ANNÉE 2013

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

En Cotutelle Internationale avec
l'Université d'Ulm, Allemagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Mathématiques et applications
Ecole doctorale Matisse

présentée par

Antonia Wachter-Zeh

préparée à l'unité de recherche 6625 du CNRS : IRMAR
Institut de Recherche Mathématique de Rennes
UFR de Mathématiques

Decoding of Block and Convolutional Codes in Rank Metric

Thèse soutenue à Ulm, Allemagne, le 04 octobre 2013 devant le jury composé de :

Martin Bossert
Université d'Ulm, directeur de thèse

Pierre Loidreau
DGA & Université de Rennes 1, directeur de thèse

Christine Bachoc
Université Bordeaux 1, rapportrice

Tom Høholdt
Technical University of Denmark, rapporteur

Maurits Ortmanns
Université d'Ulm, examinateur

Felix Ulmer
Université de Rennes 1, examinateur

Acknowledgments

This dissertation contains most of my work as a PhD student at the Institute of Communications Engineering at Ulm University, Germany, and at the Institut de Recherche Mathématiques de Rennes, France. Several people contributed to my research as well as to having a great time; therefore, I would like to express my deepest gratitude to all of them.

First of all, I am very thankful to my supervisors Martin Bossert and Pierre Loidreau.

I would like to thank Martin for enthusiastically encouraging me to work in coding theory, for supporting me in uncountable aspects during my PhD time and for providing such a great, inspiring working environment. His research network provided several possibilities to work with scientists from all over the world, he supported my research stays in Rennes and made many conference participations possible.

Pierre influenced my research in many ways by inspiring scientific discussions. The *cotutelle de thèse* with him made me experience scientific life in two different academic systems and widely enriched my view on research, teaching and international collaborations. I also thank him a lot for making my research stays in Rennes very memorable and the bureaucratic effort within the French system.

I am also very thankful to Vladimir Sidorenko with whom I enjoyed many valuable discussions and fruitful collaborations. Especially in the beginning, Volodya's "nasty questions" pointed me to new research directions and later, his constructive feedback always helped me to improve the presentation of results. It was a pleasure to learn Russian songs from him.

It was a great honor for me to defend the results of my thesis in front of an international high-quality jury. I would like to thank Tom Høholdt (Technical University of Denmark) and Christine Bachoc (Université Bordeaux 1, France) for the external reviews of my PhD thesis and for the interest in my work. Thank you, Pierre, Tom and Felix Ulmer (Université de Rennes 1, France) for coming to Ulm for my defense.

The last four years were an unforgettable time; this is to a big portion due to all the great colleagues and friends in Ulm and Rennes.

In Ulm, I am especially grateful to all (former) members of the coding group for the scientific discussions: Sabine Kampf, Wenhui Li, Mostafa H. Mohamed, Johan S. R. Nielsen, Christian Senger, Henning Zörlein, and the "younger" ones: Sven Puchinger and Sven Müelich. Thanks also to Volodya, Wenhui and Mostafa for proofreading parts of this thesis. Moreover, I would like to thank Carolin Huppert, Johannes Klotz, Frederic Knabe, David Kracht, Katharina Mir, Steffen Schober and all the others for the enjoyable coffee breaks and the discussions during lunch time, ranging from mathematics and politics to sports and senseless topics. I also want to thank Eva Peiker, with whom I worked several years ago as a student assistant and my roommate Dave for the pleasant and humorous atmosphere in our office.

I am also thankful to all the undergraduate students from Ulm who made their Bachelor's and Master's theses with me: Georg Fischer, Matthias Führle, Harris Hafeez, Mostafa H. Mohamed, Sven Puchinger, Markus Stinner and Markus Ulmschneider.

In Rennes, some people made my stays really enjoyable. Thanks to Matthieu Legeay for letting me join his office, his great humor and listening to my improvable French. Thank you also for guiding Alex and me around in the beauty of the Bretagne and enjoying several crêpes together. Thanks to all the other PhD students in Rennes for taking me to lunch many times. I would further like to thank Delphine Boucher, who always helped me whenever I had bureaucratic problems and encouraged me to give seminars in Rennes. It was a pleasure to go running with you in Dinard. I am also happy that I met Gwezheneg Robert, who explained me his interesting work and I would like to thank him and Nicolas Delfosse for proofreading my French abstract.

I would also like to thank Alexey Frolov, Pavel Rybin and Konstantin Kondrashov from the Institute for information transmission problems (IPPI), Moscow, Russia, for inviting me to a very memorable research stay in Moscow. It was a great and special experience to give a seminar at the IPPI.

Throughout my whole life, my family always supported me in my decisions. Especially my mum Maria Bibiana, my Dad Robert, together with Susi, and my siblings Joris, Ella and Finn: thank you so much for your long-lasting support.

Last, but definitely not least: Thank you, my beloved Alex, for everything you've given to me and still give to me every day. It makes me incredibly happy that we are going through the adventure of science together!

Antonia Wachter-Zeh
Ulm, October 2013

Abstract

Rank-metric codes recently attract a lot of attention due to their possible application to network coding, cryptography, space-time coding and distributed storage. An optimal-cardinality algebraic code construction in rank metric was introduced some decades ago by Delsarte, Gabidulin and Roth. This Reed–Solomon-like code class is based on the evaluation of linearized polynomials and is nowadays called Gabidulin codes.

This dissertation considers block and convolutional codes in rank metric with the objective of designing and investigating efficient decoding algorithms for both code classes.

After giving a brief introduction to codes in rank metric and their properties, we first derive sub-quadratic-time algorithms for operations with linearized polynomials and state a new bounded minimum distance decoding algorithm for Gabidulin codes. This algorithm directly outputs the linearized evaluation polynomial of the estimated codeword by means of the (fast) linearized Euclidean algorithm.

Second, we present a new interpolation-based algorithm for unique and (not necessarily polynomial-time) list decoding of interleaved Gabidulin codes. The unique decoding algorithm recovers most error patterns of rank greater than half the minimum rank distance by efficiently solving two linear systems of equations. The list decoding algorithm guarantees to return all codewords up to a certain radius.

As a third topic, we investigate the possibilities of polynomial-time list decoding of rank-metric codes in general and Gabidulin codes in particular. For this purpose, we derive three bounds on the list size. These bounds show that the behavior of the list size for both, Gabidulin and rank-metric block codes in general, is significantly different from the behavior of Reed–Solomon codes and block codes in Hamming metric, respectively. The bounds imply, amongst others, that there exists no polynomial upper bound on the list size in rank metric as the Johnson bound in Hamming metric, which depends only on the length and the minimum rank distance of the code.

Finally, we introduce a special class of convolutional codes in rank metric and propose an efficient decoding algorithm for these codes. These convolutional codes are (partial) unit memory codes, built upon rank-metric block codes. This structure is crucial in the decoding process since we exploit the efficient decoders of the underlying block codes in order to decode the convolutional code.

Résumé Français — *Décodage des codes en bloc et des codes convolutifs en métrique rang*

Les codes en métrique rang attirent l'attention depuis quelques années en raison de leur application possible au codage réseau linéaire aléatoire (*random linear network coding* [SKK08, Sil09, Gad09]), à la cryptographie à clé publique [GPT91a, Gib96, BL02, OG03, BL04, FL05, Ove06, Loi10], au codage espace-temps [GBL00, LGB03, LK04, ALR13] et aux systèmes de stockage distribué [SRV12, RSKV13]. Une construction de codes algébriques en métrique rang de cardinalité optimale a été introduite par Delsarte, Gabidulin et Roth il y a quelques décennies. Ces codes sont considérés comme l'équivalent des codes de Reed–Solomon et ils sont basés sur l'évaluation de polynômes linéarisés. Ils sont maintenant appelés les *codes de Gabidulin*.

Depuis peu, le codage réseau linéaire aléatoire est devenu un thème de recherche important. C'est un moyen efficace pour diffuser l'information dans les réseaux des quelques sources vers quelques destinations (cf. [ACLY00, HKM$^+$03, HMK$^+$06]). Le *operator channel* a été introduit par Kötter et Kschischang [KK08] comme une abstraction du codage réseau linéaire aléatoire non-cohérent. Dans ce modèle, les données par paquets sont des vecteurs d'un corps fini et la structure interne du réseau est inconnue. Chaque noeud du réseau transmet des combinaisons linéaires aléatoires de tous les paquets reçus jusqu'alors. En raison de ces combinaisons linéaires, un seul paquet erroné peut se propager largement dans le réseau et peut rendre toute la transmission inutile. Cette propagation forte des erreurs rend essentielle les codes correcteurs d'erreurs dans le codage réseau linéaire aléatoire pour reconstruire les paquets transmis.

Lorsqu'on considère les paquets transmis comme étant les lignes d'une matrice, les combinaisons linéaires des nœuds ne sont rien d'autre que des opérations élémentaires sur les lignes de cette matrice. Dans une transmission sur le *operator channel* sans erreurs et sans effacements, l'espace des lignes de la matrice transmise est donc préservé. Basés sur cette observation, Kötter et Kschischang [KK08] ont introduit des codes de sous-espaces pour la correction d'erreurs et d'effacements dans le codage réseau linéaire aléatoire. Un code de sous-espaces est un ensemble non-vide de sous-espaces d'un espace vectoriel de dimension n sur un corps fini. Chaque mot de code est un sous-espace. Comme une mesure de distance pour les codes de sous-espaces, on utilise la distance de sous-espaces (*subspace distance*), cf. [WXS03, KK08, XF09, ES09, Ska10, EV11, Sil11, BVP13].

Silva, Kschischang et Kötter [SKK08] ont montré que les codes de Gabidulin relevés (*lifted*) résultaient en des codes de sous-espaces presque optimaux pour le codage réseau linéaire aléatoire. Les codes de Gabidulin sont les analogues en métrique rang des codes de Reed–Solomon et ils ont été introduits par Delsarte, Gabidulin et Roth [Del78, Gab85, Rot91]. Un code en métrique rang de longueur $n \leq m$ peut être considéré comme un ensemble de matrices $m \times n$ dans un corps fini $\mathbb{F}_q$ ou, de manière équivalente, comme un ensemble de vecteurs de longueur n dans l'extension de corps $\mathbb{F}_{q^m}$. Le poids rang d'un tel «mot» est simplement le rang de sa représentation matricielle et la distance rang entre deux mots est le rang de leur différence. Ces définitions s'appuient sur le fait que la distance rang est une métrique.

Plusieurs constructions de codes et des propriétés de base de la métrique rang possèdent de fortes similarités avec les codes en métrique de Hamming.

Superficiellement, un code de Gabidulin relevé est un code de sous-espaces spécial où chaque mot de code est l'espace des lignes d'une matrice $\begin{bmatrix}\mathbf{I} & \mathbf{C}^T\end{bmatrix}$, où $\mathbf{I}$ désigne la matrice identité et $\mathbf{C}$ est un mot de code (en représentation matricielle) d'un code de Gabidulin fixé.

Cette thèse traite des codes en bloc et des codes convolutifs en métrique rang avec l'objectif de développer et d'étudier des algorithmes de décodage efficaces pour ces deux classes de codes. Cette thèse est structurée comme suit.

Le **chapitre 1** donne une brève motivation pour l'utilisation des codes en métriques rang dans le cadre de l'application au codage réseau linéaire aléatoire et présente un aperçu de cette thèse.

Le **chapitre 2** fournit une introduction rapide aux codes en métrique rang et leurs propriétés. Après avoir introduit des notations pour les corps finis et les bases normales, nous indiquons les définitions des codes en bloc et codes convolutifs en général. On donne quelques propriétés élémentaires et le principe de base du décodage des codes en bloc. Les codes de Gabidulin peuvent être définis comme des codes d'évaluation de polynômes linéarisés, pour cette raison, nous définissons cette classe des polynômes et montrons comment on peut effectuer les opérations mathématiques de base sur ces polynômes.

La dernière section du chapitre 2 couvre les codes en métriques rang. On définit d'abord la métrique rang et on donne des propriétés de base pour les codes en métrique rang (par exemple, les équivalents des bornes de Singleton et de Gilbert–Varshamov). Ensuite, on définit les codes de Gabidulin, on montre qu'ils atteignent la borne supérieure de Singleton pour la cardinalité et on donne leur matrices génératrice et de contrôle. Nous généralisons par la suite leur définition aux codes de Gabidulin entrelacés et on montre explicitement comment les codes de Gabidulin relevés constituent un code de sous-espaces.

Dans le **chapitre 3**, on considère des approches efficaces pour décoder les codes de Gabidulin. La première partie de ce chapitre traite des algorithmes rapides pour les opérations sur les polynômes linéarisés. Dans ce contexte, on analyse la complexité des approches connues pour les opérations dans un corps finis avec des bases normales ainsi que pour les opérations mathématiques avec des polynômes linéarisés. Ensuite, nous présentons de nouveaux algorithmes en temps sous-quadratique pour accomplir efficacement la composition linéarisé et l'algorithme d'Euclide linéarisé.

La deuxième partie de ce chapitre résume tout d'abord les techniques connues pour le décodage jusqu'à la moitié de la distance rang minimale (*bounded minimum distance decoding*) des codes de Gabidulin, qui sont basées sur les syndromes et sur la résolution d'une équation clé. Ensuite, nous présentons et nous prouvons un nouvel algorithme efficace pour le décodage jusqu'à la moitié de la distance minimale des codes de Gabidulin. Cet algorithme peut être considéré comme un équivalent de l'algorithme de Gao pour le décodage des codes de Reed–Solomon. Nous montrons comment l'algorithme d'Euclide linéarisé peut être utilisé dans ce contexte pour obtenir directement le polynôme de degré restreint d'évaluation du mot de code estimé. De plus, nous étendons cet algorithme de décodage afin de corriger non seulement des erreurs, mais aussi deux types d'effacements en métrique rang: effacements de lignes et de colonnes.

Le codage réseau linéaire aléatoire peut directement profiter d'un tel algorithme de décodage efficace pour les codes Gabidulin, car il accélère immédiatement la reconstruction des paquets transmis et donc il réduit le délai nécessaire. L'extension de notre algorithme de décodage aux combinaisons d'erreurs et d'effacements est cruciale pour gérer les pertes de paquets dans le codage réseau linéaire aléatoire.

Le **chapitre 4** est consacré aux codes de Gabidulin entrelacés et à leur décodage au-delà de la moitié de la distance rang minimale. Un mot de code d'un code de Gabidulin entrelacé peut être considéré comme s mots de code parallèles de s codes de Gabidulin normaux (pas nécessairement différents). Ces s mots de code sont corrompus par s matrices d'erreur. Lorsque ces s matrices d'erreur additives ont un espace de lignes ou de colonnes en commun, il est possible de décoder les codes de Gabidulin entrelacés au-delà de la moitié de la distance rang minimale avec une grande probabilité. Jusqu'à présent, deux approches probabilistes pour le décodage unique sont connues pour ces codes.

Dans ce chapitre, nous décrivons d'abord les deux approches connues pour le décodage unique et nous tirons une relation entre eux et leurs probabilités de défaillance. Ensuite, nous présentons un nouvel algorithme de décodage des codes de Gabidulin entrelacés basé sur l'interpolation des polynômes linéarisés. Nous prouvons la justesse de ses deux étapes principales — l'interpolation et la recherche des racines — et montrons que chacune d'elles peut être effectuée en résolvant un système d'équations linéaires.

On peut utiliser l'algorithme comme algorithme de décodage en liste des codes de Gabidulin entrelacés, qui garantit de trouver tous les mots de code dans un certain rayon. Cependant, la taille de la liste, et donc aussi au pire la complexité d'algorithme du décodage en liste, peut devenir exponentielle en la longueur du code. On peut également utiliser notre décodeur comme un décodeur probabiliste unique, en temps quadratique, avec le même rayon de décodage et la même borne supérieure de la probabilité de défaillance que les décodeurs connus. En clair, pour n'importe quel décodeur unique, au-delà de la moitié de la distance rang minimale il y aura toujours une probabilité de défaillance car il n'existe pas toujours une solution unique. Nous généralisons notre décodeur pour décoder en même temps des erreurs et des effacements de lignes et de colonnes.

Dans le codage réseau linéaire aléatoire, un code de Gabidulin relevé entrelacé contient les espaces des lignes de $\begin{bmatrix}\mathbf{I} & \mathbf{C}^{(0)T} & \mathbf{C}^{(1)T} & \dots & \mathbf{C}^{(s)T}\end{bmatrix}$, où les $\mathbf{C}^{(i)}$, pour tout $i \in [1, s]$, sont des mots de code des codes de Gabidulin sous-jacents. Ainsi, par rapport aux codes de Gabidulin relevés, le relèvement (*lifting*) des codes entrelacés de Gabidulin réduit relativement les frais généraux, qui sont causés par la matrice d'identité jointe.

Jusqu'à présent, aucun algorithme de décodage en liste en temps polynomial pour les codes de Gabidulin n'est connu et en fait il n'est même pas clair que cela soit possible. Cela nous a motivé à étudier, dans le **chapitre 5**, les possibilités du décodage en liste en temps polynomial des codes en métrique rang. Cette analyse est effectuée par le calcul de bornes sur la taille de la liste des codes en métriques rang en général et des codes de Gabidulin en particulier.

On rappelle d'abord les bornes connues sur le décodage en liste des codes en métrique de Hamming, puis on déduit des bornes sur la taille de la liste des codes en métrique rang. Nous considérons en fait le nombre maximal de mots de code dans une boule en métrique rang de rayon τ, qui est appelé la taille (maximale) de la liste. Étonnamment, ces trois nouvelles bornes révèlent toutes un comportement des codes en métrique rang qui est complètement différent de celui des codes en métrique de Hamming.

La première borne montre que la taille de la liste pour un code de Gabidulin de longueur n et de distance rang minimale d peut devenir exponentielle quand τ est au moins le rayon de Johnson $n - \sqrt{n(n-d)}$. Cela implique qu'il ne peut pas exister un algorithme de décodage en liste en temps polynomial pour les codes de Gabidulin au-delà du rayon de Johnson. Il est intéressant de noter qu'on ne sait pas ce qui se passe pour les codes de Reed–Solomon si τ est légèrement supérieur au rayon de Johnson.

Notre deuxième borne est une borne supérieure sur la taille de la liste de tous les code en métrique rang, qui est prouvé par des liens entre les codes de rang constant (*constant-rank codes*) et les codes de dimension constante (*constant-dimension codes*).

Ce sont précisément ces liens qui nous permettent de dériver la troisième borne. Avec cette borne, nous pouvons prouver qu'il existe un code en métrique rang dans $\mathbb{F}_{q^m}$ de longueur $n \leq m$ tel que la taille de la liste peut devenir exponentielle pour tout τ supérieur à la moitié de la distance rang minimale. Cela implique d'une part qu'il n'y a pas de borne supérieure polynômiale, semblable à la borne de Johnson en métrique de Hamming, et d'autre part que notre borne supérieure est presque optimale.

La pertinence d'une algorithme de décodage en liste pour le codage réseau linéaire aléatoire est évidente, car un tel décodeur pourrait tolérer plus de paquets erronés qu'un décodeur de distance minimale bornée pour les codes de Gabidulin relevés.

Enfin, dans le **chapitre 6**, on introduit des codes convolutifs en métrique rang. Ce qui nous motive à considérer ces codes est le codage réseau linéaire aléatoire *multi-shot*, où le réseau inconnu varie avec le temps et est utilisé plusieurs fois. Les codes convolutifs créent des dépendances entre les utilisations différentes du réseau afin de se adapter aux canaux difficiles.

Nous proposons des mesures de la distance pour les codes convolutifs en métrique rang par analogie avec la métrique de Hamming, à savoir la distance rang libre (*free rank distance*), la distance rang active des lignes (*active row rank distance*) et la pente (*slope*) de la distance rang active des lignes, et on prouve des bornes supérieures pour ces mesures. Basé sur des codes en bloc en métrique rang (en particulier les codes de Gabidulin), nous donnons deux constructions explicites des codes convolutifs en métrique rang : une construction à haut taux basée sur la matrice de contrôle et une construction à faible taux basée sur la matrice génératrice. Les deux définissent des codes *(partial) unit memory* et atteignent la borne supérieure de la distance rang libre.

Les codes en bloc sous-jacents nous permettent de développer un algorithme de décodage des erreurs et des effacements efficace pour la deuxième construction, qui garantit de corriger toutes les séquences d'erreurs de poids rang jusqu'à la moitié de la distance rang active des lignes. La complexité de l'algorithme de décodage est cubique en la longueur de la séquence transmise. Nous prouvons sa justesse et décrivons explicitement comment nos codes convolutifs en métrique rang peuvent être appliqués au codage réseau linéaire aléatoire *multi-shot.*

Un résumé et un aperçu des problèmes futurs de recherche sont donnés à la fin de chaque chapitre. Finalement, le **chapitre 7** conclut cette thèse.

Contents

1 Motivation and Overview **1**

2 Introduction to Codes in Rank Metric **5**

2.1 Codes over Finite Fields . . . 5

2.1.1 Notations for Finite Fields . . . 5

2.1.2 Normal Bases . . . 7

2.1.3 Basics of Block Codes and Decoding Principles . . . 9

2.1.4 Basics of Convolutional Codes . . . 13

2.2 Linearized Polynomials . . . 16

2.2.1 Definition and Properties . . . 17

2.2.2 Basic Operations . . . 18

2.2.3 Connection to Linear Maps . . . 21

2.2.4 The (Inverse) q-Transform . . . 22

2.3 Codes in Rank Metric . . . 23

2.3.1 Rank Metric and its Properties . . . 24

2.3.2 Gabidulin Codes . . . 26

2.3.3 Interleaved Gabidulin Codes . . . 29

2.3.4 Lifted Gabidulin Codes . . . 31

3 Decoding Approaches for Gabidulin Codes **33**

3.1 Fast Algorithms for Linearized Polynomials . . . 33

3.1.1 Complexity of Known Approaches and Overview of New Approaches . . . 34

3.1.2 Fast Linearized Composition Using Fragmented Polynomials . . . 37

3.1.3 Fast Linearized Composition Using Fast Multi-Point Evaluation . . . 40

3.1.4 Fast Linearized (Extended) Euclidean Algorithm . . . 43

3.2 Decoding of Gabidulin Codes . . . 46

3.2.1 Known Syndrome-Based Decoding Approaches . . . 46

3.2.2 A Gao-like Decoding Algorithm . . . 52

3.2.3 Error-Erasure Decoding . . . 55

3.2.4 Fast Error-Erasure Decoding of q-cyclic Gabidulin Codes . . . 59

3.3 Summary and Outlook . . . 61

4 Decoding Approaches for Interleaved Gabidulin Codes **63**

4.1 Known Decoding Approaches . . . 63

4.2 Principle of Interpolation-Based Decoding . . . 67

4.2.1 Interpolation Step . . . 68

4.2.2 Root-Finding Step . . . 70

4.3 Interpolation-Based Decoding Approaches . . . 72

4.3.1 A List Decoding Approach . . . 72

4.3.2 A Probabilistic Unique Decoding Approach . . . 74

4.4 Error-Erasure Decoding . . . 77

4.5 Summary and Outlook . . . 79

5 Bounds on List Decoding of Block Codes in Rank Metric 81
5.1 Known Bounds on the List Size for Codes in Hamming Metric 82
5.2 Codes Connected to the List of Decoding and Problem Statement 84
5.2.1 Connection between Constant-Dimension and Constant-Rank Codes 84
5.2.2 Problem Statement 85
5.2.3 Connection between Constant-Rank Codes and the List of Decoding 86
5.3 A Lower Bound on the List Size for Gabidulin Codes 87
5.4 An Upper Bound on the List Size for Rank-Metric Codes 89
5.5 A Lower Bound on the List Size for Rank-Metric Codes 90
5.6 Summary, Comparison to Hamming Metric and Outlook 94

6 Convolutional Codes in Rank Metric 97
6.1 Distance Measures for Convolutional Codes in Rank Metric 97
6.1.1 Definition of Distance Parameters 98
6.1.2 Upper Bounds on Distances of (Partial) Unit Memory Codes 100
6.2 Constructions of Convolutional Codes in Rank Metric 101
6.2.1 PUM Codes Based on the Parity-Check Matrix of Gabidulin Codes 101
6.2.2 PUM Codes Based on the Generator Matrix of Gabidulin Codes 105
6.3 Error-Erasure Decoding of PUM Gabidulin Codes 108
6.3.1 Bounded Row Distance Condition and Decoding Idea 108
6.3.2 Proof of Correctness of the Error-Erasure Decoding Algorithm 111
6.4 Application to Random Linear Network Coding 114
6.4.1 Multi-Shot Transmission of Lifted PUM Codes 114
6.4.2 Decoding of Lifted PUM Codes in the Operator Channel 115
6.5 Summary and Outlook 117

7 Concluding Remarks 119

A Appendix 121
A.1 Proofs for the Linearized Extended Euclidean Algorithm 121
A.2 Proof of the Generalized Transformed Key Equation 122
A.3 Comparison of Decoding Approaches for Gabidulin Codes 124

Notations

Finite Fields

q	Power of a prime
$[i] = q^i$	q-power for some integer i
$\mathbb{F}_q$	Finite field of order q
$\mathbb{F}_{q^m}$	Extension field of $\mathbb{F}_q$ of degree m
$\mathbb{F}_q^{s \times n}$	Set of all $s \times n$ matrices over $\mathbb{F}_q$
$\mathbb{F}_{q^m}^n = \mathbb{F}_{q^m}^{1 \times n}$	Set of all row vectors of length n over $\mathbb{F}_{q^m}$
$\mathcal{B} = \{\beta_0, \beta_1, \ldots, \beta_{m-1}\}$	Basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$
$\mathcal{B}^\perp = \{\beta_0^\perp, \beta_1^\perp, \ldots, \beta_{m-1}^\perp\}$	Dual basis (to $\mathcal{B}$) of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$
$\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \ldots, \beta^{[m-1]}\}$	Normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ with normal element β
$\mathcal{B}_N^\perp = \{\beta^{\perp[0]}, \beta^{\perp[1]}, \ldots, \beta^{\perp[m-1]}\}$	Dual normal basis (to $\mathcal{B}_N$) with dual normal element $\beta^\perp$
$\boldsymbol{\beta} = (\beta_0\ \beta_1\ \ldots\ \beta_{m-1})$	Ordered basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$
$\mathrm{Tr}(a) = \sum_{i=0}^{m-1} a^{[i]}$	Trace function of $a \in \mathbb{F}_{q^m}$
$\mathbf{T}_m \in \mathbb{F}_q^{m \times m}$	Multiplication table for a given normal basis
$\mathrm{comp}(\mathbf{T}_m)$	Number of non-zero elements of $\mathbf{T}_m$

Sets and Vector Spaces

$[i, j]$	Short form for $\{i, i+1, \ldots, j\}$
$\dim(\mathcal{V})$	Dimension of a vector space $\mathcal{V}$
$\mathcal{P}_q(n)$	Projective space = set of all subspaces of $\mathbb{F}_q^n$
$\mathcal{G}_q(n, r)$	Grassmannian = set of all subspaces of $\mathbb{F}_q^n$ of dimension r
$\begin{bmatrix} n \\ r \end{bmatrix}$	q-binomial = cardinality of $\mathcal{G}_q(n, r)$

Matrices

$\mathbf{A} = \left(A_{i,j}\right)_{j \in [0,n-1]}^{i \in [0,m-1]}$	$m \times n$ matrix
$\mathbf{a} = (a_0\ a_1\ \ldots\ a_{n-1})$	(Row) vector of length n
$(\mathbf{a}^T)^{\uparrow i}, (\mathbf{a}^T)^{\downarrow i}$	Cyclic up/down shift of a column vector $\mathbf{a}^T$
$\mathrm{rk}(\mathbf{A})$	Rank of matrix $\mathbf{A}$
$\ker(\mathbf{A})$	Right kernel (= nullspace) of matrix $\mathbf{A}$
$\mathrm{im}(\mathbf{A}) = \mathcal{C}_q(\mathbf{A})$	Image of $\mathbf{A}$ = column space of $\mathbf{A}$ (over $\mathbb{F}_q$)
$\mathcal{R}_q(\mathbf{A})$	Row space of $\mathbf{A}$ (over $\mathbb{F}_q$)
$\mathcal{B}_R^{(e)}(\mathbf{a}) = \mathcal{B}_R^{(e)}(\mathbf{A})$	Ball of radius e in rank metric around $\mathbf{a} \in \mathbb{F}_{q^m}^n$
$\mathcal{S}_R^{(e)}(\mathbf{a}) = \mathcal{S}_R^{(e)}(\mathbf{A})$	Sphere of radius e in rank metric around $\mathbf{a} \in \mathbb{F}_{q^m}^n$
$\mathbf{I}_s$	$s \times s$ identity matrix
$\mathrm{qvan}_s((a_0\ a_1\ \ldots\ a_{n-1}))$	$s \times n$ q-Vandermonde matrix for vector $(a_0\ a_1\ \ldots\ a_{n-1})$

Linearized Polynomials

$\mathbb{L}_{q^m}[x]$	Linearized polynomial ring in $\mathbb{F}_{q^m}$ with indeterminate x
$\mathbb{L}_{q^m}[x, y_1, \dots, y_s]$	Multivariate linearized polynomial ring in $\mathbb{F}_{q^m}$ with indeterminates $x, y_1, \dots, y_s$
$\deg_q a(x)$	q-degree of linearized polynomial $a(x)$
$\overline{a}(x)$	Full q-reciprocal of $a(x)$, defined by $\overline{a}_i = a_{-i}^{[i]} = a_{m-i}^{[i]}$
$\widehat{a}(x)$	q-transform of $a(x)$
$a(\mathbf{b}) = (a(b_0)\ a(b_1)\ \dots\ a(b_{n-1}))$	Evaluation of linearized polynomial $a(x)$ at a vector $\mathbf{b}$
$M_{u_0, u_1, \dots, u_{\dim(\mathcal{U})-1}}(x)$	Minimal subspace polynomial of a subspace $\mathcal{U}$, where $\{u_0, u_1, \dots, u_{\dim(\mathcal{U})-1}\}$ is a basis of $\mathcal{U}$

Block Codes

(n, M, d)	Code (not necessarily linear) of length n, cardinality M and minimum distance d (in some given metric)
$[n, k, d]$	Linear code of length n, dimension k and minimum distance d (in some given metric)
$(n, M, d)_R$	Code in rank metric (not necessarily linear) of length n, cardinality M and minimum rank distance d
$[n, k, d]_R$	Linear code in rank metric of length n, dimension k and minimum rank distance d
$\mathsf{MRD}(n, M)$	Maximum rank distance code (not necessarily linear) of length n and cardinality M
$\mathsf{MRD}[n, k]$	Linear maximum rank distance code of length n and dimension k
$\mathsf{Gab}[n, k]$	Gabidulin code of length n and dimension k
$\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$	Interleaved Gabidulin code of length n, elementary dimensions $k^{(i)}$ and interleaving order s
$\mathsf{CR}_{q^m}(n, M, d, r)$	Constant-rank code (not necessarily linear) of length n, cardinality M, minimum rank distance d and rank r over $\mathbb{F}_{q^m}$
$\mathsf{CD}_q(n, M, d_s, r)$	Constant-dimension code of cardinality M, minimum subspace distance d_s and dimension r (= subset of $\mathcal{G}_q(n, r)$)
$\mathsf{A}^R_{q^m}(n, d)$	Maximum cardinality of a code over $\mathbb{F}_{q^m}$ of length n and minimum rank distance d
$\mathsf{A}^R_{q^m}(n, d, r)$	Maximum cardinality of a constant-rank code of length n, minimum rank distance d and rank r over $\mathbb{F}_{q^m}$
$\mathsf{A}^S_q(n, d_s, r)$	Maximum cardinality of a constant-dimension code in $\mathcal{G}_q(n, r)$ with minimum subspace distance d_s

Convolutional Codes

$\mathbb{F}_q[D]$	Polynomial ring in $\mathbb{F}_q$ with indeterminate D
μ	Memory of a convolutional generator matrix
ν_i, ν	i-th and overall constraint length of a convolutional generator matrix
$\mathsf{UM}(n,k)$	Unit memory code of code rate $R = k/n$ and $\nu = k$
$\mathsf{PUM}(n,k\vert k^{(1)})$	Partial unit memory of code rate $R = k/n$ and $\nu = k^{(1)}$

Acronyms

BMD	bounded minimum distance
BRD	bounded row distance
LEEA	linearized extended Euclidean algorithm
ML	maximum likelihood
MRD	maximum rank distance
PUM	(partial) unit memory
RLNC	random linear network coding

CHAPTER 1

Motivation and Overview

Error-correcting codes have their origin in Shannon's seminal publication from 1948 [Sha48], where he proved that nearly error-free discrete data transmission is possible over any noisy channel when the code rate is less than the channel capacity. This statement is nowadays called the *(noisy) channel coding theorem.* The channel capacity depends on the physical properties of the channel and it is an active research area to determine the capacity of non-trivial communication channels. However, the proof of the channel coding theorem is non-constructive and therefore it is not clear how to construct error-correcting codes which actually achieve the Shannon limit. A steady stream of publications and textbooks about code constructions, their properties and decoding methods has emerged with Hamming's class of codes [Ham50] and has not yet come to an end. For most data transmission and data storage systems, the *Hamming metric* is the "proper" metric and codes defined in this metric practically perform quite well.

Quite recently, *random linear network coding* (RLNC) attracts a lot of attention. It is a powerful means for spreading information in networks from sources to sinks (see e.g., [ACLY00, HKM$^+$03, HMK$^+$06]). The *operator channel* was introduced by Kötter and Kschischang [KK08] as an abstraction of non-coherent RLNC. In this model, the packets are assumed to be vectors over a finite field while the internal structure of the network is unknown. Each node of the network forwards random linear combinations of all packets received so far. Due to these linear combinations, one single erroneous packet can propagate widely throughout the whole network and can render the whole transmission useless. This strong error propagation makes error-correcting codes in RLNC essential in order to reconstruct transmitted packets.

When the transmitted packets are considered as rows of a matrix, then the linear combinations of the nodes are nothing but elementary row operations on this matrix. During an error- and erasure-free transmission over the operator channel, the row space of the transmitted matrix is therefore preserved. Based on this observation, Kötter and Kschischang [KK08] used subspace codes for error control in RLNC. A subspace code is a non-empty set of subspaces of the vector space of dimension n over a finite field and each codeword is a subspace itself. The so-called *subspace distance* is used as a distance measure for subspace codes, compare e.g., [WXS03, KK08, XF09, ES09, Ska10, EV11, Sil11, BVP13].

Silva, Kschischang and Kötter [SKK08] showed that *lifted Gabidulin codes* result in almost optimal subspace codes for RLNC. Gabidulin codes are the rank-metric analogs of Reed–Solomon codes and were introduced by Delsarte, Gabidulin and Roth [Del78, Gab85, Rot91]. Codes in rank metric, in particular Gabidulin codes, can be seen as a set of matrices over a finite field. The rank of the difference of two matrices is called their *rank distance*, which is induces a metric for matrix codes, the *rank metric.*

Informally speaken, a *lifted* Gabidulin code is a special subspace code, where each codeword is the row space of a matrix $\begin{bmatrix}\mathbf{I} & \mathbf{C}^T\end{bmatrix}$, $\mathbf{I}$ denotes the identity matrix and $\mathbf{C}$ is a codeword (in matrix representation) of a fixed Gabidulin code.

For this reason, codes in rank metric are an active research area in the context of RLNC. Apart from RLNC [SKK08, Sil09, Gad09], the application of codes in rank metric ranges from cryptography [GPT91a, Gib96, BL02, OG03, BL04, FL05, Ove06, Loi10] to space-time coding [GBL00, LGB03, LK04, ALR13] and distributed storage systems [SRV12, RSKV13, SRKV13].

This dissertation deals with decoding approaches for block and convolutional codes in rank metric. In the following overview of the thesis, we motivate our results with their application to RLNC, but all of them are independently valid in a wider context of coding theory. Hence, we do not explicitly explain the application of our results to RLNC or other possible applications within the chapters (except for Chapter 6). This dissertation is structured as follows.

Chapter 2 provides a brief introduction to codes in rank metric and their properties. After giving basic notations for finite fields and normal bases, we state the definitions of block and convolutional codes in general. We give some elementary properties and basic decoding principles for block codes. Since Gabidulin codes can be defined by evaluating linearized polynomials, we define this class of polynomials and show how basic mathematical operations are performed on them. Finally, the last section of Chapter 2 covers codes in rank metric. We first define the rank metric and give basic properties and bounds on the cardinality of codes in rank metric (namely, equivalents of the Singleton and the Gilbert–Varshamov bound). Then, we define Gabidulin codes, show that they attain the Singleton-like upper bound on the cardinality and give their generator and parity-check matrices. We generalize this definition to interleaved Gabidulin codes and describe explicitly how lifted Gabidulin codes constitute a class of subspace codes.

Within **Chapter 3**, efficient approaches for decoding Gabidulin codes are considered. The first part of this chapter deals with fast algorithms for operations with linearized polynomials. In this context, we analyze the complexity of known approaches and present new algorithms to accomplish the linearized composition and the *linearized extended Euclidean algorithm* (LEEA) efficiently. The second part of this chapter describes known syndrome-based decoding techniques and presents a new efficient *bounded minimum distance* (BMD) decoding algorithm for Gabidulin codes. Our algorithm uses the (fast) LEEA in order to output directly the linearized evaluation polynomial of the estimated codeword. Further, we show how our algorithm can be used for error-erasure decoding of Gabidulin codes. RLNC can directly take advantage of such an efficient decoding algorithm for Gabidulin codes, since it accelerates immediately the reconstruction of the transmitted packets and reduces therefore the involved delay.

Chapter 4 is devoted to approaches for decoding interleaved Gabidulin codes. A codeword of an interleaved Gabidulin code can be considered as s parallel codewords of usual Gabidulin codes. When the s additive error matrices have one common row or column space, we can decode beyond half the minimum rank distance with high probability. We first describe two known approaches for unique decoding of interleaved Gabidulin codes and derive a relation between them. Then, we present a new interpolation-based decoding algorithm for interleaved Gabidulin codes. We prove the correctness of its two main steps—interpolation and root-finding—and show that both can be carried out by solving a linear system of equations. We outline how our decoder can be used as a (not necessarily polynomial-time) list decoder as well as a quadratic-time probabilistic unique decoder. In this context, we upper bound the failure probability of the unique decoder. In RLNC, a *lifted* interleaved Gabidulin code consists of the row spaces of $\left[\mathbf{I}\ \mathbf{C}^{(0)T}\,\mathbf{C}^{(1)T}\ \dots\ \mathbf{C}^{(s)T}\right]$, where the $\mathbf{C}^{(i)}$, for all $i \in [1, s]$, are codewords of the underlying Gabidulin codes. Hence, compared to lifted Gabidulin codes, the lifting of *interleaved* Gabidulin codes relatively reduces the overhead, which is caused by the appended identity matrix.

So far, no polynomial-time list decoding algorithm for Gabidulin codes is known and it is not even clear if it is possible at all. Therefore, **Chapter 5** deals with bounds on list decoding block codes in rank metric in general and Gabidulin codes in particular. We first recall known bounds on list decoding of codes in Hamming metric and then derive three bounds on list decoding codes in rank metric. Surprisingly, the rank-metric bounds are all significantly different from the known bounds in Hamming metric. In particular, we prove that in rank metric there exists no polynomial upper bound on the list size similar to the Johnson bound in Hamming metric. Further, one of our bounds shows that the list size can become exponential directly beyond the Johnson radius when decoding Gabidulin codes. Remarkably, it is not known if this property holds for Reed–Solomon codes. The relevance of a list decoding algorithm for RLNC is obvious, since such a decoder could tolerate more erroneous packets than a BMD decoder for the (lifted) Gabidulin code.

Finally, **Chapter 6** introduces convolutional codes in rank metric. The motivation of considering such codes lies in multi-shot RLNC, where the unknown and time variant network is used several times. Convolutional network codes create dependencies between the different shots in order to cope with difficult channels. First, we define distance measures for convolutional codes in rank metric and prove upper bounds on them. Then, we construct a special class of convolutional codes—partial unit memory (PUM) codes—based on rank metric block codes in two different ways. We present an algorithm which efficiently decodes these PUM codes when both, errors and erasures occur and prove its correctness. The decoding complexity of this decoding algorithm is cubic in the length of a transmitted block. Further, it is explicitly described how lifting of these codes can be applied for error correction in multi-shot RLNC.

Chapter 7 concludes this dissertation.

CHAPTER 2

Introduction to Codes in Rank Metric

Codes in rank metric of length $n \leq m$ can be considered as a set of $m \times n$ matrices over a finite field $\mathbb{F}_q$ or equivalently as a set of vectors of length n over the extension field $\mathbb{F}_{q^m}$. The rank weight of such a "word" is simply the rank of its matrix representation and the rank distance between two words is the rank of their difference. These definitions rely on the fact that the rank metric is indeed a metric. Several code constructions and basic properties of the rank metric show strong similarities to codes in Hamming metric.

Error-correcting codes in rank metric were first considered by Delsarte in 1978 [Del78], who proved a Singleton-like upper bound on the cardinality and constructed a class of codes achieving this bound. This class of codes was reintroduced in 1985 by Gabidulin in his fundamental paper [Gab85], where in addition several properties of codes in rank metric and an efficient decoding algorithm were shown. Since Gabidulin's publication contributed significantly to the development of error-correcting codes in rank metric, the most famous class of codes in rank metric—the equivalents of Reed–Solomon codes—are nowadays called *Gabidulin codes*. These codes can be defined by evaluating non-commutative *linearized polynomials*, proposed by Ore [Ore33a, Ore33b]. Independently of the previous work, Roth discovered in 1991 codes in rank metric in order to apply them for correcting crisscross error patterns [Rot91].

This chapter gives a brief introduction to the theory of error-correcting codes in rank metric. Section 2.1 provides definitions and notations used in this thesis for codes in finite fields. Section 2.2 introduces linearized polynomials and their main properties. Finally, Section 2.3 deals with general properties and explicit constructions of codes in the rank metric.

2.1 Codes over Finite Fields

Throughout this thesis, we consider algebraic codes over finite fields and hence, this section introduces notations and basic properties of finite fields (Subsection 2.1.1) and codes over them. In Subsection 2.1.2, we show properties of normal bases since they enable us to accomplish calculations in finite fields quite efficiently. We clarify our notations for block codes and explain well-known decoding principles for block codes in Subsection 2.1.3: bounded minimum distance decoding, nearest codeword decoding and list decoding. Further, we introduce notations and basic properties of convolutional codes in Subsection 2.1.4.

2.1.1 Notations for Finite Fields

This subsection provides notations concerning finite fields, without going into detail about their theory. An extensive study of finite fields, their properties and applications can be found in standard literature about finite fields, e.g., [LN96, MBG^{+}93], and also in books about coding theory, e.g., [Ber84, Bla03, Rot06, HP10].

Let p be a prime, then $\mathbb{F}_p = \{0, 1, \dots, p-1\}$ denotes the *prime field* of order p. Let q be a power of the prime p, then we denote by $\mathbb{F}_q$ the finite field of order q. This finite field $\mathbb{F}_q$ contains q elements and p is called its *characteristic*. An extension field (of extension degree m) of $\mathbb{F}_q$ is denoted by $\mathbb{F}_{q^m}$. This extension field $\mathbb{F}_{q^m}$ can be constructed from $\mathbb{F}_q$ and a polynomial $p(x)$ of degree m, which is irreducible in $\mathbb{F}_q$ and whose coefficients are in $\mathbb{F}_q$. For any m there is at least one such irreducible polynomial of degree m [LN96, Corollary 2.11]. Since all fields of the same size are isomorphic [LN96, Theorem 1.78], the field $\mathbb{F}_{q^m}$ does not depend on the explicit choice of $p(x)$ and is isomorphic to the polynomial ring over $\mathbb{F}_q$ modulo $p(x)$:

$$\mathbb{F}_{q^m} \cong \mathbb{F}_q/\langle p(x)\rangle.$$

Thus, different irreducible polynomials give different representations of the same extension field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. The construction of $\mathbb{F}_{q^m}$ can be done by using a root of $p(x)$ in $\mathbb{F}_{q^m}$.

A *primitive element* α of $\mathbb{F}_{q^m}$ is an element such that it generates the multiplicative group $\mathbb{F}_{q^m}^*$ by its powers, i.e.,

$$\mathbb{F}_{q^m}^* \overset{\text{def}}{=} \mathbb{F}_{q^m} \setminus \{0\} = \Big\{\alpha^i, \forall i \in [0, q^m-2]\Big\},$$

and $\alpha^{q^m-1} = 1$. A primitive element exists in any finite field [LN96, p. 51]. If the irreducible polynomial $p(x)$ has a primitive element as root, i.e., if $p(\alpha) = 0$, then $p(x)$ is called a *primitive polynomial*. If we use a primitive polynomial for the construction of the extension field, we can take advantage of the fact that $\mathbb{F}_{q^m}^*$ is a cyclic group [MS88, Chapter 4, Theorem 1].

The extension field $\mathbb{F}_{q^m}$ can be represented as a vector space over $\mathbb{F}_q$, using a basis $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. If the order of the basis elements is important, we denote the ordered basis by $\boldsymbol{\beta} = (\beta_0\ \beta_1\ \dots\ \beta_{m-1})$. In Section 2.1.2, we will explain a type of basis which is of special interest for efficient computations in finite fields, the so-called *normal basis*.

Remark 2.1 (Properties).
The following further properties/notations concerning finite fields are used in this thesis:

- *For any integer i, we denote the q-power by $[i] \overset{\text{def}}{=} q^i$.*
- *For any $a \in \mathbb{F}_{q^m}$: $a^{[m]} = a$ and for any $a \in \mathbb{F}_{q^m}$ and integer i, the q-power is calculated modulo m: $a^{[i]} = a^{[i \bmod m]}$.*
- *For any $A \in \mathbb{F}_q$ and any integer i: $A^{[i]} = A$.*
- *For any a, b in $\mathbb{F}_{q^m}$ and any integer i: $(a+b)^{[i]} = a^{[i]} + b^{[i]}$ [LN96, Theorem 1.46].*

The set of all subspaces of $\mathbb{F}_q^n$ is called the *projective space* and denoted by $\mathcal{P}_q(n)$. A *Grassmannian* of dimension r is the set of all subspaces of $\mathbb{F}_q^n$ of dimension $r \leq n$ and denoted by $\mathcal{G}_q(n,r)$. Clearly, the projective space is $\mathcal{P}_q(n) = \bigcup_{r=0}^{n} \mathcal{G}_q(n,r)$. The cardinality of $\mathcal{G}_q(n,r)$ is given by the *q-binomial* (also called Gaussian binomial coefficient) as follows.

Lemma 2.1 (Number of Subspaces [Ber84, Theorem 11.52]).
The number of r-dimensional subspaces of $\mathbb{F}_q^n$ over $\mathbb{F}_q$ is

$$\begin{bmatrix} n \\ r \end{bmatrix} \overset{\text{def}}{=} |\mathcal{G}_q(n,r)| = \prod_{i=0}^{r-1} \frac{q^n - q^i}{q^r - q^i}.$$

The q-binomial has the following upper and lower bounds (see e.g., [KK08, Lemma 4]):

$$q^{r(n-r)} \leq \begin{bmatrix} n \\ r \end{bmatrix} \leq 4q^{r(n-r)}. \tag{2.1}$$

In this thesis, we use $\mathbb{F}_q^{s \times n}$ to denote the set of all $s \times n$ matrices over $\mathbb{F}_q$ and $\mathbb{F}_{q^m}^n = \mathbb{F}_{q^m}^{1 \times n}$ for the set of all row vectors of length n over $\mathbb{F}_{q^m}$. For a given basis $\mathcal{B}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, there exists a one-to-one mapping for each vector $\mathbf{a} \in \mathbb{F}_{q^m}^n$ on a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$. This mapping is formally defined as follows.

Definition 2.1 (Mapping to Ground Field).
Let $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ denote a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Fix an order of this basis $\boldsymbol{\beta} = (\beta_0\ \beta_1\ \dots\ \beta_{m-1})$ and let $\mathbf{a}$ be a vector in $\mathbb{F}_{q^m}^n$. The extension of $\mathbf{a}$ over the ground field is given by the following bijective map:

$$\begin{aligned} \mathrm{ext}_{\boldsymbol{\beta}} :\quad & \mathbb{F}_{q^m}^n \to \mathbb{F}_q^{m \times n} \\ \mathbf{a} = (a_0\ a_1\ \dots\ a_{n-1}) \mapsto \mathbf{A} = & \begin{pmatrix} A_{0,0} & A_{0,1} & \dots & A_{0,n-1} \\ A_{1,0} & A_{1,1} & \dots & A_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m-1,0} & A_{m-1,1} & \dots & A_{m-1,n-1} \end{pmatrix}, \end{aligned}$$

where $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ is defined such that

$$a_j = \sum_{i=0}^{m-1} A_{i,j}\beta_i, \quad \forall j \in [0, n-1].$$

Therefore, $\mathbf{a} = \boldsymbol{\beta} \cdot \mathbf{A}$. If we apply $\mathrm{ext}_{\boldsymbol{\beta}}$ to a single element $a \in \mathbb{F}_{q^m}$, it is mapped to a column vector $\mathrm{ext}_{\boldsymbol{\beta}}(a) \in \mathbb{F}_q^{m \times 1}$. Throughout this thesis, we will therefore use the following notations to switch between the two representations:

$$\mathbf{A} = \mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{a}), \qquad \mathbf{a} = \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{A}).$$

Further, let $\mathrm{rk}(\mathbf{a})$ denote the (usual) rank of $\mathbf{A} = \mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{a})$ over $\mathbb{F}_q$ and let $\mathcal{R}_q(\mathbf{A})$ and $\mathcal{C}_q(\mathbf{A})$ denote the row and column space of $\mathbf{A}$ in $\mathbb{F}_q^n$ and $\mathbb{F}_q^m$, respectively. The right kernel of a matrix is denoted by $\ker(\mathbf{A})$ and as a notation, $\ker(\mathbf{a}) = \ker(\mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{a})) = \ker(\mathbf{A})$.

For any $m \times n$ matrix, the *rank nullity theorem* states that $\dim\ker(\mathbf{a}) + \mathrm{rk}(\mathbf{a}) = n$. We use the notation as a vector $\mathbf{a} \in \mathbb{F}_{q^m}^n$ or matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ equivalently, whatever is more convenient.

2.1.2 Normal Bases

Normal bases facilitate calculations in finite fields and can therefore be used to reduce the computational complexity. This fact is crucial for our efficient decoding algorithm for Gabidulin codes in Subsection 3.2.4. We shortly sum up the main properties of normal bases here; however, for further theory, the interested reader is referred to the literature, e.g., [Gao93, LN96, MBG+93].

A basis $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ is a *normal basis* if $\beta_i = \beta^{[i]}$ for all i and we denote it by $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[m-1]}\}$ in the following. We call $\beta \in \mathbb{F}_{q^m}$ a *normal element.* Lemma 2.2 shows that choosing a normal basis is not restricted to certain extension fields.

Lemma 2.2 (Existence of Normal Basis [LN96, Theorem 2.35]).
There is a normal basis for any finite extension field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, i.e., for any prime power q and any positive integer m.

The following lemma about the existence of normal bases is even stronger.

Lemma 2.3 (Existence of Normal Basis [BJ86]).
In a finite extension field $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, there is a normal basis of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$ for every positive divisor s of m. For the normal element of this basis $\beta^{[s]} = \beta$ holds.

The so-called *dual basis* $\mathcal{B}^\perp$ of a basis $\mathcal{B}$ is needed in order to switch between a polynomial and its q-transform (compare Definition 2.12). To define the dual basis for a given basis $\mathcal{B}$, we need the trace function of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ for an element $a \in \mathbb{F}_{q^m}$:

$$\begin{aligned} \mathrm{Tr}: \quad & \mathbb{F}_{q^m} \rightarrow \mathbb{F}_q \\ & a \mapsto \mathrm{Tr}(a) \overset{\text{def}}{=} \sum_{i=0}^{m-1} a^{[i]}. \end{aligned}$$

The trace function is an $\mathbb{F}_q$-linear map from $\mathbb{F}_{q^m}$ to $\mathbb{F}_q$ and hence, $\mathrm{Tr}(a) \in \mathbb{F}_q$ [LN96, Chapter 2.3]. A basis $\mathcal{B}^\perp = \{\beta_0^\perp, \beta_1^\perp, \dots, \beta_{m-1}^\perp\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ is called a dual basis to $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ if:

$$\mathrm{Tr}(\beta_i \beta_j^\perp) = \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{else.} \end{cases} \tag{2.2}$$

Lemma 2.4 (Dual of a (Normal) Basis [MBG$^+$93, Theorem 1.1 and Corollary 1.4]).
For any given basis $\mathcal{B}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, there exists a unique dual basis $\mathcal{B}^\perp$. The dual basis of a normal basis is also a normal basis.

If a basis is dual to itself, i.e., if $\mathcal{B} = \mathcal{B}^\perp$, we call it a *self-dual basis* and if it is additionally normal, we call it a *self-dual normal basis* $\mathcal{B}_N = \mathcal{B}_N^\perp$. A self-dual basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ exists if and only if either q is even or both q and m are odd [MBG$^+$93, Theorem 1.9]. Self-dual normal bases of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ exist if m is odd or if q is even and $m \equiv 2 \mod 4$ [MBG$^+$93, Theorem 1.14].

We explain now basic mathematical operations on two elements $a, b \in \mathbb{F}_{q^m}$ using a normal basis $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[m-1]}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Apply the mapping ext_β from Definition 2.1 in order to represent these two elements as vectors in $\mathbb{F}_q$:

$$\begin{aligned} (A_0 \; A_1 \; \dots \; A_{m-1})^T &\overset{\text{def}}{=} \mathrm{ext}_\beta(a) \in \mathbb{F}_q^{m \times 1}, \\ (B_0 \; B_1 \; \dots \; B_{m-1})^T &\overset{\text{def}}{=} \mathrm{ext}_\beta(b) \in \mathbb{F}_q^{m \times 1}. \end{aligned}$$

An important observation is that in a normal basis representation, the q-power of an element a in $\mathbb{F}_{q^m}$ corresponds to a cyclic shift of the corresponding vector $\mathrm{ext}_\beta(a)$ over $\mathbb{F}_q$:

$$\mathrm{ext}_\beta\big(a^{[j]}\big) = (A_{m-j} \; A_{m-j+1} \; \dots \; A_0 \; A_1 \; \dots \; A_{m-j-1})^T \overset{\text{def}}{=} \mathrm{ext}_\beta(a)^{\downarrow j} \in \mathbb{F}_q^{m \times 1}, \tag{2.3}$$

where the down arrow denotes a cyclic shift of the vector by j positions to the bottom. The efficiency of calculations with normal bases stems exactly from this property and from the application of a so-called *multiplication table* (compare [Gao93, MBG$^+$93]).

Definition 2.2 (Multiplication Table).
Let $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[m-1]}\}$ be a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. The multiplication table of $\mathcal{B}_N$ is a matrix $\mathbf{T}_m \in \mathbb{F}_q^{m \times m}$ such that:

$$\beta^{[0]} \cdot \left(\beta^{[0]} \; \beta^{[1]} \; \dots \; \beta^{[m-1]}\right)^T = \mathbf{T}_m \cdot \left(\beta^{[0]} \; \beta^{[1]} \; \dots \; \beta^{[m-1]}\right)^T.$$

The number of non-zero entries in $\mathbf{T}_m$ is called the complexity of $\mathbf{T}_m$ of $\mathcal{B}_N$ and is denoted by $\mathrm{comp}(\mathbf{T}_m)$.

The addition $a+b$ in $\mathbb{F}_{q^m}$ can be done component-wise by the addition $\mathrm{ext}_{\boldsymbol{\beta}}(a) + \mathrm{ext}_{\boldsymbol{\beta}}(b) \in \mathbb{F}_q^{m\times 1}$ and is therefore easy to implement. By means of the multiplication table, the product of $a \cdot b \in \mathbb{F}_{q^m}$ can be calculated over the ground field $\mathbb{F}_q$:

$$\text{For } a, b \in \mathbb{F}_{q^m} : \ \mathrm{ext}_{\boldsymbol{\beta}}(a \cdot b) = \sum_{i=0}^{m-1} B_i \left(\mathbf{T}_m^T \cdot \mathrm{ext}_{\boldsymbol{\beta}}(a)^{\uparrow i}\right)^{\downarrow i} \in \mathbb{F}_q^{m\times 1}, \tag{2.4}$$

where the up/down arrows denote cyclic shifts of the vector by i positions to the top/bottom.

If one of the elements is a basis element, i.e., $b = \beta^{[j]}$, then the vector $\mathrm{ext}_{\boldsymbol{\beta}}(b)$ is non-zero only in the j-th row and (2.4) becomes

$$\text{For } a, \beta \in \mathbb{F}_{q^m}, \beta \in \mathcal{B}_N : \ \mathrm{ext}_{\boldsymbol{\beta}}\big(a \cdot \beta^{[j]}\big) = \left(\mathbf{T}_m^T \cdot \mathrm{ext}_{\boldsymbol{\beta}}(a)^{\uparrow j}\right)^{\downarrow j} \in \mathbb{F}_q^{m\times 1}. \tag{2.5}$$

It becomes clear from (2.4) and (2.5) that the number of operations in $\mathbb{F}_q$ in order to determine $\mathrm{ext}_{\boldsymbol{\beta}}(a \cdot b)$ directly depends on the number on non-zero entries of $\mathbf{T}_m$, i.e., on its complexity $\mathrm{comp}(\mathbf{T}_m)$. Therefore, it is desirable that $\mathbf{T}_m$ is sparse.

This complexity is lower bounded by $\mathrm{comp}(\mathbf{T}_m) \geq 2m-1$ [MBG$^+$93, Theorem 5.1]. A normal basis with $\mathrm{comp}(\mathbf{T}_m) = 2m-1$ is an *optimal* normal basis. We call normal bases with complexity in the order of $\mathcal{O}(m)$ *low-complexity* normal bases. Optimal normal bases exist for several values[1], but for our applications low-complexity (but not necessarily optimal) normal bases are sufficient. Low-complexity normal bases with $\mathcal{O}(\mathrm{comp}(\mathbf{T}_m)) = \mathcal{O}(m)$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ exist in many cases, e.g. for $q = 2^s$ if $\gcd(m, s) = 1$ and $8 \nmid m$. For $q = 2^s$ and odd m, all these low-complexity normal bases are self-dual (see also [Gao93, Chapter 5]).

The complexity of the mentioned operations will be analyzed in detail in Subsection 3.1.1.

2.1.3 Basics of Block Codes and Decoding Principles

This subsection gives basic notations and properties of block codes. A deeper investigation of code classes, constructions and properties can be found in books on algebraic coding theory, e.g., [PW72, Bla83, Ber84, MS88, vL98, Bos98, JH04, Rot06]. In the following, we show the definition of a metric, give the notations of a (linear) block code and explain encoding and decoding principles.

Definition and Basic Properties of Block Codes

Assume, a set $\mathcal{A}$ (e.g., of vectors or matrices) is given. In order to define error-correcting codes, we need a measurement of distance between the elements in this set. A distance measure on this set $\mathcal{A}$ is called a *metric* if it fulfills the following conditions.

Definition 2.3 (Metric).
Let $\mathcal{A}$ be a set (e.g., of vectors or matrices). A distance measure $d_A(\mathbf{a}, \mathbf{b})$ on any two elements $\mathbf{a}, \mathbf{b}$ in this set $\mathcal{A}$ is a metric if it satisfies for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{A}$:

- *positive definiteness: $d_A(\mathbf{a}, \mathbf{b}) \geq 0$, where $d_A(\mathbf{a}, \mathbf{b}) = 0$ if and only if $\mathbf{a} = \mathbf{b}$,*
- *symmetry: $d_A(\mathbf{a}, \mathbf{b}) = d_A(\mathbf{a}, \mathbf{b})$,*
- *triangle inequality: $d_A(\mathbf{a}, \mathbf{b}) + d_A(\mathbf{b}, \mathbf{c}) \geq d_A(\mathbf{a}, \mathbf{c})$.*

[1] See [MBG$^+$93, Table 5.1] for all values of $m \leq 2000$ with an optimal normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.

Classical error-correcting codes are defined in *Hamming metric* and they have been subject of a large number of publications. Among codes in Hamming metric, the well-known classes of Hamming codes [Ham50], Reed–Muller codes [Ree54, Mul54], Reed–Solomon codes [RS60], cyclic codes (also called BCH codes) [Hoc59, BR60] and many others can be found. In this thesis, we consider codes in *rank metric*. This metric will be given in Subsection 2.3, Definition 2.13, for block codes in $\mathbb{F}_{q^m}$.

From a practical point of view, a block code of length n is a code, where each "block" of length n can be decoded independently from the other blocks. Based on a given metric, a block code can be defined as follows.

Definition 2.4 (Block Code).
Let a metric in $\mathbb{F}_q^n$ be given, fulfilling the requirements of Definition 2.3.

An (n, M, d) block code C over $\mathbb{F}_q$ is a set of vectors in $\mathbb{F}_q^n$ of cardinality M, where the minimum distance (in the given metric) between any two vectors of this code is d.

A block code C over $\mathbb{F}_q$ is called linear if it is a k-dimensional subspace of $\mathbb{F}_q^n$ over $\mathbb{F}_q$ and its parameters are denoted by $[n, k, d]$. The parameter k is called the dimension of C.

The fraction $R \overset{\text{def}}{=} (\log_q M)/n$ is called the code rate of C. If C is linear, then $R = k/n$.

We call all vectors in $\mathbb{F}_q^k$ *information words*. The vectors in $\mathbb{F}_q^n$ in an (n, M, d) code are called *codewords*. The cardinality of a *linear* $[n, k, d]$ block code C over $\mathbb{F}_q$ is therefore $M = q^k$ and since C is a subspace of $\mathbb{F}_q^n$, for any codewords $\mathbf{c}^{(1)}, \mathbf{c}^{(2)} \in \mathsf{C}$ and any elements $a, b \in \mathbb{F}_q$, the linear combination $a\mathbf{c}^{(1)} + b\mathbf{c}^{(2)}$ is also a codeword of C.

A linear code can be defined by its *generator matrix* using a basis of the k-dimensional subspace.

Definition 2.5 (Generator Matrix).
Let C be a linear $[n, k, d]$ code over $\mathbb{F}_q$, i.e., it is a k-dimensional subspace of $\mathbb{F}_q^n$ over $\mathbb{F}_q$. A $k \times n$ generator matrix $\mathbf{G}$ of C is a matrix whose rows are a basis of this k-dimensional vector space over $\mathbb{F}_q$.

The generator matrix can be used to *encode* the information words in $\mathbb{F}_q^k$ into codewords in $\mathbb{F}_q^n$. Thus, a codeword of an $[n, k, d]$ code is any vector in $\mathbb{F}_q^n$ which can be obtained by $\mathbf{u} \cdot \mathbf{G}$, for some $\mathbf{u} \in \mathbb{F}_q^k$. Encoding defines the bijective map of the information vectors in $\mathbb{F}_q^k$ to the codewords in $\mathbb{F}_q^n$:

$$\begin{aligned} \text{enc} : \quad & \mathbb{F}_q^k \to \mathbb{F}_q^n \\ \mathbf{u} = (u_0\ u_1\ \dots\ u_{k-1}) & \mapsto \mathbf{c} = (c_0\ c_1\ \dots\ c_{n-1}). \end{aligned}$$

Notice that there is more than one generator matrix for a given $[n, k, d]$ code C, since we can use any basis of the k-dimensional subspace C over $\mathbb{F}_q$ in an arbitrary order.

Definition 2.6 (Dual Code).
For two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$, let $\langle \mathbf{a}, \mathbf{b} \rangle \overset{\text{def}}{=} \sum_{i=0}^{n-1} a_i b_i$ define the inner product and let C be a linear $[n, k, d]$ code over $\mathbb{F}_q$. Then, the set of vectors

$$\mathsf{C}^{\perp} \overset{\text{def}}{=} \left\{ \mathbf{c}^{\perp} \in \mathbb{F}_q^n : \langle \mathbf{c}^{\perp}, \mathbf{c} \rangle = 0,\ \forall\, \mathbf{c} \in \mathsf{C} \right\}$$

is called the dual code to C.

The dual code of an $[n, k, d]$ code over $\mathbb{F}_q$ is also a linear code over $\mathbb{F}_q$ and has dimension $k^{\perp} = n - k$ and length n. Its minimum distance is denoted by $d^{\perp}$, but its value is not necessarily determined by the parameters of the $[n,k,d]$ code[2]. Therefore, the dual code $\mathsf{C}^{\perp}$ is an $[n,n-k,d^{\perp}]$ code, i.e., an $(n-k)$-dimensional subspace of $\mathbb{F}_q^n$, which can be used to define the *parity-check matrix* of C.

Definition 2.7 (Parity-Check Matrix).
An $(n-k)\times n$ matrix $\mathbf{H}$ over $\mathbb{F}_q$ is called a parity-check matrix of an $[n,k,d]$ code C over $\mathbb{F}_q$ if and only if it is a generator matrix of the $[n,n-k,d^{\perp}]$ dual code $\mathsf{C}^{\perp}$ over $\mathbb{F}_q$.

Thus, for any $\mathbf{c}\in\mathsf{C}$, the multiplication with the parity-check matrix gives $\mathbf{c}\cdot\mathbf{H}^T=\mathbf{0}$ and $\mathbf{G}\cdot\mathbf{H}^T=\mathbf{0}$. A parity-check matrix is therefore a matrix whose right kernel is the code C.

Definition 2.8 (Syndrome).
For any $\mathbf{a}\in\mathbb{F}_q^n$ and a parity-check matrix $\mathbf{H}$ of an $[n,k,d]$ code C, the vector $\mathbf{s}=\mathbf{a}\cdot\mathbf{H}^T\in\mathbb{F}_q^{n-k}$ is called the syndrome of $\mathbf{a}$.

If and only if $\mathbf{a}\in\mathsf{C}$, then the syndrome is $\mathbf{s}=\mathbf{0}$.

Decoding Principles of Block Codes

After introducing these basic notations, let us now proceed to basic decoding principles.

Lemma 2.5 (Unique Decoding Capability [MS88]).
Let C be an (n,M,d) block code over $\mathbb{F}_q$ with minimum distance d in a given metric $d_A(\cdot,\cdot)$ (see Definition 2.3) and let $\mathbf{r}$ be a word in $\mathbb{F}_q^n$.

Then, there is at most one codeword $\mathbf{c}\in\mathsf{C}$ such that $d_A(\mathbf{r},\mathbf{c})\leq\tau_0\overset{\text{def}}{=}\lfloor (d-1)/2\rfloor$. Further, if there is a codeword $\mathbf{c}\in\mathsf{C}$ such that $0<d_A(\mathbf{r},\mathbf{c})\leq d-1$, then $\mathbf{r}\notin\mathsf{C}$.

The process of reconstructing the codeword from a received word is called *decoding* and we use the expression “number of errors” throughout this thesis for $d_A(\mathbf{r},\mathbf{c})$ (in the corresponding metric). Lemma 2.5 shows that we can always *decode* uniquely up to $\tau_0=\lfloor (d-1)/2\rfloor$ errors and *detect* up to $d-1$ errors.

In this dissertation, we distinguish three decoding principles for an (n,M,d) code C over $\mathbb{F}_q$, which are illustrated in Figure 2.1 and explained in the following. For each of them, we assume that a received word $\mathbf{r}\in\mathbb{F}_q^n$ is given and denote by $\mathcal{B}^{(e)}(\mathbf{r})$ a ball in the given metric around $\mathbf{r}$ of radius e.

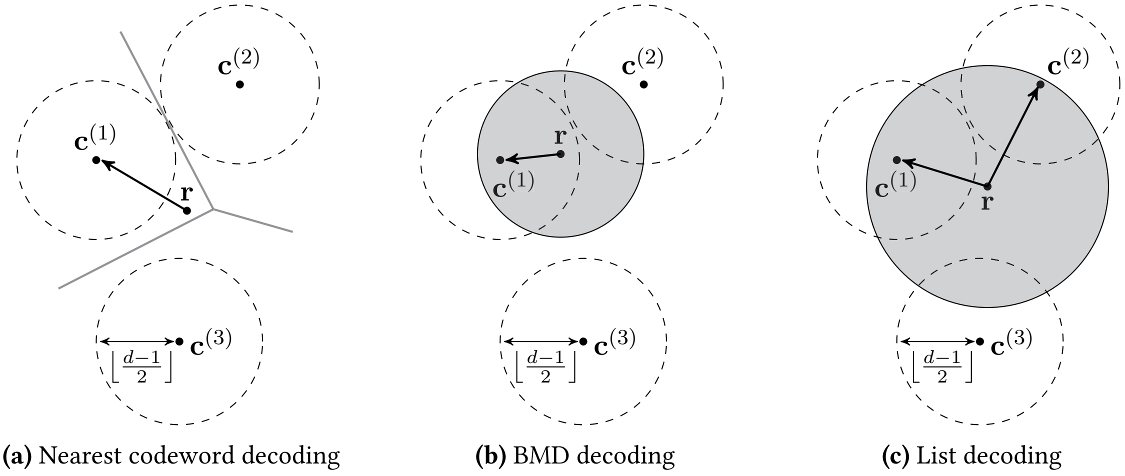
Nearest codeword decoding (see Figure 2.1a). A *nearest codeword* decoder maps the received word $\mathbf{r}$ to the closest codeword, i.e., the codeword with the smallest distance to $\mathbf{r}$. If there is more than one codeword in smallest distance to $\mathbf{r}$, we can either output all of them or choose one randomly. For a given metric $d_A(\cdot,\cdot)$, the decoding result is hence[3]:

$$\mathbf{c}'=\arg\left(\min_{\mathbf{c}\in\mathsf{C}} d_A(\mathbf{r},\mathbf{c})\right)\subseteq\mathsf{C}.$$

The output of a nearest codeword decoder is therefore always at least one codeword; a decoding failure is never declared. If we assume that a smaller error weight (in the corresponding metric) is more likely than a greater error weight, then nearest codeword decoding is equivalent to *maximum likelihood* (ML) decoding. For codes in Hamming metric, ML decoding of general linear block codes

[2] However, for some classes of codes, there is a direct connection, e.g. for maximum distance separable and maximum rank distance (MRD) codes.

[3] We have to define $\arg\min_x f(x)$ either such that it returns the *set* of all values for which $f(x)$ attains its minimum or such that it chooses one randomly.



(a) Nearest codeword decoding **(b)** BMD decoding **(c)** List decoding

Figure 2.1. Illustration of explained decoding principles, where the arrows show on which codeword(s) the received word is mapped. The dashed balls have radius $\lfloor (d-1)/2 \rfloor$ and the gray balls have radius τ_0 and τ, respectively.

is NP complete [BMVT78], and for codes in rank metric this is also conjectured. In any case, nearest codeword and ML decoders are hardly feasible due to their high computational complexity.

Bounded minimum distance decoding (see Figure 2.1b). A *bounded minimum distance* (BMD) decoder guarantees to find all codewords in radius at most $\tau_0 = \lfloor (d-1)/2 \rfloor$ from the received word. Due to Lemma 2.5, there is at most one such codeword and the decoding result is

$$\mathbf{c}' = \left(\mathsf{C} \cap \mathcal{B}^{(\tau_0)}(\mathbf{r})\right) \in \left(\mathsf{C} \cup \{\ \}\right).$$

Therefore, we obtain either a unique codeword or the empty set, in which case we can declare a decoding failure. For several algebraic code classes as Reed–Solomon or Gabidulin codes, there are efficient BMD decoding algorithms in the corresponding metric.

List decoding (see Figure 2.1c). The concept of *list decoding* can be seen as a generalization of BMD decoding and was introduced by Elias [Eli57] and Wozencraft [Woz58]. A list decoder guarantees to find *all* codewords around $\mathbf{r}$ up to a certain radius τ. Hence, the decoder outputs a list of codewords:

$$\mathcal{L} = \left\{\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(\ell)}\right\} = \left(\mathsf{C} \cap \mathcal{B}^{(\tau)}(\mathbf{r})\right) \subseteq \left(\mathsf{C} \cup \{\ \}\right).$$

If the output is the empty set, a decoding failure is declared. Such a list decoder makes sense from a practical point of view, if either the probability that the list size is greater than one is very small or if we can use the *whole* list in the further decoding process, e.g. in concatenated coding schemes or in iterative decoding. The design of efficient list decoding algorithms (with $\tau > \lfloor (d-1)/2 \rfloor$) is a widely investigated topic for some classes of codes and the existence of such a polynomial-time algorithm for codes in rank metric is investigated in Chapter 5.

For explicit decoding algorithms there are two important properties: its *performance* and its *complexity*. The performance measures the fraction of correctable errors and directly depends on the minimum distance of the code. The complexity measures the feasibility of an algorithm by counting the number of calculations in the corresponding finite field.

2.1.4 Basics of Convolutional Codes

In contrast to block codes, convolutional codes create a dependency between the different transmitted blocks of length n. For certain channels (e.g., when the number of errors in different blocks fluctuates a lot), their use might be superior to using block codes. In this subsection, we will shortly give basic notations of convolutional codes, mostly based on [Pir88, McE98, Bos98, JZ99]. We also introduce notations for *(partial) unit memory* ((P)UM) codes and prove rate restrictions on them. Distance measures, constructions and decoding of convolutional codes in rank metric are established in Chapter 6.

Definition and Basic Properties of Convolutional Codes

The algebraic theory and description of convolutional codes was investigated by Forney [For70, For73], showing that a q-ary convolutional code of rate $R = k/n$ is a k-dimensional subspace of the n-dimensional vector space $\mathbb{F}_q[D]^k$ over the field of q-ary causal Laurent series (see McEliece's chapter in the handbook of coding theory [McE98] for a detailed description of Laurent series), where D is also called the *delay operator*. Thus, encoding of convolutional codes is given by the following map:

$$\begin{gathered}\text{enc-conv} : \quad \mathbb{F}_q[D]^k \to \mathbb{F}_q[D]^n \\ \mathbf{u}(D) = \mathbf{u}^{(0)} + \mathbf{u}^{(1)}D + \mathbf{u}^{(2)}D^2 + \ldots \mapsto \mathbf{c}(D) = \mathbf{u}(D) \cdot \mathbf{G}(D) = \mathbf{c}^{(0)} + \mathbf{c}^{(1)}D + \mathbf{c}^{(2)}D^2 + \ldots,\end{gathered} \tag{2.6}$$

where $\mathbf{u}^{(i)} = (u_0^{(i)} \; u_1^{(i)} \; \ldots \; u_{k-1}^{(i)})$ and $\mathbf{c}^{(i)} = (c_0^{(i)} \; c_1^{(i)} \; \ldots \; c_{n-1}^{(i)})$, for all integers i. This map shows how to encode the semi-infinite *information sequence* $\mathbf{u}(D)$ into a semi-infinite *code sequence* $\mathbf{c}(D)$.

We call the vectors $\mathbf{u}^{(i)}$ and $\mathbf{c}^{(i)}$ of lengths k and n, respectively, information and code *blocks*. The important observation is that $\mathbf{c}^{(i)}$ is a function of not only $\mathbf{u}^{(i)}$, but also of $\mathbf{u}^{(i-1)}, \mathbf{u}^{(i-2)}, \ldots$, where the length of this influence is determined by the *memory* of the convolutional encoder. Further, we consider only *causal* sequences, i.e., $\mathbf{u}^{(i)} = \mathbf{0}$ and $\mathbf{c}^{(i)} = \mathbf{0}$ for all $i < 0$. For short-hand notation, we also denote the semi-infinite sequences by $\mathbf{u} = (\mathbf{u}^{(0)} \; \mathbf{u}^{(1)} \; \mathbf{u}^{(2)} \ldots)$ and $\mathbf{c} = (\mathbf{c}^{(0)} \; \mathbf{c}^{(1)} \; \mathbf{c}^{(2)} \ldots)$.

The matrix $\mathbf{G}(D) \in \mathbb{F}_q[D]^{k \times n}$ is called *generator matrix* and defines a convolutional code as follows.

Definition 2.9 (Convolutional Code).
A linear convolutional code C *over* $\mathbb{F}_q$ *of rate* $R = k/n$ *is defined by its* $k \times n$ *generator matrix of rank* k*:*

$$\mathbf{G}(D) = \big(g_{i,j}(D)\big)_{j \in [0,n-1]}^{i \in [0,k-1]},$$

where $g_{i,j}(D) = g_{i,j}^{(0)} + g_{i,j}^{(1)} D + \cdots + g_{i,j}^{(\mu)} D^{\mu}$ *and* $g_{i,j}^{(l)} \in \mathbb{F}_q, \forall l \in [0, \mu], \forall i \in [0, k-1]$ *and* $\forall j \in [0, n-1]$. *The parameter* μ *denotes the memory of* $\mathbf{G}(D)$ *(see Definition 2.10).*

In general, $g_{i,j}(D)$ is a rational function, $\forall i \in [0, k-1], j \in [0, n-1]$. If $g_{i,j}(D)$ is a polynomial in D, for all i, j, then $\mathbf{G}(D)$ is called *polynomial* generator matrix and it can be realized by a finite impulse response filter, see [JZ99, Bos98]. We restrict ourselves to such generator matrices in the following.

We strictly distinguish the terms "convolutional code", "generator matrix" and "convolutional encoder". A convolutional code is a set of infinite cardinality, which contains all sequences, defined by the mapping enc-conv (2.6). The generator matrix $\mathbf{G}(D)$ explicitly defines the mapping between information and code sequences and therefore, there are several generator matrices $\mathbf{G}(D)$ for one code. The encoder is a linear sequential circuit, which realizes $\mathbf{G}(D)$, and for one generator matrix, there are several encoders.

The memory and constraint length are properties of the generator matrix. In the literature, there are different notations for them; we follow Forney's notations [For70].

Definition 2.10 (Constraint Length and Memory).
The i-th constraint length of a polynomial generator matrix $\mathbf{G}(D)$ is

$$\nu_i \overset{\text{def}}{=} \max_{j\in[0,n-1]} \Big\{ \deg g_{i,j}(D) \Big\}, \quad \forall i \in [0, k-1].$$

The memory is $\mu \overset{\text{def}}{=} \max_{i\in[0,k-1]}\{\nu_i\}$, and the overall constraint length is $\nu \overset{\text{def}}{=} \sum_{i=0}^{k-1} \nu_i$.

The following remark shows several further properties of the generator matrix, most of them are due to Forney [For70] and Johannesson and Zigangirov [JZ99].

Remark 2.2 (Further Definitions and Properties).

- *Two convolutional generator matrices are called* ***equivalent****, if they generate the same code.*
- *A convolutional generator matrix is* ***catastrophic*** *if there is an information sequence $\mathbf{u}(D)$ with infinitely many non-zero elements that results in a code sequence with finitely many non-zero elements.*
- *A convolutional generator matrix is* ***delay-free*** *if at least one of its entries $g_{i,j}^{(0)}$ is non-zero.*
- *A convolutional generator matrix $\mathbf{G}(D)$ is called* ***basic*** *if it is polynomial and has a polynomial right inverse $\mathbf{G}^{-1}(D)$ such that $\mathbf{I}_k = \mathbf{G}(D) \cdot \mathbf{G}^{-1}(D)$, where $\mathbf{I}_k$ is the $k \times k$ identity matrix.*
- *A convolutional generator matrix $\mathbf{G}(D)$ is an* ***encoding matrix*** *if $\mathbf{G}(0)$ has full rank. An encoding matrix is delay-free. A basic encoding matrix is non-catastrophic.*
- *A convolutional encoder is called* ***obvious realization*** *of $\mathbf{G}(D)$ if it has k shift registers and the length of the i-th register is ν_i.*
- *A basic convolutional generator matrix $\mathbf{G}(D)$ is called* ***minimal*** *if its overall constraint length ν in the obvious realization is equal to the maximum degree of its $k \times k$ subdeterminants.*

A polynomial parity-check matrix $\mathbf{H}(D) \in \mathbb{F}_q[D]^{(n-k)\times n}$ of C has full rank and is defined such that for every codeword $\mathbf{c}(D) \in \mathsf{C}$:

$$\mathbf{c}(D) \cdot \mathbf{H}^T(D) = \mathbf{0}.$$

We denote the entries of the parity-check matrix by $\mathbf{H}(D) = \big(h_{i,j}(D)\big)_{j\in[0,n-1]}^{i\in[0,n-k-1]}$, where $h_{i,j}(D) = h_{i,j}^{(0)} + h_{i,j}^{(1)}D + h_{i,j}^{(2)}D^2 + \dots + h_{i,j}^{(\mu_H)}D^{\mu_H}$ and $h_{i,j}^{(l)} \in \mathbb{F}_q$, $\forall l \in [0, \mu_H]$ and $i \in [0, n-k-1]$, $j \in [0, n-1]$. The value μ_H denotes the memory of the dual code, shortly called *dual memory*.

We can rewrite $\mathbf{G}(D) = \mathbf{G}^{(0)} + \mathbf{G}^{(1)}D + \mathbf{G}^{(2)}D^2 + \dots + \mathbf{G}^{(\mu)}D^{\mu}$ and $\mathbf{H}(D) = \mathbf{H}^{(0)} + \mathbf{H}^{(1)}D + \mathbf{H}^{(2)}D^2 + \dots + \mathbf{H}^{(\mu_H)}D^{\mu_H}$ and represent both as semi-infinite matrices over $\mathbb{F}_q$:

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}^{(0)} & \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(\mu)} & & \\ & \mathbf{G}^{(0)} & \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(\mu)} & \\ & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} \mathbf{H}^{(0)} & & \\ \mathbf{H}^{(1)} & \mathbf{H}^{(0)} & \\ \vdots & \mathbf{H}^{(1)} & \ddots \\ \mathbf{H}^{(\mu_H)} & \vdots & \ddots \\ & \mathbf{H}^{(\mu_H)} & \ddots \\ & & \ddots \end{pmatrix}, \tag{2.7}$$

where $\mathbf{G}^{(i)} \in \mathbb{F}_q^{k\times n}$ and $\mathbf{H}^{(j)} \in \mathbb{F}_q^{(n-k)\times n}$, $\forall i \in [0, \mu]$, $j \in [0, \mu_H]$. These matrices are defined such that $\mathbf{c} = (\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \mathbf{c}^{(2)} \dots) = \mathbf{u} \cdot \mathbf{G} = (\mathbf{u}^{(0)}\ \mathbf{u}^{(1)}\ \mathbf{u}^{(2)} \dots) \cdot \mathbf{G}$ and $\mathbf{c} \cdot \mathbf{H}^T = (\mathbf{0}\ \mathbf{0}\ \dots)$. In general, the memories are not equal, i.e., $\mu \neq \mu_H$. If both $\mathbf{G}$ and $\mathbf{H}$ are in minimal basic encoding form, the overall constraint length ν is the same in both representations [For70, Theorem 7].

In practical realizations, it does not make sense to consider (semi-)infinite sequences and therefore, throughout this thesis, we consider only linear *zero-forced terminated* convolutional codes. Such a code C is defined by the following $Nk \times (n(N+\mu))$ terminated generator matrix $\mathbf{G}_{\mathsf{term}}$ over $\mathbb{F}_q$, for some integer N:

$$\mathbf{G}_{\mathsf{term}} = \begin{pmatrix} \mathbf{G}^{(0)} & \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(\mu)} & & & \\ & \mathbf{G}^{(0)} & \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(\mu)} & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & \mathbf{G}^{(0)} & \mathbf{G}^{(1)} & \dots & \mathbf{G}^{(\mu)} \end{pmatrix}, \tag{2.8}$$

i.e., we cut the matrix $\mathbf{G}$ from (2.7) after N rows. Each codeword of C is a sequence of $N+\mu$ blocks of length n over $\mathbb{F}_q$, i.e., $\mathbf{c} = (\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \dots\ \mathbf{c}^{(N+\mu-1)})$.

Convolutional codes can be described by a (minimal) code trellis and ML decoding is possible with the Viterbi algorithm [Vit67]. However, we do not explain this here and refer to the literature [McE98, Bos98, JZ99].

(Partial) Unit Memory Codes

(P)UM codes are a special class of convolutional codes of memory $\mu = 1$, introduced by Lee and Lauer [Lee76, Lau79]. The semi-infinite generator matrix consists therefore of two $k \times n$ submatrices $\mathbf{G}^{(0)}$ and $\mathbf{G}^{(1)}$. These matrices both have full rank k if we construct a $\mathsf{UM}(n,k)$ unit memory code.

For a $\mathsf{PUM}(n, k|k^{(1)})$ partial unit memory code over $\mathbb{F}_q$, $\mathrm{rk}(\mathbf{G}^{(0)}) = k$ and $\mathrm{rk}(\mathbf{G}^{(1)}) = k^{(1)} < k$ has to hold. W.l.o.g., for PUM codes, we assume that the lowermost $k - k^{(1)}$ rows of $\mathbf{G}^{(1)}$ are zero and we denote:

$$\mathbf{G}^{(0)} = \begin{pmatrix} \mathbf{G}^{(00)} \\ \mathbf{G}^{(01)} \end{pmatrix}, \quad \mathbf{G}^{(1)} = \begin{pmatrix} \mathbf{G}^{(10)} \\ \mathbf{0} \end{pmatrix}, \tag{2.9}$$

where $\mathbf{G}^{(00)}$ and $\mathbf{G}^{(10)}$ are $k^{(1)} \times n$ matrices and $\mathbf{G}^{(01)}$ is a $(k-k^{(1)}) \times n$-matrix over $\mathbb{F}_q$. The encoding rule for each code block of a (P)UM code is given by

$$\mathbf{c}^{(i)} = \mathbf{u}^{(i)} \cdot \mathbf{G}^{(0)} + \mathbf{u}^{(i-1)} \cdot \mathbf{G}^{(1)}, \quad \forall i = 0, 1, \dots, \tag{2.10}$$

where $\mathbf{u}^{(i)}$ and $\mathbf{u}^{(i-1)} \in \mathbb{F}_q^k$ for all i. The memory of (P)UM codes is $\mu = 1$, the overall constraint length of UM codes is $\nu = k$ and of PUM codes $\nu = k^{(1)}$ due to Definition 2.10.

In the following, we derive restrictions on the code rate of (P)UM codes when a certain number of full-rank submatrices of $\mathbf{H}$, denoted by $\mathbf{H}^{(i)}$ as in (2.7), should exist. This full-rank condition, $\mathrm{rk}(\mathbf{H}^{(i)}) = n-k$, $\forall i \in [0, \mu_H]$, is used in one of our constructions of PUM codes based on Gabidulin codes (see Subsection 6.2.1).

Lemma 2.6 (Rate Restriction for Unit Memory Codes).
Let the parity-check matrix $\mathbf{H}$ of a $\mathsf{UM}(n,k)$ code be in minimal basic encoding form and let it consist of $\mu_H + 1$ full-rank submatrices $\mathbf{H}^{(i)}$, see (2.7), for $\mu_H \geq 1$. Then, the $\mathsf{UM}(n,k)$ unit memory code with overall constraint length $\nu = k$ has code rate

$$R = \frac{\mu_H}{\mu_H + 1}.$$

Proof. The overall constraint length ν is the same for the generator matrix $\mathbf{G}$ and the parity-check matrix $\mathbf{H}$ if both are in minimal basic encoding form [For70]. Since $\mathrm{rk}(\mathbf{H}^{(i)}) = n-k$, $\forall i \in [1, \mu_H]$, we obtain $\nu = \mu_H \cdot (n-k)$. On the other hand, the UM code is defined by a generator matrix $\mathbf{G}$ with $\nu = k$, hence, $k = \mu_H \cdot (n-k)$ and the statement follows. ∎

In a similar way, we can establish a rate restriction for PUM codes.

Lemma 2.7 (Rate Restriction for Partial Unit Memory Codes).
Let the parity-check matrix $\mathbf{H}$ *of a* $\mathsf{PUM}(n, k|k^{(1)})$ *code be in minimal basic encoding form and let it consist of* $\mu_H + 1$ *full-rank submatrices* $\mathbf{H}^{(i)}$*, see* (2.7)*, for* $\mu_H \geq 1$*. Then, the partial unit memory code* $\mathsf{PUM}(n, k|k^{(1)})$ *with* $\nu = k^{(1)} < k$ *has code rate*

$$R = \frac{k}{n} > \frac{\mu_H}{\mu_H + 1}.$$

Proof. As before, ν is the same for $\mathbf{G}$ and $\mathbf{H}$ in minimal basic encoding form, [For70]. Since $\mathrm{rk}(\mathbf{H}^{(i)}) = n - k$ for all i, we have $\nu = \mu_H \cdot (n - k)$. For a PUM code $\nu = k^{(1)} < k$, hence, $\mu_H \cdot (n - k) < k$. ∎

The following theorem guarantees that for any parity-check matrix of certain rate, there is always a corresponding generator matrix having memory $\mu = 1$ and thus, defines a (P)UM code. This fact is useful in order to construct (P)UM codes based on a *parity-check* matrix.

Theorem 2.1 ((P)UM Code from Parity-Check Matrix).
Let $\mathbf{H}$ *be a semi-infinite parity-check matrix as in* (2.7) *in minimal basic encoding form of a convolutional code* C*, where* $\mathbf{H}^{(i)} \in \mathbb{F}_q^{(n-k)\times n}$ *has full rank,* $\forall i \in [0, \mu_H]$*, and let* $R = k/n \geq \mu_H/(\mu_H + 1)$ *with* $\mu_H \geq 1$*.*

Then, there is a generator matrix $\mathbf{G}$ *of* C *such that* C *is a (partial) unit memory code.*

Proof. The constraint length of $\mathbf{H}$ is $\nu = \mu_H(n - k)$. Since $n \leq k(\mu_H + 1)/\mu_H$, we obtain:

$$\nu = \mu_H(n - k) \leq k(\mu_H + 1) - k\mu_H = k.$$

Due to [For70], the overall constraint length ν of dual minimal encoders is equal and thus, of $\mathbf{G}$ and $\mathbf{H}$ if both are in minimal form. We choose $\mathbf{G}$ to be in minimal basic encoding form (which is always possible). Since it is in encoding form, $\mathrm{rk}(\mathbf{G}^{(0)}) = k$.
Since $\nu > 0$, the memory is $\mu \geq 1$. Corollary 2 and the corresponding remark in [For73] imply that $\mathbf{G}$ can be chosen such that μ is equal to $\lceil \nu/k \rceil \leq \lceil k/k \rceil = 1$ (in [For73, Corollary 2] the roles of $\mathbf{G}$ and $\mathbf{H}$ are interchanged). Hence, we can choose $\mathbf{G}$ such that $\mu = 1$.
Since $\mathrm{rk}(\mathbf{G}^{(0)}) = k$ and $\mu = 1$, the generator matrix $\mathbf{G}$ defines a (partial) unit memory code. ∎

2.2 Linearized Polynomials

Linearized polynomials constitute a *non-commutative* ring and will later provide the definition of *Gabidulin codes.* Apart from their application to coding theory, linearized polynomials are used e.g. in root-finding of *usual* polynomials and as permutation polynomials in cryptography.

They are also called q-polynomials and were introduced in 1933 by Ore [Ore33a] as a special case of *skew polynomials* [Ore33b]. The theory of skew polynomials is quite rich and widely investigated [Ore33b, Jac43, Gie98, Jac10] and it is even possible to construct error-correcting codes based on skew polynomials [BGU07, BU09b, BU09a, CLU09, BU12]. Skew polynomials become linearized polynomials when the derivation is zero and the Frobenius automorphism is used, i.e., when we consider only $\mathbb{F}_q$-linear maps. Gabidulin codes are based on linearized polynomials and therefore, we restrict ourselves to their description without going into detail about the theory of skew polynomials.

After basic definitions and properties (Subsection 2.2.1), we briefly show how operations with linearized polynomials work (Subsection 2.2.2), give their connection to linear maps (Subsection 2.2.3) and define the q-transform and its inverse (Subsection 2.2.4). In Chapter 3, the q-transform turns out to be a useful tool when establishing an efficient decoding algorithm for Gabidulin codes.

2.2.1 Definition and Properties

Definition 2.11 (Linearized Polynomial).
A polynomial $a(x)$ is a linearized polynomial if it has the form

$$a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}, \quad a_i \in \mathbb{F}_{q^m}, \forall i \in [0, d_a].$$

The non-commutative univariate linearized polynomial ring with indeterminate x, consisting of all such polynomials over $\mathbb{F}_{q^m}$, is denoted by $\mathbb{L}_{q^m}[x]$.

If the coefficient a_{d_a} is non-zero, we call $\deg_q a(x) \overset{\text{def}}{=} d_a$ the *q-degree* of $a(x)$.

Recall that for any $B \in \mathbb{F}_q$, $B^{[i]} = B$ holds for any integer i. This provides the following lemma about evaluating linearized polynomials.

Lemma 2.8 (Evaluation of a Linearized Polynomial [Ber84, Theorem 11.12]).
Let $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, let $a(x)$ be a linearized polynomial as in Definition 2.11 and let $b \in \mathbb{F}_{q^m}$. Denote $\text{ext}_{\mathcal{B}}(b) = (B_0 \ B_1 \ \dots \ B_{m-1})^T \in \mathbb{F}_q^{m\times 1}$ as in Definition 2.1. Then,

$$a(b) = \sum_{i=0}^{m-1} B_i a\big(\beta_i\big).$$

Lemma 2.8 establishes the origin of the name *linearized* polynomials: for all $A_1, A_2 \in \mathbb{F}_q$ and all $b_1, b_2 \in \mathbb{F}_{q^m}$ and $a(x) \in \mathbb{L}_{q^m}[x]$, the following holds:

$$a\big(A_1 b_1 + A_2 b_2\big) = A_1 a\big(b_1\big) + A_2 a\big(b_2\big).$$

Hence, any $\mathbb{F}_q$-linear combination of roots of a linearized polynomial $a(x)$ is also a root of $a(x)$.

Theorem 2.2 (Roots of a Linearized Polynomial [Ber84, Theorem 11.31]).
Let $a(x) \in \mathbb{L}_{q^m}[x]$ be a linearized polynomial and let the extension field $\mathbb{F}_{q^s}$ of $\mathbb{F}_{q^m}$ contain all roots of $a(x)$. Then, its roots form a linear space over $\mathbb{F}_q$ (a subspace of $\mathbb{F}_{q^s}$) and each root has the same multiplicity, which is a power of q.

The roots of $a(x)$ form a linear space of dimension $d_r \leq d_a$. Let $\{\beta_0, \beta_1, \dots, \beta_{d_r-1}\}$ be a basis of this d_r-dimensional root space. Then, each distinct root $r \in \mathbb{F}_{q^s}$ of $a(x)$ can be expressed uniquely as $r = \sum_{i=0}^{d_r-1} R_i \beta_i$, where $R_i \in \mathbb{F}_q, \forall i$. Conversely, the following lemma shows that the unique minimal subspace polynomial is always a linearized polynomial.

Lemma 2.9 (Minimal Subspace Polynomial [LN96, Theorem 3.52]).
Let $\mathcal{U}$ be a linear subspace of $\mathbb{F}_q^m$, considered as a vector space over $\mathbb{F}_q$. Let $u_0, u_1, \dots, u_{\dim(\mathcal{U})-1} \in \mathbb{F}_{q^m}$ be a basis of this subspace. Then, the minimal subspace polynomial

$$M_{u_0,u_1,\dots,u_{\dim(\mathcal{U})-1}}(x) \overset{\text{def}}{=} \prod_{\mathbf{u}\in\mathcal{U}} \big(x - \text{ext}_{\mathcal{B}}^{-1}(\mathbf{u})\big),$$

is a linearized polynomial over $\mathbb{F}_{q^m}$ *of* q*-degree* $\dim(\mathcal{U})$.

The q*-Vandermonde matrix* was introduced by Moore in [Moo96] and plays an important role in linearized interpolation, evaluation and the q-transform. For a vector $\mathbf{a} = (a_0 \; a_1 \; \dots \; a_{n-1}) \in \mathbb{F}_{q^m}^n$, we obtain the $s \times n$ q-Vandermonde matrix by the following map:

$$\begin{aligned} \mathrm{qvan}_s : \quad & \mathbb{F}_{q^m}^n \to \mathbb{F}_{q^m}^{s\times n} \\ \mathbf{a} = (a_0 \; a_1 \; \dots \; a_{n-1}) \mapsto \mathrm{qvan}_s(\mathbf{a}) & \overset{\mathrm{def}}{=} \begin{pmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_0^{[1]} & a_1^{[1]} & \dots & a_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ a_0^{[s-1]} & a_1^{[s-1]} & \dots & a_{n-1}^{[s-1]} \end{pmatrix}. \end{aligned} \tag{2.11}$$

Lemma 2.10 (Determinant of q-Vandermonde Matrix [LN96, Lemma 3.15]).
Let $\mathbf{a} = (a_0 \; a_1 \; \dots \; a_{n-1}) \in \mathbb{F}_{q^m}^n$. *Then, the determinant of the square* $n \times n$ q*-Vandermonde matrix, defined as in* (2.11)*, is*

$$\det\big(\mathrm{qvan}_n(\mathbf{a})\big) = a_0 \prod_{j=0}^{n-2} \prod_{B_0,\dots,B_j \in \mathbb{F}_q} \Bigg(a_{j+1} - \sum_{h=0}^{j} B_h a_h\Bigg).$$

Hence, $\det(\mathrm{qvan}_n(\mathbf{a})) \neq 0$ if and only if $a_0, a_1, \dots, a_{n-1}$ are linearly independent over $\mathbb{F}_q$. If a_0, $a_1, \dots, a_{n-1}$ are linearly independent over $\mathbb{F}_q$, then $\mathrm{qvan}_s(\mathbf{a})$ has rank $\min\{s, n\}$.

2.2.2 Basic Operations

The usual multiplication of two linearized polynomials $a(x)$ and $b(x)$ is not necessarily a linearized polynomial. However, the (usual) addition and the composition $a(b(x))$ convert the set of linearized polynomials into a non-commutative ring with identity element $x^{[0]} = x$. The linearized composition is often called *symbolic product* and will be denoted by $a(x) \circ b(x) = a(b(x))$. It is associative and distributive, but in general for $a(x)$, $b(x) \in \mathbb{L}_{q^m}[x]$, it is non-commutative[4], i.e., $a(b(x)) \neq b(a(x))$.

Let d_a and d_b denote the q-degrees of $a(x)$ and $b(x)$, respectively. Then, the linearized composition $c(x) = \sum_{j=0}^{d_a+d_b} c_j x^{[j]} = a(b(x))$ has q-degree at most $d_a + d_b$ and its coefficients are:

$$c_j = \big[a(b(x))\big]_j = \sum_{i=0}^{j} a_i b_{j-i}^{[i]}, \quad \forall j \in [0, d_a + d_b], \tag{2.12}$$

with $a_i = 0$ for $i > d_a$ and $b_i = 0$ for $i > d_b$. When we consider the linearized composition modulo $(x^{[m]} - x)$, i.e., $c(x) = \sum_{j=0}^{m-1} c_j x^{[j]} = a(b(x)) \mod (x^{[m]} - x)$ for $d_a, d_b < m$, then its coefficients can be calculated by:

$$c_j = \big[a(b(x)) \mod (x^{[m]} - x)\big]_j = \sum_{i=0}^{m-1} a_i b_{j-i}^{[i]} = \sum_{h=0}^{m-1} a_{j-h} b_h^{[j-h]}, \quad \forall j \in [0, m-1], \tag{2.13}$$

with $a_i = 0$ for $i > d_a$ and $b_i = 0$ for $i > d_b$ and all indices are calculated modulo m.

In Subsection 3.1.3, we will show that the composition of two linearized polynomials modulo $(x^{[m]} - x)$ is equivalent to multiplying their associated evaluation matrices, which provides an efficient algorithm for calculating the linearized composition.

[4] When all coefficients of $a(x)$ and $b(x)$ lie in the ground field $\mathbb{F}_q$, the linearized composition is commutative.

Ore showed in [Ore33a, Theorem 1] that for any two linearized polynomials $a(x)$ and $b(x)$ in $\mathbb{L}_{q^m}[x]$ with $d_a \geq d_b$, there exist unique polynomials $q_R(x)$, $r_R(x)$ and $q_L(x)$, $r_L(x)$ such that

$$a(x) = q_R\big(b(x)\big) + r_R(x) \quad \text{and} \quad a(x) = b\big(q_L(x)\big) + r_L(x), \tag{2.14}$$

where $\deg_q r_R(x), \deg_q r_L(x) < d_b$. Determining $q_R(x)$ and $r_R(x)$ is called *right linearized (or symbolic) division*, where $q_R(x)$ is the right (linearized) quotient and $r_R(x)$ the right (linearized) remainder. Equivalently, finding $q_L(x)$ and $r_L(x)$ is called *left linearized division.* The right/left linearized division can be done by a recursive procedure (compare [Ore33a, p. 561]). Throughout this thesis, we denote the algorithmic calculation of this right/left linearized division by

$$q_R(x); r_R(x) \leftarrow \textsc{RightDiv}\big(a(x); b(x)\big) \quad \text{and} \quad q_L(x); r_L(x) \leftarrow \textsc{LeftDiv}\big(a(x); b(x)\big).$$

The right and left divisions are shown in the following two algorithms (compare [Ore33a, p. 561]), where the subscripts "R" and "L" for "right" and "left" are omitted.

Algorithm 2.1.
$q(x); r(x) \leftarrow \textsc{RightDiv}\big(a(x); b(x)\big)$

```
Input: $a(x); b(x) \neq 0 \in \mathbb{L}_{q^m}[x]$ with
       $\deg_q a(x) \geq \deg_q b(x)$
Initialize: $i \leftarrow 1$,
            $a^{(1)}(x) \leftarrow a(x)$
            $d_b \overset{\text{def}}{=} \deg_q b(x)$
while $d_i \leftarrow \deg_q a^{(i)}(x) \geq d_b$ do
    $q^{(i)}(x) \leftarrow \frac{a^{(i)}_{d_i}}{b^{[d_i-d_b]}_{d_b}} \cdot x^{[d_i-d_b]}$
    $a^{(i+1)}(x) \leftarrow a^{(i)}(x) - q^{(i)}(b(x))$
    $i \leftarrow i+1$
$q(x) \leftarrow \sum_{j=1}^{i-1} q^{(i)}(x)$
$r(x) \leftarrow a^{(i)}(x)$
Output: $q(x); r(x)$
```

Algorithm 2.2.
$q(x); r(x) \leftarrow \textsc{LeftDiv}\big(a(x); b(x)\big)$

```
Input: $a(x); b(x) \neq 0 \in \mathbb{L}_{q^m}[x]$ with
       $\deg_q a(x) \geq \deg_q b(x)$
Initialize: $i \leftarrow 1$,
            $a^{(1)}(x) \leftarrow a(x)$
            $d_b \overset{\text{def}}{=} \deg_q b(x)$
while $d_i \leftarrow \deg_q a^{(i)}(x) \geq d_b$ do
    $q^{(i)}(x) \leftarrow \left(\frac{a^{(i)}_{d_i}}{b_{d_b}}\right)^{[-d_b]} \cdot x^{[d_i-d_b]}$
    $a^{(i+1)}(x) \leftarrow a^{(i)}(x) - b(q^{(i)}(x))$
    $i \leftarrow i+1$
$q(x) \leftarrow \sum_{j=1}^{i-1} q^{(i)}(x)$
$r(x) \leftarrow a^{(i)}(x)$
Output: $q(x); r(x)$
```

Both algorithms terminate such that $\deg_q r(x) < \deg_q b(x)$.

Since unique right/left linearized quotients and remainders always exist in $\mathbb{F}_{q^m}$ such that (2.14) holds (compare [Ore33a, Theorem 1]), there is a right and left *linearized extended Euclidean algorithm* (LEEA) in the non-commutative ring of linearized polynomials $\mathbb{L}_{q^m}[x]$. Throughout this thesis, we consider only the *right* LEEA, which is given in Algorithm 2.3. The subscript "R" for quotients and remainders is omitted when there is no ambiguity.

Let $r^{(-1)}(x) = a(x)$ and $r^{(0)}(x) = b(x)$ be two linearized polynomials with $\deg_q a(x) \geq \deg_q b(x)$. The right LEEA with a *stopping degree* $d_{stop} > 0$ calculates a linearized *quotient* $q^{(i)}(x)$ and linearized *remainder* $r^{(i)}(x)$ in each step $i > 0$ such that

$$r^{(i)}(x) = r^{(i-2)}(x) - q^{(i)}\big(r^{(i-1)}(x)\big), \tag{2.15}$$

while $\deg_q r^{(i-1)}(x) \geq d_{stop}$. In each of its steps, the q-degree of the remainders decreases, i.e., $\deg_q r^{(i)}(x) < \deg_q r^{(i-1)}(x)$. If $d_{stop} = 1$, the last non-zero remainder $r^{(i-1)}(x) \neq 0$ is the right linearized greatest common divisor of $a(x)$ and $b(x)$. The polynomials $r^{(i)}(x)$ and $q^{(i)}(x)$ are unique in each step of Algorithm 2.3 due to [Ore33a, Theorem 1]. The algorithm returns, amongst others, the first remainder $r_{out}(x)$ such that $\deg_q r_{out}(x) < d_{stop}$.

Algorithm 2.3.
$r_{out}(x); u_{out}(x); v_{out}(x) \leftarrow \text{RightLEEA}\big(a(x); b(x); d_{stop}\big)$

```
Input: $a(x); b(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q a(x) \geq \deg_q b(x)$;
       stopping degree $d_{stop}$
Initialize: $i \leftarrow 1$,
       $r^{(-1)}(x) \leftarrow a(x)$, $r^{(0)}(x) \leftarrow b(x)$,
       $u^{(-1)}(x) \leftarrow 0$, $u^{(0)}(x) \leftarrow x^{[0]}$,
       $v^{(-1)}(x) \leftarrow x^{[0]}$, $v^{(0)}(x) \leftarrow 0$
while $\deg_q r^{(i-1)}(x) \geq d_{stop}$ do
    $q^{(i)}(x); r^{(i)}(x) \leftarrow \text{RightDiv}\big(r^{(i-1)}(x); r^{(i-2)}(x)\big)$
    $u^{(i)}(x) \leftarrow u^{(i-2)}(x) - q^{(i)}(u^{(i-1)}(x))$
    $v^{(i)}(x) \leftarrow v^{(i-2)}(x) - q^{(i)}(v^{(i-1)}(x))$
    $i \leftarrow i + 1$
Output: $r_{out}(x) \leftarrow r^{(i-1)}(x)$; $u_{out}(x) \leftarrow u^{(i-1)}(x)$;
        $v_{out}(x) \leftarrow v^{(i-1)}(x)$
```

The matrix-matrix multiplication for two matrices $\mathbf{A} = \big(a_{i,j}(x)\big)_{j\in[0,n-1]}^{i\in[0,m-1]} \in \mathbb{L}_{q^m}[x]^{m\times n}$ and $\mathbf{B} = \big(b_{i,j}(x)\big)_{j\in[0,l-1]}^{i\in[0,n-1]} \in \mathbb{L}_{q^m}[x]^{n\times l}$ is a matrix $\mathbf{C} = \mathbf{A} \circ \mathbf{B} = \big(c_{i,j}(x)\big)_{j\in[0,l-1]}^{i\in[0,m-1]} \in \mathbb{L}_{q^m}[x]^{m\times l}$ with elements:

$$c_{i,j}(x) = \sum_{h=0}^{n-1} a_{i,h}\big(b_{h,j}(x)\big), \quad \forall i \in [0, m-1], j \in [0, l-1].$$

In order to use matrix-matrix multiplication in the description of the LEEA, define the following matrices:

$$\mathbf{Q}^{(i)} \overset{\text{def}}{=} \begin{pmatrix} 0 & x^{[0]} \\ x^{[0]} & -q^{(i)}(x) \end{pmatrix}, \quad \mathbf{Q}^{(i,j)} \overset{\text{def}}{=} \mathbf{Q}^{(i)} \circ \mathbf{Q}^{(i-1)} \circ \dots \circ \mathbf{Q}^{(j)}, \quad \forall i \geq j \geq 1. \tag{2.16}$$

Hence, $\mathbf{Q}^{(i,i)} = \mathbf{Q}^{(i)}$. The recursion (2.15) of the LEEA can then be rewritten by:

$$\begin{pmatrix} r^{(i-1)}(x) \\ r^{(i)}(x) \end{pmatrix} = \mathbf{Q}^{(i)} \circ \begin{pmatrix} r^{(i-2)}(x) \\ r^{(i-1)}(x) \end{pmatrix} = \mathbf{Q}^{(i,j)} \circ \begin{pmatrix} r^{(j-2)}(x) \\ r^{(j-1)}(x) \end{pmatrix} = \mathbf{Q}^{(i,1)} \circ \begin{pmatrix} r^{(-1)}(x) \\ r^{(0)}(x) \end{pmatrix}. \tag{2.17}$$

Further, we introduce auxiliary polynomials, needed for decoding Gabidulin codes. Let $u^{(-1)}(x) = 0$, $u^{(0)}(x) = x^{[0]}$ and $v^{(-1)}(x) = x^{[0]}, v^{(0)}(x) = 0$ (see also Algorithm 2.3). Then, we calculate $u^{(i)}(x)$ and $v^{(i)}(x)$, for $i > 0$, recursively, similar to the remainders:

$$\begin{pmatrix} u^{(i-1)}(x) \\ u^{(i)}(x) \end{pmatrix} = \mathbf{Q}^{(i)} \circ \begin{pmatrix} u^{(i-2)}(x) \\ u^{(i-1)}(x) \end{pmatrix}, \quad \begin{pmatrix} v^{(i-1)}(x) \\ v^{(i)}(x) \end{pmatrix} = \mathbf{Q}^{(i)} \circ \begin{pmatrix} v^{(i-2)}(x) \\ v^{(i-1)}(x) \end{pmatrix}.$$

By means of these auxiliary polynomials $u^{(i)}(x), v^{(i)}(x)$, each remainder can be rewritten as follows [Gab85, Equation (28)]:

$$r^{(i)}(x) = v^{(i)}\big(a(x)\big) + u^{(i)}\big(b(x)\big), \qquad \forall i \geq 0. \tag{2.18}$$

Similar to (2.17), we obtain

$$\begin{pmatrix} u^{(i-1)}(x) \\ u^{(i)}(x) \end{pmatrix} = \mathbf{Q}^{(i,1)} \circ \begin{pmatrix} 0 \\ x^{[0]} \end{pmatrix}, \quad \begin{pmatrix} v^{(i-1)}(x) \\ v^{(i)}(x) \end{pmatrix} = \mathbf{Q}^{(i,1)} \circ \begin{pmatrix} x^{[0]} \\ 0 \end{pmatrix}. \tag{2.19}$$

Thus, it is sufficient to calculate $\mathbf{Q}^{(j,1)}$ if we want to determine $r^{(j)}(x), u^{(j)}(x)$ and $v^{(j)}(x)$.

2.2.3 Connection to Linear Maps

Recall that Lemma 2.8 implies for any $a(x) \in \mathbb{L}_{q^m}[x]$ that $a(A_1b_1 + A_2b_2) = A_1a(b_1) + A_2a(b_2)$ holds for all $A_1, A_2 \in \mathbb{F}_q$ and for all $b_1, b_2 \in \mathbb{F}_{q^m}$. Hence, the linearized polynomial $a(x) \in \mathbb{L}_{q^m}[x]$ of q-degree $d_a < m$ induces an $\mathbb{F}_q$-linear map a from $\mathbb{F}_{q^m}$ to itself. The kernel of this map a is equivalent to the root space of $a(x)$, i.e.,

$$\ker(a) = \{b \in \mathbb{F}_{q^m} : a(b) = 0\},$$

and can also be seen as the right kernel of an associated matrix $\mathbf{A}$. This associated matrix can be obtained by evaluating $a(x)$ at a basis $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and representing the result over $\mathbb{F}_q$, i.e.:

$$\mathbf{A} \overset{\text{def}}{=} \text{ext}_{\mathcal{B}}\big((a(\beta_0)\ a(\beta_1)\ \dots\ a(\beta_{m-1})) \big) \in \mathbb{F}_q^{m \times m}.$$

We call this matrix *associated evaluation matrix* in the following. The kernel of the map a, denoted by $\ker(a)$, is equivalent to the right kernel of $\mathbf{A}$, denoted by $\ker(\mathbf{A})$. The *rank nullity theorem* relates the dimensions of the (right) kernel and the image (column space) of this matrix, respectively of this map:

$$\dim(\ker(a)) + \dim(\text{im}(a)) = m.$$

Moreover,

$$\dim(\text{im}(a)) = \text{rk}(\mathbf{A}).$$

The following lemma shows the connection between roots of $a(x)$ and the rank of the associated matrix.

Lemma 2.11 (Root Space and Rank).
Let $a(x) \in \mathbb{L}_{q^m}[x]$ be a non-zero linearized polynomial of q-degree $d_a < m$. Then, the rank of the associated evaluation matrix is $\text{rk}(\mathbf{A}) \geq m - d_a$.

Proof. Since $\deg_q a(x) = d_a$, it has at most q^{d_a} roots in $\mathbb{F}_{q^m}$ and the dimension of the root space is at most d_a. This root space is equivalent to the right kernel of $\mathbf{A}$, hence, $\dim \ker(\mathbf{A}) \leq d_a$. Due to the rank nullity theorem and since $\dim \text{im}(a) = \text{rk}(\mathbf{A})$, the statement follows. ∎

The kernel of the map a is therefore the root space of $a(x)$, represented as a vector space over $\mathbb{F}_q$. Consider now a second linearized polynomial $b(x)$, then the composition $b(a(x)) \mod (x^{[m]} - x)$ is a linear map $b(a)$, whose kernel includes the kernel of a. This is formally stated in the following lemma, which we use when decoding (interleaved) Gabidulin codes.

Lemma 2.12 (Row Space of Composition).
Let $a(x)$ and $b(x)$ denote two linearized polynomials in $\mathbb{L}_{q^m}[x]$ with $\deg_q a(x), \deg_q b(x) < m$. Let $c(x) = b(a(x))$ and let $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Let

$$\mathbf{A} = \text{ext}_{\mathcal{B}}\big((a(\beta_0)\ a(\beta_1)\ \dots\ a(\beta_{m-1})) \big), \quad \mathbf{C} = \text{ext}_{\mathcal{B}}\big((c(\beta_0)\ c(\beta_1)\ \dots\ c(\beta_{m-1})) \big).$$

Then, for the row spaces the following holds:

$$\mathcal{R}_q(\mathbf{C}) \subseteq \mathcal{R}_q(\mathbf{A}).$$

Proof. Consider the linearized polynomials as linear maps over $\mathbb{F}_{q^m}$. Then, the kernel of the map a is equivalent to the roots of $a(x)$ in $\mathbb{F}_{q^m}$, considered as a vector space over $\mathbb{F}_q$. Since the roots of $a(x)$ are also roots of $c(x) = b(a(x))$, the kernels are connected by $\ker(a) \subseteq \ker(c)$. Hence, for the right kernels $\ker(\mathbf{A}) \subseteq \ker(\mathbf{C})$ holds, and the row spaces are related by $\mathcal{R}_q(\mathbf{C}) \subseteq \mathcal{R}_q(\mathbf{A})$. ∎

2.2.4 The (Inverse) q-Transform

Gabidulin codes can be defined either by means of evaluation and interpolation of linearized polynomials or by means of the q-transform. This subsection shows basic properties of the (inverse) q-transform.

Lemma 2.3 guarantees that for any s dividing m, there is a normal basis in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$. For such a normal basis $\mathcal{B}_N$, the q-transform of a linearized polynomial $a(x)$ is defined as follows.

Definition 2.12 (q-Transform).
Let a linearized polynomial $a(x) = \sum_{i=0}^{s-1} a_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ (or a vector $\mathbf{a} = (a_0\ a_1\ \dots\ a_{s-1}) \in \mathbb{F}_{q^m}^s$) be given, where $s \mid m$, and let $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[s-1]}\}$, for $\beta \in \mathbb{F}_{q^m}$, be a normal basis of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$.

Then, the q-transform of $a(x)$ with respect to $\mathcal{B}_N$ is the linearized polynomial $\widehat{a}(x) = \sum_{j=0}^{s-1} \widehat{a}_j x^{[j]}$ (or the vector $(\widehat{a}_0\ \widehat{a}_1\ \dots\ \widehat{a}_{s-1}) \in \mathbb{F}_{q^m}^s$), given by

$$\widehat{a}_j = a\big(\beta^{[j]}\big) = \sum_{i=0}^{s-1} a_i \beta^{[i+j]}, \qquad \forall j \in [0, s-1]. \tag{2.20}$$

Let $\text{ext}_{\boldsymbol{\beta}}(a_j) \overset{\text{def}}{=} (A_{0,j}\ A_{1,j}\ \dots\ A_{m-1,j})^T$, with $A_{i,j} \in \mathbb{F}_q$ for $i \in [0, m-1]$, denote the vector representation of $a_j \in \mathbb{F}_{q^m}$ over $\mathbb{F}_q$ according to Definition 2.1 using a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. As done in Subsection 2.1.2 for the multiplication of two elements, we can use the multiplication table $\mathbf{T}_m \in \mathbb{F}_q^{m \times m}$ (compare Definition 2.2) to calculate the elements of the q-transform over the ground field $\mathbb{F}_q$.

$$\begin{aligned}\text{ext}_{\boldsymbol{\beta}}(\widehat{a}_j) = \text{ext}_{\boldsymbol{\beta}}\left(a(\beta^{[j]})\right) &= \sum_{i=0}^{d_a} \text{ext}_{\boldsymbol{\beta}}\left(a_i \beta^{[i+j]}\right) \\ &= \sum_{i=0}^{d_a} \left(\mathbf{T}_m^T \cdot \text{ext}_{\boldsymbol{\beta}}(a_i)^{\uparrow i+j}\right)^{\downarrow i+j},\end{aligned} \tag{2.21}$$

where $d_a = \deg_q a(x) < s$, i.e., $a_i = 0$ for $i > d_a$ and $\widehat{a}_j$ can then be obtained by $\text{ext}_{\boldsymbol{\beta}}^{-1}(\text{ext}_{\boldsymbol{\beta}}(\widehat{a}_j))$.

In order to switch between the polynomial and its transformed polynomial, we need an inverse mapping, called the inverse q-transform. The following theorem shows that we actually retrieve the original polynomial from its transform. In [SK09a] this was proved for the special case $s = m$.

Theorem 2.3 (Inverse q-Transform).
Let $\widehat{a}(x) = \sum_{j=0}^{s-1} \widehat{a}_j x^{[j]} \in \mathbb{L}_{q^m}[x]$ denote the q-transform of $a(x) = \sum_{i=0}^{s-1} a_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ as in Definition 2.12, where s divides m, and $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[s-1]}\}$ is a normal basis in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^s}$

over $\mathbb{F}_q$. Further, let $\mathcal{B}_N^\perp = \{\beta^{\perp[0]}, \beta^{\perp[1]}, \dots, \beta^{\perp[s-1]}\}$ be a normal basis, which is dual to $\mathcal{B}_N$. Then,

$$a_i = \widehat{a}\left(\beta^{\perp[i]}\right) = \sum_{j=0}^{m-1} \widehat{a}_j \beta^{\perp[j+i]}, \qquad \forall i \in [0, s-1]. \tag{2.22}$$

We call this the inverse q-transform of $\widehat{a}(x)$ with respect to $\mathcal{B}_N^\perp$.

Proof. The condition $s \mid m$ guarantees that there exists a dual normal basis $\mathcal{B}_N^\perp$ (see Lemma 2.3). Let us denote the following two matrices:

$$\mathbf{B} = \begin{pmatrix} \beta^{[0]} & \beta^{[1]} & \dots & \beta^{[s-1]} \\ \beta^{[1]} & \beta^{[2]} & \dots & \beta^{[0]} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{[s-1]} & \beta^{[0]} & \dots & \beta^{[s-2]} \end{pmatrix}, \quad \mathbf{B}^\perp = \begin{pmatrix} \beta^{\perp[0]} & \beta^{\perp[1]} & \dots & \beta^{\perp[s-1]} \\ \beta^{\perp[1]} & \beta^{\perp[2]} & \dots & \beta^{\perp[0]} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{\perp[s-1]} & \beta^{\perp[0]} & \dots & \beta^{\perp[s-2]} \end{pmatrix}. \tag{2.23}$$

By definition, $(\widehat{a}_0\ \widehat{a}_1\ \dots\ \widehat{a}_{s-1}) = (a_0\ a_1\ \dots\ a_{n-1}) \cdot \mathbf{B}$, see (2.20). Now, if we calculate $\mathbf{a}'$ by $a_i' = \widehat{a}(\beta^{\perp[i]})$ for $i \in [0, s-1]$ as in (2.22), we obtain:

$$\mathbf{a}' = (a_0'\ a_1'\ \dots\ a_{s-1}') = (\widehat{a}_0\ \widehat{a}_1\ \dots\ \widehat{a}_{s-1}) \cdot \mathbf{B}^\perp = (a_0\ a_1\ \dots\ a_{n-1}) \cdot \mathbf{B} \cdot \mathbf{B}^\perp = \mathbf{a} \cdot \mathbf{B} \cdot \mathbf{B}^\perp.$$

Moreover, due to the definition of the dual basis (compare (2.2)) and since $\mathrm{Tr}(\beta^{[i]}\beta^{\perp[i]}) = \mathrm{Tr}(\beta\beta^\perp)^{[i]} = \mathrm{Tr}(\beta\beta^\perp)$, we obtain:

$$\mathbf{B} \cdot \mathbf{B}^\perp = \begin{pmatrix} \mathrm{Tr}(\beta\beta^\perp) & \mathrm{Tr}(\beta\beta^{\perp[1]}) & \dots & \mathrm{Tr}(\beta\beta^{\perp[s-1]}) \\ \mathrm{Tr}(\beta^{[1]}\beta^\perp) & \mathrm{Tr}(\beta^{[1]}\beta^{\perp[1]}) & \dots & \mathrm{Tr}(\beta^{[1]}\beta^{\perp[s-1]}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Tr}(\beta^{[s-1]}\beta^\perp) & \mathrm{Tr}(\beta^{[s-1]}\beta^{\perp[1]}) & \dots & \mathrm{Tr}(\beta^{[s-1]}\beta^{\perp[s-1]}) \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}.$$

Hence, $\mathbf{a}' = \mathbf{a}$, which proves the statement. ∎

Recalling Subsection 2.2.3 shows that the q-transform and its inverse transform provide an efficient tool for switching between the map and its associated evaluated matrix. In terms of interpolation and evaluation, the inverse q-transform can be seen as the evaluation of $\widehat{a}(x)$ at the dual normal basis. Equivalently, the determination of $\widehat{a}(x)$ out of $a(x)$ can be seen as the unique linearized univariate interpolation polynomial of q-degree less than s. In Subsection 3.2.2, we explain how to calculate this unique interpolation polynomial based on linearized Lagrange basis polynomials.

2.3 Codes in Rank Metric

Gabidulin codes, introduced by Delsarte [Del78], Gabidulin [Gab85] and Roth [Rot91], are so-called *maximum rank distance* (MRD) codes since they attain the Singleton-like upper bound with equality. Further, they are also maximum distance separable codes when considered as codes in Hamming metric.

In Subsection 2.3.1, we introduce the rank metric and show fundamental bounds as the Singleton-like and the Gilbert–Varshamov-like bound. We introduce the notation of MRD codes, define Gabidulin codes as the evaluation of degree-restricted linearized polynomials in Subsection 2.3.2, derive their minimum rank distance and show how generator and parity-check matrices can be constructed.

Interleaved Gabidulin codes are defined in Subsection 2.3.3 and their minimum rank distance is proven. Finally, we briefly give basic notations of lifted Gabidulin codes, which constitute a special class of constant-dimension codes (Subsection 2.3.4).

2.3.1 Rank Metric and its Properties

The mapping from Definition 2.1 plays a fundamental role in the context of rank-metric codes. It shows that for a given basis $\mathcal{B}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, there exists a bijective mapping for each vector $\mathbf{a} \in \mathbb{F}_{q^m}^n$ on a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$. Based on this mapping, the *rank weight* and *rank distance* are defined as follows.

Definition 2.13 (Rank Weight and Rank Distance).
Let $\mathbf{a} = (a_0\ a_1\ \dots\ a_{n-1})$, $\mathbf{b} = (b_0\ b_1\ \dots\ b_{n-1}) \in \mathbb{F}_{q^m}^n$ *and let* $\mathbf{A} = \operatorname{ext}_{\mathcal{B}}(\mathbf{a}), \mathbf{B} = \operatorname{ext}_{\mathcal{B}}(\mathbf{b}) \in \mathbb{F}_q^{m \times n}$ *denote the matrix representations with respect to a basis* $\mathcal{B}$ *of* $\mathbb{F}_{q^m}$ *over* $\mathbb{F}_q$ *according to Definition 2.1. The rank weight of* $\mathbf{a}$ *is the rank of its matrix representation over* $\mathbb{F}_q$*, i.e.,*

$$\mathrm{wt}_R(\mathbf{a}) \overset{\text{def}}{=} \mathrm{rk}(\mathbf{a}) = \mathrm{rk}(\mathbf{A}).$$

The rank distance between $\mathbf{a}$ *and* $\mathbf{b}$ *is the rank of the difference of the two matrix representations:*

$$d_R(\mathbf{a}, \mathbf{b}) \overset{\text{def}}{=} \mathrm{rk}(\mathbf{a} - \mathbf{b}) = \mathrm{rk}(\mathbf{A} - \mathbf{B}).$$

Lemma 2.13 (Rank Distance is a Metric).
The rank distance as given in Definition 2.13 is a metric, fulfilling the requirements from Definition 2.3.

Proof. For any matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}_q^{m \times n}$:

- $\mathrm{rk}(\mathbf{A} - \mathbf{B}) \geq 0$ with equality if and only if $\mathbf{A} = \mathbf{B}$, proving positive definiteness;
- $\mathrm{rk}(\mathbf{A} - \mathbf{B}) = \mathrm{rk}(\mathbf{B} - \mathbf{A})$, proving symmetry;
- the known fact $\mathrm{rk}(\mathbf{A} + \mathbf{B}) \leq \mathrm{rk}(\mathbf{A}) + (\mathbf{B})$ shows that the triangle inequality is fulfilled, since $\mathrm{rk}(\mathbf{A} - \mathbf{C}) = \mathrm{rk}(\mathbf{A} - \mathbf{B} + \mathbf{B} - \mathbf{C}) \leq \mathrm{rk}(\mathbf{A} - \mathbf{B}) + \mathrm{rk}(\mathbf{B} - \mathbf{C})$.

∎

A *sphere* in rank metric of radius τ around a word $\mathbf{a} \in \mathbb{F}_{q^m}^n$ is the set of all words in rank distance exactly τ from $\mathbf{a}$ and a *ball* is the set of all words in rank distance at most τ from $\mathbf{a}$. Such a sphere will be denoted by $\mathcal{S}_R^{(\tau)}(\mathbf{a}) = \mathcal{S}_R^{(\tau)}(\mathbf{A})$ and such a ball by $\mathcal{B}_R^{(\tau)}(\mathbf{a}) = \mathcal{B}_R^{(\tau)}(\mathbf{A})$. The cardinality of $\mathcal{B}_R^{(\tau)}(\mathbf{a})$ can obviously be obtained by summing up the cardinalities of the spheres around $\mathbf{a}$ of radius from zero up to τ. The number of matrices of a certain rank is given for example in [MMO04]. Therefore,

$$|\mathcal{S}_R^{(\tau)}(\mathbf{a})| = \begin{bmatrix} m \\ \tau \end{bmatrix} \prod_{j=0}^{\tau-1} (q^n - q^j),$$

$$|\mathcal{B}_R^{(\tau)}(\mathbf{a})| = \sum_{i=0}^{\tau} |\mathcal{S}_R^{(i)}(\mathbf{a})| = \sum_{i=0}^{\tau} \begin{bmatrix} m \\ i \end{bmatrix} \prod_{j=0}^{i-1} (q^n - q^j).$$

Note that the cardinalities of $\mathcal{B}_R^{(\tau)}(\mathbf{a})$ and $\mathcal{S}_R^{(\tau)}(\mathbf{a})$ are independent of the choice of their center.

Recall from Definition 2.4 that a block code over $\mathbb{F}_{q^m}$ of length n is a set of vectors in $\mathbb{F}_{q^m}^n$. The size of this set is the cardinality of the block code. A *linear* block code can be seen as a k-dimensional subspace of $\mathbb{F}_{q^m}^n$ and its cardinality is $M = q^{mk}$, where k denotes the *dimension* of the code. Analog to Definition 2.4, we denote a code in rank metric (not necessarily linear) over $\mathbb{F}_{q^m}$ of length n, cardinality M and minimum rank distance d by $(n, M, d)_R$. A *linear* code in rank metric of length n, dimension k and minimum rank distance d is a special case of the aforementioned and is denoted by $[n, k, d]_R$. The codewords of both can be seen as vectors in $\mathbb{F}_{q^m}^n$ or equivalently as matrices in $\mathbb{F}_q^{m \times n}$. The *minimum rank distance* of a block code is defined as follows.

Definition 2.14 (Minimum Rank Distance).
For a given $(n, M, d)_R$ block code C over $\mathbb{F}_{q^m}$, the minimum rank distance is defined by

$$d \overset{\text{def}}{=} \min_{\substack{\mathbf{c}^{(1)},\mathbf{c}^{(2)} \in \mathsf{C} \\ \mathbf{c}^{(1)} \neq \mathbf{c}^{(2)}}} \Big\{ d_R(\mathbf{c}^{(1)}, \mathbf{c}^{(2)}) = \operatorname{rk}(\mathbf{c}^{(1)} - \mathbf{c}^{(2)}) \Big\}.$$

Corollary 2.1 (Minimum Rank Distance of a Linear Code).
For a linear $[n, k, d]_R$ block code C, the minimum rank distance is the minimum rank weight:

$$d = \min_{\substack{\mathbf{c} \in \mathsf{C} \\ \mathbf{c} \neq \mathbf{0}}} \Big\{ \operatorname{wt}_R(\mathbf{c}) = \operatorname{rk}(\mathbf{c}) \Big\}.$$

The following theorem shows how the minimum rank distance of a linear block code can be determined based on its parity-check matrix.

Theorem 2.4 (Minimum Rank Distance from Parity-Check Matrix [Gab85, Theorem 1]).
Let the $(n-k) \times n$ matrix $\mathbf{H}$ over $\mathbb{F}_{q^m}$ denote the parity-check matrix of a linear $[n, k, d]_R$ block code C over $\mathbb{F}_{q^m}$. If and only if for any matrix $\mathbf{A} \in \mathbb{F}_q^{(\delta-1)\times n}$ of rank $\delta - 1$ the following holds:

$$\operatorname{rk}(\mathbf{A}\mathbf{H}^T) = \delta - 1,$$

and if there exists a matrix $\mathbf{B} \in \mathbb{F}_q^{\delta \times n}$ of rank δ such that

$$\operatorname{rk}(\mathbf{B}\mathbf{H}^T) < \delta,$$

then C has minimum rank distance $d = \delta$.

The *maximum cardinality* of a code of length n and minimum rank distance d over $\mathbb{F}_{q^m}$ is denoted by $\mathsf{A}^R_{q^m}(n, d)$. On the one hand, $\mathsf{A}^R_{q^m}(n, d)$ is an upper bound on the cardinality of any $(n, M, d)_R$ code over $\mathbb{F}_{q^m}$, i.e., $M \leq \mathsf{A}^R_{q^m}(n, d)$. On the other hand, the definition of the maximum cardinality $\mathsf{A}^R_{q^m}(n, d)$ implies that an $(n, M, d)_R$ code of cardinality $M = \mathsf{A}^R_{q^m}(n, d)$ exists.

The following theorem states analogs of the *sphere packing* (Hamming) and *Gilbert–Varshamov bound* in rank metric, which can be proved similar to Hamming metric [GY06, Loi08, GY08b, Loi12].

Theorem 2.5 (Sphere Packing and Gilbert–Varshamov Bound in Rank Metric [GY06]).
Let $\mathsf{A}^R_{q^m}(n, d)$ denote the maximum cardinality of an $(n, M, d)_R$ block code over $\mathbb{F}_{q^m}$ of length n and minimum rank distance d and let $\tau_0 = \lfloor (d-1)/2 \rfloor$. Then,

$$\frac{q^{mn}}{|\mathcal{B}^{(d-1)}_R(\mathbf{0})|} \leq \mathsf{A}^R_{q^m}(n, d) \leq \frac{q^{mn}}{|\mathcal{B}^{(\tau_0)}_R(\mathbf{0})|}. \tag{2.24}$$

The LHS of (2.24) is the Gilbert–Varshamov bound in rank metric and the RHS of (2.24) is the sphere packing bound in rank metric. The rank-metric Gilbert–Varshamov bound from [Loi08, Proposition 3] is slightly different from the one stated in [GY06] and in Theorem 2.5. Note moreover that $|\mathcal{B}^{(\tau_0)}_R(\mathbf{0})|$ and $|\mathcal{B}^{(d-1)}_R(\mathbf{0})|$ are independent of their centers.

A code is called *perfect* in rank metric if it fulfills the RHS of (2.24) with equality. For a perfect code, the balls of radius $\tau_0 = \lfloor (d-1)/2 \rfloor$ around all codewords cover the whole space. However, in contrast to

Hamming metric, there are no perfect codes in rank metric [Loi08, Proposition 2].

The *Singleton bound* in rank metric is given in the following theorem.

Theorem 2.6 (Singleton Bound in Rank Metric [Del78, Theorem 5.4]).
Let C be an $(n, M, d)_R$ code over $\mathbb{F}_{q^m}$ of length n, cardinality M and minimum rank distance d. The cardinality M of C is restricted by:

$$M \leq q^{\min\{n(m-d+1),\, m(n-d+1)\}} = q^{\max\{n,m\}(\min\{n,m\}-d+1)}. \tag{2.25}$$

If the cardinality of a code fulfills (2.25) with equality, the code is called *maximum rank distance* (MRD) code. We denote an MRD (not necessarily linear) code over $\mathbb{F}_{q^m}$ of length n, cardinality $M = q^{\max\{n,m\}(\min\{n,m\}-d+1)}$ and minimum rank distance d by $\mathsf{MRD}(n, M)$.

For linear codes of length $n \leq m$ and dimension k, Theorem 2.6 implies that $d \leq n - k + 1$, see also [Gab85, Corollary, p. 2]. A *linear* MRD code over $\mathbb{F}_{q^m}$ of length $n \leq m$, dimension k and minimum rank distance $d = n - k + 1$ is therefore denoted by $\mathsf{MRD}[n, k]$ and has cardinality $M = q^{mk}$. If $n > m$, we simply transpose all matrices and apply the previous considerations. The complete (rank) weight distribution of MRD codes was derived in [Del78, Theorem 5.6] and [Gab85, Section 3].

A special class of rank-metric codes are q*-cyclic* codes, which can be seen as the analogs to cyclic codes in Hamming metric.

Definition 2.15 (q-Cyclic Code).
Let C be an $(n, M, d)_R$ code over $\mathbb{F}_{q^m}$ of length n, cardinality M and minimum rank distance d. Then, this code is called q-cyclic if

$$(c_{n-j}^{[j]}\ c_{n-j+1}^{[j]}\ \dots\ c_0^{[j]}\ c_1^{[j]}\ \dots\ c_{n-j-1}^{[j]}) \in \mathsf{C},$$

for any integer j and any codeword $(c_0\ c_1\ \dots\ c_{n-1}) \in \mathsf{C}$.

As we will see later, q-cyclic Gabidulin codes are a subclass of Gabidulin codes.

In order to introduce the notation, we also mention shortly *constant-rank codes.* A constant-rank code is a rank-metric code, where all codewords have the same rank. Such a constant-rank code over $\mathbb{F}_{q^m}$ of length n, minimum rank distance d, cardinality M and rank r is denoted by $\mathsf{CR}_{q^m}(n, M, d, r)$.

2.3.2 Gabidulin Codes

Gabidulin codes [Del78, Gab85, Rot91] are a special class of linear MRD codes and are often considered as the analogs of Reed–Solomon codes in rank metric. They are the main class of block codes in rank metric considered in this thesis.

In the following, we define Gabidulin codes as evaluation codes of degree-restricted linearized polynomials, prove that they are MRD codes and give their generator and parity-check matrices.

Definition 2.16 (Linear Gabidulin Code).
A linear Gabidulin code $\mathsf{Gab}[n, k]$ over $\mathbb{F}_{q^m}$ of length $n \leq m$ and dimension $k \leq n$ is the set of all words, which are the evaluation of a q-degree-restricted linearized polynomial $f(x) \in \mathbb{L}_{q^m}[x]$:

$$\mathsf{Gab}[n, k] \overset{\text{def}}{=} \Big\{ (f(g_0)\ f(g_1)\ \dots\ f(g_{n-1})) = f(\mathbf{g}) : \deg_q f(x) < k \Big\},$$

where the fixed elements $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$ are linearly independent over $\mathbb{F}_q$.

Alternatively, we can define the codewords of $\mathsf{Gab}[n,k]$ as the inverse q-transform (see Theorem 2.3) of the evaluation polynomials $f(x)$ of q-degree less than k. In order to do so, we need a normal basis[5] $\mathcal{B}_N^\perp = \{\beta^{\perp[0]}, \beta^{\perp[1]}, \dots, \beta^{\perp[n-1]}\}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Recall from Lemma 2.3 that such a normal basis exists in $\mathbb{F}_{q^m}$ if n divides m and clearly for $n = m$. The coefficients of the codewords can then be given by the inverse q-transform of $f(x)$ as in Theorem 2.3:

$$c_i = f\big(\beta^{\perp[i]}\big) = \sum_{j=0}^{n-1} f_j \beta^{\perp[j+i]}, \qquad \forall i \in [0, n-1]. \tag{2.26}$$

Definition 2.16 and Equation (2.26) agree if we choose $g_i = \beta^{\perp[i]}$. However, the calculation of (2.26) is only possible if there is a normal basis in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, which imposes a restriction on the length of the code. On the other hand, Definition 2.16—and also the definitions based on generator/parity-check matrix, see (2.27) and (2.28)—is valid for any $n \leq m$.

Clearly, a more general definition of the (inverse) q-transform by using an *arbitrary* basis and its dual is the same as the evaluation/interpolation of a linearized polynomial. However, we use the name "q-transform" only together with a normal basis in order to indicate that this transform can be done with low complexity (see Subection 3.1.1).

Theorem 2.7 (Minimum Rank Distance of a Gabidulin Code).
The minimum rank distance of a $\mathsf{Gab}[n,k]$ *Gabidulin code over* $\mathbb{F}_{q^m}$ *with* $n \leq m$ *is* $d = n - k + 1$.

Proof. The evaluation polynomials $f(x)$ have q-degree less than k and therefore the dimension of their root spaces over $\mathbb{F}_{q^m}$ is at most $k-1$.
Let $\mathbf{C} = \mathrm{ext}_\beta(\mathbf{c}) \in \mathbb{F}_q^{m\times n}$ denote the representation of $\mathbf{c} \in \mathsf{Gab}[n,k]$ as in Definition 2.1. Since the evaluation of a linearized polynomial at a basis is an $\mathbb{F}_q$-linear map, it follows with Lemma 2.11 that the dimension of the right kernel of $\mathbf{C} \in \mathbb{F}_q^{m\times n}$ is equal to the dimension of the root space of the corresponding evaluation polynomial $f(x)$. Therefore,

$$\dim\ker(\mathbf{c}) \leq k-1, \quad \forall \mathbf{c} \in \mathsf{Gab}[n,k].$$

There is a codeword $\mathbf{c}$ in $\mathsf{Gab}[n,k]$ of rank d and due to the rank nullity theorem, for this codeword $\dim\ker(\mathbf{c}) = n - d$ holds. Hence,

$$\dim\ker(\mathbf{c}) = n - d \leq k - 1 \quad \iff \quad d \geq n - k + 1.$$

However, the Singleton-like bound (2.25) implies that $d \leq n-k+1$ and hence, $d = n-k+1$. ∎

Thus, Gabidulin codes are MRD codes.

Based on Definition 2.16, we can give the generator matrix of a Gabidulin code using the elements $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$, which are linearly independent over $\mathbb{F}_q$.

$$\mathbf{G} = \mathrm{qvan}_k((g_0\ g_1\ \dots\ g_{n-1})) = \begin{pmatrix} g_0^{[0]} & g_1^{[0]} & \dots & g_{n-1}^{[0]} \\ g_0^{[1]} & g_1^{[1]} & \dots & g_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{[k-1]} & g_1^{[k-1]} & \dots & g_{n-1}^{[k-1]} \end{pmatrix}, \tag{2.27}$$

since the evaluation of a linearized polynomial of q-degree less than k is the same as multiplying its coefficients with the aforementioned q-Vandermonde matrix.

[5] For consistency with Theorem 2.3, we denote this normal basis by $\mathcal{B}_N^\perp$ as dual basis to $\mathcal{B}_N$.

Lemma 2.14 (Parity-Check Matrix of Gabidulin Code).
Let $\mathbf{G}$ be a generator matrix of a $\mathsf{Gab}[n,k]$ *code as in* (2.27)*, where $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$ are linearly independent over $\mathbb{F}_q$. Let $h_0, h_1, \dots, h_{n-1}$ be a non-zero solution for the following $n-1$ linear equations:*

$$\sum_{i=0}^{n-1} g_i^{[j]} h_i = 0, \quad \forall j \in [-n+k+1, k-1]. \tag{2.28}$$

Then, the $(n-k) \times n$ matrix

$$\mathbf{H} = \mathrm{qvan}_{n-k}((h_0\ h_1\ \dots\ h_{n-1})) = \begin{pmatrix} h_0^{[0]} & h_1^{[0]} & \dots & h_{n-1}^{[0]} \\ h_0^{[1]} & h_1^{[1]} & \dots & h_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{[n-k-1]} & h_1^{[n-k-1]} & \dots & h_{n-1}^{[n-k-1]} \end{pmatrix},$$

is a parity-check matrix of the $\mathsf{Gab}[n,k]$ *code.*

Proof. Since the dual of a $\mathsf{Gab}[n,k]$ code is a $\mathsf{Gab}[n,n-k]$ code [Gab85, Theorem 3], we have to prove that $\mathbf{H}$ is a generator matrix of this dual code, i.e., $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$ has to hold, which is equivalent to the following $n-1$ linear equations:

$$\begin{aligned} & \sum_{i=0}^{n-1} g_i^{[l]} h_i^{[j]} = 0, \qquad \forall l \in [0, k-1], j \in [0, n-k-1], \\ \iff \quad & \sum_{i=0}^{n-1} g_i^{[j]} h_i = 0, \qquad \forall j \in [-n+k+1, k-1]. \end{aligned}$$

Therefore, if $h_0, h_1, \dots, h_{n-1}$ are linearly independent over $\mathbb{F}_q$, $\mathbf{H}$ is a generator matrix of the dual code $\mathsf{Gab}[n,n-k]$. To prove this, denote $\widetilde{\mathbf{g}} = (g_0^{[-n+k+1]}\ g_1^{[-n+k+1]}\ \dots\ g_{n-1}^{[-n+k+1]})$. Then, (2.28) is equivalent to

$$\mathrm{qvan}_{n-1}(\widetilde{\mathbf{g}}) \cdot (h_0\ h_1\ \dots\ h_{n-1})^T = \mathbf{0}. \tag{2.29}$$

The matrix $\mathrm{qvan}_{n-1}(\widetilde{\mathbf{g}})$ is a parity-check matrix of a $\mathsf{Gab}[n,1]$ code, since $g_0^{[-n+k+1]}, g_1^{[-n+k+1]}, \dots, g_{n-1}^{[-n+k+1]} \in \mathbb{F}_{q^m}$ are linearly independent over $\mathbb{F}_q$. Hence, the vector $(h_0\ h_1\ \dots\ h_{n-1})$ is a codeword of the $\mathsf{Gab}[n,1]$ code. This $\mathsf{Gab}[n,1]$ code has minimum rank distance $d = n-1+1 = n$ and therefore $\mathrm{rk}((h_0\ h_1\ \dots\ h_{n-1})) = n$. Thus, $\mathbf{H}$ is a generator matrix of the dual $\mathsf{Gab}[n,n-k]$ code and therefore a parity-check matrix of the $\mathsf{Gab}[n,k]$ code. ∎

The following lemma investigates q-cyclic Gabidulin codes.

Lemma 2.15 (q-cyclic Gabidulin Code).
Let $\mathbf{g} = (g_0\ g_1\ \dots\ g_{n-1}) = (\beta^{\perp[0]}\ \beta^{\perp[1]}\ \dots\ \beta^{\perp[n-1]})$ be an ordered normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ and let $\mathsf{Gab}[n,k]$ *be a Gabidulin code over $\mathbb{F}_{q^m}$ as in Definition 2.16.*
Then, $\mathsf{Gab}[n,k]$ *is q-cyclic as in Definition 2.15.*

Proof. We have to show that for any integer j and any codeword $\mathbf{c} = (c_0\ c_1\ \dots\ c_{n-1}) \in \mathsf{Gab}[n,k]$, the q-cyclic shift

$$\widetilde{\mathbf{c}} = (\widetilde{c}_0\ \widetilde{c}_1\ \dots\ \widetilde{c}_{n-1}) \overset{\text{def}}{=} (c_{n-j}^{[j]}\ c_{n-j+1}^{[j]}\ \dots\ c_0^{[j]}\ c_1^{[j]}\ \dots\ c_{n-j-1}^{[j]})$$

is also a codeword of $\mathsf{Gab}[n,k]$. The coefficients of $\widetilde{\mathbf{c}}$ (indices calculated modulo n) are given by:

$$\begin{aligned}\widetilde{c}_i = c_{i-j}^{[j]} = \Big(f(\beta^{\perp[i-j]})\Big)^{[j]} &= x^{[j]} \circ f(x)\Big|_{x=\beta^{\perp[i-j]}}\\ &= x^{[j]} \circ f(x) \circ x^{[-j]}\Big|_{x=\beta^{\perp[i]}}, \quad \forall i \in [0, n-1].\end{aligned}$$

Therefore, for $f(x) = \sum_{i=0}^{k-1} f_i x^{[i]}$, we obtain:

$$\widetilde{f}(x) \overset{\text{def}}{=} x^{[j]} \circ f(x) \circ x^{[-j]} = f_0^{[j]} x^{[0]} + f_1^{[j]} x^{[1]} + \dots + f_{k-1}^{[j]} x^{[k-1]},$$

and $\deg_q \widetilde{f}(x) = \deg_q f(x) < k$. Thus,

$$\widetilde{\mathbf{c}} = (\widetilde{c}_0\ \widetilde{c}_1\ \dots\ \widetilde{c}_{n-1}) = \widetilde{f}(\mathbf{g}) = \Big(\widetilde{f}(\beta^{\perp[0]})\ \widetilde{f}(\beta^{\perp[1]})\ \dots\ \widetilde{f}(\beta^{\perp[n-1]})\Big)$$

is a codeword of $\mathsf{Gab}[n,k]$. ∎

Lemma 2.3 provides directly the following corollary.

Corollary 2.2 (Existence of *q*-cyclic Gabidulin Code).
A q-cyclic Gabidulin code $\mathsf{Gab}[n,k]$ *of length $n \leq m$ and dimension $k \leq n$ over $\mathbb{F}_{q^m}$ as in Lemma 2.15 exists for any n dividing m.*

Thus, when we recall (2.26), we see that defining Gabidulin codes by the (inverse) q-transform yields q-cyclic Gabidulin codes. This property allows to reduce the complexity of encoding and decoding (see Section 3.2.4).

Lemma 2.16 (Parity Check Matrix of *q*-cyclic Gabidulin Code).
Let $\mathsf{Gab}[n,k]$ *be a q-cyclic Gabidulin code over $\mathbb{F}_{q^m}$ as in Lemma 2.15, where $\mathbf{g} = (\beta^{\perp[0]}\ \beta^{\perp[1]}\ \dots\ \beta^{\perp[n-1]})$ is an ordered normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, $n \mid m$, and $(\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[n-1]})$ is a dual normal basis to $\mathbf{g}$.*

Then, for $\mathbf{h} \overset{\text{def}}{=} (\beta^{[k]}\ \beta^{[k+1]}\ \dots\ \beta^{[k+n-1]})$, the $(n-k) \times n$ matrix

$$\mathbf{H} = \mathrm{qvan}_{n-k}(\mathbf{h}) = \begin{pmatrix} \beta^{[k]} & \beta^{[k+1]} & \dots & \beta^{[k-1]}\\ \beta^{[k+1]} & \beta^{[k+2]} & \dots & \beta^{[k]}\\ \vdots & \vdots & \ddots & \vdots\\ \beta^{[n-1]} & \beta^{[0]} & \dots & \beta^{[n-2]}\end{pmatrix},$$

is a parity-check matrix of the $\mathsf{Gab}[n,k]$ *code.*

Proof. The proof follows from the proof of Theorem 2.3: Let $\mathbf{H}$ consist of the last $n-k$ rows of $\mathbf{B}$ (defined as in (2.23) with $s=n$) and let $\mathbf{G}^T$ be the $n \times k$ submatrix of $\mathbf{B}^\perp$ (defined as in (2.23) with $s=n$), consisting of the first k columns of $\mathbf{B}^\perp$. Then, $\mathbf{G}$ is exactly the generator matrix from (2.27) and $\mathbf{H} \cdot \mathbf{G}^T = \mathbf{0}$ as shown in the proof of Theorem 2.3.
Since $\mathbf{h} = (\beta^{[k]}\ \beta^{[k+1]}\ \dots\ \beta^{[k+n-1]})$ consists of n linearly independent elements, $\mathbf{H}$ is a parity-check matrix of this $\mathsf{Gab}[n,k]$ code. ∎

2.3.3 Interleaved Gabidulin Codes

Interleaved Gabidulin codes can be seen as s horizontally or vertically arranged codewords of (not necessarily different) Gabidulin codes. They are the analog of *interleaved Reed–Solomon* codes in Hamming metric, see [KL97, KL98, BKY07, Kra03, BMS04, JTH04, SSB09, WZB12].

Vertically interleaved Gabidulin codes were introduced by Loidreau and Overbeck in [LO06, Ove07] and rediscovered by Silva, Kschischang and Kötter [SKK08, Sil09] as the *Cartesian product* of s transposed codewords of Gabidulin codes. Later, Sidorenko and Bossert introduced *horizontally* interleaved Gabidulin codes [SB10, SJB11].

We consider *vertically* interleaved Gabidulin codes in this thesis. However, if an application requires matrices with the dimensions of a horizontally interleaved Gabidulin code, we can simply transpose the codewords of the vertically interleaved code. Known decoding approaches and a new interpolation-based decoding approach for interleaved Gabidulin codes are considered in Chapter 4.

Definition 2.17 (Interleaved Gabidulin Code).
Let $\mathbf{g} = (g_0 \; g_1 \; \dots \; g_{n-1})$*, where* $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$ *are linearly independent over* $\mathbb{F}_q$*. A linear (vertically) interleaved Gabidulin code* $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ *over* $\mathbb{F}_{q^m}$ *of length* $n \leq m$*, elementary dimensions* $k^{(1)}, \dots, k^{(s)} \leq n$ *and interleaving order* s *is defined by*

$$\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}] \overset{\text{def}}{=} \left\{ \begin{pmatrix} \mathbf{c}^{(1)} \\ \mathbf{c}^{(2)} \\ \vdots \\ \mathbf{c}^{(s)} \end{pmatrix} = \begin{pmatrix} f^{(1)}(\mathbf{g}) \\ f^{(2)}(\mathbf{g}) \\ \vdots \\ f^{(s)}(\mathbf{g}) \end{pmatrix} : \deg_q f^{(i)}(x) < k^{(i)} \leq n, \forall i \in [1, s] \right\}.$$

For $s = 1$, this defines a usual Gabidulin code $\mathsf{Gab}[n, k] = \mathsf{IGab}[1; n, k]$. Using the map ext_{β} from Definition 2.1, we can either represent the codewords of the interleaved code as a vector in $\mathbb{F}_{q^{ms}}^{n}$, a matrix in $\mathbb{F}_{q^m}^{s \times n}$ or over the ground field as a matrix in $\mathbb{F}_{q}^{sm \times n}$. Throughout this thesis, we will use any representation, whatever is more convenient and indicate which one is meant if there are ambiguities. This is illustrated in Figure 2.2.

Analog to interleaved Reed–Solomon codes, we call an interleaved Gabidulin code *homogeneous* if all elementary dimensions are equal, i.e., if $k^{(i)} = k$, $\forall i \in [1, s]$.

Lemma 2.17 (Homogeneous Interleaved Gabidulin Codes are MRD).
Let $\mathsf{IGab}[s; n, k, \dots, k]$ *be a linear interleaved Gabidulin code over* $\mathbb{F}_{q^m}$ *as in Definition 2.17 with* $k^{(i)} = k$, $\forall i \in [1, s]$*. Its minimum distance is* $d = n - k + 1$ *and it is an MRD code.*

Proof. On the one hand, any non-zero codeword of $\mathsf{IGab}[s; n, k, \dots, k]$ contains at least one non-zero codeword of a Gabidulin code $\mathsf{Gab}[n, k]$ and therefore, $d \geq n - k + 1$. On the other hand,

$$\begin{pmatrix} \mathbf{c}^{(1)} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \in \mathsf{IGab}[s; n, k, \dots, k].$$

We can choose $\mathbf{c}^{(1)} \in \mathsf{Gab}[n, k]$ such that $\text{rk}(\mathbf{c}^{(1)}) = n - k + 1$ and therefore the minimum distance of $\mathsf{IGab}[s; n, k, \dots, k]$ is exactly $d = n - k + 1$ and it is an MRD code. ∎

For arbitrary elementary dimensions, we can use the same reasoning and the minimum distance of $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ is $d = n - \max_i\{k^{(i)}\} + 1$. Notice that this is no MRD code.

Known decoding approaches and a new interpolation-based decoding approach for interleaved Gabidulin codes is described in Chapter 4.

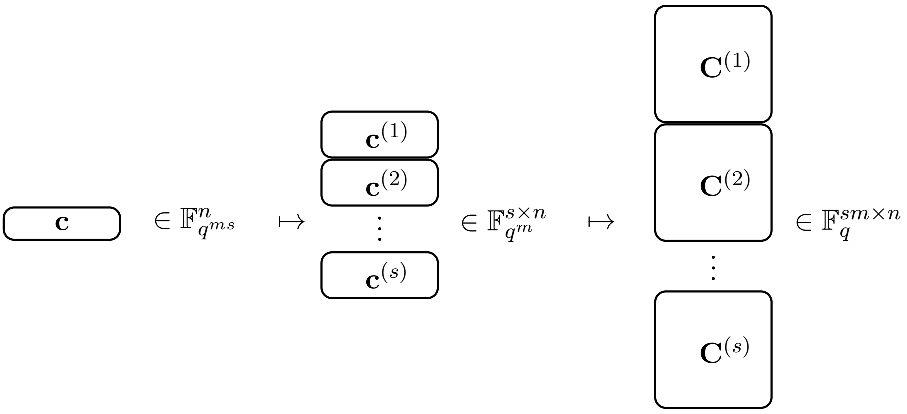


Figure 2.2. Representations of codewords of an interleaved Gabidulin code of length n and interleaving order s.

2.3.4 Lifted Gabidulin Codes

This subsection will give a brief definition of a special class of *constant-dimension codes*, constructed by *lifted Gabidulin codes.* These constant-dimension codes are used in Chapter 5 to establish bounds on list decoding block rank-metric codes. Constant-dimension codes and, more general, codes in the projective space were thoroughly investigated e.g. in [WXS03, KK08, XF09, ES09, Ska10, EV11, Sil11, BVP13, ES13].

As defined in Subsection 2.1.1, let $\mathbb{F}_q^n$ denote the vector space of dimension n over the finite field $\mathbb{F}_q$, $\mathcal{P}_q(n)$ the projective space and $\mathcal{G}_q(n,r)$ the Grassmannian of dimension r.

A distance measure for codes in the projective space is the so-called *subspace distance.* For two subspaces $\mathcal{U},\mathcal{V}$ in $\mathcal{P}_q(n)$, we denote by $\mathcal{U}+\mathcal{V}$ the smallest subspace containing the union of $\mathcal{U}$ and $\mathcal{V}$. The *subspace distance* between $\mathcal{U},\mathcal{V}$ in $\mathcal{P}_q(n)$ is defined by

$$\begin{aligned} d_s(\mathcal{U},\mathcal{V}) &= \dim(\mathcal{U}+\mathcal{V}) - \dim(\mathcal{U}\cap\mathcal{V}) \\ &= 2\dim(\mathcal{U}+\mathcal{V}) - \dim(\mathcal{U}) - \dim(\mathcal{V}). \end{aligned} \tag{2.30}$$

It can be shown that the subspace distance is indeed a metric (see e.g., [KK08, Lemma 1]) and it is connected to the so-called *injection distance* d_i as follows (see [SK09b, Equation (28)]):

$$d_i(\mathcal{U},\mathcal{V}) = \frac{1}{2}\, d_s(\mathcal{U},\mathcal{V}) + \frac{1}{2}\big|\dim(\mathcal{U}) - \dim(\mathcal{V})\big|. \tag{2.31}$$

Throughout this thesis, we will use the subspace distance as a distance measure between subspaces.

A code in the projective space (also called subspace code) is a non-empty collection of subspaces of $\mathcal{P}_q(n)$, i.e., each codeword is a subspace. A *constant-dimension code* (also called Grassmannian code) is a special subspace code, where each codeword has the same dimension. Let $\mathsf{CD}_q(n, M, d_s, r)$ denote a constant-dimension code in $\mathcal{G}_q(n,r)$ with cardinality M and minimum subspace distance d_s. This code is therefore a subset of the Grassmannian $\mathcal{G}_q(n,r)$.

For constant-dimension codes, (2.31) shows that the minimum injection distance is half the minimum subspace distance. Further details can be found e.g., in [KSK09], which provides a survey (up to the year 2009) on codes in the projective space and constant-dimension codes.

The *lifting* of a block code defines a constant-dimension code as follows.

Definition 2.18 (Lifting of a Matrix or a Code).
Consider the mapping

$$\begin{aligned} \text{lift} : \quad \mathbb{F}_q^{r \times (n-r)} &\to \mathcal{G}_q(n,r) \\ \mathbf{X} &\mapsto \mathcal{R}_q\left([\mathbf{I}_r \; \mathbf{X}]\right), \end{aligned}$$

where $\mathbf{I}_r$ denotes the $r \times r$ identity matrix. The subspace $\text{lift}(\mathbf{X}) = \mathcal{R}_q\left([\mathbf{I}_r \; \mathbf{X}]\right)$ *is called **lifting** of the matrix $\mathbf{X}$. If we apply this map on all codewords (in matrix representation) of a block code* C*, then the constant-dimension code* $\text{lift}(\mathsf{C})$ *is called lifting of* C.

The following lemma shows the properties of a lifted linear MRD code.

Lemma 2.18 (Lifted MRD Code).
Let a linear $\mathsf{MRD}[r,k]$ *code* C *over* $\mathbb{F}_{q^{n-r}}$ *of length* $r \leq n-r$*, minimum rank distance* $d = r-k+1$ *and cardinality* $M_R = q^{(n-r)k}$ *be given.*

Then, the lifting of the transposed codewords, i.e.

$$\text{lift}(\mathsf{C}^T) \overset{\text{def}}{=} \left\{ \text{lift}(\mathbf{C}^T) = \mathcal{R}_q\left([\mathbf{I}_r \; \mathbf{C}^T]\right) : \mathbf{C} \in \mathsf{C} \right\}$$

is a $\mathsf{CD}_q(n, M_s, d_s, r)$ *constant-dimension code of cardinality* $M_s = M_R = q^{(n-r)k}$*, minimum subspace distance* $d_s = 2d$ *and lies in the Grassmannian* $\mathcal{G}_q(n,r)$.

Proof. Let $\mathbf{C}_i \in \mathbb{F}_q^{(n-r)\times r}$, $\forall i \in [1, M_R]$, denote the codewords of C in matrix representation. The dimension of each subspace $\text{lift}(\mathbf{C}_i^T)$ is r since $\text{rk}([\mathbf{I}_r \; \mathbf{C}_i^T]) = r$, for all $i \in [1, M_R]$. The cardinality of this constant-dimension code is the same as the cardinality of the MRD code, which is $M_R = q^{(n-r)k}$. The subspace distance of the constant-dimension code is two times the rank distance of the MRD code (see [SKK08, Proposition 4]) since for any two $\mathbf{C}_1, \mathbf{C}_2 \in \mathsf{C}$ with (2.30):

$$\begin{aligned} d_s(\text{lift}(\mathbf{C}_1^T), \text{lift}(\mathbf{C}_2^T)) &= 2\dim\left(\text{lift}(\mathbf{C}_1^T) + \text{lift}(\mathbf{C}_2^T)\right) - \dim\left(\text{lift}(\mathbf{C}_1^T)\right) - \dim\left(\text{lift}(\mathbf{C}_2^T)\right) \\ &= 2\,\text{rk}\begin{pmatrix} \mathbf{I}_r & \mathbf{C}_1^T \\ \mathbf{I}_r & \mathbf{C}_2^T \end{pmatrix} - 2r = 2\,\text{rk}\begin{pmatrix} \mathbf{I}_r & \mathbf{C}_1^T \\ \mathbf{0} & \mathbf{C}_2^T - \mathbf{C}_1^T \end{pmatrix} - 2r \\ &= 2\left[\text{rk}(\mathbf{I}_r) + \text{rk}(\mathbf{C}_2^T - \mathbf{C}_1^T)\right] - 2r = 2\,\text{rk}(\mathbf{C}_2 - \mathbf{C}_1) = 2d_R(\mathbf{C}_1, \mathbf{C}_2). \end{aligned}$$

∎

CHAPTER 3

Decoding Approaches for Gabidulin Codes

Decoding principles for Gabidulin codes mostly rely on the similarities to Reed–Solomon codes and are therefore equivalents of well-known decoding algorithms in Hamming metric. The first *bounded minimum distance* (BMD) decoding algorithm by Gabidulin [Gab85] is based on solving a key equation with the *linearized extended Euclidean algorithm* (LEEA) and can be seen as an equivalent of the Sugiyama–Kasahara–Hirasawa–Namekawa decoding algorithm for Reed–Solomon and Goppa codes [SKHN75]. The oldest decoding principle for Reed–Solomon codes is the Peterson–Gorenstein–Zierler approach [Pet60, GZ61]. A similar algorithm for Gabidulin codes was introduced in 1991 by Roth [Rot91] and independently also by Gabidulin in 1992 [Gab92]. In this approach, the solution to the key equation is found by solving a linear system of equations based on the syndrome coefficients.

Due to its efficient use of shift-register synthesis, the Berlekamp–Massey algorithm [Ber68, Mas69] is probably the most established decoding algorithm for Reed–Solomon and cyclic codes. A linearized equivalent of it was given by Paramonov and Tretjakov [PT91] and Richter and Plass [RP04a, RP04b]. The proof of this algorithm was given later by Sidorenko, Richter and Bossert in [SRB11].

While the previously mentioned approaches solve a key equation and return an error-span polynomial, interpolation-based decoders directly output the evaluation polynomial of a codeword. A Welch–Berlekamp-like decoder [WB86] for interpolation-based BMD decoding of Gabidulin codes was presented by Loidreau in 2006 [Loi06, Loi07].

Apart from increasing the decoding radius of Gabidulin codes (which is considered in some sense in Chapters 4 and 5), the research interest lies in accelerating BMD decoding algorithms [SK09a, HS10, SB12, WAS13, SWC12] and in error-erasure decoding [SKK08, Sil09, GP08, LSC13].

This chapter deals with efficient decoding of Gabidulin codes. First, in Section 3.1, we present new methods to accomplish efficient calculations with linearized polynomials. Second, Section 3.2 briefly explains known decoding approaches for Gabidulin codes and presents a new BMD decoding algorithm, which is based on solving a transformed key equation with the LEEA and directly outputs the evaluation polynomial of the estimated codeword. Finally, we establish how this algorithm can be accelerated and generalize it for error-erasure decoding.

The results of Subsections 3.1.2 and 3.1.4 were partly published in [WSB10] and the results from Subsections 3.1.3, 3.2.2 and 3.2.4 in [WAS11, WAS13].

3.1 Fast Algorithms for Linearized Polynomials

Efficient implementations of operations with linearized polynomials are essential in order to develop fast decoding algorithms for codes in rank metric, in particular for (interleaved) Gabidulin codes. Throughout this thesis, the computational complexity is considered as operations (multiplications and additions) in $\mathbb{F}_{q^m}$ or in $\mathbb{F}_q$, where the corresponding field will be indicated.

Subsection 3.1.1 explains the complexity of known approaches and states some problems, which are treated in the subsequent subsections. In Subsections 3.1.2 and 3.1.3, we present fast algorithms for calculating the linearized composition and in Subsection 3.1.4, we give a fast equivalent of the LEEA based on the Divide & Conquer strategy.

3.1.1 Complexity of Known Approaches and Overview of New Approaches

A summary of efficient calculations in finite fields using normal bases is given in Table 3.1. Further, Table 3.2 provides an overview of the complexity of standard implementations for operations with linearized polynomials. As a last part of this subsection, we state some problems concerning efficient operations of linearized polynomials, which are picked up in the following subsections.

Operations with Normal Bases

Throughout this thesis, we assume that we can switch between the representation of a vector in $\mathbb{F}_{q^m}$ and a matrix in $\mathbb{F}_q$ (i.e., the map from Definition 2.1) without any cost. Recall Section 2.1.2 about normal bases, where we showed in (2.3) that a q-power of an element $a \in \mathbb{F}_{q^m}$ corresponds to a cyclic shift of $\text{ext}_{\boldsymbol{\beta}}(a) \in \mathbb{F}_q^{m\times 1}$, when a normal basis is used. Hence, the complexity of q-powers is negligible whenever we consider a normal basis representation.

The addition $a + b$ of $a, b \in \mathbb{F}_{q^m}$ can be implemented by $\text{ext}_{\boldsymbol{\beta}}(a) + \text{ext}_{\boldsymbol{\beta}}(b) \in \mathbb{F}_q^{m\times 1}$ in $\mathcal{O}(m)$ operations over $\mathbb{F}_q$.

The product $a \cdot b$ of any two elements $a, b \in \mathbb{F}_{q^m}$ can be calculated as in (2.4), showing that $\text{ext}_{\boldsymbol{\beta}}(a \cdot b) \in \mathbb{F}_q^{m\times 1}$ can be obtained by m multiplications of a vector in $\mathbb{F}_q^{m\times 1}$ with the multiplication table $\mathbf{T}_m$ and summing these m resulting vectors up (times a scalar in $\mathbb{F}_q$). Since $\mathbf{T}_m$ has $\text{comp}(\mathbf{T}_m)$ non-zero entries, the multiplication of $\mathbf{T}_m$ with a vector costs $\text{comp}(\mathbf{T}_m)$ operations over $\mathbb{F}_q$ and the whole calculation of $\text{ext}_{\boldsymbol{\beta}}(a \cdot b)$ costs $\mathcal{O}(m \,\text{comp}(\mathbf{T}_m)) \geq \mathcal{O}(m^2)$ operations in the ground field $\mathbb{F}_q$.

If one of the two multiplied elements is a basis element, e.g. $b = \beta^{[j]}$, and we want to compute $a \cdot \beta^{[j]}$, then (2.5) shows that the summation step from (2.4) disappears and the complexity is reduced to $\mathcal{O}(\text{comp}(\mathbf{T}_m))$ operations over $\mathbb{F}_q$.

For the calculation of the (inverse) q-transform of a linearized polynomial $a(x) \in \mathbb{L}_{q^m}[x]$ of q-degree

Table 3.1. Complexity of operations in finite fields using normal bases.

Operation	Notation	Method	Complexity
q-power of $a \in \mathbb{F}_{q^m}$	$a^{[i]}$	Eq. (2.3), p. 8	negligible
Addition of $a, b \in \mathbb{F}_{q^m}$	$a + b$	$\text{ext}_{\boldsymbol{\beta}}(a) + \text{ext}_{\boldsymbol{\beta}}(b)$	$\mathcal{O}(m)$ in $\mathbb{F}_q$
Multiplication of $a, b \in \mathbb{F}_{q^m}$	$a \cdot b$	Eq. (2.4), p. 9	$\mathcal{O}(m \,\text{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$
Multiplication of $a, \beta^{[j]} \in \mathbb{F}_{q^m}$, where $\beta^{[j]} \in \mathcal{B}_N$	$a \cdot \beta^{[j]}$	Eq. (2.5), p. 9	$\mathcal{O}(\text{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$
q-transform of $a(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q a(x) = d_a < s$, $s \mid m$	$\widehat{a}(x)$	Eq. (2.21), p. 22	$\mathcal{O}(d_a d_{\widehat{a}} \,\text{comp}(\mathbf{T}_m))$ $\leq \mathcal{O}(s^2 \,\text{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$
Inv. q-transform of $\widehat{a}(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q \widehat{a}(x) = d_{\widehat{a}} < s$, $s \mid m$	$a(x)$	analog to Eq. (2.21), p. 22	$\mathcal{O}(d_a d_{\widehat{a}} \,\text{comp}(\mathbf{T}_m))$ $\leq \mathcal{O}(s^2 \,\text{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$

$d_a < s$ (see Definition 2.12 and Theorem 2.3) recall (2.21), which shows how to obtain the coefficients of the transformed polynomial $\widehat{a}(x) = \sum_{j=0}^{d_{\widehat{a}}} \widehat{a}_j x^{[i]}$, represented as vectors over $\mathbb{F}_q$. For each coefficient $\widehat{a}_j$, we calculate $d_a + 1 \leq s$ times the product of an element $a_i \in \mathbb{F}_{q^m}$ and a basis element $\beta^{[i+j]}$ as in (2.5). Hence, the calculation $\widehat{a}_j$ for $j \in [0, d_{\widehat{a}}]$ requires $\mathcal{O}(d_a d_{\widehat{a}} \operatorname{comp}(\mathbf{T}_m)) \leq \mathcal{O}(s^2 \operatorname{comp}(\mathbf{T}_m))$ operations over $\mathbb{F}_q$. The calculation of the *inverse* q-transform can be done analogously to (2.21). To our knowledge, this efficient calculation of the (inverse) q-transform was first observed in [SK09a].

Table 3.1 summarizes the complexity of the mentioned operations using normal bases.

Operations with Linearized Polynomials

Let $\mathcal{M}(d_a, d_b)$ denote the complexity of calculating the composition of two linearized polynomials, i.e., $a(b(x)) \in \mathbb{L}_{q^m}[x]$ with $d_a = \deg_q a(x)$ and $d_b = \deg_q b(x)$. The straight-forward calculation of the coefficients of $a(b(x))$ can be done as in (2.12), where the calculation of all coefficients requires

$$\sum_{j=0}^{d_a+d_b} (j+1) = \sum_{j=1}^{d_a+d_b+1} j = \frac{(d_a+d_b+1)(d_a+d_b+2)}{2} \sim \mathcal{O}\big((d_a+d_b)^2\big)$$

additions and multiplications over $\mathbb{F}_{q^m}$, when we assume again that q-powers are negligible.

The complexity of calculating the linearized composition modulo $(x^{[m]} - x)$ as in (2.13) requires therefore at most $\mathcal{O}(m^2)$ operations in $\mathbb{F}_{q^m}$ and is denoted by $\mathcal{M}_m(m) = \mathcal{O}(m^2)$ over $\mathbb{F}_{q^m}$.

The right/left linearized division can be calculated as in Algorithm 2.1 and 2.2, respectively, and its complexity for two linearized polynomials $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ is denoted by $\mathcal{D}(d_a, d_b)$. The standard implementations from Algorithm 2.1 and 2.2, for $d_a \geq d_b$, terminate after at most $d_a - d_b + 1$ iterations. The complexity of each iteration is dominated by the linearized composition in Line 3. Since $q^{(i)}(x)$ has only one non-zero coefficient, Line 3 can be computed with $d_b + 1$ multiplications of elements in $\mathbb{F}_{q^m}$. Hence, the complexity of the linearized division is $\mathcal{D}(d_a, d_b) = \mathcal{O}((d_a - d_b)d_b)$ operations in $\mathbb{F}_{q^m}$.

The standard implementation of the LEEA from Algorithm 2.3 for two input polynomials $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a \geq d_b$ requires $\mathcal{O}(d_a^2)$ operations over $\mathbb{F}_{q^m}$ (see [GY08a]). In detail, this complexity also depends on the stopping degree d_{stop}, but the *order* of the complexity is independent of d_{stop}, and therefore we do not analyze this dependency in detail here.

Table 3.2 shows an overview of these standard implementations for linearized polynomials.

Table 3.2. Complexity of standard implementations for linearized polynomials.

Operation	Notation	Method	Complexity
Linearized composition of $a(x), b(x) \in \mathbb{L}_{q^m}[x]$	$a(b(x))$	Eq. (2.12), p. 18	$\mathcal{M}(d_a, d_b) = \mathcal{O}((d_a+d_b)^2)$ in $\mathbb{F}_{q^m}$
Linearized composition modulo $(x^{[m]} - x)$	$a(b(x)) \bmod (x^{[m]} - x)$	Eq. (2.13), p. 18	$\mathcal{M}_m(m) = \mathcal{O}(m^2)$ in $\mathbb{F}_{q^m}$
Linearized division of $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a \geq d_b$	RightDiv$\big(a(x); b(x)\big)$ LeftDiv$\big(a(x); b(x)\big)$	Algo. 2.1, p. 19 Algo. 2.2, p. 19	$\mathcal{D}(d_a, d_b) = \mathcal{O}((d_a - d_b)d_b)$in $\mathbb{F}_{q^m}$
LEEA of $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a \geq d_b$	RightLEEA$\big(a(x); b(x); d_{stop}\big)$	Algo. 2.3, p. 20	$\mathcal{O}_{EA}(d_a, d_b) = \mathcal{O}(d_a^2)$ in $\mathbb{F}_{q^m}$

Problem Statement and Overview of New Approaches

The following subproblems provide the starting point of the remainder of this section.

Problem 3.1 (Fast Linearized Operations).
Let $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a = \deg_q a(x)$ and $d_b = \deg_q b(x)$ be given.

(a) ***Fast Linearized Composition.***
Let $n = \max\{d_a, d_b\} + 1$. Find a linearized polynomial

$$c(x) = a(x) \circ b(x) = a(b(x)),$$

see also (2.12), *with complexity $\mathcal{M}(d_a, d_b) < \mathcal{O}(n^2)$ operations in $\mathbb{F}_{q^m}$.*

(b) ***Fast Linearized Composition modulo $(x^{[m]} - x)$.***
Let $d_a < m$, $d_b < m$. Find a linearized polynomial

$$c(x) \equiv a(x) \circ b(x) = a(b(x)) \mod (x^{[m]} - x),$$

see also (2.13), *with complexity $\mathcal{M}_m(m) < \mathcal{O}(m^2)$ operations in $\mathbb{F}_{q^m}$.*

(c) ***Fast Linearized Multi-Point Evaluation.***
Let $d_a < m$ and let the s points $b_0, b_1, \dots, b_{s-1} \in \mathbb{F}_{q^m}$, where $s \mid m$, be given. Find the s evaluation values

$$a(b_0), a(b_1), \dots, a(b_{s-1})$$

with complexity less than $\mathcal{O}(m^2)$ operations in $\mathbb{F}_{q^m}$.

(d) ***Fast Linearized Euclidean Algorithm.***
Let $d_a \geq d_b$. Find the output of RightLEEA$\big(a(x); b(x); d_{stop}\big)$, *Algorithm 2.3, for some $d_a > d_{stop} \geq d_a/2$, with complexity $\mathcal{O}_{EA}(d_a, d_b) < \mathcal{O}(d_a^2)$ operations in $\mathbb{F}_{q^m}$.*

As a preview, Table 3.3 provides an overview of our new algorithms which give solutions to the aforementioned problems and are explained and proven in detail in the following subsections.

In order to compare the different algorithms, recall Table 3.1 to compare operations in $\mathbb{F}_{q^m}$ with operations in $\mathbb{F}_q$. An addition of two elements in $\mathbb{F}_{q^m}$ costs $\mathcal{O}(m)$ operations in $\mathbb{F}_q$ and a multiplication of two elements in $\mathbb{F}_{q^m}$ can be realized with $\mathcal{O}(m \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$. If there exists a low-complexity normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, this multiplication can be done with $\mathcal{O}(m^2)$ operations in $\mathbb{F}_q$ and therefore, any operation in $\mathbb{F}_{q^m}$ can be implemented with at most $\mathcal{O}(m^2)$ operations in $\mathbb{F}_q$.

Hence, Algorithm 3.1 for calculating $a(b(x)) \mod (x^{[m]} - x)$ costs at most $\mathcal{O}(m^{1.69})$ operations over $\mathbb{F}_{q^m}$ and can be realized in $\mathcal{O}(m^{3.69})$ operations over $\mathbb{F}_q$.

If we want to compare Algorithm 3.1 and Algorithm 3.3 for calculating the linearized composition, we first have to remark that Algorithm 3.1 can also calculate the linearized composition *without* modulo. However, when we use low-complexity normal bases, Algorithm 3.3 is more efficient for calculating the linearized composition modulo $(x^{[m]} - x)$. Moreover, the complexity improvement is achieved also for small polynomials whereas the complexity of Algorithm 3.1 is only valid for large polynomials since the fast matrix multiplication (by the Coppersmith–Winograd algorithm) is only efficient for large matrices.

Notice that there are also asymptotically faster algorithms for realizing one operation over $\mathbb{F}_{q^m}$ in $\mathbb{F}_q$ and therefore, it depends on the concrete implementation, which algorithm for the linearized composition is faster.

Table 3.3. Complexity of new algorithms for operations with linearized polynomials.

Operation	Notation	Method	Complexity
Linearized composition of $a(x), b(x) \in \mathbb{L}_{q^m}[x]$	$a(b(x))$	Algo. 3.1, p. 39, Subsec. 3.1.2	$\mathcal{M}(d_a, d_b)$ = $\mathcal{O}((\max\{d_a, d_b\})^{1.69})$ in $\mathbb{F}_{q^m}$
Linearized composition of $a(x), b(x) \in \mathbb{L}_{q^m}[x]$ modulo $(x^{[m]} - x)$	$a(b(x))$ mod $(x^{[m]} - x)$	Algo. 3.1, p. 39, Subsec. 3.1.2	$\mathcal{M}_m(m) = \mathcal{O}(m \cdot (\max\{d_a, d_b\})^{0.69})$ $\leq \mathcal{O}(m^{1.69})$ in $\mathbb{F}_{q^m}$
		Algo. 3.3, p. 42, Subsec. 3.1.3	$\mathcal{M}_m(m) = \mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m))$ $\leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$
Linearized multi-point evaluation of $a(x) \in \mathbb{L}_{q^m}[x]$ at $b_0, b_1, \dots, b_{s-1} \in \mathbb{F}_{q^m}$, where $s \mid m$	$a(b_0), \dots, a(b_{s-1})$	Algo. 3.2, p. 41, Subsec. 3.1.3	$\mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m))$ $\leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$
First half of LEEA of $a(x)$, $b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a > d_b$	FastHalfLEEA $\big(a(x); b(x); d_{stop}\big)$	Algo. 3.4, p. 45, Subsec. 3.1.4	$\mathcal{O}_{EA}(d_a, d_b) =$ $\mathcal{O}\big(\max\{\mathcal{D}(d_a, d_b),$ $\mathcal{M}(d_a, d_b)\} \log d_a\big)$ in $\mathbb{F}_{q^m}$

3.1.2 Fast Linearized Composition Using Fragmented Polynomials

In this subsection, we present a fast algorithm that calculates the linearized composition $a(b(x))$ with complexity $\mathcal{M}(d_a, d_b) = \mathcal{O}(\max\{d_a, d_b\}^{1.69})$ operations in $\mathbb{F}_{q^m}$ instead of $\mathcal{O}((d_a + d_b)^2)$ as in (2.12). The approach is based on splitting $a(x)$ into smaller linearized polynomials and on fast matrix multiplication. A similar fragmentation was used in [BK78, Algorithm 2.1] for calculating the composition of power series. We explain the idea of the fast composition in the following, summarize it in Algorithm 3.1 and prove its complexity. Our algorithm provides a solution to Problem 3.1 (a) of calculating the linearized composition efficiently. A solution to Problem 3.1 (b) follows directly and is stated in Corollary 3.1.

Let the assumptions of Problem 3.1 (a) be satisfied and let $n^* \overset{\text{def}}{=} \lceil \sqrt{n}\, \rceil = \lceil \sqrt{\max\{d_a, d_b\} + 1}\, \rceil$. We fragment $a(x) = \sum_{j=0}^{d_a} a_j x^{[j]}$ into n^* smaller polynomials $a^{(i)}(x)$ of q-degree less than n^*:

$$a^{(i)}(x) = \sum_{j=0}^{n^*-1} a_{in^*+j}\, x^{[in^*+j]}, \quad \forall i \in [0, n^* - 1], \tag{3.1}$$

with $a_h = 0$ if $h > d_a$. Therefore,

$$a(x) = \sum_{i=0}^{n^*-1} a^{(i)}(x) = \sum_{i=0}^{n^*-1} \sum_{j=0}^{n^*-1} a_{in^*+j}\, x^{[in^*+j]}.$$

The following lemma shows how $a(b(x))$ can be calculated by one (fast) matrix multiplication.

Lemma 3.1 (Calculation by Matrix Multiplication).
Let two linearized polynomials $a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}, b(x) = \sum_{i=0}^{d_b} b_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ be given and let $n^* = \lceil \sqrt{\max\{d_a, d_b\} + 1}\ \rceil$. Denote $c(x) = a(b(x))$ and let $a^{(i)}(x)$ be defined as in (3.1). *Let $c^{(i)}(x) = a^{(i)}(b(x))$, for all $i \in [0, n^* - 1]$, and denote the coefficients of $c^{(i)}(x)$ by*

$$c^{(i)}(x) = \sum_{j=0}^{d_b+n^*-1} c_j^{(i)} x^{[in^*+j]}.$$

Then, $c(x) = \sum_{i=0}^{n^*-1} c^{(i)}(x)$ and the coefficients of $(c^{(i)}(x))^{[-in^*]}$, for all $i \in [0, n^* - 1]$, are given by the following matrix multiplication:

$$\begin{pmatrix} c_0^{(0)} & c_1^{(0)} & \dots & c_{d_b+n^*-1}^{(0)} \\ c_0^{(1)[-n^*]} & c_1^{(1)[-n^*]} & \dots & c_{d_b+n^*-1}^{(1)[-n^*]} \\ \vdots & \vdots & \ddots & \vdots \\ c_0^{(n^*)[-(n^*-1)n^*]} & c_1^{(n^*)[-(n^*-1)n^*]} & \dots & c_{d_b+n^*-1}^{(n^*)[-(n^*-1)n^*]} \end{pmatrix} = \mathbf{A} \cdot \mathbf{B} \overset{\text{def}}{=} \tag{3.2}$$

$$\begin{pmatrix} a_0 & a_1 & \dots & a_{n^*-1} \\ a_{n^*}^{[-n^*]} & a_{n^*+1}^{[-n^*]} & \dots & a_{2n^*-1}^{[-n^*]} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n^*-1)n^*}^{[-n^*(n^*-1)]} & a_{(n^*-1)n^*+1}^{[-n^*(n^*-1)]} & \dots & a_{n^*n^*-1}^{[-n^*(n^*-1)]} \end{pmatrix} \cdot \begin{pmatrix} b_0^{[0]} & b_1^{[0]} & \dots & b_{d_b}^{[0]} & & & \\ & b_0^{[1]} & b_1^{[1]} & \dots & b_{d_b}^{[1]} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & b_0^{[n^*-1]} & & \dots & b_{d_b}^{[n^*-1]} \end{pmatrix}.$$

Proof. Since $a(x) = \sum_{i=0}^{n^*-1} a^{(i)}(x)$, we immediately obtain $c(x) = \sum_{i=0}^{n^*-1} c^{(i)}(x)$ and it remains to prove the calculation of the coefficients of $c(x)$ as in (3.2).

Note that $(a(x))^{[h]} = a^{[h]}(x) = \sum_{i=0}^{d_a} a_i^{[h]} x^{[h+i]}$ for any $a(x) \in \mathbb{L}_{q^m}[x]$ and any integer h. Hence, with (3.1), we obtain $\forall i \in [0, n^* - 1]$:

$$\big(c^{(i)}(x)\big)^{[-in^*]} = a^{(i)[-in^*]}(b(x)) = \sum_{j=0}^{n^*-1} a_{in^*+j}^{[-in^*]} \cdot b^{[in^*+j-in^*]}(x) = \sum_{j=0}^{n^*-1} a_{in^*+j}^{[-in^*]} \cdot \Big(\sum_{h=0}^{d_b} b_h^{[j]} x^{[j+h]} \Big).$$

The important observation is that the expression in the sum over h does not depend on i. The coefficients of $(c^{(i)}(x))^{[-in^*]}$ are therefore:

$$(c_0^{(i)[-in^*]}\ c_1^{(i)[-in^*]}\ \dots\ c_{d_b+n^*-1}^{(i)[-in^*]}) = (a_{in^*}^{[-in^*]}\ a_{in^*+1}^{[-in^*]}\ \dots\ a_{in^*+n^*-1}^{[-in^*]}) \cdot \mathbf{B}, \qquad \forall i \in [0, n^* - 1],$$

where the $n^* \times (d_b + n^*)$ matrix $\mathbf{B}$ is defined as in (3.2). Since $\mathbf{B}$ is independent of i, we can write the calculations for all i as matrix multiplication and the statement follows. ∎

Hence, we obtain the coefficients of $(c^{(i)}(x))^{[-in^*]}$ by the matrix multiplication from (3.2) and the coefficients of $c^{(i)}(x)$ from $(c^{(i)}(x))^{[-in^*]}$ by a simple q-power. Finally, we sum up over i to obtain the linearized composition $c(x) = a(b(x)) = \sum_{i=0}^{n^*-1} c^{(i)}(x)$.

This principle is summarized in Algorithm 3.1 and the complexity is analyzed in Theorem 3.1.

Theorem 3.1 (Linearized Composition with Algorithm 3.1).
Let two linearized polynomials $a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}, b(x) = \sum_{i=0}^{d_b} b_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ be given. Then, Algorithm 3.1 calculates $c(x) = a(b(x))$ with complexity $\mathcal{M}(d_a, d_b) = \mathcal{O}((\max\{d_a, d_b\})^{1.69})$ operations in $\mathbb{F}_{q^m}$ and is therefore a solution to Problem 3.1 (a).

Proof. Lemma 3.1 proves that Algorithm 3.1 correctly calculates $c(x) = a(b(x))$. The complexity of Algorithm 3.1 is analyzed in the following, where we denote again $n = \max\{d_a, d_b\} + 1$.

- Line 1: The complexity of this step is negligible since it only splits the polynomial $a(x)$.
- Line 2: The complexity of q-powers is negligible (see Section 3.1.1).
- Line 3: This step dominates the overall complexity of Algorithm 3.1 and can be accomplished by a fast matrix multiplication as follows.
  We split the $n^* \times (d_b + n^*)$ matrix $\mathbf{B}$ into $\lceil (d_b + n^*)/n^* \rceil \leq \lceil \sqrt{n} + 1 \rceil \leq \sqrt{n} + 2 \leq \max\{d_a, d_b\} + 3$ matrices of size $n^* \times n^*$ and multiply each of these matrices from the left by $\mathbf{A}$. Calculating $\mathbf{A} \cdot \mathbf{B}$ is then equivalent to multiplying at most $\sqrt{n} + 2$ times a $n^* \times n^*$ matrix.
  Let $\mathcal{N}(n)$ denote the complexity of multiplying two $n \times n$ matrices. The *Coppersmith–Winograd algorithm* has complexity $\mathcal{N}(n) = \mathcal{O}(n^{2.376})$, see e.g. [GG03]. Thus, Line 3 can be computed with $\mathcal{O}(\sqrt{n}\, \mathcal{N}(n^*)) = \mathcal{O}(\sqrt{n}\, \mathcal{N}(\sqrt{n})) = \mathcal{O}(n^{0.5}(n^{0.5})^{2.376}) = \mathcal{O}(n^{1.69})$ operations in $\mathbb{F}_{q^m}$.
- Line 4: The q-powers are again negligible.
- Line 5: Since consecutive $c^{(i)}(x)$ overlap at most in d_b coefficients, the overall sum requires at most $n^* \cdot d_b \leq n^* \cdot n = n^{1.5}$ additions over $\mathbb{F}_{q^m}$, i.e., this step has complexity $\mathcal{O}(n^{1.5})$ in $\mathbb{F}_{q^m}$.

The overall complexity of Algorithm 3.1 is therefore dominated by the matrix multiplication and is $\mathcal{M}(n) = \mathcal{O}(n^{1.69}) = \mathcal{O}((\max\{d_a, d_b\})^{1.69})$ operations in $\mathbb{F}_{q^m}$. ∎

Algorithm 3.1.
$c(x) \leftarrow \textsc{FastLinComp}\big(a(x); b(x)\big)$

```
Input: $a(x); b(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q a(x) = d_a$, $\deg_q b(x) = d_b$
Initialize: $n^* \leftarrow \lceil \sqrt{\max\{d_a, d_b\} + 1}\, \rceil$
1 Fragmentation (see (3.1)): $a^{(i)}(x) \leftarrow \sum_{j=0}^{n^*-1} a_{in^*+j}\, x^{[in^*+j]}$, for all $i \in [0, n^* - 1]$
2 Calculate $q$-powers: $a_j^{[-n^* i]}$, for all $j \in [0, d_a]$ and $i \in [1, n^* - 1]$
                        $b_j^{[i]}$, for all $j \in [0, d_b]$ and $i \in [0, n^* - 1]$
3 Set up matrices $\mathbf{A}$ and $\mathbf{B}$ as in (3.2) and calculate $\mathbf{A} \cdot \mathbf{B}$
4 Calculate $q$-powers: obtain $c^{(i)}(x)$ out of $(c^{(i)}(x))^{[-in^*]}$, for all $i \in [0, n^* - 1]$
5 Summation: $c(x) = \sum_{i=0}^{n^*-1} c^{(i)}(x)$
Output: $c(x) = a(b(x)) \in \mathbb{L}_{q^m}[x]$ with $\deg_q c(x) = d_a + d_b$
```

In order to apply Algorithm 3.1 to Problem 3.1 (b), we replace $\mathbf{B}$ by a q-circulant matrix.

Corollary 3.1 (Linearized Composition modulo $(x^{[m]} - x)$ with Algorithm 3.1).
Let two linearized polynomials $a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}$, $b(x) = \sum_{i=0}^{d_b} b_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ with $d_a, d_b < m$ be given. Replace the $n^* \times (d_b + n^*)$ matrix $\mathbf{B}$ in Line 3 of Algorithm 3.1 by the following $n^* \times m$ matrix

$$\mathbf{B}_m = \begin{pmatrix} b_0^{[0]} & b_1^{[0]} & b_2^{[0]} & \dots & b_{m-1}^{[0]} \\ b_{m-1}^{[1]} & b_0^{[1]} & b_1^{[1]} & \dots & b_{m-2}^{[1]} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m-n^*+1}^{[n^*-1]} & b_{m-n^*+2}^{[n^*-1]} & b_{m-n^*+3}^{[n^*-1]} & \dots & b_{m-n^*}^{[n^*-1]} \end{pmatrix},$$

with $b_i \overset{\text{def}}{=} 0$ for $i > d_b$.

Then, Algorithm 3.1 calculates $c(x) = a(b(x)) \mod (x^{[m]}] - x)$ with complexity $\mathcal{M}_m(m) = \mathcal{O}(m(\max\{d_a, d_b\})^{0.69}) \leq \mathcal{O}(m^{1.69})$ operations in $\mathbb{F}_{q^m}$ and is therefore a solution to Problem 3.1 (b).

The complexity of Algorithm 3.1 might be further reduced when we use a fast matrix multiplication algorithm which takes advantage of the fact that $\mathbf{B}$ and $\mathbf{B}_m$ are highly structured matrices.

3.1.3 Fast Linearized Composition Using Fast Multi-Point Evaluation

This subsection provides an algorithm for efficiently calculating the linearized multi-point evaluation as well as an algorithm for fast linearized composition modulo $(x^{[m]} - x)$. The complexity of both algorithms is in the order of $\mathcal{O}(m^3)$ over the *ground field* $\mathbb{F}_q$.

The following theorem provides the basis of the algorithms presented in this subsection and can be seen as an equivalent of the *convolutional theorem* for linearized polynomials[1].

Theorem 3.2 (Transformed Values of Composition).
Let two linearized polynomials $a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}, b(x) = \sum_{i=0}^{d_b} b_i x^{[i]} \in \mathbb{L}_{q^m}[x]$ be given. If possible, let s be an integer such that $d_a + d_b < s$ and s divides m. If such an integer does not exist, let $s = m$.

Let $\widehat{b}(x) = \sum_{i=0}^{s-1} \widehat{b}_i x^{[i]}$ denote the q-transform of $b(x)$ with respect to an ordered normal basis $\boldsymbol{\beta} = (\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[s-1]})$ in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$ according to Definition 2.12. Moreover, let

$$c(x) \equiv a(b(x)) \mod (x^{[m]} - x),$$

and let $\widehat{c}(x) = \sum_{i=0}^{s-1} \widehat{c}_i x^{[i]}$ denote its q-transform with respect to $\boldsymbol{\beta}$. Then,

$$\widehat{c}_i = a\big(\widehat{b}_i\big), \quad \forall i \in [0, s-1].$$

Proof. If an integer s exists which divides m and additionally, $d_a + d_b < s$, we obtain $c(x) = a(b(x)) \mod (x^{[m]} - x) = a(b(x)) \mod (x^{[s]} - x)$. Clearly, this holds also if $s = m$.
Since $s \mid m$, there is a normal basis in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$ (Lemma 2.3). Due to Definition 2.12, the transformed coefficients are $\widehat{c}_i = c(\beta^{[i]})$ and $\widehat{b}_i = b(\beta^{[i]})$ for all $i \in [0, s-1]$. For $d_a + d_b < s$, the modulo operation is useless and for $s = m$, we know that $\beta^{[m]} = \beta$ holds and the modulo operation is implicitly included. Hence, $\widehat{c}_i = c(\beta^{[i]}) = a(b(\beta^{[i]})) = a\big(\widehat{b}_i\big)$. ∎

Theorem 3.2 indicates that a fast linearized composition can be realized by fast linearized multi-point evaluation. Hence, finding a solution to Problem 3.1 (c) is considered as the first task on the way to fast linearized composition. An algorithm for fast linearized multi-point evaluation is of general interest, e.g. for encoding Gabidulin codes with $g_i \neq \beta^{[i]}$. The following lemma shows how we can use a matrix multiplication over the *ground field* $\mathbb{F}_q$ to accomplish this multi-point evaluation.

Lemma 3.2 (Multi-Point Evaluation with Matrices).
Let a linearized polynomial $a(x) \in \mathbb{L}_{q^m}[x]$ with $d_a = \deg_q a(x) < m$ and s points $\widehat{b}_0, \widehat{b}_1, \dots, \widehat{b}_{s-1} \in \mathbb{F}_{q^m}$ be given, for some s dividing m. Let $\widehat{a}(x) = \sum_{i=0}^{m-1} \widehat{a}_i x^{[i]}$ denote the q-transform of $a(x)$ with respect to an ordered normal basis $\boldsymbol{\beta} = (\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[m-1]})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Denote

$$\begin{aligned} \text{ext}_{\boldsymbol{\beta}}\,(\widehat{a}_i) &= (\widehat{A}_{0,i}\ \widehat{A}_{1,i}\ \dots\ \widehat{A}_{m-1,i})^T, \quad \forall i \in [0, m-1], \\ \text{ext}_{\boldsymbol{\beta}}(\widehat{b}_i) &= (\widehat{B}_{0,i}\ \widehat{B}_{1,i}\ \dots\ \widehat{B}_{m-1,i})^T, \quad \forall i \in [0, s-1]. \end{aligned}$$

[1] The convolutional theorem for (usual) polynomials over finite fields states that polynomial multiplication modulo $(x^n - 1)$ is equivalent to element-wise multiplication of the coefficients in the transform domain, see e.g. [Bla03, Theorem 6.1.3], [Bos98, Theorem 3.6].

Then, for $\widehat{\mathbf{A}} \in \mathbb{F}_q^{m \times m}$ and $\widehat{\mathbf{B}} \in \mathbb{F}_q^{m \times s}$, the extension of the multi-point evaluation is given by:

$$\operatorname{ext}_{\boldsymbol{\beta}}\Big(\big(a(\widehat{b}_0)\ a(\widehat{b}_1)\ \dots\ a(\widehat{b}_{s-1})\big)\Big) = \widehat{\mathbf{A}} \cdot \widehat{\mathbf{B}} \overset{\text{def}}{=}$$

$$\begin{pmatrix} \widehat{A}_{0,0} & \widehat{A}_{0,1} & \dots & \widehat{A}_{0,m-1} \\ \widehat{A}_{1,0} & \widehat{A}_{1,1} & \dots & \widehat{A}_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{A}_{m-1,0} & \widehat{A}_{m-1,1} & \dots & \widehat{A}_{m-1,m-1} \end{pmatrix} \cdot \begin{pmatrix} \widehat{B}_{0,0} & \widehat{B}_{0,1} & \dots & \widehat{B}_{0,s-1} \\ \widehat{B}_{1,0} & \widehat{B}_{1,1} & \dots & \widehat{B}_{1,s-1} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{B}_{m-1,0} & \widehat{B}_{m-1,1} & \dots & \widehat{B}_{m-1,s-1} \end{pmatrix}. \tag{3.3}$$

Proof. Since $B^{[i]} = B$ for any $B \in \mathbb{F}_q$ and all i, we can rewrite the evaluation values by:

$$a(\widehat{b}_i) = a\Bigg(\sum_{j=0}^{m-1} \widehat{B}_{j,i} \cdot \beta^{[j]}\Bigg) = \sum_{j=0}^{m-1} \widehat{B}_{j,i} \cdot a\big(\beta^{[j]}\big), \quad \forall i \in [0, s-1].$$

Hence, all s evaluation values can be calculated by the following vector-matrix multiplication:

$$\big(a(\widehat{b}_0)\ a(\widehat{b}_1)\ \dots\ a(\widehat{b}_{s-1})\big) = \big(a(\beta^{[0]})\ a(\beta^{[1]})\ \dots\ a(\beta^{[m-1]})\big) \cdot \widehat{\mathbf{B}}.$$

Due to the definition of the q-transform,

$$a(\beta^{[i]}) = \widehat{a}_i = \sum_{j=0}^{m-1} \widehat{A}_{j,i} \cdot \beta^{[j]}, \quad \forall i \in [0, m-1].$$

Therefore, the evaluation values are $(a(\widehat{b}_0)\ a(\widehat{b}_1)\ \dots\ a(\widehat{b}_{s-1})) = \boldsymbol{\beta} \cdot \widehat{\mathbf{A}} \cdot \widehat{\mathbf{B}}$. ∎

When the number of evaluation points s does not divide m, we pad with zeros until the number of points divides m. Lemma 3.2 is connected to considering $a(x)$ as an $\mathbb{F}_q$-linear map as in Subection 2.2.3 and looking at its associated matrix. Similar transform-based ideas were used in [GA86]. Thus, Problem 3.1 (c) of linearized multi-point-evaluation can be solved efficiently by the following Algorithm 3.2.

It is important to remark that the q-transform in Line 1 of Algorithm 3.2 is done with regard to a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ (and not of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$), even if $d_a < s$, since this is required by the proof of Lemma 3.2.

Algorithm 3.2.
$(a(\widehat{b}_0)\ a(\widehat{b}_1)\ \dots\ a(\widehat{b}_{s-1})) \leftarrow$ FastLinMultEval$\big(a(x); \{\widehat{b}_0, \widehat{b}_1, \dots, \widehat{b}_{s-1}\}; \boldsymbol{\beta}\big)$

```
Input: $a(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q a(x) = d_a < m$;
       $\widehat{b}_0, \widehat{b}_1, \dots, \widehat{b}_{s-1} \in \mathbb{F}_{q^m}$;
       normal basis $\boldsymbol{\beta} = (\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[m-1]})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$
1 Calculate $q$-transform of $a(x)$ w.r.t. to $\boldsymbol{\beta}$: obtain $\widehat{a}_i = a(\beta^{[i]})$, $i = [0, m-1]$ as in (2.21)
2 Calculate $\operatorname{ext}_{\boldsymbol{\beta}}(\widehat{a}_i)$ and $\operatorname{ext}_{\boldsymbol{\beta}}(\widehat{b}_j)$ for all $i \in [0, m-1], j \in [0, s-1]$
3 Set up matrices $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ as in (3.3) and calculate $\mathbf{a} \leftarrow \boldsymbol{\beta} \cdot \widehat{\mathbf{A}} \cdot \widehat{\mathbf{B}}$
  Output: $\mathbf{a} = (a(\widehat{b}_0)\ a(\widehat{b}_1)\ \dots\ a(\widehat{b}_{s-1})) \in \mathbb{F}_{q^m}^s$
```

Lemma 3.3 (Linearized Multi-Point Evaluation with Algorithm 3.2).
Let a linearized polynomial $a(x) \in \mathbb{L}_{q^m}[x]$ with $d_a = \deg_q a(x) < m$ and the s points $\widehat{b}_0, \widehat{b}_1, \dots, \widehat{b}_{s-1} \in \mathbb{F}_{q^m}$, where $s \mid m$, be given. Then, Algorithm 3.2 finds the s evaluation values

$$a(\widehat{b}_0), a(\widehat{b}_1), \dots, a(\widehat{b}_{s-1})$$

with complexity $\mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m)) \leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$ and is therefore a solution to Problem 3.1 (c).

Proof. Due to Theorem 3.2 and Lemma 3.2, Algorithm 3.2 returns the correct result. For the complexity, let us analyze the steps of Algorithm 3.2 in detail.

- Line 1: The q-transform of length m for a polynomial of q-degree d_a costs $\mathcal{O}(m d_a \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$ (compare Table 3.1).
- Line 2: The mapping $\operatorname{ext}_{\boldsymbol{\beta}}$ (Definition 2.1) requires no cost.
- Line 3: $\widehat{\mathbf{A}}$ is an $m \times m$ matrix and $\widehat{\mathbf{B}}$ is an $m \times s$ matrix over $\mathbb{F}_q$ and therefore with straightforward matrix multiplication, this step costs $\mathcal{O}(sm^2)$ operations in $\mathbb{F}_q$.

Thus, Algorithm 3.2 requires overall complexity $\mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m))$ in $\mathbb{F}_q$. ∎

Based on Theorem 3.2 and Algorithm 3.2, we can calculate the linearized composition efficiently. This is shown in Algorithm 3.3, which computes the composition $c(x) = a(b(x)) \mod (x^{[m]} - x)$ in three main steps. It relies on the fact that $\widehat{c}_i = a(\widehat{b}_i)$ as proven in Theorem 3.2.

Algorithm 3.3.
$c(x) \leftarrow \textsc{FastLinCompTrans}\big(a(x); b(x); \boldsymbol{\beta}\big)$

```
Input: $a(x); b(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q a(x) = d_a$, $\deg_q b(x) = d_b$;
       normal basis $\boldsymbol{\beta} = (\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[m-1]})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$

Initialize: Define transform length:
            if $\exists i : i \mid m$ and $d_a + d_b < i$ then
                $s \leftarrow i$
            else
                $s \leftarrow m$
            Find normal basis $\boldsymbol{\beta}_s = (\beta_s^{[0]}\ \beta_s^{[1]}\ \dots\ \beta_s^{[s-1]})$ of $\mathbb{F}_{q^s}$ over $\mathbb{F}_q$ and its dual basis $\boldsymbol{\beta}_s^{\perp}$

Calculate the $q$-transform of $b(x)$ w.r.t. to $\boldsymbol{\beta}_s$: obtain $\widehat{b}_i = b(\beta^{[i]})$, for all $i \in [0, s-1]$, as in (2.21)
Multi-point evaluation:
    $(\widehat{c}_0\ \widehat{c}_1\ \dots\ \widehat{c}_{s-1}) \leftarrow \textsc{FastLinMultEval}\big(a(x); \{\widehat{b}_0, \widehat{b}_1, \dots, \widehat{b}_{s-1}\}; \boldsymbol{\beta}\big)$ with Algorithm 3.2
Define linearized polynomial $\widehat{c}(x) = \sum_{i=0}^{s-1} \widehat{c}_i x^{[i]}$
Calculate inverse $q$-transform of $\widehat{c}(x)$ w.r.t. to $\boldsymbol{\beta}_s^{\perp}$: $c_i = \widehat{c}({\beta_s^{\perp}}^{[i]})$, for all $i \in [0, s-1]$, as in (2.22)

Output: $c(x) = a(b(x)) \mod (x^{[m]} - x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q c(x) \leq \min\{d_a + d_b, m-1\}$
```

We assume in the following that the ordered normal bases $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_s$ are known and therefore, finding these bases requires no additional complexity.

Lemma 3.4 (Linearized Composition modulo $(x^{[m]} - x)$ with Algorithm 3.3).
Let $a(x) = \sum_{i=0}^{d_a} a_i x^{[i]}$ and $b(x) = \sum_{i=0}^{d_b} b_i x^{[i]}$ with $d_a, d_b < m$ be given. If existing, let s be an integer such that $d_a + d_b < s$ and s divides m. If such an integer does not exist, let $s = m$.

Then, Algorithm 3.3 calculates

$$c(x) = a(b(x)) \mod (x^{[m]} - x)$$

with complexity $\mathcal{M}_m(m) = \mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m)) \leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m))$ *operations in* $\mathbb{F}_q$ *and is therefore a solution to Problem 3.1 (b).*

Proof. The correctness of the result of Algorithm 3.3 follows from Theorem 3.2 and Lemma 3.2. For the complexity, we analyze the steps of Algorithm 3.3 in the following.

- Line 1: The first step is a q-transform of length s and requires therefore $\mathcal{O}(sd_b \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$ (compare Table 3.1).
- Line 2: The call of Algorithm 3.2 costs $\mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$ (Lemma 3.3).
- Line 3: This step has negligible complexity.
- Line 4: The last step is an inverse q-transform of length s and costs at most $\mathcal{O}(s^2 \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$, since $\deg_q \widehat{c}(x) < s$ (compare Table 3.1).

Hence, the overall complexity of Algorithm 3.3 is in the order of $\mathcal{O}(\max\{d_a, s\} m \operatorname{comp}(\mathbf{T}_m)) \leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m))$ operations in $\mathbb{F}_q$. ∎

Algorithm 3.3 establishes a connection between the linearized composition and matrix multiplication. This relation can also be used vice versa: assume, two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{m \times m}$ are given; represent them as vectors in $\mathbb{F}_{q^m}^n$, calculate their inverse q-transforms and compute the composition of the corresponding linearized polynomials. The q-transform of this result is the matrix multiplication $\mathbf{A} \cdot \mathbf{B}$. Thus, provided that (inverse) q-transforms are efficient, matrix multiplication and the linearized composition are equivalent. Linearized composition faster than matrix multiplication would imply a major breakthrough in matrix multiplication. Therefore, it is quite unlikely to find an algorithm for the linearized composition faster than Algorithm 3.3 (up to using a faster matrix multiplication algorithm).

3.1.4 Fast Linearized (Extended) Euclidean Algorithm

In the following, we derive an efficient LEEA (see Algorithm 2.3 for the standard algorithm), which is a generalization of the fast extended Euclidean algorithm for usual polynomials by Aho and Hopcroft [AH74] and Blahut [Bla85]. Our fast LEEA is based on the so-called *Divide & Conquer* strategy, which splits a problem of "size" n into two halves, each of "size" $n/2$. The structure of these halves should be the same as the original problem. The calculation can be accelerated if there exist fast solutions for the halved problems and if they can be combined with low complexity [AH74, Bla85, GG03].

In order to break the LEEA into two halves, recall from (2.16) that we can write

$$\mathbf{Q}^{(i,1)} = \mathbf{Q}^{(i,h+1)} \circ \mathbf{Q}^{(h,1)},$$

for any integer $1 \leq h \leq i-1$. Calculating $\mathbf{Q}^{(i,i)} = \mathbf{Q}^{(i)}$ is equivalent to one step of the LEEA as in (2.15). The first half of the splitted LEEA hence consists of the first h iterations and outputs $\mathbf{Q}^{(h,1)}$. The second uses $\mathbf{Q}^{(h,1)}$ as input and considers the problem of calculating $\mathbf{Q}^{(i,h+1)}$. To reduce the complexity, $\mathbf{Q}^{(i,h+1)}$ and $\mathbf{Q}^{(h,1)}$ have to be calculated efficiently. For that purpose, we use the fact that the first calculations of the LEEA only depend on some leading coefficients of the input polynomials.

Theorem 3.3 (Upper Coefficients of the Input Polynomials of the LEEA).
Let $a(x), b(x) \in \mathbb{L}_{q^m}[x]$, *with q-degrees* $d_a \geq d_b$ *be given. Split these polynomials into two parts:*

$$a(x) = a'\big(x^{[h]}\big) + a''(x), \quad b(x) = b'\big(x^{[h]}\big) + b''(x), \tag{3.4}$$

for some h, satisfying $0 \leq h \leq 2d_b - d_a$. Let $q(x), r(x)$ and $q'(x), r'(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q r(x) < d_b$ and $\deg_q r'(x) < \deg_q b'(x)$ be defined such that

$$a(x) = q\big(b(x)\big) + r(x), \quad a'(x) = q'\big(b'(x)\big) + r'(x). \tag{3.5}$$

Then,

$$q(x) = q'(x), \quad r(x) = r'\big(x^{[h]}\big) + r''(x),$$

for some $r''(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q r''(x) < h + d_a - d_b$.

Proof. The proof can be found in Appendix A.1. ∎

Therefore, Theorem 3.3 shows that if $h \leq 2d_b - d_a$, the quotient polynomial $q(x)$ does not depend on the h lower coefficients of $a(x)$ and $b(x)$. Further, these h lower coefficients influence only the $h + d_a - d_b$ lowest coefficients of the remainder $r(x)$. The following lemma shows that about half of the iterations of the LEEA can be calculated correctly without knowing $r''(x)$.

Lemma 3.5 (Quotients in the Iterations of the LEEA).
Let $a(x), b(x) \in \mathbb{L}_{q^m}[x]$, with q-degrees $d_a \geq d_b$ be given and let them be fragmented into smaller polynomials as in (3.4) *with $0 \leq h \leq 2d_b - d_a$. Let $\mathbf{Q}^{(i)}$ and $\mathbf{Q}'^{(i)}$ denote the matrices as in* (2.16) *in step i of* RightLEEA$\big(a(x); b(x); d_{stop}\big)$ *and* RightLEEA$\big(a'(x); b'(x); d'_{stop}\big)$*, respectively. Then,*

$$\mathbf{Q}^{(i)} = \mathbf{Q}'^{(i)} \tag{3.6}$$

for each i where $\deg_q r'^{(i)}(x) \geq (d_a - h)/2$ and $r'^{(i)}(x)$ is the remainder in step i of RightLEEA$\big(a'(x); b'(x); d'_{stop}\big)$.

Proof. The proof is similar to the proof of [Bla85, Theorem 10.7.3]. ∎

A fast realization of the whole LEEA can be given based on Lemma 3.5 and consists of two parts as in [Bla85]. However, for decoding, we only need the first steps of the LEEA—sometimes even less than half of the iterations. The acceleration of the first steps of the LEEA is given in Algorithm 3.4, called FastHalfLEEA. If we choose $d_{stop} = d_a/2$, then Algorithm 3.4 is the linearized equivalent of Blahut's HalfEucAlg [Bla85, Figure 10.8].

Recall from (2.17) and (2.19) that when we know $\mathbf{Q}^{(h,1)}$, we can immediately calculate $r^{(h)}(x)$, $u^{(h)}(x)$ and $v^{(h)}(x)$. For this reason, Algorithm 3.4 outputs only $\mathbf{Q}^{(h,1)}$.

As typical for Divide & Conquer algorithms, Algorithm 3.4 consists of two halves (Lines 1–9 and Lines 10–21). Each of these two halves implies a recursive call. The first recursive call is done in Line 8 with truncated polynomials. Then in Line 11, one linearized division is done, which is necessary to obtain the quotients in the recursions. In Lines 17–18, the polynomials are again truncated and the second recursive call follows in Line 20.

Assume, $\mathbf{Q}$ is the output of Algorithm 3.4, then we can calculate as in (2.17):

$$\begin{pmatrix} r^{(j-1)}(x) \\ r^{(j)}(x) \end{pmatrix} = \mathbf{Q} \circ \begin{pmatrix} a(x) \\ b(x) \end{pmatrix}. \tag{3.7}$$

These polynomials $r^{(j-1)}(x), r^{(j)}(x)$ satisfy the following lemma.

Lemma 3.6 (Degree of Output of Algorithm 3.4).
Let $\mathbf{Q} \leftarrow \textsc{FastHalfLEEA}\big(a(x); b(x); d_{stop}\big)$ *be the output of Algorithm 3.4 for some* $d_a > d_{stop} \geq d_a/2$, *where* $d_a = \deg_q a(x) \geq \deg_q b(x)$. *Let* $r^{(j-1)}(x)$ *and* $r^{(j)}(x) \in \mathbb{L}_{q^m}[x]$ *be as in* (3.8). *Then,*

$$\deg_q r^{(j-1)}(x) \geq d_{stop} \quad \textit{and} \quad \deg_q r^{(j)}(x) \leq d_{stop}. \tag{3.8}$$

Proof. The proof can be found in Appendix A.1. ∎

Algorithm 3.4.
$\mathbf{Q} \leftarrow \textsc{FastHalfLEEA}\big(a(x); b(x); d_{stop}\big)$

```
Input: $a(x); b(x) \in \mathbb{L}_{q^m}[x]$ with $d_a = \deg_q a(x) \geq d_b = \deg_q b(x)$;
       stopping degree $d_{stop}$, where $d_a > d_{stop} \geq d_a/2$
1  if $d_b \leq d_{stop}$ then
2      $\mathbf{Q} \leftarrow \begin{pmatrix} x & 0 \\ 0 & x \end{pmatrix}$
3  else
4      $h \leftarrow \lfloor d_a/2 \rfloor$
5      $a^{(1)}(x) \leftarrow \big(a(x) - (a(x) \mod x^{[h]})\big) \otimes x^{[-h]}$
6      $b^{(1)}(x) \leftarrow \big(b(x) - (b(x) \mod x^{[h]})\big) \otimes x^{[-h]}$
7      $d_{stop}^{(1)} \leftarrow \left\lfloor \frac{d_{stop}}{d_a} \cdot \deg_q a^{(1)}(x) \right\rfloor$
8      Recursive Call: $\mathbf{Q}^{(1)} \leftarrow \textsc{FastHalfLEEA}\big(a^{(1)}(x); b^{(1)}(x); d_{stop}^{(1)}\big)$
9      $\begin{pmatrix} a(x) \\ b(x) \end{pmatrix} \leftarrow \mathbf{Q}^{(1)} \otimes \begin{pmatrix} a(x) \\ b(x) \end{pmatrix}$
10     if $\deg_q b(x) > \frac{d_{stop}}{d_a} \cdot \deg_q a(x)$ then
11         $q(x); r(x) \leftarrow \textsc{RightDiv}\big(a(x); b(x)\big)$ with Algorithm 2.1
12         $\mathbf{Q} \leftarrow \begin{pmatrix} 0 & x \\ x & -q(x) \end{pmatrix}$
13         $\begin{pmatrix} a(x) \\ b(x) \end{pmatrix} \leftarrow \mathbf{Q} \otimes \begin{pmatrix} a(x) \\ b(x) \end{pmatrix}$
14         $\mathbf{Q} \leftarrow \mathbf{Q} \otimes \mathbf{Q}^{(1)}$
15         if $\deg_q b(x) > \frac{d_{stop}}{d_a} \cdot \deg_q a(x)$ then
16             $h \leftarrow \left\lfloor \frac{d_{stop}(d_a - \deg_q a(x))}{d_a - d_{stop}} \right\rfloor$
17             $a^{(1)}(x) \leftarrow (a(x) - (a(x) \mod x^{[h]})) \otimes x^{[-h]}$
18             $b^{(1)}(x) \leftarrow (b(x) - (b(x) \mod x^{[h]})) \otimes x^{[-h]}$
19             $d_{stop}^{(1)} \leftarrow \left\lfloor \frac{d_{stop}}{d_a} \cdot \deg_q a^{(1)}(x) \right\rfloor$
20             Recursive Call: $\mathbf{Q}^{(1)} \leftarrow \textsc{FastHalfLEEA}\big(a^{(1)}(x); b^{(1)}(x); d_{stop}^{(1)}\big)$
21             $\mathbf{Q} \leftarrow \mathbf{Q}^{(1)} \otimes \mathbf{Q}$
Output: $\mathbf{Q} \in \mathbb{L}_{q^m}[x]^{2 \times 2}$
```

Hence, Algorithm 3.4 outputs the matrix $\mathbf{Q}$, which provides the last remainder of q-degree at least the stopping degree d_{stop}. The degree constraints in Lines 10 and 15 of Algorithm 3.4 correspond to improvements from [GY79, BGY80]. They remarked that without these degree constraints, the algorithms by Aho and Hopcroft [AH74] and Blahut [Bla85] do not work properly. Our Algorithm 3.4 is a generalization of the algorithms by [AH74, Bla85], considering the improvements from [GY79, BGY80]. It is equivalent to their algorithms if the stopping degree is $d_{stop} = d_a/2$ and if $m = 1$, i.e., if we consider the field $\mathbb{F}_q$, since then $A^{[i]} = A$ for all elements $A \in \mathbb{F}_q$.

Lemma 3.7 (Complexity of Algorithm 3.4).
Algorithm 3.4 requires complexity $\mathcal{O}_{EA}(d_a, d_b) = \mathcal{O}(\max\{\mathcal{D}(d_a, d_b), \mathcal{M}(d_a, d_b)\} \log d_a)$ operations if $d_a = \deg_q a(x) \geq d_b = \deg_q b(x)$.

Proof. The complexity of the splitting operations in Lines 5, 6, 17, 18 and the value assignments are negligible. The linearized compositions from Lines 9, 13, 14, 21 cost in the order of $\mathcal{M}(d_a, d_b)$ operations. The linearized division in Line 11 requires in the order of $\mathcal{D}(d_a, d_b)$ operations. Recall from the proof of Lemma 3.6 that both recursive calls are done with polynomials of degree at most $d_a/2$. Hence, $\mathcal{O}_{EA}(d_a, d_b)$ is upper bounded by:

$$\mathcal{O}_{EA}(d_a, d_b) \leq 2 \cdot \mathcal{O}_{EA}(d_a/2, d_b/2) + \mathcal{M}(d_a, d_b) + \mathcal{D}(d_a, d_b).$$

It is known from the *master theorem* for linear recurrence relations (see e.g. [GG03]) that this inequality implies $\mathcal{O}_{EA}(d_a, d_b) \leq \mathcal{O}(\max\{\mathcal{M}(d_a, d_b), \mathcal{D}(d_a, d_b)\} \log d_a)$. ∎

Using fast linearized composition, this results in the following corollary.

Corollary 3.2 (Fast LEEA Using Fast Composition).
If we use Algorithm 3.1 from Subsection 3.1.2 for the linearized composition, then Algorithm 3.4 has complexity $\mathcal{O}(\max\{\mathcal{D}(d_a, d_b), \max\{d_a, d_b\}^{1.69}\} \log d_a)$ operations in $\mathbb{F}_{q^m}$.

If $d_a, d_b < m$ and we use Algorithm 3.3 (Subsection 3.1.3) for the linearized composition, then Algorithm 3.4 has complexity $\mathcal{O}(\max\{\mathcal{D}(d_a, d_b), d_a m \operatorname{comp}(\mathbf{T}_m)\} \log d_a)$ operations in $\mathbb{F}_q$.

Corollary 3.2 shows therefore that a fast linearized division would immediately provide a fast LEEA.

3.2 Decoding of Gabidulin Codes

Decoding of Gabidulin codes can generally be accomplished by two different principles: syndrome-based decoding (which relies on solving a key equation) as in [Gab85, Rot91, PT91, Gab92, RP04a] and interpolation-based decoding as in [Loi06].

In the course of this section, we first describe a known syndrome-based decoding principle of Gabidulin codes (Subsection 3.2.1) and second, we derive a new decoding algorithm in Subsection 3.2.2, which can be seen as the rank-metric equivalent of Gao's algorithm for decoding Reed–Solomon codes [Gao03]. Further, we show in Subsection 3.2.3 how this algorithm can be extended to error-erasure decoding of Gabidulin codes and how it can be accelerated based on the q-transform (Subsection 3.2.4). The results of Subsections 3.2.2 and 3.2.4 were partly published in [WAS11, WAS13].

3.2.1 Known Syndrome-Based Decoding Approaches

Following the descriptions of [Gab85, Rot91, Gab92], we explain the idea of syndrome-based BMD decoding (as defined in Section 2.1.3), without going into detail about the different algorithmic possibilities.

Let $\mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_{q^m}^n$ be the received word, where $\mathbf{c} \in \mathsf{Gab}[n, k]$. The goal of *decoding* is now to reconstruct $\mathbf{c}$, given only $\mathbf{r}$ and the code parameters. Clearly, this is possible only if the rank of the error $\mathbf{e}$ is not too big. Syndrome-based BMD decoding of Gabidulin codes follows similar steps as syndrome-based BMD decoding of Reed–Solomon codes. For Reed–Solomon codes, the two main steps are determining the "error locations" and finding the "error values", where the second step is considered to be much easier. Algebraic BMD decoding of Gabidulin codes also consists of two steps; however, the second one is not necessarily the easier one. The starting point of decoding Gabidulin codes is to decompose the error, based on the well-known rank decomposition of a matrix.

Lemma 3.8 (Rank Decomposition [MS74, Theorem 1]).
For any matrix $\mathbf{X} \in \mathbb{F}_q^{m \times n}$ of rank r there exist full rank matrices $\mathbf{Y} \in \mathbb{F}_q^{m \times r}$ and $\mathbf{Z} \in \mathbb{F}_q^{r \times n}$ such that $\mathbf{X} = \mathbf{Y}\mathbf{Z}$. Moreover, the column space of $\mathbf{X}$ is $\mathcal{C}_q(\mathbf{X}) = \mathcal{C}_q(\mathbf{Y}) \in \mathcal{G}_q(m,r)$ and the row space is $\mathcal{R}_q(\mathbf{X}) = \mathcal{R}_q(\mathbf{Z}) \in \mathcal{G}_q(n,r)$.

Therefore, we can rewrite the matrix representation of $\mathbf{e}$ with $\mathrm{rk}(\mathbf{e}) = t$ by:

$$\mathbf{E} = \mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{e}) = \mathbf{A} \cdot \mathbf{B}, \quad \text{with } \mathbf{A} \in \mathbb{F}_q^{m \times t},\ \mathbf{B} \in \mathbb{F}_q^{t \times n},$$

and if we define $\mathbf{a} \overset{\mathrm{def}}{=} \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{A}) \in \mathbb{F}_{q^m}^t$:

$$\mathbf{e} = \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{E}) = \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{A}) \cdot \mathbf{B} = \mathbf{a} \cdot \mathbf{B} = (a_0\ a_1\ \dots\ a_{t-1}) \cdot \mathbf{B}. \tag{3.9}$$

This decomposition is clearly not unique, but any of them is good for decoding. The two main steps of decoding Gabidulin codes are therefore: first, determine "a basis of the column space" of the error, i.e., find the vector $\mathbf{a}$ of a possible decomposition, and second, find the corresponding matrix $\mathbf{B}$, which fixes the row space[2]. Both steps are based on the *syndrome*, which can be calculated out of the received word (see Definition 2.8) by

$$\mathbf{s} = (s_0\ s_1\ \dots\ s_{n-k-1}) = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T, \tag{3.10}$$

where $\mathbf{H}$ is a parity-check matrix of the $\mathsf{Gab}[n,k]$ code (see (2.14)). We denote the associated syndrome polynomial by $s(x) = \sum_{i=0}^{n-k-1} s_i x^{[i]} \in \mathbb{L}_{q^m}[x]$. Its coefficients are:

$$s_i = \sum_{j=0}^{n-1} e_j h_j^{[i]} = \sum_{j=0}^{n-1} \sum_{l=0}^{t-1} a_l B_{l,j} h_j^{[i]} \overset{\mathrm{def}}{=} \sum_{l=0}^{t-1} a_l d_l^{[i]}, \quad \forall i \in [0, n-k-1], \tag{3.11}$$

with

$$d_l \overset{\mathrm{def}}{=} \sum_{j=0}^{n-1} B_{l,j} h_j. \tag{3.12}$$

We define the *error span polynomial* as the minimal subspace polynomial of the vector $\mathbf{a}$:

$$\Lambda(x) \overset{\mathrm{def}}{=} M_{a_0,a_1,\dots,a_{t-1}}(x) = \prod_{B_0=0}^{q-1} \cdots \prod_{B_{t-1}=0}^{q-1} \Big(x - \sum_{i=0}^{t-1} B_i a_i\Big). \tag{3.13}$$

Hence, due to Lemma 2.9, the error span polynomial $\Lambda(x)$ is a linearized polynomial of q-degree t and any $\mathbb{F}_q$-linear combination of roots of $\Lambda(x)$ is also a root of $\Lambda(x)$.

The first part of the decoding process is to determine $\Lambda(x)$, given the syndrome polynomial $s(x)$, and it is strongly based on the following theorem, the *key equation* for decoding Gabidulin codes.

[2] Note that it is possible to change the order of these two steps and search for a basis of the row space first and then find a corresponding matrix $\mathbf{A}$. This is a big difference to Reed–Solomon codes, where we cannot interchange the two main steps.

Theorem 3.4 (Key Equation for Decoding Gabidulin Codes [Gab85, Lemma 4]).
Let $\mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_{q^m}^n$ *be given, where* $\mathbf{c} \in \mathsf{Gab}[n,k]$ *over* $\mathbb{F}_{q^m}$ *and* $\operatorname{rk}(\mathbf{e}) = t < n-k$. *Denote by* $\mathbf{s} = (s_0\ s_1\ \dots\ s_{n-k-1}) = \mathbf{r} \cdot \mathbf{H}^T \in \mathbb{F}_{q^m}^{n-k}$ *the syndrome as in* (3.10) *and by* $s(x) = \sum_{i=0}^{n-k-1} s_i x^{[i]}$ *its associated polynomial.*

Let the error span polynomial $\Lambda(x)$ *with* $\deg_q \Lambda(x) = t$ *be defined as in* (3.13), *where* $\mathbf{a} = (a_0\ a_1\ \dots\ a_{t-1})$ *is a basis of the column space of* $\mathbf{e}$. *Then,*

$$\Omega(x) \equiv \Lambda(s(x)) = \Lambda(x) \circ s(x) \mod x^{[n-k]}, \tag{3.14}$$

for some $\Omega(x) \in \mathbb{L}_{q^m}[x]$ *with* $\deg_q \Omega(x) < t$.

Proof. With (2.12) and (3.11), the i-th coefficient of $\Lambda(s(x))$ can be calculated by

$$\Omega_i \overset{\text{def}}{=} \big[\Lambda(s(x))\big]_i = \sum_{j=0}^{i} \Lambda_j s_{i-j}^{[j]} = \sum_{j=0}^{i} \Lambda_j \left(\sum_{l=0}^{t-1} a_l d_l^{[i-j]} \right)^{[j]} = \sum_{l=0}^{t-1} d_l^{[i]} \sum_{j=0}^{i} \Lambda_j \cdot a_l^{[j]}. \tag{3.15}$$

For any $i \geq t$ this gives:

$$\Omega_i = \sum_{l=0}^{t-1} d_l^{[i]} \Lambda(a_l) = 0, \quad \forall i \in [t, n-k-1-1], \tag{3.16}$$

since $\Lambda(x)$ has $a_i, \forall i \in [0, t-1]$, as roots, see (3.13), and therefore $\deg_q \Omega(x) < \deg_q \Lambda(x) = t$. ∎

Alternatively, we can derive a key equation for the *row* space of the error word (compare e.g. [SK09a]).

Theorem 3.5 (Row Space Key Equation for Decoding Gabidulin Codes).
Let $\mathbf{r} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_{q^m}^n$ *be given, where* $\mathbf{c} \in \mathsf{Gab}[n,k]$ *over* $\mathbb{F}_{q^m}$ *and* $\operatorname{rk}(\mathbf{e}) = t < n-k$. *Denote by* $\mathbf{s} = (s_0\ s_1\ \dots\ s_{n-k-1}) = \mathbf{r} \cdot \mathbf{H}^T \in \mathbb{F}_{q^m}^{n-k}$ *the syndrome as in* (3.10) *and by* $s(x) = \sum_{i=0}^{n-k-1} s_i x^{[i]}$ *its associated polynomial.*

Let the row error span polynomial be $\Gamma(x) = M_{d_0, d_1, \dots, d_{t-1}}(x)$ *with* $\deg_q \Gamma(x) = t$, *where* d_i *is defined as in* (3.12), $\forall i \in [0, t-1]$. *Further, let*

$$\widetilde{s}_i = s_{n-k-1-i}^{[i-n+k+1]}, \quad \forall i \in [0, n-k-1] \tag{3.17}$$

and $\widetilde{s}(x) = \sum_{i=0}^{n-k-1} \widetilde{s}_i x^{[i]}$. *Then,*

$$\Phi(x) \equiv \Gamma(\widetilde{s}(x)) \mod x^{[n-k]}, \tag{3.18}$$

for some $\Phi(x) \in \mathbb{L}_{q^m}[x]$ *with* $\deg_q \Phi(x) < t$.

Proof. From (2.12) and (3.12), we obtain

$$\widetilde{s}_i = \sum_{l=0}^{t-1} a_l^{[i-n+k+1]} d_l. \tag{3.19}$$

The i-th coefficient of the linearized composition $\Gamma(\widetilde{s}(x))$ can then be calculated by

$$\Phi_i \overset{\text{def}}{=} \big[\Gamma(\widetilde{s}(x))\big]_i = \sum_{j=0}^{i} \Gamma_j \widetilde{s}_{i-j}^{[j]} = \sum_{j=0}^{i} \Gamma_j \left(\sum_{l=0}^{t-1} a_l^{[i-j-n+k+1]} d_l \right)^{[j]} = \sum_{l=0}^{t-1} a_l^{[i-n+k+1]} \sum_{j=0}^{i} \Gamma_j \cdot d_l^{[j]}.$$

For any $i \geq t$ this gives:

$$\Phi_i = \sum_{l=0}^{t-1} a_l^{[i-n+k+1]} \Gamma\big(d_l\big) = 0, \quad \forall i[t, n-k-1-1],$$

since $\Gamma(x)$ has all d_i, $\forall i \in [0, t-1]$, as roots and therefore $\deg_q \Phi(x) < \deg_q \Gamma(x) = t$. ∎

Based on the key equation from Theorem 3.4, we explain the different steps of the standard decoding process of Gabidulin codes in the following and summarize them in Algorithm 3.5. Similar steps have to be accomplished when we solve the row space key equation instead of the column space key equation.

Syndrome Calculation

As mentioned before, the first step of decoding Gabidulin codes is calculating the syndrome based on a parity-check matrix $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ and the received word $\mathbf{r} \in \mathbb{F}_{q^m}^{n}$:

$$\mathbf{s} = \big(s_0 \ s_1 \ \dots \ s_{n-k-1}\big) = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T \in \mathbb{F}_{q^m}^{(n-k)}.$$

Solving the Key Equation

The direct way to find $\Lambda(x)$ is to solve a linear system of equations based on the key equation (3.14). Due to (3.15) and (3.16):

$$\Omega_i = \sum_{j=0}^{i} \Lambda_j s_{i-j}^{[j]} = \sum_{j=0}^{t} \Lambda_j s_{i-j}^{[j]} = 0, \quad \forall i[t, n-k-1-1].$$

This is equivalent to the following homogeneous linear system of equations:

$$\begin{pmatrix} \Omega_t \\ \Omega_{t+1} \\ \vdots \\ \Omega_{n-k-1} \end{pmatrix} = \begin{pmatrix} s_t^{[0]} & s_{t-1}^{[1]} & \dots & s_0^{[t]} \\ s_{t+1}^{[0]} & s_t^{[1]} & \dots & s_1^{[t]} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n-k-1}^{[0]} & s_{n-k-2}^{[1]} & \dots & s_{n-k-1-t}^{[t]} \end{pmatrix} \cdot \begin{pmatrix} \Lambda_0 \\ \Lambda_1 \\ \vdots \\ \Lambda_t \end{pmatrix} = \mathbf{0}. \tag{3.20}$$

If the dimension of the solution space of (3.20) is one, then any solution of (3.20) provides the coefficients of the error span polynomial $\Lambda(x)$, defined as in (3.13), except for a scalar factor. This scalar factor does not pose a problem, since it does not change the root space. The following lemma provides a criterion to obtain the actual number of errors out of the syndrome matrix, see [Gab92, Lemma, p. 132].

Lemma 3.9 (Rank of Syndrome Matrix).
Let $\mathbf{r} = \mathbf{c}+\mathbf{e} \in \mathbb{F}_{q^m}^{n}$, where $\mathbf{c} \in \mathsf{Gab}[n,k]$ and $\mathrm{rk}(\mathbf{e}) = t \leq \lfloor (n-k)/2 \rfloor$ and let $\big(s_0 \ s_1 \ \dots \ s_{n-k-1}\big) \in \mathbb{F}_{q^m}^{n-k}$ denote the corresponding syndrome. Then, for any $u \geq t$, the $u \times (u+1)$ matrix

$$\mathbf{S}^{(u)} \overset{\text{def}}{=} \begin{pmatrix} s_u^{[0]} & s_{u-1}^{[1]} & \dots & s_0^{[u]} \\ s_{u+1}^{[0]} & s_u^{[1]} & \dots & s_1^{[u]} \\ \vdots & \vdots & \ddots & \vdots \\ s_{2u-1}^{[0]} & s_{2u-2}^{[1]} & \dots & s_{u-1}^{[u]} \end{pmatrix} \tag{3.21}$$

has full rank u if and only if $u = t$, where the i-th row of $\mathbf{S}^{(u)}$ is defined to be all-zero if $i + u > n - k - 1$, $\forall i = [0, u - 1]$.

Proof. Since there are $n - k$ non-zero syndrome coefficients, we can provide only $n - k - u$ non-zero rows of $\mathbf{S}^{(u)}$. Therefore, for $u > \lfloor (n-k)/2 \rfloor$, the matrix $\mathbf{S}^{(u)}$ has only $n - k - u < u$ non-zero rows and therefore rank less than u.

Let $a_i, d_i = 0$ for $i \geq t$. For $u \leq \lfloor (n-k)/2 \rfloor$, we can decompose $\mathbf{S}^{(u)}$ with (3.11) as follows:

$$\mathbf{S}^{(u)} = \begin{pmatrix} d_0^{[u]} & d_1^{[u]} & \dots & d_{u-1}^{[u]} \\ d_0^{[u+1]} & d_1^{[u+1]} & \dots & d_{u-1}^{[u+1]} \\ \vdots & \vdots & \ddots & \vdots \\ d_0^{[2u-1]} & d_1^{[2u-1]} & \dots & d_{u-1}^{[2u-1]} \end{pmatrix} \cdot \begin{pmatrix} a_0^{[0]} & a_0^{[1]} & \dots & a_0^{[u]} \\ a_1^{[0]} & a_1^{[1]} & \dots & a_1^{[u]} \\ \vdots & \vdots & \ddots & \vdots \\ a_{u-1}^{[0]} & a_{u-1}^{[1]} & \dots & a_{u-1}^{[u]} \end{pmatrix}.$$

Both matrices are q-Vandermonde matrices and due to Lemma 2.10, they both have full rank if and only if $d_0, d_1, \dots, d_{u-1}$ and $a_0, a_1, \dots, a_{u-1}$ are sets of elements which are linearly independent over $\mathbb{F}_q$. If $u > t$, this is not true, since $a_i, d_i = 0$ for $i \geq t$. If $u = t$ this is true and the left matrix is a square matrix of rank u and the right is a $u \times (u + 1)$ matrix of rank u. Since the first u columns of the right matrix constitute a matrix of rank u, the statement follows. ∎

Thus, Lemma 3.9 proves that for $t \leq \lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2 \rfloor$, $\mathbf{S}^{(t)}$ has full rank and the dimension of the solution space of (3.20) is one. For the algorithmic realization, we can set up $\mathbf{S}^{(u)}$ for $u = \lfloor (d-1)/2 \rfloor$ and check its rank. If the rank is not full, we decrease u by one, control the rank, and so on, until we find u such that the rank is full. Since we have to solve several linear systems of equations over $\mathbb{F}_{q^m}$, the complexity of this step is in the order of at least $\mathcal{O}(t^3) \leq \mathcal{O}(n^3)$ operations in $\mathbb{F}_{q^m}$ with Gaussian elimination (see [Rot91, Gab92]).

However, the matrix $\mathbf{S}^{(u)}$ is highly structured, it is a q-circulant matrix. The algorithm based on the LEEA from [Gab85] and the Berlekamp–Massey-like algorithms from [PT91, RP04a, RP04b, SRB11] take advantage of this structure and can therefore solve the key equation with complexity $\mathcal{O}(n^2)$ operations in $\mathbb{F}_{q^m}$.

Finding the Root Space of $\Lambda(x)$

After solving the key equation (3.20) for the coefficients of $\Lambda(x)$, we have to find a basis of the root space of $\Lambda(x)$. This basis corresponds to one possible $\mathbf{a} = (a_0 \; a_1 \; \dots \; a_{t-1})$ in the decomposition of (3.9). Finding a basis of the root space of a linearized polynomial is relatively easy due to the structure of their roots. Recall Subsection 2.2.3, where it is shown that we can find the root space of $\Lambda(x)$ by finding the right kernel of its associated evaluated matrix, i.e., for some basis $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, we have to determine:

$$\ker\Big(\mathrm{ext}_{\mathcal{B}}\big((\Lambda(\beta_0) \; \Lambda(\beta_1) \; \dots \; \Lambda(\beta_{m-1})) \big)\Big).$$

The kernel of this matrix is equivalent to $\mathrm{ext}_{\mathcal{B}}(\mathbf{a})$ of one possible $\mathbf{a}$. Thus, finding the root space of $\Lambda(x)$ involves solving a linear system of equations of size m over $\mathbb{F}_q$, which has complexity at most $\mathcal{O}(m^3)$ over $\mathbb{F}_q$. This root-finding procedure was explained in detail in [LN96, Ber84].

Determining the Error

Knowing a possible vector $\mathbf{a} \in \mathbb{F}_{q^m}^t$, we have to find the corresponding matrix $\mathbf{B} \in \mathbb{F}_q^{t \times n}$ such that $\mathbf{e} = \mathbf{a} \cdot \mathbf{B}$ as in (3.9). This is basically done in two substeps. Based on (3.11), we can set up the following system of equations,[3] which we have to solve for $\mathbf{d} = (d_0\ d_1\ \dots\ d_{t-1})$:

$$\begin{pmatrix} a_0^{[0]} & a_1^{[0]} & \dots & a_{t-1}^{[0]} \\ a_0^{[-1]} & a_1^{[-1]} & \dots & a_{t-1}^{[-1]} \\ \vdots & \vdots & \ddots & \vdots \\ a_0^{[-(n-k-1)]} & a_1^{[-(n-k-1)]} & \dots & a_{t-1}^{[-(n-k-1)]} \end{pmatrix} \cdot \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{t-1} \end{pmatrix} = \begin{pmatrix} s_0^{[0]} \\ s_1^{[-1]} \\ \vdots \\ s_{n-k-1}^{[-(n-k-1)]} \end{pmatrix}. \tag{3.22}$$

Solving this system of equations with Gaussian elimination requires complexity $\mathcal{O}(n^3)$ operations over $\mathbb{F}_{q^m}$, whereas the recursive algorithm from [Gab85] requires complexity $\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$ by using the q-Vandermonde structure of the involved matrix.

After having found $\mathbf{d}$, we determine the matrix $\mathbf{B}$ out of $d_l = \sum_{i=0}^{n-1} B_{l,i} h_i$ for all $l \in [0, t-1]$. The complexity of this calculation is negligible, since $(h_0\ h_1\ \dots\ h_{n-1})$ has rank n and we are looking for the representation of $\mathbf{d}$ over $\mathbb{F}_q$ using these linearly independent elements.

Finally, we calculate $\mathbf{e} = \mathbf{a} \cdot \mathbf{B}$ and can reconstruct $\mathbf{c} = \mathbf{r} - \mathbf{e}$. A summary of this decoding procedure is given in Algorithm 3.5. Notice that the decoding procedure most probably fails when $t > \lfloor (d-1)/2 \rfloor$.

Algorithm 3.5.
c or "decoding failure" ←DecodeGabidulin$(\mathbf{r}; \mathbf{H})$

```
Input: $\mathbf{r} = (r_0\ r_1\ \dots\ r_{n-1}) \in \mathbb{F}_{q^m}^n$ with $n \leq m$;
       parity-check matrix $\mathbf{H} = \mathrm{qvan}_{n-k}((h_0\ h_1\ \dots\ h_{n-1}))$ of $\mathsf{Gab}[n,k]$

Syndrome calculation: $\mathbf{s} \leftarrow \mathbf{r} \cdot \mathbf{H}^T \in \mathbb{F}_{q^m}^{n-k}$
if $\mathbf{s} = \mathbf{0}$ then
    Estimated codeword: $\mathbf{c} \leftarrow \mathbf{r}$
else
    Set up $\mathbf{S}^{(t)}$ as in (3.21) for $t = \lfloor (n-k)/2 \rfloor$
    while $\mathrm{rk}(\mathbf{S}^{(t)}) < t$ do
        $t \leftarrow t - 1$
        Set up $\mathbf{S}^{(t)}$ as in (3.21)
    Solve $\mathbf{S}^{(t)} \cdot \mathbf{\Lambda}^T = \mathbf{0}$ for $\mathbf{\Lambda} = (\Lambda_0\ \Lambda_1\ \dots\ \Lambda_t) \in \mathbb{F}_{q^m}^{t+1}$
    Find basis $(a_0\ a_1\ \dots\ a_{\varepsilon-1}) \in \mathbb{F}_{q^m}^{\varepsilon}$ of the root space of $\Lambda(x) = \sum_{i=0}^{t} \Lambda_i x^{[i]}$ over
    $\mathbb{F}_{q^m}$
    if $\varepsilon = t$ then
        Find $\mathbf{d} = (d_0\ d_1\ \dots\ d_{t-1}) \in \mathbb{F}_{q^m}^t$ by solving (3.22)
        Find $\mathbf{B} = \left(B_{i,j}\right)_{j \in [0,n-1]}^{i \in [0,t-1]} \in \mathbb{F}_q^{t \times n}$ such that $d_i = \sum_{j=0}^{n-1} B_{i,j} h_j$
        Estimated codeword: $\mathbf{c} \leftarrow \mathbf{r} - \mathbf{a} \cdot \mathbf{B}$
    else
        Declare decoding failure

Output: Estimated codeword $\mathbf{c} \in \mathbb{F}_{q^m}^n$ or "decoding failure"
```

We can use the fast LEEA from Subsection 3.1.4 in order to reduce the complexity of the algorithm from [Gab85]. However, this only accelerates the step of solving the key equation, whereas the overall

[3] Notice that this system of equations from (3.22) can be used for row-erasure-only correction, i.e., when $\mathbf{a}$ is known in advance due to the channel (compare Subsection 3.2.3).

complexity remains in the order of $\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$. Algorithm 3.6 in the next subsection shows how we can directly obtain the q-degree-restricted evaluation polynomial of the codeword by the LEEA and therefore, we do not have to solve (3.22).

3.2.2 A Gao-like Decoding Algorithm

In contrast to the two-step procedure of the previous subsection, Loidreau's Welch–Berlekamp-like interpolation based decoding algorithm [Loi06] directly outputs the evaluation polynomial of the estimated codeword and therefore, the second step of finding $\mathbf{d}$ and $\mathbf{B}$ is not necessary.

In this subsection, we present a new approach, where the LEEA directly provides the q-degree restricted evaluation polynomial of the estimated codeword. This algorithm is an equivalent of Gao's algorithm for decoding Reed–Solomon codes [Gao03, Fed05]. The advantage compared to [Loi06] is that we can accelerate our decoding algorithm based on the fast LEEA from Subsection 3.1.4 for q-cyclic Gabidulin codes. Compared to solving directly the key equation from Theorem 3.4 with the LEEA as in [Gab85], our advantage is that we do not need the (computationally intensive) step of finding $\mathbf{B}$.

Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} \in \mathsf{Gab}[n,k]$ and $t = \mathrm{rk}(\mathbf{e})$, denote the received word and $r(x) = \sum_{i=0}^{n-1} r_i x^{[i]}$ its associated linearized polynomial. Let $\mathcal{G} = \{g_0, g_1, \dots, g_{n-1}\}$ consist of n elements from $\mathbb{F}_{q^m}$, which are linearly independent over $\mathbb{F}_q$ and which are used as evaluation points of the $\mathsf{Gab}[n,k]$ code as in Definition 2.16. Let $\widehat{r}(x) \in \mathbb{L}_{q^m}[x]$ denote the unique linearized polynomial of q-degree less than n such that $\widehat{r}(g_i) = r_i$, $\forall i \in [0, n-1]$, holds. This polynomial can be calculated as follows:

$$\widehat{r}(x) = \sum_{i=0}^{n-1} r_i \cdot \frac{L_i(x)}{L_i(g_i)}, \tag{3.23}$$

where $L_i(x)$ denotes the *i-th linearized Lagrange basis polynomial* of q-degree $n-1$, see also [SK07], which is defined as the minimal subspace polynomial of $\mathcal{G} \setminus g_i = \{g_0, \dots, g_{i-1}, g_{i+1}, \dots, g_{n-1}\}$, i.e.:

$$L_i(x) = M_{\mathcal{G}\setminus g_i}(x) = \prod_{B_0=0}^{q-1} \cdots \prod_{B_{i-1}=0}^{q-1} \cdot \prod_{B_{i+1}=0}^{q-1} \cdots \prod_{B_{n-1}=0}^{q-1} \Big(x - \sum_{j=0, j\neq i}^{n-1} B_j g_j\Big). \tag{3.24}$$

Therefore,

$$\frac{L_i(g_j)}{L_i(g_i)} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{else.} \end{cases}$$

It is important to remark that for the case of $n \mid m$ and when $\mathcal{G}$ is a normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, then $\widehat{r}(x)$ is the q-transform of $r(x)$ as in Definition 2.12. This fact is used in Subsection 3.2.4 to accelerate our decoding algorithm for q-cyclic Gabidulin codes. Here, we describe the decoding algorithm in general for any $\mathsf{Gab}[n,k]$ code by using linearized Lagrange interpolation and a transformed key equation.

Theorem 3.6 (Transformed Key Equation).
Let $\mathbf{r} = (r_0\ r_1\ \dots\ r_{n-1})$ be given and let $\widehat{r}(x)$ be defined as in (3.23)*, where $\mathcal{G} = \{g_0, g_1, \dots, g_{n-1}\}$ is a set of n elements, which are linearly independent over $\mathbb{F}_q$. Let $\mathbf{a} = (a_0\ a_1\ \dots\ a_{t-1})$ denote a basis of the column space of $\mathbf{r} - (f(g_0)\ f(g_1)\ \dots\ f(g_{n-1}))$, for some $f(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q f(x) < k$.*

Then, the linearized error span polynomial $\Lambda(x)$, defined as in (3.13)*, satisfies the transformed key equation:*

$$\Lambda\big(\widehat{r}(x) - f(x)\big) \equiv 0 \mod M_{\mathcal{G}}(x). \tag{3.25}$$

Proof. Due to the definition of $\Lambda(x)$ in (3.13) and the definition of the linearized Lagrange basis polynomial (3.24):

$$\Lambda\big(\widehat{r}(g_i) - f(g_i)\big) = \Lambda\big(r_i - c_i\big) = \sum_{j=0}^{t-1} B_{j,i}\Lambda\big(a_j\big) = 0, \quad \forall i \in [0, n-1],$$

where $B_{i,j} \in \mathbb{F}_q$ for all i, j and $\mathbf{c} = (c_0\ c_1\ \dots\ c_{n-1}) \in \mathsf{Gab}[n,k]$.

Hence, for any $G_0, G_1, \dots, G_{n-1} \in \mathbb{F}_q$:

$$\Lambda\big(\widehat{r}(x) - f(x)\big)\Big|_{x=\sum\limits_{i=0}^{n-1} G_i g_i} = \sum_{i=0}^{n-1} G_i \Lambda\big(r_i - c_i\big) = 0, \quad \forall G_0, G_1, \dots, G_{n-1} \in \mathbb{F}_q.$$

Therefore, $\big(x - \sum_{i=0}^{n-1} G_i g_i\big)$ divides (in the usual sense) $\Lambda\big(\widehat{r}(x) - f(x)\big)$ for any $G_0, G_1, \dots, G_{n-1} \in \mathbb{F}_q$. This implies that

$$\Lambda\big(\widehat{r}(x) - f(x)\big) \equiv 0 \mod M_{\mathcal{G}}(x).$$

∎

For $n = m$, the minimal subspace polynomial is $M_{\mathcal{G}}(x) = (x^{[m]} - x)$ and due to Theorem 3.6, $\Lambda\big(\widehat{r}(x) - f(x)\big) \equiv 0 \mod (x^{[m]} - x)$. This special case was also proven by Silva and Kschischang in [SK09a, Sil09, Theorem 5] using properties of the q-transform. Moreover, for $n \mid m$, we can choose $\mathcal{G} = \{g_0, g_1, \dots, g_{n-1}\}$ such that it is a basis in $\mathbb{F}_{q^m}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ and then, $M_{\mathcal{G}}(x) = (x^{[n]} - x)$, since then the unique subfield $\mathbb{F}_{q^n}$ of $\mathbb{F}_{q^m}$ consists precisely of the roots of $(x^{[n]} - x)$ [LN96, p. 50].

The problem of solving the transformed key equation can be stated as follows.

Problem 3.2 (Solving Transformed Key Equation).
Let $\widehat{r}(x) \in \mathbb{L}_{q^m}[x]$ as in (3.23) for $\mathcal{G} = \{g_0, g_1, \dots, g_{n-1}\}$ with $\operatorname{rk}(g_0\ g_1\ \dots\ g_{n-1}) = n$ be given, where $\mathbf{r} \in \mathbb{F}_{q^m}^n$ denotes the received word. Let $d_R(\mathbf{r}, \mathbf{c}) = \operatorname{rk}(\mathbf{r} - \mathbf{c}) \leq \lfloor (n-k)/2 \rfloor$ for some codeword $\mathbf{c} = f(\mathbf{g}) \in \mathsf{Gab}[n,k]$.

Find $\Lambda(x) \in \mathbb{L}_{q^m}[x]$ of q-degree $t \leq \lfloor (n-k)/2 \rfloor$ and $f(x)$ of q-degree less than k such that $\Lambda(x) = M_{\mathcal{A}}(x)$ for some set $\mathcal{A} = \{a_0, a_1, \dots, a_{t-1}\}$ with $\operatorname{rk}(a_0\ a_1\ \dots\ a_{t-1}) = t$ and such that

$$\Lambda\big(\widehat{r}(x) - f(x)\big) \equiv 0 \mod M_{\mathcal{G}}(x). \tag{3.26}$$

We solve Problem 3.2 with a generalization of *Gao's algorithm* [Gao03] for linearized polynomials, given in Algorithm 3.6. The transformed key equation (3.25), Theorem 3.6, can be rewritten with a polynomial $\Omega(x) \in \mathbb{L}_{q^m}[x]$:

$$\Lambda\big(\widehat{r}(x) - f(x)\big) = -\Omega\big(M_{\mathcal{G}}(x)\big) \equiv 0 \mod M_{\mathcal{G}}(x),$$

and thus,

$$\Lambda\big(f(x)\big) = \Omega\big(M_{\mathcal{G}}(x)\big) + \Lambda\big(\widehat{r}(x)\big). \tag{3.27}$$

Recall that $\textsc{RightLEEA}\big(a(x); b(x); d_{stop}\big)$ with $\deg_q a(x) \geq \deg_q b(x)$ (Algorithm 2.3) returns unique linearized polynomials $r_{out}(x)$, $u_{out}(x)$ and $v_{out}(x)$ such that $\deg_q r_{out}(x) < d_{stop}$ and

$$r_{out}(x) = v_{out}\big(a(x)\big) + u_{out}\big(b(x)\big). \tag{3.28}$$

If we compare (3.27) and (3.28), we obtain the idea for Algorithm 3.6: we run the LEEA with the input polynomials $a(x) \leftarrow M_{\mathcal{G}}(x)$ and $b(x) \leftarrow \widehat{r}(x)$ and the stopping degree $d_{stop} \leftarrow \lfloor (n-k)/2 \rfloor + k =$

$\lfloor (n+k)/2 \rfloor$. If there exists a codeword $\mathbf{c} \in \mathsf{Gab}[n,k]$ such that $\mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \lfloor (n-k)/2 \rfloor$, Theorem 3.7 proves that the remainder $r_{out}(x)$ is equal to $a \cdot \Lambda\big(f(x)\big)$ and the auxiliary polynomials are $v_{out}(x) = a \cdot \Omega(x)$ and $u_{out}(x) = a \cdot \Lambda(x)$ for some scalar $a \in \mathbb{F}_{q^m}$. Hence, we can find $f(x)$ by a left linearized division of $r_{out}(x)$ by $u_{out}(x)$. If no such codeword exists, the remainder of this linearized division is unequal to zero, and we declare a *decoding failure*.

This idea is given in pseudo-code in Algorithm 3.6 and its correctness is proven in Theorem 3.7. Our proof does not use the same strategy as in [Gao03], since Gao's proof does not work directly for linearized polynomials, instead we use some properties of the transformed key equation.

Theorem 3.7 (Correctness of Algorithm 3.6).
Let $\mathbf{r} \in \mathbb{F}_{q^m}^n$ and $\mathbf{g} = (g_0\ g_1\ \dots\ g_{n-1})$ with $\mathrm{rk}(\mathbf{g}) = n$ be given. If $t = \mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \lfloor (n-k)/2 \rfloor$ for a codeword $\mathbf{c} = f\big(\mathbf{g}\big) \in \mathsf{Gab}[n,k]$, then Algorithm 3.6 solves Problem 3.2. Hence, it returns $f(x)$ of $\deg_q f(x) < k$ and $\Lambda(x) = M_{\mathcal{A}}(x)$, where $\mathcal{A} = \{a_0, a_1, \dots, a_{t-1}\} \in \mathbb{F}_{q^m}$, such that $(\mathbf{r}-\mathbf{c}) = (a_0\ a_1\ \dots\ a_{t-1}) \cdot \mathbf{B}$ as in (3.9).

If there is no such codeword, Algorithm 3.6 returns "decoding failure".

Algorithm 3.6.
$f(x); \Lambda(x)$ **or** "decoding failure" $\leftarrow$ DecodeGaoGabidulin$\big(\mathbf{r}; g_0, g_1, \dots, g_{n-1}\big)$

```
Input: $\mathbf{r} = (r_0\ r_1\ \dots\ r_{n-1}) \in \mathbb{F}_{q^m}^n$ with $n \leq m$;
       $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$, linearly independent over $\mathbb{F}_q$
Calculate $\widehat{r}(x) \leftarrow \sum_{i=0}^{n-1} r_i \cdot \frac{L_i(x)}{L_i(g_i)}$ as in (3.23)
$r_{out}(x); u_{out}(x); v_{out}(x) \leftarrow$ RightLEEA$\big(M_{\mathcal{G}}(x); \widehat{r}(x); d_{stop} = \lfloor (n+k)/2 \rfloor\big)$ with
  Algorithm 2.3
$f(x); r(x) \leftarrow$ LeftDiv$\big(r_{out}(x); u_{out}(x)\big)$ with Algorithm 2.2
if $r(x) = 0$ then
    Output: Estimated evaluation polynomial $f(x)$ with $\deg_q f(x) < k$;
            Estimated error span polynomial $\Lambda(x) \leftarrow u_{out}(x)$
else
    Output: "Decoding failure"
```

Proof. First, we show that, given $\widehat{r}(x)$ and $M_{\mathcal{G}}(x)$, the transformed key equation (3.26) has a unique solution for $\Lambda(x)$ of minimal q-degree and $f(x)$ of q-degree less than k. Second, we show that Algorithm 3.6 finds this solution. Third, we prove if the rank distance of $\mathbf{r}$ to any codeword $\mathbf{c}$ is greater than $\lfloor (n-k)/2 \rfloor$, there is no $f(x)$ of degree less than k, fulfilling the transformed key equation.

1.) We assume that there is a codeword $\mathbf{c} \in \mathsf{Gab}[n,k]$ such that $t = \mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \lfloor (n-k)/2 \rfloor$ and we prove that there is a unique solution of (3.26) (except for a constant factor), where $\deg_q f(x) < k$ and $\deg_q \Lambda(x) = t$. This follows either from reducing the transformed key equation (3.26) to the "classical" key equation (3.14) (which has a unique solution due to Lemma 3.9) or directly from [Loi06, Proposition 2].

2.) Now, we show that the result of the LEEA is a solution of the transformed key equation. Due to (3.28), the RightLEEA$\big(M_{\mathcal{G}}(x); \widehat{r}(x); d_{stop} = \lfloor (n+k)/2 \rfloor\big)$ outputs unique polynomials such that:

$$r_{out}(x) = u_{out}\big(\widehat{r}(x)\big) \mod M_{\mathcal{G}}(x). \tag{3.29}$$

On the one hand, $r_{out}(x) = r^{(i-1)}(x)$ is the first remainder in the iterations of the LEEA of q-degree less than $\lfloor (n+k)/2 \rfloor$ and $u_{out}(x)$ is such that $\deg_q u_{out}(x) = \deg_q a(x) - \deg_q r^{(i-2)}(x) \leq \lfloor (n-k)/2 \rfloor$. On the other hand, the transformed key equation can be rewritten as

$$\Lambda\big(f(x)\big) \equiv \Lambda\big(\widehat{r}(x)\big) \mod M_{\mathcal{G}}(x), \tag{3.30}$$

with $\deg_q \Lambda(x) = t \leq \lfloor (n-k)/2 \rfloor$ and $\deg_q f(x) < k$. In [Loi06, Propositions 1 and 2] it was shown that for $t \leq \lfloor (n-k)/2 \rfloor$, the polynomials $\Lambda(x)$ and $f(x)$ provide a solution to (3.30) if and only if $\Lambda(x)$ and some $\Phi(x)$ of q-degree less than $\lfloor (n+k)/2 \rfloor$ provide a solution to

$$\Phi(x) \equiv \Lambda\big(\widehat{r}(x)\big) \mod M_{\mathcal{G}}(x). \tag{3.31}$$

If we compare (3.29) and (3.31), it becomes clear that the output of the LEEA is such a solution, including the degree constraints. [Loi06, Proposition 2] shows that there exists only one pair of polynomials $\Phi(x), \Lambda(x)$ such that (3.31) is fulfilled with the required degree constraints. Hence, there is also only one pair of polynomials $\Lambda(x), f(x)$ such that (3.30) and its degree constraints are fulfilled and we find exactly this solution by the LEEA.

3.) Let $t > \lfloor (n-k)/2 \rfloor$, and assume nonetheless that Algorithm 3.6 returns $f'(x)$ with $\deg_q f'(x) < k$. The following holds for the output of the LEEA:

$$u_{out}\big(\widehat{r}(x)\big) \equiv u_{out}\big(f'(x)\big) \mod M_{\mathcal{G}}(x),$$

and hence also

$$u_{out}\big(\widehat{r}(g_j) - f'(g_j)\big) = u_{out}\big(r_j - c'_j\big) = 0, \qquad \forall j \in [0, n-1].$$

Due to the stopping condition, $\deg_q u_{out}(x) \leq \lfloor (n-k)/2 \rfloor$ and hence, the dimension of the root space of $u_{out}(x)$ is at most $\lfloor (n-k)/2 \rfloor$. Therefore, for $j \in [0, n-1]$, there exists a $\mathbf{c}'$ such that there are at most $\lfloor (n-k)/2 \rfloor$ linearly independent $r_j - c'_j$ and thus, $\mathrm{rk}(\mathbf{r} - \mathbf{c}') \leq \lfloor (n-k)/2 \rfloor$. This is a contradiction to the assumption and thus, Algorithm 3.6 fails when the rank distance of $\mathbf{r}$ is greater than $\lfloor (n-k)/2 \rfloor$ to any codeword. ∎

With straight-forward implementation, Algorithm 3.6 has complexity $\mathcal{O}(n^2)$ operations in $\mathbb{F}_{q^m}$.

3.2.3 Error-Erasure Decoding

Applications like random linear network coding might provide additional information about the occurred error. Such information can be used to declare *erasures* and thus, to increase decoding performance. In comparison to classical erasure decoders in Hamming metric, we distinguish two types of erasures in rank metric: row erasures and column erasures.

We thereby consider the most general form of such row and column erasures as in [SKK08, GP08] and show how the additional information can be incorporated into our decoding algorithm from Subsection 3.2.2. We consider only the case $n = m$ in this subsection. On the one hand this helps to simplify the notations, but on the other hand this is since Lemma 3.10 only holds for $n = m$ and it is not clear how to extend it for arbitrary $n < m$.

We apply the q-transform in this subsection, but all considerations except Lemma 3.10 hold straight-forward for linearized Lagrange interpolation polynomials as in the previous section. The presented error-erasure decoder is able to reconstruct a codeword of a Gab$[n, k]$ code over $\mathbb{F}_{q^m}$ for $n = m$ with asymptotic complexity $\mathcal{O}(n^2)$ operations over $\mathbb{F}_{q^m}$ if

$$2t + \varrho + \gamma \leq d - 1 = n - k, \tag{3.32}$$

where t denotes the rank of the error, ϱ the rank of the row erasures and γ the rank of the column erasures. Compared to the approaches from [GP08, SKK08, LSC13], again the advantage of our approach is that the acceleration of the LEEA (Algorithm 3.4) can be used.

In the following, we explain our notations of row/column erasures, derive a generalization of the transformed key equation and show how our decoder can be modified to incorporate also erasures.

Row and Column Erasures and Generalized Transformed Key Equation

Consider a $\mathsf{Gab}[n,k]$ code over $\mathbb{F}_{q^m}$ with $n = m$, defined in its q-cyclic form as in Lemma 2.15 with $\mathbf{g} = (g_0\ g_1\ \dots\ g_{n-1}) = (\beta^{\perp[0]}\ \beta^{\perp[1]}\ \dots\ \beta^{\perp[n-1]})$ and the corresponding parity-check matrix with $\mathbf{h} = (\beta^{[k]}\ \beta^{[k+1]}\ \dots\ \beta^{[k+n-1]})$ from Lemma 2.16.

The additional side information of the channel is assumed to be given in form of:

- ϱ row erasures (in [SKK08] called “deviations”) and
- γ column erasures (in [SKK08] called “erasures”),

such that the received matrix can be rewritten in accordance to (3.9) by

$$\mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{r}) = \mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{c}) + \underbrace{\mathbf{A}^{(R)}\mathbf{B}^{(R)} + \mathbf{A}^{(C)}\mathbf{B}^{(C)} + \mathbf{A}^{(E)}\mathbf{B}^{(E)}}_{\overset{\mathrm{def}}{=}\mathbf{E}} \in \mathbb{F}_q^{m\times n}, \tag{3.33}$$

where $\mathbf{A}^{(R)} \in \mathbb{F}_q^{m\times\varrho}$, $\mathbf{B}^{(R)} \in \mathbb{F}_q^{\varrho\times n}$, $\mathbf{A}^{(C)} \in \mathbb{F}_q^{m\times\gamma}$, $\mathbf{B}^{(C)} \in \mathbb{F}_q^{\gamma\times n}$, $\mathbf{A}^{(E)} \in \mathbb{F}_q^{m\times t}$, $\mathbf{B}^{(E)} \in \mathbb{F}_q^{t\times n}$, and $\mathbf{A}^{(R)}$ and $\mathbf{B}^{(C)}$ are known to the receiver. Further, t denotes the number of errors without side information. This decomposition is also shown in Figure 3.1.

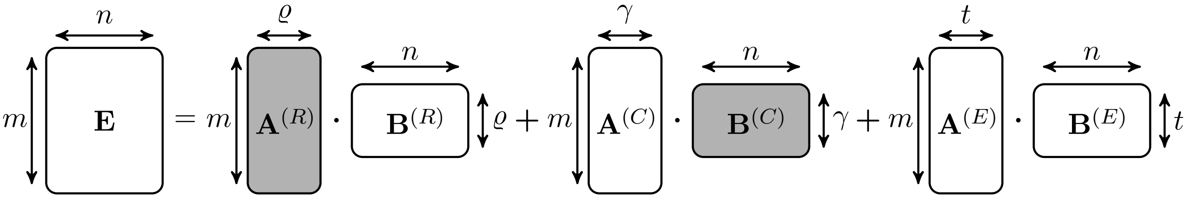


Figure 3.1. Illustration of row erasures, column erasures and (full) errors in rank metric. The known matrices (given by the channel) are filled with gray.

Similar to (3.9), we can represent the error word as a vector as follows:

$$\mathbf{e} = \mathbf{r} - \mathbf{c} = \mathbf{a}^{(R)}\mathbf{B}^{(R)} + \mathbf{a}^{(C)}\mathbf{B}^{(C)} + \mathbf{a}^{(E)}\mathbf{B}^{(E)} \overset{\mathrm{def}}{=} \mathbf{e}^{(R)} + \mathbf{e}^{(C)} + \mathbf{e}^{(E)} \in \mathbb{F}_{q^m}^n, \tag{3.34}$$

where $\mathbf{a}^{(R)} \in \mathbb{F}_{q^m}^{\varrho}$, $\mathbf{a}^{(C)} \in \mathbb{F}_{q^m}^{\gamma}$ and $\mathbf{a}^{(E)} \in \mathbb{F}_{q^m}^{t}$ and let $e^{(R)}(x)$, $e^{(C)}(x)$ and $e^{(E)}(x) \in \mathbb{L}_{q^m}[x]$ denote the linearized polynomials associated to $\mathbf{e}^{(R)}$, $\mathbf{e}^{(C)}$ and $\mathbf{e}^{(E)}$.

Figure 3.2 illustrates the simplest case of row erasures, where the known matrix $\mathbf{A}^{(R)}$ has only ϱ non-zero elements. This case was considered in earlier publications as [GPT91b, GP03, RP04b], but in the general model (introduced in [GP08, SKK08]), $\mathbf{A}^{(R)}$ can be any arbitrary matrix of rank ϱ over $\mathbb{F}_q$. The notation of (generalized) row erasures and codes correcting such erasures were also shown in [RS96] using a different terminology.

Similar to Subsection 3.2.1, we use the known matrix $\mathbf{B}^{(C)}$ in order to calculate:

$$d_i^{(C)} = \sum_{j=0}^{n-1} B_{i,j}^{(C)} g_j^{\perp} = \sum_{j=0}^{n-1} B_{i,j}^{(C)} \beta^{[j]}, \quad \forall i \in [0, \gamma-1]. \tag{3.35}$$

However, it is important to note that this definition differs from (3.12) by using $g_i^\perp = \beta^{[i]}$ here and $h_i = \beta^{[k+i]}$ in (3.12).

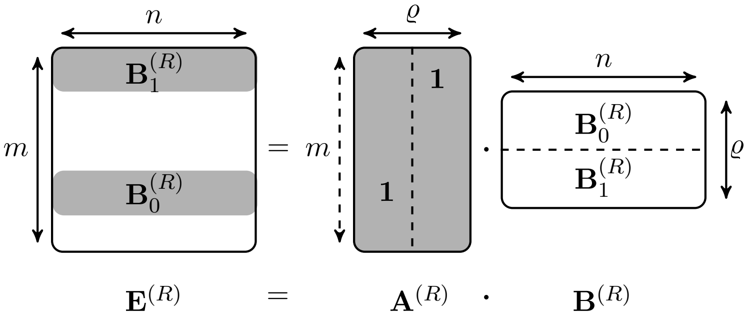


Figure 3.2. Illustration of the simplest case of row erasures, where $\varrho = 2$ and the known matrix $\mathbf{A}^{(R)}$ has only two non-zero entries.

As before, we define error span polynomials, but now three different types. In particular, we define $\Gamma^{(C)}(x)$, $\Lambda^{(R)}(x)$ and $\Lambda^{(E)}(x)$ as the linearized polynomials of smallest q-degree such that:

$$\begin{aligned} \Gamma^{(C)}\big(d_i^{(C)}\big) &= 0, \quad \forall i \in [0, \gamma-1], \\ \Lambda^{(R)}\big(a_i^{(R)}\big) &= 0, \quad \forall i \in [0, \varrho-1], \\ \Lambda^{(E)}\big(\Lambda^{(R)}(a_i^{(E)})\big) &= 0, \quad \forall i \in [0, t-1]. \end{aligned} \tag{3.36}$$

Therefore, $\Gamma^{(C)}(x) = M_{d_0^{(C)}, d_1^{(C)}, \dots, d_{\gamma-1}^{(C)}}(x)$ and $\Lambda^{(R)}(x) = M_{a_0^{(R)}, a_1^{(R)}, \dots, a_{\varrho-1}^{(R)}}(x)$ can be calculated at the beginning of the decoding process since $\mathbf{B}^{(C)}$ and $\mathbf{a}^{(R)}$ are known.

Let $\overline{p}(x) = \sum_{i=0}^{m-1} \overline{p}_i x^{[i]}$ denote the *full q-reverse linearized polynomial* of $p(x) \in \mathbb{L}_{q^m}[x]$, defined by the coefficients $\overline{p}_i = p_{-i \bmod m}^{[i]}$ as in [SK09a, Sil09]. The following lemma shows that for $n = m$, the full q-reverse is related to the transpose of the associated evaluated matrix of $p(x)$ (see Subsection 2.2.3).

Lemma 3.10 (Evaluated Matrix of q-Reverse [Sil09, Lemma 6.3]).
Let $p(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q p(x) < m$ and its full q-reverse $\overline{p}(x)$ with the coefficients $\overline{p}_i = p_{-i \bmod m}^{[i]}$, for $i \in [0, m-1]$, be given. Let $\mathcal{A} = \{\alpha_0, \alpha_1, \dots, \alpha_{m-1}\}$ and $\mathcal{B} = \{\beta_0, \beta_1, \dots, \beta_{m-1}\}$ be bases of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and let $\mathcal{A}^\perp = \{\alpha_0^\perp, \alpha_1^\perp, \dots, \alpha_{m-1}^\perp\}$ and $\mathcal{B}^\perp = \{\beta_0^\perp, \beta_1^\perp, \dots, \beta_{m-1}^\perp\}$ denote their dual bases. Let

$$(p(\alpha_0)\ p(\alpha_1)\ \dots\ p(\alpha_{m-1})) = (\beta_0\ \beta_1\ \dots\ \beta_{m-1}) \cdot \mathbf{P},$$

where $\mathbf{P} \in \mathbb{F}_q^{m \times m}$. Then,

$$\Big(\overline{p}(\beta_0^\perp)\ \overline{p}(\beta_1^\perp)\ \dots\ \overline{p}(\beta_{m-1}^\perp)\Big) = (\alpha_0^\perp\ \alpha_1^\perp\ \dots\ \alpha_{m-1}^\perp) \cdot \mathbf{P}^T.$$

Based on the previous lemma, we can establish the *generalized* transformed key equation, incorporating errors and row/column erasures.

Theorem 3.8 (Generalized Transformed Key Equation).
Let $\mathbf{r} = \mathbf{c} + \mathbf{e}$, for $\mathbf{c} \in \mathsf{Gab}[n, k]$ over $\mathbb{F}_{q^m}$ with $n = m$, be the given received word and let $\widehat{r}(x) = f(x) + \widehat{e}(x)$ be its q-transform as in Definition 2.12. Let $\Gamma^{(C)}(x)$, $\Lambda^{(R)}(x)$ and $\Lambda^{(E)}(x)$ be defined as in (3.36).

Then, these polynomials satisfy the generalized transformed key equation:

$$\Lambda^{(E)}\Big(\Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma^{(C)}}(x))\big)\Big) \equiv 0 \mod (x^{[m]} - x). \tag{3.37}$$

Proof. The proof can be found in Appendix A.2. ∎

Error-Erasure Decoding with Gao-like Algorithm

The idea of our error-erasure decoding algorithm is to modify the transformed received word in the beginning of the decoding process as follows:

$$\widehat{y}(x) \overset{\text{def}}{=} \Lambda^{(R)}\big(\widehat{r}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) \mod (x^{[m]} - x),$$

which can immediately be calculated since all polynomials on the RHS are known from the channel. This modified transformed received word can be rewritten as

$$\widehat{y}(x) = \underbrace{\Lambda^{(R)}\big(f(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)}_{\deg_q < k+\gamma+\varrho} + \Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma}^{(C)}(x^{[\gamma]}))\big) \mod (x^{[m]} - x), \tag{3.38}$$

where $f(x)$ with $\deg_q f(x) < k$ is the evaluation polynomial of the transmitted codeword such that $\mathbf{c} = f(\mathbf{g}) \in \mathsf{Gab}[n,k]$.

The idea is now to pass the modified transformed received word $\widehat{y}(x)$ from (3.38) (instead of $\widehat{r}(x)$) to Algorithm 3.6. The polynomial $\Lambda^{(R)}\big(f(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$ on the RHS of (3.38) has q-degree less than $k+\varrho+\gamma$ and can therefore be seen as the evaluation polynomial of a $\mathsf{Gab}[n, k+\varrho+\gamma]$ codeword. The polynomial $\Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$ is called *modified transformed error* in the following and it is shown in Lemma 3.11 that its evaluation has rank at most t. Therefore, error-erasure decoding of a $\mathsf{Gab}[n,k]$ is reduced to errors-only decoding of a $\mathsf{Gab}[n, k+\varrho+\gamma]$ code. In principle, any error decoding algorithm for Gabidulin codes can now be applied.

Lemma 3.11 (Rank of Modified Error Word).
Let $e^{(RC)}(x)$ (respectively $\mathbf{e}^{(RC)} \in \mathbb{F}_{q^m}^n$) with $n = m$ denote the inverse q-transform of the modified transformed error word $\Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$. Further, let $\mathbf{e}^{(E)}$ be defined as in (3.34) *with* $\operatorname{rk}(\mathbf{e}^{(E)}) = t$. *Then,*

$$\operatorname{rk}\left(\mathbf{e}^{(RC)}\right) \leq \operatorname{rk}\left(\mathbf{e}^{(E)}\right) = t.$$

Proof. Due to the proof of Theorem 3.8, $\Lambda^{(R)}\big((\widehat{e}^{(R)}(x) + \widehat{e}^{(C)}(x)) \circ \overline{\Gamma^{(C)}}(x)\big) \equiv 0 \mod (x^{[m]} - x)$ holds and therefore also $\Lambda^{(R)}\big((\widehat{e}^{(R)}(x) + \widehat{e}^{(C)}(x)) \circ \overline{\Gamma^{(C)}}(x^{[\gamma]})\big) \equiv 0 \mod (x^{[m]} - x)$ holds and we obtain

$$\Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) \equiv \Lambda^{(R)}\big(\widehat{e}^{(E)}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) \mod (x^{[m]} - x).$$

Let $\mathbf{G} = \left(G_{i,j}\right)_{j\in[0,m-1]}^{i\in[0,m-1]} \in \mathbb{F}_q^{m\times m}$ be such that $\overline{\Gamma^{(C)}}(g_j^{[\gamma]}) = \sum_{i=0}^{m-1} G_{i,j} g_i$ and thus, $\forall j \in [0, m-1]$:

$$e_j^{(RC)} = \Lambda^{(R)}\big(\widehat{e}(\overline{\Gamma^{(C)}}(g_j^{[\gamma]}))\big) = \Lambda^{(R)}\big(\widehat{e}^{(E)}(\overline{\Gamma^{(C)}}(g_j^{[\gamma]}))\big) = \sum_{i=0}^{m-1} G_{i,j} \Lambda^{(R)}\big(\widehat{e}^{(E)}(g_i)\big).$$

Hence,

$$\mathbf{e}^{(RC)} = \Big(\Lambda^{(R)}\big(\widehat{e}^{(E)}(g_0)\big)\ \Lambda^{(R)}\big(\widehat{e}^{(E)}(g_1)\big)\ \dots\ \Lambda^{(R)}\big(\widehat{e}^{(E)}(g_{m-1})\big)\Big) \cdot \mathbf{G}.$$

Due to Lemma 2.12, $\big(\Lambda^{(R)}(\widehat{e}^{(E)}(g_0))\ \Lambda^{(R)}(\widehat{e}^{(E)}(g_1))\ \dots\ \Lambda^{(R)}(\widehat{e}^{(E)}(g_{m-1}))\big)$ lies in the same row space as $\mathbf{e}^{(E)} = \big(\widehat{e}^{(E)}(g_0)\ \widehat{e}^{(E)}(g_1)\ \dots\ \widehat{e}^{(E)}(g_{m-1})\big)$ and hence, has rank at most t. The multiplication with $\mathbf{G}$ does not increase the rank. ∎

Since $\deg_q \Lambda^{(R)}\big(f(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) < k + \varrho + \gamma$ and since $\mathrm{rk}(\mathbf{e}^{(RC)}) \leq t$, we can call Algorithm 3.6 by DecodeGaoGabidulin$\big(\widehat{y}(x); \mathbf{g}\big)$ and have to skip Step 1. This outputs $\Lambda^{(R)}\big(f(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$ and $\Lambda^{(E)}(x)$ if

$$\mathrm{rk}\left(\mathbf{e}^{(RC)}\right) \leq t \leq \frac{n - \deg_q\big(\Lambda^{(R)}(f(\overline{\Gamma^{(C)}}(x^{[\gamma]})))\big) + 1}{2} = \frac{n-k-\varrho-\gamma}{2} = \frac{d-1-\varrho-\gamma}{2}.$$

This follows directly from Theorem 3.7. After calling Algorithm 3.6, we have to divide the output $\Lambda^{(R)}\big(f(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$ from the left/right by the known error span polynomials $\Lambda^{(R)}(x)$ and $\overline{\Gamma^{(C)}}(x^{[\gamma]})$ in order to obtain $f(x)$. Thus, with these modifications, we can use Algorithm 3.6 for error-erasure decoding. This error-erasure decoding principle is shown in Algorithm 3.7 in the next section together with an acceleration based on the q-transform.

3.2.4 Fast Error-Erasure Decoding of q-cyclic Gabidulin Codes

For q-cyclic Gabidulin codes, linearized Lagrange interpolation (see e.g. (3.23)) coincides with the q-transform (see Definition 2.12). A q-cyclic $\mathsf{Gab}[n,k]$ Gabidulin code over $\mathbb{F}_{q^m}$ exists for any n dividing m as shown in Corollary 2.2. In this section, we show how to accelerate the error-erasure approach from the previous section for $n = m$ using the q-transform. As before, the idea is based on the comparison of the output of RightLEEA$\big(x^{[m]} - x; \widehat{y}(x); d_{stop}\big)$, which is:

$$r_{out}(x) \equiv u_{out}\big(\widehat{y}(x)\big) \mod (x^{[m]} - x),$$

where $\widehat{y}(x) = \Lambda^{(R)}(\widehat{r}(\overline{\Gamma^{(C)}}(x^{[\gamma]})))$, and the generalized transformed key equation, shifted by γ (compare Theorem 3.8):

$$\Lambda^{(E)}\big(\Lambda^{(R)}(f(\overline{\Gamma^{(C)}}(x^{[\gamma]})))\big) \equiv \Lambda^{(E)}\big(\Lambda^{(R)}(\widehat{r}(\overline{\Gamma^{(C)}}(x^{[\gamma]})))\big) = \Lambda^{(E)}\big(\widehat{y}(x)\big) \mod (x^{[m]} - x),$$

where the q-degree of the LHS is less than $t + \varrho + k + \gamma$. Similar to Subsection 3.2.2, the decoding algorithm follows from this comparison and is given in Algorithm 3.7.

Theorem 3.9 (Correctness and Complexity of Algorithm 3.7).
Let $\mathbf{r} \in \mathbb{F}_{q^m}^n$, a low-complexity normal basis $\boldsymbol{\beta} = (\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[m-1]})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ with $\mathrm{comp}(\mathbf{T}_m) \sim \mathcal{O}(m)$, the vector $\mathbf{a}^{(R)} \in \mathbb{F}_{q^m}^{\varrho}$ and the matrix $\mathbf{B}^{(C)} \in \mathbb{F}_q^{\gamma \times n}$ as in (3.34) *be given.*

If a codeword $\mathbf{c} = f\big(\mathbf{g}\big) \in \mathsf{Gab}[n,k]$ and the corresponding decomposition of $\mathbf{r} - \mathbf{c}$ from (3.34) *satisfy $2t + \varrho + \gamma \leq n - k$, then Algorithm 3.7 returns $f(x)$ of $\deg_q f(x) < k$. If there is no such codeword, Algorithm 3.7 returns "decoding failure".*

If there is a fast (right and left) linearized division such that $\mathcal{D}(m) = \mathcal{M}_m(m)$, then error-only decoding can be accomplished with complexity $\mathcal{O}(m^3 \log m)$ operations over $\mathbb{F}_q$ and if, in addition, there is also a method to calculate the minimal subspace polynomial with complexity $\mathcal{M}_m(m)$, then error-erasure decoding can be accomplished by Algorithm 3.7 with complexity $\mathcal{O}(m^3 \log m)$ operations over $\mathbb{F}_q$.

Else, the overall complexity is $\mathcal{O}(m^2)$ operations over $\mathbb{F}_{q^m}$.

Algorithm 3.7.
$f(x)$ **or** "decoding failure" $\leftarrow$ FastDecodeGaoGabidulin$\big(r(x); \mathbf{a}^{(R)}; \mathbf{B}^{(C)}; \boldsymbol{\beta}\big)$

```
Input: $r(x) \in \mathbb{L}_{q^m}[x]$ with $\deg_q r(x) < n = m$;
       $\mathbf{a}^{(R)} = (a_0^{(R)} \; a_1^{(R)} \; \dots \; a_{\varrho-1}^{(R)}) \in \mathbb{F}_{q^m}^{\varrho}$;
       $\mathbf{B}^{(C)} = \big(B_{i,j}\big)_{j \in [0,n-1]}^{i \in [0,\gamma-1]} \in \mathbb{F}_q^{\gamma \times n}$;
       Ordered normal basis $\boldsymbol{\beta} = (\beta^{[0]} \; \beta^{[1]} \; \dots \; \beta^{[m-1]})$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$
1  Calculate $d_i^{(C)} \leftarrow \sum_{j=0}^{n-1} B_{i,j}^{(C)} \beta^{[j]}, \forall i \in [0, \gamma-1]$, as in (3.35)
2  Calculate minimal subspace polynomials: $\Gamma^{(C)}(x) \leftarrow M_{d_0^{(C)}, d_1^{(C)}, \dots, d_{\gamma-1}^{(C)}}(x)$
                                          $\Lambda^{(R)}(x) \leftarrow M_{a_0^{(R)}, a_1^{(R)}, \dots, a_{\varrho-1}^{(R)}}(x)$
3  Caculate $q$-reverse $\overline{\Gamma^{(C)}}(x)$ with $\overline{\Gamma_i^{(C)}} = \Gamma_{-i \bmod m}^{(C)[i]}, \forall i \in [0, m-1]$
4  Calculate $\widehat{r}(x)$ by $q$-transform: $\widehat{r}_i \leftarrow r\big(\beta^{[i]}\big), \forall i \in [0, m-1]$ as in Definition 2.12
5  Calculate $\widehat{y}(x) \leftarrow \Lambda^{(R)}\big(\widehat{r}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) \mod (x^{[m]} - x)$
6  $r_{out}(x), u_{out}(x), v_{out}(x) \leftarrow$ FastHalfLEEA$\Big(x^{[m]} - x; \widehat{y}(x); d_{stop} = \Big\lfloor \frac{n+k+\varrho+\gamma}{2} \Big\rfloor\Big)$ with Algorithm 3.4
7  $f'(x); r'(x) \leftarrow$ LeftDiv$\big(r_{out}(x); u_{out}(\Lambda^{(R)}(x))\big)$ with Algorithm 2.2
8  $f(x); r(x) \leftarrow$ RightDiv$\big(f'(x); \overline{\Gamma^{(C)}}(x^{[\gamma]}) \mod (x^{[m]} - x)\big)$ with Algorithm 2.1
9  if $r(x) = 0$ and $r'(x) = 0$ then
       Output: Estimated evaluation polynomial $f(x)$ with $\deg_q f(x) < k$
10 else
       Output: "Decoding failure"
```

Proof. The correctness follows directly from Theorem 3.7 and Theorem 3.8. The complexity can be analyzed as follows, where the complexity of not-mentioned steps is negligible.

- Line 2: The minimal subspace polynomials can be calculated recursively with complexity $\mathcal{O}(\gamma^2)$ and $\mathcal{O}(\varrho^2)$ over $\mathbb{F}_{q^m}$ with the recursive procedure described in [SKK08, pp. 3961–3962].
- Line 4: The q-transform can be accomplished with $\mathcal{O}(n^2 \operatorname{comp}(\mathbf{T}_m)) \sim \mathcal{O}(n^2 m)$ operations in $\mathbb{F}_q$ as shown in Table 3.1.
- Line 5: The linearized composition modulo $(x^{[m]} - x)$ can be done with $\mathcal{M}_m(m) \leq \mathcal{O}(m^{1.69})$ operations in $\mathbb{F}_{q^m}$ if we use Algorithm 3.1 or with $\mathcal{M}_m(m) \leq \mathcal{O}(m^2 \operatorname{comp}(\mathbf{T}_m)) \sim \mathcal{O}(m^3)$ operations over $\mathbb{F}_q$ if we use Algorithm 3.3.
- Line 6: The call of the LEEA can be accomplished by using the fast LEEA from Algorithm 3.4 with complexity $\mathcal{O}(\max\{\mathcal{D}(m), \mathcal{M}(m)\} \log m)$. It is important to remark that both of our algorithms for calculating the linearized composition, Algorithms 3.1 and 3.3, depend on the degree of the involved polynomials and not only on m. This is necessary, since the polynomials in the recursions of the fast LEEA have degree much smaller than m and the whole algorithm can only be accelerated if the complexity of each step depends on this degree and not on m.
- Lines 7 and 8: The left and right division require complexity $\mathcal{D}(m)$.

∎

Hence, Theorem 3.9 shows that as soon as an efficient way to calculate the linearized division and the minimal subspace polynomial is found, we obtain a fast error-erasure decoding algorithm for Gabidulin codes with complexity $\mathcal{O}(m^3 \log m)$ over the *ground* field $\mathbb{F}_q$. The complexity of known decoding approaches are all in the order $\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$, with some improvements of sub-steps. An overview of decoding approaches is given in Table A.1 in Appendix A.3.

3.3 Summary and Outlook

The first part of this chapter deals with efficient algorithms for operations with linearized polynomials. We have analyzed the complexity of operations in finite fields using normal bases and of standard implementations for operations with linearized polynomials. Then, we have shown two methods for reducing the complexity of the linearized composition, one based on a fragmentation of the involved polynomials and one using linearized multi-point evaluation in the transform domain. In this context, also an efficient algorithm for calculating the linearized multi-point evaluation was given. Based on the Divide & Conquer principle, a fast linearized Euclidean algorithm was presented.

The second part of this chapter covers decoding approaches for Gabidulin codes. First, we have briefly summarized a well-known syndrome-based decoding approach by deriving two types of key equations and showing how to reconstruct the transmitted codeword if the rank of the additive error is at most half the minimum rank distance. Second, we have presented a new BMD decoding approach, which solves a transformed key equation by means of the linearized Euclidean algorithm and directly outputs the evaluation polynomial of the estimated codeword. This algorithm can be seen as the rank-metric equivalent to Gao's algorithm. Finally, we have shown how this algorithm can be extended to correct not only errors, but also row and column erasures simultaneously and how it can be accelerated by means of the fast linearized Euclidean algorithm.

In future, a fast linearized division and a fast calculation of the minimal subspace polynomial should be found. This will immediately speed up our decoding algorithm. Further, polynomial-time decoding of Gabidulin codes *beyond* half the minimum rank distance is a challenging open problem. An investigation of the possibilities of list decoding Gabidulin codes will be given in Chapter 5.

CHAPTER 4

Decoding Approaches for Interleaved Gabidulin Codes

Interleaved Gabidulin codes were introduced in Subsection 2.3.3 and can be seen as s parallel codewords of Gabidulin codes. When applied to random linear network coding, they can be advantageous compared to usual Gabidulin codes since only *one* identity matrix is appended to s Gabidulin codewords in order to construct constant-dimension codes. Therefore, the relative "overhead" is reduced. Independently from this application, for certain types of errors, the decoding capability of interleaved Gabidulin codes is higher than the usual BMD decoding capability.

This chapter is devoted to decoding interleaved Gabidulin codes. First, in Section 4.1, we explain two known decoding approaches [LO06, SB10] and prove a connection between them. In the subsequent sections, a new interpolation-based approach for decoding interleaved Gabidulin codes of length n, interleaving order s and elementary dimensions $k^{(i)}$, $\forall i \in [1, s]$, is presented. Our decoding principle relies on constructing a multi-variate linearized polynomial by interpolating the received words. We prove that the evaluation polynomials (of q-degree less than $k^{(i)}$) of any interleaved Gabidulin codeword in rank distance less than $(sn - \sum_{i=1}^{s} k^{(i)} + s)/(s+1)$ are roots of this multi-variate polynomial. Our decoding approach uses similar principles as Guruswami and Wang for folded/derivative Reed–Solomon codes [Gur11, GW13] and Mahdavifar and Vardy for folded Gabidulin codes [MV12].

Section 4.2 explains the basic principle of this decoder and shows how the two main steps—interpolation and root-finding—can each be accomplished by solving a linear systems of equations. Our decoder is first interpreted as a (not necessarily polynomial-time) list decoding algorithm in Subsection 4.3.1 and second, as a unique decoding algorithm with a certain failure probability in Subsection 4.3.2. To our knowledge, it is the first list decoding algorithm for interleaved Gabidulin codes. For the unique decoder, we derive a connection to the known unique decoding approaches from [LO06, SB10], which provides an upper bound on the failure probability. Finally, in Section 4.4, we show how our algorithm can be generalized to error-erasure decoding.

Parts of the results in Sections 4.2 and 4.3 were published in [WZ13].

4.1 Known Decoding Approaches

So far, there exist two approaches for decoding interleaved Gabidulin codes: one based on solving a system of equations constructed by the received words by Loidreau and Overbeck [LO06] and one based on the syndromes by Sidorenko and Bossert [SB10]. Both are unique probabilistic decoding algorithms and correct with high probability up to the radius $\tau = \lfloor s(n-k)/(s+1) \rfloor$ when $k^{(i)} = k$ for all $i \in [1, s]$.

In the following, we shortly summarize the two principles and prove a relation between them. It is important to remark that the approach from [SB10] was originally shown for *horizontally* interleaved

Gabidulin codes, i.e., where an interleaved codeword is defined by $(f^{(1)}(\mathbf{g})\ f^{(2)}(\mathbf{g})\ \dots\ f^{(s)}(\mathbf{g}))$. However, in the following, we describe it for *vertically* interleaved Gabidulin codes as in Definition 2.17.

Let $\mathbf{r}^{(i)} = (r_0^{(i)}\ r_1^{(i)}\ \dots\ r_{n-1}^{(i)})$, $\forall i \in [1,s]$, denote the s elementary received words, i.e., $\mathbf{r}^{(i)} = \mathbf{c}^{(i)} + \mathbf{e}^{(i)}$ and $\mathbf{c}^{(i)} \in \mathsf{Gab}[n, k^{(i)}]$ as in Definition 2.17. Further, let $t^{(i)} = \mathrm{rk}(\mathbf{e}^{(i)})$ and let $t \overset{\text{def}}{=} \mathrm{rk}(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T})$. We assume throughout this chapter that every interleaved error matrix $(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T})^T \in \mathbb{F}_{q^m}^{s \times n}$ of rank t is *equi-probable*.

With a usual BMD decoder, we could correct up to the radius $\lfloor (n-k^{(i)})/2 \rfloor$, $\forall i \in [1,s]$, with each elementary $\mathsf{Gab}[n, k^{(i)}]$ code. However, if the row spaces of the error words are connected, interleaved Gabidulin codes can correct more errors with high probability.

For the explanation of the two decoding principles, we assume that we know the actual rank of the error t, which enables us to directly set up the corresponding system of equations in the appropriate size. A straight-forward algorithmic realization would therefore solve this system of equations for every t, where $\lfloor (d-1)/2 \rfloor + 1 \leq t \leq \tau$, but this principle can easily be improved.

A Decoding Approach based on the Received Word

In [LO06], Loidreau and Overbeck established an approach for unique decoding of interleaved Gabidulin codes with $k^{(i)} = k, \forall i \in [1,s]$, up to the radius $\tau = \lfloor s(n-k)/(s+1) \rfloor$ with high probability. Clearly, their algorithm also works when the $k^{(i)}$ are different. We show the main properties of this general case in the following; for details the reader is referred to [LO06, Ove07, Ove08]. For some $t \leq \tau$, the main step of their decoding algorithm is to solve a linear system of equations

$$\mathbf{R}_R \cdot \boldsymbol{\lambda}^T = \mathbf{0}, \tag{4.1}$$

for $\boldsymbol{\lambda} = (\lambda_0\ \lambda_1\ \dots\ \lambda_{n-1})$, where the $(n-t-1+s(n-t)-\sum_{i=1}^{s} k^{(i)}) \times n$ matrix $\mathbf{R}_R$ depends on $\mathbf{g} = (g_0\ g_1\ \dots\ g_{n-1})$ and the received words:

$$\mathbf{R}_R = \begin{pmatrix} \mathbf{G}_R \\ \mathbf{R}_R^{(1)} \\ \mathbf{R}_R^{(2)} \\ \vdots \\ \mathbf{R}_R^{(s)} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} \mathrm{qvan}_{n-t-1}(\mathbf{g}) \\ \mathrm{qvan}_{n-k^{(1)}-t}(\mathbf{r}^{(1)}) \\ \mathrm{qvan}_{n-k^{(2)}-t}(\mathbf{r}^{(2)}) \\ \vdots \\ \mathrm{qvan}_{n-k^{(s)}-t}(\mathbf{r}^{(s)}) \end{pmatrix}. \tag{4.2}$$

If the right kernel of $\mathbf{R}_R$ has dimension one, the nearest interleaved codeword can be reconstructed since any vector $\boldsymbol{\lambda}$ in this right kernel has rank weight $n-t$ and reveals the error pattern, see [LO06] and [Ove07, Algorithm 3.2.1]. However, when $\mathbf{R}_R$ has rank less than $n-1$, the codeword cannot be reconstructed in most cases. Thus, the probability that $\mathrm{rk}(\mathbf{R}_R)$ is less than $n-1$, upper bounds the probability of a *decoding failure* (or equivalently, the fraction of non-correctable errors of rank t).

The first $k^{(i)}$ rows of $\mathbf{G}_R$, for $i \in [1,s]$, constitute the generator matrix of the $\mathsf{Gab}[n, k^{(i)}]$ code, which is the i-th elementary code of the $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ code. This is due to the fact that $t \leq \tau \leq n - \max_i\{k^{(i)}\} - 1$, and hence, $k^{(i)} \leq n-t-1, \forall i \in [1,s]$. Therefore, the right kernel of $\mathbf{R}_R$ can also be expressed in terms of the elementary *error* words:

$$\ker\big(\mathbf{R}_R\big) = \ker \begin{pmatrix} \mathrm{qvan}_{n-t-1}(\mathbf{g}) \\ \mathrm{qvan}_{n-k^{(1)}-t}(\mathbf{e}^{(1)}) \\ \mathrm{qvan}_{n-k^{(2)}-t}(\mathbf{e}^{(2)}) \\ \vdots \\ \mathrm{qvan}_{n-k^{(s)}-t}(\mathbf{e}^{(s)}) \end{pmatrix} \overset{\text{def}}{=} \ker\left(\mathbf{E}_R\right). \tag{4.3}$$

The rank of $\mathbf{G}_R$ is $n-t-1$ (compare Lemma 2.10) and the overall rank of the lower s submatrices of $\mathbf{E}_R$ is $t \leq \tau$. Hence, the overall rank is $\mathrm{rk}(\mathbf{R}_R) = \mathrm{rk}(\mathbf{E}_R) \leq n-1$. For $s \leq t$, the probability that $\mathrm{rk}(\mathbf{R}_R) < n-1$ was upper bounded by [LO06, Equation (6)], [Ove07, Equation (12)] as follows:

$$P\big(\mathrm{rk}(\mathbf{R}_R) < n-1\big) \leq 1 - \left(1 - \frac{4}{q^m}\right)\left(1 - q^{m(s-t)}\right)^s. \tag{4.4}$$

A Syndrome-Based Decoding Approach

The Sidorenko–Bossert approach [SB10, SJB11] considers unique decoding of interleaved Gabidulin codes of arbitrary dimensions $k^{(i)}$ and is to some extent a generalization of the key equation-based decoding approach for Gabidulin codes (see Subsection 3.2.1). Here, we will use the *row* space key equation from Theorem 3.5, but the principle was originally described for the "usual" column space key equation (Theorem 3.4). Denote the s syndrome vectors of length $n-k^{(i)}$ by:

$$\mathbf{s}^{(i)} = \mathbf{r}^{(i)} \cdot \mathbf{H}^{(i)T} = \mathbf{e}^{(i)} \cdot \mathbf{H}^{(i)T} = (s_0^{(i)}\ s_1^{(i)}\ \dots\ s_{n-k^{(i)}-1}^{(i)}), \quad \forall i \in [1,s],$$

where $\mathbf{H}^{(i)}$ is a parity-check matrix of $\mathsf{Gab}[n, k^{(i)}]$. Further, define the coefficients of the s modified syndromes as in (3.17) by:

$$\widetilde{s}_j^{(i)} = s_{n-k^{(i)}-1-j}^{(i)[j-n+k^{(i)}+1]}, \quad \forall i \in [1,s],\ \forall j \in [0, n-k^{(i)}-1],$$

and denote the s corresponding (modified) syndrome polynomials by $\widetilde{s}^{(i)}(x)$. Then, we obtain a row space key equation for each of the s modified syndromes as in (3.18):

$$\Gamma^{(i)}\big(\widetilde{s}^{(i)}(x)\big) \equiv \Phi^{(i)}(x) \mod x^{[n-k^{(i)}]}, \quad \forall i \in [1,s],$$

where $\deg_q \Gamma^{(i)}(x) = t^{(i)} = \mathrm{rk}(\mathbf{e}^{(i)})$ and $\deg_q \Phi^{(i)}(x) < \deg_q \Gamma^{(i)}(x)$.

Since we assume that $t = \mathrm{rk}\left(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T}\right)$, for the row space of the elementary errors $\mathcal{R}_q\left(\mathbf{e}^{(i)}\right) \subseteq \mathcal{R}_q\left(\mathbf{B}\right)$ holds for some $\mathbf{B} \in \mathbb{F}_q^{t \times n}$ of rank $t \leq \tau$ and $i \in [1,s]$. Hence, we can search for *one common* (row) error span polynomial $\Gamma(x)$ for all s key equations:

$$\Gamma\big(\widetilde{s}^{(i)}(x)\big) \equiv \Phi^{(i)}(x) \mod x^{[n-k^{(i)}]}, \quad \forall i \in [1,s],$$

where $\Gamma(x) = \sum_{i=0}^{t} \Gamma_i x^{[i]} = M_{\mathcal{D}}(x)$, where $\mathcal{D}$ is a basis of the overall row space, i.e., of $\mathcal{R}_q\left(\mathbf{e}^{(1)}\right) + \mathcal{R}_q\left(\mathbf{e}^{(2)}\right) + \dots + \mathcal{R}_q\left(\mathbf{e}^{(s)}\right)$, of dimension $t \leq \tau$.

Setting up these s key equations as a system of equations with the coefficients of $\Gamma(x)$ as unknowns (similar to (3.14)) provides the following linear system of equations (see also [Gab92, Equation (16)]):

$$\mathbf{S} \cdot \mathbf{\Gamma}^T = \begin{pmatrix} \mathbf{S}^{(1)} \\ \mathbf{S}^{(2)} \\ \vdots \\ \mathbf{S}^{(s)} \end{pmatrix} \cdot \mathbf{\Gamma}^T = \mathbf{0}, \tag{4.5}$$

where $\mathbf{\Gamma} = (\Gamma_0 \, \Gamma_1 \, \dots \, \Gamma_t)$ and

$$\mathbf{S}^{(i)} = \begin{pmatrix} \widetilde{s}_{t}^{(i)[0]} & \widetilde{s}_{t-1}^{(i)[1]} & \dots & \widetilde{s}_{0}^{(i)[t]} \\ \widetilde{s}_{t+1}^{(i)[0]} & s_{t}^{(i)[1]} & \dots & \widetilde{s}_{1}^{(i)[t]} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{s}_{n-k^{(i)}-1}^{(i)[0]} & \widetilde{s}_{n-k^{(i)}-2}^{(i)[1]} & \dots & \widetilde{s}_{n-k^{(i)}-1-t}^{(i)[t]} \end{pmatrix}$$

$$= \begin{pmatrix} s_{n-k^{(i)}-1-t}^{(i)[t-n+k^{(i)}+1]} & s_{n-k^{(i)}-t}^{(i)[t-n+k^{(i)}+1]} & \dots & s_{n-k^{(i)}-1}^{(i)[t-n+k^{(i)}+1]} \\ s_{n-k^{(i)}-2-t}^{(i)[t-n+k^{(i)}+2]} & s_{n-k^{(i)}-t-1}^{(i)[t-n+k^{(i)}+2]} & \dots & s_{n-k^{(i)}-2}^{(i)[t-n+k^{(i)}+2]} \\ \vdots & \vdots & \ddots & \vdots \\ s_{0}^{(i)[0]} & s_{1}^{(i)[0]} & \dots & s_{t}^{(i)[0]} \end{pmatrix}. \tag{4.6}$$

Thus, if $\mathrm{rk}(\mathbf{S}) = t$, we obtain a unique solution of $\Gamma(x)$ (except for a scalar factor) and we continue with finding the error vectors for each elementary Gabidulin code separately as in Subsection 3.2.1. The matrix $\mathbf{S}$ from (4.5) provides at most $\sum_{i=1}^{s}(n-k^{(i)}-t)$ linearly independent equations. In order to obtain $\mathrm{rk}(\mathbf{S}) = t$, the parameters have to satisfy

$$\sum_{i=1}^{s}(n-k^{(i)}-t) \geq t.$$

When $k^{(i)} = k, \forall i \in [1,s]$, the maximum decoding radius, which can be achieved by solving the linear system of equations from (4.5), is then $\tau = \lfloor s(n-k)/(s+1) \rfloor$.

For the approach from [SB10], the probability of failure can be upper bounded by the probability that $\mathbf{S}$ from (4.5) has rank less than t, which is bounded in [SB10, Theorem 5] for $s \leq \tau$ as follows:

$$P\big(\mathrm{rk}(\mathbf{S}) < t\big) \leq 3.5 \; q^{-m\big((s+1)(\tau-t)+1\big)} < \frac{4}{q^m}.$$

This bound improves the bound from [LO06] and in general we can use $P_f < 4/q^m$ as simplified upper bound on the failure probability of both cases.

Moreover, in [SB10, SJB11] an efficient algorithm based on linearized multi-sequence shift-register synthesis for decoding interleaved Gabidulin codes was developed, which implicitly finds the real value of t and solves (4.5). However, for analyzing the connection to [LO06] and to our approach, the interpretation as a linear system of equations is more convenient.

Connection Between the two Known Approaches

In the following lemma, we derive a connection between the two approaches from [LO06] and from [SB10] when decoding interleaved Gabidulin codes with $k^{(i)} = k, \forall i \in [1,s]$.

Lemma 4.1 (Relation Between Ranks of Decoding Interleaved Gabidulin Codes).
Let $k^{(i)} = k, \forall i \in [1,s]$, let $t \leq \tau = \lfloor s(n-k)/(s+1) \rfloor$ and let $\mathbf{R}_R$ and $\mathbf{S}$ be defined as in (4.2) *and* (4.5), (4.6)*. Then,* $\mathrm{rk}(\mathbf{S}) < t$ *if and only if* $\mathrm{rk}(\mathbf{R}_R) < n-1$.

Proof. First recall the matrix $\mathbf{R}_R$ from (4.2). The submatrix $\mathbf{G}_R$ is a generator matrix of a $\mathsf{Gab}[n, n-t-1]$ code. Let $\mathbf{h} = (h_0 \, h_1 \, \dots \, h_{n-1})$ define an $(n-k) \times n$ parity-check matrix $\mathbf{H}^{(0)}$ of the elementary $\mathsf{Gab}[n,k]$ code (which defines the $\mathrm{IGab}[s; n, k, \dots, k]$ code) as in Lemma 2.14.

Then, $\mathbf{H} = \text{qvan}_{t+1}((h_0^{[n-k-t-1]} \; h_1^{[n-k-t-1]} \; \dots \; h_{n-1}^{[n-k-t-1]}))$ is a $(t+1) \times n$ parity check matrix of a $\mathsf{Gab}[n, n-t-1]$ code and is a $(t+1) \times n$ submatrix of $\mathbf{H}^{(0)}$, consisting of the lowermost $t+1$ rows of $\mathbf{H}^{(0)}$.
Multiplying $\mathbf{R}_R$ by $\mathbf{H}^T$ and comparing the result to (4.6) gives:

$$\mathbf{R}_R \cdot \mathbf{H}^T = \begin{pmatrix} \text{qvan}_{n-t-1}(\mathbf{g}) \\ \text{qvan}_{n-k-t}(\mathbf{r}^{(1)}) \\ \text{qvan}_{n-k-t}(\mathbf{r}^{(2)}) \\ \vdots \\ \text{qvan}_{n-k-t}(\mathbf{r}^{(s)}) \end{pmatrix} \cdot \mathbf{H}^T = \begin{pmatrix} \text{qvan}_{n-t-1}(\mathbf{g}) \\ \text{qvan}_{n-k-t}(\mathbf{e}^{(1)}) \\ \text{qvan}_{n-k-t}(\mathbf{e}^{(2)}) \\ \vdots \\ \text{qvan}_{n-k-t}(\mathbf{e}^{(s)}) \end{pmatrix} \cdot \mathbf{H}^T = \begin{pmatrix} \mathbf{0} \\ \mathbf{S}^{(1)[n-k-t-1]} \\ \mathbf{S}^{(2)[n-k-t-1]} \\ \vdots \\ \mathbf{S}^{(s)[n-k-t-1]} \end{pmatrix}. \quad (4.7)$$

For any integer i, $\text{rk}(\mathbf{A}) = \text{rk}(\mathbf{A}^{[i]})$, where $\mathbf{A}^{[i]}$ means that every entry is taken to the q-power i.
Based on (4.7), we first prove the if part. Calculate by Gaussian elimination of $\mathbf{R}_R$ the matrix

$$\widetilde{\mathbf{E}} = \begin{pmatrix} \mathbf{G}_R \\ \widetilde{\mathbf{E}}_R \end{pmatrix}$$

such that $\text{rk}(\mathbf{R}_R) = \text{rk}(\widetilde{\mathbf{E}}) = \text{rk}(\mathbf{G}_R) + \text{rk}(\widetilde{\mathbf{E}}_R) = n - t - 1 + \text{rk}(\widetilde{\mathbf{E}}_R)$ (i.e., such that the ranks sum up). Notice that $\widetilde{\mathbf{E}}_R$ does not necessarily consist of the s lower submatrices of $\mathbf{E}_R$ from (4.3).
These elementary row operations do not change the rank and we obtain from (4.7)

$$\text{rk}\left(\mathbf{S}\right) = \text{rk}\left(\mathbf{S}^{[n-k-t-1]}\right) = \text{rk}\left(\mathbf{R}_R \cdot \mathbf{H}^T\right) = \text{rk}\left(\widetilde{\mathbf{E}} \cdot \mathbf{H}^T\right) = \text{rk}\left(\widetilde{\mathbf{E}}_R \cdot \mathbf{H}^T\right).$$

Now, if $\text{rk}(\mathbf{R}_R) < n-1$, then $\text{rk}(\widetilde{\mathbf{E}}_R) < t$ since $\text{rk}(\mathbf{G}_R) = n-t-1$. Then, also $\text{rk}\left(\widetilde{\mathbf{E}}_R \cdot \mathbf{H}^T\right) < t$ and therefore $\text{rk}(\mathbf{S}) < t$.
Second, let us prove the only if part. Due to Sylvester's rank inequality

$$\text{rk}\left(\mathbf{R}_R\right) + \text{rk}\left(\mathbf{H}^T\right) - n \leq \text{rk}\left(\mathbf{R}_R \cdot \mathbf{H}^T\right) = \text{rk}\left(\mathbf{S}^{[n-k-t-1]}\right) = \text{rk}\left(\mathbf{S}\right).$$

Clearly, $\text{rk}(\mathbf{H}) = t+1$. Hence, if $\text{rk}(\mathbf{S}) < t$, then $\text{rk}\left(\mathbf{R}_R\right) \leq n - t - 1 + \text{rk}\left(\mathbf{S}\right) < n - 1$.

∎

Thus, we proved that both approaches fail for exactly the same error matrices and clearly also have the same fraction of correctable error matrices when $k^{(i)} = k$, $\forall i \in [1, s]$. This means that the tighter bound on the failure probability from [SB10] can also be used to bound the failure probability of [LO06].

However, for arbitrary $k^{(i)} = k$, it is not clear if the matrix on the RHS of (4.7) has the same rank as $\mathbf{S}$ since the q-powers of each submatrix differ.

4.2 Principle of Interpolation-Based Decoding

Sudan [Sud97] and Guruswami and Sudan [GS99] introduced polynomial-time list decoding of Reed–Solomon and Algebraic Geometry codes based on interpolating bivariate (usual) polynomials. For *linearized* polynomials, however, it is not clear how to define mixed terms (i.e., monomials containing more than one indeterminate) and how to design a list decoding algorithm for Gabidulin codes similar to [Sud97, GS99]. When we define bivariate linearized polynomials *without* mixed terms, it is possible to decode a $\mathsf{Gab}[n, k]$ code up to the BMD radius $\lfloor (n-k)/2 \rfloor$, which was done in [Loi06].

Our decoding approach for *interleaved* Gabidulin codes relies on interpolating a *multi*-variate linearized polynomials without mixed terms. The principle consists of an interpolation and a root-finding step. First, we give interpolation constraints for a multi-variate linearized interpolation polynomial $Q(x, y_1, \dots, y_s) = Q_0(x) + \sum_{i=1}^{s} Q_i(y_i)$ and prove that the q-degree-restricted evaluation polynomials $f^{(1)}(x), \dots, f^{(s)}(x)$ of the interleaved Gabidulin codeword are roots of $Q(x, y_1, \dots, y_s)$ up to a certain radius τ. Second, we show how the root finding step can be accomplished by solving a linear system of equations.

4.2.1 Interpolation Step

The conditions on our multi-variate linearized interpolation polynomial are as follows.

Problem 4.1 (Interpolation Step for Decoding Interleaved Gabidulin Codes).
Let $r^{(i)}(x) = \sum_{j=0}^{n-1} r_j^{(i)} x^{[j]} \in \mathbb{L}_{q^m}[x]$, $\forall i \in [1, s]$, and $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$, which are linearly independent over $\mathbb{F}_q$, be given.

Find an $(s+1)$-variate linearized polynomial of the form

$$Q(x, y_1, \dots, y_s) = Q_0(x) + Q_1(y_1) + \dots + Q_s(y_s),$$

which satisfies for given integers $n, \tau, k^{(1)}, \dots, k^{(s)}$:

- $Q(g_j, r_j^{(1)}, \dots, r_j^{(s)}) = 0, \forall j \in [0, n-1]$,
- $\deg_q Q_0(x) < n - \tau$,
- $\deg_q Q_i(y_i) < n - \tau - (k^{(i)} - 1), \forall i \in [1, s]$.

Let us denote the coefficients of the univariate polynomials by

$$Q_0(x) = \sum_{j=0}^{n-\tau-1} q_{0,j} x^{[j]}, \quad Q_i(y_i) = \sum_{j=0}^{n-\tau-k^{(i)}} q_{i,j} y_i^{[j]}, \quad \forall i \in [1, s].$$

A solution to Problem 4.1 can be found by solving a linear system of equations, which is denoted by $\mathbf{R} \cdot \mathbf{q}^T = \mathbf{0}$, where $\mathbf{g} = (g_0 \ g_1 \ \dots \ g_{n-1})$ and $\mathbf{R}$ is an $n \times (n - \tau + \sum_{i=1}^{s}(n - \tau - k^{(i)} + 1))$ matrix as follows:

$$\mathbf{R} = \left(\mathrm{qvan}_{n-\tau}(\mathbf{g})^T \ \mathrm{qvan}_{n-\tau-k^{(1)}+1}(\mathbf{r}^{(1)})^T \ \dots \ \mathrm{qvan}_{n-\tau-k^{(s)}+1}(\mathbf{r}^{(s)})^T \right), \tag{4.8}$$

and $\mathbf{q} = (q_{0,0} \ \dots \ q_{0,n-\tau-1} \mid q_{1,0} \ \dots \ q_{1,n-\tau-k^{(1)}} \mid \dots \mid q_{s,0} \ \dots \ q_{s,n-\tau-k^{(s)}})$.

Lemma 4.2 (Maximum Radius).
There exists a non-zero $Q(x, y_1, \dots, y_s)$, fulfilling the conditions of Problem 4.1 if

$$\tau < \frac{sn - \sum_{i=1}^{s} k^{(i)} + s}{s+1}. \tag{4.9}$$

Proof. The number of linearly independent equations is given by the interpolation constraints (i.e., the number of rows of $\mathbf{R}$ in (4.8)), which is is at most n and has to be less than the number of unknowns (given by the length of $\mathbf{q}$) in order to guarantee that there is a non-zero solution. Hence:

$$n < n - \tau + \sum_{i=1}^{s} \left(n - \tau - k^{(i)} + 1 \right) \quad \iff \quad \tau(s+1) < sn + s - \sum_{i=1}^{s} k^{(i)}.$$

■

For the special case $k^{(i)} = k, \forall i \in [1, s]$, this gives $\tau < s(n-k+1)/(s+1)$.

The unique decoding approaches from [LO06, SB10] (see Section 4.1) have maximum decoding radius $\tau_u = (sn - \sum_{i=1}^{s} k^{(i)})/(s+1)$. A comparison to the maximum value of τ given by Lemma 4.2 provides the following corollary, showing that our decoding radius is at least the same as τ_u.

Corollary 4.1 (Decoding Radii for Interpolation-Based and Joint Decoding).
Let τ be the greatest integer fulfilling (4.9) *and let $\tau_u = \left\lfloor (sn - \sum_{i=1}^{s} k^{(i)})/(s+1) \right\rfloor$. Then, $1 \geq \tau - \tau_u \geq 0$.*

The following theorem shows that the evaluation words of the interleaved Gabidulin code are a root of any interpolation polynomial, which fulfills Problem 4.1.

Theorem 4.1 (Roots of Interpolation Polynomial).
Let $\mathbf{c}^{(i)} = f^{(i)}(\mathbf{g})$, where $\deg_q f^{(i)}(x) < k^{(i)}$, and let $\mathbf{r}^{(i)} = \mathbf{c}^{(i)} + \mathbf{e}^{(i)}$, $\forall i \in [1, s]$. Assume, $t = \operatorname{rk}\left(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T}\right) \leq \tau$, where τ satisfies (4.9)*. Let a non-zero $Q(x, y_1, \dots, y_s)$ be given, fulfilling the interpolation constraints from Problem 4.1. Then,*

$$F(x) \overset{\text{def}}{=} Q\left(x, f^{(1)}(x), \dots, f^{(s)}(x)\right) = 0. \tag{4.10}$$

Proof. Define $\widehat{r}^{(i)}(x)$ and $\widehat{e}^{(i)}(x)$ such that $\widehat{r}^{(i)}(g_j) = r_j^{(i)}$ and $\widehat{e}^{(i)}(g_j) = e_j^{(i)} = r_j^{(i)} - c_j^{(i)}$, $\forall j \in [0, n-1]$ and $\forall i \in [1, s]$ using linearized Lagrange interpolation as in (3.23), (3.24).
Further, denote $R(x) \overset{\text{def}}{=} Q\big(x, \widehat{r}^{(1)}(x), \dots, \widehat{r}^{(s)}(x)\big)$. Since all polynomials are linearized,

$$\begin{aligned} R(x) - F(x) &= Q\big(0, \widehat{e}^{(1)}(x), \dots, \widehat{e}^{(s)}(x)\big) \\ &= Q_1\big(\widehat{e}^{(1)}(x)\big) + Q_2\big(\widehat{e}^{(2)}(x)\big) + \dots + Q_s\big(\widehat{e}^{(s)}(x)\big). \end{aligned}$$

Then,

$$\begin{aligned} R(\mathbf{g}) - F(\mathbf{g}) &= \sum_{i=1}^{s} Q_i\big(\widehat{e}^{(i)}(\mathbf{g})\big) = \sum_{i=1}^{s} Q_i\big(\mathbf{e}^{(i)}\big) \\ &= \Big(\sum_{i=1}^{s} Q_i(e_0^{(i)})\ \sum_{i=1}^{s} Q_i(e_1^{(i)})\ \dots\ \sum_{i=1}^{s} Q_i(e_{n-1}^{(i)}) \Big). \end{aligned}$$

Lemma 2.12 shows that the row spaces fulfill

$$\mathcal{R}_q \left(\sum_{i=1}^{s} Q_i\big(\mathbf{e}^{(i)}\big) \right) \subseteq \mathcal{R}_q \left((\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T})^T \right).$$

Because of the interpolation constraints, $R(\mathbf{g}) = \mathbf{0}$ and hence $\operatorname{rk}\left(F(\mathbf{g})\right) = \operatorname{rk}(\sum_{i=1}^{s} Q_i(\mathbf{e}^{(i)})) \leq \operatorname{rk}(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T}) = t \leq \tau$.
If $\operatorname{rk}(F(\mathbf{g})) \leq \tau$, then the dimension of the root space of $F(x)$ in $\mathbb{F}_{q^m}$ has to be at least $n - \tau$, which is only possible if its q-degree is at least $n - \tau$. However, $\deg_q F(x) \leq n - \tau - 1$ due to the interpolation constraints and therefore $F(x) = 0$. ■

The interpolation step can be accomplished by solving the linear system of equations based on the matrix $\mathbf{R}$ from (4.8), which requires cubic complexity in $\mathbb{F}_{q^m}$ when using Gaussian elimination. Instead of this, the efficient interpolation from [XYS11] can be used and the complexity of the interpolation step is reduced to $\mathcal{O}(s^2 n(n-\tau))$ operations over $\mathbb{F}_{q^m}$ (in their notation $L = s$, $C = n$ and $D = n - \tau - 1$ and the complexity of their algorithm is $\mathcal{O}(L^2 CD)$).

4.2.2 Root-Finding Step

Similar to Guruswami and Wang in [Gur11, GW13] for folded/derivative Reed–Solomon codes and to Mahdavifar and Vardy in [MV12] for folded Gabidulin codes, the root-finding step of our approach results in solving a *linear* system of equations.

Assume, $Q(x, y_1, \dots, y_s)$ is given, fulfilling the interpolation constraints from Problem 4.1. Then, the task of the root-finding step is to find all tuples of polynomials $f^{(1)}(x), f^{(2)}(x), \dots, f^{(s)}(x)$ such that

$$F(x) = Q_0(x) + Q_1\big(f^{(1)}(x)\big) + Q_2\big(f^{(2)}(x)\big) + \dots + Q_s\big(f^{(s)}(x)\big) = 0.$$

The important observation is that this is a linear system of equations over $\mathbb{F}_{q^m}$ in the coefficients of $f^{(1)}(x), f^{(2)}(x), \dots, f^{(s)}(x)$. Recall for this purpose that $(a+b)^{[i]} = a^{[i]} + b^{[i]}$ for any $a, b \in \mathbb{F}_{q^m}$ and any integer i and let us demonstrate the root-finding step with an example.

Example 4.1 (Root-Finding).
Let $s = 2$, $n = m = 7$, $k^{(1)} = k^{(2)} = 2$ and $\tau = 3$. Find all pairs $f^{(1)}(x)$, $f^{(2)}(x)$ with $\deg_q f^{(1)}(x) = \deg_q f^{(2)}(x) < 2$ such that $F(x) = F_0 x^{[0]} + F_1 x^{[1]} + \dots + F_{n-\tau-1} x^{[n-\tau-1]} = 0$. Due to the constraints of Problem 4.1, $\deg_q F(x) \leq n - \tau - 1 = 3$. Thus,

$$\begin{aligned}
F_0 &= 0 = q_{0,0} + q_{1,0} f_0^{(1)} + q_{2,0} f_0^{(2)},\\
F_1 &= 0 = q_{0,1} + q_{1,1} f_0^{(1)[1]} + q_{1,0} f_1^{(1)} + q_{2,1} f_0^{(2)[1]} + q_{2,0} f_1^{(2)},\\
F_2 &= 0 = q_{0,2} + q_{1,2} f_0^{(1)[2]} + q_{1,1} f_1^{(1)[1]} + q_{2,2} f_0^{(2)[2]} + q_{2,1} f_1^{(2)[1]},\\
F_3 &= 0 = q_{0,3} + q_{1,2} f_1^{(1)[2]} + q_{2,2} f_1^{(2)[2]}.
\end{aligned}$$

Therefore, given $Q(x, y_1, y_2)$, we can calculate the coefficients of all possible pairs $f^{(1)}(x)$, $f^{(2)}(x)$ of q-degree less than two by the following linear system of equations:

$$\begin{pmatrix}
q_{1,0} & q_{2,0} & & \\
q_{1,1}^{[-1]} & q_{2,1}^{[-1]} & q_{1,0}^{[-1]} & q_{2,0}^{[-1]}\\
q_{1,2}^{[-2]} & q_{2,2}^{[-2]} & q_{1,1}^{[-2]} & q_{2,1}^{[-2]}\\
 & & q_{1,2}^{[-3]} & q_{2,2}^{[-3]}
\end{pmatrix} \cdot \begin{pmatrix}
f_0^{(1)}\\
f_0^{(2)}\\
f_1^{(1)[-1]}\\
f_1^{(2)[-1]}
\end{pmatrix} = \begin{pmatrix}
-q_{0,0}\\
-q_{0,1}^{[-1]}\\
-q_{0,2}^{[-2]}\\
-q_{0,3}^{[-3]}
\end{pmatrix}. \tag{4.11}$$

In order to set up (4.11) in general, we can use more than one $Q(x, y_1, \dots, y_s)$. Namely, we can use all polynomials corresponding to different basis vectors of the solution space of the interpolation step. This also decreases the probability that the system of equations for the root-finding step does not have full rank (see also Subsection 4.3.2). In order to calculate the dimension of the solution space of the interpolation step, denoted by d_I, we need the rank of the interpolation matrix.

Lemma 4.3 (Rank of Interpolation Matrix).
Let $\operatorname{rk}\left(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T}\right) = t \leq \tau$, where τ satisfies (4.9). Then, for the interpolation matrix from (4.8), $\operatorname{rk}(\mathbf{R}) \leq n - \tau + t$ holds.

Proof. The first $k^{(i)}$ columns of $\mathbf{R}$ contain the (transposed) generator matrices of the Gabidulin codes $\mathsf{Gab}[n, k^{(i)}]$. Hence, for calculating the rank of $\mathbf{R}$, we can subtract the codewords and their q-powers from the s right submatrices such that these submatrices only depend on the error. Hence, the rank of $\mathbf{R}$ depends on $\operatorname{rk}(\mathrm{qvan}_{n-\tau}(\mathbf{g}))$, which is $n - \tau$, and on the rank of the error matrix, which is t. Hence, $\operatorname{rk}(\mathbf{R}) \leq n - \tau + t$. ∎

The dimension of the solution space of the interpolation step is therefore:

$$\begin{aligned} d_I \overset{\text{def}}{=} \dim\ker(\mathbf{R}) &\geq (s+1)(n-\tau) - \sum_{i=1}^{s}(k^{(i)}-1) - (n-\tau+t) \\ &= s(n-\tau+1) - \sum_{i=1}^{s} k^{(i)} - t, \end{aligned} \tag{4.12}$$

and for $k^{(i)} = k$ for all $i \in [1,s]$, we obtain $d_I \geq s(n-\tau-k+1)-t$.

In the following, let $Q^{(h)}(x,y_1,\dots,y_s)$, $\forall h \in [1,d_I]$, denote the interpolation polynomials corresponding to different basis vectors of the solution space of the interpolation step. Let us denote the following matrices:

$$\mathbf{Q}_j^{[i]} \overset{\text{def}}{=} \begin{pmatrix} q_{1,j}^{(1)[i]} & q_{2,j}^{(1)[i]} & \dots & q_{s,j}^{(1)[i]} \\ q_{1,j}^{(2)[i]} & q_{2,j}^{(2)[i]} & \dots & q_{s,j}^{(2)[i]} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1,j}^{(d_I)[i]} & q_{2,j}^{(d_I)[i]} & \dots & q_{s,j}^{(d_I)[i]} \end{pmatrix}, \quad \mathbf{f}_j^{[i]} \overset{\text{def}}{=} \begin{pmatrix} f_j^{(1)[i]} \\ f_j^{(2)[i]} \\ \vdots \\ f_j^{(s)[i]} \end{pmatrix}, \quad \mathbf{q}_{0,j}^{[i]} \overset{\text{def}}{=} \begin{pmatrix} q_{0,j}^{(1)[i]} \\ q_{0,j}^{(2)[i]} \\ \vdots \\ q_{0,j}^{(d_I)[i]} \end{pmatrix}. \tag{4.13}$$

For $k = \max_i\{k^{(i)}\}$, the linear system of equations for finding the roots of $Q(x,y_1,\dots,y_s)$ is:

$$Q^{(h)}\big(x, f^{(1)}(x),\dots,f^{(s)}(x)\big) = Q_0^{(h)}(x) + Q_1^{(h)}(f^{(1)}(x)) + \dots + Q_s^{(h)}(f^{(s)}(x)) = 0, \quad \forall h \in [1,d_I]$$

$$\Longleftrightarrow$$

$$\underbrace{\begin{pmatrix} \mathbf{Q}_0^{[0]} & & & \\ \mathbf{Q}_1^{[-1]} & \mathbf{Q}_0^{[-1]} & & \\ \mathbf{Q}_2^{[-2]} & \mathbf{Q}_1^{[-2]} & \mathbf{Q}_0^{[-2]} & \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & \mathbf{Q}_{n-\tau-k}^{[-(n-\tau-3)]} & \mathbf{Q}_{n-\tau-k-1}^{[-(n-\tau-3)]} & \mathbf{Q}_{n-\tau-k-2}^{[-(n-\tau-3)]} \\ & & \mathbf{Q}_{n-\tau-k}^{[-(n-\tau-2)]} & \mathbf{Q}_{n-\tau-k-1}^{[-(n-\tau-2)]} \\ & & & \mathbf{Q}_{n-\tau-k}^{[-(n-\tau-1)]} \end{pmatrix}}_{\mathbf{Q}} \cdot \underbrace{\begin{pmatrix} \mathbf{f}_0 \\ \mathbf{f}_1^{[-1]} \\ \vdots \\ \mathbf{f}_{k-1}^{[-(k-1)]} \end{pmatrix}}_{\mathbf{f}} = \underbrace{\begin{pmatrix} -\mathbf{q}_{0,0} \\ -\mathbf{q}_{0,1}^{[-1]} \\ \vdots \\ -\mathbf{q}_{0,n-\tau-1}^{[-(n-\tau-1)]} \end{pmatrix}}_{\mathbf{q}_0}, \tag{4.14}$$

where $\mathbf{Q}$ is a $((n-\tau)d_I) \times sk$ matrix and where we assume that $f_j^{(i)} = 0$ if $j \geq k^{(i)}$ and $q_{i,j} = 0$ when $j \geq n-\tau-k^{(i)}$, $\forall i \in [1,s]$.

Lemma 4.4 (Complexity of the Root-Finding Step).
Let $Q^{(h)}(x,y_1,\dots,y_s)$, $\forall h \in [1,d_I]$, be given, satisfying the interpolation constraints from Problem 4.1. Then, the basis of the subspace, which contains the coefficients of all tuples $f^{(1)}(x),\dots,f^{(s)}(x)$ such that

$$F(x) = Q\left(x, f^{(1)}(x),\dots,f^{(s)}(x)\right) = Q_0(x) + Q_1(f^{(1)}(x)) + \dots + Q_s(f^{(s)}(x)) = 0,$$

can be found recursively with complexity at most $\mathcal{O}(s^3k^2)$ operations over $\mathbb{F}_{q^m}$.

Proof. The complexity of calculating q-powers is negligible (compare Table 3.1). The solution of (4.14) can be found by the following recursive procedure. First, solve the linear system of equations $\mathbf{Q}_0^{[0]} \cdot \mathbf{f}_0 = -\mathbf{q}_{0,0}$ of size $d_I \times s$ for $\mathbf{f}_0$ with complexity at most $\mathcal{O}(s^3)$ when using Gaussian elimination. Afterwards, calculate $\mathbf{Q}_1^{[-1]} \cdot \mathbf{f}_0$ with $sd_I \approx s^2$ multiplications over $\mathbb{F}_{q^m}$ and solve the system $\mathbf{Q}_1^{[-1]} \cdot \mathbf{f}_0 + \mathbf{Q}_0^{[-1]} \cdot \mathbf{f}_1^{[-1]} = -\mathbf{q}_{0,1}^{[-1]}$ for $\mathbf{f}_1$ with complexity at most $\mathcal{O}(s^3)$ operations. We continue this until we obtain all coefficients of $f^{(1)}(x), \dots, f^{(s)}(x)$, where for $\mathbf{f}_j$, we first have to calculate $(j-1) \cdot s \cdot d_I$ multiplications over $\mathbb{F}_{q^m}$ and solve a $d_I \times s$ linear system of equations. Hence, the overall complexity for the root-finding step is upper bounded by $\sum_{j=1}^{k} \big((j-1) \cdot s \cdot d_I + s^3\big) \leq \mathcal{O}(s^2k^2 + s^3k) \leq \mathcal{O}(s^3k^2)$ operations over $\mathbb{F}_{q^m}$. ∎

4.3 Interpolation-Based Decoding Approaches

The decoding principle from the previous section can either be used as a list decoding algorithm, which returns all codewords of the interleaved Gabidulin code in rank distance at most τ from the received word, where τ satisfies (4.9) (described in Subsection 4.3.1), or as a probabilistic unique decoding algorithm (described in Subsection 4.3.2). For the probabilistic algorithm, we derive a relation to the known unique decoding algorithms and bound the failure probability.

4.3.1 A List Decoding Approach

Our decoding approach for interleaved Gabidulin codes can be seen as a list decoding algorithm, consisting of solving two linear systems of equations. Except for pruning the solution space of the root-finding step, we find the solution(s) with complexity at most $\mathcal{O}(s^3n^2)$ operations in $\mathbb{F}_{q^m}$. However, our algorithm is *not* a polynomial-time list decoding algorithm (with respect to n) for interleaved Gabidulin codes since the list size can become exponential in n as shown in the following lemma.

Lemma 4.5 (Maximum List Size).
Let $\mathbf{r}^{(i)}$*,* $\forall i \in [1,s]$*, be given and let* τ *satisfy* (4.9)*. Then, the list size* ℓ_I*, i.e., the number of codewords from* $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ *in rank distance at most* τ *to* $\mathbf{r} = (\mathbf{r}^{(1)T}\ \mathbf{r}^{(2)T}\ \dots\ \mathbf{r}^{(s)T})^T$*, is*

$$\ell_I \overset{\text{def}}{=} \max_{\mathbf{r} \in \mathbb{F}_{q^m}^{s \times n}} \left\{ \left| \mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}] \cap \mathcal{B}_R^{(\tau)}(\mathbf{r}) \right| \right\} \leq q^{m\left(\sum_{i=1}^{s} k^{(i)} - \min_i\{k^{(i)}\}\right)}.$$

Proof. The list size can be upper bounded by the maximum number of solutions for the root-finding step (4.14). There exists an integer $i \in [1,s]$ such that $Q_i(x) \neq 0$, since $Q(x, y_1, \dots, y_s) \neq 0$. Note that $Q_0(x) \neq 0$ and $Q_i(x) = 0$, $\forall i \in [1,s]$, is not possible since $(\mathrm{qvan}_{n-\tau}(\mathbf{g}))^T$ is a full-rank matrix.
Hence, let $i \in [1,s]$ be such that $Q_i(x) \neq 0$ and let j be the smallest integer such that $q_{i,j} \neq 0$. Consider the submatrix of $\mathbf{Q}$, which consists of the columns corresponding to the coefficients of $f^{(i)}(x)$. For some $h \in [1,s]$, this submatrix contains at least one $k^{(i)} \times k^{(i)}$ lower triangular matrix with $q_{i,j}^{(h)[-j]}, q_{i,j}^{(h)[-(j+1)]}, \dots, q_{i,j}^{(h)[-(j+k^{(i)}-1)]}$ on the diagonal. Therefore, $\mathrm{rk}(\mathbf{Q}) \geq \min_i\{k^{(i)}\}$ and the dimension of the solution space is at most $(\sum_{i=1}^{s} k^{(i)} - \min_i\{k^{(i)}\})$. ∎

It is not clear whether the list size ℓ_I can really be that great. Moreover, finding the actual list of codewords out of the solution space of (4.14) further reduces the list size.

When $\ell_I > 1$, the system of equations for the root-finding step (4.14) cannot have full rank. The following lemma estimates the average list size. For most parameters, this value is almost one (see Example 4.2). The proof proceeds similar to McEliece's proof for the average list size in the Guruswami–Sudan algorithm [McE03].

Lemma 4.6 (Average List Size).
Let $\mathbf{c}^{(i)} = f^{(i)}(\mathbf{g})$, $\forall i \in [1,s]$, *where* $\deg_q f^{(i)}(x) < k^{(i)}$ *and let* $\mathbf{r}^{(i)} = \mathbf{c}^{(i)} + \mathbf{e}^{(i)}$. *Let* $\mathrm{rk}(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \ldots\ \mathbf{e}^{(s)T}) = t \leq \tau$ *and let* τ *satisfy* (4.9). *Then, the average list size, i.e., the average number of codewords* $(\mathbf{c}^{(1)T}\ \mathbf{c}^{(2)T}\ \ldots\ \mathbf{c}^{(s)T})^T \in \mathsf{IGab}[s;n,k^{(1)},\ldots,k^{(s)}]$ *such that*

$$\mathrm{rk}\left((\mathbf{r}^{(1)T}\ \mathbf{r}^{(2)T}\ \ldots\ \mathbf{r}^{(s)T}) - (\mathbf{c}^{(1)T}\ \mathbf{c}^{(2)T}\ \ldots\ \mathbf{c}^{(s)T})\right) \leq \tau,$$

is upper bounded by

$$\overline{\ell_I} < 1 + 4\left(q^{m\sum_{i=1}^{s} k^{(i)}} - 1\right) q^{(sm+n)\tau - \tau^2 - smn}.$$

Proof. Let R be a random variable, uniformly distributed over all matrices in $\mathbb{F}_{q^m}^{s\times n}$ and let $\mathbf{r}$ be a realization of R, i.e., the s elementary received words written as rows of a matrix. Let $\mathbf{c} \in \mathsf{IGab}[s;n,k^{(1)},\ldots,k^{(s)}]$ be the fixed transmitted codeword. Then, $P(\mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \tau) = P(\mathrm{rk}(\mathbf{r}) \leq \tau)$, which is the probability that a random $sm \times n$ matrix over $\mathbb{F}_q$ has rank at most τ. Let $\mathsf{IGab}^*[s;n,k^{(1)},\ldots,k^{(s)}]$ be the code $\mathsf{IGab}[s;n,k^{(1)},\ldots,k^{(s)}]$ *without* the transmitted codeword.

Let us further consider another random variable X, which depends on R:

$$X(R) = \left|\left\{\mathsf{IGab}^* \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})\right\}\right|,$$

where $\mathbf{r} \in \mathbb{F}_{q^m}^{s\times n}$. Denote by $\mathbf{1}(..)$ the indicator function, then the expectation of X is given by:

$$\begin{aligned} E[X] &= \sum_{\mathbf{r}\in\mathbb{F}_{q^m}^{s\times n}} P(R=\mathbf{r})X(R) \\ &= \sum_{\mathbf{c}\in\mathsf{IGab}^*} \sum_{\mathbf{r}\in\mathbb{F}_{q^m}^{s\times n}} \mathbf{1}(\mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \tau) P(R=\mathbf{r}) \\ &= \sum_{\mathbf{c}\in\mathsf{IGab}^*} E\big[\mathbf{1}(\mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \tau)\big] \\ &= \sum_{\mathbf{c}\in\mathsf{IGab}^*} P(\mathrm{rk}(\mathbf{r}-\mathbf{c}) \leq \tau) \\ &= \sum_{\mathbf{c}\in\mathsf{IGab}^*} P(\mathrm{rk}(\mathbf{r}) \leq \tau). \end{aligned}$$

Therefore

$$\begin{aligned} E[X] &= \left|\mathsf{IGab}^*\right| \cdot \frac{\left|\mathbf{R} \in \mathbb{F}_q^{sm\times n} : \mathrm{rk}(\mathbf{R}) \leq \tau\right|}{q^{smn}} \\ &< \left((q^m)^{\sum_{i=1}^{s} k^{(i)}} - 1\right) \frac{4q^{(sm+n)\tau-\tau^2}}{q^{smn}}. \end{aligned}$$

The average list size is $\overline{\ell_I} = E[X] + 1$, since we have to add the transmitted codeword. ∎

Unfortunately, it is not clear if it is possible that $\ell_I = 1$ and nonetheless, the system of equations for the root-finding step (4.14) does *not* have full rank. Thus, Lemma 4.6 does not bound the probability that the rank of $\mathbf{Q}$ is not full; this will be done in Lemma 4.9.

Theorem 4.2 summarizes the properties of our list decoding algorithm.

Theorem 4.2 (List Decoding of Interleaved Gabidulin Codes).
Let the interleaved Gabidulin code $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ *over* $\mathbb{F}_{q^m}$ *consist of the elementary codewords* $\mathbf{c}^{(i)} = f^{(i)}(\mathbf{g})$*, where* $\deg_q f^{(i)}(x) < k^{(i)}$ *and let the elementary received words* $\mathbf{r}^{(i)}$*,* $\forall i \in [1, s]$*, be given.*

Then, we can find a basis of the subspace, containing all tuples of polynomials $f^{(1)}(x), \dots, f^{(s)}(x)$*, such that their evaluation at* $\mathbf{g}$ *is in rank distance*

$$\tau < \frac{sn - \sum_{i=1}^{s} k^{(i)} + s}{s+1}$$

to $(\mathbf{r}^{(1)T}\ \mathbf{r}^{(2)T}\ \dots\ \mathbf{r}^{(s)T})^T$ *with overall complexity at most* $\mathcal{O}(s^3n^2)$*.*

The complexity of finding the basis of the list is quadratic in n, but the worst-case complexity for finding explicitly the whole list can be exponential in n. Therefore, this is not a polynomial-time list decoder, although in most cases the list size is one and in then, the complexity of finding the unique solution is quadratic in the length of the code.

4.3.2 A Probabilistic Unique Decoding Approach

In this section, we consider our decoding approach as a probabilistic unique decoding algorithm. Since the list size might be greater than one, there is not always a unique solution. We accomplish the interpolation step as before and declare a decoding failure as soon as the rank of the root-finding matrix $\mathbf{Q}$ (see (4.14)) is not full. We upper bound this probability and call it *failure probability.* Moreover, we show a relation to the unique decoding approaches from [LO06, SB10]. The upper bound as well as simulation results show that the failure probability is quite small. Therefore, we can use our decoder as probabilistic unique decoder which basically consists of solving two structured linear systems of equations and has overall complexity at most $\mathcal{O}(s^3n^2)$, where $s \ll n$ is usually a small fixed integer.

It is important to observe that we always set up the system of equations for the interpolation step (Problem 4.1) with maximum possible τ, but—in contrast to solving the systems of equations from (4.1) and (4.5) — we also find the unique solution (if it exists) if $t < \tau$ without decreasing the size of the matrix, since the rank of the matrix $\mathbf{R}$ from (4.8) is not important.

Recall the matrix notations from (4.13) and denote additionally the $d_I \times (s+1)$ matrix

$$\overline{\mathbf{Q}}_0 \overset{\text{def}}{=} \begin{pmatrix} q_{0,0}^{(1)} & q_{1,0}^{(1)} & \dots & q_{s,0}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ q_{0,0}^{(d_I)} & q_{1,0}^{(d_I)} & \dots & q_{s,0}^{(d_I)} \end{pmatrix}. \tag{4.15}$$

The rank of any matrix $\mathbf{A} \in \mathbb{F}_{q^m}^{l \times n}$ satisfies $\mathrm{rk}(\mathbf{A}^{[i]}) = \mathrm{rk}(\mathbf{A})$ for any integer i since the q-power on the whole matrix is a linear operation. The matrix $\mathbf{Q}$ of the root finding step (4.14) contains a lower block triangular matrix, providing the following lemma.

Lemma 4.7 (Rank of Root-Finding Matrix).
Let $\mathbf{Q}$ *be defined as in* (4.14) *and* $\mathbf{Q}_0^{[0]}$ *as in* (4.13)*. If* $\mathrm{rk}(\mathbf{Q}_0^{[0]}) = s$*, then* $\mathrm{rk}(\mathbf{Q}) = sk$*.*

Proof. This holds since $\mathbf{Q}$ contains a lower block triangular matrix with $\mathbf{Q}_0^{[0]}, \dots, \mathbf{Q}_0^{[k-1]}$ on the diagonal of the first k blocks and since $\mathrm{rk}(\mathbf{Q}_0^{[0]}) = \mathrm{rk}(\mathbf{Q}_0^{[i]})$. ∎

The $d_I \times s$ matrix $\mathbf{Q}_0^{[0]}$ can have rank s only if $d_I \geq s$, which is guaranteed for $t = \tau$ if (compare (4.12)):

$$d_I = \dim\ker(\mathbf{R}) \geq s(n-\tau+1) - \sum_{i=1}^{s} k^{(i)} - t \geq s \quad \iff \quad t \leq \frac{sn - \sum_{i=1}^{s} k^{(i)}}{(s+1)}. \tag{4.16}$$

This is equivalent to the decoding radius of joint decoding and slightly different to (4.9), which is the maximum decoding radius when we consider our algorithm as a list decoder (see Section 4.3.1).

In the following, we will show a connection between the probability that $\mathbf{Q}$ does not have full rank and that the matrix $\mathbf{R}_R$ from [LO06], see (4.2), does not have full rank.

Lemma 4.8 (Connection Between Matrices of Different Approaches).
Let $\overline{\mathbf{Q}}_0$ be defined as in (4.15) *and $\mathbf{R}_R$ as in* (4.2) *for $t = \tau = \lfloor (sn - \sum_{i=1}^{s} k^{(i)})/(s+1) \rfloor$. If $\mathrm{rk}(\overline{\mathbf{Q}}_0) < s$, then $\mathrm{rk}(\mathbf{R}_R) < n-1$.*

Proof. If $\mathrm{rk}(\overline{\mathbf{Q}}_0) < s$, then by linearly combining the $d_I \geq s$ dimensional basis of the solution space of the interpolation step, there exists a non-zero interpolation polynomial $Q(x, y_1, \dots, y_s)$, which fulfills Problem 4.1 and has the coefficients $q_{0,0} = q_{1,0} = \dots = q_{s,0} = 0$. Since $Q(x, y_1, \dots, y_s) \neq 0$ (Lemma 4.2), the interpolation matrix without the first column of each submatrix (i.e., the columns corresponding to $q_{0,0}, q_{1,0}, \dots, q_{s,0}$), denoted by $\widetilde{\mathbf{R}}$, does not have full rank.
Moreover $\mathbf{R}_R^{[1]} = \widetilde{\mathbf{R}}^T$ and hence,

$$\mathrm{rk}(\mathbf{R}_R) = \mathrm{rk}(\widetilde{\mathbf{R}}) < \sum_{i=0}^{s} \deg_q Q_i(x) = (s+1)(n-\tau) - \sum_{i=1}^{s} k^{(i)} - 1.$$

For $\tau = \left\lfloor \frac{sn - \sum_{i=1}^{s} k^{(i)}}{(s+1)} \right\rfloor$, this gives $\mathrm{rk}(\mathbf{R}_R) < n-1$. ∎

Combining the last two lemmas, we obtain the following theorem.

Theorem 4.3 (Connection Between Failure Probabilities of Different Approaches).
Assume that $\mathbf{r}^{(i)}, \forall i \in [1,s]$, consists of random elements uniformly distributed over $\mathbb{F}_{q^m}$. Let $\mathbf{R}_R$ be as in (4.2) *and $\mathbf{S}$ as in* (4.5) *for $t = \tau = \lfloor (sn - \sum_{i=1}^{s} k^{(i)})/(s+1) \rfloor$. Then, for $k = \max_i\{k^{(i)}\}$:*

$$P\big(\mathrm{rk}(\mathbf{Q}) < sk\big) \leq P\big(\mathrm{rk}(\overline{\mathbf{Q}}_0) < s\big) \leq P\big(\mathrm{rk}(\mathbf{R}_R) < n-1\big). \tag{4.17}$$

Therefore, for $\tau \geq s$:

$$P\big(\mathrm{rk}(\mathbf{Q}) < sk\big) \leq 1 - \left(1 - \frac{4}{q^m}\right)\left(1 - q^{m(s-\tau)}\right)^s.$$

If $k^{(i)} = k, \forall i \in [1,s]$, additionally $P\big(\mathrm{rk}(\mathbf{Q}) < sk\big) \leq P\big(\mathrm{rk}(\mathbf{S}) < \tau\big)$ holds.

Proof. Since $\tau = \lfloor (sn - \sum_{i=1}^{s} k^{(i)})/(s+1) \rfloor$, we obtain $d_I = s$ and $\mathrm{rk}(\overline{\mathbf{Q}}_0) = \mathrm{rk}(\mathbf{Q}_0^{[0]})$. The first inequality of (4.17) follows from Lemma 4.7 and the second from Lemma 4.8. Hence, we can bound $P\big(\mathrm{rk}(\mathbf{Q}) < sk\big)$ by the failure probability from [LO06]. Due to Lemma 4.1, the failure probability from [LO06] is the same as the failure probability of [SB10] for $k^{(i)} = k, \forall i \in [1,s]$. ∎

The assumption of random received vectors and the restriction $\tau \geq s$ follow from [Ove07, Theorem 3.11]. We conjecture that $\tau \geq s$ is only a technical restriction and that the results hold equivalently for $\tau < s$.

Alternatively, we can bound the failure probability as follows. Assume, the matrix $\mathbf{Q}_0^{[0]}$ consists of random values over $\mathbb{F}_{q^m}$. This assumption seems to be reasonable, since in [LO06] and [SB10] it is assumed that $\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(s)}$ are random vectors in $\mathbb{F}_{q^m}^n$. In our approach, the values of $\mathbf{Q}_0^{[0]}$ are obtained from a linear system of equations, where each $q_{i,0}$ is multiplied with the coefficients of a different $\mathbf{r}^{(i)}$.

Lemma 4.9 (Alternative Calculation of Failure Probability).
Let $\mathrm{rk}\left(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T}\right) = t \leq \tau$, *where* $\tau = \left\lfloor (sn - \sum_{i=1}^{s} k^{(i)})/(s+1) \right\rfloor$, *let* $k = \max_i\{k^{(i)}\}$, *let* $\mathbf{Q}$ *be defined as in* (4.14) *and let* $q_{1,0}^{(j)}, q_{2,0}^{(j)}, \dots, q_{s,0}^{(j)}$ *for* $j = 1, \dots, d_I$ *be random elements uniformly distributed over* $\mathbb{F}_{q^m}$. *Then,*

$$P\big(\mathrm{rk}(\mathbf{Q}) < sk\big) \leq \frac{4}{q^{(m(d_I+1-s))}} = 4q^{-m\left(s(n-\tau)-\sum_{i=1}^{s} k^{(i)}-t+1\right)}.$$

Proof. Due to $d_I \geq s$ and Lemma 4.7, if $\mathrm{rk}(\mathbf{Q}_0^{[0]}) = s$, then $\mathrm{rk}(\mathbf{Q}) = sk$. Hence, $P\left(\mathrm{rk}(\mathbf{Q}) < sk\right) \leq P(\mathrm{rk}(\mathbf{Q}_0^{[0]}) < s)$. When $q_{1,0}^{(j)}, \dots, q_{s,0}^{(j)}$ for $j = [1, d_I]$ are random elements from $\mathbb{F}_{q^m}$, we bound $P(\mathrm{rk}(\mathbf{Q}_0^{[0]}) < s)$ by the probability that a random $(d_I \times s)$-matrix over $\mathbb{F}_{q^m}$ has rank less than s:

$$\begin{aligned} P\big(\mathrm{rk}(\mathbf{Q}) < sk\big) &\leq P\big(\mathrm{rk}(\mathbf{Q}_0^{[0]}) < s\big) \\ &\leq \frac{\sum\limits_{j=0}^{s-1} \prod\limits_{h=0}^{j-1} \frac{q^{d_I}-q^h}{q^j-q^h} \prod\limits_{i=0}^{j-1} (q^s - q^i)}{q^{msd_I}} \\ &< \frac{4q^{m\left((d_I+s)(s-1)-(s-1)^2\right)}}{q^{msd_I}} \\ &= \frac{4}{q^{m(d_I-s+1)}} = 4q^{-m\left(s(n-\tau)-\sum_{i=1}^{s} k^{(i)}-t+1\right)}. \end{aligned}$$

■

Lemma 4.9 does not have the technical restriction $\tau \geq s$ as Theorem 4.3 and the bounds from [LO06, SB10]. The following theorem summarizes our results.

Theorem 4.4 (Unique Decoding of Interleaved Gabidulin Codes).
Let the interleaved Gabidulin code $\mathsf{IGab}[s; n, k^{(1)}, \dots, k^{(s)}]$ *over* $\mathbb{F}_{q^m}$ *consist of the elementary codewords* $\mathbf{c}^{(i)} = f^{(i)}(\mathbf{g})$, *where* $\deg_q f^{(i)}(x) < k^{(i)}$, $\forall i \in [1, s]$, *and let the given elementary received words* $\mathbf{r}^{(i)}$, $\forall i \in [1, s]$, *consist of random elements uniformly distributed over* $\mathbb{F}_{q^m}$. *Then, with probability at least*

$$1 - 4q^{-m\left(s(n-\tau)-\sum_{i=1}^{s} k^{(i)}-t+1\right)},$$

we can find a unique solution $f^{(1)}(x), \dots, f^{(s)}(x)$ *such that its evaluation at* $\mathbf{g}$ *is in rank distance*

$$t \leq \tau = \left\lfloor \frac{sn - \sum_{i=1}^{s} k^{(i)}}{s+1} \right\rfloor$$

to $\left(\mathbf{r}^{(1)T}\ \mathbf{r}^{(2)T}\ \dots\ \mathbf{r}^{(s)T}\right)^T$ *with overall complexity at most* $\mathcal{O}(s^3n^2)$.

Example 4.2 (Failure Probabilities).
Consider the $\mathsf{IGab}[s = 2; n = 7, k^{(1)} = 2, k^{(2)} = 2]$ *code over* $\mathbb{F}_{2^7}$*. The maximum decoding radius for unique as well as for list decoding according to* (4.9) *and* (4.16) *is* $\tau = 3$ *whereas a BMD decoder guarantees to correct all errors of rank at most* $\tau_0 = 2$.

In order to estimate the failure probability, we simulated 10^7 *random error matrices* $(\mathbf{e}^{(1)T}\ \mathbf{e}^{(2)T}\ \dots\ \mathbf{e}^{(s)T})^T \in \mathbb{F}_{q^m}^{s\times n}$*, uniformly distributed over all matrices of rank* $t = \tau = 3$*. The following simulated probabilities occurred:*

$$P\big(\operatorname{rk}(\mathbf{Q}) < sk\big) = P\big(\operatorname{rk}(\mathbf{S}) < \tau\big) = P\big(\operatorname{rk}(\mathbf{R}_R) < n-1\big) = 6.12 \cdot 10^{-5}.$$

As a comparison, the average list size calculated with Lemma 4.6 is $\overline{\ell_I} < 1 + 6.104 \cdot 10^{-5}$*, the upper bound from Theorem 4.3 (and therefore the upper bound from* (4.4)*, [LO06]) gives*

$$P\big(\operatorname{rk}(\mathbf{Q}) < sk\big) \leq P\big(\operatorname{rk}(\mathbf{R}_R) < n-1\big) \leq 0.04632,$$

and the bound from Lemma 4.9 gives $P\big(\operatorname{rk}(\mathbf{Q}) < sk\big) \leq 4q^{-m(s(n-k-\tau)-\tau+1)} = 2.44 \cdot 10^{-4}$.

Due to the simulation results, we conjecture that *if and only if* $\operatorname{rk}(\mathbf{Q}) < sk$, then $\operatorname{rk}(\mathbf{S}) < \tau$ and $\operatorname{rk}(\mathbf{R}_R) < n-1$. Hence, we believe that Lemma 4.8 holds in *both* directions.

4.4 Error-Erasure Decoding

This section is in some sense a generalization of Subsection 3.2.3 and outlines briefly how interpolation-based error-erasure decoding of *interleaved* Gabidulin codes over $\mathbb{F}_{q^m}$ with $n = m$ can be done. We assume that γ column erasures and $\varrho^{(i)}$ row erasures, $\forall i \in [1, s]$, occurred. This notation is based on the following decomposition of the interleaved error, which generalizes (3.34):

$$\begin{pmatrix} \mathbf{e}^{(1)} \\ \mathbf{e}^{(2)} \\ \vdots \\ \mathbf{e}^{(s)} \end{pmatrix} = \begin{pmatrix} \mathbf{a}^{(1,R)} \cdot \mathbf{B}^{(1,R)} \\ \mathbf{a}^{(2,R)} \cdot \mathbf{B}^{(2,R)} \\ \vdots \\ \mathbf{a}^{(s,R)} \cdot \mathbf{B}^{(s,R)} \end{pmatrix} + \begin{pmatrix} \mathbf{a}^{(1,C)} \\ \mathbf{a}^{(2,C)} \\ \vdots \\ \mathbf{a}^{(s,C)} \end{pmatrix} \cdot \mathbf{B}^{(C)} + \begin{pmatrix} \mathbf{a}^{(1,E)} \\ \mathbf{a}^{(2,E)} \\ \vdots \\ \mathbf{a}^{(s,E)} \end{pmatrix} \cdot \mathbf{B}^{(E)} \in \mathbb{F}_{q^m}^{s\times n}, \tag{4.18}$$

where $\mathbf{a}^{(i,R)} \in \mathbb{F}_{q^m}^{\varrho^{(i)}}$, $\mathbf{B}^{(i,R)} \in \mathbb{F}_q^{\varrho^{(i)}\times n}$, $\mathbf{a}^{(i,C)} \in \mathbb{F}_{q^m}^{\gamma}$, $\mathbf{B}^{(C)} \in \mathbb{F}_q^{\gamma\times n}$, $\mathbf{a}^{(i,E)} \in \mathbb{F}_{q^m}^{t}$, $\mathbf{B}^{(E)} \in \mathbb{F}_q^{t\times n}$ for all $i \in [1, s]$, and $\mathbf{a}^{(1,R)}, \mathbf{a}^{(2,R)}, \dots, \mathbf{a}^{(s,R)}$ and $\mathbf{B}^{(C)}$ are known on the receiver side. Lemma 4.10 shows later why the $\mathbf{a}^{(i,R)}$ and $\mathbf{B}^{(i,R)}$ can be different whereas $\mathbf{B}^{(C)}$ has to be common for all $i \in [1, s]$. Moreover, this model of errors and erasures is slightly more general than the one in [LSC13, Equation (19)], since there the $\mathbf{a}^{(i,R)}$ are assumed to be equal.

Based on the known matrix $\mathbf{B}^{(C)}$ and as in (3.35), we can calculate the following basis of the row space of the column erasures prior to the decoding process:

$$d_i^{(C)} = \sum_{j=0}^{n-1} B_{i,j}^{(C)} g_j^{\perp} = \sum_{j=0}^{n-1} B_{i,j}^{(C)} \beta^{[j]}, \quad \forall i \in [0, \gamma-1]. \tag{4.19}$$

As in (3.36), we define $\Gamma^{(C)}(x)$ and $\Lambda^{(i,R)}(x)$, $\forall i \in [1, s]$, as linearized polynomials of smallest q-degree such that:

$$\begin{aligned} \Gamma^{(C)}\big(d_j^{(C)}\big) &= 0, \quad \forall j \in [0, \gamma-1], \\ \Lambda^{(i,R)}\big(a_j^{(i,R)}\big) &= 0, \quad \forall j \in [0, \varrho^{(i)}-1],\ i \in [1, s]. \end{aligned} \tag{4.20}$$

Let $\widehat{r}^{(i)}(x)$ denote the q-transform of $r^{(i)}(x)$, $\forall i \in [1,s]$, as in Definition 2.12, then we define s modified transformed received words (similar to Subsection 3.2.3) by:

$$\widehat{y}^{(i)}(x) \overset{\text{def}}{=} \Lambda^{(i,R)}\big(\widehat{r}^{(i)}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big) \mod (x^{[m]} - x), \quad \forall i \in [1,s],$$

where $\overline{\Gamma^{(C)}}(x)$ is the full q-reverse of $\Gamma^{(C)}(x)$ as in Lemma A.1.

Lemma 4.10 (Rank of Modified Interleaved Error).
Let $n = m$ and let $\mathbf{e}^{(i,RC)} = \Lambda^{(i,R)}\big(\widehat{e}^{(i)}(\overline{\Gamma^{(C)}}(\mathbf{g}^{[\gamma]}))\big) \in \mathbb{F}_{q^m}^n$, $\forall i \in [1,s]$. Further, let $\mathbf{e}^{(i,E)} = \mathbf{a}^{(i,E)} \cdot \mathbf{B}^{(E)}$, $\forall i \in [1,s]$, as in (4.18) with $\operatorname{rk}\big(\mathbf{e}^{(1,E)T}\ \mathbf{e}^{(2,E)T}\ \dots\ \mathbf{e}^{(s,E)T}\big) = t$. Then,

$$\operatorname{rk}\begin{pmatrix}\mathbf{e}^{(1,RC)}\\ \mathbf{e}^{(2,RC)}\\ \vdots\\ \mathbf{e}^{(s,RC)}\end{pmatrix} \leq \operatorname{rk}\begin{pmatrix}\mathbf{e}^{(1,E)}\\ \mathbf{e}^{(2,E)}\\ \vdots\\ \mathbf{e}^{(s,E)}\end{pmatrix} = t.$$

Proof. The proof is a straight-forward generalization of the proof of Lemma 3.11 and we obtain:

$$\begin{pmatrix}\mathbf{e}^{(1,RC)}\\ \mathbf{e}^{(2,RC)}\\ \vdots\\ \mathbf{e}^{(s,RC)}\end{pmatrix} = \begin{pmatrix}\Lambda^{(1,R)}\big(\widehat{e}^{(1,E)}(g_0)\big) & \Lambda^{(1,R)}\big(\widehat{e}^{(1,E)}(g_1)\big) & \dots & \Lambda^{(1,R)}\big(\widehat{e}^{(1,E)}(g_{m-1})\big)\\ \Lambda^{(2,R)}\big(\widehat{e}^{(2,E)}(g_0)\big) & \Lambda^{(2,R)}\big(\widehat{e}^{(2,E)}(g_1)\big) & \dots & \Lambda^{(2,R)}\big(\widehat{e}^{(2,E)}(g_{m-1})\big)\\ \vdots & \vdots & \ddots & \vdots\\ \Lambda^{(s,R)}\big(\widehat{e}^{(s,E)}(g_0)\big) & \Lambda^{(s,R)}\big(\widehat{e}^{(s,E)}(g_1)\big) & \dots & \Lambda^{(s,R)}\big(\widehat{e}^{(s,E)}(g_{m-1})\big)\end{pmatrix} \cdot \mathbf{G},$$

where $\mathbf{G} = \big(G_{i,j}\big)_{j\in[0,m-1]}^{i\in[0,m-1]} \in \mathbb{F}_q^{m\times m}$ is defined such that $\overline{\Gamma^{(C)}}(g_j) = \sum_{i=0}^{m-1} G_{i,j} g_i$ and the statement follows as in Lemma 3.11. ∎

Lemma 4.10 requires that $\Gamma^{(C)}(x)$ is common for all $i \in [1,s]$, whereas $\Lambda^{(i,R)}(x)$ can be different. This clarifies why $\mathbf{B}^{(C)}$ has to be independent of i.

Hence, similar to error-erasure decoding of Gabidulin codes in Subsection 3.2.3, we use $\widehat{y}^{(i)}(\mathbf{g})$, $\forall i \in [1,s]$, as the input of interpolation-based decoding and treat $\widehat{y}^{(i)}(\mathbf{g})$ in the same way as the transform of a codeword of an interleaved Gabidulin code of elementary dimensions $k^{(i)} + \varrho^{(i)} + \gamma$, which is corrupted by an error of overall rank t.

As in Problem 4.1, we look for an $(s+1)$-variate linearized polynomial $Q(x, y_1, \dots, y_s) = Q_0(x) + Q_1(y_1) + \dots + Q_s(y_s)$, which satisfies for given integers n, τ, $k^{(i)}$, $\varrho^{(i)}$, γ, $\forall i \in [1,s]$:

- $Q(g_j, \widehat{y}^{(1)}(g_j), \widehat{y}^{(2)}(g_j), \dots, \widehat{y}^{(s)}(g_j)) = 0, \quad \forall j \in [0, n-1]$,
- $\deg_q Q_0(x) < n - \tau$,
- $\deg_q Q_i(y_i) < n - \tau - (k^{(i)} - \gamma - \varrho^{(i)} - 1), \quad \forall i \in [1,s]$.

Similar to Lemma 4.2, a non-zero interpolation polynomial $Q(x, y_1, \dots, y_s)$, which satisfies the above mentioned conditions, exists if

$$\tau < \frac{sn - \sum_{i=1}^{s}(k^{(i)} + \varrho^{(i)} + \gamma) + s}{s+1}.$$

If $k^{(i)} = k$, and $\varrho^{(i)} = \varrho$, $\forall i \in [1,s]$, we obtain $\tau < s(n-k+1-\varrho-\gamma)/(s+1)$.

The interpolation and root-finding procedure is straight forward to the errors-only approach from Section 4.2 and returns $\Lambda^{(i,R)}\big(f^{(i)}(\overline{\Gamma^{(C)}}(x^{[\gamma]}))\big)$, $\forall i \in [1,s]$, in rank distance at most τ. In order to obtain $f^{(i)}(x)$, we have to divide from the left and right by $\Lambda^{(i,R)}(x)$ and $\overline{\Gamma^{(C)}}(x^{[\gamma]})$, respectively, $\forall i \in [1, s]$, as in Subsection 3.2.3.

With this principle, our interpolation-based decoding algorithm can be applied to (unique or list) error-erasure decoding of interleaved Gabidulin codes.

For interpolation-based error-erasure decoding in Hamming metric it is more common to puncture the code at the erased positions and interpolate an (interleaved) code of smaller length and same dimension(s) as the original code, whereas we interpolate a code, which has the same length as the original code, but higher dimension(s).

4.5 Summary and Outlook

This chapter considers decoding approaches for interleaved Gabidulin codes. First, two known decoding principles are described and a relation between them is proven. Second, we have presented a new approach for decoding interleaved Gabidulin codes based on interpolating a multi-variate linearized polynomial. The procedure consists of two steps: an interpolation step and a root-finding step, where both can be accomplished by solving a linear system of equations. This new decoder for interleaved Gabidulin codes can be used as a list decoder as well as a unique decoder with a certain failure probability. The complexity of the unique decoder as well as finding a basis of all solutions of the list decoder is quadratic in the length of the code. The output of both decoders is a unique decoding result with high probability. Further, we have derived a connection to the two known approaches for decoding interleaved Gabidulin codes. This relation provides an upper bound on the failure probability of our unique decoder.

For future work, the big challenge is to increase the decoding radius (for Gabidulin as well as interleaved Gabidulin codes). Nearby goals are to apply re-encoding in order to reduce the complexity and to use subspace evasive subsets for the elimination of the valid solutions in the list decoder as Guruswami and Wang did in [GW13].

CHAPTER 5

Bounds on List Decoding of Block Codes in Rank Metric

The idea of list decoding was introduced by Elias [Eli57] and Wozencraft [Woz58] stating that a *list decoder* returns the list of *all* codewords in distance at most τ from any given word. The *Johnson bound* in Hamming metric [Joh62, Bas65, Gur99] shows that for *any* code of length n and minimum Hamming distance d_H, the size of this list is polynomial in n for any τ less than the Johnson radius, i.e., $\tau < \tau_J = n - \sqrt{n(n-d_H)}$. Although this fact has been known since the 1960s, a polynomial-time list decoding algorithm for Reed–Solomon codes up to the Johnson radius was found not earlier than 1999 by Guruswami and Sudan [GS99] as a generalization of the Sudan algorithm [Sud97]. Moreover, in Hamming metric, it can be shown that there exists a code such that the list size becomes *exponential* in n beyond the Johnson radius [GRS00], [Gur99, Chapter 4]. It is not known whether such an exponential list beyond the Johnson radius also exists for Reed–Solomon codes. Several publications show an exponential behavior of the list size for Reed–Solomon codes only for a radius rather greater than τ_J (see e.g. Justesen and Høholdt [JH01] and Ben-Sasson, Kopparty and Radhakrishnan [BKR10]).

However, for Gabidulin codes, so far there exists *no* polynomial-time list decoding algorithm (beyond half the minimum distance) and it is not even known whether it can exist or not. The contributions by Mahdavifar and Vardy [MV10, MV12] and by Guruswami and Xing [GX12] provide list decoding algorithms for special classes of Gabidulin codes and subcodes of Gabidulin codes.

In this chapter, we investigate bounds on list decoding rank-metric codes in general and Gabidulin codes in particular in order to understand if polynomial-time list decoding algorithms can exist or not. We derive three bounds on the maximum list size when decoding rank-metric codes. In spite of the numerous similarities between Hamming metric and rank metric and even more between Reed–Solomon and Gabidulin codes, all three bounds reveal a strongly different behavior compared to Hamming metric.

On the one hand, a lower bound on the maximum list size, which is exponential in the length n of the code, rules out the possibility of polynomial-time list decoding since already writing down the list has exponential complexity. On the other hand, a polynomial upper bound—similar to the Johnson bound for Hamming metric—shows that a polynomial-time list decoding algorithm might exist.

In Section 5.1, we state (partly informally) known bounds on list decoding in Hamming metric. In Section 5.2, we explain connections between constant-dimension codes, constant-rank codes and the list of codewords and state the problem. We derive a lower bound (Bound I) for Gabidulin codes of length n and minimum rank distance d in Section 5.3. It proves that the list size can become exponential if the radius is at least the Johnson radius $\tau_J = n - \sqrt{n(n-d)}$. The second bound (Bound II, Section 5.4) is an exponential upper bound for any rank-metric code, which provides no conclusion about polynomial-time list decodability. Finally, in Section 5.5, the third bound (Bound III) shows that there exists a rank-metric code over $\mathbb{F}_{q^m}$ of length $n \leq m$ such that the list size is exponential in the length n when the decoding radius is greater than *half the minimum distance.* An interpretation of our bounds and a comparison to bounds on list decoding in Hamming metric is shown in Section 5.6.

The bound presented in Section 5.3 was published in [Wac12] and the bounds from Sections 5.4 and 5.5 in [Wac13b]. The journal paper [Wac13a] contains a detailed description of all three bounds.

5.1 Known Bounds on the List Size for Codes in Hamming Metric

Without going into depth, we want to state some of the bounds on list decoding of codes in Hamming metric, in particular Reed–Solomon codes.

Definition 5.1 (Hamming Weight and Hamming Distance).
The Hamming weight of $\mathbf{a} = (a_0\ a_1\ \dots\ a_{n-1}) \in \mathbb{F}_q^n$ *is defined as*

$$\mathrm{wt}_H(\mathbf{a}) \overset{\text{def}}{=} |\operatorname{supp}(\mathbf{a})| \overset{\text{def}}{=} |a_i \neq 0, i \in [0, n-1]|,$$

and the Hamming distance between $\mathbf{a}$ *and* $\mathbf{b} \in \mathbb{F}_q^n$ *is the Hamming weight of the difference:*

$$d_H(\mathbf{a}, \mathbf{b}) \overset{\text{def}}{=} \mathrm{wt}_H(\mathbf{a} - \mathbf{b}) = |\operatorname{supp}(\mathbf{a} - \mathbf{b})|.$$

In conformance with the notations for codes in rank metric, an $(n, M, d)_H$ code C over $\mathbb{F}_q$ is a code of length n, cardinality M and minimum *Hamming* distance d.

The q-ary and the alphabet-independent *Johnson bounds* in Hamming metric (stated in the following) show that from a combinatorial point of view, list decoding of any (not necessarily linear) code in Hamming metric is feasible up to the Johnson radius. A thorough discussion of the Johnson bound and related combinatorial aspects can be found in Guruswami's books [Gur07, Chapter 3] and [Gur99, Chapter 3].

Theorem 5.1 (*q*-ary Johnson Bound in Hamming Metric [Joh62, Joh63, Bas65]).
For any $(n, M, d)_H$ *code* C *over* $\mathbb{F}_q$ *of length* n *and minimum Hamming distance* d *and any integer* τ *such that* $\tau^2 > (1 - {}^1\!/_q)\,(2\tau - d)n$*, the list size* ℓ_H *is upper bounded by*

$$\ell_H \overset{\text{def}}{=} \max_{\mathbf{r} \in \mathbb{F}_q^n} \left\{ \left| \mathsf{C} \cap \mathcal{B}_H^{(\tau)}(\mathbf{r}) \right| \right\} \leq \frac{(1 - {}^1\!/_q)\, nd}{\tau^2 - (1 - {}^1\!/_q)\,(2\tau - d)n}, \tag{5.1}$$

where $\mathcal{B}_H^{(\tau)}(\mathbf{r})$ *denotes a ball around* $\mathbf{r}$ *of radius* τ *in Hamming metric.*

The *alphabet-independent* (or *generic*) Johnson bound (5.2) follows from upper bounding (5.1) for any $q > 1$. Clearly, when q is large, then $(1 - {}^1\!/_q) \to 1$ and the two bounds are equivalent. However, for small q (especially for binary codes), it can be much better to take into account the alphabet size and to use (5.1).

Corollary 5.1 (Johnson Bound in Hamming Metric).
For any $(n, M, d)_H$ *code* C *over* $\mathbb{F}_q$ *of length* n *and minimum Hamming distance* d *and any integer* $\tau < \tau_J \overset{\text{def}}{=} n - \sqrt{n(n-d)}$*, the list size* ℓ_H *is upper bounded by*

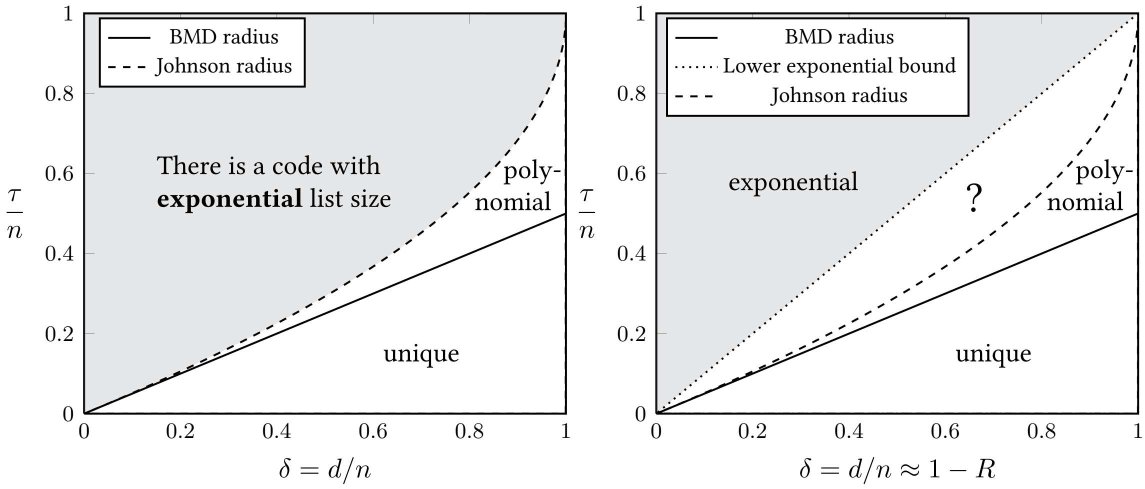
$$\ell_H \overset{\text{def}}{=} \max_{\mathbf{r} \in \mathbb{F}_q^n} \left\{ \left| \mathsf{C} \cap \mathcal{B}_H^{(\tau)}(\mathbf{r}) \right| \right\} \leq \frac{nd}{\tau^2 - (2\tau - d)n} = \frac{nd}{(n-\tau)^2 - n(n-d)}. \tag{5.2}$$

The restriction $\tau < \tau_J$ holds since the denominator has to be greater than zero. An improvement of the numerator of (5.2) from nd to $n(d - \tau)$ was shown by Cassuto and Bruck in [CB04].

Thus, the Johnson bound proves that *any* ball in Hamming metric of radius less than the Johnson radius $\tau_J = n - \sqrt{n(n-d)}$ always contains a *polynomial* number of codewords of *any* code in Hamming metric of length n and minimum Hamming distance d.

From a combinatorial point of view, the Johnson bound is tight as a relation between list decodability and the minimum Hamming distance since it can be shown that there exist codes in Hamming metric such that the list size becomes *exponential* in n if the radius is slightly greater than the Johnson radius. For general (not necessarily linear) codes this was shown by Goldreich, Rubinfeld and Sudan [GRS00]. Guruswami extended this result to *linear* codes, stating that there exists a *linear* code in Hamming metric such that the size of the list grows super-polynomially in n when the radius of the ball is at least the Johnson radius [Gur99, Chapter 4]. However, he proved this only using a widely-accepted number theoretic conjecture [Gur99, Theorem 4.7]. These results do not imply that the Johnson bound is tight for any code in Hamming metric—it rather means that there are some codes for which it is tight.

In particular, it is not known whether such an exponential list slightly beyond the Johnson radius also exists for Reed–Solomon codes. Justesen and Høholdt [JH01] and Ben-Sasson, Kopparty and Radhakrishnan [BKR10] showed an exponential behavior of the list size for Reed–Solomon codes only for a radius rather greater than τ_J. However, Guruswami and Rudra's limits to list *recovery* (which is a more general scenario) of Reed–Solomon codes indicate that the Johnson bound might be tight also for Reed–Solomon codes [GR06].



(a) General codes in Hamming metric

(b) Reed–Solomon codes

Figure 5.1. Decoding regions of codes in Hamming metric

Figure 5.1 illustrates the asymptotic behavior of the list size depending on the relative distance $\delta = d/n$. The existence of a code in Hamming metric with exponential list size beyond the Johnson radius is shown in Figure 5.1a. For Reed–Solomon (and maximum distance separable codes in general), the relative distance is $\delta = d/n = 1 - R + 1/n$ and for large lengths, $\delta \approx 1 - R$ and therefore the list size is displayed in dependency of the code rate in Figure 5.1b.

5.2 Codes Connected to the List of Decoding and Problem Statement

This section shows relations between constant-dimension and constant-rank codes, states the problem and shows a connection between constant-rank codes and the resulting list when list decoding codes in rank metric. These relations are used for our bounds in Sections 5.4 and 5.5.

5.2.1 Connection between Constant-Dimension and Constant-Rank Codes

This section recalls and generalizes some of the connections between constant-dimension and constant-rank codes by Gadouleau and Yan [GY10]. The first lemma shows a connection between the subspace distance and the rank distance and is a special case of [GY10, Theorem 1].

Lemma 5.1 (Connection between Subspace and Rank Distance [GY10, Theorem 1]). *Let* $\mathbf{X}, \mathbf{Y} \in \mathbb{F}_q^{m \times n}$ *with* $\mathrm{rk}(\mathbf{X}) = \mathrm{rk}(\mathbf{Y})$. *Then:*

$$\begin{aligned}
&\frac{1}{2}\, d_s\big(\mathcal{R}_q\left(\mathbf{X}\right), \mathcal{R}_q\left(\mathbf{Y}\right)\big) + \frac{1}{2}\, d_s\big(\mathcal{C}_q\left(\mathbf{X}\right), \mathcal{C}_q\left(\mathbf{Y}\right)\big)\\
&\leq d_R(\mathbf{X}, \mathbf{Y})\\
&\leq \min\left\{\frac{1}{2}\, d_s\big(\mathcal{R}_q\left(\mathbf{X}\right), \mathcal{R}_q\left(\mathbf{Y}\right)\big), \frac{1}{2}\, d_s\big(\mathcal{C}_q\left(\mathbf{X}\right), \mathcal{C}_q\left(\mathbf{Y}\right)\big)\right\} + \mathrm{rk}(\mathbf{X}).
\end{aligned}$$

Proof. Let us denote $r \overset{\text{def}}{=} \mathrm{rk}(\mathbf{X}) = \mathrm{rk}(\mathbf{Y})$. As in Lemma 3.8, we decompose $\mathbf{X} = \mathbf{C}^T\mathbf{R}$ and $\mathbf{Y} = \mathbf{D}^T\mathbf{S}$, where $\mathbf{C}, \mathbf{D} \in \mathbb{F}_q^{r \times m}$ and $\mathbf{R}, \mathbf{S} \in \mathbb{F}_q^{r \times n}$ and all four matrices have full rank. Hence, $\mathbf{X} - \mathbf{Y} = (\mathbf{C}^T | - \mathbf{D}^T) \cdot (\mathbf{R}^T | \mathbf{S}^T)^T$. In general, it is well-known that $\mathrm{rk}(\mathbf{AB}) \leq \min\{\mathrm{rk}(\mathbf{A}), \mathrm{rk}(\mathbf{B})\}$ and $\mathrm{rk}(\mathbf{AB}) \geq \mathrm{rk}(\mathbf{A}) + \mathrm{rk}(\mathbf{B}) - n$ when $\mathbf{A}$ has n columns and $\mathbf{B}$ has n rows. Therefore,

$$\begin{aligned}
\mathrm{rk}(\mathbf{C}^T | - \mathbf{D}^T) + \mathrm{rk}(\mathbf{R}^T | \mathbf{S}^T) - 2r \leq \mathrm{rk}(\mathbf{X} - \mathbf{Y}) &= \mathrm{rk}\left((\mathbf{C}^T | - \mathbf{D}^T) \cdot (\mathbf{R}^T | \mathbf{S}^T)^T\right) \qquad (5.3)\\
&\leq \min\left\{\mathrm{rk}(\mathbf{C}^T | - \mathbf{D}^T), \mathrm{rk}(\mathbf{R}^T | \mathbf{S}^T)\right\}.
\end{aligned}$$

Let $\mathcal{C}_q\left(\mathbf{C}^T\right) + \mathcal{C}_q\left(\mathbf{D}^T\right)$ denote the smallest subspace containing both column spaces. Then,

$$\begin{aligned}
\mathrm{rk}(\mathbf{C}^T | - \mathbf{D}^T) &= \dim(\mathcal{C}_q\left(\mathbf{C}^T\right) + \mathcal{C}_q\left(\mathbf{D}^T\right))\\
&= \dim(\mathcal{C}_q\left(\mathbf{C}^T\right) + \mathcal{C}_q\left(\mathbf{D}^T\right)) - \frac{1}{2}\left\{\dim(\mathcal{C}_q\left(\mathbf{C}^T\right)) + \dim(\mathcal{C}_q\left(\mathbf{D}^T\right))\right\}\\
&\qquad + \frac{1}{2}\left\{\dim(\mathcal{C}_q\left(\mathbf{C}^T\right)) + \dim(\mathcal{C}_q\left(\mathbf{D}^T\right))\right\}\\
&= \frac{1}{2}\, d_s(\mathcal{C}_q\left(\mathbf{C}^T\right), \mathcal{C}_q\left(\mathbf{D}^T\right)) + r = \frac{1}{2}\, d_s(\mathcal{C}_q\left(\mathbf{X}\right), \mathcal{C}_q\left(\mathbf{Y}\right)) + r,
\end{aligned}$$

and in the same way

$$\mathrm{rk}(\mathbf{R}^T | \mathbf{S}^T) = \frac{1}{2}\, d_s(\mathcal{R}_q\left(\mathbf{X}\right), \mathcal{R}_q\left(\mathbf{Y}\right)) + r.$$

Inserting this into (5.3), the statement follows. ∎

Lemma 5.1 can equivalently be derived using [MS74, Equation (4.3)], which also results in (5.3) and then we can use the same reformulations for the subspace distance.

For the proof of the upper bound in Theorem 5.3 (see Section 5.4), the following upper bound on the maximum cardinality of a constant-rank code is applied. It shows a relation between the maximum cardinalities of a (not necessarily linear) constant-rank and a constant-dimension code.

Proposition 5.1 (Maximum Cardinality [GY10]).
For all q and $1 \leq \delta \leq r \leq n \leq m$, the maximum cardinality of a $\mathsf{CR}_{q^m}(n, M, d_R = \delta + r, r)$ constant-rank code over $\mathbb{F}_{q^m}$ is upper bounded by the maximum cardinality of a constant-dimension code as follows:

$$\mathsf{A}^R_{q^m}\left(n, d_R = \delta + r, r\right) \leq \mathsf{A}^S_q\left(n, d_s = 2\delta, r\right).$$

However, the connections between constant-dimension and constant-rank codes are even more far-reaching. The following proposition shows explicitly how to construct constant-rank codes out of constant-dimension codes and is a generalization of [GY10, Proposition 3] to arbitrary cardinalities.

Proposition 5.2 (Construction of a Constant-Rank Code).
Let M be a $\mathsf{CD}_q(m, |\mathsf{M}|, d_{s,M}, r)$ and N be a $\mathsf{CD}_q(n, |\mathsf{N}|, d_{s,N}, r)$ constant-dimension code with $r \leq \min\{n, m\}$ and cardinalities $|\mathsf{M}|$ and $|\mathsf{N}|$. Then, there exists a $\mathsf{CR}_{q^m}(n, M_R, d_R, r)$ constant-rank code C of cardinality $M_R = \min\{|\mathsf{M}|, |\mathsf{N}|\}$ with $\mathcal{C}_q\left(\mathsf{C}\right) \subseteq \mathsf{M}$ and $\mathcal{R}_q\left(\mathsf{C}\right) \subseteq \mathsf{N}$. Further, the minimum rank distance d_R of C is

$$d_R \geq \frac{1}{2}\, d_{s,M} + \frac{1}{2}\, d_{s,N},$$

and if $|\mathsf{M}| = |\mathsf{N}|$ additionally:

$$d_R \leq \frac{1}{2} \min\left\{d_{s,M}, d_{s,N}\right\} + r.$$

Proof. Let $\mathbf{G}_i \in \mathbb{F}_q^{r \times m}$ and $\mathbf{H}_i \in \mathbb{F}_q^{r \times n}$, $\forall i \in [1, \min\{|\mathsf{M}|, |\mathsf{N}|\}]$, be full-rank matrices, whose row spaces are $\min\{|\mathsf{M}|, |\mathsf{N}|\}$ codewords (which are subspaces themselves) of M and N, respectively. Let C be a $\mathsf{CR}_{q^m}(n, M_R, d_R, r_R)$ constant-rank code, defined by the set of codewords $\mathbf{A}_i = \mathbf{G}_i^T \mathbf{H}_i$, $\forall i \in [1, \min\{|\mathsf{M}|, |\mathsf{N}|\}]$. All such codewords $\mathbf{A}_i$ are distinct, since the row spaces of all $\mathbf{G}_i$, respectively $\mathbf{H}_i$, are different. These codewords $\mathbf{A}_i$ are $m \times n$ matrices of rank exactly $r_R = r$ since $\mathbf{G}_i \in \mathbb{F}_q^{r \times m}$ and $\mathbf{H}_i \in \mathbb{F}_q^{r \times n}$ have rank r. The cardinality is $|\mathsf{C}| = M_R = \min\{|\mathsf{M}|, |\mathsf{N}|\}$ and $\mathcal{C}_q\left(\mathsf{C}\right) \subseteq \mathsf{M}$ and $\mathcal{R}_q\left(\mathsf{C}\right) \subseteq \mathsf{N}$ by Lemma 3.8.
The lower bound on the minimum rank distance follows from Lemma 5.1 for different $\mathbf{A}_i, \mathbf{A}_j$:

$$d_R \geq \frac{1}{2}\, d_s(\mathcal{R}_q\left(\mathbf{A}_i\right), \mathcal{R}_q\left(\mathbf{A}_j\right)) + \frac{1}{2}\, d_s(\mathcal{C}_q\left(\mathbf{A}_i\right), \mathcal{C}_q\left(\mathbf{A}_j\right)) \geq \frac{1}{2}\, d_{s,N} + \frac{1}{2}\, d_{s,M}.$$

If $|\mathsf{M}| = |\mathsf{N}|$, there exist two matrices $\mathbf{A}_i, \mathbf{A}_j$ such that $d_s(\mathcal{R}_q\left(\mathbf{A}_i\right), \mathcal{R}_q\left(\mathbf{A}_j\right)) = d_{s,N}$. Then, Lemma 5.1 gives $d_R \leq d_{s,N} + r$. If we choose $\mathbf{A}_i$ and $\mathbf{A}_j$ such that $d_s(\mathcal{C}_q\left(\mathbf{A}_i\right), \mathcal{C}_q\left(\mathbf{A}_j\right)) = d_{s,M}$, then $d_R \leq d_{s,M} + r$ and the upper bound on the rank distance follows. ∎

5.2.2 Problem Statement

We analyze the question of *polynomial-time list decodability* of rank-metric codes. Thus, we want to bound the maximum number of codewords in a ball of radius τ around a received word $\mathbf{r}$. This number will be called the maximum *list size* ℓ in the following. The worst-case complexity of a possible list decoding algorithm directly depends on ℓ.

Problem 5.1 (Maximum List Size).
Let C be an $(n, M, d)_R$ code over $\mathbb{F}_{q^m}$ of length $n \leq m$, cardinality M and minimum rank distance $d_R = d$. Let $\tau < d$. Find lower and upper bounds on the maximum number of codewords ℓ in a ball of

rank radius τ around a word $\mathbf{r} = (r_0\; r_1\; \dots\; r_{n-1}) \in \mathbb{F}_{q^m}^n$. Hence, find bounds on

$$\ell \overset{\text{def}}{=} \ell\big(m, n, d, \tau\big) \overset{\text{def}}{=} \max_{\mathbf{r} \in \mathbb{F}_{q^m}^n} \left\{ \big|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})\big| \right\}.$$

Whenever the parameters m, n, d, τ are clear from the context, we will use the short-hand notation ℓ for the maximum list size. For an upper bound on ℓ, we have to show that the bound holds for *any* received word $\mathbf{r}$, whereas for a lower bound on ℓ it is sufficient to show that there exists (at least) one $\mathbf{r}$ for which this bound on the list size is valid.

W.l.o.g. we assume throughout this chapter that $n \leq m$. If this is not the case, we consider the transpose of all matrices such that also $n \leq m$ holds. We call n the *length* of such a block code in rank metric over $\mathbb{F}_{q^m}$.

Moreover, if we restrict ourselves to Gabidulin codes rather than arbitrary rank-metric codes, the task becomes more difficult due to the additional imposed structure of the code.

Let us denote the list of all codewords of an $(n, M, d)_R$ code C in the ball of rank radius τ around a given word $\mathbf{r}$ by:

$$\mathcal{L}\big(\mathsf{C}, \mathbf{r}\big) \overset{\text{def}}{=} \mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r}) = \left\{ \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(|\mathcal{L}|)} : \mathbf{c}^{(i)} \in \mathsf{C} \text{ and } \operatorname{rk}(\mathbf{r} - \mathbf{c}^{(i)}) \leq \tau,\ \forall i \right\}. \tag{5.4}$$

Clearly, the cardinality is $|\mathcal{L}\big(\mathsf{C}, \mathbf{r}\big)| \leq \ell$.

5.2.3 Connection between Constant-Rank Codes and the List of Decoding

Before proving our bounds, let us explain the connection between the list size for decoding a certain rank-metric code and the cardinality of a certain constant-rank code. As in (5.4), denote the list of codewords for an $(n, M, d_R = d)_R$ code C and for $\tau < d$ by

$$\mathcal{L}\big(\mathsf{C}, \mathbf{r}\big) = \left\{ \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(|\mathcal{L}|)} \right\} = \mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r}) = \sum_{i=0}^{\tau} \big(\mathsf{C} \cap \mathcal{S}_R^{(i)}(\mathbf{r})\big),$$

for some (received) word $\mathbf{r} \in \mathbb{F}_{q^m}^n$. If we consider only the codewords with rank distance *exactly* τ from the received word, i.e., on the sphere $\mathcal{S}_R^{(\tau)}(\mathbf{r})$:

$$\left\{ \mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(\overline{\ell})} \right\} \overset{\text{def}}{=} \mathsf{C} \cap \mathcal{S}_R^{(\tau)}(\mathbf{r}),$$

we obtain a lower bound on the maximum list size: $\ell \geq \overline{\ell} = |\mathsf{C} \cap \mathcal{S}_R^{(\tau)}(\mathbf{r})|$.

Now, consider a *translate* of all codewords on the sphere of radius τ as follows:

$$\overline{\mathcal{L}}\big(\mathsf{C}, \mathbf{r}\big) \overset{\text{def}}{=} \left\{ \mathbf{r} - \mathbf{c}^{(1)}, \mathbf{r} - \mathbf{c}^{(2)}, \dots, \mathbf{r} - \mathbf{c}^{(\overline{\ell})} \right\}.$$

This set $\overline{\mathcal{L}}\big(\mathsf{C}, \mathbf{r}\big)$ is a $\mathsf{CR}_{q^m}(n, M_R, d_R \geq d, \tau)$ constant-rank code over $\mathbb{F}_{q^m}$ since $\operatorname{rk}(\mathbf{r} - \mathbf{c}^{(i)}) = \tau$, $\forall i \in [1, \overline{\ell}]$, and its minimum rank distance is at least d, since

$$\operatorname{rk}(\mathbf{r} - \mathbf{c}^{(i)} - \mathbf{r} + \mathbf{c}^{(j)}) = \operatorname{rk}(\mathbf{c}^{(i)} - \mathbf{c}^{(j)}) \geq d, \quad \forall i, j \in [1, \overline{\ell}],\ i \neq j.$$

The cardinality of this constant-rank code is $M_R = \overline{\ell}$. For $\tau < d$, this constant-rank code is non-linear (or a translate of a linear code if C is linear), since the rank of its codewords is τ, but its minimum distance is at least d.

Hence, a translate of the list of all codewords of rank distance exactly τ from the received word can be interpreted as a constant-rank code. This interpretation makes it possible to use bounds on the cardinality of a constant-rank codes to obtain bounds on the list size ℓ for decoding rank-metric codes. This is also illustrated in Figure 5.2.

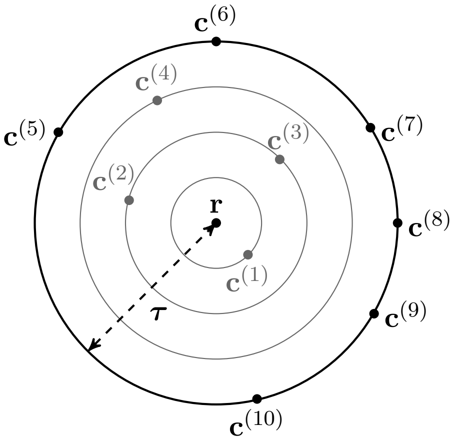


Figure 5.2. Interpretation of the decoding list as a constant-rank code, where all codewords (gray and black) constitute the decoding list and the black codewords constitute a constant-rank code.

5.3 A Lower Bound on the List Size for Gabidulin Codes

In this section, we provide a lower bound on the list size when decoding Gabidulin codes. The proof is based on the evaluation of linearized polynomials and is inspired by Justesen and Høholdt's [JH01] and Ben-Sasson, Kopparty, and Radhakrishnan's [BKR10] approaches for bounding the list size of Reed–Solomon codes.

Theorem 5.2 (Bound I: Lower Bound on the List Size).
Let the linear Gabidulin code $\mathsf{Gab}[n,k]$ *over* $\mathbb{F}_{q^m}$ *with* $n \leq m$ *and* $d_R = d = n-k+1$ *be given. Let* $\tau < d$*. Then, there exists a word* $\mathbf{r} \in \mathbb{F}_{q^m}^n$ *such that the maximum list size* ℓ *satisfies*

$$\ell = \ell\big(m,n,d,\tau\big) \geq \left|\mathsf{Gab}[n,k] \cap \mathcal{S}_R^{(\tau)}(\mathbf{r})\right| \geq \frac{\left[{n \atop n-\tau}\right]}{(q^m)^{n-\tau-k}}$$
$$\geq q^m q^{\tau(m+n)-\tau^2-md}, \tag{5.5}$$

and for the special case of $n=m$:

$$\ell \geq q^n q^{2n\tau-\tau^2-nd}.$$

Proof. Since we assume $\tau < d = n-k+1$, also $k-1 < n-\tau$ holds. Let us consider all monic linearized polynomials of q-degree exactly $n-\tau$ whose root spaces have dimension $n-\tau$ and all roots lie in $\mathbb{F}_{q^n}$. There are exactly (see e.g. [Ber84, Theorem 11.52]) $\left[{n \atop n-\tau}\right]$ such polynomials.
Now, let us consider a subset of these polynomials, denoted by $\mathcal{P}$: all polynomials where the q-monomials of q-degree greater than or equal to k have the same coefficients. Due to the pigeonhole principle, there exist coefficients such that the number of such polynomials is

$$|\mathcal{P}| \geq \frac{\left[{n \atop n-\tau}\right]}{(q^m)^{n-\tau-k}},$$

since there are $(q^m)^{n-\tau-k}$ possibilities to choose the highest $n-\tau-(k-1)$ coefficients of a *monic* linearized polynomial with coefficients $\mathbb{F}_{q^m}$.
Note that the difference of any two polynomials in $\mathcal{P}$ is a linearized polynomial of q-degree strictly less than k and therefore the evaluation polynomial of a codeword of $\mathsf{Gab}[n,k]$.

Let $\mathbf{r}$ be the evaluation of $p(x) \in \mathcal{P}$ at a basis $\mathcal{A} = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$:

$$\mathbf{r} = (r_0 \, r_1 \, \dots \, r_{n-1}) = \big(p(\alpha_0) \, p(\alpha_1) \, \dots \, p(\alpha_{n-1})\big).$$

Further, let also $q(x) \in \mathcal{P}$, then $p(x) - q(x)$ has q-degree less than k. Let $\mathbf{c}$ denote the evaluation of $p(x) - q(x)$ at $\mathcal{A}$. Then, $\mathbf{r} - \mathbf{c}$ is the evaluation of $p(x) - p(x) + q(x) = q(x) \in \mathcal{P}$, whose root space has dimension $n - \tau$ and all roots lie in $\mathbb{F}_{q^n}$. Thus, $\dim \ker(\mathbf{r} - \mathbf{c}) = n - \tau$ and $\dim \mathcal{C}_q(\mathbf{r} - \mathbf{c}) = \mathrm{rk}(\mathbf{r} - \mathbf{c}) = \tau$.
Therefore, for *any* $q(x) \in \mathcal{P}$, the evaluation of $p(x) - q(x)$ is a codeword of $\mathsf{Gab}[n, k]$ and has rank distance τ from $\mathbf{r}$. Hence,

$$\left|\mathsf{Gab}[n,k] \cap \mathcal{S}_R^{(\tau)}(\mathbf{r})\right| \geq |\mathcal{P}|.$$

Using (2.1), this provides the following lower bound on the maximum list size:

$$\ell \geq |\mathcal{P}| \geq \frac{q^{(n-\tau)\tau}}{(q^m)^{n-\tau-k}} \geq q^m q^{\tau(m+n)-\tau^2-md},$$

and for $n = m$ the special case follows. ∎

This lower bound is valid for any $\tau < d$, but we want to know, which is the smallest value for τ such that this expression grows *exponentially* in n.

For arbitrary $n \leq m$, we can rewrite (5.5) by

$$\ell \geq q^{m(1-\epsilon)} \cdot q^{\tau(m+n)-\tau^2-m(d-\epsilon)},$$

where the first part is exponential in $n \leq m$ for any $0 \leq \epsilon < 1$. The second exponent is positive for

$$\tau \geq \frac{m+n}{2} - \sqrt{\frac{(m+n)^2}{4} - m(d-\epsilon)} \overset{\text{def}}{=} \tau_J^*.$$

For $n = m$, this simplifies to

$$\tau \geq n - \sqrt{n(n-d+\epsilon)} \overset{\text{def}}{=} \tau_J. \tag{5.6}$$

Therefore, our lower bound (5.5) shows that the maximum list size is exponential in n for any $\tau \geq \tau_J^*$. For $n = m$, the value τ_J is basically the Johnson radius for codes in Hamming metric.

Faure obtained a similar result in [Fau06, Fau09] by using probabilistic arguments.

This reveals a difference between the known limits to list decoding of Gabidulin and Reed–Solomon codes. For Reed–Solomon codes, polynomial-time list decoding up to the Johnson radius can be accomplished by the Guruswami–Sudan algorithm. However, it is not proven that the Johnson radius is tight for Reed–Solomon codes, i.e., it is not known if the list size is polynomial in n between the Johnson radius and the known exponential lower bounds (see e.g. [JH01, BKR10]).

The result of Theorem 5.2 can also be obtained by interpreting the decoding list as a constant-rank code as in Subsection 5.2.3. For this purpose, we can use [GY10, Lemma 2] as follows.

Let C be a $\mathsf{Gab}[n, n-d+1]$ code of minimum rank distance d and B be a $\mathsf{Gab}[n, d-\tau]$ code of minimum rank distance $n - d + \tau + 1$. Let C be defined as in Definition 2.16 with the elements $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$, which are linearly independent over $\mathbb{F}_q$, and let B be defined with $g_0^{[n-d+1]}, g_1^{[n-d+1]}, \dots, g_{n-1}^{[n-d+1]}$. The corresponding generator matrices according to (2.27) are denoted by $\mathbf{G}_{\mathsf{C}}$ and $\mathbf{G}_{\mathsf{B}}$.

Then, the direct sum code $\mathsf{C} \oplus \mathsf{B}$ has the generator matrix $(\mathbf{G}_{\mathsf{C}}^T \, \mathbf{G}_{\mathsf{B}}^T)^T$ and is a $\mathsf{Gab}[n, n-\tau+1]$ code with minimum rank distance τ.

The rank weight distribution of MRD codes was given in [Gab85, Section 3] and therefore the number of codewords of rank τ in $\mathsf{C} \oplus \mathsf{B}$ is

$$W_\tau(\mathsf{C} \oplus \mathsf{B}) = \begin{bmatrix} n \\ \tau \end{bmatrix} (q^m - 1).$$

The cardinality of the code B is $|\mathsf{B}| = q^{m(d-\tau)}$ and therefore, with the pigeonhole principle, there exists a vector $\mathbf{b} \in \mathsf{B}$ such that the number of codewords in the translated code $\mathsf{C} \oplus \mathbf{b}$ is lower bounded by

$$W_\tau(\mathsf{C} \oplus \mathbf{b}) \geq \frac{\begin{bmatrix} n \\ \tau \end{bmatrix} (q^m - 1)}{q^{m(d-\tau)}}. \tag{5.7}$$

Hence, the number of codewords of C in rank distance τ from $\mathbf{b}$ is $W_\tau(\mathsf{C} \oplus \mathbf{b})$ and (5.7) yields the same lower bound on ℓ as Theorem 5.2.

Example 5.1 (List Decoding of Gab$[12, 6]$ Code).
For the Gabidulin code $\mathsf{Gab}[12, 6]$ *over* $\mathbb{F}_{2^{12}}$ *with* $d = 7$*, the BMD decoding radius is* $\tau_0 = \lfloor (d-1)/2 \rfloor = 3$*. The radius from* (5.6) *with* $\epsilon = 0.9$ *is* $\tau_J = \lceil 3.58 \rceil = 4$*. Hence, for this code of rate* $k/n = 1/2$*, no polynomial time list-decoding beyond* τ_0 *is possible.*

5.4 An Upper Bound on the List Size for Rank-Metric Codes

In this section, we will derive an upper bound on the list size when decoding rank-metric codes. This upper bound holds for *any* rank-metric code and *any* received word.

Theorem 5.3 (Bound II: Upper Bound on the List Size).
Let $\lfloor (d-1)/2 \rfloor \leq \tau < d \leq n \leq m$*. Then, for any* $(n, M, d)_R$ *code* C *in rank metric, the maximum list size is upper bounded as follows:*

$$\begin{aligned} \ell = \ell\big(m, n, d, \tau\big) &= \max_{\mathbf{r} \in \mathbb{F}_{q^m}^n} \left\{ \left| \mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r}) \right| \right\} \\ &\leq 1 + \sum_{t = \lfloor \frac{d-1}{2} \rfloor + 1}^{\tau} \frac{\begin{bmatrix} n \\ 2t+1-d \end{bmatrix}}{\begin{bmatrix} t \\ 2t+1-d \end{bmatrix}} \\ &\leq 1 + 4 \sum_{t = \lfloor \frac{d-1}{2} \rfloor + 1}^{\tau} q^{(2t-d+1)(n-t)} \\ &\leq 1 + 4 \cdot \left(\tau - \left\lfloor \tfrac{d-1}{2} \right\rfloor\right) \cdot q^{(2\tau-d+1)(n - \lfloor (d-1)/2 \rfloor - 1)}. \end{aligned} \tag{5.8}$$

Proof. Let $\{\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(\overline{\ell})}\}$ denote the intersection of the sphere $\mathcal{S}_R^{(t)}(\mathbf{r})$ in rank metric around $\mathbf{r}$ and the code C. As explained in Section 5.2.3,

$$\overline{\mathcal{L}}\big(\mathsf{C}, \mathbf{r}\big) = \big\{\mathbf{r} - \mathbf{c}^{(1)}, \mathbf{r} - \mathbf{c}^{(2)}, \dots, \mathbf{r} - \mathbf{c}^{(\overline{\ell})}\big\}$$

can be seen as a $\mathsf{CR}_{q^m}(n, M_R, d_R \geq d, t)$ constant-rank code over $\mathbb{F}_{q^m}$ for a word $\mathbf{r} \in \mathbb{F}_{q^m}^n$. Therefore, for any word $\mathbf{r} \in \mathbb{F}_{q^m}^n$, the cardinality of $\overline{\mathcal{L}}\big(\mathsf{C}, \mathbf{r}\big)$ can be upper bounded by the maximum cardinality of a constant-rank code with the corresponding parameters:

$$|\overline{\mathcal{L}}\big(\mathsf{C}, \mathbf{r}\big)| = \left|\mathsf{C} \cap \mathcal{S}_R^{(t)}(\mathbf{r})\right| \leq \mathsf{A}_{q^m}^R\,(n, d_R \geq d, t) \leq \mathsf{A}_{q^m}^R\,(n, d, t)\,.$$

We can upper bound this maximum cardinality by Proposition 5.1 with $\delta = d - t$ and $r = t$ by the maximum cardinality of a constant-dimension code:

$$\mathsf{A}^{R}_{q^m}\left(n, d, t\right) \leq \mathsf{A}^{S}_{q}\left(n, d_s = 2(d-t), t\right).$$

For upper bounding the cardinality of such a constant-dimension code, we use the Wang–Xing–Safavi-Naini bound [WXS03] (often also called anticode bound) and obtain:

$$\mathsf{A}^{S}_{q}\left(n, d_s = 2(d-t), t\right) \leq \frac{\begin{bmatrix} n \\ t-(d-t)+1 \end{bmatrix}}{\begin{bmatrix} t \\ t-(d-t)+1 \end{bmatrix}}. \tag{5.9}$$

In the ball of radius $\lfloor (d-1)/2 \rfloor$ around $\mathbf{r}$, there can be at most one codeword of C and therefore, the contribution to the list size is at most one. For higher t, we sum up (5.9) from $t = \lfloor (d-1)/2 \rfloor + 1$ to τ, use the upper bound on the q-binomial (2.1) and upper bound the sum. ∎

In [Wac12, Theorem 2], we showed an alternative proof of Theorem 5.3 based on the intersection of subspaces, but implicitly it re-derives the Wang–Xing–Safavi-Naini bound [WXS03].

The bound can slightly be improved if we use better upper bounds on the maximum cardinality of constant-dimension codes instead of (5.9) in the derivation, for example the iterated Johnson bound for constant-dimension codes [XF09, Corollary 3]. In this case, we obtain:

$$\ell = \ell\big(m, n, d, \tau\big) \leq 1 + \sum_{t=\left\lfloor \frac{d-1}{2} \right\rfloor + 1}^{\tau} \left\lfloor \frac{q^n - 1}{q^t - 1} \left\lfloor \frac{q^{n-1} - 1}{q^{t-1} - 1} \left\lfloor \dots \left\lfloor \frac{q^{n+d-2t} - 1}{q^{d-t} - 1} \right\rfloor \dots \right\rfloor \right\rfloor \right\rfloor.$$

However, the Wang–Xing–Safavi-Naini bound provides a nice closed-form expression and is asymptotically tight. Therefore, using better upper bounds for constant-dimension codes does not change the asymptotic behavior of our upper bound. Unfortunately, our upper bound on the list size of rank-metric codes is exponential in the length of the code and not polynomial as the Johnson bound for Hamming metric. However, the lower bound in Section 5.5 will show that any upper bound depending only on the length $n \leq m$ and the minimum rank distance d will be exponential in $(\tau - \lfloor (d-1)/2 \rfloor)(n - \tau)$, since there exists a rank-metric code with such a list size.

5.5 A Lower Bound on the List Size for Rank-Metric Codes

The bound presented in this section shows the most significant difference to bounds for codes in Hamming metric. We show the existence of a rank-metric code with exponential list size for any decoding radius *greater than half the minimum distance*. First, we prove the existence of a certain constant-rank code in the following theorem.

Theorem 5.4 (Constant-Rank Code).
Let $\lfloor (d-1)/2 \rfloor + 1 \leq \tau < d \leq n \leq m$ and $\tau \leq n - \tau$. Then, there exists a $\mathsf{CR}_{q^m}(n, M_R, d_R \geq d, \tau)$ constant-rank code over $\mathbb{F}_{q^m}$ of cardinality $M_R = q^{(n-\tau)(\tau - \lfloor (d-1)/2 \rfloor)}$.

Proof. First, assume d is even. Let us construct a $\mathsf{CD}_q(m, |M|, d, \tau)$ constant-dimension code M and a $\mathsf{CD}_q(n, |N|, d, \tau)$ code N by lifting an $\mathsf{MRD}[\tau, \tau - d/2 + 1]$ code over $\mathbb{F}_{q^{m-\tau}}$ of minimum rank distance $d/2$ and an $\mathsf{MRD}[\tau, \tau - d/2 + 1]$ code over $\mathbb{F}_{q^{n-\tau}}$ of minimum rank distance $d/2$ as in Lemma 2.18. Then, with Lemma 2.18:

$$|\mathsf{N}| = q^{(n-\tau)(\tau - d/2 + 1)} \leq |\mathsf{M}| = q^{(m-\tau)(\tau - d/2 + 1)}.$$

From Proposition 5.2, we know therefore there exists a $\mathsf{CR}_{q^m}(n, M_R, d_R, \tau)$ code of cardinality

$$M_R = \min\left\{|\mathsf{N}|, |\mathsf{M}|\right\} = q^{(n-\tau)(\tau-d/2+1)} = q^{(n-\tau)(\tau-\lfloor (d-1)/2 \rfloor)}.$$

For its rank distance by Proposition 5.2, the following holds:

$$d_R \geq \frac{1}{2}\, d_{S,M} + \frac{1}{2}\, d_{S,N} = d.$$

Second, assume d is odd. Let M be a $\mathsf{CD}_q(m, |M|, d-1, \tau)$ code constructed by the lifting of an $\mathsf{MRD}[\tau, \tau - (d-1)/2 + 1]$ code over $\mathbb{F}_{q^{m-\tau}}$ and let N be a $\mathsf{CD}_q(n, |N|, d+1, \tau)$ code, constructed by lifting an $\mathsf{MRD}[\tau, \tau - (d+1)/2 + 1]$ code over $\mathbb{F}_{q^{n-\tau}}$ code as in Lemma 2.18. Then,

$$|\mathsf{N}| = q^{(n-\tau)(\tau-(d+1)/2+1)} \leq |\mathsf{M}| = q^{(m-\tau)(\tau-(d-1)/2+1)}.$$

From Proposition 5.2, we know that there exists a $\mathsf{CR}_{q^m}(n, M_R, d_R, \tau)$ code of cardinality

$$M_R = \min\left\{|\mathsf{N}|, |\mathsf{M}|\right\} = |\mathsf{N}| = q^{(n-\tau)(\tau-(d-1)/2)} = q^{(n-\tau)(\tau-\lfloor (d-1)/2 \rfloor)}.$$

With Proposition 5.2, the rank distance d_R is lower bounded by:

$$d_R \geq \frac{1}{2}\, d_{S,M} + \frac{1}{2}\, d_{S,N} = \frac{1}{2}\,(d-1) + \frac{1}{2}\,(d+1) = d.$$

∎

This constant-rank code can now directly be used to show the existence of a rank-metric code with exponential list size.

Theorem 5.5 (Bound III: Lower Bound on the List Size).
Let $\lfloor (d-1)/2 \rfloor + 1 \leq \tau < d \leq n$ and $\tau \leq n - \tau$. Then, there exists an $(n, M, d_R \geq d)_R$ code C over $\mathbb{F}_{q^m}$ of length $n \leq m$ and minimum rank distance $d_R \geq d$, and a word $\mathbf{r} \in \mathbb{F}_{q^m}^n$ such that

$$\ell = \ell\big(m, n, d, \tau\big) \geq \left|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})\right| \geq q^{(n-\tau)(\tau-\lfloor (d-1)/2 \rfloor)}. \tag{5.10}$$

Proof. Let the $\mathsf{CR}_{q^m}(n, M_R, d_R \geq d, \tau)$ constant-rank code from Theorem 5.4 consist of the codewords:

$$\big\{\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(|\mathsf{N}|)}\big\}.$$

This code has cardinality $M_R = |\mathsf{N}| = q^{(n-\tau)(\tau-\lfloor (d-1)/2 \rfloor)}$ (see Theorem 5.4). Choose $\mathbf{r} = \mathbf{0}$, and hence, $\mathrm{rk}(\mathbf{r} - \mathbf{a}^{(i)}) = \mathrm{rk}(\mathbf{a}^{(i)}) = \tau$, $\forall i \in [1, |\mathsf{N}|]$ since the $\mathbf{a}^{(i)}$ are codewords of a constant-rank code of rank τ. Moreover, $d_R(\mathbf{a}^{(i)}, \mathbf{a}^{(i)}) = \mathrm{rk}(\mathbf{a}^{(i)} - \mathbf{a}^{(j)}) \geq d$ since the constant-rank code has minimum rank distance at least d.
Therefore, $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(|\mathsf{N}|)}$ are codewords of an $(n, M, d_R \geq d)_R$ code C over $\mathbb{F}_{q^m}$ in rank metric, which all lie on the sphere of rank radius τ around $\mathbf{r} = \mathbf{0}$ (which is not a codeword of C).
Hence, there exists an $(n, M, d_R \geq d)_R$ code C over $\mathbb{F}_{q^m}$ of length $n \leq m$ such that $\ell \geq |\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})| \geq |\mathsf{C} \cap \mathcal{S}_R^{(\tau)}(\mathbf{r})| = |\mathsf{N}| = q^{(n-\tau)(\tau-\lfloor (d-1)/2 \rfloor)}$. ∎

Notice that this $(n, M, d_R \geq d)_R$ code in rank metric is non-linear since it has codewords of weight $\tau < d$, but minimum rank distance at least d.

For constant code rate $R = k/n$ and constant relative decoding radius τ/n, where $\tau > \lfloor {}^{(d-1)}/_2 \rfloor$, (5.10) gives

$$\ell \geq q^{n^2(1-\tau/n)(\tau/n-(1-R)/2)} = q^{n^2 \cdot const}.$$

Therefore, the lower bound for this $(n, M, d_R \geq d)_R$ code is exponential in $n \leq m$ for any $\tau > \lfloor {}^{(d-1)}/_2 \rfloor$. Hence, Theorem 5.5 shows that there exist rank-metric codes, where the number of codewords in a rank-metric ball around the all-zero word is exponential in n, thereby prohibiting a polynomial-time list decoding algorithm. However, this does not mean that this holds for *any* rank metric code. In particular, the theorem does not provide a conclusion if there exists a *linear* code or even a *Gabidulin* code with this list size.

Remark 5.1 (Non-Zero Received Word).
The rank-metric code C *shown in Theorem 5.5 is clearly not linear. Instead of choosing* $\mathbf{r} = \mathbf{0}$*, we can choose for example* $\mathbf{r} = \mathbf{a}^{(1)}$*. The codewords of the* $\mathsf{CR}_{q^m}(n, M_R, d_R \geq d, \tau)$ *constant-rank code from Theorem 5.4 of cardinality* $M_R = |\mathsf{N}| = q^{(n-\tau)(\tau - \lfloor {}^{(d-1)}/_2 \rfloor)}$ *are denoted by:*

$$\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(|\mathsf{N}|)}.$$

Then, the following set of words

$$\left\{\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \dots, \mathbf{c}^{(|\mathsf{N}|)}\right\} \overset{\text{def}}{=} \left\{\mathbf{0}, \mathbf{a}^{(1)} - \mathbf{a}^{(2)}, \mathbf{a}^{(1)} - \mathbf{a}^{(3)}, \dots, \mathbf{a}^{(1)} - \mathbf{a}^{(|\mathsf{N}|)}\right\}$$

consists of codewords of an $(n, M, d_R \geq d)_R$ *code* C *over* $\mathbb{F}_{q^m}$ *since* $d_R(\mathbf{c}^{(i)}, \mathbf{c}^{(j)}) = \mathrm{rk}(\mathbf{c}^{(i)} - \mathbf{c}^{(j)}) = \mathrm{rk}(\mathbf{a}^{(1)} - \mathbf{a}^{(i)} - \mathbf{a}^{(1)} + \mathbf{a}^{(j)}) = \mathrm{rk}(\mathbf{a}^{(j)} - \mathbf{a}^{(i)}) \geq d$ *for* $i \neq j$ *since* $\mathbf{a}^{(i)}, \mathbf{a}^{(j)}$ *are codewords of the constant-rank code of minimum rank distance* d_R*. Moreover, all codewords* $\mathbf{c}^{(i)}$ *have rank distance exactly* τ *from* $\mathbf{r}$ *since* $\mathrm{rk}(\mathbf{r} - \mathbf{c}^{(i)}) = \mathrm{rk}(\mathbf{a}^{(i)}) = \tau$ *and the same bound on the list size of* C *follows as in Theorem 5.5. This* $(n, M, d_R \geq d)_R$ *rank-metric code over* $\mathbb{F}_{q^m}$ *is not necessarily linear, but also not necessarily not linear.*

The next corollary shows that the restriction $\tau \leq n - \tau$ does not limit the code rate for which Theorem 5.5 shows an exponential behavior of the list size. For the special case of $\tau = \lfloor {}^{(d-1)}/_2 \rfloor + 1$, the condition $\tau \leq n - \tau$ is always fulfilled for even minimum distance since $d \leq n$. For odd minimum $d - 1 \leq n$ has to hold. Notice that $d = n$ is a trivial code.

Corollary 5.2 (Special Case $\tau = \lfloor {}^{(d-1)}/_2 \rfloor + 1$).
Let $n \leq m$, $\tau = \lfloor {}^{(d-1)}/_2 \rfloor + 1$ *and* $d \leq n - 1$ *be odd. Then, there exists an* $(n, M, d_R \geq d)_R$ *code* C *and a word* $\mathbf{r} \in \mathbb{F}_{q^m}^n$ *such that* $|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})| \geq q^{(n-\tau)}$.

This corollary hence shows that for any $n \leq m$ and *any code rate* there exists a rank-metric code of rank distance at least d whose list size can be exponential in n.

For the special case when d is even, $\tau = d/2$ and $n = m$, the minimum rank distance of C is *exactly* d since the lower and upper bound on d_R in Proposition 5.2 coincide.

Corollary 5.3 (Special Case $\tau = d/2$).
Let $n = m$, d *be even and* $\tau = d/2$*. Then, there exists an* $(n, M, d_R = d)_R$ *code* C *in rank metric and a word* $\mathbf{r} \in \mathbb{F}_{q^m}^n$ *such that* $|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})| \geq q^{(n-\tau)}$.

Corollaries 5.2 and 5.3 show that the condition $\tau \leq n - \tau$ does not restrict lists of exponential size to a certain code rate. However, the following remark shows anyway what happens if we assume $\tau > n - \tau$.

Remark 5.2 (Case $\tau > n-\tau$).
Let $\lfloor (d-1)/2 \rfloor + 1 \leq \tau < d \leq n \leq m$ and $\tau > n-\tau$. Here, we can apply the same strategy as before: construct a constant-dimension code and show the existence of a constant-rank code of certain cardinality. For simplicity, let us consider only the case when d is even, the case of odd minimum distance follows straight-forward. Consider the lifting of a linear $\mathsf{MRD}[n-\tau, n-\tau-d/2+1]$ code C over $\mathbb{F}_{q^\tau}$ of minimum rank distance $d/2$. The lifting is denoted by $\mathrm{lift}(\mathsf{C})$, i.e., we consider $[\mathbf{I}_\tau \ \mathbf{C}_i]$ with $\mathbf{C}_i \in \mathbb{F}_q^{\tau \times (n-\tau)}$ for all $i = 1, \dots, |\mathsf{C}|$. In contrast to Lemma 2.18, we do not transpose the codewords of the MRD code here. The subspaces defined by this lifting are a $\mathsf{CD}_q(n, |N|, d_s = d, \tau)$ constant-dimension code of cardinality $q^{\tau(n-\tau-d/2+1)}$.

Then, with the same method as in Theorems 5.4 and 5.5 and a $\mathsf{CD}_q(m, |M|, d, \tau)$ code M and a $\mathsf{CD}_q(n, |N|, d, \tau)$ code N, there exists an $(n, M_R, d_R \geq d)_R$ code C in rank metric and a word $\mathbf{r} \in \mathbb{F}_{q^m}^n$ such that

$$\left|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})\right| \geq q^{\tau(n-\tau-d/2+1)}.$$

*However, the interpretation of this value is not so easy, since it depends on the concrete values of τ, d and n if the exponent is positive and if this bound is exponential in n or not. Moreover, as mention before, we do not need this investigation for polynomial-time list decodability as we recall that Theorem 5.5 shows that the list size is lower bounded by $q^{(n-\tau)}$ if we choose $\tau = \lfloor (d-1)/2 \rfloor + 1$ for codes of **any** rate, since then $\tau \leq n-\tau$ is fulfilled.*

The following lemma shows an improvement in the exponent of the lower bound of Theorem 5.5 for the case $\tau = d/2$ or when m is quite large compared to n.

Lemma 5.2 (Bound of Theorem 5.5 for $\tau = d/2$ or Large Extension Degree m).
Let $\lfloor (d-1)/2 \rfloor < \tau < d < n$ and $\tau \leq n-\tau$. If either $\tau = d/2$ or $m \geq (n-\tau)(2\tau - d + 1) + \tau + 1$, then there exists an $(n, M, d_R = d)_R$ code C over $\mathbb{F}_{q^m}$ of length $n \leq m$ and minimum rank distance d, and a word $\mathbf{r} \in \mathbb{F}_{q^m}^n$ such that

$$\ell = \ell\big(m, n, d, \tau\big) \geq \left|\mathsf{C} \cap \mathcal{B}_R^{(\tau)}(\mathbf{r})\right| \geq q^{(n-\tau)(2\tau-d+1)}. \tag{5.11}$$

Proof. We use [GY10, Theorem 2], which shows that for $2r \leq n \leq m$ and $1 \leq \delta \leq r$ there exists a constant-rank code of cardinality

$$\mathsf{A}_{q^m}^R\left(n, \delta + r, r\right) = \mathsf{A}_q^S\left(n, d_s = 2\delta, r\right)$$

if either $\delta = r$ or $m \geq (n-r)(r-d+1) + r + 1$.
Thus, similar to the proof of Theorem 5.3, we choose $r = \tau$ and $\delta = d - \tau$. Hence, there exists a $\mathsf{CR}_{q^m}(n, M_R, d, \tau)$ constant-rank code of cardinality

$$M_R = \mathsf{A}_q^S\left(n, d_s = 2(d-\tau), \tau\right) \geq q^{(n-\tau)(\tau-(d-\tau)+1)} = q^{(n-\tau)(2\tau-d+1)},$$

where we used the cardinality of a constant-dimension code based on a lifted MRD code (see Lemma 2.18) as lower bound. Analog to Theorem 5.5, we can use this constant-rank code to bound the list size. ∎

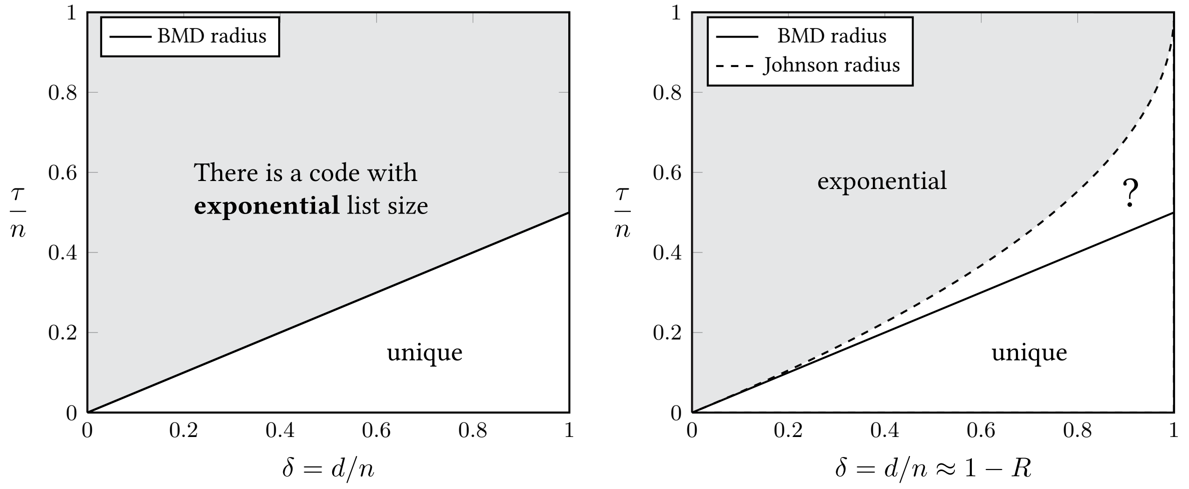
For the case $\tau = d/2$, this results in Corollary 5.3. Hence, for the cases of Lemma 5.2, the lower bound on the list size (5.11) and the upper bound (5.8) show the same asymptotic behavior and the upper bound is therefore asymptotically tight.

5.6 Summary, Comparison to Hamming Metric and Outlook

Let us interpret the results from the previous sections and compare them to known bounds on list decoding in Hamming metric (see e.g. [Gur99, Chapters 4 and 6]).

Theorem 5.5 (Bound III) shows that there is a code over $\mathbb{F}_{q^m}$ of length $n \leq m$ of rank distance at least d such that there is a ball of any radius $\tau > \lfloor (d-1)/2 \rfloor$, containing a number of codewords that grows exponentially in the length n. For this rank-metric code, *no* polynomial-time list decoding algorithm beyond half the minimum distance exists. This bound is tight as a function of d and n, since below we can clearly always decode uniquely. It does not mean that there is no code in rank metric with a polynomial list size for a decoding radius greater than half the minimum distance, but in order to find a polynomial upper bound, it will be necessary to use further properties of the code in the derivation of such bounds (linearity or the explicit code structure).

In particular, for Gabidulin codes, there is still an unknown region between half the minimum distance and the Johnson radius. With Bound I, we have proven that the list size can become exponential if the radius is at least the Johnson radius (see Theorem 5.2). These decoding regions are shown in Figure 5.3, depending on the relative normalized minimum rank distance $\delta = d/n$.



(a) General codes in rank metric

(b) Gabidulin codes

Figure 5.3. Decoding regions of codes in rank metric

Further, our lower bound from Theorem 5.5 (Bound III) shows that there does not exist a polynomial upper bound depending only on n and d similar to the Johnson bound for Hamming metric. Hence, our upper bound from Theorem 5.3 is relatively tight (except for a factor of two in the exponent), since it has the same asymptotic behavior as the lower bound from Theorem 5.5.

These results show a surprising difference to codes in Hamming metric. *Any* ball in Hamming metric of radius less than the Johnson radius $\tau_J = n - \sqrt{n(n-d)}$ always contains a *polynomial* number of codewords of *any* code of length n and minimum Hamming distance d (compare Section 5.1). Moreover, it can be shown that there exist codes in Hamming metric with an exponential number of codewords if the radius is at least the Johnson radius [GRS00, Gur99]. However, it is not known whether this bound is also tight for special classes of codes, e.g. Reed–Solomon codes. This points out another difference between Gabidulin and Reed–Solomon codes: For Reed–Solomon codes, the minimum radius for which an exponential list size is proven is much higher [JH01, BKR10] than for Gabidulin codes (see Bound I, Theorem 5.2).

Nevertheless, it is often believed that the Johnson bound is tight not only for codes in Hamming metric in general, but also for Reed–Solomon codes. Drawing a parallel conclusion for Gabidulin codes would mean that the maximum list size of Gabidulin codes could become exponential directly beyond half the minimum distance—but this requires additional research.

For future research, it is challenging to find a bound for the unknown region when list decoding Gabidulin codes. However, this seems to be quite difficult since the gap between the Johnson radius and the known lower exponential bounds for Reed–Solomon codes seems to translate into the gap between half the minimum distance and the Johnson radius for Gabidulin codes. Despite numerous publications on this topic, nobody could close the gap for Reed–Solomon codes so far. As a first step towards revealing the gap for Gabidulin codes, it might be possible to prove something like Theorem 5.5 for *linear* codes in rank metric.

CHAPTER 6

Convolutional Codes in Rank Metric

Partial unit memory (PUM) codes are convolutional codes with memory one (compare Subsection 2.1.4). All convolutional codes can be written as (P)UM codes when we join several blocks into one block as in [Bos98, Theorem 8.28].

Further, they can be constructed based on block codes, e.g., *Reed–Solomon* [ZS94, PMA88, Jus93] or cyclic codes [DS93, DS92]. The underlying block codes make an algebraic description of the convolutional code possible, enable us to estimate the distance properties and allow us to take into account existing efficient block decoders in order to decode the convolutional code.

A convolutional code in Hamming metric can be characterized by its *active row distance*, which in turn is basically determined by the *free distance* and the *slope*. These distance measures determine the error-correcting capability of the convolutional code. In [Lee76, Lau79, TJ83, PMA88], upper bounds on the free (Hamming) distance and the slope of (P)UM codes are derived.

In the context of network coding, dependencies between different blocks transmitted over a network can be created by *convolutional* codes. In multi-shot network coding, the network is used several times to transmit several blocks. In such a scenario, dependencies between the different shots can help to correct more errors than the classical approach based on rank-metric block codes.

In this chapter, we introduce (P)UM codes in (sum) rank metric. The *sum rank metric* is motivated by multi-shot network coding [NU10] and we use it to define the *free rank distance* and the *active row rank distance* in Subsection 6.1.1. In Subsection 6.1.2, we derive upper bounds on the free rank distance and the slope of the active row rank distance of (P)UM codes. Section 6.2 provides two explicit constructions of UM and PUM codes based on Gabidulin codes. The construction in Subsection 6.2.1 is based on the parity-check matrix and we give a lower bound on its distance parameters for dual memory $\mu_H = 1$. In Subsection 6.2.2, we construct PUM codes based on the generator matrix of Gabidulin codes and calculate their distance properties. Section 6.3 provides an efficient decoding algorithm based on rank-metric block decoders, which is able to handle errors and row/column erasures. This decoding algorithm can be seen as a generalization of the Dettmar–Sorger algorithm [DS95]. Finally, in Section 6.4, we show—similar to [SKK08]—how lifted PUM codes can be applied in *random linear network coding* (RLNC) and how decoding in RLNC reduces to error-erasure decoding of PUM codes based on Gabidulin codes.

The results presented in Section 6.1 and Subsection 6.2.1 were partly published in [WSBZ11a, WSBZ11b] and the results from Subsection 6.2.2 and Sections 6.3 and 6.4 in [WS12].

6.1 Distance Measures for Convolutional Codes in Rank Metric

In this section, we define distance measures for convolutional codes based on a special rank metric and prove upper bounds on them. This special rank metric—the *sum rank metric*—was proposed by Nóbrega and Uchôa-Filho under the name “extended rank metric” in [NU10] for multi-shot transmissions in a network. Furthermore, they modified the lifting construction such that it suits the sum rank metric.

6.1.1 Definition of Distance Parameters

In [NU10], it is shown that the sum rank distance and the subspace distance of the modified lifting construction are related in the same way as the rank distance and the subspace distance of the lifting construction, see [SKK08] and Lemma 2.18. Hence, the use of the sum rank metric for multi-shot network coding can be seen as the analog to using the rank metric for single-shot network coding.

The sum rank weight and distance are defined as follows.

Definition 6.1 (Sum Rank Weight and Sum Rank Distance).
Let two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{q^m}^{nN}$ be given and let them be decomposed into N subvectors of length n such that:

$$\mathbf{a} = (\mathbf{a}^{(0)}\ \mathbf{a}^{(1)}\ \dots\ \mathbf{a}^{(N-1)}), \quad \mathbf{b} = (\mathbf{b}^{(0)}\ \mathbf{b}^{(1)}\ \dots\ \mathbf{b}^{(N-1)}),$$

with $\mathbf{a}^{(i)}, \mathbf{b}^{(i)} \in \mathbb{F}_{q^m}^{n}$, $\forall i \in [0, N-1]$. The sum rank weight of $\mathbf{a}$ is the sum of the ranks of the subvectors:

$$\mathrm{wt}_{\Sigma,R}(\mathbf{a}) \overset{\text{def}}{=} \sum_{i=0}^{N-1} \mathrm{wt}_R(\mathbf{a}^{(i)}) = \sum_{i=0}^{N-1} \mathrm{rk}(\mathbf{a}^{(i)}). \tag{6.1}$$

The sum rank distance between $\mathbf{a}$ and $\mathbf{b}$ is the sum rank weight of the difference of the vectors:

$$d_{\Sigma,R}(\mathbf{a}, \mathbf{b}) \overset{\text{def}}{=} \mathrm{wt}_{\Sigma,R}(\mathbf{a} - \mathbf{b}) = \sum_{i=0}^{N-1} \mathrm{rk}(\mathbf{a}^{(i)} - \mathbf{b}^{(i)}). \tag{6.2}$$

Since the rank distance is a metric (see Lemma 2.13), the sum rank distance is also a metric.

An important measure for convolutional codes in Hamming metric is the free distance, and consequently, we define the *free rank distance* in a similar way in the sum rank metric.

Definition 6.2 (Free Rank Distance).
The free rank distance of a convolutional code C is the minimum sum rank distance (6.2) between any two different codewords $\mathbf{a}, \mathbf{b} \in \mathsf{C}$:

$$d_{f,R} \overset{\text{def}}{=} \min_{\substack{\mathbf{a},\mathbf{b}\in\mathsf{C},\\ \mathbf{a}\neq\mathbf{b}}} \left\{ d_{\Sigma,R}(\mathbf{a}, \mathbf{b}) \right\} = \min_{\substack{\mathbf{a},\mathbf{b}\in\mathsf{C},\\ \mathbf{a}\neq\mathbf{b}}} \left\{ \sum_{i=0}^{\infty} \mathrm{rk}(\mathbf{a}^{(i)} - \mathbf{b}^{(i)}) \right\}.$$

For a linear convolutional code $d_{f,R} = \min_{\mathbf{a}\in\mathsf{C},\mathbf{a}\neq\mathbf{0}} \left\{ \mathrm{wt}_{\Sigma,R}(\mathbf{a}) \right\}$ holds. Throughout this chapter, we consider only linear convolutional codes.

Any convolutional code can be described by a minimal code trellis, which has a certain number of states and the input/output blocks are associated to the edges of the trellis. The current state in the trellis of a (P)UM code over $\mathbb{F}_q$ can be associated with the vector $\mathbf{s}^{(i)} = \mathbf{u}^{(i-1)}\mathbf{G}^{(1)}$, see e.g., [Jus93], and therefore there are $q^{k^{(1)}}$ possible states. We call the current state *zero state* if $\mathbf{s}^{(i)} = \mathbf{0}$. A code sequence of a (P)UM code with N non-zero consecutive blocks can therefore be considered as a path in the trellis, which starts in the zero state and, after N edges (with non-zero output blocks), ends in the zero state.

The error-correcting capability of convolutional codes is determined by *active distances*—a fact that will become obvious in view of our decoding algorithm in Section 6.3. In the following, we define the *active row/column/reverse column rank distances* analog to active distances in Hamming metric [TJ83, HJZZ99, JZ99]. In the literature, there are different definitions of active distances in Hamming metric. Informally stated, for a j-th order active distance of C, we simply look at all sequences of length j, including conditions on the passed states in the minimal code trellis of C.

Let $\mathsf{C}_j^{(r)}$ denote the set of all codewords in a convolutional code C, corresponding to paths in the minimal code trellis which diverge from the zero state at depth zero and return to the zero state for the *first* time after j branches at depth j. W.l.o.g., we assume that we start at depth zero, as we only consider time-invariant convolutional codes. This set is illustrated in Figure 6.1.

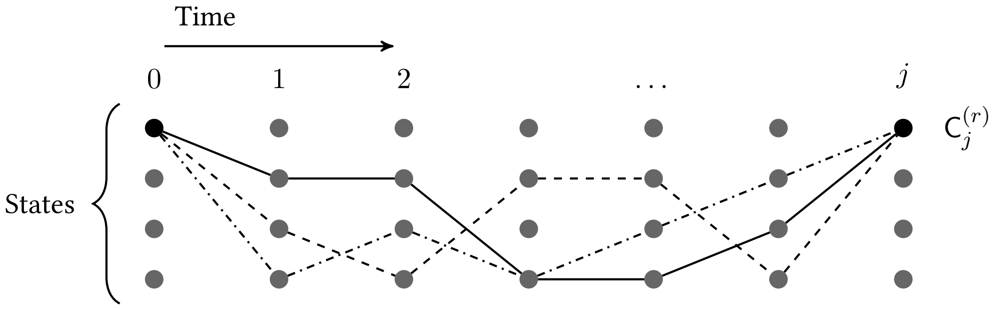


Figure 6.1. Illustration of the set $\mathsf{C}_j^{(r)}$: it consists of all codewords of C having paths in the minimal code trellis which diverge from the zero state at depth 0 and return to the zero state for the first time at depth j.

Definition 6.3 (Active Row Rank Distance).
The active row rank distance of order j of a linear convolutional code is defined as

$$d_{j,R}^{(r)} \overset{\text{def}}{=} \min_{\mathbf{c} \in \mathsf{C}_j^{(r)}} \Big\{ \mathrm{wt}_{\Sigma,R}(\mathbf{c}) \Big\}, \quad \forall j \geq 1.$$

Clearly, for non-catastrophic encoders, the minimum of the active row rank distances of different orders is the same as the free rank distance, see Definition (6.2):

$$d_{f,R} = \min_j \Big\{ d_{j,R}^{(r)} \Big\}.$$

The *slope* of the active row rank distance is defined as follows.

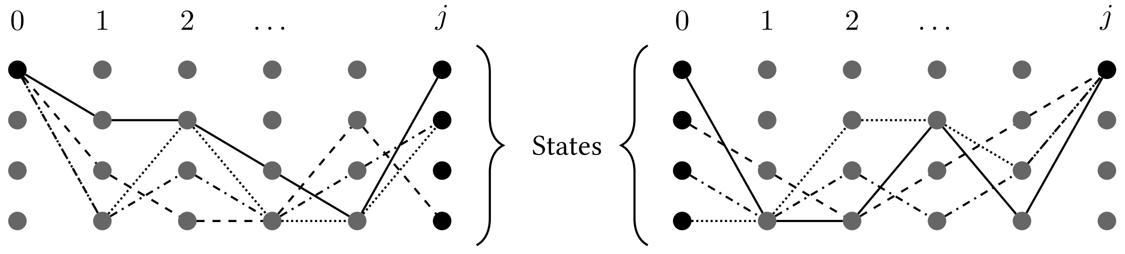
Definition 6.4 (Slope of Active Row Rank Distance).
The average linear increase (slope) of the active row rank distance (Definition 6.3) is

$$\sigma_R \overset{\text{def}}{=} \lim_{j \to \infty} \left\{ \frac{d_{j,R}^{(r)}}{j} \right\}.$$

As in Hamming metric [JPZ04, Theorem 1], [Jor02, Theorem 2.7], the active row rank distance of order j can be lower bounded by a linear function $d_{j,R}^{(r)} \geq \max\{j \cdot \sigma_R + \beta, d_{f,R}\}$ for some maximum $\beta \leq d_{f,R}$.

Similar to Hamming metric, we can introduce an active column rank distance and an active reverse column rank distance. Let $\mathsf{C}_j^{(c)}$ denote the set of all codewords leaving the zero state at depth zero and ending in any state at depth j and let $\mathsf{C}_j^{(rc)}$ denote the set of all codewords starting in any state at depth zero and ending in the zero state in depth j, both without zero states in between (see Figure 6.2). The active column rank distance and the active reverse column rank distance are then defined by:

$$d_{j,R}^{(c)} \overset{\text{def}}{=} \min_{\mathbf{c}\in\mathsf{C}_j^{(c)}} \Big\{ \mathrm{wt}_{\Sigma,R}(\mathbf{c}) \Big\}, \quad d_{j,R}^{(rc)} \overset{\text{def}}{=} \min_{\mathbf{c}\in\mathsf{C}_j^{(rc)}} \Big\{ \mathrm{wt}_{\Sigma,R}(\mathbf{c}) \Big\}, \qquad \forall j \geq 1. \tag{6.3}$$



(a) $\mathsf{C}_j^{(c)}$: all codewords of C diverging from the zero state at depth 0.

(b) $\mathsf{C}_j^{(rc)}$: all codewords of C ending in the zero state at depth j.

Figure 6.2. Illustration of the sets $\mathsf{C}_j^{(c)}$ and $\mathsf{C}_j^{(rc)}$, where no zero states between depths 0 and j are allowed.

6.1.2 Upper Bounds on Distances of (Partial) Unit Memory Codes

In this section, we derive upper bounds on the free rank distance $d_{f,R}$ (Definition 6.2) and the slope σ_R (Definition 6.4) for UM and PUM codes based on the sum rank metric (6.1), (6.2). The derivation of the bounds uses known bounds for (P)UM codes in Hamming metric [Lee76, Lau79, PMA88].

Theorem 6.1 (Connection between Distances in Hamming and Sum Rank Metric).
Let the free rank distance be defined as in Definition 6.2 and the active row rank distance as in Definition 6.3. The active row Hamming distance $d_{j,H}^{(r)}$ and the free Hamming distance $d_{f,H}$ are defined by replacing the sum rank weight/distance in Definitions 6.2 and 6.3 by the Hamming weight/distance. Then,

$$\begin{aligned} d_{f,R} &\leq d_{f,H}, \\ d_{j,R}^{(r)} &\leq d_{j,H}^{(r)}, \quad \forall j \geq 1. \end{aligned}$$

Proof. The rank and Hamming weight of a vector $\mathbf{a} \in \mathbb{F}_{q^m}^n$ are $\mathrm{rk}(\mathbf{a}) \leq \mathrm{wt}_H(\mathbf{a})$. Hence:

$$\mathrm{wt}_{\Sigma,R}(\mathbf{a}) = \sum_{i=0}^{N-1} \mathrm{rk}(\mathbf{a}^{(i)}) \leq \mathrm{wt}_H(\mathbf{a}^{(0)}\ \mathbf{a}^{(1)}\ \ldots\ \mathbf{a}^{(N-1)}).$$

and the statement follows with (6.1) and Definitions 6.2, 6.3. ∎

Thus, the upper bounds for the free Hamming distance and the slope of (P)UM codes from [TJ83, PMA88] also hold for (P)UM codes in sum rank metric.

Corollary 6.1 (Upper Bounds).
For a $\mathsf{UM}(n,k)$ code, where $\nu = k$, the free rank distance is upper bounded by:

$$d_{f,R} \leq 2n - k + 1. \tag{6.4}$$

For a $\mathsf{PUM}(n,k|k^{(1)})$ *code, where* $\nu = k^{(1)} < k$*, the free rank distance is upper bounded by:*

$$d_{f,R} \leq n - k + \nu + 1. \tag{6.5}$$

For both, UM and PUM codes, the average linear increase (slope) is upper bounded by:

$$\sigma_R \leq n - k. \tag{6.6}$$

6.2 Constructions of Convolutional Codes in Rank Metric

This section provides two constructions of (P)UM codes based on Gabidulin codes. One of them uses the parity-check matrix (Subsection 6.2.1) and the other one the generator matrix (Subsection 6.2.2) of the convolutional code for the definition.

6.2.1 PUM Codes Based on the Parity-Check Matrix of Gabidulin Codes

The construction in this subsection is similar to the construction of (P)UM codes based on Reed–Solomon codes from [ZS94]. Theorem 6.2 shows sufficient conditions that the parity-check matrix (based on Gabidulin codes) defines a (P)UM code.

Theorem 6.2 ((P)UM Code Based on Gabidulin Codes).
Let $\mu_H \geq 1$ *and let* $\mathbf{H}$ *be the semi-infinite parity-check matrix of a convolutional code* C *over* $\mathbb{F}_{q^m}$ *as in* (2.7) *with code rate* $R = k/n \geq \mu_H/(\mu_H+1)$*. Let each submatrix* $\mathbf{H}^{(i)}$ *be the parity-check matrix of a* $\mathsf{Gab}[n,k]$ *code, i.e.:*

$$\mathbf{H}^{(i)} = \mathrm{qvan}_{n-k}(\mathbf{h}^{(i)}) = \mathrm{qvan}_{n-k}((h_0^{(i)}\ h_1^{(i)}\ \ldots\ h_{n-1}^{(i)})), \quad \forall i \in [0,\mu_H],$$

where $h_0^{(i)}, h_1^{(i)}, \ldots, h_{n-1}^{(i)} \in \mathbb{F}_{q^m}$ *are linearly independent over* $\mathbb{F}_q$*. Additionally, let*

$$\mathbf{H}^{(c)} \overset{\text{def}}{=} \begin{pmatrix} \mathbf{H}^{(0)} \\ \mathbf{H}^{(1)} \\ \vdots \\ \mathbf{H}^{(\mu_H)} \end{pmatrix} \text{ define a } \mathsf{Gab}[n, n-(\mu_H+1)(n-k)] \text{ code,} \tag{6.7}$$

and let $\mathbf{H}^{(r(i))} \overset{\text{def}}{=} \begin{pmatrix}\mathbf{H}^{(i)} & \mathbf{H}^{(i-1)} & \ldots & \mathbf{H}^{(0)}\end{pmatrix}$ *define a* $\mathsf{Gab}[(i+1)n, in+k]$ *code,* $\forall i \in [1,\mu_H]$*.*

Then, $\mathbf{H}$ *is the parity-check matrix of a rate* $R \geq \mu_H/(\mu_H+1)$ *(partial) unit memory code over* $\mathbb{F}_{q^m}$*.*

Proof. The proof follows from Theorem 2.1 (since the rate restriction is fulfilled and all $\mathbf{H}^{(i)}$ have full rank) and from Lemma 6.2, which shows that $\mathbf{H}$ is in minimal basic encoding form. ∎

Hence, not only each submatrix $\mathbf{H}^{(i)}$ defines a Gabidulin code, but also specified blocks of submatrices. The latter one is not necessary to guarantee that it is a (P)UM code, but it results in good distance properties (see Theorem 6.3). Lemma 6.1 gives an explicit construction, satisfying all requirements of Theorem 6.2.

In order to fulfill (6.7), $\mathbf{H}^{(c)}$ has to be a q-Vandermonde matrix:

$$\mathbf{H}^{(c)} = \mathrm{qvan}_{(\mu_H+1)(n-k)}(\mathbf{h}^{(0)}) = \mathrm{qvan}_{(\mu_H+1)(n-k)}((h_0^{(0)}\ h_1^{(0)}\ \ldots\ h_{n-1}^{(0)})). \tag{6.8}$$

To satisfy also the conditions on $\mathbf{H}^{(r(i))}$, $\forall i \in [1, \mu_H]$, we have to ensure that all elements in the set

$$\begin{aligned}\mathcal{H} &\overset{\text{def}}{=} \left\{h_0^{(0)},\dots,h_{n-1}^{(0)},h_0^{(1)},\dots,h_{n-1}^{(1)},\dots,h_0^{(\mu_H)},\dots,h_{n-1}^{(\mu_H)}\right\}\\ &= \left\{h_0^{(0)},\dots,h_{n-1}^{(0)},h_0^{(0)[n-k]},\dots,h_{n-1}^{(0)[n-k]},\dots,h_0^{(0)[\mu_H(n-k)]},\dots,h_{n-1}^{(0)[\mu_H(n-k)]}\right\}\end{aligned} \tag{6.9}$$

of cardinality $|\mathcal{H}| = (\mu_H + 1) \cdot n$, are in $\mathbb{F}_{q^m}$ and are linearly independent over $\mathbb{F}_q$.

Therefore, if the elements of $\mathcal{H}$ (6.9) are linearly independent and $\mathbf{H}^{(c)} \in \mathbb{F}_{q^m}^{(\mu_H+1)(n-k)\times n}$ is a q-Vandermonde matrix as in (6.8), all requirements of Theorem 6.2 are fulfilled.

The following lemma shows explicitly how to choose the set $\mathcal{H}$ using a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ (see also Subsection 2.1.2) in order to construct a (P)UM code based on Gabidulin codes.

Lemma 6.1 (Explicit Construction with Normal Basis).
Let $\mu_H \geq 1$, the code rate $R = k/n \geq \mu_H/(\mu_H+1)$ and $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[m-1]}\}$ be a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, where the field size satisfies

$$m \geq \mu_H(n-k)\left\lceil \frac{n}{n-k} \right\rceil + n. \tag{6.10}$$

Further, define $\mathbf{h}^{(0)} \in \mathbb{F}_{q^m}^n$ by

$$\begin{aligned}\mathbf{h}^{(0)} = \big(\beta^{[0]}\ \beta^{[1]}\ \dots\ \beta^{[n-k-1]} \mid \beta^{[(\mu_H+1)(n-k)]}\ \beta^{[(\mu_H+1)(n-k)+1]}\ \dots\ \beta^{[(\mu_H+2)(n-k)-1]} \mid \\ \beta^{[2(\mu_H+1)(n-k)]}\ \beta^{[2(\mu_H+1)(n-k)+1]}\ \dots\ \beta^{[2(\mu_H+2)(n-k)-1]} \mid \dots\big).\end{aligned} \tag{6.11}$$

Let $\mathbf{H}^{(c)}$ be defined by (6.8) *using $\mathbf{h}^{(0)}$. Let the semi-infinite parity-check matrix $\mathbf{H}$ as in* (2.7) *be defined with the set $\mathcal{H}$ from* (6.9).

Then, $\mathbf{H}$ consists of $\mu_H + 1$ submatrices $\mathbf{H}^{(i)}$ and satisfies all requirements of Theorem 6.2.

Proof. To prove that $\mathbf{H}^{(r(i))}$ defines a Gabidulin code, $\forall i \in [1, \mu_H]$, with the parameters as in Theorem 6.2, it is sufficient that all elements in $\mathcal{H}$ are linearly independent (6.9). There are at most m linearly independent elements in $\mathbb{F}_{q^m}$. If $(n-k)$ divides n, then $\mathbf{h}^{(0)}$ can be divided into subvectors, each of length $(n-k)$ as in (6.11), and the field size has to be $m \geq (\mu_H+1)\cdot n$. In general, the last subvector in $\mathbf{h}^{(0)}$ might be shorter than $n-k$ and the linear independence within $\mathcal{H}$ (6.9) is guaranteed if (6.10) is fulfilled.
Therefore, $h_i^{(j)} = h_i^{(0)[j(n-k)]}$ as in (6.8) and the elements in $\mathcal{H}$ (6.9) are linearly independent. Hence, $\mathbf{H}^{(c)}$ and $\mathbf{H}^{(r(i))}$ define Gabidulin codes with the parameters required in Theorem 6.2. ∎

The following lemma shows that the parity-check matrix constructed in such a way is in minimal basic encoding form.

Lemma 6.2 (Construction is in Minimal Basic Encoding Form).
Let a (P)UM code based on Gabidulin codes be defined by its parity-check matrix $\mathbf{H}$ as in Theorem 6.2, explicitly written e.g. as in Lemma 6.1. Then, $\mathbf{H}$ is in minimal basic encoding form.

Proof. Let us denote the corresponding polynomial parity-check matrix by $\mathbf{H}(D) = \mathbf{H}^{(0)} + \mathbf{H}^{(1)}D + \dots + \mathbf{H}^{(\mu_H)}D^{\mu_H}$ and compare Remark 2.2 for the required properties.
First, $\mathbf{H}$ is in *encoding* form since $\mathbf{H}^{(0)}$ is a q-Vandermonde matrix and therefore has full rank.
Second, we show that $\mathbf{H}$ is in *basic* form. According to [For70, Definition 4], $\mathbf{H}(D)$ is basic if it is polynomial and if there exists a polynomial right inverse $\mathbf{H}^{-1}(D)$, such that $\mathbf{H}(D) \cdot \mathbf{H}^{-1}(D) = \mathbf{I}_{(n-k)\times(n-k)}$. By definition, $\mathbf{H}(D)$ is polynomial. A polynomial right inverse exists if and only

if $\mathbf{H}(D)$ is non-catastrophic and hence if the slope is $\sigma_R > 0$ [Det94, Theorem A.4]. We will calculate the slope in Theorem 6.4, proving that $\sigma_R > 0$.
Third, we show that $\mathbf{H}(D)$ is in *minimal* form by analyzing the degree of the determinants of all $(n-k) \times (n-k)$ submatrices of $\mathbf{H}(D)$. Denote these submatrices by $\mathbf{H}_\ell(D)$ for $\ell = 0, 1, \dots$. Clearly, $\deg(h_{ij}(D)) = \mu_H$ for all $i \in [0, n-k-1]$, $j \in [0, n-1]$. Thus, $\deg[\det(\mathbf{H}_\ell(D))] \leq \mu_H(n-k)$. The coefficient of $D^{\mu_H(n-k)}$ of $\det(\mathbf{H}_\ell(D))$ is exactly $\det(\mathbf{H}_\ell^{(\mu_H)})$, where $\mathbf{H}_\ell^{(\mu_H)}$ is a $(n-k) \times (n-k)$-submatrix of $\mathbf{H}^{(\mu_H)}$. Since $\mathbf{H}_\ell^{(\mu_H)}$ is an $(n-k) \times (n-k)$ q-Vandermonde matrix, $\det(\mathbf{H}_\ell^{(\mu_H)}) \neq 0$ and $\deg[\det(\mathbf{H}_\ell(D))] = \mu_H(n-k), \forall \ell = 0, 1, \dots$. This is equal to the constraint length in obvious realization $\nu = \mu_H(n-k)$ and hence, $\mathbf{H}(D)$ is in minimal basic encoding form. ∎

Alternatively, we could use [JZ99, Theorem 2.22, (iii)] to prove that $\mathbf{H}$ is in basic form. Since we consider non-binary convolutional codes, notice that the corresponding matrix $[\mathbf{H}(D)]_h$ (in the notation of [JZ99]) is a matrix in $\mathbb{F}_{q^m}$ with the highest coefficient of $h_{i,j}(D)$ at entry (i,j) and with the entry 0 if $h_{i,j}(D) = 0$. For our construction, $[\mathbf{H}(D)]_h = \mathbf{H}^{(\mu_H)}$, which has full rank and therefore, due to [JZ99, Theorem 2.22, (iii)], $\mathbf{H}$ is a basic encoding matrix.

Example 6.1 (Construction of PUM Code based on Parity-Check Matrix).
Let us construct a $\mathsf{PUM}(6, 4|2)$ *code* C *with* $\mu_H = 1$ *and code rate* $R = 2/3 \geq \mu_H/(\mu_H + 1) = 1/2$. *Due to* (6.10)*, the field size is* $m \geq 12$ *and we define the code over* $\mathbb{F}_{q^m} = \mathbb{F}_{2^{12}}$. *Let* $\mathcal{B}_N = \{\beta^{[0]}, \beta^{[1]}, \dots, \beta^{[11]}\}$ *be a normal basis of* $\mathbb{F}_{2^{12}}$ *over* $\mathbb{F}_2$.

With (6.11) *and* (6.9)*, we obtain*

$$\mathbf{h}^{(0)} = \begin{pmatrix} \beta^{[0]} & \beta^{[1]} & | & \beta^{[4]} & \beta^{[5]} & | & \beta^{[8]} & \beta^{[9]} \end{pmatrix},$$
$$\mathbf{h}^{(1)} = \begin{pmatrix} \beta^{[2]} & \beta^{[3]} & | & \beta^{[6]} & \beta^{[7]} & | & \beta^{[10]} & \beta^{[11]} \end{pmatrix}.$$

The semi-infinite parity-check matrix $\mathbf{H}$ *is then given by*

$$\mathbf{H} = \left(\begin{array}{cccccc|cccccc|c}
\beta^{[0]} & \beta^{[1]} & \beta^{[4]} & \beta^{[5]} & \beta^{[8]} & \beta^{[9]} & & & & & & & \\
\beta^{[1]} & \beta^{[2]} & \beta^{[5]} & \beta^{[6]} & \beta^{[9]} & \beta^{[10]} & & & & & & & \\
\hline
\beta^{[2]} & \beta^{[3]} & \beta^{[6]} & \beta^{[7]} & \beta^{[10]} & \beta^{[11]} & \beta^{[0]} & \beta^{[1]} & \beta^{[4]} & \beta^{[5]} & \beta^{[8]} & \beta^{[9]} & \\
\beta^{[3]} & \beta^{[4]} & \beta^{[7]} & \beta^{[8]} & \beta^{[11]} & \beta^{[12]} & \beta^{[1]} & \beta^{[2]} & \beta^{[5]} & \beta^{[6]} & \beta^{[9]} & \beta^{[10]} & \\
\hline
 & & & & & & \beta^{[2]} & \beta^{[3]} & \beta^{[6]} & \beta^{[7]} & \beta^{[10]} & \beta^{[11]} & \ddots \\
 & & & & & & \beta^{[3]} & \beta^{[4]} & \beta^{[7]} & \beta^{[8]} & \beta^{[11]} & \beta^{[12]} & \ddots \\
\hline
\end{array}\right).$$

As required by Theorem 6.2, the matrices $\mathbf{H}^{(0)}$, $\mathbf{H}^{(1)}$, $(\mathbf{H}^{(0)}\ \mathbf{H}^{(1)})$ *and* $\begin{pmatrix}\mathbf{H}^{(0)} \\ \mathbf{H}^{(1)}\end{pmatrix}$ *define Gabidulin codes.*

Theorem 6.2 guarantees that there is a generator matrix of C *with memory* $\mu = 1$*, consisting of two* (4×6)*-submatrices* $\mathbf{G}^{(0)}$ *and* $\mathbf{G}^{(1)}$. *The submatrices* $\mathbf{G}^{(00)}$, $\mathbf{G}^{(01)}$, $\mathbf{G}^{(10)}$ *are all* (2×6)*-matrices, since* $k^{(1)} = \nu = \mu_H(n-k) = 2$. *Hence,* $\mathbf{H}$ *defines a rate* $R = 2/3$ *PUM code based on Gabidulin codes.*

Let us now calculate the active row rank distance of the construction of Theorem 6.2 with dual memory $\mu_H = 1$. Implicitly, this calculation provides the free rank distance $d_{f,R}$ and the slope σ_R. Denote the minimum rank distances of the codes defined by the parity-check matrices $\mathbf{H}^{(0)}$, $\mathbf{H}^{(1)}$ and $\begin{pmatrix}\mathbf{H}^{(1)} & \mathbf{H}^{(0)}\end{pmatrix}$ by d_0, d_1 and d_{10}, respectively. They are $d_0 = d_1 = d_{10} = n - k + 1$.

Theorem 6.3 (Active Row Rank Distance of our Construction).
Let a $\mathsf{PUM}(n,k|k^{(1)})$ *code of rate* $R \geq 1/2$ *with* $\mu_H = 1$ *be given, where the submatrices of* $\mathbf{H}$ *define Gabidulin codes as in Theorem 6.2. Then, the active row rank distance* $d_{j,R}^{(r)}$ *(Definition 6.3) is bounded by*

$$\begin{aligned} d_{1,R}^{(r)} &= 2(n-k)+1, \\ d_{j,R}^{(r)} &\geq \left\lceil \frac{j+1}{2} \right\rceil \cdot (n-k+1), \qquad j \geq 2. \end{aligned} \tag{6.12}$$

Proof. The derivation of the active row rank distances for $\mu_H = 1$ is similar to the analysis by Zyablov and Sidorenko [ZS94]. In order to estimate the active row rank distance $d_{j,R}^{(r)}$ from Definition 6.3, consider all paths in the set $\mathsf{C}_j^{(r)}$ (compare Figure 6.1) for $j \geq 1$.

- For the active row rank distance of order one, we have to consider only code sequences of the form $(\mathbf{c}^{(0)}\ \mathbf{0}\ \mathbf{0}\ \dots)$. Using the $2(n-k) \times n$ parity-check matrix $\widetilde{\mathbf{H}} = (\mathbf{H}^{(0)T}\ \mathbf{H}^{(1)T})^T$, the codewords have to satisfy $\widetilde{\mathbf{H}} \cdot \mathbf{c}^{(0)T} = \mathbf{0}$. Since $\widetilde{\mathbf{H}}$ defines a Gabidulin code of minimum rank distance $2(n-k)+1$, we obtain $d_{1,R}^{(r)} = 2(n-k)+1$.
- For $j = 2$, we have to investigate all code sequences of the form $\big(\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \mathbf{0}\ \dots\big)$. These code sequences are defined by the $3(n-k) \times 2n$ parity–check matrix

$$\widetilde{\mathbf{H}} = \begin{pmatrix} \mathbf{H}^{(0)} & \\ \mathbf{H}^{(1)} & \mathbf{H}^{(0)} \\ & \mathbf{H}^{(1)} \end{pmatrix}.$$

i.e., $\widetilde{\mathbf{H}} \cdot \big(\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\big)^T = \mathbf{0}$. Hence, amongst others, the following equations must be fulfilled:

$$\mathbf{H}^{(0)} \cdot \mathbf{c}^{(0)T} = \mathbf{0}, \quad \mathbf{H}^{(1)} \cdot \mathbf{c}^{(1)T} = \mathbf{0}.$$

Due to the definition of the sum rank distance, $d_{2,R}^{(r)} \geq d_0 + d_1 = 2(n-k+1)$.
- For $j = 3$ and code sequences of the form $\big(\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \mathbf{c}^{(2)}\ \mathbf{0}\ \dots\big)$, the same two equations have to hold and therefore, $d_{3,R}^{(r)} \geq d_0 + d_1 = 2(n-k+1)$.
- For $j = 4$ and code sequences $\big(\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \mathbf{c}^{(2)}\ \mathbf{c}^{(3)}\ \mathbf{0}\ \dots\big)$, we have to consider the $5(n-k) \times 4n$ parity-check matrix $\widetilde{\mathbf{H}}$, for which $\widetilde{\mathbf{H}} \cdot \big(\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \mathbf{c}^{(2)}\ \mathbf{c}^{(3)}\big)^T = \mathbf{0}$ and in particular:

$$\mathbf{H}^{(0)} \cdot \mathbf{c}^{(0)T} = \mathbf{0}, \quad \mathbf{H}^{(1)} \cdot \mathbf{c}^{(1)T} + \mathbf{H}^{(0)} \cdot \mathbf{c}^{(2)} = \mathbf{0}, \quad \mathbf{H}^{(1)} \cdot \mathbf{c}^{(3)T} = \mathbf{0},$$

and therefore, $d_{4,R}^{(r)} \geq d_0 + d_{10} + d_1 = 3(n-k+1)$.

Similarly, $d_{5,R}^{(r)} \geq d_0 + d_{10} + d_1 = 3(n-k+1)$ and $d_{6,R}^{(r)} \geq d_0 + d_{10} + d_{10} + d_1 = 4(n-k+1)$. In general, by continuing this strategy, we obtain the statement. ∎

The following theorem shows that our construction achieves the upper bound on the free rank distance for PUM codes (6.5) and half the optimal slope (6.6).

Theorem 6.4 (Free Rank Distance and Slope of our Construction for $\mu_H = 1$).
For $R > 1/2$, the $\mathsf{PUM}(n, k|k^{(1)})$ code based on Gabidulin codes as in Theorem 6.2 with $\mu_H = 1$ achieves the upper bound on the free rank distance (6.5) *and half the optimal slope* (6.6)*:*

$$d_{f,R} = 2(n-k)+1 = n-k+\nu+1,$$
$$\sigma_R = \frac{n-k+1}{2}.$$

Proof. The overall constraint length is $\nu = n-k$. Hence,

$$d_{f,R} = \min_j \left\{ d_{j,R}^{(r)} \right\} = d_{1,R}^{(r)} = 2(n-k)+1 = n-k+\nu+1.$$

The slope is calculated as in Definition 6.4:

$$\sigma_R = \lim_{j \to \infty} \left\{ \frac{d_{j,R}^{(r)}}{j} \right\} = \frac{n-k+1}{2}.$$

∎

However, it is not clear what happens when $\mu_H > 1$. Based on simulations, we conjecture the following.

Conjecture 6.1 (Free Rank Distance and Slope of our Construction for Arbitrary μ_H).
For $R > \mu_H/(\mu_H+1)$, the $\mathsf{PUM}(n, k|k^{(1)})$ code based on Gabidulin codes as in Theorem 6.2 with $\mu_H \geq 1$ achieves

$$d_{f,R} = (\mu_H+1)(n-k)+1 = n-k+\nu+1,$$
$$\sigma_R \geq \frac{n-k+1}{\mu_H+1}.$$

The lower bound on the slope can actually be proven similar to Theorem 6.3, but for the free rank distance it is not clear if $d_{1,R}^{(r)}$ is the minimum of the active row rank distances.

This conjecture implies that the free rank distance *increases* with higher μ_H and the slope *decreases*. Hence,—if the conjecture is true—a trade-off between the free rank distance and the slope is possible. A similar behavior was observed by Jordan, Pavlushkov and Zyablov [JPZ04] in Hamming metric, since they showed that for general convolutional codes, the upper and lower bounds on the free distance increase with increasing memory whereas the upper and lower bounds for the slope decrease.

6.2.2 PUM Codes Based on the Generator Matrix of Gabidulin Codes

In the previous subsection, (P)UM codes were constructed such that the submatrices of the *parity-check* matrix define Gabidulin codes. In this subsection, we construct such codes based on the *generator* matrix. Our construction is an adaptation of the construction from [DS95] to rank metric.

Definition 6.5 ((P)UM Code based on Generator Matrices of Gabidulin Codes).
Let $k + k^{(1)} \leq n \leq m$, where $k^{(1)} \leq k$, and let $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$ be linearly independent over $\mathbb{F}_q$. For $k^{(1)} = k$ we define a $\mathsf{UM}(n,k)$ code and for $k^{(1)} < k$ a $\mathsf{PUM}(n, k|k^{(1)})$ code over $\mathbb{F}_{q^m}$ by a zero-forced terminated generator matrix $\mathbf{G}_{\mathsf{term}}$ as in (2.8) *with $\mu = 1$ and the $k \times n$ submatrices $\mathbf{G}^{(0)}$*

and $\mathbf{G}^{(1)}$*:*

$$\mathbf{G}^{(0)} = \left(\begin{array}{c} \mathbf{G}^{(00)} \\ \hline \mathbf{G}^{(01)} \end{array}\right) = \left(\begin{array}{cccc} g_0 & g_1 & \cdots & g_{n-1} \\ g_0^{[1]} & g_1^{[1]} & \cdots & g_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{[k^{(1)}-1]} & g_1^{[k^{(1)}-1]} & \cdots & g_{n-1}^{[k^{(1)}-1]} \\ \hline g_0^{[k^{(1)}]} & g_1^{[k^{(1)}]} & \cdots & g_{n-1}^{[k^{(1)}]} \\ g_0^{[k^{(1)}+1]} & g_1^{[k^{(1)}+1]} & \cdots & g_{n-1}^{[k^{(1)}+1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{[k-1]} & g_1^{[k-1]} & \cdots & g_{n-1}^{[k-1]} \end{array}\right), \tag{6.13}$$

and

$$\mathbf{G}^{(1)} = \left(\begin{array}{c} \mathbf{G}^{(10)} \\ \hline \mathbf{0} \end{array}\right) = \left(\begin{array}{cccc} g_0^{[k]} & g_1^{[k]} & \cdots & g_{n-1}^{[k]} \\ g_0^{[k+1]} & g_1^{[k+1]} & \cdots & g_{n-1}^{[k+1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{[k+k^{(1)}-1]} & g_1^{[k+k^{(1)}-1]} & \cdots & g_{n-1}^{[k+k^{(1)}-1]} \\ \hline \multicolumn{4}{c}{0} \end{array}\right). \tag{6.14}$$

Table 6.1 denotes the Gabidulin codes, defined by submatrices of the generator matrix, their minimum rank distances and their block rank-metric error-erasure *bounded minimum distance* (BMD) decoders—realized e.g. by the decoder from Subsection 3.2.3. These BMD decoders decode correctly if (3.32) is fulfilled for the corresponding minimum rank distance. If we consider unit memory codes with $k = k^{(1)}$, then $d_{00} = d_{10} = n - k + 1$, $d_\sigma = n - 2k + 1$ and $d_{01} = \infty$, since $\mathbf{G}^{(01)}$ does not exist.

Table 6.1. Submatrices of (P)UM code from Definition 6.5 and their block codes.

Generator matrix	Defined code	Code parameters	Minimum rank distance	BMD decoder
$\mathbf{G}^{(0)}$	C_0	$\mathsf{Gab}[n,k]$	$d_0 = n - k + 1$	$\mathsf{BMD}(\mathsf{C}_0)$
$\begin{pmatrix} \mathbf{G}^{(01)} \\ \mathbf{G}^{(10)} \end{pmatrix}$	C_1	$\mathsf{Gab}[n,k]$	$d_1 = n - k + 1$	$\mathsf{BMD}(\mathsf{C}_1)$
$\mathbf{G}^{(00)}$	C_{00}	$\mathsf{Gab}[n,k^{(1)}]$	$d_{00} = n - k^{(1)} + 1$	not needed
$\mathbf{G}^{(01)}$	C_{01}	$\mathsf{Gab}[n,k-k^{(1)}]$	$d_{01} = n - k + k^{(1)} + 1$	$\mathsf{BMD}(\mathsf{C}_{01})$
$\mathbf{G}^{(10)}$	C_{10}	$\mathsf{Gab}[n,k^{(1)}]$	$d_{10} = n - k^{(1)} + 1$	not needed
$\mathbf{G}_\sigma = \begin{pmatrix} \mathbf{G}^{(00)} \\ \mathbf{G}^{(01)} \\ \mathbf{G}^{(10)} \end{pmatrix}$	C_σ	$\mathsf{Gab}[n,k+k^{(1)}]$	$d_\sigma = n - k - k^{(1)} + 1$	$\mathsf{BMD}(\mathsf{C}_\sigma)$

Lemma 6.3 (Construction is in Minimal Basic Encoding Form).
Let a (P)UM code based on Gabidulin codes be defined by its generator matrix $\mathbf{G}$ *as in Definition 6.5. Then,* $\mathbf{G}$ *is in minimal basic encoding form.*

Proof. The proof is straight-forward to the proof of Lemma 6.2. ∎

In the following, we calculate the active row rank distance (Definition 6.3) by cutting the semi-infinite generator matrix of the PUM code from Definition 6.5 into parts. Pay attention that each code block of length n can be seen as a codeword of C_σ.

Theorem 6.5 (Lower Bound on Active Distances).
Let $k + k^{(1)} \leq n \leq m$, where $k^{(1)} \leq k$. Let C be a $\mathsf{UM}(n, k)$, respectively $\mathsf{PUM}(n, k|k^{(1)})$, code over $\mathbb{F}_{q^m}$ as in Definition 6.5. The active row, column and reverse column rank distances $d^{(r)}_{j,R}$, $d^{(c)}_{j,R}$ and $d^{(rc)}_{j,R}$ (Definition 6.3 and Equation (6.3)) of C are lower bounded by

$$\begin{aligned} d^{(r)}_{1,R} &\geq \delta^{(r)}_{1,R} = d_{01}, \qquad d^{(r)}_{j,R} \geq \delta^{(r)}_{j,R} = d_0 + (j-2) \cdot d_\sigma + d_1, \quad \forall j \geq 2, \\ d^{(c)}_{j,R} &\geq \delta^{(c)}_{j,R} = d_0 + (j-1) \cdot d_\sigma, \quad \forall j \geq 1, \\ d^{(rc)}_{j,R} &\geq \delta^{(rc)}_{j,R} = (j-1) \cdot d_\sigma + d_1, \quad \forall j \geq 1, \end{aligned}$$

where $d_{01} = n - k + k^{(1)} + 1$ for $k^{(1)} < k$ and $d_{01} = \infty$ for $k^{(1)} = k$; $d_0 = d_1 = n - k + 1$ and $d_\sigma = n - k - k^{(1)} + 1$.

Proof. For the estimation of the active row rank distance, the encoder starts in the zero state hence, $\mathbf{u}^{(-1)} = \mathbf{0}$. For the first order active row distance $d^{(r)}_{1,R}$, we look at all code sequences of the form $(\dots\ \mathbf{0}\ \mathbf{c}^{(0)}\ \mathbf{0}\ \dots)$, which is only possible if $\mathbf{u}^{(0)} = (0\ \dots\ 0\ u^{(0)}_{k^{(1)}}\ \dots\ u^{(0)}_{k-1})$ and $\mathbf{u}^{(i)} = \mathbf{0}$, $\forall i \geq 1$. In this case, $\mathbf{c}^{(0)} \in \mathsf{C}_{01}$ and the encoder returns immediately to the zero state. For the UM case, also $\mathbf{u}^{(0)} = \mathbf{0}$ and the only codeword in $\mathsf{C}^{(r)}_0$ is the all-zero codeword and thus, $d^{(r)}_{1,R} = \infty$.
For higher orders of $d^{(r)}_{j,R}$, we have to consider all code sequences, starting with $\mathbf{c}^{(0)} \in \mathsf{C}_0$ (since $\mathbf{u}^{(-1)} = \mathbf{0}$), followed by $j-2$ codewords of C_σ and one final code block, resulting from $\mathbf{u}^{(j-1)} = (0\ \dots\ 0\ u^{(j-1)}_{k^{(1)}}\ \dots\ u^{(j-1)}_{k-1})$ and for the UM case $\mathbf{u}^{(j-1)} = \mathbf{0}$. For the UM and the PUM case, the block $\mathbf{u}^{(j-2)}$ is arbitrary, therefore $\mathbf{c}^{(j-1)} = \mathbf{u}^{(j-1)} \cdot \mathbf{G}^{(0)} + \mathbf{u}^{(j-2)} \cdot \mathbf{G}^{(1)} \in \mathsf{C}_1$.
For the estimation of $d^{(c)}_{j,R}$, the encoder starts in the zero state but ends in any state. Thus, $\mathbf{c}^{(0)} \in \mathsf{C}_0$ is followed by $j-1$ arbitrary information blocks resulting in codewords from C_σ.
For the active reverse column rank distances, we start in any, hence, all first $j-1$ blocks are from C_σ. The last block is from C_1 in order to end in the zero state. ∎

We call the lower bounds of $d^{(r)}_{j,R}$, $d^{(c)}_{j,R}$, $d^{(rc)}_{j,R}$ *designed* active distances $\delta^{(r)}_{j,R}$, $\delta^{(c)}_{j,R}$, $\delta^{(rc)}_{j,R}$ in the following.

Corollary 6.2 (Free Rank Distance and Slope of our Construction).
Let $k + k^{(1)} \leq n \leq m$, where $k^{(1)} \leq k$. Let C be a $\mathsf{UM}(n, k)$, respectively $\mathsf{PUM}(n, k|k^{(1)})$, code over $\mathbb{F}_{q^m}$ as in Definition 6.5. The free rank distance $d_{f,R}$ for $k^{(1)} = k$ is

$$d_{f,R} \geq \min_j \left\{ \delta^{(r)}_{j,R} \right\} = d_0 + d_1 = 2(n - k + 1),$$

and for $k^{(1)} < k$:

$$d_{f,R} = \min_j \left\{ \delta^{(r)}_{j,R} \right\} = d_{01} = n - k + k^{(1)} + 1 = n - k + \nu + 1.$$

The slope σ_R of C for both cases is:

$$\sigma_R \geq \lim_{j\to\infty}\left\{\frac{\delta_{j,R}^{(r)}}{j}\right\} = d_\sigma = n-k-k^{(1)}+1.$$

Thus, for any $k^{(1)} < k$, the construction achieves the upper bound on the free rank distance of PUM codes (6.5). When $k^{(1)} = k = 1$, we meet the upper bound on the free rank distance of UM codes (6.4). For $k^{(1)} = 1 \leq k$, the upper bound on the slope is attained.

If we compare this to the construction from Subsection 6.2.1,—with free rank distance and slope as in Theorem 6.4—we see that they both attain the upper bound on the free rank distance for $k < k^{(1)}$. It depends on the concrete parameters n, k, $k^{(1)}$, which slope is higher.

The construction based on the parity-matrix (Theorem 6.2) requires that $R = k/n \geq \mu_H/(\mu_H+1)$ and provides therefore a high-rate code, whereas the construction based on the generator matrix (Definition 6.5) results in a low-rate code since $k+k^{(1)} \leq n$ has to hold.

6.3 Error-Erasure Decoding of PUM Gabidulin Codes

This section provides an efficient error-erasure decoding algorithm for (P)UM codes as in Definition 6.5, using the block rank-metric decoders of the underlying Gabidulin codes of Table 6.1.

6.3.1 Bounded Row Distance Condition and Decoding Idea

We consider the terminated generator matrix of a (P)UM code as in (2.8) and therefore we look at blocks of length $N+\mu = N+1$. Let the received sequence $\mathbf{r} = (\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \dots\ \mathbf{r}^{(N)}) \in \mathbb{F}_{q^m}^{n(N+1)}$ be given and let the matrix sequence $\mathbf{R} = (\mathbf{R}^{(0)}\ \mathbf{R}^{(1)}\ \dots\ \mathbf{R}^{(N)}) \in \mathbb{F}_q^{m\times n(N+1)}$ denote the matrix representation of $\mathbf{r}$ according to the mapping from Definition 2.1. Let $\mathbf{r}^{(i)} = \mathbf{c}^{(i)} + \mathbf{e}^{(i)}$, for all $i \in [0,N]$. The matrix representation $\mathbf{R}^{(i)} \in \mathbb{F}_q^{m\times n}$ can be decomposed as in (3.33), including $t^{(i)}$ errors, $\varrho^{(i)}$ row erasures and $\gamma^{(i)}$ column erasures in rank metric, for all $i \in [0,N]$.

Analog to Justesen's definition in Hamming metric [Jus93], we define a bounded (row rank) distance decoder for convolutional codes in rank metric, incorporating additionally erasures.

Definition 6.6 (Bounded Row Distance Error–Erasure Decoder in Rank Metric).
Given a received sequence $\mathbf{r} = \mathbf{c}+\mathbf{e} \in \mathbb{F}_{q^m}^{n(N+1)}$, a bounded row distance (BRD) error-erasure decoder in rank metric for a convolutional code C guarantees to find the code sequence $\mathbf{c} \in \mathsf{C}$ if

$$\sum_{h=i}^{i+j-1}\left(2\cdot t^{(h)} + \varrho^{(h)} + \gamma^{(h)}\right) < \delta_{j,R}^{(r)} \leq d_{j,R}^{(r)}, \qquad \forall i \in [0,N], j \in [0, N-i+1], \tag{6.15}$$

where $t^{(h)}$, $\varrho^{(h)}$, $\gamma^{(h)}$ denote the number of errors, row and column erasures in block $\mathbf{e}^{(h)} \in \mathbb{F}_{q^m}^n$ as in (3.34).

In Algorithm 6.1, we present such a BRD rank-metric error-erasure decoder for (P)UM codes constructed as in Definition 6.5. It is a generalization of the Dettmar–Sorger algorithm [DS95] to rank metric and to error-erasure correction. The generalization to error-erasure decoding can be done in a similar way in Hamming metric.

In the course of this subsection, we explain the idea and the different steps of Algorithm 6.1 in detail.

In Subsection 6.3.2, we prove that the algorithm is actually a BRD error-erasure rank-metric decoder as in Definition 6.6 and we show that its complexity is cubic with the length n of a code block.

The main idea of Algorithm 6.1 is to take advantage of the algebraic structure of the underlying block codes and their efficient decoders (see Table 6.1). We use the outputs of these block decoders to build a reduced trellis. As a final step of our decoder, the usual Viterbi algorithm is applied to this reduced trellis, which has only very few states and therefore the Viterbi algorithm has low complexity.

The first step of Algorithm 6.1 is to decode $\mathbf{r}^{(i)}$, $\forall i \in [1, N-1]$, with $\mathsf{BMD}(\mathsf{C}_\sigma)$, since each code block $\mathbf{c}^{(i)}$ is a codeword of C_σ, $\forall i \in [1, N-1]$. Because of the termination, the first and the last block can be decoded in the codes C_0 and C_{01}, respectively, which have a higher minimum rank distance than C_σ. Let $\mathbf{c}^{(i)\prime}$, for all $i \in [0, N]$, denote the result of decoding $\mathbf{r}^{(i)}$ if it is successful.

For all $i \in [0, N]$, we draw an edge in a reduced trellis with the following metric:

$$m^{(i)} = \begin{cases} \operatorname{rk}(\mathbf{r}^{(i)} - \mathbf{c}^{(i)\prime}), & \text{if } \mathsf{BMD}(\mathsf{C}_\sigma), \mathsf{BMD}(\mathsf{C}_0), \mathsf{BMD}(\mathsf{C}_{01}) \\ & \text{in blocks } 0, [1, N-1], N \text{ is successful,} \\ \left\lfloor \dfrac{d_\sigma + 1 + \varrho^{(i)} + \gamma^{(i)}}{2} \right\rfloor, & \text{else.} \end{cases} \quad \forall i \in [0, N]. \tag{6.16}$$

The metric for the successful case is always smaller than the metric for the non-successful case since

$$\operatorname{rk}(\mathbf{r}^{(i)} - \mathbf{c}^{(i)\prime}) = t^{(i)} + \varrho^{(i)} + \gamma^{(i)} \leq \left\lfloor \frac{d_\sigma + 1 + \varrho^{(i)} + \gamma^{(i)}}{2} \right\rfloor - 1.$$

If the block error-erasure decoder $\mathsf{BMD}(\mathsf{C}_\sigma)$ decodes correctly, the result is $\mathbf{c}^{(i)\prime} = \mathbf{u}^{(i)}\mathbf{G}^{(0)} + (u_0^{(i-1)}\ u_1^{(i-1)}\ \dots\ u_{k^{(1)}-1}^{(i-1)}) \cdot \mathbf{G}^{(10)}$. Since the minimum distance $d_\sigma \geq 1$, we can reconstruct the whole information vector $\mathbf{u}^{(i)} = (u_0^{(i)}\ u_1^{(i)} \dots\ u_{k-1}^{(i)})$ and the mentioned part of the previous information $(u_0^{(i-1)}\ u_1^{(i-1)} \dots\ u_{k^{(1)}-1}^{(i-1)})$.

Assume, we reconstructed $\mathbf{u}^{(i)}$ and $(u_0^{(i-1)}\ u_1^{(i-1)}\ \dots\ u_{k^{(1)}-1}^{(i-1)})$ in Step 1, then with (2.9) and the encoding rule (2.10), we can calculate:

$$\mathbf{r}^{(i+1)} - (u_0^{(i)}\ u_1^{(i)}\ \dots\ u_{k^{(1)}-1}^{(i)}) \cdot \mathbf{G}^{(10)} = \mathbf{u}^{(i+1)}\mathbf{G}^{(0)} + \mathbf{e}^{(i+1)} \tag{6.17}$$
$$\mathbf{r}^{(i-1)} - (u_0^{(i-1)}\ u_1^{(i-1)}\ \dots\ u_{k^{(1)}-1}^{(i-1)}) \cdot \mathbf{G}^{(00)} = (u_{k^{(1)}}^{(i-1)} \dots\ u_{k-1}^{(i-1)} \mid u_0^{(i-2)} \dots\ u_{k^{(1)}-1}^{(i-2)}) \begin{pmatrix} \mathbf{G}_{01} \\ \mathbf{G}_{10} \end{pmatrix} + \mathbf{e}^{(i-1)}.$$

Hence, Step 2 uses the information from block i to decode $\ell_f^{(i)}$ blocks forward with $\mathsf{BMD}(\mathsf{C}_0)$ and $\ell_b^{(i)}$ blocks backward with $\mathsf{BMD}(\mathsf{C}_1)$ from *any* node found in Step 1. This closes (most of) the gaps between two blocks *correctly* decoded by $\mathsf{BMD}(\mathsf{C}_\sigma)$ (of course, it is not known, which are decoded correctly).

The values $\ell_f^{(i)}$ and $\ell_b^{(i)}$ are defined as follows.

$$\ell_f^{(i)} = \min_j \left(j \;\middle|\; \sum_{h=1}^{j} \left(d_\sigma - m^{(i+h)}\right) \geq \frac{\delta_{j,R}^{(c)} - \sum_{h=1}^{j} \left(\varrho^{(i+h)} + \gamma^{(i+h)}\right)}{2} \right), \tag{6.18}$$

$$\ell_b^{(i)} = \min_j \left(j \;\middle|\; \sum_{h=1}^{j} \left(d_\sigma - m^{(i-h)}\right) \geq \frac{\delta_{j,R}^{(rc)} - \sum_{h=1}^{j} \left(\varrho^{(i-h)} + \gamma^{(i-h)}\right)}{2} \right). \tag{6.19}$$

These definitions are chosen such that we can guarantee correct decoding if the BRD condition (6.15) is fulfilled (see Section 6.3.2).

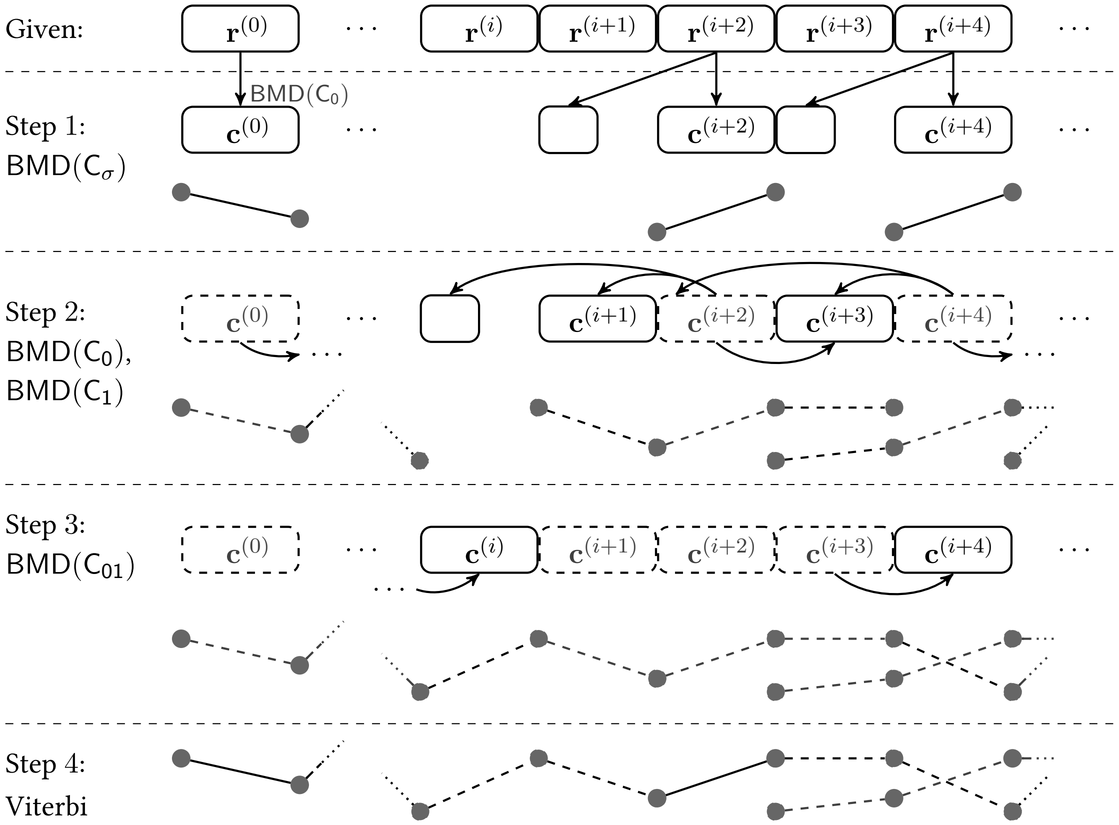


Figure 6.3. Illustration of the different steps of Algorithm 6.1: The received sequence $(\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \dots\ \mathbf{r}^{(N)})$ is given and the different steps and their decoding results are shown. Dashed blocks/edges illustrate that they were found in a previous step.

For Step 3 and some $i \in [0, N-1]$, assume we know $(u_0^{(i+1)}\ u_1^{(i+1)}\ \dots\ u_{k^{(1)}-1}^{(i+1)})$ and $\mathbf{u}^{(i)}$ from Step 1 or 2, then as in (6.17), we can calculate

$$\mathbf{r}^{(i+1)} - (u_0^{(i+1)}\ u_1^{(i+1)}\ \dots\ u_{k^{(1)}-1}^{(i+1)}) \cdot \mathbf{G}^{(00)} - (u_0^{(i)}\ u_1^{(i)}\ \dots\ u_{k^{(1)}-1}^{(i)}) \cdot \mathbf{G}^{(10)} = \\ (u_{k^{(1)}}^{(i+1)}\ u_{k^{(1)}+1}^{(i+1)}\ \dots\ u_{k-1}^{(i+1)}) \cdot \mathbf{G}^{(01)} + \mathbf{e}^{(i+1)},$$

which shows that we can use BMD(C$_{01}$) to close a remaining gap in block $i+1$.

After Step 3, assign as metric to each edge

$$m^{(i)} = \begin{cases} \operatorname{rk}(\mathbf{r}^{(i)} - \mathbf{c}^{(i)\prime}), & \text{if BMD}(\mathsf{C}_0),\ \text{BMD}(\mathsf{C}_1) \text{ or BMD}(\mathsf{C}_{01}) \text{ is successful}, \\ \left\lfloor \dfrac{d_{01} + 1 + \varrho^{(i)} + \gamma^{(i)}}{2} \right\rfloor, & \text{else}, \end{cases} \quad \forall i \in [0, N], \tag{6.20}$$

where $\mathbf{c}^{(i)\prime}$ denotes the result of a successful decoding. For one received block $\mathbf{r}^{(i)}$, there can be several decoding results $\mathbf{c}^{(i)\prime}$ from the different BMD decoders. Thus, there can be more than one edge in the reduced trellis at depth i. Each edge is labeled with regard to (6.20) using its corresponding code block.

Finally, we use the Viterbi algorithm to find the path with the smallest sum rank weight in this reduced trellis. As in [DS95], we use $m^{(i)}$, for all $i \in [0, N]$, as edge metric and the sum over different edges as path metric. The different steps of our decoding algorithm are roughly summarized in Algorithm 6.1, the details can be found in the preceding description and Figure 6.3 illustrates our decoding algorithm.

Algorithm 6.1.
$\mathbf{c} \leftarrow$ BoundedRowDistanceDecoderPUM$(\mathbf{r})$

```
Input: Received sequence $\mathbf{r} = (\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \dots\ \mathbf{r}^{(N)}) \in \mathbb{F}_{q^m}^{n(N+1)}$
Step 1: Decode block $\mathbf{r}^{(0)}$ with $\mathsf{BMD}(\mathsf{C}_0)$
        Decode block $\mathbf{r}^{(i)}$ with $\mathsf{BMD}(\mathsf{C}_\sigma)$, for all $i \in [1, N-1]$
        Decode block $\mathbf{r}^{(N)}$ with $\mathsf{BMD}(\mathsf{C}_{01})$
        Assign metric $m^{(i)}$ as in (6.16), for all $i \in [0, N]$
Step 2: For all found blocks $\mathbf{c}^{(i)}$: decode $\ell_f^{(i)}$ steps forward with $\mathsf{BMD}(\mathsf{C}_0)$,
                                       decode $\ell_b^{(i)}$ steps backward with $\mathsf{BMD}(\mathsf{C}_1)$
Step 3: For all found blocks $\mathbf{c}^{(i)}$: decode $\mathbf{r}^{(i+1)}$ with $\mathsf{BMD}(\mathsf{C}_{01})$
        Assign metric $m^{(i)}$ as in (6.20), for all $i \in [0, N]$
Step 4: Find complete path with smallest sum rank metric using the Viterbi algorithm
Output: Codeword sequence $\mathbf{c} = (\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \dots\ \mathbf{c}^{(N)}) \in \mathbb{F}_{q^m}^{n(N+1)}$
```

In Section 6.3.2, we prove that if (6.15) is fulfilled, then after the three block decoders, all gaps are closed and the Viterbi algorithm finds the path with the smallest sum rank weight.

6.3.2 Proof of Correctness of the Error-Erasure Decoding Algorithm

In the following, we prove that decoding with Algorithm 6.1 is successful if the BRD condition (6.15) is fulfilled. The proof follows the proof of Dettmar and Sorger [Det94, DS95]. Lemma 6.4 shows that the gaps between two *correct* results of Step 1 are not too big and Lemmas 6.5 and 6.6 show that the gap size after Steps 1 and 2 is at most one if the BRD condition (6.15) is fulfilled. Theorem 6.6 shows that these gaps can be closed with $\mathsf{BMD}(\mathsf{C}_{01})$ and the Viterbi algorithm finds the correct path.

Lemma 6.4 (Gap Between two Correct Results of Step 1).
If the BRD condition (6.15) *is satisfied, then the length of any gap between two correct decisions in Step 1 of Algorithm 6.1, denoted by* $\mathbf{c}^{(i)}$, $\mathbf{c}^{(i+j)}$ *is less than* $\min\left\{L_f^{(i)}, L_b^{(i)}\right\}$, *where*

$$L_f^{(i)} = \min_j \left(j \mid \sum_{h=1}^{j} \left(d_\sigma - m^{(i+h)}\right) \geq \frac{\delta_{j,R}^{(r)} - \sum_{h=1}^{j}(\varrho^{(i+h)} + \gamma^{(i+h)})}{2} \right),$$

$$L_b^{(i)} = \min_j \left(j \mid \sum_{h=1}^{j} \left(d_\sigma - m^{(i-h)}\right) \geq \frac{\delta_{j,R}^{(r)} - \sum_{h=1}^{j}(\varrho^{(i-h)} + \gamma^{(i-h)})}{2} \right).$$

Proof. Decoding of a block $\mathbf{r}^{(i)}$ in Step 1 fails or outputs a wrong result if there are at least $(d_\sigma - \varrho^{(i)} - \gamma^{(i)})/2$ errors in rank metric. In such a case, the metric $m^{(i)} = \lfloor (d_\sigma + 1 + \varrho^{(i)} + \gamma^{(i)})/2 \rfloor$ is assigned.

In order to prove the statement, assume there is a gap of at least $L_f^{(i)}$ blocks after Step 1. Then,

$$\sum_{h=1}^{L_f^{(i)}} t^{(i+h)} \geq \sum_{h=1}^{L_f^{(i)}} \frac{d_\sigma - \varrho^{(i+h)} - \gamma^{(i+h)}}{2} \geq \sum_{h=1}^{L_f^{(i)}} \left(d_\sigma - m^{(i+h)}\right)$$
$$\geq \frac{\delta^{(r)}_{L_f^{(i)},R} - \sum_{h=1}^{L_f^{(i)}} \left(\varrho^{(i+h)} + \gamma^{(i+h)}\right)}{2},$$

which follows from the definition of the metric (6.16) and from the definition of $L_f^{(i)}$. This contradicts the BRD condition (6.15). Similarly, we can prove this for $L_b^{(i+j)}$ and the gap size is less than $\min\left\{L_f^{(i)}, L_b^{(i)}\right\}$. ∎

Lemma 6.5 (Correct Path for Few Errors).
Let $\mathbf{c}^{(i)}$ *and* $\mathbf{c}^{(i+j)}$ *be decoded correctly in Step 1 of Algorithm 6.1. Let Step 2 of Algorithm 6.1 decode* $\ell_f^{(i)}$ *blocks in forward direction starting in* $\mathbf{c}^{(i)}$ *and* $\ell_b^{(i+j)}$ *blocks in backward direction starting in* $\mathbf{c}^{(i+j)}$ *(see also* (6.18), (6.19)*).*

Then, the correct path is in the reduced trellis if the BRD condition (6.15) *is satisfied and if in each block less than* $\min\left\{(d_0 - \varrho^{(i)} - \gamma^{(i)})/2, (d_1 - \varrho^{(i)} - \gamma^{(i)})/2\right\}$ *rank errors occurred.*

Proof. If there are less than $\min\left\{(d_0 - \varrho^{(i)} - \gamma^{(i)})/2, (d_1 - \varrho^{(i)} - \gamma^{(i)})/2\right\}$ errors in a block, $\mathsf{BMD}(\mathsf{C}_0)$ and $\mathsf{BMD}(\mathsf{C}_1)$ always yield the correct decision. Due to the definition of $\ell_f^{(i)}$, see (6.18), the forward decoding with $\mathsf{BMD}(\mathsf{C}_0)$ terminates as soon as

$$\sum_{h=1}^{\ell_f^{(i)}} t^{(i+h)} \geq \sum_{h=1}^{\ell_f^{(i)}} \frac{d_\sigma - \varrho^{(i+h)} - \gamma^{(i+h)}}{2} \geq \sum_{h=1}^{\ell_f^{(i)}} \left(d_\sigma - m^{(i+h)}\right)$$
$$\geq \frac{\delta^{(c)}_{\ell_f^{(i)},R} - \sum\limits_{h=1}^{\ell_f^{(i)}} \left(\varrho^{(i+h)} + \gamma^{(i+h)}\right)}{2} = \frac{d_0}{2} + \left(\ell_f^{(i)} - 1\right)\frac{d_\sigma}{2} - \frac{\sum_{h=1}^{\ell_f^{(i)}} \left(\varrho^{(i+h)} + \gamma^{(i+h)}\right)}{2},$$

where the first inequality holds since the decoding result could not be found in Step 1 and the second and third hold due to the definition of the metric (6.16) and the definition of $\ell_f^{(i)}$. Similarly, the backward decoding with $\mathsf{BMD}(\mathsf{C}_1)$ terminates if

$$\sum_{h=1}^{\ell_b^{(i+j)}} t^{(i+j-h)} \geq \frac{d_1}{2} + \left(\ell_b^{(i+j)} - 1\right)\frac{d_\sigma}{2} - \frac{\sum_{h=1}^{\ell_b^{(i+j)}} \left(\varrho^{(i+j-h)} + \gamma^{(i+j-h)}\right)}{2}.$$

The correct path is in the reduced trellis if $\ell_f^{(i)} + \ell_b^{(i+j)} \geq j - 1$, since the gap is then closed. Assume now on the contrary that $\ell_f^{(i)} + \ell_b^{(i+j)} < j - 1$. Since Step 1 was not successful for the blocks in the gap, at least $(d_\sigma - \varrho^{(h)} - \gamma^{(h)})/2$ rank errors occured in every block $\mathbf{r}^{(h)}$, $\forall h \in [i + \ell_f^{(i)} + 1, i + j - \ell_b^{(i+j)} - 1]$, i.e, in the blocks in the gap between the forward and the backward path.

Then,

$$\begin{aligned}
\sum_{h=1}^{j-1} t^{(i+h)} &\geq \sum_{h=1}^{\ell_f^{(i)}} t^{(i+h)} + \sum_{h=1}^{\ell_b^{(i+j)}} t^{(i+j-h)} + \sum_{h=\ell_f^{(i)}+1}^{j-1-\ell_b^{(i+j)}} \frac{d_\sigma - \varrho^{(i+h)} + \gamma^{(i+h)}}{2} \\
&\geq \frac{d_0}{2} + \big(\ell_f^{(i)} - 1\big)\frac{d_\sigma}{2} - \frac{\sum_{h=1}^{\ell_f^{(i)}}(\varrho^{(i+h)} + \gamma^{(i+h)})}{2} \\
&\quad + \frac{d_1}{2} + \big(\ell_b^{(i+j)} - 1\big)\frac{d_\sigma}{2} - \frac{\sum_{h=1}^{\ell_b^{(i+j)}}\big(\varrho^{(i+j-h)} + \gamma^{(i+j-h)}\big)}{2} \\
&\quad + \frac{\big(j-1-\ell_f^{(i)} - \ell_b^{(i+j)}\big)}{2} d_\sigma - \frac{\sum_{h=\ell_f^{(i)}+1}^{j-1-\ell_b^{(i+j)}}\big(\varrho^{(i+h)} + \gamma^{(i+h)}\big)}{2} \\
&\geq \frac{d_0}{2} + \frac{d_1}{2} + \frac{(j-3)}{2} d_\sigma - \frac{\sum_{h=1}^{j-1}\big(\varrho^{(i+h)} + \gamma^{(i+h)}\big)}{2} \\
&= \frac{\delta_{j-1,R}^{(r)} - \sum_{h=1}^{j-1}\big(\varrho^{(i+h)} + \gamma^{(i+h)}\big)}{2},
\end{aligned}$$

which is a contradiction to the bounded row distance condition (6.15) and the statement follows. ∎

Lemma 6.6 (Gap Size is at Most One After Steps 1 and 2).
Let $\mathbf{c}^{(i)}$ and $\mathbf{c}^{(i+j)}$ be decoded correctly in Step 1 of Algorithm 6.1 (with no other correct decisions in between) and let the BRD condition (6.15) *be fulfilled. Let $d_0 = d_1$.*
Then, there is at most one error block $\mathbf{e}^{(h)}$, $h \in [i+1, i+j-1]$, of rank at least $(d_0 - \varrho^{(i)} - \gamma^{(i)})/2$.

Proof. To fail in Step 1, there have to be at least $(d_\sigma - \varrho^{(i)} - \gamma^{(i)})/2$ errors in $\mathbf{r}^{(i)}$, $\forall i \in [i+1, i+j-1]$. If two error blocks in this gap have rank at least $(d_0 - \varrho^{(i)} - \gamma^{(i)})/2$, then

$$\begin{aligned}
\sum_{h=1}^{j-1} t^{(i+h)} &\geq 2 \cdot \frac{d_0}{2} + (j-3) \cdot \frac{d_\sigma}{2} - \frac{\sum_{h=1}^{j-1}\big(\varrho^{(i+h)} + \gamma^{(i+h)}\big)}{2} \\
&\geq \frac{\delta_{j-1,R}^{(r)}}{2} - \frac{\sum_{h=1}^{j-1}\big(\varrho^{(i+h)} + \gamma^{(i+h)}\big)}{2},
\end{aligned}$$

which contradicts (6.15). ∎

Lemmas 6.5 and 6.6 show that if the BRD condition is satisfied, then the correct path is in the reduced trellis after Steps 1 and 2, except for at most one block.

Theorem 6.6 (Correct Path is in Reduced Trellis after Steps 1–3).
If the BRD condition (6.15) *is satisfied, then the correct path is in the reduced trellis after Steps 1–3 of Algorithm 6.1.*

Proof. Lemma 6.6 guarantees that after Step 2, at most one block is missing from the correct path. This gap can be closed in Step 3 with $\mathsf{BMD}(\mathsf{C}_{01})$, which is able to find the correct solution since $d_{01} \geq \delta_{1,R}^{(r)} = d_{f,R}$. ∎

The complexity is determined by the complexity of the BMD rank block error-erasure decoders from Table 6.1 (realized e.g., as in [SKK08, GPB10] and Subsection 3.2.3), which are all in the order $\mathcal{O}(n^2)$

operations in $\mathbb{F}_{q^m}$. Hence, the calculation of the complexity is straight-forward to [DS95, Theorem 3] and we can give the following bound on the complexity without proof.

Theorem 6.7 (Bounded Row Distance Decoding with Algorithm 6.1).
Let $k + k^{(1)} \leq n \leq m$, where $k^{(1)} \leq k$. Let C be a zero-forced terminated $\mathsf{UM}(n, k)$ or $\mathsf{PUM}(n, k|k^{(1)})$ code over $\mathbb{F}_{q^m}$ as in Definition 6.5. Let a received sequence $\mathbf{r} = (\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \dots\ \mathbf{r}^{(N)}) \in \mathbb{F}_{q^m}^{n(N+1)}$ be given.

Then, Algorithm 6.1 finds the code sequence $\mathbf{c} = (\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \dots\ \mathbf{c}^{(N)}) \in \mathbb{F}_{q^m}^{n(N+1)}$ with smallest sum rank distance to $\mathbf{r}$ if the BRD condition is satisfied (6.15). The complexity of decoding one block of length n is upper bounded by

$$\mathcal{O}(d_\sigma n^2) \leq \mathcal{O}(n^3).$$

The analysis of what happens if too many errors occur or if the BRD condition (6.15) is not fulfilled in one or several blocks, is analog to [DS95]. We can give a condition similar to (6.15) for a single block and see that the algorithm returns to the correct path relatively fast.

6.4 Application to Random Linear Network Coding

The motivation for considering convolutional codes in rank metric is to apply them in multi-shot *random linear network coding* (RLNC). In this section, we first explain the model of multi-shot network coding and show how to define lifted (P)UM code in rank metric. Afterwards, we show how decoding of these lifted (P)UM codes reduces to error-erasure decoding of (P)UM codes in rank metric.

There are other contributions devoted to convolutional network codes (see e.g. [EF04, LY06, PR10, GCSM11]). However, none of these code constructions is based on rank metric and deals with the transmission over the operator channel as ours. Our contribution can be seen as an equivalent for convolutional codes to the block code construction from [SKK08].

6.4.1 Multi-Shot Transmission of Lifted PUM Codes

As network channel model we assume a *multi-shot* transmission over the so-called *operator channel*. The operator channel was defined by Kötter and Kschischang in [KK08] and the concept of multi-shot transmission over the operator channel was first considered by Nóbrega and Uchôa-Filho [NU10].

In this network model, a source transmits packets (which are vectors over a finite field) to a sink. The network has several directed links between the source, some internal nodes and the sink. The source and sink apply coding techniques for error control, but have no knowledge about the structure of the network. This means, we consider *non-coherent* RLNC. In a multi-shot transmission, we use the network several times and the internal structure may change in every time instance. In detail, we assume that we use it $N + 1$ times. In the following, we shortly give basic notations for this network channel model. The notations are similar to [SKK08], but we include additionally the time dependency.

Let $\mathbf{X}^{(i)} \in \mathbb{F}_q^{n\times(n+m)}$, $\forall i \in [0, N]$. The rows represent the transmitted packets $X_0^{(i)}, X_1^{(i)}, \dots, X_{n-1}^{(i)}$ $\in \mathbb{F}_q^{n+m}$ at time instance (shot) i. Similarly, let $\mathbf{Y}^{(i)} \in \mathbb{F}_q^{n^{(i)}\times(n+m)}$ be a matrix whose $n^{(i)}$ rows correspond to the received packets $Y_0^{(i)}, Y_1^{(i)}, \dots, Y_{n^{(i)}-1}^{(i)} \in \mathbb{F}_q^{m+n}$. Notice that n and $n^{(i)}$ do not have to be equal since packets can be erased and/or additional error packets might be inserted.

The term *random linear network coding* originates from the behavior of the internal nodes: they create random linear combinations of the packets received so far in the current shot i, $\forall i \in [0, N]$. Additionally, erroneous packets might be inserted into the network and transmitted packets might be lost or erased.

Let the links in the network be indexed from 0 to $\ell - 1$, then, as in [SKK08], let the rows of a matrix $\mathbf{Z}^{(i)} \in \mathbb{F}_q^{\ell \times (n+m)}$ contain the error packets $Z_0^{(i)}, Z_1^{(i)}, \dots, Z_{\ell-1}^{(i)}$ inserted at the links 0 to $\ell - 1$ at shot i. If $Z_j^{(i)} = 0, j \in [0, \ell-1]$, then no corrupt packet was inserted at link $j \in [0, \ell - 1]$ and time i. Due to the linearity of the network, the output can be written as:

$$\mathbf{Y}^{(i)} = \mathbf{A}^{(i)}\mathbf{X}^{(i)} + \mathbf{B}^{(i)}\mathbf{Z}^{(i)}, \tag{6.21}$$

where $\mathbf{A}^{(i)} \in \mathbb{F}_q^{n^{(i)} \times m}$ and $\mathbf{B}^{(i)} \in \mathbb{F}_q^{n^{(i)} \times \ell}$ are the (unknown) channel transfer matrices at time i.

When there are no errors or erasures in the network, the row space of $\mathbf{Y}^{(i)}$ is the same as the row space of $\mathbf{X}^{(i)}$. In [KK08, SKK08] it was shown that subspace codes constructed by lifted MRD codes (as in Lemma 2.18) provide an almost optimal solution to error control in the operator channel. Such lifted MRD codes are a special class of constant-dimension codes (see Subsection 2.3.4). In the following, we define *lifted PUM codes* based on Gabidulin codes in order to use these constant-dimension codes for error correction in multi-shot network coding.

Definition 6.7 (Lifted (Partial) Unit Memory Code).
Let C *be a zero-forced terminated* $\mathsf{PUM}(n, k|k^{(1)})$ *code over* $\mathbb{F}_{q^m}$ *as in Definition 6.5. Represent each code block* $\mathbf{c}^{(i)} \in \mathbb{F}_{q^m}^n$, $\forall i \in [0, N]$, *as matrix* $\mathbf{C}^{(i)} \in \mathbb{F}_q^{m \times n}$ *according to Definition 2.1.*

Then, the lifting of C *is defined by the following set of subspace sequences:*

$$\mathrm{lift}(\mathsf{C}) = \Bigg\{ \Big(\mathcal{R}_q\big([\mathbf{I}_n\ \mathbf{C}^{(0)T}]\big) \quad \mathcal{R}_q\big([\mathbf{I}_n\ \mathbf{C}^{(1)T}]\big) \ \dots \ \mathcal{R}_q\big([\mathbf{I}_n\ \mathbf{C}^{(N)T}]\big) \Big) : \\ \Big(\mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(0)}) \quad \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(1)}) \ \dots \ \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(N)}) \Big) \in \mathsf{C} \Bigg\}.$$

As in Definition 2.18, we denote $\mathrm{lift}(\mathbf{C}^{(i)T}) = \mathcal{R}_q\big([\mathbf{I}_n\ \mathbf{C}^{(i)T}]\big), \forall i \in [0, N]$. We transmit this sequence of subspaces over the operator channel such that each transmitted matrix is a lifted block of a codeword of the rank-metric PUM code, i.e., $\mathbf{X}^{(j)} = [\mathbf{I}_n\ \mathbf{C}^{(i)T}]$, $\forall i \in [0, N]$. Of course, any other basis of the row space can also be chosen as transmitted matrix.

By means of this lifted PUM code, we create dependencies between the different shots in the network. Since each code block of length n is a codeword of the block code C_σ, each transmitted subspace is a codeword of a $\mathsf{CD}_q(n + m, d_s = 2d_\sigma, n)$ constant-dimension code, lying in $\mathcal{G}_q(n + m, n)$, see [SKK08, Proposition 4] and Lemma 2.18.

However, the lifted (P)UM code contains additionally dependencies between the different blocks and for decoding, we obtain therefore a better performance than simply lifting the *block* code C_σ as in Lemma 2.18. Since the PUM code transmits k information symbols per shot, a comparison with a lifted block code of rate k/n is much fairer than comparing it with C_σ (see Example 6.2).

6.4.2 Decoding of Lifted PUM Codes in the Operator Channel

In this section, we will show how the decoding problem in the operator channel reduces to error-erasure decoding of PUM codes based on Gabidulin codes—analog to [SKK08], where it reduces to error-erasure decoding of Gabidulin codes. Since each code block of length n of a $\mathsf{PUM}(n, k|k^{(1)})$ code is a codeword of the block code C_σ, we can directly use the reformulations of Silva, Kschischang and Kötter [SKK08].

Let the transmitted matrix at time instance i be $\mathbf{X}^{(i)} = [\mathbf{I}_n\ \mathbf{C}^{(i)T}]$ and denote by $\mathbf{Y}^{(i)} = [\widehat{\mathbf{A}}^{(i)}\ \widehat{\mathbf{Y}}^{(i)}] \in \mathbb{F}_q^{n^{(i)} \times (n+m)}$ the received matrix after the multi-shot transmission over the operator channel as in (6.21).

The channel transfer matrices $\mathbf{A}^{(i)}$ and $\mathbf{B}^{(i)}$ can be time-variant. Moreover, assume $\mathrm{rk}(\mathbf{Y}^{(i)}) = n^{(i)}$, since linearly dependent received packets are directly discarded. Then, as in [SKK08], we denote the column and row deficiency of $\widehat{\mathbf{A}}^{(i)}$ by:

$$\gamma^{(i)} \overset{\mathrm{def}}{=} n - \mathrm{rk}(\widehat{\mathbf{A}}^{(i)}), \quad \varrho^{(i)} \overset{\mathrm{def}}{=} n^{(i)} - \mathrm{rk}(\widehat{\mathbf{A}}^{(i)}), \qquad \forall i \in [0, N].$$

If we calculate the *reduced row echelon* (RRE) form of $\mathbf{Y}^{(i)}$ (and fill it up with zero rows, if necessary), we obtain the following matrix in $\mathbb{F}_q^{(n+\varrho^{(i)})\times(n+m)}$ (similar to [SKK08, Proposition 7], but in our notation):

$$\mathrm{RRE}_0\left(\mathbf{Y}^{(i)}\right) = \begin{pmatrix} \mathbf{I}_n + \mathbf{B}^{(i,C)T}\mathbf{I}_{\mathcal{U}^{(i)}}^T & \mathbf{R}^{(i)T} \\ \mathbf{0} & \mathbf{A}^{(i,R)T} \end{pmatrix}, \tag{6.22}$$

for a set $\mathcal{U}^{(i)} \subseteq \{1, 2, \dots, n\}$ with $|\mathcal{U}^{(i)}| = \gamma^{(i)}$ such that $\mathbf{I}_{\mathcal{U}^{(i)}}^T \mathbf{R}^{(i)T} = \mathbf{0}$ and $\mathbf{I}_{\mathcal{U}^{(i)}}^T \mathbf{B}^{(i,C)T} = -\mathbf{I}_{\gamma^{(i)}}$, and $\mathbf{I}_{\mathcal{U}^{(i)}}$ denotes the submatrix of $\mathbf{I}_n$ consisting of the columns indexed by $\mathcal{U}^{(i)}$. Moreover, $\mathbf{B}^{(i,C)T} \in \mathbb{F}_q^{n\times\gamma^{(i)}}$ and $\mathbf{A}^{(i,R)T} \in \mathbb{F}_q^{\varrho^{(i)}\times n}$.

Furthermore, it was shown in [SKK08] that $\mathbf{R}^{(i)}$ can be decomposed into

$$\mathbf{R}^{(i)} = \mathbf{C}^{(i)} + \mathbf{A}^{(i,R)}\mathbf{B}^{(i,R)} + \mathbf{A}^{(i,C)}\mathbf{B}^{(i,C)} + \mathbf{A}^{(i,E)}\mathbf{B}^{(i,E)}, \quad \forall i \in [0, N],$$

where $\left(\mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(0)}) \;\; \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(1)}) \;\; \dots \;\; \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{C}^{(N)})\right) \in \mathsf{C}$ and $\mathbf{A}^{(i,R)}$ and $\mathbf{B}^{(i,C)}$ are known to the receiver, since the matrix from (6.22) can be calculated from the channel output. Comparing this equation to (3.33) makes clear that the problem of decoding lifted PUM codes in the operator channel reduces to error-erasure decoding of the PUM code in rank metric. For this purpose, we can use our decoding algorithm from Section 6.3, which is based on rank-metric error-erasure *block* decoders.

Now, let the received matrix sequence $\mathbf{Y} = (\mathbf{Y}^{(0)}\ \mathbf{Y}^{(1)}\ \dots\ \mathbf{Y}^{(N)})$ as output of the operator channel be given, then we show in Algorithm 6.2 how to reconstruct the transmitted information sequence.

Algorithm 6.2.
$\mathbf{u} = (\mathbf{u}^{(0)}\ \mathbf{u}^{(1)}\ \dots, \mathbf{u}^{(N-1)}) \leftarrow$ NetworkPUMDecoder$(\mathbf{Y})$

```
Input: Received sequence $\mathbf{Y} = (\mathbf{Y}^{(0)}\ \mathbf{Y}^{(1)}\ \dots, \mathbf{Y}^{(N)})$,
       where $\mathbf{Y}^{(i)} \in \mathbb{F}_q^{n^{(i)}\times(n+m)}$, $\forall i \in [0, N]$
$\gamma^{(i)} \leftarrow n - \mathrm{rk}(\widehat{\mathbf{A}}^{(i)})$, $\forall i \in [0, N]$
$\varrho^{(i)} \leftarrow n^{(i)} - \mathrm{rk}(\widehat{\mathbf{A}}^{(i)})$, $\forall i \in [0, N]$
Calculate $\mathrm{RRE}_0(\mathbf{Y}^{(i)})$ and therefore $\mathbf{R}^{(i)}$ as in (6.22), $\forall i \in [0, N]$
$\mathbf{r} = (\mathbf{r}^{(0)}\ \mathbf{r}^{(1)}\ \dots\ \mathbf{r}^{(N)}) \leftarrow \left(\mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{R}^{(0)})\ \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{R}^{(1)})\ \dots\ \mathrm{ext}_{\boldsymbol{\beta}}^{-1}(\mathbf{R}^{(N)})\right)$
$\mathbf{c} = (\mathbf{c}^{(0)}\ \mathbf{c}^{(1)}\ \dots\ \mathbf{c}^{(N)}) \leftarrow$ BoundedRowDistanceDecoderPUM$(\mathbf{r})$ with Algorithm 6.1
Reconstruct $\mathbf{u} = (\mathbf{u}^{(0)}\ \mathbf{u}^{(1)}\ \dots, \mathbf{u}^{(N-1)})$
Output: Information sequence $\mathbf{u} = (\mathbf{u}^{(0)}\ \mathbf{u}^{(1)}\ \dots, \mathbf{u}^{(N-1)}) \in \mathbb{F}_{q^m}^{kN}$
```

The asymptotic complexity of Algorithm 6.2 for decoding one matrix $\mathbf{Y}^{(i)}$ of size $n^{(i)} \times (n+m)$ scales cubic in n, since calculating the RRE is at most cubic in n if we use Gaussian elimination. Also, Algorithm 6.1 has asymptotic complexity $\mathcal{O}(n^3)$. The reconstruction of the information sequence out of the code sequence is negligible.

Example 6.2 (Lifted PUM Code for Network Coding).
Let $N+1=7$, $n=8 \leq m$, $k=4$, $k^{(1)}=2$ and therefore $d_0=d_1=5$, $d_{01}=7$ and $d_\sigma=3$ (Table 6.1). Let C be a $\mathsf{PUM}(n,k|k^{(1)})$ code as in Definition 6.5. Construct the lifting of C as in Definition 6.7. Assume, $\mathbf{Y}=(\mathbf{Y}^{(0)}\ \mathbf{Y}^{(1)}\ \ldots\ \mathbf{Y}^{(6)})$ is given as output of the operator channel and apply Algorithm 6.2.

After calculating the RRE (and filling the matrix up with zero rows as in (6.22)*), let the number of errors, row erasures and column erasures in each block be as in Table 6.2. The results of the different decoding steps of Algorithm 6.1 for error-erasure decoding of PUM codes are also shown. In this example the BRD condition* (6.15) *is fulfilled and correct decoding is therefore guaranteed due to Theorem 6.6.*

The code rate of C is $1/2$ and as a comparison with the (lifted) Gabidulin codes from [SKK08], the last line in Table 6.2 shows the decoding of a block Gabidulin code of rate $1/2$ and minimum rank distance $d=5$. For fairness, the last block is also decoded with a $\mathsf{Gab}[8,2]$ code. The block decoder fails in Shots 1 and 5.

However, similar to the ongoing discussion whether block or convolutional codes are better, it depends on the distribution of the errors and erasures, i.e., on the channel, whether the construction from [SKK08] or ours performs better.

Table 6.2. Example for error-erasure decoding of lifted (partial) unit memory codes based on Gabidulin codes.

	Shot i	0	1	2	3	4	5	6
	$\varrho^{(i)}+\gamma^{(i)}$	0	1	3	1	1	0	2
	$t^{(i)}$	2	2	0	1	0	3	2
PUM code	Decoding with C_σ,		×	×	×	✓	×	
	block 0 with C_0,	✓						
	block N with C_{10}							✓
	Decoding with C_0, C_1		×	✓	✓		×	
	Decoding with C_{01}		✓				✓	
Block code	Decoding with $\mathsf{Gab}[8,4]$	✓	×	✓	✓	✓	×	✓

6.5 Summary and Outlook

The topic of this chapter are convolutional codes in rank metric, their decoding and their application to random linear network coding.

First, we have defined general distance measures for convolutional codes based on a modified rank metric—the sum rank metric—and have derived upper bounds on the free rank distance and the slope of (P)UM codes based on the sum rank metric.

Second, we have given two explicit constructions of (partial) unit memory codes based on Gabidulin codes and have calculated their free rank distances and slopes. The first (high-rate) construction is based on the parity-check matrix and the second (low-rate) construction on the generator matrix. Both constructions achieve the upper bound on the free rank distance and it depends on the concrete parameters whose slope is higher.

Third, we have presented an efficient error-erasure decoding algorithm for the (P)UM construction based on the generator matrix. The algorithm guarantees to correct up to half the active row rank distance and its complexity is cubic in the length. Finally, we have shown how constant-dimension codes, which were constructed by lifting the (P)UM code, can be applied for error control in random linear network coding.

As an outlook, it will be interesting to prove Conjecture 6.1 and, more far reaching, to find new codes for error control in network coding, e.g. low-density-parity-check codes in rank metric.

CHAPTER 7

Concluding Remarks

Within this thesis, decoding of block and convolutional codes in rank metric has been considered. Since the invention of codes in rank metric by Delsarte, Gabidulin and Roth, several authors have investigated the properties of such codes. A couple of efficient decoding algorithms for a class of maximum rank distance codes—nowadays called Gabidulin codes—were presented within the last years, most of them similar to renown decoding algorithms for Reed–Solomon codes.

In the course of this dissertation, we have developed a new efficient bounded minimum distance decoding algorithm for Gabidulin codes and an interpolation-based decoding procedure for interleaved Gabidulin codes. Further, we have derived bounds on the list decoding radius of rank-metric codes, and introduced and decoded a class of convolutional codes in rank metric. The main results of this dissertation are summarized in the following.

Chapter 3 is dedicated to decoding of Gabidulin codes. First, we have shown efficient algorithms for calculations with linearized polynomials, including two algorithms for calculating the linearized composition with sub-quadratic complexity. Second, we have presented a bounded minimum distance decoding algorithm for Gabidulin codes, similar to Gao's decoding algorithm for Reed–Solomon codes. We have proven how the linearized Euclidean algorithm can be used in this context to output directly the q-degree-restricted linearized evaluation polynomial of the estimated codeword. Moreover, we have extended this decoding algorithm in order to incorporate not only errors, but also two types of erasures in rank metric: row and column erasures.

Chapter 4 covers interleaved Gabidulin codes and their decoding beyond half the minimum distance. So far, two probabilistic unique decoding approaches have been known for these codes, which both fail with a certain probability since there might be more than one codeword within the decoding radius. We have presented an interpolation-based decoding approach, which relies on solving two linear systems of equations, one for the interpolation step and one for the root-finding step. It can be used as a list decoding algorithm for interleaved Gabidulin codes and guarantees to find all codewords within a certain radius. However, the list size and therefore also the worst-case complexity of the list decoder can become exponential in the length of the code. Alternatively, our decoder can be used as a probabilistic unique decoder, with the same decoding radius and the same upper bound on the failure probability as the known decoders. We have further generalized our decoder to error-erasure decoding.

Up to now, there exists no algorithm which decodes Gabidulin codes beyond half the minimum distance. This motivated us to investigate in Chapter 5 the possibilities of polynomial-time list decoding of rank-metric codes in general and Gabidulin codes in particular. We have derived three bounds on the list size, i.e., on the maximum number of codewords in a ball of radius τ. All three bounds reveal a behavior which is completely different from the one of codes in Hamming metric. The first bound shows that the list size for Gabidulin codes can become exponential when τ is at least the Johnson radius. This implies that there cannot be a polynomial-time list decoding algorithm of Gabidulin codes beyond the Johnson radius. Interesting enough, it is not known for Reed–Solomon codes what happens if τ is slightly greater than the Johnson radius. Our second bound is an upper bound on the list size of any code in rank metric, which we have proven by connections between constant-rank and constant-dimension codes. Exactly these connections helped us to derive the third bound. This bound proves that there exists a code in rank metric over $\mathbb{F}_{q^m}$ of length $n \leq m$ such that the list size can become exponential for any τ greater than half the minimum distance. This implies on the one hand that there is no polynomial upper bound similar to the Johnson bound in Hamming metric and on the other hand, it also shows that our upper bound is almost tight.

Finally, Chapter 6 deals with convolutional codes in rank metric. We have proposed distance measures for convolutional codes in rank metric analog to Hamming metric, namely, the free rank distance, the active row rank distance and the slope and we have derived upper bounds on them. Based on rank-metric block codes (Gabidulin codes), we have given two explicit constructions of convolutional codes in rank metric, one high-rate construction based on the parity-check and one low-rate construction based on the generator matrix. Both define so-called (partial) unit memory codes and achieve the upper bound on the free rank distance. The underlying block codes have enabled us to design an efficient error-erasure decoding algorithm for the second construction, which guarantees to correct all error sequences of rank weight up to half the active row rank distance. We have proven its correctness and have outlined how our convolutional rank-metric codes can be applied to multi-shot random linear network coding.

Future research directions have been given in a short outlook at the end of each chapter.

Appendix

A.1 Proofs for the Linearized Extended Euclidean Algorithm

Proof of Theorem 3.3. With (3.4) and (3.5):

$$a(x) = q'(b'(x^{[h]})) + r'(x^{[h]}) + a''(x) = q'(b(x)) - q'(b''(x)) + r'(x^{[h]}) + a''(x).$$

Define $r''(x) \overset{\text{def}}{=} a''(x) - q'(b''(x))$ and therefore $a(x) = q'(b(x)) + r'(x^{[h]}) + r''(x)$. If $\deg_q \big(r'(x^{[h]}) + r''(x)\big) < d_b$, then $q'(x) = q(x)$ since the linearized division is unique [Ore33a, Theorem 1].

We verify this degree constraint by showing that it holds for each of the terms:

- $\deg_q \big(r'(x^{[h]})\big) < \deg_q b'(x) + h = \deg_q b(x) = d_b$,
- $\deg_q \big(a''(x)\big) < h \leq 2d_b - d_a \leq d_b$,
- $\deg_q \big(q'(b''(x))\big) < \deg_q q'(x) + h \leq \deg_q a'(x) - \deg_q b'(x) + h$
  $\leq (d_a - h) - (d_b - h) + h \leq d_b$.

Hence, $\deg_q r''(x) < d_a - d_b + h$ and $q(x) = q'(x)$. ∎

Proof of Lemma 3.6. We can prove this lemma by induction, assuming that for the outputs of the recursions Equation (3.8) holds with the corresponding stopping degrees. We analyze the degrees of the polynomials in the different lines of Algorithm 3.4 in the following.

- Lines 5–6: $\deg_q a^{(1)}(x) = d_a - h$ and $\deg_q b^{(1)}(x) = d_b - h < d_a/2$.
- Line 8: For the explanation, let us define

  $$\begin{pmatrix} r^{(1)(j-1)}(x) \\ r^{(1)(j)}(x) \end{pmatrix} = \mathbf{Q}^{(1)} \otimes \begin{pmatrix} a^{(1)}(x) \\ b^{(1)}(x) \end{pmatrix},$$

  where $\mathbf{Q}^{(1)}$ is the output of the recursive call. Due to the induction assumption, the degrees of these remainders are restricted by: $\deg_q r^{(1)(j-1)}(x) \geq d_{stop}^{(1)} = \left\lfloor \frac{d_{stop}}{d_a} \cdot \deg_q a^{(1)}(x) \right\rfloor$ and $\deg_q r^{(1)(j)}(x) \leq d_{stop}^{(1)}$.
- Line 9: After the linearized matrix-multiplication with $\mathbf{Q}^{(1)}$, we obtain $\deg_q a(x) = \deg_q r^{(1)(j-1)}(x) + h \geq d_{stop}^{(1)} + h \geq d_{stop} + \frac{d_a - 1}{2d_a}(d_a - d_{stop}) \geq \frac{1}{2}(d_a + d_{stop}) - 1$, where the "$-1$" comes from the floor operation in cases where d_a is odd. Moreover, $\deg_q b(x) = \deg_q r^{(1)(j)}(x) + h \leq \frac{1}{2}(d_a + d_{stop})$, since $\lfloor d_a/2 \rfloor \leq d_a/2$. Note that these values lie in the middle between d_a and d_{stop}, i.e., we have already accomplished the first half of the degree reduction.
- Line 13: After the linearized division in Line 11, $a(x)$ becomes the previous $b(x)$ from Line 9 and therefore, $\deg_q a(x) \leq \frac{1}{2}(d_a + d_{stop})$. Moreover, the q-degree of $b(x)$ reduces by at least one, i.e., $\deg_q b(x) < \frac{1}{2}(d_a + d_{stop})$.
- Line 16: We obtain $h \geq d_{stop}/2$ due to the q-degree restriction of $a(x)$ from Line 13.
- Lines 17–18: The truncation results in polynomials with the following q-degrees: $\deg_q a^{(1)}(x) = \deg_q a(x) - h \leq \frac{1}{2}(d_a + d_{stop}) - d_{stop}/2 = d_a/2$ and $\deg_q b^{(1)}(x) < d_a/2$. Thus, the

recursive call in Line 20 is done with polynomials having *at most half* the q-degree of the original input polynomials.

- Line 20: For the derivation of the q-degree, let us define again

$$\begin{pmatrix} r^{(1)(j-1)}(x) \\ r^{(1)(j)}(x) \end{pmatrix} = \mathbf{Q}^{(1)} \otimes \begin{pmatrix} a^{(1)}(x) \\ b^{(1)}(x) \end{pmatrix},$$

  where $\mathbf{Q}^{(1)}$ is the output of the recursive call. Then, $\deg_q r^{(1)(j-1)}(x) \geq d_{stop}^{(1)}$ and $\deg_q r^{(1)(j)}(x) \leq d_{stop}^{(1)} = \lfloor \deg_q a^{(1)}(x) d_{stop}/d_a \rfloor \leq d_{stop}/2$.
- Output: We define $r^{(j-1)}(x)$ and $r^{(j)}(x)$ as in (3.7) and therefore,

$$\begin{aligned} \deg_q r^{(j-1)}(x) &= \deg_q r^{(1)(j-1)}(x) + h \geq d_{stop}^{(1)} + h = \left\lfloor \tfrac{d_{stop}}{d_a} \cdot (\deg_q a(x) - h) \right\rfloor + h \\ &\geq h\big(1 - \tfrac{d_{stop}}{d_a}\big) + \tfrac{d_{stop}}{d_a}(\deg_q a(x) - 1) \\ &= d_{stop} - \tfrac{d_{stop}}{d_a} \deg_q a(x) + \tfrac{d_{stop}}{d_a}(\deg_q a(x) - 1) > d_{stop} - 1, \end{aligned}$$

  i.e., $\deg_q r^{(j-1)}(x) \geq d_{stop}$ and $\deg_q r^{(j)}(x) = \deg_q r^{(1)(j)}(x) + h \leq d_{stop}^{(1)} + h \leq d_{stop}$.

The same holds inductively within the recursions and the statement follows. ∎

A.2 Proof of the Generalized Transformed Key Equation

First, the following lemma is needed.

Lemma A.1 (Transformed Key Equation for Column Erasures).
Let $\overline{\Gamma^{(C)}}(x)$ be the full q-reverse of $\Gamma^{(C)}(x)$, which is defined as in (3.36) *and let $\widehat{e}^{(C)}(x)$ denote the q-transform of $e^{(C)}(x)$. Then,*

$$\widehat{e}^{(C)}\big(\overline{\Gamma^{(C)}}(x)\big) \equiv 0 \mod (x^{[m]} - x).$$

Proof. The statement holds if and only if $\overline{\widehat{e}^{(C)}(\overline{\Gamma^{(C)}}(x))} \equiv 0 \mod (x^{[m]} - x)$. The i-th coefficient of a linearized polynomial $a(x)$ is denoted by $[a(x)]_i$ in the following. Hence, the i-th coefficient of the aforementioned q-reverse polynomial is (the indices are calculated modulo m):

$$\left[\overline{\widehat{e}^{(C)}(\overline{\Gamma^{(C)}}(x))}\right]_i = \left[\widehat{e}^{(C)}\big(\overline{\Gamma^{(C)}}(x)\big)\right]_{-i}^{[i]} = \sum_{h=0}^{m-1} \widehat{e}_h^{(C)[i]} \overline{\Gamma_{-i-h}^{(C)}}^{[i+h]} = \sum_{j=0}^{m-1} \widehat{e}_{-j}^{(C)[i]} \overline{\Gamma_{j-i}^{(C)}}^{[i-j]}.$$

With

$$\widehat{e}_{-j}^{(C)} = \overline{\widehat{e}_j^{(C)}}^{[-j]}, \qquad \overline{\Gamma_{j-i}^{(C)}}^{[i-j]} = \Gamma_{i-j}^{(C)},$$

we obtain

$$\left[\overline{\widehat{e}^{(C)}(\overline{\Gamma^{(C)}}(x))}\right]_i = \sum_{j=0}^{m-1} \overline{\widehat{e}_j^{(C)}}^{[i-j]} \Gamma_{i-j}^{(C)} = \sum_{j=0}^{m-1} \overline{\widehat{e}_{i-j}^{(C)}}^{[j]} \Gamma_j^{(C)} = \left[\Gamma^{(C)}\big(\overline{\widehat{e}^{(C)}}(x)\big)\right]_i, \quad \forall i \in [0, m-1].$$

Let $\mathbf{g} = \boldsymbol{\beta}^\perp$ (and $\mathbf{g}^\perp = \boldsymbol{\beta}$) and $\widehat{e}^{(C)}(\mathbf{g}) = \widehat{e}^{(C)}(\boldsymbol{\beta}^\perp) = \boldsymbol{\beta} \cdot \mathrm{ext}_{\boldsymbol{\beta}}\left(\mathbf{e}^{(C)}\right) = \boldsymbol{\beta} \cdot \mathbf{E}^{(C)}$. Then, Lemma 3.10 states that $\overline{\widehat{e}^{(C)}}(\boldsymbol{\beta}^\perp) = \overline{\widehat{e}^{(C)}}(\mathbf{g}) = \mathbf{g}^\perp \cdot \mathbf{E}^{(C)T} = \boldsymbol{\beta} \cdot \mathbf{E}^{(C)T}$. Thus,

$$\Gamma^{(C)}\big(\overline{\widehat{e}^{(C)}}(\boldsymbol{\beta}^\perp)\big) = \Gamma^{(C)}\big(\overline{\widehat{e}^{(C)}}(\mathbf{g})\big) = \Gamma^{(C)}\left(\boldsymbol{\beta} \cdot \big(\mathrm{ext}_{\boldsymbol{\beta}}(\mathbf{e}^{(C)})\big)^T\right) = \Gamma^{(C)}\left(\boldsymbol{\beta} \cdot \mathbf{B}^{(C)T} \cdot \mathbf{A}^{(C)T}\right).$$

Hence,

$$\Gamma^{(C)}\big(\overline{\widehat{e}^{(C)}}(g_i)\big) = \Gamma^{(C)}\Big(\sum_{j=0}^{t-1} d_j^{(C)} A_{i,j}^{(C)}\Big) = \sum_{j=0}^{t-1} A_{i,j}^{(C)} \Gamma^{(C)}\big(d_j^{(C)}\big) = 0, \quad \forall i \in [0, m-1],$$

where the $d_j^{(C)}$ are defined as in (3.35) and similar to the proof of Theorem 3.6, the statement follows.

∎

Based on this, we can now give the proof of the generalized transformed key equation (Theorem 3.8).

Proof of Theorem 3.8. Let us split the transformed error into three parts, which correspond to row erasures, column erasures and (full) errors: $\widehat{e}(x) = \widehat{e}^{(R)} + \widehat{e}^{(C)} + \widehat{e}^{(E)}$.
1.) First, consider only column erasures. Due to Lemma A.1, $\widehat{e}^{(C)}(\overline{\Gamma^{(C)}}(x)) = 0 \mod (x^{[m]} - x)$ and therefore also $\Lambda^{(E)}\big(\Lambda^{(R)}\big(\widehat{e}^{(C)}(\overline{\Gamma^{(C)}}(x))\big)\big) \equiv 0 \mod (x^{[m]} - x)$.
2.) Second, consider the row erasures. Since any element in $\mathbb{F}_{q^m}$ can be represented as a linear combination of the elements g_i with coefficients from $\mathbb{F}_q$, we can rewrite $\overline{\Gamma^{(C)}}(g_i) = \sum_{j=0}^{m-1} G_{i,j} g_j$, where $G_{i,j} \in \mathbb{F}_q$. Moreover, $\widehat{e}^{(R)}(g_j) = e_j^{(R)} = \sum_{h=0}^{\varrho-1} B_{h,j}^{(R)} \mathbf{a}_h^{(R)}$ with $B_{h,j}^{(R)} \in \mathbb{F}_q$.
Hence, for all $i \in [0, m-1]$:

$$\begin{aligned}\Lambda^{(R)}\Big(\widehat{e}^{(R)}\big(\overline{\Gamma^{(C)}}(g_i)\big)\Big) &= \Lambda^{(R)}\Big(\widehat{e}^{(R)}\Big(\sum_{j=0}^{m-1} G_{i,j} g_j\Big)\Big) = \sum_{j=0}^{m-1} G_{i,j} \Lambda^{(R)}\Big(\widehat{e}^{(R)}(g_j)\Big) \\ &= \sum_{j=0}^{m-1} G_{i,j} \Lambda^{(R)}\Big(\sum_{h=0}^{\varrho-1} B_{h,j}^{(R)} a_h^{(R)}\Big) = \sum_{j=0}^{m-1} G_{i,j} \sum_{h=0}^{\varrho-1} B_{h,j}^{(R)} \Lambda^{(R)}\big(a_h^{(R)}\big) = 0,\end{aligned}$$

due to the definition of $\Lambda^{(R)}(x)$. Hence, $\Lambda^{(E)}\big(\Lambda^{(R)}(\widehat{e}^{(R)}(\overline{\Gamma^{(C)}}(g_i)))\big) = 0$ for all $i \in [0, m-1]$ and thus, $\Lambda^{(E)}\big(\Lambda^{(R)}(\widehat{e}^{(R)}(\overline{\Gamma^{(C)}}(x)))\big) \equiv 0 \mod (x^{[m]} - x)$.
3.) Third, consider the errors. We denote again $\overline{\Gamma^{(C)}}(g_i) = \sum_{j=0}^{m-1} G_{i,j} g_j$, where $G_{i,j} \in \mathbb{F}_q$ and $\widehat{e}^{(E)}(g_j) = \sum_{h=0}^{t-1} B_{h,j}^{(E)} \mathbf{a}_h^{(E)}$ with $B_{h,j}^{(E)} \in \mathbb{F}_q$. Hence, for all $i \in [0, m-1]$:

$$\begin{aligned}\Lambda^{(R)}\Big(\widehat{e}^{(E)}\big(\overline{\Gamma}^{(C)}(g_i)\big)\Big) &= \Lambda^{(R)}\Big(\widehat{e}^{(E)}\big(\sum_{j=0}^{m-1} G_{i,j} g_j\big)\Big) = \sum_{j=0}^{m-1} G_{i,j} \Lambda^{(R)}\Big(\widehat{e}^{(E)}(g_j)\Big) \\ &= \sum_{j=0}^{m-1} G_{i,j} \Lambda^{(R)}\Big(\sum_{h=0}^{t-1} B_{h,j}^{(E)} a_h^{(E)}\Big) = \sum_{j=0}^{m-1} G_{i,j} \sum_{h=0}^{t-1} B_{h,j}^{(E)} \Lambda^{(R)}\big(a_h^{(E)}\big),\end{aligned}$$

and thus,

$$\Lambda^{(E)}\Big(\Lambda^{(R)}\big(\widehat{e}^{(E)}(\overline{\Gamma^{(C)}}(g_i))\big)\Big) = \sum_{j=0}^{m-1} G_{i,j} \sum_{h=0}^{t-1} B_{h,j}^{(E)} \Lambda^{(E)}\Big(\Lambda^{(R)}\big(a_h^{(E)}\big)\Big) = 0,$$

due to the definition of $\Lambda^{(E)}(x)$ and similar to the proof of Theorem 3.6, it follows that $\Lambda^{(E)}\big(\Lambda^{(R)}\big(\widehat{e}^{(E)}(\overline{\Gamma}^{(C)}(x))\big)\big) \equiv 0 \mod (x^{[m]} - x)$.
The sum of the three parts gives exactly the LHS of (3.37) and the statement follows. ∎

A.3 Comparison of Decoding Approaches for Gabidulin Codes

Let us compare the complexity of our decoding approach from Subsection 3.2.4 to the complexity of known approaches for decoding Gabidulin codes in Table A.1, where the degree of all input polynomials is in the order of n and we do not consider constant factors (which would result in a slight difference between the decoding algorithms from [Gab85, PT91, RP04b, Loi06]).

If we assume that each operation in $\mathbb{F}_{q^m}$ costs $\mathcal{O}(m^2)$ operations over $\mathbb{F}_q$ (see Table 3.1), then $\mathcal{O}(m^3 \log m)$ operations over $\mathbb{F}_q$ is smaller than $\mathcal{O}(n^2)$ operations over $\mathbb{F}_q$.

Table A.1. Complexity of decoding Gabidulin codes

Algorithm	Overall decoding complexity	Methods/Details
Gabidulin [Gab85]	$\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$	Solves key equation by LEEA, recursive procedure to determine error
Gabidulin [Gab92]	$\mathcal{O}(n^3)$ over $\mathbb{F}_{q^m}$	Solves key equation by Gaussian elimination
Paramonov–Tretjakov [PT91], Richter–Plass [RP04a, RP04b]	$\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$	Berlekamp–Massey-like algorithm to solve the key equation
Loidreau [Loi06]	$\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$	Welch–Berlekamp-like algorithm, outputs evaluation polynomial of codeword
Silva–Kschischang [SK09a]	$\mathcal{O}(n^2m^2)$ over $\mathbb{F}_q$	Syndrome calculation with $\mathcal{O}(m^3)$ in $\mathbb{F}_q$, Calculating the root space of the error span polynomial: $\mathcal{O}(m^3)$ over $\mathbb{F}_q$ as in [Ber84].
Hassan–Sidorenko [HS10]	$\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$	Solves key equation with fast Berlekamp–Massey-like algorithm with complexity $\mathcal{O}(\mathcal{M}_m(m) \log m)$
This thesis (Section 3.2.4)	$\mathcal{O}(n^2)$ over $\mathbb{F}_{q^m}$	Gao-like algorithm, solves transformed key equation with LEEA; error-erasure decoding If $\mathcal{D}(m) = \mathcal{M}_m(m)$ and if the calculation of a minimal subspace polynomial costs $\mathcal{M}_m(m)$, then $\mathcal{O}(m^3 \log m)$ over $\mathbb{F}_q$.

Bibliography

References

[ACLY00] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network Information Flow,” *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, Aug. 2000. (Cited on pp. v and 1)

[AH74] A. V. Aho and J. E. Hopcroft, *The Design and Analysis of Computer Algorithms*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., 1974. (Cited on pp. 43 and 46)

[ALR13] D. Augot, P. Loidreau, and G. Robert, “Rank Metric and Gabidulin Codes in Characteristic Zero,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, Istanbul, Turkey. (Cited on pp. v and 2)

[Bas65] L. A. Bassalygo, “New Upper Bounds for Error Correcting Codes,” *Probl. Inf. Transm.*, vol. 1, no. 4, pp. 41–44, 1965. (Cited on pp. 81 and 82)

[Ber68] E. R. Berlekamp, “Nonbinary BCH decoding,” *IEEE Trans. Inform. Theory*, vol. 14, no. 2, p. 242, Mar. 1968. (Cited on p. 33)

[Ber84] E. R. Berlekamp, *Algebraic Coding Theory*, revised ed. Aegean Park Press, Jun. 1984. (Cited on pp. 5, 6, 9, 17, 50, 87, and 124)

[BGU07] D. Boucher, W. Geiselmann, and F. Ulmer, “Skew Cyclic Codes,” *Appl. Algebra Engrg. Comm. Comput.*, vol. 18, no. 4, Aug. 2007. (Cited on p. 16)

[BGY80] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, “Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximants,” *J. Algorithms*, vol. 1, no. 3, pp. 259–295, 1980. (Cited on p. 46)

[BJ86] D. Blessenohl and K. Johnsen, “Eine Verschärfung des Satzes von der Normalbasis,” *J. Algebra*, vol. 103, pp. 141–159, 1986. (Cited on p. 8)

[BK78] R. P. Brent and H. T. Kung, “Fast Algorithms for Manipulating Formal Power Series,” *J. ACM*, vol. 25, no. 4, pp. 581–595, Oct. 1978. (Cited on p. 37)

[BKR10] E. Ben-Sasson, S. Kopparty, and J. Radhakrishnan, “Subspace Polynomials and Limits to List Decoding of Reed–Solomon Codes,” *IEEE Trans. Inform. Theory*, vol. 56, no. 1, pp. 113–120, Jan. 2010. (Cited on pp. 81, 83, 87, 88, and 94)

[BKY07] D. Bleichenbacher, A. Kiayias, and M. Yung, “Decoding Interleaved Reed–Solomon Codes over Noisy Channels,” *Theor. Comput. Sci.*, vol. 379, no. 3, pp. 348–360, Jul. 2007. (Cited on p. 29)

[BL02] T. Berger and P. Loidreau, “Security of the Niederreiter Form of the GPT Public-Key Cryptosystem,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2002, p. 267, Lausanne, Switzerland. (Cited on pp. v and 2)

[BL04] T. Berger and P. Loidreau, "Designing an Efficient and Secure Public-Key Cryptosystem Based on Reducible Rank Codes," in *Indocrypt*, 2004, pp. 218–229. (Cited on pp. v and 2)

[Bla83] R. E. Blahut, *Theory and Practice of Error Control Codes*, 1st ed. Addison-Wesley, 1983. (Cited on p. 9)

[Bla85] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, 1985. (Cited on pp. 43, 44, and 46)

[Bla03] R. E. Blahut, *Algebraic Codes for Data Transmission*, 1st ed. Cambridge University Press, Mar. 2003. (Cited on pp. 5 and 40)

[BMS04] A. Brown, L. Minder, and A. Shokrollahi, "Probabilistic Decoding of Interleaved RS Codes on the q-ary Symmetric Channel," in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2004, p. 326, Chicago, Illinois, USA. (Cited on p. 29)

[BMVT78] E. R. Berlekamp, R. J. McEliece, and H. C. A. Van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Trans. Inform. Theory*, vol. 24, no. 3, pp. 384–386, May 1978. (Cited on p. 12)

[Bos98] M. Bossert, *Kanalcodierung*, 2nd ed. Teubner, 1998. (Cited on pp. 9, 13, 15, 40, and 97)

[BR60] R. C. Bose and D. K. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960. (Cited on p. 10)

[BU09a] D. Boucher and F. Ulmer, "Codes as Modules over Skew Polynomial Rings," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, M. Parker, Ed. Springer, 2009, vol. 5921, pp. 38–55. (Cited on p. 16)

[BU09b] D. Boucher and F. Ulmer, "Coding with Skew Polynomial Rings," *J. Symbolic Comput.*, vol. 44, no. 12, pp. 1644–1656, Dec. 2009. (Cited on p. 16)

[BU12] D. Boucher and F. Ulmer, "Linear Codes Using Skew Polynomials with Automorphisms and Derivations," *Des. Codes Cryptogr.*, pp. 1–27, Jun. 2012. (Cited on p. 16)

[BVP13] C. Bachoc, F. Vallentin, and A. Passuello, "Bounds for Projective Codes from Semidefinite Programming," *Adv. Math. Commun.*, vol. 7, no. 2, pp. 127–145, May 2013. (Cited on pp. v, 1, and 31)

[CB04] Y. Cassuto and J. Bruck, "A Combinatorial Bound on the List Size," California Institute of Technology, Pasadena, CA, USA, Technical Report, 2004. (Cited on p. 83)

[CLU09] L. Chaussade, P. Loidreau, and F. Ulmer, "Skew Codes of Prescribed Distance or Rank," *Des. Codes Cryptogr.*, vol. 50, no. 3, pp. 267–284, Mar. 2009. (Cited on p. 16)

[Del78] P. Delsarte, "Bilinear Forms over a Finite Field with Applications to Coding Theory," *J. Combin. Theory Ser. A*, vol. 25, no. 3, pp. 226–241, 1978. (Cited on pp. v, 1, 5, 23, and 26)

[Det94] U. Dettmar, "Partial Unit Memory Codes," Ph.D. dissertation, University of Darmstadt, Darmstadt, Germany, Jun. 1994. (Cited on pp. 103 and 111)

[DS92] U. Dettmar and S. Shavgulidze, "New Optimal Partial Unit Memory Codes," *Electronic Letters*, vol. 28, pp. 1748–1749, Aug. 1992. (Cited on p. 97)

[DS93] U. Dettmar and U. K. Sorger, “New Optimal Partial Unit Memory Codes based on Extended BCH Codes,” *Electronics Letters*, vol. 29, no. 23, pp. 2024–2025, Nov. 1993. (Cited on p. 97)

[DS95] U. Dettmar and U. K. Sorger, “Bounded Minimum Distance Decoding of Unit Memory Codes,” *IEEE Trans. Inform. Theory*, vol. 41, no. 2, pp. 591–596, 1995. (Cited on pp. 97, 105, 108, 110, 111, and 114)

[EF04] E. Erez and M. Feder, “Convolutional Network Codes,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2004, p. 146, Chicago, IL, USA. (Cited on p. 114)

[Eli57] P. Elias, “List Decoding for Noisy Channels,” Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report 335, 1957. (Cited on pp. 12 and 81)

[ES09] T. Etzion and N. Silberstein, “Error-Correcting Codes in Projective Spaces Via Rank-Metric Codes and Ferrers Diagrams,” *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 2909–2919, Jul. 2009. (Cited on pp. v, 1, and 31)

[ES13] T. Etzion and N. Silberstein, “Codes and Designs Related to Lifted MRD Codes,” *IEEE Trans. Inform. Theory*, vol. 59, pp. 1004–1017, Feb. 2013. (Cited on p. 31)

[EV11] T. Etzion and A. Vardy, “Error-Correcting Codes in Projective Space,” *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 1165–1173, Feb. 2011. (Cited on pp. v, 1, and 31)

[Fau06] C. Faure, “Average Number of Gabidulin Codewords within a Sphere,” in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Sep. 2006, pp. 86–89, Zvenigorod, Russia. (Cited on p. 88)

[Fau09] C. Faure, “Etudes de Systèmes Cryptographiques construits à l’aide de Codes Correcteurs, en Métrique de Hamming et en Métrique Rang (in French),” Ph.D. dissertation, École Polytechnique, Paris, France, Mar. 2009. (Cited on p. 88)

[Fed05] S. V. Fedorenko, “A Simple Algorithm for Decoding Reed–Solomon Codes and its Relation to the Welch-Berlekamp Algorithm,” *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 1196–1198, Mar. 2005. (Cited on p. 52)

[FL05] C. Faure and P. Loidreau, “A New Public-Key Cryptosystem Based on the Problem of Reconstructing p-Polynomials,” in *Int. Workshop Coding Cryptogr. (WCC)*, 2005, pp. 304–315. (Cited on pp. v and 2)

[For70] G. D. Forney, “Convolutional Codes I: Algebraic Structure,” *IEEE Trans. Inform. Theory*, vol. 16, no. 6, pp. 720–738, 1970. (Cited on pp. 13, 14, 15, 16, and 102)

[For73] G. D. Forney, “Structural Analysis of Convolutional Codes via Dual Codes,” *IEEE Trans. Inform. Theory*, vol. 19, no. 4, pp. 512–518, Jul. 1973. (Cited on pp. 13 and 16)

[GA86] E. M. Gabidulin and V. B. Afanassiev, *Kodirovanie v Radioelektronike (Coding in Radio Electronics), in Russian*, M. Radio, Ed., 1986. (Cited on p. 41)

[Gab85] E. M. Gabidulin, “Theory of Codes with Maximum Rank Distance,” *Probl. Inf. Transm.*, vol. 21, no. 1, pp. 3–16, 1985. (Cited on pp. v, 1, 5, 20, 23, 25, 26, 28, 33, 46, 48, 50, 51, 52, 89, and 124)

[Gab92] E. M. Gabidulin, “A Fast Matrix Decoding Algorithm for Rank-Error-Correcting Codes,” *Algebraic Coding*, vol. 573, pp. 126–133, 1992. (Cited on pp. 33, 46, 49, 50, 65, and 124)

[Gad09] M. Gadouleau, “Algebraic Codes for Random Linear Network Coding,” Ph.D. dissertation, Lehigh University, Bethlehem, PA, USA, Apr 2009. (Cited on pp. v and 2)

[Gao93] S. Gao, “Normal Bases over Finite Fields,” Ph.D. dissertation, University of Waterloo, Waterloo, Canada, 1993. (Cited on pp. 7, 8, and 9)

[Gao03] S. Gao, “A New Algorithm for Decoding Reed–Solomon Codes,” *Commun. Inform. Network Sec.*, vol. 712, pp. 55–68, 2003. (Cited on pp. 46, 52, 53, and 54)

[GBL00] E. M. Gabidulin, M. Bossert, and P. Lusina, “Space-Time Codes Based on Rank Codes,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2000, p. 284, Sorrento, Italy. (Cited on pp. v and 2)

[GCSM11] W. Guo, N. Cai, X. Shi, and M. Médard, “Localized Dimension Growth in Random Network coding: A Convolutional Approach,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2011, pp. 1156–1160, St. Petersburg, Russia. (Cited on p. 114)

[GG03] J. Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, 2003. (Cited on pp. 39, 43, and 46)

[Gib96] K. Gibson, “The Security of the Gabidulin Public Key Cryptosystem,” *Advances in Cryptology*, vol. 1070, pp. 212–223, 1996. (Cited on pp. v and 2)

[Gie98] M. Giesbrecht, “Factoring in Skew-polynomial Rings over Finite Fields,” *J. Symb. Computation*, vol. 26, no. 4, pp. 463–486, Oct. 1998. (Cited on p. 16)

[GP03] E. M. Gabidulin and N. I. Pilipchuk, “A New Method of Erasure Correction by Rank Codes,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2003, p. 423, Yokohama, Japan. (Cited on p. 56)

[GP08] E. M. Gabidulin and N. I. Pilipchuk, “Error and Erasure Correcting Algorithms for Rank Codes,” *Des. Codes Cryptogr.*, vol. 49, no. 1-3, pp. 105–122, 2008. (Cited on pp. 33, 55, and 56)

[GPB10] E. M. Gabidulin, N. I. Pilipchuk, and M. Bossert, “Correcting Erasures and Errors in Random Network Coding,” in *Int. Telecomm. Symp. (ITS)*, 2010, Amazonas, Brazil. (Cited on p. 113)

[GPT91a] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, “Ideals over a Noncommutative Ring and their Applications to Cryptography,” in *Eurocrypt*, 1991, Brighton, UK. (Cited on pp. v and 2)

[GPT91b] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, “Rank Errors and Rank Erasures Correction,” in *Int. Colloq. Coding Theory*, 1991, Dilijan, Armenia. (Cited on p. 56)

[GR06] V. Guruswami and A. Rudra, “Limits to List Decoding Reed–Solomon Codes,” *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3642–3649, Aug. 2006. (Cited on p. 83)

[GRS00] O. Goldreich, R. Rubinfeld, and M. Sudan, “Learning Polynomials with Queries: the Highly Noisy Case,” *SIAM J. Discrete Math.*, vol. 13, no. 4, 2000. (Cited on pp. 81, 83, and 94)

[GS99] V. Guruswami and M. Sudan, "Improved Decoding of Reed–Solomon and Algebraic–Geometry Codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999. (Cited on pp. 67 and 81)

[Gur99] V. Guruswami, *List Decoding of Error-Correcting Codes*. Springer, Dec. 1999. (Cited on pp. 81, 82, 83, and 94)

[Gur07] V. Guruswami, *Algorithmic Results in List Decoding*. Now Publishers Inc, Jan. 2007. (Cited on p. 82)

[Gur11] V. Guruswami, "Linear-Algebraic List Decoding of Folded Reed–Solomon Codes," in *IEEE Conf. Comput. Complex.*, Jun. 2011, pp. 77–85. (Cited on pp. 63 and 70)

[GW13] V. Guruswami and C. Wang, "Linear-Algebraic List Decoding for Variants of Reed–Solomon Codes," *IEEE Trans. Inform. Theory*, vol. 59, no. 6, pp. 3257–3268, Jun. 2013. (Cited on pp. 63, 70, and 79)

[GX12] V. Guruswami and C. Xing, "List Decoding Reed–Solomon, Algebraic-Geometric, and Gabidulin Subcodes up to the Singleton Bound," *Electronic Colloq. Comp. Complexity*, vol. 19, no. 146, 2012. (Cited on p. 81)

[GY79] F. Gustavson and D. Yun, "Fast Algorithms for Rational Hermite Approximation and Solution of Toeplitz Systems," *IEEE Trans. Circuits Syst.*, vol. 26-9, pp. 750–755, Sep. 1979. (Cited on p. 46)

[GY06] M. Gadouleau and Z. Yan, "Properties of Codes with the Rank Metric," in *IEEE Global Telecomm. Conf. (GLOBECOM)*, Nov. 2006, pp. 1–5, San Francisco, CA, USA. (Cited on p. 25)

[GY08a] M. Gadouleau and Z. Yan, "Complexity of Decoding Gabidulin Codes," in *42nd Annual Conf. Inform. Sciences and Systems (CISS)*, Mar. 2008, pp. 1081–1085, Princeton, NJ, USA. (Cited on p. 35)

[GY08b] M. Gadouleau and Z. Yan, "Packing and Covering Properties of Rank Metric Codes," *IEEE Trans. Inform. Theory*, vol. 54, no. 9, pp. 3873–3883, 2008. (Cited on p. 25)

[GY10] M. Gadouleau and Z. Yan, "Constant-Rank Codes and Their Connection to Constant-Dimension Codes," *IEEE Trans. Inform. Theory*, vol. 56, no. 7, pp. 3207–3216, Jul. 2010. (Cited on pp. 84, 85, 88, and 93)

[GZ61] D. Gorenstein and N. Zierler, "A Class of Error-Correcting Codes in p^m," *J. Appl. Ind. Math.*, vol. 9, no. 2, pp. 207–214, 1961. (Cited on p. 33)

[Ham50] R. Hamming, "Error-Detecting and Error-Correcting Codes," *The Bell Systems Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950. (Cited on pp. 1 and 10)

[HJZZ99] S. Höst, R. Johannesson, K. S. Zigangirov, and V. V. Zyablov, "Active Distances for Convolutional Codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 658–669, Mar. 1999. (Cited on p. 99)

[HKM$^+$03] T. Ho, R. Kötter, M. Médard, D. R. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2003, p. 442, Yokohama, Japan. (Cited on pp. v and 1)

[HMK$^+$06] T. Ho, M. Médard, R. Kötter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006. (Cited on pp. v and 1)

[Hoc59] A. Hocquenghem, "Codes Correcteurs d'Erreurs," *Chiffres (Paris)*, vol. 2, pp. 147–156, Sep. 1959. (Cited on p. 10)

[HP10] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2010. (Cited on p. 5)

[HS10] Y. Hassan and V. R. Sidorenko, "Fast Recursive Linearized Feedback Shift Register Synthesis," in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Sep. 2010, pp. 162–167, Novosibirsk, Russia. (Cited on pp. 33 and 124)

[Jac43] N. Jacobson, *The Theory of Rings*. American Mathematical Society, Dec. 1943. (Cited on p. 16)

[Jac10] N. Jacobson, *Finite-Dimensional Division Algebras over Fields*, 1st ed. Springer, Jan. 2010. (Cited on p. 16)

[JH01] J. Justesen and T. Høholdt, "Bounds on List Decoding of MDS Codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 4, pp. 1604–1609, May 2001. (Cited on pp. 81, 83, 87, 88, and 94)

[JH04] J. Justesen and T. Høholdt, *A Course in Error-Correcting Codes*. European Mathematical Society, Jan. 2004. (Cited on p. 9)

[Joh62] S. Johnson, "A new Upper Bound for Error-Correcting Codes," *IRE Trans. Inform. Theory*, vol. 8, no. 3, pp. 203–207, Apr. 1962. (Cited on pp. 81 and 82)

[Joh63] S. Johnson, "Improved Asymptotic Bounds for Error-Correcting Codes," *IEEE Trans. Inform. Theory*, vol. 9, no. 3, pp. 198–205, Jul. 1963. (Cited on p. 82)

[Jor02] R. Jordan, "Design Apects of Woven Convolutional Coding," Ph.D. dissertation, Ulm University, Ulm, Germany, Apr. 2002. (Cited on p. 99)

[JPZ04] R. Jordan, V. Pavlushkov, and V. V. Zyablov, "Maximum Slope Convolutional Codes," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2511–2526, 2004. (Cited on pp. 99 and 105)

[JTH04] J. Justesen, C. Thommesen, and T. Høholdt, "Decoding of Concatenated Codes with Interleaved Outer Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2004, p. 328, Chicago, Illinois, USA. (Cited on p. 29)

[Jus93] J. Justesen, "Bounded Distance Decoding of Unit Memory Codes," *IEEE Trans. Inform. Theory*, vol. 39, no. 5, pp. 1616–1627, 1993. (Cited on pp. 97, 98, and 108)

[JZ99] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, 1999. (Cited on pp. 13, 14, 15, 99, and 103)

[KK08] R. Kötter and F. R. Kschischang, "Coding for Errors and Erasures in Random Network Coding," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3579–3591, Jul. 2008. (Cited on pp. v, 1, 6, 31, 114, and 115)

[KL97] V. Y. Krachkovsky and Y. X. Lee, "Decoding for Iterative Reed-Solomon Coding Schemes," *IEEE Trans. Magnetics*, vol. 33, no. 5, pp. 2740–2742, Sep. 1997. (Cited on p. 29)

[KL98] V. Y. Krachkovsky and Y. X. Lee, "Decoding of Parallel Reed–Solomon Codes with Applications to Product and Concatenated Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 1998, p. 55, Cambridge, MA, USA. (Cited on p. 29)

[Kra03] V. Y. Krachkovsky, "Reed–Solomon Codes for Correcting Phased Error Bursts," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2975–2984, Nov. 2003. (Cited on p. 29)

[KSK09] A. Khaleghi, D. Silva, and F. R. Kschischang, "Subspace Codes," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, 2009, vol. 5921, pp. 1–21. (Cited on p. 31)

[Lau79] G. S. Lauer, "Some Optimal Partial-Unit Memory Codes," *IEEE Trans. Inform. Theory*, vol. 23, no. 2, pp. 240–243, Mar. 1979. (Cited on pp. 15, 97, and 100)

[Lee76] L.-N. Lee, "Short Unit-Memory Byte-Oriented Binary Convolutional Codes Having Maximal Free Distance," *IEEE Trans. Inform. Theory*, pp. 349–352, May 1976. (Cited on pp. 15, 97, and 100)

[LGB03] P. Lusina, E. M. Gabidulin, and M. Bossert, "Maximum Rank Distance Codes as Space-Time Codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2757–2760, Oct. 2003. (Cited on pp. v and 2)

[LK04] H.-F. Lu and P. V. Kumar, "Generalized Unified Construction of Space-Time Codes with Optimal Rate-Diversity Tradeoff," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2004, p. 95, Chicago, IL, USA. (Cited on pp. v and 2)

[LN96] R. Lidl and H. Niederreiter, *Finite Fields*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Oct. 1996. (Cited on pp. 5, 6, 7, 8, 17, 18, 50, and 53)

[LO06] P. Loidreau and R. Overbeck, "Decoding Rank Errors Beyond the Error Correcting Capability," in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Sep. 2006, pp. 186–190, Zvenigorod, Russia. (Cited on pp. 30, 63, 64, 65, 66, 67, 69, 74, 75, 76, and 77)

[Loi06] P. Loidreau, "A Welch–Berlekamp Like Algorithm for Decoding Gabidulin Codes," *Coding and Cryptography — Revised selected papers of WCC 2005*, vol. 3969, pp. 36–45, 2006. (Cited on pp. 33, 46, 52, 54, 55, 67, and 124)

[Loi07] P. Loidreau, "Métrique rang et cryptographie (in French)," Mémoire d'habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris 6, Sep 2007. (Cited on p. 33)

[Loi08] P. Loidreau, "Properties of Codes in Rank Metric," in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Jun. 2008, pp. 192–198, Pamporovo, Bulgaria. (Cited on pp. 25 and 26)

[Loi10] P. Loidreau, "Designing a Rank Metric Based McEliece Cryptosystem," in *Post-Quantum Cryptography*, 2010, pp. 142–152, Darmstadt, Germany. (Cited on pp. v and 2)

[Loi12] P. Loidreau, "Asymptotic Behaviour of Codes in Rank Metric over Finite Fields," *Des. Codes Cryptogr.*, pp. 1–14, Jul. 2012. (Cited on p. 25)

[LSC13] W. Li, V. R. Sidorenko, and D. Chen, “On Transform-Domain Decoding of Gabidulin Codes,” in *Int. Workshop Coding Cryptogr. (WCC)*, Apr. 2013, Bergen, Norway. (Cited on pp. 33, 56, and 77)

[LY06] S. Y. R. Li and R. W. Yeung, “On Convolutional Network Coding,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2006, pp. 1743–1747, Seattle, WA, USA. (Cited on p. 114)

[Mas69] J. Massey, “Shift-Register Synthesis and BCH Decoding,” *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969. (Cited on p. 33)

[MBG$^+$93] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, 1st ed. Springer, 1993. (Cited on pp. 5, 7, 8, and 9)

[McE98] R. J. McEliece, *The Algebraic Theory of Convolutional Codes.* Elsevier Science B.V., 1998, ch. 12, Handbook of Coding Theory. (Cited on pp. 13 and 15)

[McE03] R. J. McEliece, “On the Average List Size for the Guruswami–Sudan Decoder,” in *Int. Symp. Commun. Theory Appl. (ISCTA)*, 2003, Ambleside, UK. (Cited on p. 73)

[MMO04] T. Migler, K. E. Morrison, and M. Ogle, “Weight and Rank of Matrices over Finite Fields,” Mar. 2004. [Online] http://arxiv.org/abs/math/0403314 (Cited on p. 24)

[Moo96] E. H. Moore, “A two-fold Generalization of Fermat’s Theorem,” *Bull. Amer. Math. Soc.*, vol. 2, pp. 189–199, 1896. (Cited on p. 18)

[MS74] G. Matsaglia and G. Styan, “Equalities and Inequalities for Ranks of Matrices,” *Linear and Multilinear Algebra*, vol. 2, no. 3, pp. 269–292, Jan. 1974. (Cited on pp. 47 and 84)

[MS88] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* North Holland Publishing Co., 1988. (Cited on pp. 6, 9, and 11)

[Mul54] D. E. Muller, “Application of Boolean Algebra to Switching Circuit Design and to Error Detection,” *IRE Trans. Electr. Computers*, vol. 3, pp. 6–12, 1954. (Cited on p. 10)

[MV10] H. Mahdavifar and A. Vardy, “Algebraic List-Decoding on the Operator Channel,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 1193–1197, Austin, TX, USA. (Cited on p. 81)

[MV12] H. Mahdavifar and A. Vardy, “List-Decoding of Subspace Codes and Rank-Metric Codes up to Singleton Bound,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2012, pp. 1488–1492, Cambridge, MA, USA. (Cited on pp. 63, 70, and 81)

[NU10] R. W. Nóbrega and B. F. Uchôa-Filho, “Multishot Codes for Network Coding Using Rank-Metric Codes,” in *IEEE Wireless Network Coding Conf. (WiNC)*, Jun. 2010, pp. 1–6, Boston, MA, USA. (Cited on pp. 97, 98, and 114)

[OG03] A. V. Ourivski and E. M. Gabidulin, “Column scrambler for the GPT Cryptosystem,” *Discrete Appl. Math.*, vol. 128, no. 1, pp. 207–221, May 2003. (Cited on pp. v and 2)

[Ore33a] Ø. Ore, “On a Special Class of Polynomials,” *Trans. Amer. Math. Soc.*, vol. 35, pp. 559–584, 1933. (Cited on pp. 5, 16, 19, 20, and 121)

[Ore33b] Ø. Ore, “Theory of Non-Commutative Polynomials,” *Ann. Math.*, vol. 34, no. 3, pp. 480–508, 1933. (Cited on pp. 5 and 16)

[Ove06] R. Overbeck, “Extending Gibson’s Attacks on the GPT Cryptosystem,” *Coding and Cryptography — Revised selected papers of WCC 2005*, vol. 3969, pp. 178–188, 2006. (Cited on pp. v and 2)

[Ove07] R. Overbeck, “Public Key Cryptography based on Coding Theory,” Ph.D. dissertation, TU Darmstadt, Darmstadt, Germany, 2007. (Cited on pp. 30, 64, 65, and 76)

[Ove08] R. Overbeck, “Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes,” *J. Cryptology*, vol. 21, no. 2, pp. 280–301, Apr. 2008. (Cited on p. 64)

[Pet60] W. Peterson, “Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes,” *IEEE Trans. Inform. Theory*, vol. 6, no. 4, pp. 459–470, Sep. 1960. (Cited on p. 33)

[Pir88] P. Piret, *Convolutional Codes: An Algebraic Approach.* MIT Press Cambridge, MA, USA, 1988. (Cited on p. 13)

[PMA88] F. Pollara, R. J. McEliece, and K. A. S. Abdel-Ghaffar, “Finite-State Codes,” *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 1083–1089, 1988. (Cited on pp. 97 and 100)

[PR10] K. Prasad and B. S. Rajan, “On Network-Error Correcting Convolutional Codes Under the BSC Edge Error Model,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 2418–2422, Austin, TX, USA. (Cited on p. 114)

[PT91] A. V. Paramonov and O. V. Tretjakov, “An Analogue of Berlekamp-Massey Algorithm for Decoding Codes in Rank Metric,” in *Moscow Inst. Physics and Technology (MIPT)*, 1991. (Cited on pp. 33, 46, 50, and 124)

[PW72] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*, 2nd ed. The MIT Press, 1972. (Cited on p. 9)

[Ree54] I. S. Reed, “A Class of Multiple-Error-Correcting Codes and the Decoding Scheme,” *IRE Trans. Inform. Theory*, vol. 4, no. 4, pp. 38–49, Sep. 1954. (Cited on p. 10)

[Rot91] R. M. Roth, “Maximum-Rank Array Codes and their Application to Crisscross Error Correction,” *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 328–336, 1991. (Cited on pp. v, 1, 5, 23, 26, 33, 46, and 50)

[Rot06] R. M. Roth, *Introduction to Coding Theory.* Cambridge University Press, 2006. (Cited on pp. 5 and 9)

[RP04a] G. Richter and S. Plass, “Error and Erasure Decoding of Rank-Codes with a Modified Berlekamp-Massey Algorithm,” in *ITG Conf. Source Channel Coding (SCC)*, 2004, Erlangen, Germany. (Cited on pp. 33, 46, 50, and 124)

[RP04b] G. Richter and S. Plass, “Fast Decoding of Rank-Codes with Rank Errors and Column Erasures,” in *IEEE Int. Symp. Inf. Theory (ISIT)*, 2004, p. 398, Chicago, IL, USA. (Cited on pp. 33, 50, 56, and 124)

[RS60] I. S. Reed and G. Solomon, “Polynomial Codes Over Certain Finite Fields,” *J. Appl. Ind. Math.*, vol. 8, no. 2, pp. 300–304, 1960. (Cited on p. 10)

[RS96] R. M. Roth and G. Seroussi, “Location-Correcting Codes,” *IEEE Trans. Inform. Theory*, vol. 42, no. 2, pp. 554–565, Mar. 1996. (Cited on p. 56)

[RSKV13] A. S. Rawat, N. Silberstein, O. O. Koyluoglu, and S. Vishwanath, "Optimal Locally Repairable Codes with Local Minimum Storage Regeneration via Rank-Metric Codes," in *Inform. Theory Applications Workshop (ITA)*, Feb. 2013, pp. 1–8, San Diego, CA, USA. (Cited on pp. v and 2)

[SB10] V. R. Sidorenko and M. Bossert, "Decoding Interleaved Gabidulin Codes and Multisequence Linearized Shift-Register Synthesis," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 1148–1152, Austin, TX, USA. (Cited on pp. 30, 63, 65, 66, 67, 69, 74, 75, and 76)

[SB12] V. R. Sidorenko and M. Bossert, "Fast Skew-Feedback Shift-Register Synthesis," *accepted for Des. Codes Cryptogr.*, pp. 1–13, Apr. 2012. (Cited on p. 33)

[Sha48] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell Systems Technical Journal*, vol. 27, pp. 379–423 and 623–656, Jul. 1948. (Cited on p. 1)

[Sil09] D. Silva, "Error Control for Network Coding," Ph.D. dissertation, University of Toronto, Toronto, Canada, 2009. (Cited on pp. v, 2, 30, 33, 53, and 57)

[Sil11] N. Silberstein, "Coding Theory and Projective Spaces," Ph.D. dissertation, Technion—Israel Institute of Technology, Haifa, Israel, Sep 2011. (Cited on pp. v, 1, and 31)

[SJB11] V. R. Sidorenko, L. Jiang, and M. Bossert, "Skew-Feedback Shift-Register Synthesis and Decoding Interleaved Gabidulin Codes," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 621–632, Feb. 2011. (Cited on pp. 30, 65, and 66)

[SK07] D. Silva and F. R. Kschischang, "Rank-Metric Codes for Priority Encoding Transmission with Network Coding," in *Canadian Workshop Inform. Theory (CWIT)*, Jun. 2007, pp. 81–84, Alberta, Canada. (Cited on p. 52)

[SK09a] D. Silva and F. R. Kschischang, "Fast Encoding and Decoding of Gabidulin Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2009, pp. 2858–2862, Seoul, Korea. (Cited on pp. 22, 33, 35, 48, 53, 57, and 124)

[SK09b] D. Silva and F. R. Kschischang, "On Metrics for Error Correction in Network Coding," *IEEE Trans. Inform. Theory*, vol. 55, no. 12, pp. 5479–5490, Dec. 2009. (Cited on p. 31)

[Ska10] V. Skachek, "Recursive Code Construction for Random Networks," *IEEE Trans. Inform. Theory*, vol. 56, no. 3, pp. 1378–1382, Mar. 2010. (Cited on pp. v, 1, and 31)

[SKHN75] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *Information and Control*, vol. 27, no. 1, pp. 87–99, 1975. (Cited on p. 33)

[SKK08] D. Silva, F. R. Kschischang, and R. Kötter, "A Rank-Metric Approach to Error Control in Random Network Coding," *IEEE Trans. Inform. Theory*, vol. 54, no. 9, pp. 3951–3967, 2008. (Cited on pp. v, 1, 2, 30, 32, 33, 55, 56, 60, 97, 98, 113, 114, 115, 116, and 117)

[SRB11] V. R. Sidorenko, G. Richter, and M. Bossert, "Linearized Shift-Register Synthesis," *IEEE Trans. Inform. Theory*, vol. 57, no. 9, pp. 6025–6032, 2011. (Cited on pp. 33 and 50)

[SRKV13] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Optimal Locally Repairable Codes via Rank-Metric Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, Istanbul, Turkey. (Cited on p. 2)

[SRV12] N. Silberstein, A. S. Rawat, and S. Vishwanath, "Error Resilience in Distributed Storage via Rank-Metric Codes," in *Allerton Conf. Communication, Control, Computing (Allerton)*, Oct. 2012, pp. 1150–1157, Monticello, IL, USA. (Cited on pp. v and 2)

[SSB09] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Collaborative Decoding of Interleaved Reed–Solomon Codes and Concatenated Code Designs," *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 2991–3012, 2009. (Cited on p. 29)

[Sud97] M. Sudan, "Decoding of Reed–Solomon Codes beyond the Error-Correction Bound," *J. Complexity*, vol. 13, no. 1, pp. 180–193, Mar. 1997. (Cited on pp. 67 and 81)

[SWC12] V. R. Sidorenko, A. Wachter-Zeh, and D. Chen, "On fast Decoding of Interleaved Gabidulin Codes," in *Int. Symp. Probl. Redundancy Inf. Control Systems*, Sep. 2012, pp. 78–83, St. Petersburg, Russia. (Cited on p. 33)

[TJ83] C. Thommesen and J. Justesen, "Bounds on Distances and Error Exponents of Unit Memory Codes," *IEEE Trans. Inform. Theory*, vol. 29, no. 5, pp. 637–649, 1983. (Cited on pp. 97, 99, and 100)

[Vit67] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967. (Cited on p. 15)

[vL98] J. H. van Lint, *Introduction to Coding Theory*, 3rd ed. Springer, Dec. 1998. (Cited on p. 9)

[WB86] L. R. Welch and E. R. Berlekamp, "Error Correction for Algebraic Block Codes," USA Patent 4 633 470, 1986. (Cited on p. 33)

[Woz58] J. M. Wozencraft, "List Decoding," Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report, 1958. (Cited on pp. 12 and 81)

[WXS03] H. Wang, C. Xing, and R. Safavi-Naini, "Linear Authentication Codes: Bounds and Constructions," *IEEE Trans. Inform. Theory*, vol. 49, no. 4, pp. 866–872, Apr. 2003. (Cited on pp. v, 1, 31, and 90)

[WZB12] A. Wachter-Zeh, A. Zeh, and M. Bossert, "Decoding Interleaved Reed-Solomon Codes Beyond Their Joint Error-Correcting Capability," *accepted for Des. Codes Cryptogr.*, 2012. (Cited on p. 29)

[XF09] S. Xia and F. Fu, "Johnson Type Counds on Constant Dimension Codes," *Des. Codes Cryptogr.*, vol. 50, no. 2, pp. 163–172, Feb. 2009. (Cited on pp. v, 1, 31, and 90)

[XYS11] H. Xie, Z. Yan, and B. W. Suter, "General Linearized Polynomial Interpolation and Its Applications," in *IEEE Int. Symp. Network Coding (Netcod)*, Jul. 2011, pp. 1–4, Beijing, China. (Cited on p. 69)

[ZS94] V. V. Zyablov and V. R. Sidorenko, *On Periodic (Partial) Unit Memory Codes with Maximum Free Distance*, ser. Lecture Notes in Computer Science, 1994, vol. 829, pp. 74–79. (Cited on pp. 97, 101, and 104)

Publications Containing Parts of this Thesis

[Wac12] A. Wachter-Zeh, "Bounds on List Decoding Gabidulin Codes," in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Jun. 2012, pp. 329–334, Pomorie, Bulgaria. (Cited on pp. 82 and 90)

[Wac13a] A. Wachter-Zeh, "Bounds on List Decoding of Rank Metric Codes," *accepted for IEEE Trans. Inform. Theory*, 2013. [Online] http://arxiv.org/abs/1301.4643 (Cited on p. 82)

[Wac13b] A. Wachter-Zeh, "Bounds on Polynomial-Time List Decoding of Rank Metric Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2013, Istanbul, Turkey. (Cited on p. 82)

[WAS11] A. Wachter, V. B. Afanassiev, and V. R. Sidorenko, "Fast Decoding of Gabidulin Codes," in *Int. Workshop Coding Cryptogr. (WCC)*, Apr. 2011, pp. 433–442, Paris, France. (Cited on pp. 33 and 46)

[WAS13] A. Wachter-Zeh, V. B. Afanassiev, and V. R. Sidorenko, "Fast Decoding of Gabidulin Codes," *Des. Codes Cryptogr.*, vol. 66, no. 1, pp. 57–73, Jan. 2013. (Cited on pp. 33 and 46)

[WS12] A. Wachter-Zeh and V. R. Sidorenko, "Rank Metric Convolutional Codes for Random Linear Network Coding," in *IEEE Int. Symp. Network Coding (Netcod)*, Jul. 2012, pp. 1–6, Boston, MA, USA. (Cited on p. 97)

[WSB10] A. Wachter, V. R. Sidorenko, and M. Bossert, "A Fast Linearized Euclidean Algorithm for Decoding Gabidulin Codes," in *Int. Workshop Alg. Combin. Coding Theory (ACCT)*, Sep. 2010, pp. 298–303, Novosibirsk, Russia. (Cited on p. 33)

[WSBZ11a] A. Wachter, V. R. Sidorenko, M. Bossert, and V. V. Zyablov, "On (Partial) Unit Memory Codes Based on Gabidulin Codes," *Probl. Inf. Transm.*, vol. 47, no. 2, pp. 38–51, Jun. 2011. (Cited on p. 97)

[WSBZ11b] A. Wachter, V. R. Sidorenko, M. Bossert, and V. V. Zyablov, "Partial Unit Memory Codes Based on Gabidulin Codes," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Aug. 2011, pp. 2487–2491, St Petersburg, Russia. (Cited on p. 97)

[WZ13] A. Wachter-Zeh and A. Zeh, "Interpolation-Based Decoding of Interleaved Gabidulin Codes," in *Int. Workshop Coding Cryptogr. (WCC)*, Apr. 2013, Bergen, Norway. (Cited on p. 63)