} \in L(G)$ such that $\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}}) = \boldsymbol{c}$.

1: $f_{\boldsymbol{y}} \leftarrow$ function in $L(G')$ whose evaluation vector at the points of $\mathcal{P}$ equals $\boldsymbol{y}$
2: Choose F with $t + g \leq \deg F < \mathrm{d}^* - t$;
3: $\lambda \leftarrow$ nonzero element of $S(F)$;
4: $f_{\boldsymbol{c}} \leftarrow \frac{\pi_0(\lambda f_{\boldsymbol{y}})}{\lambda}$;
5: **return** $f_{\boldsymbol{c}}$;

Theorem 3.1.7. *The algorithm works whenever* $t \leq \frac{\mathrm{d}^*-1-g}{2}$*.*

Proof. Let us consider a divisor F which fulfills (3.4). We know that the algorithm works whenever $L(F - D_{\boldsymbol{e}}) \neq \{0\}$ and $S(F) = L(F - D_{\boldsymbol{e}})$. These properties are fulfilled if, by Remark 3.1.2 and Corollary 3.1.6,

$$t + g \leq \deg F < \mathrm{d}^* - t.$$

Therefore, it is easy to see that there exists a divisor F satisfying these two inequalities if $t \leq \frac{\mathrm{d}^*-1-g}{2}$. □

3.1.1 The key equation

We proposed here Porter's algorithm from a point of view which will be useful in the following sections. In [Por88] and [Ehr92] though, the same algorithm is presented as a *key equation* algorithm. We now briefly show how these two points of view are related. Using again the language of functions, in the original Porter's algorithm, one looks for $(\lambda, \mu, \alpha) \in (L(F) \setminus \{0\}) \times L(F + G) \times L(F + G' - D)$ such that the following key equation holds

$$\lambda f_{\boldsymbol{y}} = \mu + \alpha. \tag{3.7}$$

Of course, for any solution (λ, μ, α) to (3.7), λ is an element of $S(F)$. On the other hand, given $\lambda \in S(F)$, by Assumption 2 there exist unique $\mu \in L(F + G)$ and $\alpha \in L(F + G' - D)$ such that (3.7) holds.

Remark 3.1.8. We point out that in order to solve (3.7) or, equivalently, to compute the space $S(F)$, one can simply solve a linear system. Indeed it is easy to see that it is possible to construct in advance the projection $\tilde{\pi} : L(F + G') \longrightarrow Z_1$ (where Z_1 is defined in (3.2)) and that

$$S(F) = \{\lambda \in L(F) \mid \tilde{\pi}(\lambda f_{\boldsymbol{y}}) = 0\}.$$

Remark 3.1.9. We alluded before to the fact that the error correcting pairs algorithm was simply an extension of the basic algorithm to other codes. Here this relation becomes evident. Indeed, given (λ, μ, α) satisfying (3.7), if we evaluate the entire key equation in the points of $\mathcal{P}$, then α vanishes, while λ becomes clearly an element of M_1 (see (1.9)). The error correcting pairs then simplify this key equation in some way. Though the information given by α is completely lost. In the following section, we will see that, instead of following this flow of codewords language, Ehrhard comes back to the language of functions. In this way, he efficiently gets to use the information given by α and, by doing so, is able to erase the genus penalty from the decoding radius of the algorithm.

3.2 Erasing the genus penalty: unique decoding

We are now ready to present the results of Pellikaan [Pel89] and Ehrhard [Ehr93]. Before these two achievements, several attempts were made to get rid of the genus penalty from the correction capacity for unique decoding. Some of these attempts could increase the decoding radius, without achieving though half the designed distance of the code. It was not even sure that an algorithm with such correction capacity exists, until Pellikaan presented this existence result.

3.2.1 Some prerequisites

Let $\mathcal{X}$ be a curve over $\mathbb{F}_q$. We recall that $\mathrm{Princ}(\mathcal{X})$ is the set of principal divisors and that it is a subgroup of $\mathrm{Div}(\mathcal{X})$, hence it is abelian. We can then consider the quotient $\mathrm{Div}(\mathcal{X})/\mathrm{Princ}(\mathcal{X})$ also called *Picard group* of $\mathcal{X}$. Given a curve $\mathcal{X}$, we will focus in particular on the following subgroup of its Picard group.

Definition 3.2.1. The *Jacobian* of a curve $\mathcal{X}$ is defined as

$$J(\mathcal{X}) \stackrel{\text{def}}{=} \{D \in \mathrm{Div}(\mathcal{X}) \mid \deg(D) = 0\}/\mathrm{Princ}(\mathcal{X}).$$

We define now $N_1 \stackrel{\text{def}}{=} |\mathcal{X}(\mathbb{F}_q)|$, that is the number of rational points of $\mathcal{X}$. It is not always easy to compute the value of N_1. However, the following is a well known bound for every algebraic curve.

Proposition 3.2.2 (The Hasse–Weil bound)**.** *If $\mathcal{X}$ is a curve over $\mathbb{F}_q$ with genus g, then*

$$|N_1 - (q+1)| \leq 2g\sqrt{q}.$$

Proof. The reader can find the proof in [Sti09]. □

Definition 3.2.3. A curve $\mathcal{X}$ such that $N_1 - (q+1) = 2g\sqrt{q}$ is called *maximal*.

Now, we denote by $\mathbb{D}_k$ the set of effective divisors of degree k. The following proposition will be useful for Pellikaan's existence result. Its proof use several properties of the *Zeta function* of a curve. We will not introduce this object, in order to make the reading lighter. The reader anyway, if interested, can find more details in [Sti09, vLvdG88].

Proposition 3.2.4. *If $\mathcal{X}$ is a maximal curve, then*

$$|\mathbb{D}_{g-1}| \leq \frac{|J(\mathcal{X})|}{q-1}.$$

Proof. See [Pel89, Corollary 2, Remark 5]. □

3.2.2 Pellikaan's result: the existence of an algorithm with no genus penalty

We have seen in the previous section that the equality $L(F - D_{\boldsymbol{e}}) = S(F)$ is the one which makes Porter's algorithm work. To insure that, by Proposition 3.1.5, it suffices to have a divisor F such that

$$L(G + F - D + D_{\boldsymbol{e}}) = \{0\}. \tag{3.8}$$

In this section we present Pellikaan's result, that is, we show that it is possible to achieve the correction capacity of $\frac{\mathrm{d}^* - 1}{2}$. The aim is still the same, that is to guarantee (3.8). Anyway, to do so, since we want to increase the correction capacity of the algorithm, we can no longer use Corollary 3.1.6, as the bound

$$\deg F < \mathrm{d}^* - t,$$

is the one that induces the genus penalty on the decoding radius. In his paper, Pellikaan succeeds in proving with counting arguments that, for maximal curves (see Definition 3.2.3), there exists a divisor F such that

- the hypothesis $\deg F < \mathrm{d}^* - t$ is not fulfilled
- (3.8) is satisfied
- Porter's algorithm can correct amounts of errors up to half the designed distance of the code $\frac{\mathrm{d}^* - 1}{2}$.

We recall that $\mathbb{D}_k$ is the set of effective divisors of degree k and $J(\mathcal{X})$ is the Jacobian of the curve $\mathcal{X}$ (see Definition 3.2.1). For the moment we will suppose the designed distance of the code d^* to be odd, the case d^* even being treated later, in Remark 3.2.7. Now, the argument of Pellikaan works whenever $|\mathbb{D}_{g-1}| < |J(\mathcal{X})|$. This argument is exposed and proved in Theorem 3.2.6. First, we need to prove the following classical Lemma.

Lemma 3.2.5. *Let $u \geq g$ and F_0 be a divisor with $\deg F_0 = u$. Then the following map is surjective*

$$\begin{array}{rcl} \psi_u : \mathbb{D}_u & \longrightarrow & J(\mathcal{X}) \\ F & \longrightarrow & [F - F_0]. \end{array}$$

Proof. Let $[E] \in J(\mathcal{X})$. We show we can find an effective divisor $F \in \mathbb{D}_u$ such that

$$[F - F_0] = [E].$$

We have $\deg(F_0 + E) = \deg F_0 = u$, hence by Riemann Roch theorem and the hypothesis $u \geq g$

$$\ell(F_0 + E) \geq \deg(F_0 + E) - g + 1 \geq 1,$$

that is, there exists $0 \neq f \in \mathbb{F}_q(\mathcal{X})$ such that $(f) + F_0 + E \geq 0$. Defining then

$$F \stackrel{\text{def}}{=} (f) + F_0 + E,$$

we have $F \in \mathbb{D}_u$ and $\psi_u(F) = [E]$, hence the map ψ_u is surjective. □

Theorem 3.2.6. *Assume $|\mathbb{D}_{g-1}| < |J(\mathcal{X})|$ and $t = \frac{\mathrm{d}^* - 1}{2}$. Then, there exists a divisor F satisfying $\deg F = t + g$ and $L(F + G - D + D_{\boldsymbol{e}}) = \{0\}$.*

Proof. Let us suppose by way of contradiction that, for every divisor F with $\deg F = t + g$, we have

$$L(F + G - D + D_{\boldsymbol{e}}) \neq \{0\}, \tag{3.9}$$

and let us choose a certain divisor $F_0 \geq 0$ with $\deg F_0 = t + g$. Then, we know that there exists $0 \neq f \in L(F_0 + G - D + D_{\boldsymbol{e}})$ and we can define

$$R_0 \stackrel{\text{def}}{=} (f) + F_0 + G - D + D_{\boldsymbol{e}} \geq 0.$$

It is easy to verify that, since $t = \frac{\mathrm{d}^* - 1}{2}$ and d^* is odd, we have $\deg R_0 = g - 1$. Now, the map

$$\begin{array}{rcl} \psi_{g-1} : \mathbb{D}_{g-1} & \longrightarrow & J(\mathcal{X}) \\ R & \longrightarrow & [R - R_0] \end{array}$$

is not surjective since by hypothesis $|\mathbb{D}_{g-1}| < |J(\mathcal{X})|$. Hence, there exists $[E] \in J(\mathcal{X}) \setminus \mathrm{Im}(\psi_{g-1})$. On the other hand by Proposition 3.2.5, the map

$$\begin{array}{rcl} \psi_{t+g} : \mathbb{D}_{t+g} & \longrightarrow & J(\mathcal{X}) \\ F & \longrightarrow & [F - F_0] \end{array}$$

is surjective, hence there exists $F_1 \in \mathbb{D}_{t+g}$ such that $\psi_{t+g}(F_1) = [E]$. Since $\deg F_1 = \deg F_0 = t + g$, by hypothesis we have $L(F_1 + G - D + D_{\boldsymbol{e}}) \neq \{0\}$, that is, there exists $0 \neq h \in L(F_1 + G - D + D_{\boldsymbol{e}})$ and we can define

$$R_1 \stackrel{\text{def}}{=} (h) + F_1 + G - D + D_{\boldsymbol{e}} \geq 0.$$

Also, we have $\deg R_1 = \deg R_0 = g - 1$. Now, combining all these objects we obtain

$$[E] = \psi_{t+g}(F_1) = [F_1 - F_0] = [R_1 - R_0] = \psi_{g-1}(R_1),$$

which is a contradiction. □

Clearly then, by Proposition 3.2.4, if $\mathcal{X}$ is a maximal curve and $q \geq 3$, then there exists a divisor F for which it is possible to push the decoding radius up to $\frac{\mathrm{d}^* - 1}{2}$.

Remark 3.2.7. Note that if the divisor G is such that d^* is even, we need to have $|\mathbb{D}_{g-2}| < |J(\mathcal{X})|$. However, since $\mathcal{X}$ disposes of rational points, we have $|\mathbb{D}_{g-2}| \leq |\mathbb{D}_{g-1}|$. Hence, an estimation of $|\mathbb{D}_{g-1}|$ is enough.

Other cases

After Pellikaan, Vlădut [Vl0] proved that Theorem 3.2.6 can be applied to every algebraic curve with $q \geq 37$ or with a sufficiently large genus g and $q \geq 16$. We report here the two results, but we omit the proofs, which can be found in [Vl0].

Proposition 3.2.8. *Let $q \geq 37$. Then for any algebraic curve $\mathcal{X}$ over $\mathbb{F}_q$ one has*

$$|\mathbb{D}_{g-1}| \leq \frac{2|J(X)|}{q-1}.$$

Proposition 3.2.9. *Let $q \geq 16$. Then there exists $g_0 = g_0(q)$ such that for any curve $\mathcal{X}$ over $\mathbb{F}_q$ of genus $g \geq g_0$ with $N_1 \geq g$, one has*

$$|\mathbb{D}_{g-1}| \leq \frac{2|J(\mathcal{X})|}{q-1}.$$

Remark 3.2.10. Note that a bounded decoding algorithm correcting up to half the designed distance of the code $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G) \subseteq \mathbb{F}_q^n$ exists also if $q < 37$. Indeed, one can consider the code $\mathcal{C}_{q^m} = \mathcal{C} \otimes \mathbb{F}_{q^m}$ over an extension $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ such that $q^m > 37$. By Proposition 3.2.8 and Theorem 3.2.6, there exists a bounded decoding algorithm for $\mathcal{C}_{q^m}$ correcting up to $\frac{\mathrm{d}^*(\mathcal{C}_{q^m})-1}{2}$. Now, since the designed distance of a code does not depend on the ground field we are considering, we get $\mathrm{d}^*(\mathcal{C}) = \mathrm{d}^*(\mathcal{C}_{q^m})$. Hence, if there exists a solution $\boldsymbol{c} \in \mathcal{C}$ with $\mathrm{d}(\boldsymbol{c}, \boldsymbol{y}) \leq \frac{\mathrm{d}^*(\mathcal{C})-1}{2}$, then this algorithm recovers $\boldsymbol{c}$. The algorithm for $\mathcal{C}$ consists then in running the algorithm for $\mathcal{C}_{q^m}$ and, if a solution exists, in checking if it belongs to $\mathbb{F}_q^n$.

We conclude by recalling that with this result, Pellikaan proves only the existence of an algorithm with this correction capacity, without providing, though, a constructive method to find it. The first constructive result was finally found by Ehrhard and it is described in the following subsection.

3.2.3 Ehrhard's algorithm

Ehrhard's algorithm comes as a modification of Porter's algorithm and represents the first constructive method to erase the genus penalty from the decoding radius $\frac{\mathrm{d}^*-1-g}{2}$. Ehrhard's idea, like Pellikaan's, is to find a specific divisor F, avoiding the hypothesis of Corollary 3.1.6, $\deg F + t \leq \mathrm{d}^*$, such that the equality $L(F - D_{\boldsymbol{e}}) = S(F)$ holds anyway. To do so, let us consider at first a generic divisor F. The main result of Ehrhard's paper is the following.

Proposition 3.2.11. *Assume $L(F - D_{\boldsymbol{e}}) \neq \{0\}$ and $\deg(F) \leq \mathrm{d}^* - g - 1$. Then one and only one of the following statements holds:*

- *$S(F) = L(F - D_{\boldsymbol{e}})$;*
- *There exists a rational point $P \in \mathrm{supp}(D)$ with $\dim S(F - P) \leq \dim S(F) - 2$.*

We emphasize that there are no hypotheses on the support of F and that in particular the result remains true if $\mathrm{supp}(F) \cap \mathcal{P} \neq \emptyset$. Proposition 3.2.11 tells us that either we already have $S(F) = L(F - D_{\boldsymbol{e}})$, or we can construct a new divisor $F - P$ for some $P \in \mathrm{supp}(D)$, such that $\dim S(F - P)$ decreases quite fast with respect to $\dim S(F)$. Before proving Proposition 3.2.11 though, we need some more results. All proofs can be found in [Ehr93] but we report them here in the language of functions for the sake of completeness.

Lemma 3.2.12. *If $L(F - D_{\boldsymbol{e}}) \neq S(F)$, then there exist at most $\deg(F + D_{\boldsymbol{e}}) - \mathrm{d}^*$ rational points $P \in \mathrm{supp}(D_{\boldsymbol{e}})$ such that $S(F) \subseteq L(F - P)$.*

Proof. Without loss of generality, after reindexing, one can suppose that for a certain $m < n$, $P_1, \ldots, P_m$ are the points in $\mathrm{supp}(D_{\boldsymbol{e}})$ such that

$$S(F) \subseteq L(F - P_i) \quad \forall i = 1, \ldots, m.$$

In particular, if we define $\tilde{D} \stackrel{\mathrm{def}}{=} \sum_{i=1}^m P_i$, we have $S(F) \subseteq \bigcap_{i=1}^m L(F - P_i) = L(F - \tilde{D})$. By assumption, there exists

$$\lambda \in S(F) \setminus L(F - D_{\boldsymbol{e}}).$$

By Proposition 3.1.5, we have $\Phi(\lambda) = \lambda f_{\boldsymbol{e}} - \pi(\lambda f_{\boldsymbol{y}}) \neq 0$ where $\pi(\lambda f_{\boldsymbol{y}}) \in L(F + G' - D)$. We get then

$$(\lambda f_{\boldsymbol{e}} - \pi(\lambda f_{\boldsymbol{y}})) \geq \min(-F + \tilde{D} - G' + D - D_{\boldsymbol{e}}, -F - G' + D) = -G' - F + D - (D_{\boldsymbol{e}} - \tilde{D}),$$

since by assumption $\tilde{D} \leq D_{\boldsymbol{e}}$. By definition of Φ in Proposition 3.1.5, we get in particular

$$0 \neq \lambda f_{\boldsymbol{e}} - \pi(\lambda f_{\boldsymbol{y}}) \in L(G' + F - D + (D_{\boldsymbol{e}} - \tilde{D})) \cap L(G + F - D + D_{\boldsymbol{e}}) = L(G + F - D + (D_{\boldsymbol{e}} - \tilde{D})).$$

Hence, $\deg(G + F - D + (D_{\boldsymbol{e}} - \tilde{D})) \geq 0$, that is

$$m \leq \deg(F + D_{\boldsymbol{e}}) - \mathrm{d}^* .$$

□

Lemma 3.2.13. *If $L(F - D_{\boldsymbol{e}}) \neq \{0\}$, then there are at most g rational points $P \in \operatorname{supp}(D_{\boldsymbol{e}})$ such that $S(F - P) = S(F) \cap L(F - P)$.*

Proof. As a consequence of Riemann Roch theorem, there are at most g points P in $\operatorname{supp}(D_{\boldsymbol{e}})$ such that $L(F - D_{\boldsymbol{e}} - P) = L(F - D_{\boldsymbol{e}})$. Indeed again without loss of generality we can suppose that $P_1, \ldots, P_m$ are the points in $\operatorname{supp}(D_{\boldsymbol{e}})$ such that $L(F - D_{\boldsymbol{e}}) = L(F - D_{\boldsymbol{e}} - P_i)$ for all $i = 1, \ldots, m$. In particular we have

$$L(F - D_{\boldsymbol{e}}) = L\left(F - D_{\boldsymbol{e}} - \sum_{i=1}^{m} P_i\right).$$

Furthermore, by Proposition 1.7.50,

$$\ell(F - D_{\boldsymbol{e}}) \leq \ell(F - D_{\boldsymbol{e}}) - \ell\left(\sum_{i=1}^{m} P_i\right) + 1 \leq \ell(F - D_{\boldsymbol{e}}) - m + g - 1 + 1.$$

Hence, $m \leq g$. Now, it suffices to prove that given a point $P \in \operatorname{supp}(D_{\boldsymbol{e}})$

$$L(F - D_{\boldsymbol{e}} - P) \neq L(F - D_{\boldsymbol{e}}) \implies S(F - P) \neq S(F) \cap L(F - P).$$

Let us consider $\lambda \in L(F - D_{\boldsymbol{e}}) \setminus L(F - D_{\boldsymbol{e}} - P)$. In particular we have $\lambda \in L(F - D_{\boldsymbol{e}}) \subseteq S(F) \cap L(F - P)$. We now show that $\lambda \notin S(F - P)$. If it was, we would have $\lambda f_{\boldsymbol{y}} = g + h$ with $g \in L(F + G - P)$ and $h \in L(F + G' - D - P)$. On the other hand, $\lambda f_{\boldsymbol{y}} = \lambda f_{\boldsymbol{c}} + \lambda f_{\boldsymbol{e}} \in L(F + G) \oplus L(F + G' - D)$. Both decompositions are in $L(F + G) \oplus L(F + G' - D)$, hence the uniqueness gives

$$h = \lambda f_{\boldsymbol{e}} \in L(F + G' - D - P). \tag{3.10}$$

Though, since $\lambda \in L(F - D_{\boldsymbol{e}}) \setminus L(F - D_{\boldsymbol{e}} - P)$ we have $v_P(\lambda f_{\boldsymbol{e}}) = v_P(\lambda) + v_P(f_{\boldsymbol{e}}) = -v_P(F) + 1$, which contradicts (3.10). □

We can now prove Proposition 3.2.11.

Proof Proposition 3.2.11. If $S(F) = L(F - D_{\boldsymbol{e}})$, then for any point $P \in \operatorname{supp}(D)$, we get

$$\dim S(F - P) \geq \ell(F - P - D_{\boldsymbol{e}}) \geq \ell(F - D_{\boldsymbol{e}}) - 1 = \dim S(F) - 1.$$

Now, if $S(F) \neq L(F - D_{\boldsymbol{e}})$, by Proposition 3.1.5 we have $L(G + F - D + D_{\boldsymbol{e}}) \neq \{0\}$. Hence we get $\deg(G + F - D + D_{\boldsymbol{e}}) \geq 0$ that is $\deg(F + D_{\boldsymbol{e}}) - \mathrm{d}^* \geq 0$. Hence by applying Lemma 3.2.12 and Lemma 3.2.13 and using the hypothesis $\deg(F) \leq \mathrm{d}^* - g - 1$, we get that there exist at least

$$\deg D_{\boldsymbol{e}} - g - \deg(F + D_{\boldsymbol{e}}) + \mathrm{d}^* = \mathrm{d}^* - \deg F - g \geq 1$$

points P in $\operatorname{supp}(D_{\boldsymbol{e}})$ such that $S(F) \not\subseteq L(F - P)$ and $S(F - P) \neq S(F) \cap L(F - P)$. Hence for such a point we get

$$S(F - P) \not\subseteq S(F) \cap L(F - P) \not\subseteq S(F).$$

In particular $\dim S(F - P) \leq \dim S(F) - 2$. □

Now, let us consider a divisor F satisfying the hypotheses in Proposition 3.2.11 and $P \in \mathrm{supp}(D)$. We have

$$\ell(F - D_{\boldsymbol{e}}) - 1 \leq \ell(F - P - D_{\boldsymbol{e}}) \leq \ell(F - D_{\boldsymbol{e}}). \tag{3.11}$$

Let us denote by $\{P_{i_m}\}_{m \geq 1}$ the sequence of points in the support of D such that for any $m \geq 0$

$$\dim S(F - \sum_{j=1}^{m} P_{i_j} - P_{i_{m+1}}) \leq \dim S(F - \sum_{j=1}^{m} P_{i_j}) - 2,$$

and by F_{m+1} the divisor $F_m - P_{i_{m+1}}$, where $F_0 \stackrel{\text{def}}{=} F$. Since the sequence

$$\dim S(F) \geq \dim S(F_1) \geq \dim S(F_2) \geq \cdots$$

decreases faster than the sequence

$$\ell(F - D_{\boldsymbol{e}}) \geq \ell(F_1 - D_{\boldsymbol{e}}) \geq \ell(F_2 - D_{\boldsymbol{e}}) \geq \cdots$$

and $L(F_m - D_{\boldsymbol{e}}) \subseteq S(F_m)$ for any m, if the sequence is long enough, there will be an equality for some m. The length of the sequence depends on how many F_m's fulfill the two hypotheses of Proposition 3.2.11 $L(F_m - D_{\boldsymbol{e}}) \neq \{0\}$ and $\deg(F_m) \leq \mathrm{d}^* - g - 1$. The result in Theorem 3.2.16 will emphasize the role of the decoding radius $t \leq \frac{\mathrm{d}^* - 1}{2}$ in this problem.

Proposition 3.2.14. *We have* $\ell(F - D_{\boldsymbol{e}}) \leq \dim S(F) \leq \ell(F - D_{\boldsymbol{e}}) + \ell(G + F - D + D_{\boldsymbol{e}})$.

Proof. This result is a straightforward consequence of Proposition 3.1.5. □

Lemma 3.2.15. *Let $\mathcal{X}$ be a curve of genus g and $C = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ an algebraic geometry code on $\mathcal{X}$ with designed distance d^*. Let F be any divisor of degree $\deg F = t + 2g$. If $t = \mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}^* - 1}{2}$, then*

$$\dim S(F) - \ell(F - D_{\boldsymbol{e}}) \leq g.$$

Proof. By Proposition 3.2.14 we get

$$\dim S(F) - \ell(F - D_{\boldsymbol{e}}) \leq \ell(G + F - D + D_{\boldsymbol{e}}). \tag{3.12}$$

Since $t \leq \frac{\mathrm{d}^* - 1}{2}$ and $\mathrm{d}^* = n - \deg G$, we have in particular

$$\deg(G + F - D + D_{\boldsymbol{e}}) = n - \mathrm{d}^* + t + 2g - n + t = 2t - \mathrm{d}^* + 2g \leq 2g - 1. \tag{3.13}$$

Now we claim that $\ell(G + F - D + D_{\boldsymbol{e}}) \leq g$. If $\deg(G + F - D + D_{\boldsymbol{e}}) = 2g - 1 > 2g - 2$, we have by Riemann Roch theorem

$$\ell(G + F - D + D_{\boldsymbol{e}}) = \deg(G + F - D + D_{\boldsymbol{e}}) - g + 1 = 2g - 1 - g + 1 = g.$$

Otherwise, if $\deg(G + F - D + D_{\boldsymbol{e}}) \leq 2g - 2$, by Clifford's Theorem (see Theorem 1.7.52), we get

$$\ell(G + F - D + D_{\boldsymbol{e}}) \leq 1 + \frac{1}{2} \deg(G + F - D + D_{\boldsymbol{e}}) \leq g.$$

□

Theorem 3.2.16. *Let $\mathcal{X}$ be a curve of genus g and $C = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ an algebraic geometry code on $\mathcal{X}$ with designed distance $\mathrm{d}^* \geq 6g$. Let F be any divisor of degree $\deg F = t + 2g$. Then Algorithm 7 corrects every vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ with $t = \mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}^* - 1}{2}$.*

Proof. The proof can be found in [Ehr93], but we report it here to explicit the role of the decoding radius. Let F be a divisor with $\deg F = t + 2g$, where $t = \mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}^* - 1}{2}$. We denote by $F_0 = F, F_1, F_2 \ldots$ the sequence of divisors constructed by applying Proposition 3.2.11. First, let us prove that this sequence exists, that is, for any m smaller than a certain bound, the hypotheses of Proposition 3.2.11 hold for F_m. One can observe that since by hypothesis $t \leq \frac{\mathrm{d}^* - 1}{2}$ and $\mathrm{d}^* \geq 6g$, then

$$\mathrm{d}^* - t \geq \mathrm{d}^* - \frac{\mathrm{d}^* - 1}{2} \geq \frac{6g + 1}{2} = 3g + \frac{1}{2}.$$

In particular $t \leq \mathrm{d}^* - 3g - 1$. Therefore, for any m we have

$$\deg F_m = 2g + t - m \leq 2g + t \leq \mathrm{d}^* - g - 1.$$

We now prove that $L(F_m - D_{\boldsymbol{e}}) \neq \{0\}$ for any $m \leq g$. By Riemann-Roch theorem we get

$$\ell(F_m - D_{\boldsymbol{e}}) \geq t + 2g - m - t - g + 1 \geq 1. \tag{3.14}$$

Therefore the hypotheses of Proposition 3.2.11 are fulfilled for at least $F_0, \ldots, F_g$, which means that we can actually construct this sequence of divisors. We define

$$\Delta_m \stackrel{\text{def}}{=} \dim S(F_m) - \ell(F_m - D_{\boldsymbol{e}}). \tag{3.15}$$

By Lemma 3.2.15, we know that $\Delta_0 = \dim S(F) - \ell(F - D_{\boldsymbol{e}}) \leq g$. We show now that the sequence $(\Delta_m)_m$ is strictly decreasing. By using (3.11) and the definition of the P_{i_m}'s, we have

$$\begin{aligned} \Delta_{m+1} &= \dim S(F_m - P_{i_{m+1}}) - \ell(F_m - P_{i_{m+1}} - D_{\boldsymbol{e}}) \\ &\leq \dim S(F_m) - 2 - \ell(F_m - D_{\boldsymbol{e}}) + 1 \\ &= \Delta_m - 1. \end{aligned}$$

For any $m \leq g$, if $\Delta_m = 0$ then the algorithm stops. Otherwise the algorithm constructs F_{m+1} and gets $\Delta_{m+1} \leq \Delta_m - 1$. In this way, we can construct for sure at least $g+1$ divisors $F_0, \ldots, F_g$ and it is enough. Indeed, since $\Delta_0 \leq g$ and the sequence is strictly decreasing, we will get $\Delta_m = 0$ for some $m \leq g$. □

Algorithm 7 Ehrhard algorithm - unique decoding

Inputs: $\mathcal{C} = \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$, $f_{\boldsymbol{y}} = f_{\boldsymbol{c}} + f_{\boldsymbol{e}} \in \mathbb{F}_q^n$ where $\boldsymbol{c} \in \mathcal{C}$ and $\mathrm{w}(\boldsymbol{e}) \leq \frac{\mathrm{d}^* - 1}{2}$, $t = \mathrm{w}(\boldsymbol{e})$ and the projection $\pi_0 : L(F + G') \to L(F + G)$ with respect to the decomposition (3.2).
Output: $f_{\boldsymbol{c}} \in L(G)$ such that $\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}}) = \boldsymbol{c}$.

1: Choose F with $\mathrm{supp}(F) \cap \mathcal{P} = \emptyset$ and $\deg F = t + 2g$;
2: $j \leftarrow 0$ and $F_0 \leftarrow F$;
3: Look for a point $P \in \{P_1, \ldots, P_n\}$ such that $\dim(S(F_j - P)) \leq \dim(S(F_j)) - 2$;
4: **if** such a point P exists **then**
5: $\quad F_{j+1} \leftarrow F_j - P$;
6: $\quad j \leftarrow j + 1$;
7: $\quad$ go to Step 3;
8: **else** compute $f_{\boldsymbol{c}} = \frac{\pi_0(\Lambda f_{\boldsymbol{y}})}{\Lambda}$ for some $\Lambda \in S(F_j)$;
9: **return** $f_{\boldsymbol{c}}$;

Remark 3.2.17. Given the initial divisor F with $\deg F = t + 2g$, by Lemma 3.2.15, we know we have

$$\dim S(F) - \ell(F - D_{\boldsymbol{e}}) \leq g.$$

Therefore, since the sequence of the Δ_m's (see (3.15)) is strictly decreasing, g is the maximum number of steps that the algorithm can do. In particular, if the algorithm gets to the gth step, we have $S(F_g) = L(F_g - D_{\boldsymbol{e}})$. Hence, to conclude the algorithm after g steps, it is not necessary to check that any point $P \in \mathcal{P}$ verifies

$$\dim S(F_g) - \dim S(F_g - P) \leq 1.$$

An empirical consideration

In the process of adaptation of the divisor F, the points $P_{i_m} \in \mathrm{supp}(D)$ such that

$$\dim S(F_m - P_{m+1}) \leq \dim S(F_m) - 2, \tag{3.16}$$

do not belong necessarily to $\mathrm{supp}(D_{\boldsymbol{e}})$. Though theoretically, as shown, it is easier to prove the existence of points belonging to the support of $D_{\boldsymbol{e}}$ which do not satisfy the properties enounced in Lemma 3.2.12 and Lemma 3.2.13:

$$S(F - P) = S(F) \cap L(F - P) \qquad S(F) \subseteq L(F - P).$$

The reader can find in Table 3.1 some easy tests run on Ehrhard's algorithm. We denote by F_0 the chosen divisors of degree $t+2g$, by "pts" the set of points $\{P_{i_m}\}_m$ which verify (3.16) and by P a random point which does not belong to $\mathcal{P}$. One can note that actually condition (3.16) is fulfilled also by points P_i such that $P_i \notin \mathrm{supp}(D_{\boldsymbol{e}})$ and actually, it is quite difficult to find a point which does not make the dimension of $S(F)$ decrease fast. The tests are conducted as follow: the value of t is set at $\frac{\mathrm{d}^*-1}{2}$, a divisor $F=(t+g)P_\infty$ is defined and a random $\boldsymbol{y}\in\mathbb{F}_q^n$ is given until a $\boldsymbol{y}$ is found such that Porter's algorithm does not work on the chosen F. Therefore Ehrhard's algorithm is run on $\boldsymbol{y}$ and a divisor $F_0=F+\tilde{F}$ with $\deg\tilde{F}=g$. We observed that, whenever we considered $F_0=(t+2g)P_\infty$ and $\mathrm{supp}(F_0)\cap\mathrm{supp}(D)=\emptyset$, then only one point is tested, which is P_1. Hence we have $F_m=F_0-mP_1$. While, if we consider

$$F_0=(t+g)P_\infty+gP_1,$$

we usually have $F_m=F_0-mP_1$ for every $m<g$ and

$$F_g=F_0-(g-1)P_1-P_2=(t+g)P_\infty+P_1-P_2.$$

Indeed, by construction Porter's algorithm cannot work on $F=(t+g)P_\infty$.

In these tests the curves which are considered are the following:

1. $X^6+Y^6+XZ^5=0$ on $\mathbb{F}_{7^3}$;
2. $X^8-YZ^7-ZY^7=0$ on $\mathbb{F}_{7^2}$;
3. $ZY^5-X^6-XZ^5-Z^6=0$ on $\mathbb{F}_{11^3}$;

q	$\mathcal{X}$	g	n	$\deg G$	F_0	$\frac{\mathrm{d}^*-1-g}{2}$	$\frac{\mathrm{d}^*-1}{2}$	t	pts$\subseteq\mathrm{supp}(D_{\boldsymbol{e}})$	Δ_0	$\Delta_i-\Delta_{i+1}$
7^3	1	10	200	$2g-1$	$(t+2g)P_\infty$	85	90	90	false	10	1
7^3	1	10	200	$2g-1$	$(t+g)P_\infty+gP_1$	85	90	90	false	10	1
7^3	1	10	200	$2g$	$(t+g)P+gP_1$	84	89	89	false	9	1
7^2	2	21	230	$2g-1$	$(t+g)P_\infty+gP_1$	83	94	94	false	21	1
7^2	2	21	230	$2g-1$	$(t+g)P+gP_\infty$	83	94	94	false	21	1
7^2	2	21	230	$2g+1$	$(t+g)P+gP_\infty$	82	93	93	false	21	1
11^3	3	10	200	$2g-1$	$(t+2g)P_\infty$	85	90	90	false	10	1

Table 3.1: Tests on Ehrhard's algorithm.

3.3 Erasing the genus penalty: beyond the unique decoding bound

We now generalize the process seen in the previous sections: given a parameter $\ell\geq 1$ we will first prove the existence of a list decoding algorithm able to achieve Sudan's decoding radius with no genus penalty, which is

$$\frac{2\ell n-\ell(\ell+1)\deg G-2}{2(\ell+1)}. \tag{3.17}$$

For the sake of simplicity we first present a detailed proof for $\ell=2$. After that, we provide the indications for the easy generalization to $\ell\geq 2$. For that, we will combine together and extend some techniques and objects presented in Sudan's, Ehrhard's and Pellikaan's results.

3.3.1 Context

Let us consider as before $\mathcal{C}\stackrel{\mathrm{def}}{=}\mathcal{C}_L(\mathcal{X},\mathcal{P},G)$ with $g-1\leq\deg G<n$. Again, we suppose we have $\boldsymbol{y}=\boldsymbol{c}+\boldsymbol{e}$ as in Assumption 1 and that $\deg F$ verifies Assumption 3, that is, $\deg(F+2G)<n$. Moreover, let us consider, as in §3.1, a divisor G' such that $G'\geq G$ and $\ell(W+D-G')=0$. We recall that, under these hypotheses, every vector of $\mathbb{F}_q^n$ can be seen as the evaluation at the points of $\mathcal{P}$ of a function of $L(G')$. In particular, there exist $f_{\boldsymbol{c}}\in L(G)$, $f_{\boldsymbol{y}}\in L(G')$ and $f_{\boldsymbol{e}}\in L(G'-D+D_{\boldsymbol{e}})$ such that

$$\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{c}})=\boldsymbol{c},\quad \mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{y}})=\boldsymbol{y},\quad \mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{e}})=\boldsymbol{e}.$$

3.3.2 A reformulation of the spaces $S(F)$ and $L(F - D_{\boldsymbol{e}})$

First, for any divisor F, we present a new interpretation of the spaces $L(F - D_{\boldsymbol{e}})$ and $S(F)$ (see (3.5)).

Definition 3.3.1. Let F be a divisor. We define

$$GS(F) \stackrel{\text{def}}{=} \{(\lambda, \mu) \in L(F) \times L(F + G) \mid \lambda f_{\boldsymbol{y}} + \mu \in L(F + G' - D) \ \wedge \ \lambda f_{\boldsymbol{c}} + \mu = 0\}$$
$$AS(F) \stackrel{\text{def}}{=} \{(\lambda, \mu) \in L(F) \times L(F + G) \mid \lambda f_{\boldsymbol{y}} + \mu \in L(F + G' - D)\}.$$

Proposition 3.3.2. *Given $S(F)$ as in (3.5), we have*

$$GS(F) \simeq L(F - D_{\boldsymbol{e}})$$
$$AS(F) \simeq S(F).$$

Proof. Let us consider the map $\pi : AS(F) \to S(F)$ such that $\pi(\lambda, \mu) = \lambda$ for any $(\lambda, \mu) \in AS(F)$. This map is well defined, as for any $(\lambda, \mu) \in AS(F)$, there exists $\alpha \in L(F + G' - D)$ such that

$$\lambda f_{\boldsymbol{y}} = -\mu + \alpha \in L(F + G) \oplus L(F + G' - D).$$

It is clear that π is linear. It is also injective, as if $\pi(\lambda, \mu) = 0$, then $\lambda = 0$ and

$$\mu \in L(F + G) \cap L(F + G' - D) = \{0\}$$

by Assumption 2. We prove now that it is surjective: given $\lambda \in S(F)$ we have $\lambda f_{\boldsymbol{y}} = \beta + \alpha$ with unique $\beta \in L(F + G)$ and $\alpha \in L(F + G' - D)$. Then $\pi(\lambda, -\beta) = \lambda$.

We now prove that $GS(F) \simeq L(F - D_{\boldsymbol{e}})$: let us consider the map $\rho : GS(F) \to L(F - D_{\boldsymbol{e}})$ such that $\rho(\lambda, \mu) = \lambda$ for any $(\lambda, \mu) \in GS(F)$. First we prove that this map is well defined. Let $(\lambda, \mu) \in GS(F)$. Since $\lambda f_{\boldsymbol{c}} + \mu = 0$, we have

$$\lambda f_{\boldsymbol{e}} = \lambda f_{\boldsymbol{y}} + \mu \in L(F + G' - D). \tag{3.18}$$

Now, given $P \in \mathrm{supp}(D_{\boldsymbol{e}})$, since $\mathrm{supp}(G') \cap \mathcal{P} = \emptyset$ and $f_{\boldsymbol{e}}(P) \neq 0$, we have $v_P(f_{\boldsymbol{e}}) = 0$. Thus, by (3.18) we get

$$-v_P(F) + 1 \leq v_P(\lambda f_{\boldsymbol{e}}) = v_P(\lambda) + v_P(f_{\boldsymbol{e}}) = v_P(\lambda).$$

Since this holds for any $P \in \mathrm{supp}(D_{\boldsymbol{e}})$, we have $\lambda \in L(F - D_{\boldsymbol{e}})$. Now, the map ρ is clearly linear and it is easy to see, using the same argument as for π, that it is also injective. To conclude, we prove that it is surjective. Given $\lambda \in L(F - D_{\boldsymbol{e}})$, we have seen that

$$\lambda f_{\boldsymbol{y}} = \lambda f_{\boldsymbol{c}} + \lambda f_{\boldsymbol{e}},$$

where $\lambda f_{\boldsymbol{c}} \in L(F + G)$ and $\lambda f_{\boldsymbol{e}} \in L(F + G' - D)$. Therefore, we get $(\lambda, -\lambda f_{\boldsymbol{c}}) \in GS(F)$ and

$$\rho(\lambda, -\lambda f_{\boldsymbol{c}}) = \lambda,$$

that is, ρ is an isomorphism. □

It is possible to see the defined spaces $AS(F)$ and $GS(F)$ as spaces of particular polynomials of degree 1 by the following map

$$(\lambda, \mu) \longrightarrow \lambda Y + \mu.$$

In particular, we can see $GS(F)$ as the space of *Good Sudan polynomials*, that is, the polynomials which vanish at $f_{\boldsymbol{c}}$ and, hence, which make possible to recover $\boldsymbol{c}$. In contraposition with this idea, the space $AS(F)$ is seen as the space of *All Sudan polynomials*. From this point of view, Ehrhard's algorithm simply consists in looking for the space of good Sudan polynomials whose degree, since we considered $t \leq \frac{\mathrm{d}^* - 1}{2}$, is 1. We have seen that Sudan's algorithm is able to correct more errors by considering specific polynomials of larger degree, as all solutions will be among their roots. Hence, the aim of this chapter is to adapt $AS(F)$ and $GS(F)$ to have Sudan polynomials of larger degree and use the process proposed by Ehrhard to deduce a list decoding algorithm with no genus penalty in its decoding radius. Before that,

we need to do some considerations.

First, we introduce a parameter $\ell \geq 1$ which will have the role of the degree of the Sudan polynomials we are going to work with. In particular ℓ is also the degree of the good Sudan polynomial we look for. Here, we set $\ell = 2$. Hence, we can consider the vector $\boldsymbol{y}^2$. Let us consider the function $f_{\boldsymbol{y}} \in L(G')$ where, we recall, $\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{y}}) = \boldsymbol{y}$. In particular, we get $f_{\boldsymbol{y}}^2 \in L(2G')$ and $\mathrm{ev}_{\mathcal{P}}(f_{\boldsymbol{y}}^2) = \boldsymbol{y}^2$. We now denote $f_{\boldsymbol{y}}^2$ by $f_{\boldsymbol{y}^2}$ and define the following map

$$\begin{aligned} L(F) &\longrightarrow L(F + 2G') \\ \lambda &\longrightarrow \lambda f_{\boldsymbol{y}^2}. \end{aligned}$$

One can easily prove that this map is well-defined. We would like to see the space $L(F + 2G')$ as the direct sum of some particular subspaces (as for $L(F + G')$ in §3.2). Both the spaces $L(F + 2G)$ and $L(F + 2G' - D)$ are included in $L(F + 2G')$ and it holds

$$L(F + 2G) \cap L(F + 2G' - D) = L(F + 2G - D).$$

Assumption 3. We assume that $\deg(F + 2G) < n$.

Under Assumption 3, we get $\deg(F + 2G - D) < 0$, hence $L(F + 2G - D) = \{0\}$ and there exists a subspace Z_2 of $L(F + 2G')$ such that

$$L(F + 2G') = L(F + 2G) \oplus L(F + 2G' - D) \oplus Z_2. \tag{3.19}$$

It is actually possible to compute the dimension of the space Z_2.

Lemma 3.3.3. *Given Z_2 as in* (3.19)*, then*

$$\dim Z_2 = \deg(D - F - 2G) + g - 1.$$

Proof. First, we show that $\deg(F + 2G) = \deg(F + 2G' - D) > 2g - 2$. Since we took G' such that $\ell(W + D - G') = 0$, by Riemann-Roch theorem we have

$$0 = \ell(W + D - G') \geq 2g - 2 + n - \deg G' - g + 1,$$

that is $\deg G' \geq n + g - 1$. Thus, since $\deg F \geq t + g$, we get

$$\deg(F + 2G' - D) \geq \deg(F + G' - D) > 2g - 2. \tag{3.20}$$

Furthermore, since in §3.3.1 we assumed $\deg G \geq g - 1$, we have

$$\deg(F + 2G) \geq \deg(F + G) \geq t + g + g - 1 > 2g - 2.$$

Therefore, using Proposition 1.7.50 we can compute from (3.19)

$$\begin{aligned} \dim Z_2 &= \ell(F + 2G') - \ell(F + 2G) - \ell(F + 2G' - D) \\ &= \deg F + 2\deg G' - g + 1 - \deg F - 2\deg G + g - 1 - \deg F - 2\deg G' + n + g - 1 \\ &= \deg(D - F - 2G) + g - 1. \end{aligned}$$

□

It is now possible to give the definition of $AS(F)$ and $GS(F)$ for $\ell = 2$.

Definition 3.3.4. We define

$$\begin{aligned} AS(F) &\stackrel{\text{def}}{=} \{(\lambda_0, \lambda_1, \lambda_2) \in L(F) \times L(F + G) \times L(F + 2G) \mid \lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 \in L(F + 2G' - D)\}, \\ GS(F) &\stackrel{\text{def}}{=} \{(\lambda_0, \lambda_1, \lambda_2) \in L(F) \times L(F + G) \times L(F + 2G) \mid \lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 \in L(F + 2G' - D) \ \wedge \\ &\qquad\qquad \lambda_0 f_{\boldsymbol{c}}^2 + \lambda_1 f_{\boldsymbol{c}} + \lambda_2 = 0\}. \end{aligned}$$

Note that $GS(F)$ depends on the solution $f_{\boldsymbol{c}}$. In particular, we know that, for values of t larger than half the minimum distance, we can have a list of solutions $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \dots\}$. To any solution $\boldsymbol{c}_i$ then, corresponds a space $GS_i(F)$. On the other hand, there exists only one space $AS(F)$, since it is independent from the $\boldsymbol{c}_i$'s. To avoid heavy notations, except for situations where it is required, we will write generically $GS(F)$, meaning the space relative to a specific solution $\boldsymbol{c}$. Moreover, we use the same notation for Definition 3.3.1 and Definition 3.3.4. There will be no ambiguity since from now on, except for some case we will precise, whenever we will use the notation $AS(F)$ and $GS(F)$, we will mean the spaces with polynomial's degree equal to $\ell = 2$.

Proposition 3.3.5. *The following sequence*

$$0 \longrightarrow GS(F) \xrightarrow{\ i\ } AS(F) \xrightarrow{\ \Phi\ } L(F + 2G - D + D_{\boldsymbol{e}}),$$

where $\Phi(\lambda_0, \lambda_1, \lambda_2) = \lambda_0 f_{\boldsymbol{c}}^2 + \lambda_1 f_{\boldsymbol{c}} + \lambda_2$*, is exact.*

Proof. The proof follows exactly the same path as the one of Proposition 3.1.5. □

Remark 3.3.6. It is clear that as we have $L(2G + F - D + D_{\boldsymbol{e}}) = \{0\}$, then all polynomial in $AS(F)$ will vanish in $f_{\boldsymbol{c}}$. However, the space $L(2G + F - D + D_{\boldsymbol{e}})$ depends on a particular error $\boldsymbol{e}$. In order to have a list decoding algorithm then, we should have $L(2G + F - D + D_{\tilde{\boldsymbol{e}}}) = \{0\}$ for any $\tilde{\boldsymbol{e}}$ with small weight and such that $\boldsymbol{y} = \tilde{\boldsymbol{c}} + \tilde{\boldsymbol{e}}$ for some $\tilde{\boldsymbol{c}} \in \mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$.

We now give some estimations for the dimension of $AS(F)$ and $GS(F)$ which will be useful in what follows.

Proposition 3.3.7. *Given $AS(F)$ and $GS(F)$ as in Definition 3.3.4, the following statements hold*

1) $\ell(F) + \ell(F + G) - \dim Z_2 \leq \dim AS(F) \leq \ell(F) + \ell(F + G)$,

2) $\ell(F) + \ell(F + G) - t \leq \dim GS(F) \leq \ell(F) + \ell(F + G)$.

Proof. To prove (1) we first observe that if

$$\overline{AS}(F) \stackrel{\text{def}}{=} \{(\lambda_0, \lambda_1) \in L(F) \times L(F + G) \mid \lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} \in L(F + 2G) \oplus L(F + 2G' - D)\},$$

then there exists a map $\varphi : AS(F) \to \overline{AS}(F)$ such that $\varphi(\lambda_0, \lambda_1, \lambda_2) = (\lambda_0, \lambda_1)$ for any $(\lambda_0, \lambda_1, \lambda_2) \in AS(F)$. It is easy to verify that it is well defined and linear. Also, φ is bijective. Indeed, given $(\lambda_0, \lambda_1) \in \overline{AS}(F)$, we have

$$\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} = \alpha + \beta, \tag{3.21}$$

where $\alpha \in L(F+2G)$ and $\beta \in L(F+2G'-D)$. Hence $(\lambda_0, \lambda_1, -\alpha) \in AS(F)$ and $\varphi(\lambda_0, \lambda_1, -\alpha) = (\lambda_0, \lambda_1)$. Furthermore, the decomposition in (3.21) is unique by (3.19). Hence φ is also injective. So we have $AS(F) \simeq \overline{AS}(F)$ and hence we can estimate $\dim \overline{AS}(F)$ instead of $\dim AS(F)$. Clearly, we have

$$\dim \overline{AS}(F) \leq \ell(F) + \ell(F + G).$$

Moreover, we can consider the projection $\pi_2 : L(F + 2G') \to Z_2$ with respect to (3.19) and the following sequence:

$$0 \longrightarrow \overline{AS}(F) \xrightarrow{\ i\ } L(F) \times L(F + G) \xrightarrow{\ \Psi\ } Z_2,$$

where $\Psi(\lambda_0, \lambda_1) = \pi_2(\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}})$. It is easy to verify that this sequence is exact. Hence, we have

$$\dim \overline{AS}(F) \geq \ell(F) + \ell(F + G) - \dim Z_2.$$

To prove statement (2), similarly we define

$$\overline{GS}(F) \stackrel{\text{def}}{=} \{(\lambda_0, \lambda_1) \in L(F) \times L(F + G) \mid \lambda_0(2 f_{\boldsymbol{c}} f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}} \in L(F + 2G' - D)\} \tag{3.22}$$

and exhibit the map $\psi : GS(F) \to \overline{GS}(F)$ such that $\psi(\lambda_0, \lambda_1, \lambda_2) = (\lambda_0, \lambda_1)$ for any $(\lambda_0, \lambda_1, \lambda_2) \in GS(F)$. Again, it is easy to see that this map is well defined, linear and injective. Let us show that it is also surjective. Given $(\lambda_0, \lambda_1) \in \overline{GS}(F)$, we have

$$\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} = (\lambda_0 f_{\boldsymbol{c}}^2 + \lambda_1 f_{\boldsymbol{c}}) + (\lambda_0(2 f_{\boldsymbol{c}} f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}}).$$

In particular if we denote $\lambda_2 \stackrel{\text{def}}{=} -(\lambda_0 f_{\boldsymbol{c}}^2 + \lambda_1 f_{\boldsymbol{c}})$, we have $\lambda_2 \in L(F + 2G)$, $(\lambda_0, \lambda_1, \lambda_2) \in GS(F)$ and $\psi(\lambda_0, \lambda_1, \lambda_2) = (\lambda_0, \lambda_1)$. We then have $GS(F) \simeq \overline{GS}(F)$ and $\dim \overline{GS}(F) \leq \ell(F) + \ell(F + G)$. On the other hand we observe that given $(\lambda_0, \lambda_1) \in L(F) \times L(F + G)$, for any point $P \notin \mathrm{supp}(D_{\boldsymbol{e}})$ it holds

$$v_P(\lambda_0(2 f_{\boldsymbol{c}} f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}}) \geq -v_P(F) + 1.$$

Hence for (λ_0, λ_1) to belong to $\overline{GS}(F)$ one has to impose at most t conditions. We get then

$$\dim GS(F) = \dim \overline{GS}(F) \geq \ell(F) + \ell(F + G) - t.$$

□

Remark 3.3.8. We point out that the proof above gives a characterization of the elements of $GS(F)$ and $\overline{GS}(F)$, where the latter is defined in (3.22). Indeed, we have that $(\lambda_0, \lambda_1, \lambda_2) \in GS(F)$, or equivalently, $(\lambda_0, \lambda_1) \in \overline{GS}(F)$, if and only if, for any point $P \in \mathrm{supp}(D_{\boldsymbol{e}})$, we have

$$v_P(\lambda_0(2 f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) \geq -v_P(F) + 1.$$

Remark 3.3.9. Under some conditions on the degree of F and G it is possible to prove that

$$\dim \overline{GS}(F) = \ell(F) + \ell(F + G) - t.$$

This result will be proven later (see Proposition 3.3.19).

3.3.3 The existence of a list decoding algorithm with no genus penalty

In this section we give a proof of the existence of several list decoding algorithms. In particular, for any $\ell \leq M$ for a certain M, we show that there exists a list decoding algorithm with a genus-free decoding radius and retrieving at most ℓ solutions. As anticipated, we will give a detailed proof for $\ell = 2$ and, later, the indication to find the easy generalization to $\ell \geq 2$. The proof follows the results of Pellikaan and Vlădut proposed in §3.2.2. Let us suppose there exist $\{\boldsymbol{c}_1, \boldsymbol{c}_2\} \subseteq \mathcal{C}$ and $\{\boldsymbol{e}_1, \boldsymbol{e}_2\} \subseteq \mathbb{F}_q^n$ such that

$$\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \boldsymbol{c}_2 + \boldsymbol{e}_2$$

and $\mathrm{w}(\boldsymbol{e}_i) \leq t_{max}(2)$, where we set

$$t_{max}(2) = \frac{2n - 3\deg G - 2}{3}. \tag{3.23}$$

For $i = 1, 2$, we define $GS_i(F)$ the space $GS(F)$ relative to $\boldsymbol{c}_i$. By Proposition 3.3.5 and Remark 3.3.6 we want to find a divisor F such that

$$L(F + 2G - D + D_{\boldsymbol{e}_1}) = L(F + 2G - D + D_{\boldsymbol{e}_2}) = \{0\}.$$

At the same time though, we need to have $GS_i(F) \neq \{0\}$, hence, by Proposition 3.3.7, it suffices to impose $\ell(F) + \ell(F + G) - t_{max}(2) > 0$, which holds if

$$\deg F \geq \frac{t_{max}(2) - \deg G + 2g - 1}{2}.$$

Theorem 3.3.10. *Assume* $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{2}$, $\deg G \leq \frac{2n-5}{6}$ *and suppose that* $\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \boldsymbol{c}_2 + \boldsymbol{e}_2$ *with* $\mathrm{w}(\boldsymbol{e}_i) \leq t_{max}(2)$ *for* $i = 1, 2$. *Then there exists a divisor* F *with degree*

$$\deg F = \left\lceil \frac{t_{max}(2) - \deg G + 2g - 1}{2} \right\rceil, \tag{3.24}$$

satisfying the condition

$$L(F + 2G - D + D_{\boldsymbol{e}_1}) = L(F + 2G - D + D_{\boldsymbol{e}_2}) = \{0\}. \tag{3.25}$$

Proof. Let us denote $t_i = \mathrm{w}(\boldsymbol{e}_i)$ for $i = 1, 2$ and suppose that $t_1 \leq t_2$. Now, if for any divisor F of degree as in (3.24) we have

$$L(F + 2G - D + D_{\boldsymbol{e}_1}) = \{0\},$$

then, by following the same argument of Theorem 3.2.6, we can exhibit a divisor F_0 with degree as in (3.24), such that $L(F_0 + 2G - D + D_{\boldsymbol{e}_2}) = \{0\}$ as well.
We suppose now that there exists F_0 with degree as in (3.24) verifying

$$L(F_0 + 2G - D + D_{\boldsymbol{e}_1}) \neq \{0\}.$$

Then there exists $0 \neq f \in L(F_0 + 2G - D + D_{\boldsymbol{e}_1})$ and we can consider the divisor R_0 defined as follows

$$R_0 = (f) + F_0 + 2G - D + D_{\boldsymbol{e}_1} \geq 0.$$

We now define $k_1 = \deg R_0$ and $k_2 \stackrel{\text{def}}{=} \deg R_0 - t_2 + t_1$. It is easy to see that since

$$t_1 \leq t_{max}(2) = \frac{2n - 3\deg G - 2}{3},$$

we have $k_1 \leq g - 1$. Moreover, by the assumption $t_2 \geq t_1$ and the definition of k_2 we have $k_2 \leq k_1$, hence $k_2 \leq g - 1$ as well. We now introduce $i = 1, 2$ the map

$$\begin{aligned} \varphi_i : \mathbb{D}_{k_i} &\longrightarrow J(\mathcal{X}) \\ R &\longmapsto [R - R_0 - D_{\boldsymbol{e}_i} + D_{\boldsymbol{e}_1}] \end{aligned}$$

Since we have $k_i \leq g - 1$ for any $i = 1, 2$, by hypothesis and the fact that $\mathcal{X}$ disposes of rational points, we get $|\mathbb{D}_{k_i}| \leq |\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{2}$. Hence $\mathrm{Im}(\varphi_1) \cup \mathrm{Im}(\varphi_2) \subsetneq J(\mathcal{X})$. There exists then

$$[E] \in J(\mathcal{X}) \setminus \mathrm{Im}(\varphi_1) \cup \mathrm{Im}(\varphi_2).$$

On the other hand, by the hypothesis $\deg G \leq \frac{2n-5}{6}$, we have $\deg F_0 \geq g$. Hence by Proposition 3.2.5 the following map is instead surjective:

$$\begin{aligned} \varphi : \mathbb{D}_{\deg F_0} &\longrightarrow J(\mathcal{X}) \\ F &\longmapsto [F - F_0]. \end{aligned}$$

Hence there exists $F \in \mathbb{D}_{\deg F_0}$ such that $[E] = [F - F_0]$. Now we prove that F has to verify (3.25) for both $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. If, by way of contradiction,

$$L(F + 2G - D + D_{\boldsymbol{e}_1}) \neq \{0\},$$

then there exists $R_1 \in \mathbb{D}_{k_1}$ such that $0 \leq R_1 \sim F + 2G - D + D_{\boldsymbol{e}_1}$. Thus we would get

$$[E] = [F - F_0] = [R_1 - R_0] = \varphi_1(R_1), \tag{3.26}$$

which is a contradiction since $[E] \notin \mathrm{Im}(\varphi_1)$. The same argument works for $\boldsymbol{e}_2$. We have then

$$L(F + 2G - D + D_{\boldsymbol{e}_1}) = L(F + 2G - D + D_{\boldsymbol{e}_2}) = \{0\}.$$

□

One can observe that there are no results yet insuring that the number of solutions of the list decoding problem for $t_{max}(2)$ is bounded by 2. Though this existence result entails this property.

Corollary 3.3.11. *Let $\boldsymbol{y} \in \mathbb{F}_q^n$. If $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{3}$, then there exist at most 2 solutions in the code $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ to the bounded decoding problem with parameter $t_{max}(2)$.*

Proof. Let us suppose by way of contradiction that there exist 3 solutions $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ for $\boldsymbol{y}$. Then, following the same argument of the proof of Theorem 3.3.10, we can show that there exists a divisor F such that $L(F + 2G - D + D_{\boldsymbol{e}_i}) = \{0\}$ for any $1 \leq i \leq 3$. But then, by Proposition 3.3.5, we get

$$GS_1(F) = GS_2(F) = GS_3(F) = AS(F).$$

In particular, any polynomial belonging to $AS(F)$, will vanish in $f_{\boldsymbol{c}_1}, f_{\boldsymbol{c}_2}$ and $f_{\boldsymbol{c}_3}$. Though these polynomials have degree 2 and $\deg F$ is such that $GS_i(F) \neq \{0\}$, thus we get a contradiction. □

Now, it is possible to generalize Theorem 3.3.10 and, hence, Corollary 3.3.11 to any $\ell \geq 2$. We define

$$t_{max}(\ell) \stackrel{\text{def}}{=} \frac{2\ell n - \ell(\ell+1)\deg G - 2\ell}{2(\ell+1)}$$

and consider $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, with $\mathrm{w}(\boldsymbol{e}) \leq t_{max}(\ell)$. The spaces $AS(F)$ and $GS(F)$ become

$$\begin{aligned} AS(F) &\stackrel{\text{def}}{=} \{(\lambda_0, \ldots, \lambda_\ell) \in L(F) \times \cdots \times L(F+\ell G) \mid \quad \lambda_0 f_{\boldsymbol{y}}^\ell + \lambda_1 f_{\boldsymbol{y}}^{\ell-1} + \cdots + \lambda_\ell \in L(F + \ell G' - D)\}, \\ GS(F) &\stackrel{\text{def}}{=} \{(\lambda_0, \ldots, \lambda_\ell) \in L(F) \times \cdots \times L(F+\ell G) \mid \quad \lambda_0 f_{\boldsymbol{y}}^\ell + \lambda_1 f_{\boldsymbol{y}}^{\ell-1} + \cdots + \lambda_\ell \in L(F + \ell G' - D)\} \ \wedge \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \lambda_0 f_{\boldsymbol{c}}^\ell + \lambda_1 f_{\boldsymbol{c}}^{\ell-1} \cdots + \lambda_\ell = 0\}. \end{aligned}$$

Proposition 3.3.12. *The following sequence*

$$0 \longrightarrow GS(F) \xrightarrow{\quad i \quad} AS(F) \xrightarrow{\Phi} L(F + \ell G - D + D_{\boldsymbol{e}}),$$

where $\Phi(\lambda_0, \ldots, \lambda_\ell) = \lambda_0 f_{\boldsymbol{c}}^\ell + \cdots + \lambda_{\ell-1} f_{\boldsymbol{c}} + \lambda_\ell$*, is exact.*

Again, from Proposition 3.3.12, it is clear that if $L(F + \ell G - D + D_{\boldsymbol{e}}) = \{0\}$, then $GS(F) = AS(F)$. In particular, suppose there exist ℓ solutions $\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \cdots = \boldsymbol{c}_\ell + \boldsymbol{e}_\ell$ with $\mathrm{w}(\boldsymbol{e}_i) \leq t_{max}(\ell)$ for any $i = 1, \ldots, \ell$. Then, to find all of them, one needs to have a divisor F such that

$$L(F + \ell G - D + D_{\boldsymbol{e}_i}) = \{0\} \quad \forall i = 1, \ldots, \ell.$$

Furthermore, if we assume $\deg(F + \ell G) < n$, we can prove in particular that

$$\dim GS(F) \geq \sum_{i=0}^{\ell-1} \ell(F + iG) - t_{max}(\ell),$$

(see the case $\ell = 2$ in Proposition 3.3.7). Hence, if

$$\deg F \geq \frac{t_{max}(\ell) - \ell(\ell-1)\deg G + \ell g - \ell + 1}{\ell},$$

then, we have $GS(F) \neq \{0\}$.

Theorem 3.3.13. *Assume* $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell}$ *and* $\deg G \leq \frac{2(\ell n - \ell^2 - \ell + 1)}{\ell(\ell+1)(2\ell-1)}$*. Moreover, suppose we have*

$$\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \cdots = \boldsymbol{c}_\ell + \boldsymbol{e}_\ell,$$

where $\mathrm{w}(\boldsymbol{e}_i) \leq t_{max}(\ell)$ *for any* $i \in \{1, \ldots, \ell\}$*. Then there exists a divisor* F *with degree*

$$\deg F = \left\lceil \frac{t_{max}(\ell) - \ell(\ell-1)\deg G + \ell g - \ell + 1}{\ell} \right\rceil \tag{3.27}$$

satisfying the condition

$$L(F + \ell G - D + D_{\boldsymbol{e}_i}) = \{0\} \quad \forall i = 1, \ldots, \ell. \tag{3.28}$$

Proof. Again let us denote $t_i = \mathrm{w}(\boldsymbol{e}_i)$ for every $t = 1, \ldots, \ell$ and let us suppose that $t_1 \leq t_2 \leq \cdots \leq t_\ell$. We consider the minimum index $1 \leq i \leq \ell$ such that there exists F_0 with degree as in (3.27) and

$$L(F + \ell G - D + D_{\boldsymbol{e}_i}) \neq \{0\}.$$

Then there exists $0 \neq f \in L(F_0 + \ell G - D + D_{\boldsymbol{e}_i})$ and we can consider the divisor R_0 defined as follows

$$R_0 = (f) + F_0 + \ell G - D + D_{\boldsymbol{e}_i} \geq 0.$$

We now define $k_j \stackrel{\text{def}}{=} \deg R_0 - t_j + t_i$. It is easy to see that since $t_i \leq t_{max}(\ell)$, we get $k_i \leq g-1$. Moreover, since $t_j \geq t_i$ for any $j \geq i$, we have $k_j \leq k_i$, hence $k_j \leq g-1$ for any $j \geq i$. We now introduce for any $j \geq i$ the map

$$\begin{aligned} \varphi_i : \mathbb{D}_{k_j} &\longrightarrow J(\mathcal{X}) \\ R &\longmapsto [R - R_0 - D_{\boldsymbol{e}_j} + D_{\boldsymbol{e}_i}] \end{aligned}$$

Since we have $k_j \leq g-1$ for any $j \geq i$, by hypothesis we get $|\mathbb{D}_{k_j}| \leq |\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell}$. Hence, we get $\bigcup_{j=i}^{\ell} \mathrm{Im}(\varphi_j) \subsetneq J(\mathcal{X})$. One can then conclude using the same argument of the proof for the case $\ell = 2$. $\square$

Corollary 3.3.14. *Let $\ell \geq 1$ and $\boldsymbol{y} \in \mathbb{F}_q^n$. If $|\mathbb{D}_{g-1}| < \frac{|J(\mathcal{X})|}{\ell+1}$, then there exist at most ℓ solutions in the code $\mathcal{C}_L(\mathcal{X}, \mathcal{P}, G)$ to the bounded decoding problem with parameter $t_{max}(\ell)$.*

Note that this result can be applied to all curves proposed by Pellikaan and Vlădut in [Pel89] and [Vl0]. In particular, we are in the hypothesis of Theorem 3.3.10 whenever $\deg G$ is not too large and one of the following properties is fulfilled:

- by Proposition 3.2.4, the curve is maximal, $q > 3$ and $\ell = q - 1$;
- by Proposition 3.2.8, $q \geq 37$ and $\ell \leq \frac{q-3}{2}$;
- by Proposition 3.2.9, $q \geq 16$ and the genus is large enough and $\ell \leq \frac{q-3}{2}$.

Remark 3.3.15. Note that, if $\mathcal{C} \subseteq \mathbb{F}_q^n$, there exists a list decoding algorithm for $\mathcal{C}$ for any $\ell \geq 1$ and q. Indeed, given ℓ and q, one can consider a power m of q, such that $q^m > 37$ and $\ell \leq \frac{q^m-3}{2}$. By Proposition 3.2.8 and Theorem 3.3.10, there exists a list decoding algorithm for the code build over $\mathbb{F}_{q^m}$. Let L_{q^m} and L_q be respectively the lists of solutions for the code over $\mathbb{F}_{q^m}$ and the one over $\mathbb{F}_q$. The length of L_{q^m} is less than ℓ, by Corollary 3.3.11. We can obtain $L_q = L_{q^m} \cap \mathbb{F}_q^n$, where in particular $|L_q| \leq \ell$ as well.

3.3.4 A new algorithm for $\ell = 2$

In this section, we provide a list decoding algorithm with decoding radius

$$t_{max}(2) \stackrel{\text{def}}{=} \frac{2n - 3\deg G - 2}{3}. \tag{3.29}$$

The idea behind this new algorithm is similar to the one characterizing Ehrhahrd's result: given a divisor F with high degree and $AS(F), GS(F)$ be as in Definition 3.3.4, we create a sequence $F_0 = F, F_1, \ldots$ of divisors with decreasing degree, the aim being to obtain $AS(F_m) = GS(F_m)$ for a certain m. To ensure that, we show that for any $0 \leq i \leq M$ for a sufficiently large M, there exists a point $P \in \mathcal{P}$ such that $\dim AS(F_i - P)$ decreases faster than $\dim GS(F_i - P)$. To do so, we need to prove some results and to introduce the following objects. We precise that we will consider the notion of $AS(F)$ and $GS(F)$ with respect to $\ell = 2$, that is, we will work with polynomials of degree 2.

Definition 3.3.16. Given $AS(F)$ and $GS(F)$ as in Definition 3.3.4 and an effective divisor H with $0 \leq \mathrm{supp}(H) \leq D$, we denote

- $AS(F, H) \stackrel{\text{def}}{=} AS(F) \cap (L(F - H) \times L(F + G) \times L(F + 2G))$;
- $GS(F, H) \stackrel{\text{def}}{=} GS(F) \cap (L(F - H) \times L(F + G) \times L(F + 2G))$;
- $AS(F, H, H) \stackrel{\text{def}}{=} AS(F) \cap (L(F - H) \times L(F + G - H) \times L(F + 2G))$;
- $GS(F, H, H) \stackrel{\text{def}}{=} GS(F) \cap (L(F - H) \times L(F + G - H) \times L(F + 2G))$.

Note that we did not bother introducing the objects "$AS(F, H, H, H)$" and "$GS(F, H, H, H)$", since by the following result, they equal respectively $AS(F, H, H)$ and $GS(F, H, H)$.

Proposition 3.3.17. *For any divisor H with $0 \leq \mathrm{supp}(H) \leq D$, the following statements hold*

$$AS(F) \cap (L(F - H) \times L(F + G - H) \times L(F + 2G - H)) = AS(F, H, H), \tag{3.30}$$
$$GS(F) \cap (L(F - H) \times L(F + G - H) \times L(F + 2G - H)) = GS(F, H, H). \tag{3.31}$$

Proof. Note that (3.30) implies (3.31). Indeed, let us assume (3.30): on the one hand we clearly have

$$GS(F) \cap (L(F - H) \times L(F + G - H) \times L(F + 2G - H) \subseteq GS(F, H, H).$$

On the other hand, let us consider an element $(\lambda_0, \lambda_1, \lambda_2) \in GS(F, H, H)$. In particular, since $GS(F, H, H) \subseteq AS(F, H, H)$ and (3.30) holds, we get $\lambda_2 \in L(F - H)$, that is, we have (3.31). We prove, then,

only the first statement. It is clear that we have the inclusion "$\subseteq$". Conversely, let us now consider $(\lambda_0, \lambda_1, \lambda_2) \in AS(F, H, H)$, that is, *a priori* $\lambda_2 \in L(F+2G)$. We want to prove that $\lambda_2 \in L(F+2G-H)$. We know that

$$\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 = \alpha \in L(F + 2G' - D).$$

Hence for every point $P \in \mathrm{supp}(H)$, since $H \leq D$, $\mathrm{supp}(G') \cap \mathcal{P} = \emptyset$, we get $v_P(\alpha) \geq -v_P(F) + 1$. On the other hand, since $\lambda_0 \in L(F - H)$ and $\lambda_1 \in L(F + G - H)$, we have $v_P(\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}})) \geq -v_P(F) + 1$. Thus we get

$$\begin{aligned} v_P(\lambda_2) &= v_P(\alpha - \lambda_0 f_{\boldsymbol{y}}^2 - \lambda_1 f_{\boldsymbol{y}}) \\ &\geq \min\{v_P(\alpha), v_P(\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}})\} \\ &\geq \min\{-v_P(F) + 1, -v_P(F) + 1\} = -v_P(F) + 1, \end{aligned}$$

that is, $\lambda_2 \in L(F + 2G - H)$. □

We now want to imitate the adaptation process seen in Ehrhard's algorithm [Ehr93] (see §3.2). First, we give the following notion.

Definition 3.3.18. Given a divisor F and a point $P \in \mathcal{P}$, we say that P *narrows the gap* for F, if

$$\dim AS(F) - \dim AS(F - P) > \dim GS(F) - \dim GS(F - P).$$

There are situations in which it is possible to know whether a point narrows the gap for a divisor F or not, even with no information about $GS(F)$. However, to deduce this information, we need to study the behavior of $GS(F - P)$. The following result will give us the precise dimension of $GS(F)$ under some conditions and, hence, it will help to estimate $\dim GS(F) - \dim GS(F - P)$ for some point $P \in \mathcal{P}$.

Proposition 3.3.19. *If* $\deg(F + G) \geq t + 2g - 1$, *then* $\dim GS(F) = \ell(F) + \ell(F + G) - t$.

In order to prove this result we need to introduce some notations. Given $f \in \mathbb{F}(\mathcal{X})$ and P a place of $\mathcal{X}$, we define the $\mathrm{coeff}_P(f, i)$'s which are the coefficients of the Laurent power series of f in P, i.e.

$$f = \sum_{i \in \mathbb{Z}} \mathrm{coeff}_P(f, i) t_P^i,$$

where t_P is a local parameter in P.

Remark 3.3.20. Note that $v_P(f)$ is the minimum index i such that $\mathrm{coeff}_P(f, i) \neq 0$.

Proof. We know that $GS(F) \simeq \overline{GS}(F)$, where $\overline{GS}(F)$ is defined in (3.22). In particular, by Remark 3.3.8, for an element (λ_0, λ_1) to belong to $\overline{GS}(F)$, we need

$$v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) \geq -v_P(F) + 1$$

for any point $P \in \mathrm{supp}(D_{\boldsymbol{e}})$. Let us assume $\mathrm{supp}(D_{\boldsymbol{e}}) = \{P_{i_1}, \ldots, P_{i_t}\}$. For any $j \in \{1, \ldots, t\}$, we introduce the linear map $\psi_j : L(F) \times L(F + G) \longrightarrow \mathbb{F}_q$ such that for any $(\lambda_0, \lambda_1) \in L(F) \times L(F + G)$,

$$\psi_j(\lambda_0, \lambda_1) \stackrel{\mathrm{def}}{=} \mathrm{coeff}_{P_{i_j}}(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1, -v_{P_{i_j}}(F)).$$

Finally, we can construct $\Psi : L(F) \times L(F + G) \longrightarrow \mathbb{F}_q^t$, such that for any $(\lambda_0, \lambda_1) \in L(F) \times L(F + G)$

$$\Psi(\lambda_0, \lambda_1) = (\psi_1(\lambda_0, \lambda_1), \ldots, \psi_t(\lambda_0, \lambda_1)).$$

It is easy to see that Ψ is a linear map and $\overline{GS}(F) = \ker(\Psi)$. In order to prove the statement $\dim(\overline{GS}(F)) = \ell(F) + \ell(F+G) - t$, we show that $\mathrm{Im}(\Psi) = \mathbb{F}_q^t$. Since by hypothesis $\deg(F+G) \geq t+2g-1$, by Corollary 1.7.51, we have

$$L(F + G - D_{\boldsymbol{e}} + P_{i_j}) \neq L(F + G - D_{\boldsymbol{e}}) \quad \forall j = 1, \ldots, t.$$

In particular for any point $P_{i_j} \in \mathrm{supp}(D_{\boldsymbol{e}})$ there exists $\mu_j \in L(F+G)$ such that

$$\begin{aligned}\mathrm{coeff}_{P_{i_j}}(\mu_j, -v_{P_{i_j}}(F)) &= 1\\ \mathrm{coeff}_{P_{i_h}}(\mu_j, -v_{P_{i_h}}(F)) &= 0 \quad \forall h \neq j.\end{aligned}$$

Therefore we have

$$\begin{aligned}\Psi(0,\mu_1) &= (\mathrm{coeff}_{P_{i_1}}(\mu_1), \ldots, \mathrm{coeff}_{P_{i_t}}(\mu_1)) = \ (1, 0 \ldots, 0)\\ &\cdots \qquad\qquad\qquad\qquad\qquad \cdots\\ \Psi(0,\mu_t) &= (\mathrm{coeff}_{P_{i_1}}(\mu_t), \ldots, \mathrm{coeff}_{P_{i_t}}(\mu_t)) = \ (0 \ \ldots, 0, 1),\end{aligned}$$

that is, $\mathrm{span}\{\Psi(0,\mu_1), \ldots, \Psi(0,\mu_t)\} = \mathbb{F}_q^t$. □

Corollary 3.3.21. *If* $\deg(F+G) \geq t + 2g$*, then for any point* $P \in \mathcal{P}$ *we have*

$$\dim GS(F-P) = \begin{cases} \dim GS(F) - 1 & \text{if } \ell(F-P) = \ell(F)\\ \dim GS(F) - 2 & \text{if } \ell(F-P) \neq \ell(F).\end{cases}$$

Proof. By hypothesis, $\deg(F+G) \geq t + 2g > 2g - 2$, hence we get

$$\begin{aligned}\dim GS(F-P) &= \ell(F-P) + \ell(F+G-P) - t\\ &= \ell(F-P) + \ell(F+G) - 1 - t.\end{aligned}$$

Now, one can notice that if $\ell(F-P) = \ell(F)$, then $\dim GS(F-P) = \dim GS(F) - 1$. Otherwise, by Proposition 1.7.50-*(i)*, we have $\ell(F-P) = \ell(F) - 1$ and then $\dim GS(F-P) = \dim GS(F) - 2$. □

Thanks to this result, we have that the dimension of $GS(F)$ cannot decrease by more than 2 and that this gap depends strictly on $L(F-P)$. The following two lemmas will add some more elements to fully describe these dimension gaps.

Lemma 3.3.22. *If* $P \in \mathrm{supp}(D_{\boldsymbol{e}})$*, then* $GS(F,P) = GS(F,P,P)$.

Proof. Let $P \in \mathrm{supp}(D_{\boldsymbol{e}})$. It is clear that $GS(F,P,P) \subseteq GS(F,P)$. Now, given $(\lambda_0, \lambda_1, \lambda_2) \in GS(F,P)$, we want to prove that $\lambda_1 \in L(F+G-P)$ or equivalently that $v_P(\lambda_1) \geq -v_P(F) + 1$. By definition of $GS(F,P,P)$, we have

$$\begin{cases}\lambda_0 f_{\boldsymbol{c}}^2 + \lambda_1 f_{\boldsymbol{c}} + \lambda_2 = 0\\ \lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 \in L(F + 2G' - D).\end{cases}$$

Thus, subtracting these quantities, we get $\lambda_0(2f_{\boldsymbol{c}}f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}} \in L(F + 2G' - D)$. In particular, since $f_{\boldsymbol{e}} \in L(G' - D + D_{\boldsymbol{e}})$ and $f_{\boldsymbol{e}}(P) \neq 0$ for any $P \in \mathrm{supp}(D_{\boldsymbol{e}})$, we have

$$v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) = v_P(\lambda_0(2f_{\boldsymbol{c}}f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}}) \geq -v_P(F) + 1. \tag{3.32}$$

Now since $\lambda_0 \in L(F-P)$, using Proposition 1.7.27 we have that

$$v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) \geq \min\{v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}})), v_P(\lambda_1)\}$$

where the equality holds if $v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}})) \neq v_P(\lambda_1)$. Now, assume $v_P(\lambda_1) = -v_P(F)$ by way of contradiction. Then,

$$\begin{cases}v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}})) \geq -v_P(F) + 1\\ v_P(\lambda_1) = -v_P(F)\end{cases} \implies v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) = -v_P(F) \tag{3.33}$$

which is in contradiction with (3.32). We get then $GS(F,P) = GS(F,P,P)$. □

Lemma 3.3.23. *If* $P \notin \mathrm{supp}(D_{\boldsymbol{e}})$*, then* $GS(F,P,P) = GS(F-P)$.

Proof. Let $P \notin \operatorname{supp}(D_{\boldsymbol{e}})$. It is clear that $GS(F-P) \subseteq GS(F,P,P)$. For the other inclusion, by Proposition 3.3.17, it is equivalent to show that

$$GS(F) \cap (L(F-P) \times L(F+G-P) \times L(F+2G-P)) \subseteq GS(F-P). \tag{3.34}$$

We consider then an element $(\lambda_0, \lambda_1, \lambda_2)$ belonging to the left hand space in (3.34) and we prove that

$$\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 \in L(F - P + 2G' - D).$$

Again, by assumption $(\lambda_0, \lambda_1, \lambda_2) \in GS(F)$, we know that $\lambda_0 f_{\boldsymbol{y}}^2 + \lambda_1 f_{\boldsymbol{y}} + \lambda_2 = \lambda_0(2f_{\boldsymbol{c}} f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}}$. In particular, since $f_{\boldsymbol{e}}$ belongs to $L(G' - D + D_{\boldsymbol{e}})$, then $v_P(f_{\boldsymbol{e}}) \geq 1$ and we have

$$v_P(\lambda_0(2f_{\boldsymbol{c}} f_{\boldsymbol{e}} + f_{\boldsymbol{e}}^2) + \lambda_1 f_{\boldsymbol{e}}) \geq 1 + v_P(\lambda_0(2f_{\boldsymbol{c}} + f_{\boldsymbol{e}}) + \lambda_1) \geq -v_P(F) + 2.$$

□

Given a point $P \in \mathcal{P}$, if $\deg(F+G) \geq t + 2g$, thanks to Corollary 3.3.21, Lemma 3.3.22 and Lemma 3.3.23, we can then summarize the possible cases as follows:

	$GS(F-P)$	$\subseteq$	$GS(F,P,P)$	$\subseteq$	$GS(F,P)$	$\subseteq$	$GS(F)$
$\ell(F-P) = \ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$=$		$\neq$		$=$
$\ell(F-P) \neq \ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$=$		$\neq$		$\neq$
$\ell(F-P) = \ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\neq$		$=$		$=$
$\ell(F-P) \neq \ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\neq$		$=$		$\neq$

Figure 3.1: Dimension gaps for $GS(F)$ with $\deg(F+G) \geq t + 2g$.

We want to prove now that, if $\deg(F+G) \geq t+2g$, there exists a point $P \in \mathcal{P}$ such that $\dim AS(F-P)$ decreases faster than $\dim GS(F-P)$. We will focus in particular on the points P such that $P \in \operatorname{supp}(D_{\boldsymbol{e}})$. It is easy to verify that, any time we have a strict inclusion for the GS's, then we have one also for the AS's. Hence, for $AS(F)$ the situation can be described as follows:

	$AS(F-P)$	$\subseteq$	$AS(F,P,P)$	$\subseteq$	$AS(F,P)$	$\subseteq$	$AS(F)$
$\ell(F-P) = \ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\subseteq$		$\neq$		$=$
$\ell(F-P) \neq \ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\subseteq$		$\neq$		$\neq$
$\ell(F-P) = \ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\neq$		$\subseteq$		$=$
$\ell(F-P) \neq \ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) \implies$			$\neq$		$\subseteq$		$\neq$

Figure 3.2: Dimension gaps for $AS(F)$ with $\deg(F+G) \geq t + 2g$.

Thanks to this description of the inclusions between $AS(F)$ and $AS(F-P)$, one can see that, whenever $\deg(F+G) \geq t+2g$ and $P \in \operatorname{supp}(D_{\boldsymbol{e}})$, P narrows the gap for F if $AS(F,P,P) \neq AS(F,P)$. The following result, shows that such a point exists.

Lemma 3.3.24. *There exist at most* $\dim Z_2 + g$ *points* $P \in \operatorname{supp}(D_{\boldsymbol{e}})$ *such that*

$$AS(F,P,P) = AS(F,P). \tag{3.35}$$

Proof. Without loss of generality, we can suppose the points of the support of $D_{\boldsymbol{e}}$ satisfying 3.35 to be $\{P_1, \ldots, P_m\}$. Let us define $\tilde{D} = \sum_{i=1}^m P_i$. We have

$$AS\left(F, \tilde{D}, \tilde{D}\right) = \bigcap_{i=1}^m AS(F, P_i, P_i) = \bigcap_{i=1}^m AS(F, P_i) = AS\left(F, \tilde{D}\right).$$

Hence we have

$$\begin{aligned}\ell(F-\tilde{D})+\ell(F+G)-\dim Z_2 &\leq \dim AS(F,\tilde{D})\\ &\leq \dim AS(F,\tilde{D},\tilde{D})\\ &\leq \ell(F-\tilde{D})+\ell(F+G-\tilde{D})\\ &\leq \ell(F-\tilde{D})+\ell(F+G)-\ell(\tilde{D})+1\\ &\leq \ell(F-\tilde{D})+\ell(F+G)-m+g.\end{aligned}$$

Hence $m \leq \dim Z_2 + g$ by Proposition 3.3.7. □

Remark 3.3.25. Note that, if $\deg(F+G) \geq t+2g$ and $t > \frac{n-\deg G-1}{2}$, then there exists a point P in $\mathrm{supp}(D_{\boldsymbol{e}})$ such that $AS(F,P,P) \neq AS(F,P)$. Indeed we get $\dim Z_2 + g < t$ whenever $t > \frac{n-\deg G-1}{2}$.

Proposition 3.3.26. *If* $\deg(F+G) \geq t+2g$ *and* $t > \frac{\mathrm{d}^*-1}{2}$ *then one and only one of the following statements holds:*

(i) $AS(F) = GS(F)$*;*

(ii) there exists $P \in \mathrm{supp}(D)$ *such that* $\dim AS(F) - \dim AS(F-P) > \dim GS(F) - \dim GS(F-P)$.

Proof. If $AS(F) = GS(F)$, then by Proposition 3.3.19, we have $\dim AS(F-P) \geq \dim AS(F) - 2$. Now let us prove that if $AS(F) \neq GS(F)$, then there exists a point P such that $\dim AS(F-P)$ decreases faster than $\dim GS(F-P)$. By Lemma 3.3.24 and Remark 3.3.25, we have that

$$t - (n - \deg F - 2\deg G + 2g - 1) \geq 1.$$

Hence, there exists a point $P \in \mathrm{supp}(D_{\boldsymbol{e}})$ such that

$$AS(F,P,P) \subsetneq AS(F,P),$$

that is, $\dim AS(F) - \dim AS(F-P) > \dim GS(F) - \dim GS(F-P)$. □

We now want to prove that this algorithm is a list decoder. As Sudan's algorithm, this result provides another proof that the size of the list up to this decoding radius has to be the degree of the considered polynomials.

Theorem 3.3.27. *Suppose* $\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \boldsymbol{c}_2 + \boldsymbol{e}_2$ *and let* $GS_1(F)$ *and* $GS_2(F)$ *be the spaces of good Sudan polynomials relative respectively to* $\boldsymbol{c}_1$ *and* $\boldsymbol{c}_2$*. If* $AS(F) = GS_1(F)$ *and* $\deg(F+G) \geq t+2g$*, then* $AS(F) = GS_2(F)$.

Proof. We first prove that if P is a point which satisfies (ii) of Proposition 3.3.26 for $GS_1(F)$, then it satisfies it also for $GS_2(F)$. Then this fact is clearly verified if $P \in \mathrm{supp}(D_{\boldsymbol{e}_1}) \cap \mathrm{supp}(D_{\boldsymbol{e}_2})$ or $P \notin \mathrm{supp}(D_{\boldsymbol{e}_1}) \cup \mathrm{supp}(D_{\boldsymbol{e}_2})$, since the inclusions for $GS_1(F)$ and $GS_2(F)$ are described by the same case in Figure 3.1. Let us now suppose that $P \in \mathrm{supp}(D_{\boldsymbol{e}_1}) \setminus \mathrm{supp}(D_{\boldsymbol{e}_2})$ and that $L(F-P) = L(F)$ (it is possible to deduce the case $L(F-P) \neq L(F)$ following the same idea). Since P satisfies (ii) of Proposition 3.3.26 for $GS_1(F)$, by Figure 3.2, we have

$$AS(F-P) \neq AS(F,P,P) \neq AS(F,P) = AS(F).$$

Though, since $P \notin \mathrm{supp}(D_{\boldsymbol{e}_2})$ and $L(F-P) = L(F)$, by Figure 3.1 we get

$$GS_2(F-P) = GS_2(F,P,P) \neq GS_2(F,P) = GS_2(F),$$

that is, P fulfills (ii) of Proposition 3.3.26 for $GS_2(F)$. Then, since by Proposition 3.3.26 only one of the two statements holds and it holds at the same time for $GS_1(F)$ and $GS_2(F)$, we have the implication $AS(F) = GS_1(F) \implies AS(F) = GS_2(F)$. □

Note that the results above hold only for $\deg(F+G) \geq t+2g$. In particular, if the algorithm makes an amount of steps such that $\deg(F_j+G) < t+2g$ for some j, then we no longer have the proof that, whenever $AS(F) \neq GS(F)$, a point narrowing the gap for F always exists. The number of steps of the algorithm which are necessary to have $AS(F) = GS(F)$, can be estimated as follows.

Lemma 3.3.28. *The number of steps of the algorithm is bounded by* $\dim AS(F) - \dim GS(F) \leq t$.

Proof. By Proposition 3.3.7, we have $\dim AS(F) \leq \ell(F)+\ell(F+G)$ and $\dim GS(F) \geq \ell(F)+\ell(F+G)-t$, hence we get $\dim AS(F) - \dim GS(F) \leq t$. □

Remark 3.3.29. We recall that, to estimate the dimensions of $AS(F)$ and $GS(F)$, we assume

$$\deg(F_j + 2G) < n \quad \forall j \geq 0.$$

If one sets F_0 with $\deg F_0 = n - 1 - 2\deg G$, then if $6g + 1 \leq n$ and $t \leq t_{max}(2)$, we get

$$t \leq \frac{2n - 3\deg G - 2}{3} \leq n - 1 - \deg G - 2g,$$

that is, $\deg(F_0 + G) \geq t + 2g$. In particular, by Proposition 3.3.19, we have

$$\dim GS(F) = \ell(F) + \ell(F + G) - t,$$

that is, it is possible to compute the dimension of $GS(F)$. Therefore, at the beginning of the algorithm one can compute in advance the maximum number of steps required by the algorithm.

The algorithm becomes then

Algorithm 8 New algorithm - list decoding

Inputs: $f_{\boldsymbol{y}} = f_{\boldsymbol{c}} + f_{\boldsymbol{e}} \in \mathbb{F}_q^n$ where $\boldsymbol{c} \in \mathcal{C}$ and $t = \mathrm{w}(\boldsymbol{e}) \leq t_{max}(2) = \frac{2n-3\deg G-2}{3}$.
Output: $\{f \in L(G) \mid \mathrm{d}(\mathrm{ev}_{\mathcal{P}}(f), \boldsymbol{y}) \leq t\}$.

```
Choose F with deg F = n − 1 − 2 deg G;
j ← 0 and F_0 ← F;
Δ ← dim AS(F) − dim GS(F);
while j < Δ do
    Look for a point P ∈ {P_1, ..., P_n} which narrows the gap for F_j;
    if such a point P exists then
        F_{j+1} ← F_j − P;
        j ← j + 1;
    else return "?";
p ← nonzero element of AS(F);
return {f | f is a rooth of p and d(ev_P(f), y) ≤ t};
```

How to understand if a point P narrows the gap for F? We did not find a complete characterization of the point narrowing the gap for a divisor F. Though the following two results permit to identify some of them.

Proposition 3.3.30. *Suppose* $\deg(F + G) \geq t + 2g$. *Then, the two following statements hold*

(i) if $\ell(F - P) = \ell(F)$, *then* $\dim AS(F) - \dim AS(F - P) = 2 \iff P$ *narrows the gap for* F;

(ii) if $\ell(F - P) \neq \ell(F)$, *then* $\dim AS(F) - \dim AS(F - P) = 3 \iff P$ *narrows the gap for* F.

Proof. The proof can be easily deduced by Figure 3.1 and Figure 3.2. □

If $\deg(F + G) < t + 2g$, we have a weaker statement.

Lemma 3.3.31. *The following two statements hold:*

(i) if $\ell(F - P) = \ell(F)$, *then* $\dim AS(F) - \dim AS(F - P) = 2 \implies P$ *narrows the gap for* F;

(ii) if $\ell(F - P) \neq \ell(F)$, *then* $\dim AS(F) - \dim AS(F - P) = 3 \implies P$ *narrows the gap for* F.

Proof. We report here all possible cases for the inclusions between $GS(F)$ and $GS(F-P)$.

$$\begin{array}{lccccc} & GS(F-P) & \subseteq & GS(F,P,P) & \subseteq GS(F,P) & \subseteq GS(F) \\ \ell(F-P)=\ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) & \Longrightarrow & & = & \subseteq & = \\ \ell(F-P)\neq\ell(F) \wedge P \notin \operatorname{supp}(D_{\boldsymbol{e}}) & \Longrightarrow & & = & \subseteq & \subseteq \\ \ell(F-P)=\ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) & \Longrightarrow & & \subseteq & = & = \\ \ell(F-P)\neq\ell(F) \wedge P \in \operatorname{supp}(D_{\boldsymbol{e}}) & \Longrightarrow & & \subseteq & = & \subseteq \end{array}$$

Figure 3.3: Dimension gaps for $GS(F)$.

It is clear then, that if $\ell(F-P)=\ell(F)$ we have $\dim GS(F) - \dim GS(F-P) \leq 1$. Note that, since $\ell(F-P)=\ell(F)$, we have $AS(F)=AS(F,P)$, hence we cannot have $\dim AS(F) - \dim AS(F-P) = 3$. Though, if $\dim AS(F) - \dim AS(F-P) = 2$, then P narrows the gap for F. On the other hand, if $\ell(F-P)\neq\ell(F)$, to ensure that P narrows the gap, we need $\dim AS(F) - \dim AS(F-P) = 3$. □

3.3.5 Some empirical considerations

Let us consider, for our tests, an algebraic geometry code $\mathcal{C}=\mathcal{C}_L(\mathcal{X},\mathcal{P},G)$ and a vector $\boldsymbol{y}=\boldsymbol{c}+\boldsymbol{e}$, where $\boldsymbol{c}\in\mathcal{C}$ and $t=\mathrm{w}(\boldsymbol{e})=t_{max}(2)$, that is,

$$t=\frac{2n-3\deg G-2}{3}.$$

The curve we run our tests on, are the following:

1. $X^6+Y^6+XZ^5=0$ in $\mathbb{F}_{7^3}$;
2. $Y^5Z-X^6-XZ^5-Z^6=0$ in $\mathbb{F}_{11^3}$.

The reader can find some results of these tests in Table 3.2. We indicate by t_{Sud} the decoding radius of Sudan algorithm with $\ell=2$

$$t_{Sud}\stackrel{\text{def}}{=}\frac{2n-3\deg G-2-2g}{3}$$

and by Δ the upper bound for the number of steps of the algorithm, that is,

$$\Delta\stackrel{\text{def}}{=}\dim AS(F_0)-GS(F_0).$$

We recall that, by Remark 3.3.8 it is possible to compute Δ at the beginning of the algorithm. Finally the reader can find in Table 3.2 the values of the genus of the curve g, the degree of the initial divisor F_0 and can check whether the points which narrow the gap belong or not to $\operatorname{supp}(D_{\boldsymbol{e}})$. In the last column, we indicate if the algorithm recovers $\boldsymbol{c}$.

q	$\mathcal{X}$	g	n	$\deg G$	$\deg F_0$	$\frac{\mathrm{d}^*-1}{2}$	t_{Sud}	Δ	t	pts$\subseteq \operatorname{supp}(D_{\boldsymbol{e}})$	finds $\boldsymbol{c}$
7^3	1	10	200	40	$n-1-2\deg G$	79	86	82	92	false	true
11^3	2	10	200	38	$n-1-2\deg G$	80	86	85	94	false	true
7^3	1	10	200	40	$2t+g-2\deg G$	79	86	77	92	false	true
7^3	1	10	200	40	$t+g$	79	86	65	92	false	true
11^3	2	10	200	38	$t+g$	80	86	65	94	false	true

Table 3.2: List decoding algorithm for algebraic geometry codes.

Note that, empirically the algorithm works. As for Ehrhard's algorithm, given a divisor F_i of the sequence, it is rare to find a point P which does not narrow the gap for F_i. In particular, most of the times, the last divisor of the sequence constructed by the algorithm F_{fin}, and hence satisfying

$$AS(F_{fin})=GS(F_{fin}),$$

is of the form $F = F_0 - MP_1$ for some $M \leq t$.

We now present an empirical observation on the parameters on the algorithm. As said, by Remark 3.3.30 if $\deg F_0$ is large enough, then we know exactly how to recognize the points which narrow the gap between $AS(F)$ and $GS(F)$. However, in the tests we conducted, the sequence of divisors is long enough to have a j such that $\deg F_j + \deg G \leq t + 2g$. Anyway, in these cases the points narrowing the gap are easy to recognize, as we always find a point P such that

$$\dim AS(F) - \dim AS(F - P) = 3.$$

The number of steps Δ of the algorithm results empirically less than t. In particular, if F_{fin} is the final divisor of the sequence, that is, $AS(F_{fin}) = GS(F_{fin})$, we always get

$$\deg F_{fin} = \deg F_0 - \Delta \geq \frac{t_{max}(2) + 1}{2} - \frac{\deg G}{2} + g - 1,$$

which, by Proposition 3.3.7, ensures $GS(F_{fin}) \neq \{0\}$. In particular, we also tried to run the algorithm with values of $\deg F_0$ relatively small with respect to $n-1-2\deg G$. In this case Δ and, in particular, the number of steps, adjusts automatically for the algorithm to find $GS(F_{fin}) \neq \{0\}$. This fact suggests that there could be a more accurate way to estimate the gap between $AS(F)$ and $GS(F)$ before running the algorithm and hence, with no help from Magma to compute $\dim GS(F)$. In the following, there are some results of tests run on vectors of the form $\boldsymbol{y} = \boldsymbol{c}_1 + \boldsymbol{e}_1 = \boldsymbol{c}_2 + \boldsymbol{e}_2$, where $\mathrm{w}(\boldsymbol{e}_2) \leq t = \mathrm{w}(\boldsymbol{e}_1) = t_{max}(2)$ and $\mathrm{supp}(\boldsymbol{e}_1) \cap \mathrm{supp}(\boldsymbol{e}_2) = \emptyset$. Note that we proved that if the algorithm terminates with $\deg(F_{fin}+G) \geq t+2g$, then it finds both the solutions $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ (see Theorem 3.3.27). Though, since in our tests, we have for a certain j that $\deg(F_j + G) < t + 2g$, *a priori* it is not guaranteed that the algorithm can recover the two of them.

q	$\mathcal{X}$	g	n	$\deg G$	$\deg F_0$	$\frac{\mathrm{d}^* - 1}{2}$	t_{Sud}	Δ	t	pts$\subseteq$ supp$(D_{\boldsymbol{e}})$	finds $\{\boldsymbol{c}_1, \boldsymbol{c}_2\}$
11^3	2	10	200	46	$n-1-2\deg G$	76	80	75	86	false	true
11^3	2	10	200	38	$n-1-2\deg G$	80	88	83	94	false	true
11^3	2	10	200	38	70	76	80	32	86	false	true

Table 3.3: List decoding algorithm for algebraic geometry codes with two solutions.

From the results of Table 3.3, one can note that the algorithm does recover both the solutions $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. An argument to explain that, is the following: let us consider $GS_1(F)$ and $GS_2(F)$ which are respectively the spaces of good Sudan polynomials with respect to $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. At the beginning of the algorithm we compute $\Delta = \dim AS(F) - \dim GS_1(F)$. Now, since $\mathrm{w}(\boldsymbol{e}_2) \leq \mathrm{w}(\boldsymbol{e}_1)$, using Proposition 3.3.19 for both $GS_1(F)$ and $GS_2(F)$, we get

$$\dim AS(F) - \dim GS_2(F) \leq \dim AS(F) - \dim GS_1(F).$$

In particular, if at each step $\dim AS(F_j) - \dim AS(F_j - P) = 3$, it is sure that once $AS(F_{fin}) = GS_1(F_{fin})$ is verified, then we get also $AS(F_{fin}) = GS_2(F_{fin})$.

Remark 3.3.32. In our tests we already know the solutions we look for. In particular then, we can compute the space $GS(F)$ and see how it behaves when the divisor $F - P$ is considered. In the tests where the solution is unique, at each step, we get

$$\dim GS(F_j) - \dim GS(F_j - P) = 2.$$

In the case where two solutions exist, we compute $GS_1(F)$, $GS_2(F)$ and also $GS_1(F) \cap GS_2(F)$. Whenever $GS_1(F_j - P) \neq GS_2(F_j - P)$, we note in particular the following changes of dimension

$$\dim GS_1(F_j) - \dim GS_1(F_j - P) = 2, \quad \dim GS_2(F_j) - \dim GS_2(F_j - P) = 2,$$
$$\dim GS_1(F_j) \cap GS_2(F_j) - \dim GS_1(F_j - P) \cap GS_2(F_j - P) = 1.$$

In particular at each step, $GS_1(F_j)$ and $GS_2(F_j)$ get closer to $GS_1(F_j) \cap GS_2(F_j)$. Thus, as soon as we get the equality $GS_1(F_j) = GS_2(F_j) = GS_1(F_j) \cap GS_2(F_j)$, we have

$$\dim GS_1(F_j) - \dim GS_1(F_j - P) = \dim GS_2(F_j) - \dim GS_2(F_j - P) = 1.$$

We point out that we have the same changes of dimension for points $P \notin \text{supp}(\boldsymbol{e}_1) \cup \text{supp}(\boldsymbol{e}_2)$ and points $P \in \text{supp}(\boldsymbol{e}_1) \cap \text{supp}(\boldsymbol{e}_2)^c$.

Conclusion

We conclude with some future prospects for the works presented in this thesis. We have seen that the power error locating pairs algorithm disposes, as the error correcting pairs algorithm, from a certain versatility. Hence, it would be interesting to find other linear codes, disposing from a power error locating pairs, to improve the correction capacity for them. Moreover, a possible improvement of this result, concerns the correction capacity of PELP algorithm for Reed–Solomon and, more in general, algebraic geometry codes. Indeed, since there exists an equivalence between this algorithm and the power decoding algorithm (equivalence which is proved for Reed–Solomon codes), and the power decoding has been extended, getting its decoding radius to achieve the Johnson bound (see [Joh62, Ros18]), then we estimate that it may be possible to design a multiplicity version as well of the power error locating pairs algorithm. This generalization could then provide an attack for the McEliece encryption scheme with algebraic geometry codes [JM96] with amounts of errors up to the Johnson bound and, hence, break the scheme proposed in [ZZ18].
About the algorithm presented in Chapter 3, there are several points to improve: first it would be important to prove that, whenever the space of *Good Sudan polynomials* $GS(F)$ and the one of *All Sudan polynomials* $AS(F)$ differ, a point narrowing the gap always exists. Secondly, it would be important to provide a method to deduce if a point P narrows the gap for F, with the very knowledge of $\dim AS(F)$ and $\dim AS(F-P)$. Finally, in order to reduce as possible the number of the steps of the algorithm, it would be interesting to give a better estimation of the dimension of $AS(F)$ and, hence, of the quantity $\dim AS(F) - \dim GS(F)$. Finally, with the same idea, it could be interesting to adapt this construction to Guruswami–Sudan algortihm [GS99] and, in this way, erase the genus penalty $\frac{g}{s}$, where s is the multiplicity parameter, from its decoding radius.

Bibliography

[BH08] Peter Beelen and Tom Høholdt. The decoding of algebraic geometry codes. In *Advances in algebraic geometry codes*, volume 5 of *Ser. Coding Theory Cryptol.*, pages 49–98. World Sci. Publ., Hackensack, NJ, 2008.

[BL17] Vincent Beck and Cédric Lecouvey. Additive combinatorics methods in associative algebras. *Confluentes Math.*, 9(1):3–27, 2017.

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inform. Theory*, 24(3):384–386, 1978.

[CH15] Henry Cohn and Nadia Heninger. Ideal forms of Coppersmith's theorem and Guruswami-Sudan list decoding. *Advances in Mathematics of Communications*, 9(3):311–339, 2015.

[CMCP17] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Trans. Inform. Theory*, 63(8):5404–5418, August 2017.

[COT17] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial time attack on wild McEliece over quadratic extensions. *IEEE Trans. Inform. Theory*, 63(1):404–427, Jan 2017.

[DK94] Iwan M. Duursma and Ralf Kötter. Error-locating pairs for cyclic codes. *IEEE Trans. Inform. Theory*, 40(4):1108–1121, July 1994.

[Duu93] Iwan M. Duursma. Algebraic decoding using special divisors. *IEEE Trans. Inform. Theory*, 39(2):694–698, 1993.

[Ehr92] Dirk Ehrhard. Decoding algebraic-geometric codes by solving a key equation. In Henning Stichtenoth and Michael A. Tsfasman, editors, *Coding Theory and Algebraic Geometry*, pages 18–25, Berlin, Heidelberg, 1992.

[Ehr93] Dirk Ehrhard. Achieving the designed error capacity in decoding algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 39(3):743–751, 1993.

[FM08] Cédric Faure and Lorenz Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99–107, Pamporovo, Bulgaria, June 2008.

[FR93] G. L. Feng and T. R. N. Rao. Decoding algebraic-geometric codes up to the designed minimum distance. *IEEE Trans. Inform. Theory*, 39(1):37–45, 1993.

[Ful69] W. Fulton. *Algebraic curves.* Mathematics Lecture Note Series. New York-Amsterdam: W.A. Benjamin, Inc. XIII, 226 p. (1969), 1969.

[Gop81] V. D. Goppa. Codes on algebraic curves. *Dokl. Akad. Nauk SSSR*, 259:1289–1290, 1981.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed–Solomon and Algebraic–Geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999.

[GS00] Shuhong Gao and M. Amin Shokrollahi. Computing Roots of Polynomials over Function Fields of Curves. In David Joyner, editor, *Coding Theory and Cryptography*, pages 214–228, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[HJ99] Agnes Eileen Heydtmann and Jørn Møller Jensen. *On the Equivalence of the Berlekamp-Massey and the Euclidean Algorithm for Decoding*. 1999.

[HP95] Tom Høholdt and Ruud Pellikaan. On the decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 41(6):1589–1614, Nov 1995.

[HvLP] Thomas Høholdt, Jacobus H. van Lint, and Ruud Pellikaan. Algebraic geometry codes. In *Handbook of Coding Theory*, volume 1. V.S. Pless, W.C. Huffman and R.A. Brualdi Eds., Elsevier, Amsterdam 1998.

[JH04] Jørn Justesen and Tom Høholdt. *A Course in Error-Correcting Codes*. European Mathematical Society Publishing House, first edition, 2004.

[JLJ+89] J. Justesen, K. J. Larsen, H. E. Jensen, A. Havemose, and T. Høholdt. Construction and decoding of a class of algebraic geometry codes. *IEEE Trans. Inform. Theory*, 35(4):811–821, 1989.

[JM96] Heeralal Janwa and Oscar Moreno. McEliece public key cryptosystem using algebraic-geometry codes. *Des. Codes Cryptogr.*, 8:293–307, 1996.

[Joh62] Selmer M. Johnson. A new upper bound for error-correcting codes. *IRE Trans. Inform. Theory*, 8(3):203–207, April 1962.

[Köt92] Ralf Kötter. A unified description of an error locating procedure for linear codes. In *Proceedings Algebraic and Combinatorial Coding Theory III*, pages 113–117. Hermes, 1992.

[KP95] C. Kirfel and R. Pellikaan. The minimum distance of codes in an array coming from telescopic semigroups. *IEEE Transactions on Information Theory*, 41(6):1720–1732, 1995.

[KS80] Wolfgang Knapp and Peter Schmid. Codes with prescribed automorphism group. *J. Algebra*, 67(2):415–435, 1980.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[McE03a] Robert J. McEliece. On the Average List Size for the Guruswami-Sudan decoder. In *7th Inernational Symposium on Communications Theory and Applications (ISCTA)*, 2003.

[McE03b] Robert J. McEliece. The Guruswami–Sudan decoding algorithm for Reed–Solomon codes. *Interplanetary Network Progress Report*, 153:1–60, January 2003.

[Min07] Lorenz Minder. *Cryptography based on error correcting codes*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2007.

[Mum70] David Mumford. Varieties defined by quadratic equations. In *Questions on algebraic varieties, C.I.M.E., III Ciclo, Varenna, 1969*, pages 29–100. Edizioni Cremonese, Rome, 1970.

[MZ15] Diego Mirandola and Gilles Zémor. Critical pairs for the product Singleton bound. *IEEE Trans. Inform. Theory*, 61(9):4928–4937, 2015.

[NB15] Johan S. R. Nielsen and Peter Beelen. Sub–quadratic decoding of one-point Hermitian codes. *IEEE Trans. Inform. Theory*, 61(6):3225–3240, June 2015.

[Nie14] Johan S. R. Nielsen. Fast Kötter-Nielsen-høholdt interpolation in the Guruswami-Sudan algorithm, 2014. International Workshop on Algebraic and Combinatorial Coding Theor.

[Pel89] Ruud Pellikaan. On a decoding algorithm for codes on maximal curves. *IEEE Trans. Inform. Theory*, 35:1228 – 1232, December 1989.

[Pel92] Ruud Pellikaan. On decoding by error location and dependent sets of error positions. *Discrete Math.*, 106–107:369–381, 1992.

[Pel93] Ruud Pellikaan. On the efficient decoding of algebraic-geometric codes. In *Eurocode'92 (Udine, 1992)*, volume 339 of *CISM Courses and Lectures*, pages 231–253. Springer, Vienna, 1993.

[Por88] Sidney C. Porter. *Decoding codes arising from Goppa's construction on algebraic curves.* PhD thesis, Yale Univ., December 1988.

[PRB19] Sven Puchinger, Johan Rosenkilde, and Irene Bouw. Improved power decoding of interleaved one–point Hermitian codes. *Des. Codes Cryptogr.*, 87:589–607, 2019.

[PRS21] Sven Puchinger, Johan Rosenkilde, and Grigory Solomatov. Improved power decoding of algebraic geometry codes. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 509–514, 2021.

[PSP92] Sidney C. Porter, B.Z. Shen, and Ruud Pellikaan. Decoding geometric Goppa codes using an extra place. *IEEE Trans. Inform. Theory*, 38(6):1663–1676, 1992.

[RnN15] Johan Rosenkilde (né Nielsen). Power decoding of Reed–Solomon codes revisited. In *Coding Theory and Applications*, pages 297–305, Cham, 2015. Springer International Publishing.

[Roo83] Cornelis Roos. A new lower bound for the minimum distance of a cyclic code. *IEEE Trans. Inform. Theory*, 29:330 – 332, June 1983.

[Ros18] Johan Sebastian Rosenkilde. Power decoding Reed–Solomon codes up to the Johnson radius. *Adv. Math. Commun.*, 12:81–106, 2018.

[Rot06] Ron M. Roth. *Introduction to coding theory.* Cambridge university press, 2006.

[SS92] Vladimir Michilovich Sidelnikov and S.O. Shestakov. On the insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Math. Appl.*, 1(4):439–444, 1992.

[SSB09] Georg Schmidt, Vladimir R. Sidorenko, and Martin Bossert. Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs. *IEEE Trans. Inform. Theory*, 55(7):2991–3012, July 2009.

[SSB10] Georg Schmidt, Vladimir R. Sidorenko, and Martin Bossert. Syndrome Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis. *IEEE Trans. Inform. Theory*, 56(10):5245–5252, October 2010.

[Sti09] Henning Stichtenoth. *Algebraic Function Fields and Codes.* Springer Publishing Company, Incorporated, 2nd edition, 2009.

[Sud97] Madhu Sudan. Decoding of Reed–Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.

[SV90] Alexei Skorobogatov and Serge Vlăduţ. On the decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 36:1051 – 1060, October 1990.

[SW99] M. Amin Shokrollahi and Hal Wasserman. List decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 45(2):432–437, March 1999.

[TV06] T. Tao and V. H. Vu. *Additive combinatorics*, volume 105 of *Cambridge studies in advanced mathematics.* Cambridge University Press, 2006.

[TVN07] Michael Tsfasman, Serge Vlăduţ, and Dmitrii Nogin. *Algebraic Geometric Codes: Basic Notions.* January 2007.

[TVZ82] M. Tsfasman, S. G. Vlădut, and T. Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Mathemat. Nachricht.*, 109:21–28, 1982.

[Vl0] Serge Vlădut. On the decoding of algebraic-geometric codes over $\mathbb{F}_q$ for $q \geq 16$. *IEEE Trans. Inform. Theory*, 36(6):1461–1463, 1990.

[vLvdG88] J. van Lint and G. van der Geer. *Introduction to Coding Theory and Algebraic Geometry.* 1988.

[WB83] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, 1983. US patent number 4,633,470.

[ZZ18] Fangguo Zhang and Zhuoran Zhang. Code-based cryptosystem from quasi-cyclic elliptic codes. Cryptology ePrint Archive, Report 2018/1182, 2018. https://eprint.iacr.org/2018/1182.

ECOLE DOCTORALE

Titre: Sur des algorithmes de décodage de codes géométriques au delà de la moitié de la distance minimale

Mots clés: Codes géométriques, décodage, error correcting pair, algorithme de Sudan, power decoding

Résumé: Cette thèse porte sur les codes géométriques et leur décodage. Ces codes sont constitués de vecteurs d'evaluations de fonctions spécifiques en des points d'une courbe algébrique. La structure algébrique sous-jacente de ces codes a permis de concevoir plusieurs algorithmes de décodage. Un premier algorithme pour les codes provenant de courbes planes est proposé en 1989 par Justesen, Larsen, Jensen, Havemose et Hoholdt. Il est ensuite étendu à toute courbe par Skorobatov et Vladut et appelé "basic algorithm" dans la literature. Quelques années plus tard, Pellikaan et indépendamment Koetter en donnent une formulation sans géométrie algébrique utilisant simplement le langage des codes. Cette nouvelle interprétation prend le nom d'algorithme "Error Correcting Pairs" (ECP) et représente une percée en théorie des codes, car l'algorithme s'applique à tout code muni d'une certaine structure qui se décrit uniquement en termes de produits coordonnées par coordonnées de codes. Le rayon de décodage de cet algorithme dépend du code auquel il est appliqué. Pour les codes de Reed-Solomon, il atteint la moitié de la distance minimale,seuil d'unicité de la solution. Pour les codes géométriques, l'algorithme arrive à décoder presque toujours une quantité d'erreurs égale à la moitié de la distance construite. Toutefois, le bon fonctionnement de l'algorithme n'est garanti que pour une quantité d'erreurs inférieure à la moitié de la distance construite moins un multiple du genre de la courbe. Plusieurs tentatives ont ensuite été menées pour effacer cette penalité dûe au genre. Un premier résultat déterminant a été celui de Pellikaan, qui a prouvé l'existence d'un algorithme avec rayon de décodage égal à la moitié de la distance construite. Puis,en 1993 Ehrhard est parvenu à une procédure effective pour construire un tel algorithme. En plus des algorithmes pour le décodage unique,les codes géométriques disposent d'algorithmes corrigeant une quantité d'erreurs supérieure à la moitié de la distance construite. Au delà de cette quantité, l'unicité de la solution pourrait ne pas être assurée. On utilise alors des algorithmes dits de "decodage en liste" qui renvoient la liste des solutions possibles. C'est le cas de l'algorithme de Sudan. Une autre approche consiste à concevoir des algorithmes qui renvoient une unique solution mais peuvent échouer. C'est le cas du "power decoding". Les algorithmes de Sudan et du power decoding ont d'abord été conçus pour les codes de Reed-Solomon,puis étendus aux codes géométriques. On observe que ces extensions n'ont pas les mêmes rayons de décodage: celui de l'algorithme de Sudan est inférieur à celui du Power decoding, la différence étant proportionnelle au genre de la courbe. Dans cette thèse nous présentons deux résultats principaux. Premièrement, nous proposons un nouvel algorithme que nous appelons "power error locating pairs" qui, comme l'algorithme ECP, peut être appliqué à tout code muni d'une certaine structure se décrivant en termes de produits coordonnées par coordonnées. Comparé à l'algorithme ECP, cet algorithme peut corriger des erreurs au delà de la moitié de la distance construite du code. Appliqué aux codes de Reed–Solomon ou, plus généralement, aux codes géométriques, il est equivalent à l'algorithme du power decoding. Mais il peut aussi être appliqué à des codes cycliques spécifiques pour lesquels il permet de décoder au delà de la moitié de la borne de Roos. Par ailleurs, cet algorithme appliqué aux codes géométriques fait abstraction de la structure géométrique sous-jascente ce qui ouvre d'intéressantes applications en cryptanalyse. Le second résultat a pour but d'effacer la penalité proportionnelle au genre dans le rayon de décodage de l'algorithme de Sudan pour les codes géométriques. D'abord, en suivant la méthode de Pellikaan, nous prouvons que un tel algorithme existe. Puis, en généralisant les travaux de Ehrhard et Sudan, nous donnons une procédure effective pour construire cet algorithme.

Title: On decoding algorithms for algebraic geometry codes beyond half the minimum distance

Keywords: Algebraic geometry codes, decoding, error correcting pair, Sudan's algorithm, power decoding

Abstract: This thesis deals with algebraic geometric (AG) codes and their decoding. Those codes are composed of vectors constructed by evaluating specific functions at points of an algebraic curve. The underlying algebraic structure of these codes made it possible to design several decoding algorithms. A first one, for codes from plane curves is proposed in 1989 by Justesen, Larsen, Jensen, Havemose and Hoholdt. It is then extended to any curve by Skorobatov and Vladut and called "basic algorithm" in the literature. A few years later, Pellikaan and independently Koetter, give a formulation without algebraic geometry using simply the language of codes. This new interpretation, takes the name "Error Correcting Pairs" (ECP) algorithm and represents a breakthrough in coding theory since it applies to every code having a certain structure which is described only in terms of component-wise products of codes. The decoding radius of this algorithm depends on the code to which it is applied. For Reed-Solomon codes, it reaches half the minimum distance, which is the threshold for the solution to be unique. For AG, the algorithm almost always manages to decode a quantity of errors equal to half the designed distance. However, the success of the algorithm is only guaranteed for a quantity of errors less than half the designed distance minus some multiple curve's genus. Several attempts were then made to erase this genus-proportional penalty. A first decisive result was that of Pellikaan, who proved the existence of an algorithm with a decoding radius equal to half the designed distance. Then in 1993 Ehrhard obtained an effective procedure for constructing such an algorithm. In addition to the algorithms for unique decoding, AG codes have algorithms correcting amount of errors greater than half the designed distance. Beyond this quantity, the uniqueness of the solution may not be guaranteed. We then use a so-called "list decoding" algorithm which returns the list of any possible solutions. This is the case of Sudan's algorithm for Reed-Solomon codes. Another approach consists in designing algorithms, which returns a single solution but may fail. This is the case of the "power decoding". Sudan's and power decoding algorithms have first been designed for Reed-Solomon codes, then extended to AG codes. We observe that these extensions do not have the same decoding radii: that of Sudan algorithm is lower than that of the power decoding, the difference being proportional to the genus of the curve. In this thesis we present two main results. First, we propose a new algorithm that we call "power error locating pairs" which, like the ECP algorithm, can be applied to any code with a certain structure described in terms of component-wise products. Compared to the ECP algorithm, this algorithm can correct errors beyond half the designed distance of the code. Applied to Reed-Solomon or to AG codes, it is equivalent to the power decoding algorithm. But it can also be applied to specific cyclic codes for which it can be used to decode beyond half the Roos bound. Moreover, this algorithm applied to AG codes disregards the underlying geometric structure which opens up interesting applications in cryptanalysis. The second result aims to erase the penalty proportional to the genus in the decoding radius of Sudan's algorithm for AG codes. First, by following Pellikaan's method, we prove that such an algorithm exists. Then, by combining and generalizing the works of Ehrhard and Sudan, we give an effective procedure to build this algorithm.

Institut Polytechnique de Paris
91120 Palaiseau, France