Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Coordenação dos Curso de Pós-Graduação em
Engenharia Elétrica

École Nationale Supérieure des Télécommunications
Département Communications et Électronique

Arquitetura para um Decodificador de Códigos Algébrico-Geométricos Baseados em Curvas de Hermite

Leocarlos Bezerra da Silva Lima

Tese submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Brasil, e à École Nationale Supérieure des Télécommunications, França, como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no domínio da Engenharia Elétrica.

Área de Concentração: Processamento de Informação – Comunicações

Francisco Marcos de Assis, Dr., UFCG
Lirida Alves de Barros Naviner, Dra., ENST
Orientadores

Campina Grande, Paraíba, Brasil

©Leocarlos Bezerra da Silva Lima, 15 de setembro de 2004

Ficha Catalográfica

L732a Lima, Leocarlos Bezerra da Silva.

Arquitetura para um Decodificador de Códigos Algébrico-Geométricos Baseados em Curvas de Hermite / Leocarlos Bezerra da Silva Lima . – Campina Grande : UFCG, 2004.

161 p.: il.

Inclui bibliografia.

Orientadores: Francisco Marcos de Assis e Lirida Alves de Barros Naviner.

Tese (doutorado) UFCG / CCT, Brasil, e ENST, França.

1. Decodificação. 2. Codificação de canal. 3. Geometria algébrica. 4. Arquitetura de circuitos integrados. I. Título.

UFCG/BC CDU 519.725:621.3.049.77

Arquitetura para um Decodificador de Códigos Algébrico-Geométricos Baseados em Curvas de Hermite

Leocarlos Bezerra da Silva Lima

Francisco Marcos de Assis, Dr., UFCG
Orientador

Lirida Alves de Barros Naviner, Dra., ENST
Orientadora

Reginaldo Palazzo Júnior, Ph.D, UNICAMP
Componente da Banca

Ivan Saraiva Silva, Dr., UFRN
Componente da Banca

Jean-François Naviner, Dr., ENST
Componente da Banca

Raimundo Carlos Silvério Freire, Dr., UFCG
Componente da Banca

Campina Grande, Paraíba, Brasil, 15 de setembro de 2004

"Ainda que eu falasse línguas, as dos homens e as dos anjos, se eu não tivesse o amor, seria como um bronze que soa ou como um címbalo que tine. Ainda que eu tivesse o dom da profecia, o conhecimento de todos os mistérios e de toda a ciência, ainda que eu tivesse toda a fé, a ponto de transportar montanhas, se não tivesse o amor, eu nada seria. (...) O amor jamais passará. Quanto às profecias, desaparecerão. Quanto às línguas, cessarão. Quanto à ciência, também desaparecerá. Pois o nosso conhecimento é limitado, e limitada é a nossa profecia. (...)"

1Cor 13

Dedicatória

À minha família, pais, irmãos, tios, avós, e a todos aqueles que sempre tiveram por mim apreço e amor, por tudo que esta conquista representa para eles, assim como para mim.

Agradecimentos

A Ana Paula, que durante estes anos de trabalho foi muito mais que uma esposa, foi minha amiga, minha fortaleza. Obrigado por abdicar dos próprios projetos em favor dos meus e por acreditar em mim acima de tudo.

À minha família, que representa para mim aquilo pelo que mais tenho apreço, minhas raízes, o que me norteia.

Aos meus orientadores, Francisco e Lírida, por terem sido, além de excelentes mestres, amigos.

Aos colegas e amigos Luís Felipe, Edmar, Rex, Protásio, Ioannis, Karim, Kalled, Ivan, Judson, Waslon, Juraci, Wamberto, Ronaldo, Bruno, Chantal, Ângela, dentre outros, que partilharam comigo tanto minhas alegrias quanto minhas desesperanças durante este longo trajeto de desenvolvimento deste trabalho.

Resumo

Esta tese consiste na descrição de uma arquitetura eficiente para um algoritmo de decodificação de códigos algébrico-geométricos (AG) baseados em curvas de Hermite. Este trabalho abrange dois temas distintos que se complementam: o estudo dos algoritmos de decodificação para códigos AG e o desenvolvimento de arquiteturas de implementação em hardware para estes decodificadores. O algoritmo objeto deste trabalho busca iterativamente funções localizadoras e avaliadoras de erros que satisfaçam um critério de equação chave. Uma nova arquitetura é proposta para este decodificador. São descritos operadores otimizados para implementar os cálculos mais freqüentes do decodificador. A descrição da arquitetura deste decodificador segue a descrição de arquiteturas para unidades aritméticas em corpos finitos de característica 2, necessárias à implementação em hardware de qualquer sistema de codificação / decodificação de canal usando códigos de bloco.

Résumé

Cette thèse consiste en une description d'une architecture efficace pour un algorithme de décodage de codes algébriques-géométriques (AG) basés sur des courbes d'Hermite. Ce travail embrasse deux compétences complémentaires distinctes : l'étude des algorithmes de décodage pour des codes AG et le développement d'architectures pour l'implantation matérielle de ces décodeurs. L'algorithme objet de ce travail recherche itérativement les fonctions localisatrices et évaluatrices d'erreurs qui satisfont un critère d'équation clé. Une nouvelle architecture pour ce décodeur est proposée. Des opérateurs optimisés pour les calculs les plus fréquents dans le décodeur sont encore décrits. La description de l'architecture de ce décodeur suit la description des architectures pour les unités arithmétiques sur des corps finis de caractéristique 2, nécessaires à l'implantation de n'importe quel système de codage / décodage de canal en utilisant des codes de bloc.

Abstract

This thesis consists on a description of an efficient architecture for a decoding algorithm of algebraic-geometric codes (AG codes) based on Hermitian curves. This work embraces two distinct complementing competences: the study of decoding algorithms for AG codes and the development of architectures for hardware implementation of these decoders. The algorithm, object of this work, searches error locator and evaluator functions iteratively that satisfy a key equation criterion. A new architecture is proposed for this decoder. Optimized operators to implement the most frequent calculations in the decoder are still proposed. The description of the architecture of this decoder follows the description of architectures for arithmetical units in finite fields of characteristic 2, necessary to implemente any channel coding / decoding system using block codes.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathcal{B}_m$	Base para o espaço vetorial de funções $L(mQ)$, característico de uma curva de Hermite;
C	Um código para correção de erros;
$\vec{c}$	Palavra código (vetor de elementos de um corpo finito);
C_m	Código de Hermite gerado por um espaço de funções $L(mQ)$;
$C(D, G)$	Código algébrico-geométrico construído por funções e definido pelos divisores disjuntos D e G;
$C^*(D, G)$	Código algébrico-geométrico construído por diferenciais e definido pelos divisores disjuntos D e G;
d	Distância mínima de um código de bloco;
D	Divisor dos pontos afins de uma curva algébrica;
$\dim[L]$	Dimensão de um espaço L;
$\deg(G)$	Grau de um divisor G;
$\vec{e}$	Vetor de erros ocorridos na transmissão;
(f)	Divisor principal de uma função f;
$\mathbb{F}_q$	Corpo finito de q elementos;
$\mathbb{F}_q^n$	Espaço vetorial de dimensão n com elementos em $\mathbb{F}_q$;
$\mathbb{F}_q[x, y]$	Anel de polinômios nas variáveis x e y;
$\mathbb{F}_q[x, y]\backslash\mathcal{X}$	Anel de polinômios módulo a equação da curva $\mathcal{X}$ nas variáveis x e y;
$\mathbb{F}_q(x, y)$	Corpo de funções racionais em x e y;

$\mathbb{F}_q(\mathcal{X})$	Corpo de funções definido por uma curva $\mathcal{X}$;
G	Divisor de pontos de uma curva algébrica disjuntos de D;
k	Dimensão de um código de bloco;
$L(G)$	Espaço de funções racionais gerado por um divisor G;
Λ	Conjunto de anti-lacunas em Q;
n	Comprimento de um código de bloco;
(n, k)	Código de bloco de comprimento n e dimensão k;
(n, k, d)	Código de bloco de comprimento n, dimensão k e distância mínima d;
$\mathcal{O}(g)$	Complexidade algorítmica, em que a função g expressa a ordem de grandeza do número de operações básicas em corpo finito (ou iterações) necessárias para a obtenção do resultado;
(ω)	Divisor principal de um diferencial ω;
$\Omega(G)$	Espaço de diferenciais gerado por um divisor G;
$\Omega_{\mathcal{X}}$	Conjunto de diferenciais associados a uma curva $\mathcal{X}$;
$\mathbb{P}^k$	Espaço projetivo de dimensão k;
$P_1, \ \ldots, P_n$	Pontos de uma curva algébrica;
Q	Ponto no infinito de um espaço projetivo;
r	Parâmetro que define uma curva de Hermite $y^r + y = x^{r+1}$;
R	Taxa de informação de um código de blocos. No capítulo 5 em particular representa um anel de funções racionais;
$\mathrm{Res}_P(\omega)$	Resíduo de um diferencial ω em um ponto P;
$\vec{S}$	Vetor de síndromes calculadas a partir de um vetor recebido;
$v_{\mathcal{P}}(f)$	Função de valorização discreta de uma função racional f em um lugar (ponto) $\mathcal{P}$;
$\vec{v}$	Vetor de elementos de um corpo finito recebido após transmissão, possivelmente contaminado pela influência de erros;
$\mathcal{X}$	Curva algébrica;

Lista de Acrônimos e Definições de Termos

AG Classe de códigos conhecidos como códigos algébrico-geométricos;

BCH Classe de códigos lineares cíclicos devida a Bose, Chaudhuri e Hocquenghem;

BMS Algoritmo de decodificação atribuído a Berlekamp, Massey e Sakata. De fato, constitui uma generalização proposta por Sakata do conhecido algoritmo de Berlekamp-Massey;

FPGA Dispositivo composto por uma matriz de células digitais programáveis que implementam funções lógicas segundo sua programação (do inglês *Field Programmable Gate Array*);

GMD Algoritmo de decodificação a decisão suave conhecido como algoritmo de distância mínima generalizada (do inglês *generalized minimum-distance*);

MSR Tipo de circuito multiplicador para elementos em corpos finitos de característica 2 baseado em registradores de deslocamento modificados (do inglês *modified shift register*);

PGZ Algoritmo de decodificação de códigos BCH proposto por Peterson, Gorenstein e Zierler;

RS Classe de códigos cíclicos lineares proposta por Reed e Solomon;

VHDL Linguagem de descrição de hardware de projeto VHSIC (do inglês *VHSIC Hardware Description Language*);

VHSIC Circuitos integrados de muito grande velocidade (do inglês *Very High Speed Integrated Circuits*);

Résumé Étendu

Nous présentons ici un résumé étendu du travail de thèse qui est décrit au long de tout ce document. Cette thèse a été développée dans le contexte d'un doctorat en co-tutelle exécuté à l'Universidade Federal de Campina Grande – UFCG, à Campina Grande-PB, au Brésil, et à l'École Nationale Supérieure des Télécommunications – ENST, à Paris, en France, dans la période de septembre 2000 à août 2004. Ce doctorat a fait partie d'un accord de coopération entre les deux établissements (accord CAPES/COFECUB) et a reçu l'appui financier de la Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES (*cf.* `http://www.capes.gov.br/`), organisation gouvernementale brésilienne.

Ce doctorat consiste en un développement d'une architecture pour un décodeur de codes algébriques-géométriques (AG) basés sur des courbes d'Hermite. Ce sujet peut être démembré en deux plans différents qui se complètent. Dans un premier plan, le travail comprend l'étude des algorithmes efficaces de décodage des codes AG (plan algorithmique), et a été développé par simulation sur ordinateur de ces décodeurs à l'aide des outils Macaulay 2 (*cf.* `http://www.math.uiuc.edu/Macaulay2`) et Matlab. Dans un deuxième plan, le travail comprend la recherche et développement des architectures d'implantation matérielle des décodeurs pour les codes AG (plan architectural), et a été développé par la description en VHDL, simulation et synthèse sur ordinateur ayant employé l'outil ModelSim de la Mentor Graphics (*cf.* `http://www.model.com/`), LeonardoSpectrum de la Mentor Graphics (*cf.* `http://www.mentor.com/leonardospectrum/`) et Quartus II de l'Altera (*cf.* `http://www.altera.com/support/software/sof-quartus.html`).

Sujet de la thèse

Les codes AG ont été proposés premièrement en 1982 par M. A. Tsfasman, S. G. Vlăduţ et T. Zink dans [71]. Ils ont combiné les résultats plus récents de la géométrie algébrique avec l'idée posée par V. D. Goppa en 1970 dans [25] de construire des codes à partir de courbes algébriques sur corps finis. Ces codes présentent des paramètres, comme longueur et distance minimale, meilleurs que d'autres codes couramment utilisés. Par exemple, les codes de Reed-Solomon (RS) sont un cas particulier des codes AG lorsque

la courbe algébrique utilisée est juste une droite. Comme les courbes algébriques peuvent présenter beaucoup plus de points qu'une simple droite, un code AG peut présenter une longueur beaucoup plus grande que les codes RS.

Du point de vue de la relation entre le taux d'information asymptotique, qui exprime la baisse du taux de transmission d'information due à l'utilisation du code, et la distance minimale relative, qui exprime la capacité de correction d'erreurs du code, nous pouvons affirmer que les codes AG sont des *bons codes*. C'est-à-dire, les codes AG ne compromettent pas le taux de transmission au détriment de la capacité de correction, ou vice versa, pour une longueur de code $n \to \infty$, et présentent des paramètres meilleurs que des codes couramment utilisés, comme les codes RS [18], [40].

Le premier algorithme de décodage de codes AG a été proposé en 1989 par Jørn Justesen et autres dans [32]. Ce décodeur comprend la recherche des polynômes localisateurs d'erreurs en deux variables, c'est-à-dire, des polynômes qui ont parmi les zéros communs les points associés aux positions des erreurs. Après cela, plusieurs algorithmes ont été proposés pour le décodage de ces codes.

Dans [22], G.-L. Feng et autres ont proposé un algorithme de décodage pour des codes AG définis sur courbes algébriques plates, qui présente une basse complexité temporelle $\mathcal{O}(mn^2)$, où m est le degré de la courbe algébrique et n est la longueur du code. Un tel algorithme comprend une matrice de syndrome Hankel-structurée par bloc combinée avec une élimination gaussienne efficace qui intègre un algorithme de vote par majorité.

Dans [48], O'Sullivan a présenté un algorithme de décodage pour codes AG définis par un diviseur d'un seul point et basés sur la solution d'une équation clé. Une architecture pour cet algorithme a été rapportée dans [50]. Dans [49], O'Sullivan a étendu cet approche pour fournir simultanément des polynômes localisateurs et évaluateurs d'erreurs. Cette nouvelle approche simplifie le procédé pour déterminer le vecteur erreur.

En ce qui concerne le développement d'architectures pour l'implantation matérielle de décodeurs pour codes AG, peu a été publié. En 1998, Ralf Kötter a proposé un algorithme de décodage de structure régulière et simple en vue de l'implantation VLSI [35]. Outre ce travail, juste quelques publications sont trouvées à propos de ce sujet [2,4,37,42,50,53,62]. Comme contribution dans ce domaine, nous proposons une nouvelle architecture pour l'algorithme de O'Sullivan qui est bien adaptée au décodage matériel de codes basés sur les courbes d'Hermite [16, 69].

Codes d'Hermite

Les codes AG basés sur les courbes d'Hermite, appelés codes d'Hermite, constituent une des classes les plus explorées de codes AG, du fait de ses excellents paramètres et de la simplicité de sa structure.

Considérez un corps fini $\mathbb{F}_q$ de $q = r^2$ éléments, où q est une puissance d'un entier principal. Considérez une *courbe d'Hermite* $\mathcal{X}$ donné par l'équation affine [69]

$$\mathcal{X} : y^r + y = x^{r+1}.$$

Ce type de courbe est caractérisé par un genre $g = \frac{r^2-r}{2}$ et présente r^3 points affines en plus d'un point Q dans l'infini d'un plan projectif.

Considérez maintenant un diviseur $G = mQ$, pour un certain paramètre entier $m \geq 0$. Comme la variable x présente un pole d'ordre r et y un pole d'ordre $r+1$ en Q, on vérifie que

$$\mathcal{B}_m = \left\{x^i y^j : 0 \leq i \leq r,\ j \geq 0,\ ir + j\,(r+1) \leq m\right\}$$

constitue une base pour l'espace vectoriel de fonctions $L(G)$ produit par le diviseur G. On dit, alors, que cela est l'espace de fonctions produit par la courbe $\mathcal{X}$ pour un paramètre m.

Définissez l'ensemble d'entiers

$$\Lambda = \{ir + j\,(r+1) : 0 \leq i \leq r,\ j \geq 0\}.$$

On peut vérifier que Λ, nommé l'ensemble des *anti-lacunes* de $L(G)$, constitue l'ensemble des ordre de pôle des fonctions dans $L(G)$. Une *lacune*, par conséquence, est un ordre de pôle d'une fonction qui n'appartient pas à $L(G)$.

Considérez les deux diviseurs $G = mQ$ et $D = P_1 + \cdots + P_n$, où $P_1, \ldots, P_n$ sont les points rationaux de la courbe $\mathcal{X}$ et $n \leq r^3$.

Definição 1 (Code d'Hermite) *Un* code d'Hermite *dénoté par C_m est donné par*

$$C_m = \{(f(P_1), \ldots, f(P_n)) \mid f \in L(G)\}.$$

La construction du code C_m est directe, considérant la base $\mathcal{B}_m$ pour l'espace de fonctions $L(G)$.

Les codes d'Hermite C_m et $C_{r^3+r^2-r-2-m}$ sont duals entre eux-mêmes. En particulier, si r est un nombre pair et $m = \frac{1}{2}(r^3 + r^2 - r - 2)$, le code C_m est auto-dual.

Cas où $n = r^3$

Considérez la longueur maximale du code d'Hermite C_m, où $n = r^3$. La dimension k de C_m est $k = 0$, pour $m < 0$, et $k = n = r^3$, pour $m > r^3 + r^2 - r - 2$. Vu le paramètre m dans l'intervalle

$$0 \leq m \leq r^3 + r^2 - r - 2,$$

la *dimension* k du code C_m est donnée par [69]

$$k = \begin{cases} |\mathcal{B}_m|, & m \leq r^2 - r - 2, \\ m + 1 - \frac{(r^2 - r)}{2}, & r^2 - r - 2 < m < r^3, \\ r^3 - |\mathcal{B}_{r^3 + r^2 - r - 2 - m}|, & m \geq r^3, \end{cases}$$

où $|\mathcal{B}_m|$ est le nombre d'éléments de la base $\mathcal{B}_m$ pour l'espace $L(mQ)$.

La *matrice génératrice* M_m d'un code d'Hermite C_m, pour $0 \leq m < r^3$, est une matrice $|\mathcal{B}_m| \times r^3$ donnée par

$$M_m = \left[a^i b^j\right]_{|\mathcal{B}_m| \times r^3},$$

où $a, b \in \mathbb{F}_q$, $b^r + b = a^{r+1}$, $0 \leq i \leq r$, $j \geq 0$ et $ir + j(r+1) \leq m$. Prenant le paramètre m dans l'intervalle $r^2 - r - 2 < m < r^3 + r^2 - r - 2$, on a que la matrice $M_{r^3 + r^2 - r - 2 - m}$ consiste en la matrice de parité du code C_m.

Considérant une équation $ir + j(r+1) = k$, où $0 \leq i \leq r$ et $j \geq 0$, on a que

$$(i + j)(r + 1) - i = k \Rightarrow$$
$$i = (-k) \mod (r + 1) \quad \text{e} \quad j = \frac{k - ir}{r + 1}.$$

Ces équations sont utiles dans une implantation logicielle ou matérielle pour la détermination des matrices génératrices et de parité, ainsi que dans une procédure de décodage.

Considérez maintenant $m = ir + j(r+1) < r^3$, où $0 \leq i \leq r$ et $j \geq 0$. Si $j = 0$ (c'est-à-dire, $m \equiv 0 \mod r$) ou $m \leq r^3 - r^2$, alors la *distance minimale* d du code C_m est donnée par

$$d = r^3 - m.$$

Un codeur consiste matériellement en une multiplication d'un vecteur d'information par la matrice M_m.

L'algorithme de O'Sullivan

Un processus complet de décodage utilisant l'algorithme de O'Sullivan inclut les tâches suivantes (*cf.* figure 5.1) :

a) Calcul des syndromes (vecteur $\vec{S}$) à partir du vecteur reçu $\vec{v}$ et de la matrice de parité du code ;

b) Détermination des polynômes évaluateurs ϕ et localisateurs f des erreurs (l'algorithme de décodage proprement dit) ;

c) Détermination du vecteur d'erreurs $\vec{e}$ à partir des polynômes évaluateurs et localisateurs d'erreurs ;

d) Correction du vecteur reçu ;

e) Conversion du vecteur corrigé $\vec{c}$ en vecteur d'information $\vec{i}$.

L'algorithme proposé par O'Sullivan dans [49] que nous décrivons ici et l'architecture que nous présentons ensuite concernent uniquement la tâche (b). En effet, les algorithmes de décodage proposés sont généralement restreints à cette tâche, considérée comme le noyau du processus de décodage [6]. Les autres tâches sont considérées des problèmes de solutions connues.

Considérez un corps fini $\mathbb{F}_q$, où $q = r^2$ et r est un entier (en pratique, pour l'implantation matérielle, on considère toujours r une puissance de 2, tel que $\mathbb{F}_q$ soit un corps de caractéristique 2). Considérez un code d'Hermite de longueur n comme décrit avant. Considérez $\theta_k = x^{\mu(k)}y^{\nu(k)}$ un monôme associé à un ordre de pôle $k = \mu(k)r + \nu(k)(r+1)$ en Q.

Soit $R = \mathbb{F}_q[x,y] / (y^r + y = x^{r+1})$ l'anneau des fonctions avec pôles uniquement dans les points de la courbe, un syndrome $S(f)$ pour une fonction $f = f_{pl-1}\theta_{pl-1} + \cdots + f_1\theta_1 + f_0 \in R$ quelconque est calculé par

$$\begin{aligned} S(f) &= \sum_{k=1}^{n} v_k f(P_k) \\ &= \sum_{i=0}^{pl-1} f_i \sum_{k=1}^{n} v_k \theta_i(P_k) \\ &= \sum_{i=0}^{pl-1} f_i \vec{S}, \end{aligned}$$

où $\vec{v}$ est le vecteur reçu.

Le calcul itératif des polynômes localisateurs et évaluateurs d'erreurs par l'algorithme de O'Sullivan est entièrement basé sur la solution d'une équation clé. Cette équation clé est dérivée d'une série de syndromes h_e définie par

$$h_e = \frac{x^r}{x} \sum_{i=0}^{r} \sum_{j=0}^{\infty} s_{ij} x^{-i} y^{-j},$$

où $0 \leq i \leq r$, $j \geq 0$ et $s_{ij} = S(x^i y^j)$ est un syndrome en considérant $f = x^i y^j$.

Le noyau de l'algorithme est le corollaire 3.11 en [49] qui détermine que f est une fonction localisatrice d'erreurs (fonction qui a des zéros dans les points correspondant aux positions des erreurs) si et seulement si $fh_e \in R$. Basé sur ce résultat, l'algorithme cherche itérativement des pairs de fonctions $f, \phi \in R$ (fonctions localisatrices et évaluatrices d'erreurs, respectivement) qui satisfont une équation clé

$$fh_e = \phi.$$

À chaque itération, l'algorithme élargit l'espace de recherche de fonctions et vérifie si les paires de fonctions produites dans la dernière itération $\dot{f}$ et $\dot{\phi}$ (un point sur une fonction indique qu'elle a été produite dans la dernière itération) satisfont encore l'équation clé. Si une paire $\dot{f}$ et $\dot{\phi}$ ne satisfait plus l'équation clé, une nouvelle paire f et ϕ est calculée à partir de $\dot{f}$ et $\dot{\phi}$ et de fonctions auxiliaires g et ψ. Ces fonctions auxiliaires sont des fonctions f et ϕ, respectivement, qui n'ont pas satisfait l'équation clé dans une itération précédente. La mise à jour des fonctions constitue le pas 3 de l'algorithme 2 présenté ensuite.

À travers la proposition 4.1 en [49], le vecteur d'erreurs $\vec{e}$ est déterminé par l'équation

$$e_i = \frac{\phi}{f'}(P_i),$$

où f' dénote la dérivée première de f. Après un nombre suffisant d'itérations, on garantit que, parmi les fonctions localisatrices fournies par l'algorithme, il y a au moins une fonction f tel que f' ne s'annule pas dans un point associé à une position d'une erreur.

Comme référence, l'algorithme de O'Sullivan est décrit dans l'algorithme 2. Dans une itération it, on dénote $\alpha, \beta \in \mathbb{F}_q$ des valeurs de syndromes, $\sigma, \delta \subset \Lambda$ les ensembles de contrôle, et f, g, ϕ, $\psi \in R$ les fonctions localisatrices et évaluatrices d'erreurs et ses fonctions auxiliaires. Des ensembles $\Sigma, \Delta \subset \Lambda$, complémentaires sur Λ, sont définis dans [49]. Les ensembles σ et δ consistent en les minima et maxima de Σ et Δ, respectivement, selon un ordre partiel défini par $a \preceq b$ si $b - a \in \Lambda$ (*cf.* [49]).

À chaque itération, des syndromes $\alpha = S(f)$ sont calculés. Basé sur les valeurs de ces syndromes, les paramètres de contrôle σ, δ sont mis à jour. Ensuite, en utilisant ces nouveaux paramètres de contrôle, les fonctions f, g, ϕ et ψ sont mises à jour. On dénote $\widetilde{f}$ la partie R de f, c'est-à-dire, la fonction f en excluant les termes θ_i, où i est une lacune. On dénote $\overline{\phi}$ la fonction comprenant les termes de ϕ avec évaluation maximale $m - t - 2g + 1$. Les fonctions max et min fournissent les sous-ensembles de Λ des valeurs maximales et minimales par rapport à l'ordre partiel défini par $a \preceq b$ si $b - a \in \Lambda$.

Algoritmo 2 (Décodeur d'O'Sullivan)

Entrées :

– *Le vecteur de syndromes* $\vec{S} = [S(\theta_{n-k}) \cdots S(\theta_0)]$ *du vecteur reçu* $\vec{v}$*, où* θ_i *sont les fonctions de base de l'espace orthogonal au code. Ce vecteur est obtenu par le produit de* $\vec{v}$ *par la matrice de parité du code.* $S(\theta_i) = 0$ *si* i *est une lacune.*

Sorties :

– *Les fonctions* f *et* ϕ *localisatrices et évaluatrices d'erreurs, respectivement.*

Initialisation :

– $it = -1$;
– $\sigma_{-1} = \{0\}$;
– $f(0) = 1$;
– $\phi(0) = 0$.

<<< Pas 1 : Calcul des syndromes $\alpha = S(f)$ >>>

– *Incrémenter* $it = it + 1$;
– *Enregistrer valeurs précédentes* $\dot{f} = f$, $\dot{\phi} = \phi$, $\dot{g} = g$, $\dot{\psi} = \psi$, $\dot{\beta} = \beta$, $\dot{\delta} = \delta$, $\dot{\sigma} = \sigma$;
– *Pour chaque* $s \in \dot{\sigma}$*, calculer* $\alpha(s) = S(\dot{f}\theta_{it-s})$*. Si* $S(\theta_{it})$ *(nécessaire à chaque itération) n'est pas disponible, calculer* $S(\dot{f}\theta_{it-s} - \theta_{it})$ *et employer le vote par majorité pour déterminer* $S(\theta_{it})$*. Ensuite, calculer* $\alpha(s) = S(\dot{f}\theta_{it-s} - \theta_{it}) + S(\theta_{it})$.
– *Le vote par majorité consiste à calculer l'ensemble* $\Gamma = \dot{\Sigma} \cap \left(it - \dot{\Sigma}\right)$ *et, pour chaque* $a \in \Gamma$*, trouver un* $s \in \dot{\sigma}$ *tel que* $s \preceq a$*. Ensuite, choisir* $\zeta \in \mathbb{F}_{r^2}$ *tel que* $g = \theta_{it} + \zeta \dot{f}(s)\theta_{it-s}$ *présente un degré plus petit que* it*. La valeur de* $S(\theta_{it})$ *sera la majorité des valeurs* $S(g)$ *calculées pour chaque* s *choisi.*

<<< Pas 2 : Détermination des paramètres de contrôle >>>

– *Calculer* $\delta' = \{it - s : s \in \dot{\sigma},\ it - s \in \Lambda \text{ et } \alpha(s) \neq 0\}$;
– *Calculer* $\delta = \max\{\delta' \cup \dot{\delta}\}$;
– *Calculer* $\sigma = \min\{t : \nexists\, c \in \delta \text{ with } t \prec c\}$.

<<< Pas 3 : Mise à jour des fonctions >>>

– *Pour chaque* $c \in \delta$*, faire*
  – $g(c) = \begin{cases} \dot{g}(c), & si\ c \in \dot{\delta}, \\ \dot{f}(it - c), & autrement; \end{cases}$
  – $\psi(c) = \begin{cases} \dot{\psi}(c), & si\ c \in \dot{\delta}, \\ \dot{\phi}(it - c), & autrement; \end{cases}$
  – $\beta(c) = \begin{cases} \dot{\beta}(c), & si\ c \in \dot{\delta}, \\ \alpha(it - c), & autrement; \end{cases}$
– *Pour chaque* $t \in \sigma$*, trouver* $s \in \dot{\sigma}$ *et* $a \in \Lambda$ *tels que* $s + a = t$.
  – *Si* $it - t$ *est une lacune, calculer*
    – $f(t) = \dot{f}(s)\theta_a$;
    – $\phi(t) = \dot{\phi}(s)\theta_a + \alpha(s)\theta_{2g-1-(it-t)}$;
  – *Si* $it - t$ *est une anti-lacune et* $\alpha(s) \neq 0$*, trouver* $c \in \dot{\delta}$ *et* $b \in \Lambda$ *tels que* $c - s = it - t$*. Si* $it - t$ *est une anti-lacune et* $\alpha(s) = 0$*, alors* $t = s$ *et on peut choisir* b *et* c *quelconques. Soit* $\gamma = \alpha(s)/\dot{\beta}(c)$*, calculer*
    – $f(t) = \dot{f}(s)\theta_a - \gamma\dot{g}(c)\theta_b$;
    – $\phi(t) = \dot{\phi}(s)\theta_a - \gamma\dot{\psi}(c)\theta_b$;

L'architecture

Par une simple analyse, on arrive à conclure que le décodeur de O'Sullivan présente une complexité plus importante concentrée sur la partie de contrôle de l'algorithme (pas 2 et détermination des entiers a et b dans le pas 3). Cela diffère de ce qu'on trouve sur d'autres décodeurs pour codes AG, qui ont la complexité concentrée surtout sur le calcul de syndromes et la mise à jour des fonctions.

Dans [49], O'Sullivan ne présente pas une discussion sur la complexité de son algorithme, comme c'est l'habitude dans ce type d'article. Il ne donne même pas les informations nécessaires pour déterminer la complexité, comme la longueur des registres des polynômes. Nous avons déduit tous les paramètres manquants pour la détermination de la complexité et pour l'implantation matérielle de ce décodeur. Ici nous proposons une architecture d'implantation pour cet algorithme, ainsi que les résultats obtenus après l'implantation du cas particulier d'un code d'Hermite (64, 54) sur $\mathbb{F}_{16}$.

Unités arithmétiques

Dans le décodeur de O'Sullivan, toutes les opérations sur les symboles de $\mathbb{F}_q$, comme les valeurs de syndromes, les coefficients de fonctions en R etc., sont effectuées dans ce corps, c'est-à-dire, en utilisant des unités arithmétiques adaptées aux corps finis. Le travail de Mastrovito est une excelente référence sur ce sujet [45]. Les architectures des unités que nous avons implantées sont décrites en [15]. Les figures 4.2, 4.6 e 4.9 présentent les opérateurs d'addition, de multiplication et d'inversion que nous avons implantés pour $\mathbb{F}_{16}$. Un symbole de $\mathbb{F}_{r^2}$ est représenté par $\log_2 r^2$ bits.

Une fonction de R consiste en un vecteur de pl symboles, où pl est le nombre maximum d'itération. Tous les entiers dans le circuit sont représentés par il bits.

Calculs plus fréquents

Parmi les calculs effectués dans l'algorithme, nous pouvons en identifier quelques-uns ayant un plus fort impact sur la performance de l'implantation et pour lesquels nous proposons des architectures particulières :

– **Un produit** $f.\theta_c$, où $f, \theta_c \in R$. Cette opération est systématiquement effectuée dans les calculs de syndrome et dans les mises à jour des fonctions (étapes 1 et 3 de l'algorithme). Cela implique la substitution $x^{r+1} = y^r + y$ et peut être implanté par l'architecture de la figure 5.2, qui utilise un registre à décalage changé pour inclure des éléments de décision et des additionneurs entre les cellules de coefficients. Ce module reçoit une fonction f et des valeurs associées à un entier c (signaux de

contrôle déterminés à partir de c) et fournit une fonction, résultat du produit $f.\theta_c$. Toutes les opérations de ce module sont déterminées par un signal de contrôle D, généré par le module contrôleur du décodeur. Ce signal D est constitué de 3 bits, qui déterminent 5 états ou opérations dans un module $f.\theta_c$:

État $\star$00 Indique que le module doit se charger avec la fonction d'entrée f' ;

État $\star$01 Indique que le module doit effectuer l'addition de composants $f_k + f_{k-r^2+1}$, ce qui correspond à l'opération modulo l'équation de la courbe ;

État 010 Indique l'opération de décalage du registre à gauche ;

État 110 Indique l'opération de décalage du registre à droite ;

État $\star$11 Indique l'état de suspension, où les valeurs du registre sont préservées.

– **Une fonction** $\mu(c) = i$, tel que $ir + j(r+1) = c$. Elle est utilisée dans l'opération $f.\theta_c$ pour déterminer si la valeur du k-ème élément du registre spécial de $f.\theta_c$ doit être additionnée à la valeur f_{k-r^2+1} (*cf.* la figure 5.2). On peut réaliser la fonction $\mu(c)$ par l'opération $\mu(c) = -c \mod (r+1)$. L'opération modulo $(r+1)$ peut être implantée par une rangée au maximum de

$$u^2 - u - \lceil \log (r+1) \rceil^2 + \lceil \log (r+1) \rceil$$

additionneurs complets. Un exemple d'architecture pour opération modulo 5 sur un entier de 7 bits est présenté dans la figure 5.3.

– **Un test d'appartenance à** Λ. Lors de presque toutes les décisions dans l'algorithme, il faut tester si $c \in \Lambda$. Ce test peut être réalisé par recherche sur table, mais aussi au moyen de la fonction $\mu(c)$. Remarquez que $c \in \Lambda$ si et seulement si $\mu(c)r \leq c$. Sur l'espace de fonctions relativement petit du code (64,54) (il n'y a que six lacunes : 1, 2, 3, 6, 7 et 11), nous avons choisi d'employer la méthode de recherche sur table.

Modules opérationnels

La figure 5.4 présente l'architecture générale pour le processeur de décodage employant l'algorithme de O'Sullivan. Elle inclut r blocs MP (processeur principal), r blocs AP (processeurs auxiliaires), un tampon pour les syndromes d'entrée et un processeur de contrôle qui fournit les signaux de contrôle pour les séquences d'opération. Les blocs MP et AP sont interconnectés entre eux via un bus bidirectionnel. Le décodeur reçoit un vecteur de syndromes $\vec{S}$ et fournit un ensemble de r fonctions localisatrices d'erreurs f et r fonctions évaluatrices d'erreurs ϕ.

Une architecture pour l'unité MP est présentée dans la figure 5.5. Chaque unité MP opère sur une fonction localisatrice d'erreurs f et une fonction évaluatrice d'erreurs ϕ. Elle implante le calcul d'un syndrome α à partir d'une fonction f et toutes les opérations de mise à jour des fonctions f et ϕ. Ce module emploie deux opérateurs $f.\theta_c$:

– l'un pour la mise à jour de f et le calcul de α,

– l'autre pour la mise à jour de ϕ.

Elle utilise aussi un registre pour mémoriser α, deux registres pour mémoriser f et ϕ, et deux registres pour mémoriser les fonctions $g\theta_b$ et $\psi\theta_b$ reçues des modules AP (les produits $g\theta_b$ et $\psi\theta_b$ sont réalisés par les modules AP).

Dans les modules MP, un décodeur DEC1 génère un vecteur $0 \ldots 010 \ldots 0$ à partir d'un entier tr, où l'unique bit 1 occupe la tr-ème position du vecteur. Ce vecteur est utilisé pour implanter l'addition d'une fonction pour un monôme θ_c dans la mise à jour des fonctions ϕ. Un module DEC2 opère de façon analogue, fournissant un vecteur $1 \ldots 10 \ldots 0$, avec bits 1 jusqu'à la tr-ème position. Ce vecteur est utilisé pour tronquer une fonction.

Des détails d'une architecture pour l'unité AP sont présentés dans la figure 5.6. Ce module emploie deux unités $f.\theta_c$ pour calculer les produits $g.\theta_b$ et $psi.\theta_b$ utilisés dans la mise à jour des fonctions f et ϕ dans une unité MP. Il emploie aussi des registres pour mémoriser un paramètre β (il garde un syndrome α relatif à la fonction f associée à g), ainsi que $\dot{\beta}$, $\dot{g}$ et $\dot{\psi}$ (valeurs de la dernière iteration).

Module Contrôleur

Le circuit de contrôle du décodeur génère tous les signaux de contrôle pour les modules MP et AP, ainsi que pour le tampon des syndromes d'entrée et pour la sortie du décodeur. Il est aussi responsable pour gérer internement les ensembles σ et δ et, par conséquence, prendre toutes les décisions relatives au fonctionnement du décodeur. C'est lui qui détermine, par exemple, si une fonction doit être mise à jour, pour quel type d'opération et en associant quelles autres fonctions.

Le contrôleur implante aussi le processus de vote par majorité. Dans une itération it où $S(\theta_{it})$ est inconnu, les modules MP calculent normalement les valeurs estimées de α, une fois que $S(\theta_{it}) = 0$ dans le registre des syndromes localisé dans le module tampon des syndromes d'entrée. Le module de vote par majorité interne au contrôleur calcule l'ensemble Γ et les valeurs de σ associées. Donc, il peut choisir parmi les valeurs de α estimés lequel correspond à $S(\theta_{it})$. Cette valeur est copiée dans le registre des syndromes et additionnée aux valeurs de α dans tous les modules MP.

Parmi les signaux générés par le module contrôleur, on distingue les signaux D, qui contrôlent tout le fonctionnement des modules $f.\theta_c$.

Parmi les décisions prises par le contrôleur, on distingue la détermination des anti-lacunes a et b utilisées dans le calcul des produits $f.\theta_a$, $\phi.\theta_a$, $g.\theta_b$ et $\psi.\theta_b$, ainsi que par l'acheminement des valeurs des modules MP vers les modules AP spécifiques, et vice versa. Deux sous-modules du contrôleur sont responsables pour ces calculs et cette prise de décision : les modules NONGAPMP et NONGAPAP (*cf.* figures 5.7 e 5.8).

La complexité

Dans [49], O'Sullivan ne traite pas la question de la complexité de son algorithme. Il affirme que la relation entre la capacité de correction et le numéro d'itérations nécessaires pour décoder le mot reçu n'est pas triviale. En vue d'implanter matériellement ce décodeur, il a fallu déterminer invariablement ces paramètres. Ici, nous considérons la valeur suggérée par Feng et Rao dans [21]. Soit $m = n - k$ le nombre de lignes de la matrice de parité du code et g le genre de la courbe, Feng et Rao ont affirmé que dans $it_{max} = m + g$ itérations, le décodage utilisant le schéma de vote par majorité qu'ils ont proposé corrige jusqu'à la moitié de la distance minimale du code.

On peut vérifier facilement que, dans une itération it quelconque, une opération $f.\theta_c$ fournit une fonction d'ordre de pôle maximal it. En conséquence, l'ordre de pôle maximal d'une fonction quelconque du décodeur dans une itération it est toujours it. Donc, on a besoin d'implanter des modules $f.\theta_c$ et registres pour fonctions de longueur $pl = it_{max} + 1$.

L'implantation

L'architecture présentée a été implantée en VHDL pour un code AG (64, 54) correcteur de 2 erreurs, défini par une courbe d'Hermite $y^4 + y = x^5$ de genre $g = 6$ sur $\mathbb{F}_{16}$, donc de paramètres $r = 4$ et $m = 59$. (Pourtant, la description du décodeur en VHDL a été faite de manière générique pour permettre la synthèse sur n'importe quels paramètres r et m). Cette courbe présente 64 points rationaux plus un point Q sur le plan projectif, raison pour laquelle le code a une longueur $n = 64$.

Le code est défini par un espace de fonctions $L(59Q)$. L'espace orthogonal à ce code, qu'on utilise pour le décodage, est l'espace $L((r^3 + r^2 - r - 2 - m)Q) = L(15Q)$. Ce code présente un ensemble d'anti-lacunes $\{0, 4, 5, 8, 9, 10, 12, 13, 14, 15\}$ de 10 éléments et, alors, une matrice de parité de 10 lignes. Ce sont 10 syndromes à calculer à partir du vecteur reçu.

Ce décodeur a été synthétisé pour un FPGA Altera FLEX 10KE. Dans ce cas, le décodeur peut opérer en 50 MHz, dès que 3 cycles d'horloge sont réservés pour le processus du module MAJVOTING, responsable par le vote par majorité, et 6 cycles d'horloge

sont réservés pour le module UPDDELTASIGMA, responsable par le calcul des nouveaux ensembles δ et σ. Au total, sont dépensés 1503 cycles d'horloge pour décoder un vecteur reçu, compris dans 22 itérations de 68 cycles chacune plus 7 cycles d'horloge pour l'initialisation et finalisation du procès. Avec une fréquence de 50 MHz, cela implique un taux maximal de décodage de 33266 mots code par seconde, ou un taux de 8,5 Mb/s (bits de code), ou 7,2 Mb/s (bits d'information).

Notre implantation VHDL permet de régler le nombre de cycles d'horloge réservés pour les opérations des modules MAJVOTING et UPDDELTASIGMA, ainsi que pour le calcul de α, la mise à jour des fonctions et toutes les autres opérations du décodeur. Ce réglage est fait à travers les constantes cycleestalph, cyclealpha, cyclempap, cyclefcalc, cycleapmp, cycleiterat et period, spécifiées dans le fichier 'HermConstants.vhd'. Les résultats présentés dans le dernier paragraphe sont dus à la technologie utilisée : un FPGA Altera FLEX 10KE. Dans les cas d'une technologie différente, un réglage des constantes sera probablement nécessaire de façon à optimiser la performance du décodeur.

Conclusions

Dans ce document de thèse, nous avons présenté une architecture pour le décodage de codes AG basés sur les courbes d'Hermite.

Le décodeur matériel consiste en unités parallèles chargées de la mise à jour des fonctions localisatrices et évaluatrices d'erreurs, et interconnectées entre elles par un bus bidirectionnel. Nous avons aussi proposé des opérateurs optimisés pour les calculs à plus forte répétition de l'algorithme.

Ce travail constitue une contribution dans le domaine du développement des systèmes de codage de canal par codes AG. Malgré les meilleurs paramètres présentés par ces codes, il y a encore des barrières à leur utilisation dues à la complexité des algorithmes de décodage. À partir de l'architecture décrite, nous pouvons conclure que les décodeurs pour codes d'Hermite présentent une complexité environ r fois plus grande que les décodeurs pour codes RS. Pourtant, les codes RS ont besoin d'employer des corps finis plus grands, donc des unités arithmétiques plus complexes, pour corriger des vecteurs de même longueur. De cette façon, nous estimons que la complexité générale pour des longueurs égales (en nombre de bits) est équivalente pour les deux codes RS et d'Hermite.

Capítulo 1

Introdução

Em um sistema de comunicação digital um dos mais importantes componentes é a chamada *codificação de canal* [56]. É ela a responsável, através da inserção de redundâncias controladas ao sinal, pela detecção e correção de erros que porventura ocorram durante a comunicação, erros esses resultantes da ação de elementos interferentes no sistema, como o ruído, o desvanecimento, etc. Portanto, a codificação de canal, também chamada de *codificação para controle de erros* [6,72], visa fundamentalmente diminuir os efeitos destes fenômenos e aumentar a confiabilidade do sistema, tornando-o assim mais robusto.

Na codificação para controle de erros, uma das classes de códigos de bloco mais amplamente utilizada tem sido a dos conhecidos *códigos de Reed-Solomon* (RS). Os códigos RS utilizam como blocos de código os valores de todos os polinômios definidos sobre um corpo finito até um determinado grau. Eles têm tido aplicação em diversas áreas, como no desenvolvimento de padrões para comunicação móvel celular, no armazenamento em CDs, na comunicação por satélite etc. [28, 73]. Contudo, eles apresentam uma limitação importante: seu comprimento de bloco não pode ser maior que a ordem do corpo finito utilizado. Uma conseqüência direta da teoria de Shannon[1] é que bons códigos devem ter grandes comprimentos [41].

Apesar da larga utilização dos códigos RS, há atualmente outras opções na teoria de codificação de canal que apresentam melhores características assintóticas, como é o caso dos códigos algébrico-geométricos baseados na teoria matemática conhecida como *geometria algébrica*.

A geometria algébrica é uma ferramenta matemática que tem se mostrado útil no desenvolvimento de soluções em diversas áreas da engenharia, como na codificação para controle de erros, na criptografia, na robótica, no projeto de CADs, entre outras [9,10,68].

[1] Claude Shannon apresentou em 1948 um dos fundamentos da teoria da informação, resultado importante para a evolução das comunicações, demonstrando que a todo canal de comunicação está associada uma capacidade de transmissão.

Ela estabelece uma série de relações entre estruturas algébricas, como espaços e ideais de funções, e estruturas geométricas, como variedades. A presente tese trata de uma destas aplicações, a dos chamados códigos algébrico-geométricos, seus esquemas de decodificação e sua implementação em hardware.

1.1 Códigos algébrico-geométricos

Na década de 1970, o matemático russo V. D. Goppa propôs que, ao invés de avaliar polinômios em pontos simples (valores do corpo finito) como nos códigos de Reed-Solomon, poder-se-ia avaliar funções algébricas em pontos de curvas definidas sobre corpos finitos [25]. Além disso, ele mostrou como substituir a condição sobre os graus dos polinômios por condições sobre funções algébricas. Em 1982, M. A. Tsfasman, S. G. Vlăduţ e T. Zink combinaram a idéia da construção de códigos a partir de curvas algébricas sobre corpos finitos aos recentes resultados da geometria algébrica, produzindo um desenvolvimento importante na teoria de códigos para correção de erros, os chamados *códigos algébrico-geométricos* (AG) ou códigos de Goppa geométricos [7, 11, 19, 40, 44, 71]. Os códigos RS são um caso particular dos códigos AG quando a curva algébrica adotada é apenas uma reta. As curvas sobre corpos finitos podem ter muito mais pontos que uma simples reta, de forma que os códigos AG podem apresentar comprimento muito maior que os códigos RS.

Do ponto de vista da relação entre a taxa de informação assintótica, que representa o comprometimento da taxa de transmissão de informação devido à utilização do código (introdução de redundâncias), e a distância mínima relativa, que representa a capacidade de correção de erros do código, pode-se afirmar que os códigos AG fazem parte da classe dos chamados *bons códigos*, que são aqueles que não comprometem a taxa de transmissão em detrimento da capacidade de correção, ou vice versa, para um comprimento de código $n \to \infty$. Os códigos AG são, portanto, excelentes códigos, que apresentam parâmetros como comprimento, capacidade de correção e taxa de informação melhores que códigos já estabelecidos, como os de Reed-Solomon [18, 40].

1.2 Decodificação de códigos AG

O primeiro esquema de decodificação para os códigos AG foi proposto em 1989 por Jørn Justesen et al., e consistia numa generalização do algoritmo PGZ para decodificação de códigos BCH, fornecendo um polinômio localizador de erros em duas variáveis, que possuía dentre seus zeros as posições dos erros ocorridos [32, 41]. Este algoritmo, que decodifica

apenas códigos AG definidos sobre curvas planas e tem baixa capacidade de correção de erros (corrige $g/2$ menos erros que o permitido pela capacidade de correção do código, sendo g o gênero da curva algébrica que define o código) e alta complexidade algorítmica ($O(n^3)$, sendo n o comprimento do código), foi generalizado para curvas algébricas arbitrárias em 1990 por Skorobogatov e Vlăduţ, tendo ficado conhecido como *algoritmo básico* [67]. O algoritmo básico é hoje a base para uma grande quantidade de esquemas de decodificação de códigos AG encontrados na literatura. Nele, o passo fundamental consiste na determinação de funções localizadoras de erros, ou seja, funções que se anulem nos pontos da curva algébrica associados aos erros ocorridos no vetor recebido.

Uma segunda classe de algoritmos de decodificação para códigos AG foi proposta inicialmente em 1992 por S. C. Porter e constitui uma generalização do algoritmo de Euclides para solução da equação chave [54].

Outra abordagem de decodificação dos códigos AG é a chamada *decodificação de lista*, que consiste na busca pelo conjunto de palavras código que se encontram até uma distância de Hamming especificada, em geral maior que a metade da distância mínima do código (tradicionalmente, a decodificação consiste na busca pela única palavra código, caso exista, situada dentro da esfera de decodificação de diâmetro igual à distância mínima do código). Esta idéia alternativa de decodificação foi proposta na década de 1950 por P. Elias [20] e J. M. Wozencraft [74], tendo sido aplicada à decodificação de códigos AG inicialmente em 1999 por M. A. Shokrollahi e H. Wasserman [66]. Recentemente, alguns trabalhos têm sido publicados nesta linha [26, 76].

Merece menção ainda uma abordagem de decodificação de códigos AG que envolve a utilização de esquemas de decisão suave, em que o demodulador fornece informações extras sobre a mensagem recebida ao decodificador, que as utiliza para melhorar seu desempenho na decodificação [5, 34, 36, 52].

As principais metas na otimização dos esquemas de decodificação para códigos AG têm sido desenvolver algoritmos que permitam explorar toda a capacidade de correção de erros do código e reduzir a complexidade dos decodificadores, tornando-os tão eficientes quanto os decodificadores de outros códigos já estabelecidos, como os de Reed-Solomon (a complexidade dos primeiros algoritmos, como o algoritmo básico e o algoritmo de Porter, $O(n^3)$, tem sido proibitiva para aplicações práticas).

Com relação ao problema de decodificar até a máxima capacidade de correção do código, em 1994 G.-L. Feng e T. R. N. Rao propuseram uma elegante solução denominada *algoritmo de decisão por maioria*, em que as síndromes desconhecidas, por demanda, são determinadas a partir das síndromes já conhecidas a cada iteração do algoritmo de decodificação [21]. O esquema de Feng e Rao permitiu uma volta de olhares para o problema da complexidade dos algoritmos de decodificação dos códigos AG. Em 1990, S.

Sakata propôs uma generalização em várias variáveis do algoritmo de Berlekamp-Massey, que passou a ser conhecido com *algoritmo BMS* [60]. O algoritmo BMS, juntamente com o esquema de decisão por maioria de Feng e Rao, tem sido a base para a construção de diversos algoritmos de decodificação rápida para códigos AG [30, 33, 34, 58, 59, 61, 63, 64].

1.3 Arquiteturas para decodificadores de códigos AG

Apesar dos códigos AG apresentarem comprovadamente melhores parâmetros e extrapolarem as limitações dos códigos RS [18], sua utilização prática ainda enfrenta barreiras, como a da complexidade de seus algoritmos de decodificação. De modo a contribuir no sentido de tornar os códigos AG competitivos mediante outros códigos comerciais, como os códigos RS, deve-se desenvolver esquemas de decodificação mais eficientes e arquiteturas que permitam a implementação otimizada destes esquemas em hardware.

No que concerne ao desenvolvimento de arquiteturas de implementação em hardware de decodificadores para códigos AG, pouco foi até então apresentado. Em 1998, Ralf Kötter propôs um algoritmo de decodificação para códigos AG de estrutura regular e simples voltado à implementação em VLSI [35]. A idéia central na proposta de Kötter era utilizar uma configuração em paralelo de vários algoritmos BMS modificados, de modo a obter um esquema que apresentasse a mesma complexidade de tempo que um decodificador de Berlekamp-Massey para códigos de Reed-Solomon. Além deste trabalho, são poucas as publicações encontradas abordando esta questão da implementação em hardware de decodificadores para códigos AG [2, 37, 42, 50, 53, 62].

O propósito geral do presente trabalho repousa na integração do estudo dos sistemas de codificação / decodificação de códigos AG com o desenvolvimento de arquiteturas para implementação em hardware no sentido de proporcionar uma contribuição científica nesta área. Neste sentido, uma nova arquitetura para um decodificador de códigos AG baseado na solução de uma equação chave foi desenvolvido [16, 17]. A arquitetura proposta foi descrita em VHDL, simulada e validada. Este trabalho incluiu ainda a descrição, também em VHDL (exceto o módulo inversor, descrito em Verilog), de unidades aritméticas em corpo finito responsáveis pelas operações aritméticas no decodificador.

1.4 Metodologia

O presente trabalho de tese é fruto de um doutorado em co-tutela realizado na Universidade Federal de Campina Grande – UFCG, em Campina Grande-PB, no Brasil, e na École Nationale Supérieure des Télécommunications – ENST, em Paris, na França, no

período de setembro de 2000 a agosto de 2004. Este doutorado fez parte de um acordo de cooperação entre as duas instituições (acordo internacional CAPES / COFECUB) e recebeu suporte financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES (http://www.capes.gov.br/), organismo governamental brasileiro. Neste contexto, ele se dividiu em três fases. A primeira, realizada na UFCG durante 2 anos, teve sua ênfase no estudo das diferentes abordagens de decodificação dos códigos AG. A segunda fase, realizada na ENST durante um período de 1 ano e meio, teve sua ênfase no estudo das arquiteturas de implementação em hardware para os algoritmos de decodificação de códigos AG. Ambas as fases tiveram início com um levantamento bibliográfico sobre a respectiva ênfase. A terceira e última fase foi realizada durante os últimos 6 meses de tese na UFCG e consistiu basicamente na elaboração do texto final da tese.

O trabalho pode ser ainda separado em dois diferentes planos. Num primeiro plano, o trabalho consistiu no estudo de algoritmos eficientes para decodificação de códigos AG (plano algorítmico) e foi desenvolvido através da simulação em computador destes decodificadores utilizando a ferramenta Macaulay 2 (*cf.* http://www.math.uiuc.edu/Macaulay2) e a ferramenta Matlab (*cf.* http://www.mathworks.com/). Num segundo plano, o trabalho consistiu na análise dos algoritmos voltada à implementação em hardware de decodificadores para os códigos AG (plano arquitetural), e foi desenvolvido através da simulação em computador utilizando as ferramenta ModelSim da Mentor Graphics (*cf.* http://www.model.com/), LeonardoSpectrum da Mentor Graphics (*cf.* http://www.mentor.com/leonardospectrum/) e Quartus II da Altera (*cf.* http://www.altera.com/support/software/sof-quartus.html).

1.5 Estrutura do texto

Este texto está organizado em 6 capítulos e 4 apêndices, cujos conteúdos são descritos a seguir.

Este capítulo introdutório tem por objetivo principal contextualizar o problema da codificação / decodificação de códigos AG e da implementação desses algoritmos em hardware, além de descrever a metodologia do trabalho de doutorado realizado.

No capítulo 2, são definidos os códigos AG, em especial os criados a partir das curvas de Hermite, que também são apresentadas neste capítulo. A discussão sobre os códigos AG, e posteriormente sobre seus esquemas de decodificação, exige conhecimentos prévios sobre álgebra de corpos finitos, sobre teoria da codificação para correção de erros e sobre geometria algébrica. Como referência, para possibilitar o mais fácil acesso ao assunto objeto deste trabalho, são apresentadas nos apêndices A, B e C descrições resumidas

sobre cada uma destas três teorias.

O capítulo 3 trata do problema da decodificação dos códigos AG. Nele, são apresentadas as abordagens de decodificação mais importantes destes códigos.

O capítulo 4 descreve as arquiteturas dos operadores aritméticos em corpo finito que foram implementadas de modo a viabilizar a implementação do decodificador. No apêndice D, é apresentada uma lista de polinômios primitivos de peso mínimo em $\mathbb{F}_2$. Estes polinômios são parâmetro de projeto para as unidades aritméticas implementadas, sendo necessários para a geração de corpos finitos de característica 2.

O capítulo 5 descreve o algoritmo de decodificação de O'Sullivan, a nova arquitetura proposta para ele e os detalhes de sua implementação.

O capítulo 6 apresenta as conclusões e perspectivas deste trabalho.

Capítulo 2

Códigos Algébrico-Geométricos

O capítulo anterior contextualizou o problema da codificação / decodificação de códigos AG e da implementação desses algoritmos em hardware. O presente capítulo tem por objetivo definir estes códigos, e, em particular, a classe dos códigos AG construídos sobre as curvas de Hermite, também apresentadas neste capítulo. Por fim, é discutida a questão dos limites inferiores para bons códigos obtido pelos códigos AG, melhores que o limite de Gilbert-Varshamov.

Sobre estes conceitos apresentados neste capítulo, há diversas referências nas quais podem ser obtidos maiores detalhes [8,32,41,44,46,55,69,70]. Para facilitar o acesso a esta teoria, uma descrição resumida sobre algebra dos corpos finitos, codificação para correção de erros e geometria algébrica é apresentada nos apêndices A, B e C, respectivamente.

2.1 Códigos algébrico-geométricos

São dois os diferentes tipos de construção de códigos AG [12, 14]:

1. No primeiro tipo, chamado aqui de *construção por funções*, o código consiste na avaliação de funções racionais em um conjunto de pontos distintos de uma curva;

2. No segundo tipo, chamado aqui de *construção por diferenciais*, o código consiste no resíduo de diferenciais em um conjunto de pontos distintos de uma curva.

2.1.1 Construção por funções

Considere $\mathcal{X}$ uma curva projetiva não singular de gênero g, $P_1, \ldots, P_n$ pontos racionais de $\mathcal{X}$ e $D = P_1 + \cdots + P_n$ um divisor desta curva. Considere G um divisor de $\mathcal{X}$ cujo suporte seja disjunto de D, e suponha que

$$2g - 2 < \deg(G) < n.$$

Definição 3 (Código por funções) *Define-se o código AG denotado por $C(D,G)$, sobre o corpo finito $\mathbb{F}_q$, como o mapeamento linear $\alpha : L(G) \to \mathbb{F}_q^n$ dado por*

$$\alpha(f) = [f(P_1), \ldots, f(P_n)], \tag{2.1}$$

em que

$$L(G) = \{f \in \mathbb{F}_q(\mathcal{X}) : (f) + G \succ 0\} \cup \{0\}$$

é o espaço de funções gerado pelo divisor G, $\mathbb{F}_q(\mathcal{X})$ é o corpo de funções da curva $\mathcal{X}$,

$$(f) = \sum_{P_i} v_{P_i}(f) P_i$$

é o divisor principal da função f e $v_{P_i}(f)$ é a ordem da função f no ponto $P_i \in \mathcal{X}$.

Observe que o núcleo do mapeamento de (2.1) é o conjunto de funções

$$\{f \in L(G) : v_{P_i}(f) > 0,\ i = 1, \ldots, n\},$$

que constitui, na verdade, o espaço de funções $L(G-D)$. Então, a *dimensão* k do código $C(D,G)$ é dada por

$$k = \dim[L(G)] - \dim[L(G-D)].$$

Observe que, sendo $\deg(G) < n$ por hipótese, então $\deg[(G-D)] < 0$. Com isso, para qualquer função $f \in F_q(\mathcal{X})$, tem-se que $\deg[(f)+G-D] < 0$. Isto implica que $\dim[L(G-D)] = 0$. Considerando também o teorema de Riemann-Roch (*cf.* (C.8)), tem-se que

$$k = \deg(G) - g + 1. \tag{2.2}$$

A *distância mínima* d do código $C(D,G)$ definido em (2.1) satisfaz a desigualdade

$$d \geq n - \deg(G),$$

uma vez que qualquer função $f \in L(G)$ possui no máximo $\deg(G)$ zeros, ou seja, o peso de qualquer palavra código $\alpha(f)$ não é menor que $n - \deg(G)$. Do limite de Singleton[1], tem-se que

$$\begin{aligned} d &\leq n - k + 1 \\ &= n - \deg(G) + g \Rightarrow \\ n - \deg(G) \leq d &\leq n - \deg(G) + g. \end{aligned} \tag{2.3}$$

[1]O limite de Singleton afirma que, para um código linear (n,k,d), de comprimento n, dimensão k e distância mínima d, tem-se que

$$d \leq n - k + 1.$$

2.1.2 Construção por diferenciais

Considere $\mathcal{X}$, $P_1, \ldots, P_n$, D e G da mesma forma da seção anterior (os pontos racionais $P_1, \ldots, P_n$ equivalem a lugares de grau 1 da curva $\mathcal{X}$).

Definição 4 (Código por diferenciais) *Define-se o código AG construído por diferenciais, denotado por $C^*(D,G)$, sobre um corpo finito $\mathbb{F}_q$ (corpo algebricamente fechado), como o mapeamento linear $\alpha^* : \Omega(G-D) \to \mathbb{F}_q^n$ dado por*

$$\alpha^*(\omega) = [\mathrm{Res}_{P_1}(\omega), \ldots, \mathrm{Res}_{P_n}(\omega)], \tag{2.4}$$

em que

$$\Omega(G-D) = \{\omega \in \Omega_{\mathcal{X}} : (\omega) \succ G-D\} \cup \{0\}$$

é o espaço de diferenciais gerado pelo divisor $G-D$, $\Omega_{\mathcal{X}}$ é o conjunto de diferenciais associados à curva $\mathcal{X}$, $\mathrm{Res}_{P_i}(\omega)$ é o resíduo de ω no ponto P_i,

$$(\omega) = \sum_{P_i} v_{P_i}(f)\, P_i$$

é o divisor do diferencial $\omega = f\,dt$ e $v_{P_i}(f) = v_{P_i}(\omega)$ é sua ordem no ponto $P_i \in \mathcal{X}$.

O núcleo do mapeamento α^* definido em (2.4) é o espaço de diferenciais $\Omega(G)$. Com isso, a *dimensão* k^* do código $C^*(D,G)$ é dada por

$$k^* = \dim[\Omega(G-D)] - \dim[\Omega(G)] = i(G-D) - i(G).$$

Considerando a hipótese inicial $\deg(G) > 2g-2$, tem-se que $i(G) = 0$. Considerando o teorema de Riemann-Roch (*cf.* (C.8)), tem-se que a dimensão do código $C^*(D,G)$ é dada por[2]

$$k^* = n - \deg(G) + g - 1. \tag{2.5}$$

A *distância mínima* d^* do código $C^*(D,G)$ definido em (2.4) satisfaz a desigualdade[3]

$$d^* \geq \deg(G) - 2g + 2.$$

Do limite de Singleton, tem-se que

$$\begin{aligned} d^* &\leq n - k^* + 1 \\ &= \deg(G) - g + 2 \Rightarrow \\ \deg(G) - 2g + 2 \leq d^* &\leq \deg(G) - g + 2. \end{aligned} \tag{2.6}$$

[2]Uma prova detalhada deste resultado pode ser obtida em Stichtenoth [70, pp. 45–46].

[3]A prova deste resultado pode ser vista em Stichtenoth [70, pp. 45–46].

De (2.2) e (2.5), observe que $k + k^* = n$. Considere agora $f \in L(G)$ e $\omega \in \Omega(G - D)$. Das definições dos códigos $C(D, G)$ e $C^*(D, G)$, observa-se que o diferencial $f\omega$ não possui pólos, exceto possivelmente nos pontos $P_1, \ldots, P_n$. O resíduo de $f\omega$ no ponto P_i é igual a $f(P_i)\,\mathrm{Res}_{P_i}(\omega)$. Do teorema de resíduos (*cf.* (C.6)), tem-se que

$$\sum_{i=1}^{n} f(P_i)\,\mathrm{Res}_{P_i}(\omega) = 0,$$

que é o produto interno de duas palavras código $\alpha(f)$ e $\alpha^*(\omega)$. Conclui-se, portanto, que $C(D, G)$ e $C^*(D, G)$ são códigos duais.

Além disso, mostra-se que existe um diferencial ω com pólos simples e resíduo 1 nos pontos $P_1, \ldots, P_n$ [70, p. 48] (a determinação de diferenciais requer subsídios teóricos que divergem dos objetivos deste trabalho), tal que

$$C^*(D, G) = C(D, (\omega) + D - G), \tag{2.7}$$

em que (ω) é o divisor de ω. Isto implica que a construção de resíduos fornece a mesma classe de códigos da construção de funções.

Usualmente, os trabalhos envolvendo a decodificação dos códigos AG associam os códigos utilizados à construção por diferenciais, de modo que o espaço do código dual, sobre o qual se trabalha na decodificação, pode ser construído de forma mais simples, por funções.

2.2 Curvas de Hermite

A classe de curvas algébricas, denominadas *curvas de Hermite*, tem sido amplamente utilizada, principalmente na construção de códigos para proteção contra erros. Isto se deve aos bons parâmetros de comprimento e distância mínima relativa obtidos pelos códigos construídos a partir destas curvas, como também à simplicidade da estrutura do espaço de funções definido por esta classe de curvas (isto se traduz em simplicidade nas regra de construção de uma base para este espaço).

Considere o corpo finito $\mathbb{F}_q$ de $q = r^2$ elementos, em que q é uma potência de algum inteiro primo. Considere o corpo de funções $\mathbb{F}_q(\mathcal{X})$ da *curva de Hermite* $\mathcal{X}$ dada pela equação afim

$$\mathcal{X} : u^{r+1} + v^{r+1} + 1 = 0. \tag{2.8}$$

Considere $a, b \in \mathbb{F}_q$, tais que $a^r + a = b^{r+1} = -1$. Considere agora $x, y \in \mathbb{F}_q(\mathcal{X})$, em que

$$x = \frac{b}{v - bu} \quad \text{e} \quad y = ux - a = \frac{b(1 + a)u - av}{v - bu}.$$

Pode-se verificar que $(v-bu)^{r+1}(y^r+y-x^{r+1})=0$. Portanto, a curva de Hermite $\mathcal{X}$ (*cf.* (2.8)) pode ser escrita numa forma mais conveniente dada pela equação

$$\mathcal{X}: y^r+y=x^{r+1}. \tag{2.9}$$

A versão projetiva no espaço projetivo $\mathbb{P}^2$, definido sobre $\mathbb{F}_q$, da curva de Hermite é dada pela equação (forma homogênea de (2.8))

$$\mathcal{X}: u^{r+1}+v^{r+1}+z^{r+1}=0. \tag{2.10}$$

Com relação ao número de pontos racionais desta curva, considere inicialmente o caso em que uma das coordenadas em (2.10) é nula, por exemplo z. Sem perda de generalidade, pode-se fazer $v=1$, obtendo-se $u^{r+1}+1=0$, que apresenta $r+1$ soluções em $\mathbb{F}_q$. Conclui-se, então, que a curva $\mathcal{X}$ apresenta $3\,(r+1)$ pontos racionais quando $uvz=0$. Para o caso em que $uvz\neq 0$, considere $z=1$ e v igual a qualquer elemento não nulo de $\mathbb{F}_q$, tal que $v^{r+1}\neq 1$. Para cada valor diferente de v, existem $r+1$ soluções para u em (2.10). Neste caso há, portanto, $(r-2)\,(r+1)^2$ pontos racionais em $\mathcal{X}$. Totalizando, tem-se que a curva de Hermite $\mathcal{X}$ apresenta sempre

$$3\,(r+1)+(r-2)\,(r+1)^2=1+r^3$$

pontos racionais, que são:

1. O ponto no infinito $Q=(a,b,0)$, com $a,b\in\mathbb{F}_q$, em que $a^{r+1}+b^{r+1}=0$ (*cf.* (2.10));
2. Os r^3 pontos afins (a,b), com $a,b\in\mathbb{F}_q$, em que $b^r+b=a^{r+1}$ (*cf.* (2.9)).

A curva de Hermite $\mathcal{X}$ é não singular, portanto irredutível. Sendo o grau de (2.9) igual a $r+1$, tem-se que o gênero g (*cf.* (C.7)) de $\mathcal{X}$ é dado por

$$g=\frac{q-r}{2}. \tag{2.11}$$

Considere agora um divisor $G=mQ$, para algum parâmetro inteiro $m\geq 0$. Como a variável x apresenta um pólo de ordem r e y um pólo de ordem $r+1$ no ponto Q, pode-se verificar [69] que o conjunto

$$\mathcal{B}_m=\left\{x^iy^j: 0\leq i,\ 0\leq j\leq r-1,\ ir+j\,(r+1)\leq m\right\} \tag{2.12}$$

constitui uma base para o espaço vetorial de funções $L\,(G)$ gerado pelo divisor $G=mQ$, característico da curva de Hermite $\mathcal{X}$. Diz-se, então, que este é o espaço de funções gerado pela curva $\mathcal{X}$ para um parâmetro m.

Defina agora o conjunto de números inteiro

$$\Lambda=\left\{ir+j\,(r+1): 0\leq i\leq r,\ j\geq 0\right\}. \tag{2.13}$$

Pode-se verificar que Λ constitue um conjunto de anti-lacunas de Q. Ele representa as ordens de pólo das funções que pertencem aos espaço de funções $L(G)$.

2.3 Códigos de Hermite

Os códigos AG baseados em curvas de Hermite (para detalhes sobre as curvas de Hermite, veja seção 2.2) constituem hoje uma das mais exploradas classes de códigos AG, em vista dos excelentes parâmetros que apresenta e da simplicidade de sua estrutura. Esta seção descreve esta classe de códigos, suas propriedades e analisa o caso em que o comprimento do código é r^3 (comprimento máximo).

Considere os dois divisores $G = mQ$ e $D = P_1 + \cdots + P_n$, em que $P_1, \ldots, P_n$ são os pontos racionais da curva $\mathcal{X}$ (*cf.* (2.9)) e $n \leq r^3$.

Definição 5 (Código de Hermite) *O* código de Hermite $C(D, G)$, *denotado por* C_m, *é dado por*

$$C_m = \{[f(P_1), \ldots, f(P_n)] : f \in L(G)\}. \tag{2.14}$$

O código definido em (2.14) é obtido considerando a base $\mathcal{B}_m$ fornecida por (2.12) apresentada na seção 2.2 para o espaço de funções $L(G)$.

Considere a função $z \in \mathbb{F}_q[x, y]$ dada por

$$z = x^q - x,$$

cujo divisor principal (z) é dado por

$$(z) = D - r^3 Q.$$

Considere o diferencial $\omega = \frac{dz}{z}$, cujo resíduo é igual a 1 em todos os pontos de $D = P_1 + \cdots + P_n$ (como P_i é um zero simples de z, então $\frac{dz}{z}$ possui um pólo simples em P_i com resíduo 1). Tem-se que [69]

$$\begin{aligned}(\omega) &= (dz) - (z) \\ &= (-dx) - (z) \\ &= (r(r-1) - 2)Q - D + r^3 Q \\ &= \left(r^3 + r^2 - r - 2\right)Q - D.\end{aligned}$$

Utilizando-se (2.7) e considerando um divisor $G^* = (\omega) + D - G$, em que $G = mQ$, obtém-se que

$$\begin{aligned}G^* &= \left(r^3 + r^2 - r - 2\right)Q - D + D - mQ \\ &= \left(r^3 + r^2 - r - 2 - m\right)Q \Rightarrow \\ C^*(D, mQ) &= C\left(D, \left(r^3 + r^2 - r - 2 - m\right)Q\right).\end{aligned}$$

Portanto, conclui-se que os códigos de Hermite C_m e $C_{r^3+r^2-r-2-m}$ são duais entre si, uma vez que equivalem a $C(D, mQ)$ e $C^*(D, mQ)$, respectivamente. Particularmente, se r é um número par e $m = \frac{1}{2}(r^3 + r^2 - r - 2)$, então o código C_m é auto-dual.

Caso do comprimento máximo

Considerando o comprimento máximo do código de Hermite C_m, em que $n = r^3$, serão descritas agora a dimensão deste código, sua matriz geradora, além de outras propriedades [69].

É fácil verificar que a dimensão k do código C_m é $k = 0$, para $m < 0$, e $k = n = r^3$, para $m > r^3 + r^2 - r - 2$. Considerando o parâmetro m no intervalo

$$0 \leq m \leq r^3 + r^2 - r - 2,$$

tem-se que a *dimensão* k do código C_m é dada por [69]

$$k = \begin{cases} |\mathcal{B}_m|, & m \leq r^2 - r - 2, \\ m + 1 - \frac{(r^2-r)}{2}, & r^2 - r - 2 < m < r^3, \\ r^3 - |\mathcal{B}_{r^3+r^2-r-2-m}|, & m \geq r^3, \end{cases} \tag{2.15}$$

sendo $|\mathcal{B}_m|$ a cardinalidade da base $\mathcal{B}_m$ (*cf.* (2.12)) para o espaço $L(mQ)$.

A *matriz geradora* M_m de um código de Hermite C_m, para $0 \leq m < r^3$, é uma matriz $|\mathcal{B}_m| \times r^3$ dada por

$$M_m = \left[a^i b^j\right]_{|\mathcal{B}_m| \times r^3}, \tag{2.16}$$

em que:

- $a, b \in \mathbb{F}_q$, $b^r + b = a^{r+1}$;
- $i \geq 0$, $0 \leq j \leq r - 1$;
- e $ir + j(r + 1) \leq m$.

Tomando o parâmetro m no intervalo

$$r^2 - r - 2 < m < r^3 + r^2 - r - 2,$$

tem-se que a matriz $M_{r^3+r^2-r-2-m}$ consiste também na matriz de paridade do código C_m.

Considerando uma equação $ir + j(r + 1) = k$, na qual $0 \leq i \leq r$ e $j \geq 0$, tem-se que

$$\begin{gathered} (i + j)(r + 1) - i = k \Rightarrow \\ i = (-k) \mod (r + 1) \quad \text{e} \quad j = \frac{k - ir}{r + 1}. \end{gathered} \tag{2.17}$$

Estas equações são úteis em implementações para a determinação das matrizes geradoras e de paridade, assim como num procedimento de decodificação.

Considere agora $m = ir + j(r + 1) < r^3$, com $i \geq 0$ e $0 \leq j \leq r - 1$. Se $j = 0$ (ou seja, $m \equiv 0 \mod r$) ou $m \leq r^3 - r^2$, então a *distância mínima* d do código C_m é dada por

$$d = r^3 - m. \tag{2.18}$$

Em suma, as informações contidas nesta seção permitem construir computacionalmente um código de Hermite C_m definido por (2.14) (*cf.* algoritmo 6), de parâmetro m, comprimento $n = r^3$ e dimensão e distância mínima dados por (2.15) e (2.18), respectivamente. Materialmente, o codificador consiste num operador de multiplicação de um vetor de informação pela matriz M_m fornecida pelo algoritmo 6.

Algoritmo 6 (Construção do código de Hermite C_m)

Entradas:

- *Corpo $\mathbb{F}_q$, em que $q = r^2$ é uma potência de algum inteiro primo;*
- *Curva $\mathcal{X} : y^r + y = x^{r+1}$;*
- *Parâmetro $m \leq r^3 + r^2 - r - 2$.*

Saídas:

- *Divisor D dos pontos da curva $\mathcal{X}$;*
- *Gênero g de $\mathcal{X}$;*
- *Base $\mathcal{B}_m$ para o espaço $L(mQ)$;*
- *Parâmetros n, k, d e matriz M_m do código C_m.*

Inicialização:

- $a, b \in \mathbb{F}_q$;
- $g = \frac{q-r}{2}$;
- $n = r^3$;
- $d = r^3 - m$.
- $D = \varnothing$;
- $\mathcal{B}_m = \varnothing$.

<<< **Passo 1 : Determinação do divisor** D >>>
FOR $a = 0$ ***TO*** $q - 1$
  FOR $b = 0$ ***TO*** $q - 1$
    IF $b^r + b == a^{r+1}$ ***THEN*** $D \leftarrow D + (a, b)$

<<< **Passo 2 : Determinação da base** B_m **e da dimensão** k >>>
FOR $k = 0$ ***TO*** m
  $i = (-k) \mod (r + 1)$
  $j = \frac{k - ir}{r+1}$
  $\mathcal{B}_m \leftarrow x^i y^j$

<<< **Passo 3 : Determinação da matriz geradora** M_m >>>
FOR $i = 0$ ***TO*** k
  FOR $j = 0$ ***TO*** n
    $(a, b) = D[j]$ *e* $f(x, y) = \mathcal{B}_m[i]$
    $M_m[i][j] = f(a, b)$

Prova. Para detalhes sobre a prova deste algoritmo, veja [69]. ■

2.3.1 Exemplo de código de Hermite

Observe um exemplo simples de um código de Hermite C_{18}, de comprimento $n = 27$, em $\mathbb{F}_9$.

O corpo $\mathbb{F}_9$ consiste no conjunto $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, 0\}$ associado às operações "+" e "×" descritas na tabela 2.1.

+	0	1	α	α^2	α^3	α^4	α^5	α^6	α^7
0	0	1	α	α^2	α^3	α^4	α^5	α^6	α^7
1	1	α^4	α^7	α^3	α^5	0	α^2	α	α^6
α	α	α^7	α^5	1	α^4	α^6	0	α^3	α^2
α^2	α^2	α^3	1	α^6	α	α^5	α^7	0	α^4
α^3	α^3	α^5	α^4	α	α^7	α^2	α^6	1	0
α^4	α^4	0	α^6	α^5	α^2	1	α^3	α^7	α
α^5	α^5	α^2	0	α^7	α^6	α^3	α	α^4	1
α^6	α^6	α	α^3	0	1	α^7	α^4	α^2	α^5
α^7	α^7	α^6	α^2	α^4	0	α	1	α^5	α^3

×	0	1	α	α^2	α^3	α^4	α^5	α^6	α^7
0	0	0	0	0	0	0	0	0	0
1	0	1	α	α^2	α^3	α^4	α^5	α^6	α^7
α	0	α	α^2	α^3	α^4	α^5	α^6	α^7	1
α^2	0	α^2	α^3	α^4	α^5	α^6	α^7	1	α
α^3	0	α^3	α^4	α^5	α^6	α^7	1	α	α^2
α^4	0	α^4	α^5	α^6	α^7	1	α	α^2	α^3
α^5	0	α^5	α^6	α^7	1	α	α^2	α^3	α^4
α^6	0	α^6	α^7	1	α	α^2	α^3	α^4	α^5
α^7	0	α^7	1	α	α^2	α^3	α^4	α^5	α^6

Tabela 2.1: Operações "+" e "×" relacionadas ao conjunto $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, 0\}$, que constituem o corpo $\mathbb{F}_9$.

Sendo $q = 9 \Rightarrow r = 3$, tem-se que a curva de Hermite correspondente (veja equação 2.9) é dada por

$$\mathcal{X} : y^3 + y = x^4.$$

O gênero desta curva é $g = 3$ (equação 2.11). Os pontos racionais de $\mathcal{X}$ são o ponto Q no infinito e os $r^3 = 27$ pontos descritos na tabela 2.2.

Nº	Ponto	Nº	Ponto	Nº	Ponto
1	$(1, \alpha^4)$	10	$(\alpha^3, 1)$	19	(α^6, α^4)
2	$(1, \alpha^5)$	11	(α^3, α)	20	(α^6, α^5)
3	$(1, \alpha^7)$	12	(α^3, α^3)	21	(α^6, α^7)
4	$(\alpha, 1)$	13	(α^4, α^4)	22	$(\alpha^7, 1)$
5	(α, α)	14	(α^4, α^5)	23	(α^7, α)
6	(α, α^3)	15	(α^4, α^7)	24	(α^7, α^3)
7	(α^2, α^4)	16	$(\alpha^5, 1)$	25	$(0, \alpha^2)$
8	(α^2, α^5)	17	(α^5, α)	26	$(0, \alpha^6)$
9	(α^2, α^7)	18	(α^5, α^3)	27	$(0, 0)$

Tabela 2.2: Pontos racionais da curva de Hermite $y^3 + y = x^4$ em $\mathbb{F}_9$.

O divisor D, cujo suporte são os 27 pontos racionais da curva $\mathcal{X}$ (tabela 2.2), é

$$\begin{aligned} D = &\left(1, \alpha^4\right) + \left(1, \alpha^5\right) + \left(1, \alpha^7\right) + (\alpha, 1) + (\alpha, \alpha) + \left(\alpha, \alpha^3\right) + \left(\alpha^2, \alpha^4\right) + \\ &\left(\alpha^2, \alpha^5\right) + \left(\alpha^2, \alpha^7\right) + \left(\alpha^3, 1\right) + \left(\alpha^3, \alpha\right) + \left(\alpha^3, \alpha^3\right) + \left(\alpha^4, \alpha^4\right) + \left(\alpha^4, \alpha^5\right) + \\ &\left(\alpha^4, \alpha^7\right) + \left(\alpha^5, 1\right) + \left(\alpha^5, \alpha\right) + \left(\alpha^5, \alpha^3\right) + \left(\alpha^6, \alpha^4\right) + \left(\alpha^6, \alpha^5\right) + \left(\alpha^6, \alpha^7\right) + \\ &\left(\alpha^7, 1\right) + \left(\alpha^7, \alpha\right) + \left(\alpha^7, \alpha^3\right) + \left(0, \alpha^2\right) + \left(0, \alpha^6\right) + (0, 0)\,. \end{aligned}$$

A base para o espaço $L\,(18Q)$ (veja equação 2.12) é o conjunto dos monômios $x^i y^j$, em que

$$3i + 4j \leq 18 \text{ e } i \leq 3.$$

Tem-se, então, que

$$\mathcal{B}_{18} = \left\{ \underset{1}{1}, \underset{2}{x}, \underset{3}{x^2}, \underset{4}{x^3}, \underset{5}{x^4}, \underset{6}{x^5}, \underset{7}{x^6}, \underset{8}{y}, \underset{9}{xy}, \underset{10}{x^2y}, \underset{11}{x^3y}, \underset{12}{x^4y}, \underset{13}{y^2}, \underset{14}{xy^2}, \underset{15}{x^2y^2}, \underset{16}{x^3y^2} \right\}.$$

Os parâmetros deste código C_{18} são $n = r^3 = 27$, $k = 16$ (equação 2.15) e $d = 9$

(equação 2.18). Sua matriz geradora (equação 2.16) é

$$M_{18} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & \cdots & 23 & 24 & 25 & 26 & 27 \end{matrix} \\ \left[\begin{array}{ccccccccccc} 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \alpha & \alpha & \cdots & \alpha^7 & \alpha^7 & 0 & 0 & 0 \\ 1 & 1 & 1 & \alpha^2 & \alpha^2 & \cdots & \alpha^6 & \alpha^6 & 0 & 0 & 0 \\ 1 & 1 & 1 & \alpha^3 & \alpha^3 & \cdots & \alpha^5 & \alpha^5 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^2 & \alpha^6 & 1 & \alpha^2 & \cdots & \alpha^2 & \alpha^6 & \alpha^4 & \alpha^4 & 0 \\ 1 & \alpha^2 & \alpha^6 & \alpha & \alpha^2 & \cdots & \alpha & \alpha^5 & 0 & 0 & 0 \\ 1 & \alpha^2 & \alpha^6 & \alpha^2 & \alpha^4 & \cdots & 1 & \alpha^4 & 0 & 0 & 0 \\ 1 & \alpha^2 & \alpha^6 & \alpha^3 & \alpha^5 & \cdots & \alpha^7 & \alpha^3 & 0 & 0 & 0 \end{array}\right] \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ \\ 13 \\ 14 \\ 15 \\ 16 \end{matrix} \end{array}.$$

O espaço de funções ortogonal a este código C_{18} é $L(13Q)$. A base para este espaço é (equação 2.12)

$$\mathcal{B}_{13} = \left\{ \underset{1}{1}, \underset{2}{x}, \underset{3}{x^2}, \underset{4}{x^3}, \underset{5}{x^4}, \underset{6}{y}, \underset{7}{xy}, \underset{8}{x^2y}, \underset{9}{x^3y}, \underset{10}{y^2}, \underset{11}{xy^2} \right\}.$$

Portanto, a matriz de paridade do código C_{18} é (equação 2.16)

$$H_{18} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & \cdots & 23 & 24 & 25 & 26 & 27 \end{matrix} \\ \left[\begin{array}{ccccccccccc} 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \alpha & \alpha & \cdots & \alpha^7 & \alpha^7 & 0 & 0 & 0 \\ 1 & 1 & 1 & \alpha^2 & \alpha^2 & \cdots & \alpha^6 & \alpha^6 & 0 & 0 & 0 \\ 1 & 1 & 1 & \alpha^3 & \alpha^3 & \cdots & \alpha^5 & \alpha^5 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha^4 & \alpha^5 & \alpha^7 & \alpha^2 & \alpha^3 & \cdots & \alpha^7 & \alpha & 0 & 0 & 0 \\ \alpha^4 & \alpha^5 & \alpha^7 & \alpha^3 & \alpha^4 & \cdots & \alpha^6 & 1 & 0 & 0 & 0 \\ 1 & \alpha^2 & \alpha^6 & 1 & \alpha^2 & \cdots & \alpha^2 & \alpha^6 & \alpha^4 & \alpha^4 & 0 \\ 1 & \alpha^2 & \alpha^6 & \alpha & \alpha^2 & \cdots & \alpha & \alpha^5 & 0 & 0 & 0 \end{array}\right] \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{matrix} \end{array}$$

2.4 Novos limitantes dos códigos

Um dos principais motivos que despertaram grande interesse nos códigos AG foi o fato destes constituírem bons códigos com parâmetros que ultrapassam o limite de Gilbert-Varshamov (*cf.* apêndice B) em uma determinada faixa, para um corpo de dimensão mínima.

Considere o código $C(D, G)$ definido por (2.1), em que a curva $\mathcal{X}$ de gênero g possui os $n+1$ pontos racionais $P_1, \ldots, P_n$ e Q e o divisor $G = mQ$, com $2g-2 < \deg(G) = m < n$. Defina $\gamma(\mathcal{X}) = \frac{g}{n}$. Foi mostrado por Tsfasman, Vlăduţ e Zink [71] que existe um seqüência

de curvas $\mathcal{X}_i$s, cujos códigos AG correspondentes apresentam parâmetros que ultrapassam o limite de Gilbert-Varshamov (*cf.* (B.5)). De fato, eles provaram o teorema que segue.

Teorema 7 *Considere q uma potência de um inteiro primo e o quadrado de algum inteiro. Existe uma seqüência de curvas $\mathcal{X}_i$s sobre $\mathbb{F}_q$, tal que $\mathcal{X}_i$ possui $n_i + 1$ pontos racionais e gênero g_i, em que $n \to \infty$ e $\gamma(\mathcal{X}_i) \to \frac{1}{\sqrt{q}-1}$ quando $i \to \infty$.*

De (2.2) e (2.3), tem-se que um código $C_i = C(D, m_i Q)$ definido sobre uma curva $\mathcal{X}_i$ possui taxa de informação

$$R_i = \frac{m_i - g_i + 1}{n_i}$$

e distância mínima

$$d_i \geq n_i - m_i \Rightarrow \delta_i \geq 1 - \frac{m_i}{n_i}.$$

Daí, obtém-se que

$$\begin{aligned} R_i + \delta_i &\geq \frac{m_i - g_i + 1}{n_i} + 1 - \frac{m_i}{n_i} \\ &= 1 - \gamma(\mathcal{X}_i) + \frac{1}{n_i}. \end{aligned}$$

Fazendo n tender para infinito e eliminando o termo que tende a zero, obtém-se o chamado *limite de Tsfasman-Vlăduţ-Zink* dado por

$$R(\delta) \geq 1 - \frac{1}{\sqrt{q} - 1} - \delta. \tag{2.19}$$

Deduz-se facilmente que este limitante inferior é melhor que o limitante dado em (B.5) em uma determinada faixa de valores, para $q \geq 43$. Como q deve ser o quadrado de algum inteiro, então o limite de Tsfasman-Vlăduţ-Zink é melhor que o limite de Gilbert-Varshamov para $q \geq 49$. A figura 2.1 compara os limites de (B.5) e (2.19) para $q = 64$.

Em 2001, C. Xing demonstrou que ambos os limites de Gilbert-Varshamov e de Tsfasman-Vlăduţ-Zink podem ser melhorados em torno dos pontos em que as curvas dos dois limitantes (figura 2.1) se interceptam [77].

2.5 Conclusões

O presente capítulo definiu os códigos AG e suas diferentes formas de construção. Foram descritos e analisados especialmente os códigos AG construídos a partir de curvas de Hermite, os chamados códigos de Hermite, muito utilizados na maioria das publicações nesta área.

Em seguida, foi discutida a questão dos limitantes dos códigos e mostrado que os códigos AG apresentam melhores parâmetros que os códigos até então conhecidos, e estabelecem um limitante inferior para a taxa dos códigos melhor que o de Gilbert-Varshamov.

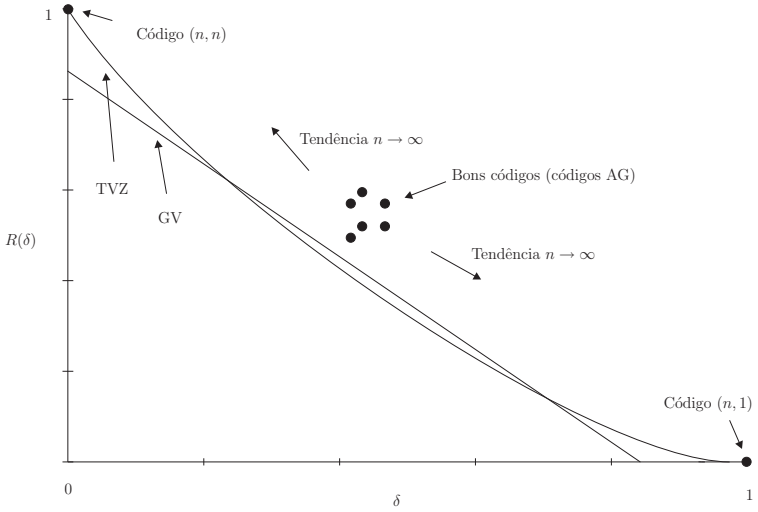

Figura 2.1: Comparação do limite de Gilbert-Varshamov (curva GV) com o limite de Tsfasman-Vlăduţ-Zink (curva TVZ) para $q = 64$. Bons códigos situam-se próximos ao centro da curva, como é o caso dos códigos AG, enquanto os demais tendem para os extremos da curva quando $n \to \infty$.

O capítulo seguinte é dedicado à discussão do problema da decodificação dos códigos AG. Ele apresenta as principais abordagens de decodificação destes códigos e discute sobre a questão da implementação destes decodificadores.

Capítulo 3

Decodificação de Códigos AG

O capítulo anterior introduziu o tema do presente trabalho de doutorado. Ele apresentou os códigos AG, suas formas de construção, as curvas de Hermite e os códigos definidos sobre estas curvas, além de uma discussão sobre os limitantes de códigos obtidos pelos códigos AG, melhores que o limite de Gilbert-Varshamov.

O presente capítulo tem como principal objetivo apresentar um panorama geral sobre a questão da decodificação dos códigos AG. Inicialmente, é descrito e analisado o problema geral da decodificação. Em seguida, são descritas a principais abordagens de decodificação dos códigos AG, a saber:

- A abordagem do algoritmo básico;
- A abordagem do algoritmo de Euclides;
- A decodificação de lista;
- A decodificação a decisão suave.

Por fim, é apresentada uma discussão sobre a questão da implementação em hardware de decodificadores para estes códigos.

No apêndice C, como referência, são definidos os conceitos da geometria algébrica relativos à discussão apresentada neste trabalho sobre os códigos AG e seus esquemas de decodificação.

3.1 Problema da decodificação

Considere C um código em $\mathbb{F}_q^n$ de distância mínima[1] d. Em um sistema de comunicação digital, se $\vec{c} \in C$ for uma palavra código transmitida através do canal de comunicação e

[1] A distância mínima considerada neste capítulo é a chamada distância mínima projetada do código.

se $\vec{v} = \vec{c} + \vec{e}$ for a palavra recebida correspondente, então o vetor $\vec{e} \in \mathbb{F}_q^n$ será dito o *vetor erro*, $\{i : e_i \neq 0\}$ será o conjunto das *posições dos erros*, os e_is serão os *valores dos erros* naquelas posições e $w(\vec{e})$ (peso do vetor $\vec{e}$) será o *número de erros* ocorridos na palavra recebida. Se $w(\vec{e}) \leq \frac{d-1}{2}$, então a palavra recebida $\vec{v}$ pode ser univocamente associada pelo decodificador à palavra código mais próxima $\vec{c}$.

Considere ainda que C é um código linear de dimensão k e taxa de informação $R = \frac{k}{n}$. Sua matriz geradora é uma matriz $G_{k \times n}$, tal que $C = \left\{\vec{x}G : \vec{x} \in \mathbb{F}_q^k\right\}$. Portanto, um determinado mapeamento

$$\psi : \mathbb{F}_q^k \to \mathbb{F}_q^n$$

definido por $\psi(\vec{x}) = \vec{x}G$ é dito ser um *codificador* linear, que codifica palavras de comprimento k em palavras código de C de comprimento n.

Um mapeamento

$$\delta : \mathbb{F}_q^n \to C^*,$$

em que $C^* = C \cup C^?$ e $C \cap C^? = \varnothing$, é chamado *decodificador* para um código C, se $\delta(\vec{v}) = \vec{c} \in C$, ou se $\delta(\vec{v}) \in C^?$, sendo $C^?$ o conjunto que compreende os casos em que o decodificador não encontra uma palavra código correspondente à palavra recebida $\vec{v}$.

Definição 8 (Decodificador de menor distância) *Um* decodificador de menor distância *para um código C é um decodificador δ, em que $\delta(\vec{v}) = \vec{c}$ é a palavra código mais próxima de $\vec{v}$, ou $\delta(\vec{v}) \in C^?$.*

Diz-se que ocorreu *erro de decodificação* quando a palavra decodificada é diferente da palavra código transmitida.

Observe que o código linear C pode também ser definido em função de sua matriz de paridade $H_{(n-k) \times n}$ por $C = \left\{\vec{c} \in \mathbb{F}_q^n : H\vec{c}^T = 0\right\}$. As linhas $\vec{h}_i$, com $i = 1, \ldots, n-k$, da matriz H formam, por definição, uma base para o código $C^\perp$ dual de C (ou ortogonal ao espaço vetorial C).

Considere agora $\vec{v} = \vec{c} + \vec{e}$ uma palavra recebida, em que $\vec{v}, \vec{e} \in \mathbb{F}_q^n$ e $\vec{c} \in C$. As $n-k$ *síndromes* de $\vec{v}$ podem ser definidas por[2]

$$S_i(\vec{v}) = \vec{h}_i \vec{v}^T, \ i = 1, \ldots, n-k. \tag{3.1}$$

Observe que, como por definição $S_i(\vec{c}) = \vec{h}_i \vec{c}^T = 0$, tem-se que $S_i(\vec{v}) = \vec{h}_i(\vec{c} + \vec{e})^T = \vec{h}_i \vec{e}^T$. Conclui-se que se pode obter informações importantes acerca dos erros ocorridos na comunicação (vetor $\vec{e}$) por meio das síndromes da palavra recebida $\vec{v}$. Em geral, as síndromes

[2]Há uma abordagem diferente na decodificação, em que as síndromes são definidas como elementos de um anel afim (um anel de funções racionais), ao invés de na forma de um mapeamento de um subespaço linear de funções em um corpo de localização (corpo finito), como descrito acima. Ambas as abordagens serão analisadas nas seções seguintes.

são as únicas informações que se dispõe no decodificador acerca dos erros ocorridos (numa decodificação a decisão suave, o decodificador dispõe ainda de informações sobre a confiabilidade de cada elemento do vetor recebido). Na prática, a tarefa de decodificação consiste basicamente na determinação do vetor $\vec{e}$ ocorrido através do uso de síndromes, partindo-se da premissa de que o peso de $\vec{e}$ não ultrapassa um limite t (*decodificador de distância limitada* definido a seguir), caso em que ocorreria erro de decodificação (decodificador de t erros)[3].

Um esquema importante na decodificação de códigos AG consiste em considerar uma extensão da matriz H denotada por $\hat{H}_{n\times n}$, cujas n linhas $\vec{\hat{h}}_i$, com $i = 1, \ldots, n$, constituem uma base para o espaço $\mathbb{F}_q^n$ e as primeiras $n-k$ linhas coincidem com a matriz de paridade H. As n síndromes obtidas das linhas da matriz $\hat{H}$ determinam univocamente o vetor erro, mas apenas as $n-k$ primeiras são conhecidas. As k síndromes restantes são chamadas *síndromes desconhecidas*. Numa subseção seguinte, será apresentado um procedimento baseado em um esquema denominado *decisão por maioria* ("majority voting"), que determina as síndromes desconhecidas, permitindo determinar o vetor erro e, conseqüentemente, a palavra código original.

O conjunto de todas as palavras de $\mathbb{F}_q^n$ que apresentam a mesma síndrome de uma palavra $\vec{v}$ é dito uma *classe lateral*. A função síndrome, então, estabelece uma relação de equivalência entre as palavras, sendo cada classe lateral uma classe de equivalência (*cf.* tabela 3.1). O elemento de uma classe lateral de menor peso é dito o *líder da classe lateral*.

					Esfera de decodificação
	0	$\vec{c}_2$	$\vec{c}_3$	$\cdots$	$\vec{c}_{q^k}$
	$\vec{v}_2$	$\vec{c}_2+\vec{v}_2$	$\vec{c}_3+\vec{v}_2$	$\cdots$	$\vec{c}_{q^k}+\vec{v}_2$
Classe Lateral	$\vec{v}_3$	$\vec{c}_2+\vec{v}_3$	$\vec{c}_3+\vec{v}_3$	$\cdots$	$\vec{c}_{q^k}+\vec{v}_3$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$\vec{v}_{q^{n-k}}$	$\vec{c}_2+\vec{v}_{q^{n-k}}$	$\vec{c}_3+\vec{v}_{q^{n-k}}$	$\cdots$	$\vec{c}_{q^k}+\vec{v}_{q^{n-k}}$
	Líderes				

Tabela 3.1: Organização do espaço decodificável em classes laterais de palavras.

Um esquema simples de decodificação de menor distância consiste na procura exaustiva pelo líder da classe lateral correspondente à palavra recebida. Outra possibilidade seria listar e armazenar todos os líderes de classes laterais, de modo a evitar o esforço de procura.

[3]Há um tipo de decodificação, chamada decodificação de lista, que permite a ocorrência de mais de t erros. Neste caso, o decodificador fornece um conjunto de palavras código que estão a uma certa distância do vetor recebido maior que a distância mínima do código.

No primeiro caso, é necessária a busca entre os q^k elementos da classe lateral da palavra recebida pelo elemento de peso mínimo. No segundo caso, é necessário armazenar q^{n-k} líderes de classes laterais. Observe que ambos os esquemas implicam uma complexidade[4] computacional que é função exponencial do comprimento do código, o que inviabiliza sua utilização em aplicações práticas. Os decodificadores de menor distância conhecidos, todos eles possuem complexidade exponencial, não sendo mais abordados neste trabalho.

Definição 9 (Decodificador de distância limitada) *Um decodificador δ para um código C é dito ser um* decodificador de distância limitada *corretor de t erros, se $\delta(\vec{v}) = \vec{c}$ for a palavra código mais próxima de $\vec{v}$ e $d(\vec{v}, \vec{c}) \leq t$, para todo $\vec{v} \in \mathbb{F}_q^n$, ou se $\delta(\vec{v}) \in C^?$.*

Sendo C um código de distância mínima d, no caso em que $t = \frac{d-1}{2}$, o decodificador δ é dito *decodificar até metade da distância mínima*.

Considere agora C um código linear em $\mathbb{F}_q^n$ com uma matriz de paridade H. Suponha uma palavra recebida $\vec{v}$ associada a um vetor erro $\vec{e}$ de peso $w(\vec{e}) \leq \frac{d-1}{2}$, em que o conjunto $\{i : e_i \neq 0\}$ das até $\frac{d-1}{2}$ posições dos erros ocorridos é conhecido. Observe que o vetor $\vec{e}$ é a única solução da equação

$$H\vec{x}^T = H\vec{v}^T, \tag{3.2}$$

em que $x_j = 0$, para todo $j \notin \{i : e_i \neq 0\}$.

É óbvio que o vetor erro $\vec{e}$ é uma solução desta equação. Supondo que $\vec{x}$ seja uma solução diferente de $\vec{e}$, tem-se que $H\vec{x}^T = H\vec{e}^T \Rightarrow H(\vec{x} - \vec{e})^T = 0$. Portanto, $\vec{x} - \vec{e}$ é um elemento de C e, além disso, tem seu suporte em $\{i : e_i \neq 0\}$. Como seu peso é menor ou igual a $\frac{d-1}{2}$, conclui-se que $\vec{x} - \vec{e} = 0$, ou seja, $\vec{x} = \vec{e}$ é a única solução possível para (3.2).

Desta forma, vê-se que o problema da decodificação fica reduzido à procura pelas posições dos erros. Uma vez conhecido o conjunto $\{i : e_i \neq 0\}$ das posições dos erros ocorridos, pode-se determinar o vetor $\vec{e}$ através de (3.2).

A grande parte dos algoritmos de decodificação existentes fornece um polinômio, um vetor, uma função ou um ideal de funções que localiza os erros. O conjunto $\{i : e_i \neq 0\}$ das posições dos erros consiste, na prática, no conjunto dos zeros da função ou das funções fornecidas pelos algoritmos.

A seção que segue apresenta as principais abordagens de decodificação dos códigos AG, no sentido de fornecer um panorama acerca do problema da decodificação destes códigos.

[4]Neste texto, é utilizada a notação $\mathcal{O}(g(n))$ para a complexidade dos algoritmos descritos, em que $g(n)$ é normalmente um polinômio e expressa a ordem de grandeza do número de operações básicas em corpo finito (ou iterações algorítmicas) necessárias para a obtenção do resultado.

3.2 Decodificação dos códigos AG

O primeiro esquema de decodificação para os códigos AG foi proposto no final da década de 80 por Justesen, Larsen, Jensen, Havemose e Høholdt [32] e consiste numa generalização do algoritmo PGZ (Peterson-Gorenstein-Zierler) [6] para decodificação de códigos BCH. Ele, de modo semelhante ao PGZ, fornece um polinômio localizador de erros em duas variáveis, que possui entre seus zeros as posições dos erros ocorridos na palavra recebida.

Em 1990, Skorobogatov e Vlăduţ [67] propuseram uma generalização do algoritmo de Justesen et al para curvas arbitrárias (ao invés de planas apenas). O algoritmo proposto ficou conhecido como *algoritmo básico* e é capaz de corrigir até $\frac{d-g-1}{2}$ erros ocorridos na palavra recebida, em que g é o gênero da curva usada na geração do código e d sua distância mínima projetada. Sua complexidade algorítmica é $\mathcal{O}\left(d^2n + g^2n\right) \leq \mathcal{O}\left(n^3\right)$, sendo n o comprimento do código.

Ainda neste mesmo artigo, Skorobogatov e Vlăduţ apresentaram uma modificação do algoritmo básico, conhecida como *algoritmo modificado*, que corrige até $\frac{d-1}{2} - \sigma$ erros, em que σ é o chamado *defeito de Clifford* [67], que é aproximadamente igual a $\frac{g}{2}$ no caso de curvas planas. O algoritmo modificado tem complexidade $\mathcal{O}\left(n^4\right)$.

Uma abordagem diferente na decodificação dos códigos AG, que consiste numa generalização do algoritmo de Euclides para solução da *equação chave* [13, 23], foi proposta por Porter [54] em 1992. O algoritmo proposto corrige até $\frac{d-1}{2} - \sigma$ erros. Neste caso, o conjunto dos zeros correspondentes às posições dos erros é obtido de um ideal, ao invés de um polinômio. Mostrou-se que ambos, o algoritmo modificado e o algoritmo de Porter, são, na verdade, equivalentes.

Um dos primeiros trabalhos a proporcionar uma decodificação até metade da distância mínima foi proposto por Feng e Rao [21]. A elegante solução apresentada utiliza um esquema de decisão por maioria para determinar as síndromes desconhecidas da palavra recebida. Como foi afirmado na seção anterior, conhecidas todas as n síndromes, pode-se determinar o vetor erro ocorrido. Este esquema de decisão por maioria de Feng e Rao foi aplicado posteriormente ao algoritmo de Porter por Shen e Tzeng [65].

A complexidade dos algoritmos acima descritos é considerada muito elevada para aplicações práticas, em que necessita-se utilizar códigos com comprimento elevado. No entanto, diversos esquemas rápidos de decodificação, ou seja, com menor complexidade, têm sido propostos.

Em 1990, Sakata [58,60], apresentou uma generalização em várias variáveis do clássico algoritmo de Berlekamp-Massey [6], que passou a ser conhecida como *algoritmo BMS*. Com base neste algoritmo, diversas implementações rápidas têm sido apresentadas, tais como uma versão do algoritmo modificado por Justesen, Larsen, Jensen e Høholdt [33] e

uma versão usando decisão por maioria por Sakata, Justesen, Madelung, Jensen e Høholdt [61], além de outras.

Em [22], Feng, Wei, Rao e Tzeng propuseram um esquema de decodificação baseado em matrizes de blocos de Hankel de códigos sobre curvas planas que, segundo o que apresentaram, proporciona uma considerável redução da complexidade do esquema de decisão por maioria. Eles afirmam que o algoritmo rápido proposto tem, no pior caso, complexidade $\mathcal{O}\left((r+1)\,n^2\right)$, em que $r+1$ é o grau da curva algébrica usada na definição dos códigos.

Com relação à correção de erros e apagamentos, o próprio algoritmo básico de Skorobogatov e Vlăduţ [67] a faz. Existem diversos outros trabalhos propostos no sentido de se corrigir erros e apagamentos, a exemplo do esquema de Sakata, Leonard, Jensen e Høholdt [64], que associa o algoritmo BMS à decisão por maioria e à correção de apagamentos.

Em 1999, M. A. Shokrollahi e H. Wasserman apresentaram um esquema de decodificação de códigos AG chamado *decodificação de lista*, que consiste na busca pelo conjunto de palavras código que se encontram a uma determinada distância maior que a metade da distância mínima do código [66]. Esta idéia alternativa de decodificação foi proposta na década de 1950 por P. Elias [20] e J. M. Wozencraft [74], contrariando o conceito tradicional em que a decodificação consiste na busca pela única palavra código, caso exista, situada dentro da esfera de decodificação de diâmetro igual à distância mínima do código. Recentemente, alguns trabalhos têm sido publicados nesta linha envolvendo os códigos AG [26, 76].

Para maiores detalhes sobre a história da decodificação dos códigos AG, veja o artigo de Høholdt e Pellikaan [29].

3.2.1 Abordagens de decodificação

Pode-se observar algumas abordagens distintas no desenvolvimento dos algoritmos de decodificação para códigos AG. Algumas consideradas principais são descritas neste texto:

1. Uma primeira abordagem, que é a utilizada no algoritmo básico de Skorobogatov e Vlăduţ [67], em que a síndrome é definida como um mapeamento de um subespaço linear de funções em um corpo de localização[5] (caso de (3.1)). Neste caso, a decodificação consiste na solução de um conjunto de equações lineares sobre este corpo;

2. Uma segunda abordagem, que é a utilizada no algoritmo de Porter [54], em que a síndrome é definida como um elemento em um anel afim. Neste caso, a decodificação

[5]Este corpo de localização é, de fato, um corpo finito.

consiste na solução de uma equação chave neste anel;

3. Uma terceira abordagem, da decodificação de lista [66], que consiste na busca pelo conjunto de palavras código que se encontram a uma certa distância maior que a metade da distância mínima do código, em que se admite a ocorrência de mais erros que a capacidade de correção deste;

4. Uma quarta abordagem que merece menção é a que envolve a utilização de esquemas de decisão suave, em que o demodulador fornece informações extras sobre a mensagem recebida ao decodificador, que as utiliza para melhorar seu desempenho na decodificação [5, 34, 36, 52].

De modo a possibilitar uma melhor compreensão destas abordagens, serão descritos nas seções seguintes o algoritmo básico (primeira abordagem), o algoritmo de Porter (segunda abordagem), o algoritmo de Shokrollahi e Wasserman [66] (terceira abordagem), e o algoritmo GMD [31] (quarta abordagem).

São também esquemas importantes que devem ser observados o de decisão por maioria de Feng e Rao e o algoritmo BMS, que possibilitaram implementações mais eficientes na decodificação dos códigos AG. Estes dois esquemas serão apresentados logo após a descrição do algoritmo básico.

3.2.2 Algoritmo básico (primeira abordagem)

A primeira abordagem utilizada no desenvolvimento de esquemas de decodificação para códigos AG é aquela em que as síndromes são definidas como um mapeamento de um subespaço linear de funções em um corpo de localização. O algoritmo básico apresentado por Skorobogatov e Vlăduţ [67] é um exemplo típico desta primeira abordagem.

Considere $\mathcal{X}$ uma curva algébrica de gênero g. Considere $\mathcal{P}_{\mathcal{X}} = \{P_1, \ldots, P_n\}$ um conjunto de pontos racionais (equivalentes a lugares de grau 1) da curva $\mathcal{X}$ sob o corpo algebricamente fechado $\mathbb{F}_q$ e o divisor $D = P_1 + \cdots + P_n$, cujo suporte é $\mathcal{P}_{\mathcal{X}}$. Considere $G = aQ$ um divisor de grau $a \geq 0$, em que $Q \notin \mathcal{P}_{\mathcal{X}}$ é um ponto de $\mathcal{X}$ (suporte de G disjunto de $\mathcal{P}_{\mathcal{X}}$). Suponha também que

$$2g - 2 < a \leq n + g - 1.$$

Considere aqui o código AG $C^*(D, G)$ (dual do código $C(D, G)$), que será denotado

simplesmente por C, cuja matriz de paridade é

$$H_G = \begin{bmatrix} f_1(P_1) & f_1(P_2) & \cdots & f_1(P_n) \\ f_2(P_1) & f_2(P_2) & \cdots & f_2(P_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_m(P_1) & f_m(P_2) & \cdots & f_m(P_n) \end{bmatrix}, \tag{3.3}$$

em que $\{f_1, \ldots, f_m\}$ é uma base para o espaço $L(G)$[6].

Considere uma palavra recebida $\vec{v} \in \mathbb{F}_q^n$ e as funções $f_1, \ldots, f_m$ da base de $L(G)$. As síndromes de $\vec{v}$ são definidas (*cf.* (3.1)) por

$$S(\vec{v}, f_i) = \sum_{j=1}^{n} v_j f_i(P_j), \quad i = 1, \ldots, m, \tag{3.4}$$

caracterizando a primeira abordagem supracitada.

O algoritmo básico proposto por Skorobogatov e Vlăduţ [67] permite a correção simultânea de erros e apagamentos[7]. Considere $\vec{e} \in \mathbb{F}_q^n$ o vetor de erros ocorridos, em que $w(\vec{e}) = t$, e $\vec{r} \in \mathbb{F}_q^n$ o vetor de apagamentos, sendo $w(\vec{r}) = \tau$. Denote por $\{E_1, \ldots, E_t\} \subset \mathcal{P}_\mathcal{X}$ e por $\{R_1, \ldots, R_\tau\} \subset \mathcal{P}_\mathcal{X}$ os conjuntos disjuntos de pontos de $\mathcal{P}_\mathcal{X}$ correspondentes às posições dos erros e dos apagamentos, respectivamente. Deve-se observar que, de fato, o algoritmo trata estas posições (pontos de $\mathcal{P}_\mathcal{X}$) como elementos de $\mathbb{F}_q$.

Além do divisor $G = aQ$, o algoritmo básico (algoritmo 10 a seguir) depende ainda de um divisor auxiliar $F = bQ$, em que $b \leq a$. A capacidade do algoritmo básico de corrigir t erros e τ apagamentos está condicionada a este divisor F, que pode ser determinado pelas inequações [67]

$$l(F) > t + \tau \tag{3.5}$$

e

$$a - b > t + 2g - 2. \tag{3.6}$$

Relativas a este divisor auxiliar F, são consideradas ainda as matrizes

$$H_F = \begin{bmatrix} k_1(P_1) & k_1(P_2) & \cdots & k_1(P_n) \\ k_2(P_1) & k_2(P_2) & \cdots & k_2(P_n) \\ \vdots & \vdots & \ddots & \vdots \\ k_s(P_1) & k_s(P_2) & \cdots & k_s(P_n) \end{bmatrix}, \tag{3.7}$$

[6]Normalmente, os algoritmos de decodificação trabalham sobre códigos AG construídos por resíduos de diferenciais ($C^*(D, G)$), ao invés de códigos AG construídos pela avaliação de funções racionais ($C(D, G)$). Isto, porque, no primeiro caso, a descrição da matriz de paridade é mais simples, feita pela avaliação de funções racionais em pontos da curva.

[7]A única diferença entre erros e apagamentos é que as posições dos apagamentos são previamente conhecidas pelo decodificador, restando ao algoritmo determinar apenas os valores destes apagamentos.

em que $\{k_1, \ldots, k_s\}$ é uma base para o espaço de funções $L(F)$, e

$$H_{G-F} = \begin{bmatrix} h_1(P_1) & h_1(P_2) & \cdots & h_1(P_n) \\ h_2(P_1) & h_2(P_2) & \cdots & h_2(P_n) \\ \vdots & \vdots & \ddots & \vdots \\ h_k(P_1) & h_k(P_2) & \cdots & h_k(P_n) \end{bmatrix}, \tag{3.8}$$

sendo $\{h_1, \ldots, h_k\}$ uma base para o espaço $L(G-F)$.

Algoritmo 10 (Algoritmo básico)

Entradas:

- *Uma palavra recebida* $\vec{v} = [v_1 \ \cdots \ v_n]$*;*
- *O conjunto* $\{R_1, \ldots, R_\tau\}$ *das posições de apagamentos;*
- *As matrizes* H_G (cf. *(3.3)*), H_F (cf. *(3.7)*) *e* H_{G-F} (cf. *(3.8)*).

Saídas:

- *O vetor* $\vec{e} + \vec{r}$ *de erros e apagamentos.*

<<< **Passo 1** >>>
Determine a matriz

$$H_{F-R} = \begin{bmatrix} g_1(P_1) & g_1(P_2) & \cdots & g_1(P_n) \\ g_2(P_1) & g_2(P_2) & \cdots & g_2(P_n) \\ \vdots & \vdots & \ddots & \vdots \\ g_l(P_1) & g_l(P_2) & \cdots & g_l(P_n) \end{bmatrix},$$

em que $\{g_1, \ldots, g_l\}$ *é uma base para o espaço de funções* $L(F - \sum R_i)$*. Observe que este espaço consiste nas funções de* $L(F)$ *que possuem zeros nas posições* $R_1, \ldots, R_\tau$*. Como* $L(F - \sum R_i) \subseteq L(F)$*, então as funções da base* $\{g_1, \ldots, g_l\}$ *podem ser escritas na forma* $\sum_j x_j k_j$*, sendo* $x_j \in \mathbb{F}_q$*. Portanto,*

$$\sum_j k_j(R_i)\, x_j = 0.$$

Tem-se, pois, o sistema

$$\begin{bmatrix} k_1(R_1) & k_2(R_1) & \cdots & k_s(R_1) \\ k_1(R_2) & k_2(R_2) & \cdots & k_s(R_2) \\ \vdots & \vdots & \ddots & \vdots \\ k_1(R_\tau) & k_2(R_\tau) & \cdots & k_s(R_\tau) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Do espaço de soluções deste sistema, $l(F - \sum R_i) = \deg(F - \sum R_i) + 1 - g$ *soluções linearmente independentes podem ser obtidas. As combinações lineares das funções da base de* $L(F)$ *correspondentes às soluções obtidas neste sistema determinam as funções da base de* $L(F - \sum R_i)$*, das quais deriva a matriz* H_{F-R}*.*

<<< **Passo 2** >>>

Observe que $g_i h_j \in L(G)$. *Determine as* lk *síndromes* $S(\vec{v}, g_i h_j)$ *(*cf. *(3.4)), com* $i = 1, \ldots, l$ *e* $j = 1, \ldots, k$.

<<< **Passo 3** >>>

Determine uma solução não trivial $(y_1, \ldots, y_l)$ *para o sistema*

$$\begin{bmatrix} S(\vec{v}, g_1 h_1) & S(\vec{v}, g_2 h_1) & \cdots & S(\vec{v}, g_l h_1) \\ S(\vec{v}, g_1 h_2) & S(\vec{v}, g_2 h_2) & \cdots & S(\vec{v}, g_l h_2) \\ \vdots & \vdots & \ddots & \vdots \\ S(\vec{v}, g_1 h_k) & S(\vec{v}, g_2 h_k) & \cdots & S(\vec{v}, g_l h_k) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3.9}$$

A condição dada por (3.5) garante que este sistema possui pelo menos uma solução não trivial.

<<< **Passo 4** >>>

Dada a solução $(y_1, \ldots, y_l)$ *obtida no passo 3, determine o conjunto de zeros* $\{Q_1, \ldots, Q_u\} \subset \mathcal{P}_{\mathcal{X}}$ *da equação*

$$g_y = y_1 g_1 + \cdots + y_l g_l. \tag{3.10}$$

Isto é feito examinando-se os pontos de $\mathcal{P}_{\mathcal{X}}$ *um a um em (3.10). A condição de (3.6) garante que* g_y *possui entre seus zeros as posições dos erros e apagamentos ocorridos.*

<<< **Passo 5** >>>

Determine as síndromes $S(\vec{v}, f_i)$ *(*cf. *(3.4)), com* $i = 1, \ldots, m$.

<<< **Passo 6** >>>

*Determine uma solução para o sistema linear (*cf. *(3.2))*

$$\begin{bmatrix} f_1(Q_1) & f_1(Q_2) & \cdots & f_1(Q_u) \\ f_2(Q_1) & f_2(Q_2) & \cdots & f_2(Q_u) \\ \vdots & \vdots & \ddots & \vdots \\ f_m(Q_1) & f_m(Q_2) & \cdots & f_m(Q_u) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_u \end{bmatrix} = \begin{bmatrix} S(\vec{v}, f_1) \\ S(\vec{v}, f_2) \\ \vdots \\ S(\vec{v}, f_m) \end{bmatrix},$$

que determina os valores dos erros e apagamentos indicados por g_y *(*cf. *(3.10)).*

Prova. Para detalhes sobre a prova deste algoritmo, veja [67]. ■

O algoritmo básico possui uma capacidade de correção de t erros e τ apagamentos, em que

$$2t + \tau \leq d - g - 1 = a - 3g + 1. \tag{3.11}$$

Com relação à sua complexidade algorítmica, obedecido o limite para t e τ (*cf.* (3.11)), tem-se que ela não é maior que $\mathcal{O}(d^2 n + g^2 n) \leq \mathcal{O}(n^3)$ [67].

3.2.3 Algoritmo BMS

A grande parte dos primeiros algoritmos de decodificação para códigos AG propostos está baseada no processo de eliminação de Gauss, o que implica uma complexidade algorítmica $\mathcal{O}(n^3)$, em que n é o comprimento do código. Deve-se observar que complexidades altas

são proibitivas para aplicações práticas em que são necessários códigos com comprimento elevado.

Em vista desta necessidade de esquemas de decodificação para códigos AG mais rápidos, ou seja, de menor complexidade algorítmica, diversos trabalhos vêm sendo apresentados neste sentido. O protagonista ou elemento mais importante, que tem tornado possível o desenvolvimento destes esquemas mais eficientes de decodificação, é o algoritmo BMS proposto por Sakata [58, 60]. O algoritmo BMS tem possibilitado implementações rápidas do algoritmo de Porter e do algoritmo modificado de Skorobogatov e Vlăduţ, mas, principalmente, tem dado origem a algoritmos rápidos de decodificação associado ao esquema de decisão por maioria proposto por Feng e Rao [21]. O primeiro é descrito nesta subseção e o segundo na subseção que segue.

Dado um inteiro $\mu > 0$, considere $\mathbb{Z}_+^\mu$ o conjunto das μ-úplas de inteiros não negativos. Sendo $\alpha = (\alpha_1, \ldots, \alpha_\mu) \in \mathbb{Z}_+^\mu$, considere a notação $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_\mu^{\alpha_\mu}$ para um monômio nas μ variáveis $x_1, \ldots, x_\mu$. Defina, então,

$$f(x_1, \ldots, x_\mu) = \sum_\alpha f_\alpha x^\alpha$$

como um polinômio em $\mathbb{F}_q[x_1, \ldots, x_\mu]$.

Denote por S um arranjo de dimensão μ de elementos $S_\alpha \in \mathbb{F}_q$, em que $\alpha \in \mathbb{Z}_+^\mu$. Diz-se que este arranjo S satisfaz a *relação de recursão linear μ-dimensional* com polinômio característico f se

$$\sum_\alpha f_\alpha S_{\alpha+\gamma} = 0, \tag{3.12}$$

para todo $\gamma \in \mathbb{Z}_+^\mu$, em que $S_{\alpha+\gamma}$ existe. A relação de recursão linear μ-dimensional representada pelo polinômio f é dita ser *válida para o arranjo S*, se (3.12) for satisfeita. O conjunto dos polinômios característicos de todas as relações de recursão lineares μ-dimensionais válidas para o arranjo S constitui um ideal de funções e será denotado aqui por $\mathbf{I}(S)$.

O algoritmo BMS (Berlekamp-Massey-Sakata) constitue uma generalização em μ variáveis do clássico algoritmo de Berlekamp-Massey para decodificação de códigos BCH [6]. O algoritmo de Berlekamp-Massey determina através de relações de recursão lineares (registradores de deslocamento) em uma variável, relações estas válidas para as síndromes da palavra recebida, um polinômio localizador dos erros ocorridos. Já o algoritmo BMS, que tem como entrada um arranjo μ-dimensional S de elementos de $\mathbb{F}_q$, fornece um conjunto de polinômios mínimos característicos (uma base de Gröbner mínima para o ideal de funções $\mathbf{I}(S)$) correspondente às relações de recursão lineares μ-dimensionais válidas para o arranjo S dado.

Num processo de decodificação, em que $\vec{v} = \vec{c} + \vec{e}$ é o vetor recebido, sendo $\vec{e}$ o vetor de erros ocorridos, se S é um arranjo μ-dimensional de síndromes de $\vec{v}$, então tem-se que S satisfaz as relações de recursão lineares μ-dimensionais com polinômio característico f se e só se $f(P) = 0$, para todo $P \in \sup(\vec{e})$. Portanto, o algoritmo BMS é utilizado para construir, a partir de um conjunto completo de síndromes de um vetor recebido, uma base para um ideal de funções (conjunto de polinômios mínimos característicos) em cujos zeros estão as posições dos erros ocorridos. No entanto, deve-se observar que o algoritmo BMS consiste apenas num dos componentes de um esquema de decodificação, uma vez que apenas uma parte do conjunto de síndromes é conhecido. Um algoritmo de decodificação necessita ainda de um componente adicional que determine as síndromes desconhecidas, tal como o esquema de decisão por maioria descrito na subseção seguinte.

Ordenação de monômios

Considere agora a chamada ordenação de monômios lexicográfica graduada reversa. Nesta ordenação, que é denotada por $<_{grev}$, sendo $\alpha = (\alpha_1, \ldots, \alpha_\mu)$ e $\beta = (\beta_1, \ldots, \beta_\mu)$, diz-se que $x^\alpha <_{grev} x^\beta$ ou $\alpha <_{grev} \beta$ se e só se

$$|\alpha| = \sum_i \alpha_i < |\beta| = \sum_i \beta_i$$

ou

$$|\alpha| = |\beta|, \alpha_1 = \beta_1, \ldots, \alpha_{j-1} = \beta_{j-1} \text{ e } \alpha_j > \beta_j.$$

Por exemplo, para $\mu = 2$, tem-se que $(0,0) <_{grev} (1,0) <_{grev} (0,1) <_{grev} (2,0) <_{grev} (1,1) <_{grev} (0,2) <_{grev} (3,0) <_{grev} (2,1) <_{grev} (1,2) <_{grev} \cdots$, ou $1 <_{grev} x <_{grev} y <_{grev} x^2 <_{grev} xy <_{grev} y^2 <_{grev} x^3 <_{grev} x^2y <_{grev} xy^2 <_{grev} y^3 <_{grev} \cdots$.

Recordando da seção C.1.3, dado um polinômio $f(x_1, \ldots, x_\mu) = \sum_\alpha f_\alpha x^\alpha$, segundo esta ordenação $<_{grev}$, denota-se por $\deg(f)$ e $\mathrm{lc}(f)$ o expoente e o coeficiente do monômio líder (monômio de maior ordem), respectivamente.

Será considerada também uma *ordenação simples* de μ-úplas denotada por $<$, em que $\alpha < \beta$ se e só se $\alpha_i < \beta_i$, para todo $1 \leq i \leq \mu$. Esta ordenação simples não constitui propriamente uma ordenação de monômios, segundo a definição da seção C.1.3, mas servirá para expressar a divisibilidade de um monômio por outro.

Conjunto de polinômios mínimos

Uma base de Gröbner mínima $\mathcal{F}$ para o ideal $\mathbf{I}(S)$ descrito a pouco consiste em polinômios característicos de relações de recursão lineares μ-dimensionais que sejam válidas para o arranjo μ-dimensional S, e que possuam apenas monômios líderes mínimos segundo a

ordenação de monômios $<_{grev}$ adotada. Diz-se, daí, que $\mathcal{F}$ é um *conjunto de polinômios mínimos* para o arranjo S de entrada.

Considere agora o caso em que (3.12) é satisfeita apenas em parte do arranjo S. Considere os elementos válidos do arranjo S ordenados segundo a ordem $<_{grev}$ de seus índices (μ-úplas). Se S_α é o elemento válido de maior ordem, em que $\alpha = \deg(f) + \gamma$, para todo $\gamma \in \mathbb{Z}_+^\mu$, então (3.12) pode ser escrita expressando S_α em função dos demais elementos S_β, em que $\beta <_{grev} \alpha$, ou seja,

$$S_\alpha = \frac{-1}{\mathrm{lc}(f)} \sum_{\beta <_{grev} \alpha} f_{\deg(f)-\alpha+\beta} S_\beta. \tag{3.13}$$

A relação de recursão linear μ-dimensional representada pelo polinômio $f(x_1, \ldots, x_\mu)$ é dita agora *válida para o arranjo S até a entrada S_α*, se $\alpha \geq \deg(f)$ (ordenação simples) e (3.13) for satisfeita, ou se $\alpha \ngeq \deg(f)$. Caso contrário, se $\alpha \geq \deg(f)$ e (3.13) não for satisfeita, ela é dita *inválida*.

O conjunto dos polinômios característicos de todas as relações de recursão lineares μ-dimensionais válidas para o arranjo S até a entrada S_α é denotado por $\mathbf{I}_\alpha(S)$. Observe que este conjunto $\mathbf{I}_\alpha(S)$ não constitui um ideal, uma vez que não é fechado sob a adição. Apesar disso, é fechado sob a multiplicação de monômios, ou seja, se $f(x_1, \ldots, x_\mu) \in \mathbf{I}_\alpha(S)$, então $x^\gamma f(x_1, \ldots, x_\mu) \in \mathbf{I}_\alpha(S)$. Além disso, a definição de um conjunto de polinômios mínimos para este conjunto $\mathbf{I}_\alpha(S)$ coincide ainda com a definição de uma base de Gröbner.

Reformulando (3.13), tem-se que um polinômio f pertence ao conjunto $\mathbf{I}_\alpha(S)$ se e só se

$$\sum_\alpha f_\alpha S_{\alpha+\gamma} = 0, \tag{3.14}$$

para todo $\gamma \in \mathbb{Z}_+^\mu$, tal que $\deg(f) + \gamma \leq_{grev} \alpha$.

Conjunto de sentinelas e conjunto delta

Defina o *conjunto delta*, denotado por $\Delta(\mathbf{I}_\alpha(S))$, como o conjunto dos monômios (na verdade, índices) que não constituem termos líderes em qualquer polinômio de $\mathbf{I}_\alpha(S)$. Por esta razão, Sakata denominou $\Delta(\mathbf{I}_\alpha(S))$ de *conjunto de pontos excluídos*. Um conjunto $\mathcal{F} \subset \mathbf{I}_\alpha(S)$ é um conjunto de polinômios mínimos para $\mathbf{I}_\alpha(S)$, se $\Delta(\mathcal{F}) = \Delta(\mathbf{I}_\alpha(S))$.

A validade dos polinômios característicos de um conjunto $\mathcal{F}$, que constitui a saída do algoritmo BMS, pode ser verificada por (3.14), contudo é necessário ainda verificar se os polinômios de $\mathcal{F}$ são mínimos (se seus monômios líderes são mínimos). Para realizar esta verificação, será necessário utilizar um segundo conjunto $\mathcal{G}$ de polinômios, chamado *conjunto de sentinelas*, definido em seguida.

Considere f um polinômio característico de uma relação de recursão linear μ-dimensional válida para o arranjo S até todas as entradas S_β, com $\beta <_{grev} \alpha$, mas não necessariamente até S_α. Defina o *valor predito* P_α para a entrada S_α associado ao polinômio f pela equação

$$P_\alpha(f) = \frac{-1}{\mathrm{lc}(f)} \sum_{\beta <_{grev} \alpha} f_{\deg(f)-\alpha+\beta} S_\beta. \tag{3.15}$$

Observe que esta expressão consiste simplesmente no lado direito de (3.13). Diz-se, então, que a relação de recursão linear μ-dimensional representada pelo polinômio f é válida até a entrada S_α do arranjo S se e só se o valor real de S_α for igual ao valor predito P_α.

Considere agora um polinômio característico $g(x_1, \ldots, x_\mu)$ de uma relação de recursão linear μ-dimensional que é válida para o arranjo S μ-dimensional até todas as entradas S_β, com $\beta <_{grev} \alpha$, mas que é inválida até a entrada S_α. Defina, então, a *extensão* de g como sendo o vetor (índice) dado por

$$\mathrm{Span}(g) = \alpha - \deg(g) \tag{3.16}$$

e sua *discrepância* por

$$\delta_g = \mathrm{lc}(g)\left[S_\alpha - P_\alpha(g)\right] = \sum_\alpha g_\alpha S_{\alpha+\mathrm{Span}(g)} \neq 0. \tag{3.17}$$

Caso $g \notin \mathbf{I}_\alpha(S)$, ou seja, caso $P_\alpha(g) \neq S_\alpha$, tem-se que $\mathrm{Span}(g) \in \Delta(\mathbf{I}_\alpha(S))$. Neste caso, o polinômio $g(x_1, \ldots, x_\mu)$ é dito ser um *sentinela* para o ponto $\mathrm{Span}(g)$.

Defina o conjunto Δ como tendo um canto interior associado a cada sentinela do conjunto de sentinelas $\mathcal{G} \subset \mathbb{F}_q[x_1, \ldots, x_\mu] \backslash \mathbf{I}(S)$. Estes cantos interiores de Δ são também membros do conjunto $\Delta(\mathbf{I}(S))$.

Por fim, a verificação de se um conjunto $\mathcal{F}$ é um conjunto de polinômios mínimos, dado um conjunto de sentinelas $\mathcal{G}$, é feita com base no fato que segue. Se $\mathcal{F} \subset \mathbf{I}_\alpha(S)$ e $\mathcal{G} \subset \mathbb{F}_q[x_1, \ldots, x_\mu] \backslash \mathbf{I}_\alpha(S)$ é um conjunto de sentinelas para o conjunto delta $\Delta(\mathcal{F})$, então $\Delta(\mathcal{F}) = \Delta(\mathbf{I}_\alpha(S))$, o que implica que $\mathcal{F}$ é um conjunto de polinômios mínimos para $\mathbf{I}_\alpha(S)$. Se $\mathcal{F} \subset \mathbf{I}(S)$ e $\mathcal{G} \subset \mathbb{F}_q[x_1, \ldots, x_\mu] \backslash \mathbf{I}(S)$ é um conjunto de sentinelas para o conjunto delta $\Delta(\mathcal{F})$, então $\Delta(\mathcal{F}) = \Delta(\mathbf{I}(S))$, o que implica que $\mathcal{F}$ é uma base de Gröbner mínima para o ideal $\mathbf{I}(S)$.

Descrição do algoritmo

O algoritmo BMS (algoritmo 11) basicamente opera sobre dois conjuntos: um conjunto de polinômios mínimos $\mathcal{F}$ e um conjunto de sentinelas $\mathcal{G}$. Cada iteração do algoritmo toma como entrada um conjunto de polinômios mínimos $\mathcal{F}$ para um conjunto $\mathbf{I}_\alpha(S)$ e um conjunto de sentinelas $\mathcal{G}$ para um conjunto delta $\Delta = \Delta(\mathbf{I}_\alpha(S))$, e produz um conjunto

de polinômios mínimos $\mathcal{F}^+$ para um conjunto $\mathbf{I}_{\alpha^+}(S)$ e um conjunto de sentinelas $\mathcal{G}^+$ para um conjunto delta $\Delta^+ = \Delta(\mathbf{I}_{\alpha^+}(S))$, sendo α^+ o índice de ordem imediatamente superior a α (ordenamento $<_{grev}$). Observe que a saída do algoritmo consiste apenas numa atualização da entrada.

Algoritmo 11 (Algoritmo BMS)

Entradas:

- *Um arranjo μ-dimensional S de elementos de $\mathbb{F}_q$;*
- *Um índice $\alpha \in \mathbb{Z}_+^\mu$;*
- *Um conjunto de polinômios mínimos $\mathcal{F}$ para $\mathbf{I}_\alpha(S)$;*
- *Um conjunto de sentinelas $\mathcal{G}$ para $\Delta(\mathbf{I}_\alpha(S))$, com suas extensões e discrepâncias.*

Saídas:

- *Um conjunto de polinômios mínimos $\mathcal{F}^+$ para $\mathbf{I}_{\alpha^+}(S)$;*
- *Um conjunto de sentinelas $\mathcal{G}^+$ para $\Delta(\mathbf{I}_{\alpha^+}(S))$, com suas extensões e discrepâncias.*

<<< **Passo 1** >>>
Considere o conjunto $\mathcal{F}' = \{f \in \mathcal{F} : \deg(f) \leq \alpha^+\}$.
Para cada $f \in \mathcal{F}'$, *calcule o valor predito (*cf. *(3.15))*

$$P_{\alpha^+}(f) = \frac{-1}{\mathrm{lc}(f)} \sum_{\beta <_{grev} \alpha^+} f_{\deg(f)-\alpha^+ +\beta} S_\beta .$$

Considere o conjunto $\mathcal{N} = \{f \in \mathcal{F}' : P_{\alpha^+}(f) \neq S_{\alpha^+}\}$.

<<< **Passo 2** >>>
Faça $\mathcal{G}^+ = \mathcal{G} \cup \mathcal{N}$.
Para cada $f \in \mathcal{N}$, *calcule e armazene a extensão (*cf. *3.16))*

$$\mathrm{Span}(f) = \alpha^+ - \deg(f).$$

Faça $\Delta^+ = \Delta \cup \{\mathrm{Span}(f) : f \in \mathcal{N}\}$.
Para cada $f \in \mathcal{N}$, *calcule e armazene a discrepância (*cf. *(3.17))*

$$\delta_f = \mathrm{lc}(f)\left[S_{\alpha^+} - P_{\alpha^+}(f)\right].$$

<<< **Passo 3** >>>
Para cada $\beta \in \mathrm{Ext}\,\Delta^+$ *(cantos externos de* Δ^+*), proceda o seguinte:*

- *Primeiro, se existir um* $f \in \mathcal{F}\backslash\mathcal{N}$, *tal que* $\deg(f) = \beta$, *então faça*

$$h^{(\beta)}(x_1, \ldots, x_\mu) = f(x_1, \ldots, x_\mu);$$

  - *Em caso contrário, se* $\beta \nleq \alpha^+$, *tome um* $f \in \mathcal{N}$, *tal que* $\deg(f) \leq \beta$, *e faça*

$$h^{(\beta)}(x_1, \ldots, x_\mu) = x^{\beta - \deg(f)} f(x_1, \ldots, x_\mu);$$

– *Em último caso, tome um* $f \in \mathcal{N}$*, tal que* $\deg(f) \leq \beta$*, e um* $g \in \mathcal{G}$*, tal que* $\mathrm{Span}(g) \geq \alpha^+ - \beta$*. Considere os índices* $q = \beta - \deg(f)$ *e* $p = \mathrm{Span}(g) - \alpha^+ + \beta$*. Faça, então,*

$$h^{(\beta)}(x_1, \ldots, x_\mu) = x^q f(x_1, \ldots, x_\mu) - \frac{\delta_f}{\delta_g} x^p g(x_1, \ldots, x_\mu);$$

Por fim, tem-se que

$$\mathcal{F}^+ = \left\{ h^{(\beta)}(x_1, \ldots, x_\mu) : \beta \in \mathrm{Ext}\,\Delta^+ \right\}.$$

Prova. Para detalhes sobre a prova deste algoritmo, veja [58, 60]. ■

No passo 1 do algoritmo BMS (algoritmo 11), a validade dos polinômios de $\mathcal{F}$, que, por hipótese, correspondem a relações de recursão lineares μ-dimensionais válidas para todas as entradas do arranjo S até a entrada S_α, é testada para a próxima entrada S_{α^+}. Os polinômios inválidos para a entrada S_{α^+} podem ser usados como sentinelas, sendo armazenados no conjunto $\mathcal{N}$.

No passo 2, o conjunto de pontos excluídos $\Delta = \Delta(\mathbf{I}_\alpha(S))$ é atualizado usando os novos sentinelas contidos no conjunto $\mathcal{N}$. Observe que pode ocorrer de um ou mais $f \in \mathcal{N}$ serem sentinelas de pontos excluídos $\mathrm{Span}(f)$ que já pertençam ao conjunto Δ. Além disso, deve-se levar em consideração que ao acrescentar um novo ponto excluído γ ao conjunto Δ, deve-se certificar que todos os pontos $\beta \leq \gamma$ também pertençam ao conjunto atualizado Δ^+ (acrescentá-los se necessário), de modo que este conjunto Δ^+ também seja um conjunto delta, segundo a definição. Os valores de $\mathrm{Span}(f)$ e δ_f são armazenados para um possível uso posteriormente no passo 3.

O passo 3 consiste na determinação do conjunto atualizado $\mathcal{F}^+$ de polinômios válidos para o arranjo S até a entrada S_{α^+}. Este conjunto $\mathcal{F}^+$ é um conjunto de polinômios mínimos para $\mathbf{I}_{\alpha^+}(S)$ e $\mathcal{G}^+$ é um conjunto de sentinelas para $\Delta(\mathbf{I}_{\alpha^+}(S))$.

3.2.4 Decisão por maioria

No corpo $\mathbb{C}$ dos números complexos, a transformada de Fourier discreta de um vetor $\vec{u} = [u_0 \ \cdots \ u_{n-1}]$ de números complexos é um vetor $\vec{U} = [U_0 \ \cdots \ U_{n-1}]$, em que

$$U_k = \sum_{i=0}^{n-1} e^{-j\frac{2\pi}{n}ki} u_i, \ k = 0, \ldots, n-1.$$

Observe que o termo $e^{-j\frac{2\pi}{n}}$ na equação acima constitui uma raiz n-ésima da unidade em $\mathbb{C}$, ou seja, $\left(e^{-j\frac{2\pi}{n}}\right)^n = 1$. Num corpo finito $\mathbb{F}_q$, um elemento α de ordem n também constitui uma raiz n-ésima da unidade. Por analogia, define-se a transformada de Fourier de um vetor $\vec{v} = [v_1 \ \cdots \ v_n]$ em um corpo finito $\mathbb{F}_q$ como sendo o vetor $\vec{V} = [V_1 \ \cdots \ V_n]$, em que

$$V_j = \sum_{i=1}^{n} \alpha^{ij} v_i, \ j = 1, \ldots, n, \tag{3.18}$$

sendo $\alpha \in \mathbb{F}_q$ um elemento de ordem n. O vetor $\vec{v}$ pode ser obtido pela transformada inversa dada por

$$v_i = \frac{1}{n} \sum_{j=1}^{n} \alpha^{-ij} V_j,\ i = 1, \ldots, n,$$

em que $n \equiv 0 \bmod p$, sendo p um inteiro primo e $q = p^m$, para algum inteiro m.

Observe que (3.18) equivale à expressão das síndromes de uma palavra recebida $\vec{v}$ para um código em uma variável (códigos BCH, de Reed-Solomon, etc.). De um modo geral, a transformada de Fourier de um vetor recebido $\vec{v} = \vec{c} + \vec{e}$, em que $\vec{e}$ é um vetor erro, para um código (n, k) qualquer, pode ser vista como o conjunto das síndromes (*cf.* (3.1))

$$S_i(\vec{v}) = S_i(\vec{e}) = \vec{h}_i \vec{e}^T,\ i = 1, \ldots, n,$$

em que $\vec{h}_i$ são as linhas da matriz de paridade do código. O importante neste fato é que, uma vez conhecidas todas as n síndromes de $\vec{v}$ (apenas $n - k$ síndromes são conhecidas a princípio), é possível determinar o vetor $\vec{e}$ ocorrido através da transformada inversa de Fourier.

O esquema de decisão por maioria proposto por Feng e Rao [21] permite se obter recursivamente as k síndromes desconhecidas, possibilitando, assim, a decodificação de $\vec{v}$. Na presente subseção, o esquema de decisão por maioria será descrito numa forma semelhante à apresentada por Feng e Rao, o que envolve o conceito de anti-lacunas apresentado na subseção C.5 do capítulo 3.

Anti-lacunas

Considere uma curva algébrica plana $\mathcal{X}$ de gênero g sobre um corpo $\mathbb{F}_q$. Considere o divisor $D = P_1 + \cdots + P_n$ de pontos racionais de $\mathcal{X}$ e o ponto racional Q também de $\mathcal{X}$, mas disjunto do suporte de D.

Denote por C_{m_k} o código AG $C^*(D, m_k Q)$ de comprimento n e dimensão $n - m_k + g - 1$. Assuma $m_k > 2g$ e considere $k = m_k - g + 1$ a dimensão do código dual.

Denote por $(m_i : i \in \mathbb{N})$ a seqüência das anti-lacunas de Q, em que

$$0 < m_1 < \cdots < m_{g-1} < 2g$$

e $m_i = i + g$, para $i = g, g + 1, \ldots, m_k - g$.

Denote por g_i uma função racional que possui um pólo de ordem m_i no ponto Q, e nenhum outro pólo mais. Tem-se que $g_1, \ldots, g_k$ constitui uma base para o espaço $L(m_k Q)$ ortogonal ao código C_{m_k}.

Matriz de síndromes bi-dimensionais

Se $\vec{e}$ é um vetor erro ocorrido em uma palavra recebida, defina S como a *matriz de síndromes bi-dimensionais* correspondentes a $\vec{e}$ dada por

$$S = \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,k} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ S_{k,1} & S_{k,2} & \cdots & S_{k,k} \end{bmatrix},$$

cujos elementos são dados por

$$S_{i,j}\left(\vec{e}\right) = \sum_{l=1}^{t} e_{u_l} g_i\left(P_{u_l}\right) g_j\left(P_{u_l}\right),$$

em que os u_ls são as posições dos t erros ocorridos. Se $\vec{v} = \vec{c} + \vec{e}$ é uma palavra recebida, para $\vec{c} \in C_{m_k}$, e $m_i + m_j = m_p \leq m_k$, então $g_i g_j \in L\left(m_p Q\right) \subseteq L\left(m_k Q\right)$ e $S_{i,j}\left(\vec{e}\right) = S_{i,j}\left(\vec{v}\right)$. Portanto, $S_{i,j}$ é uma entrada conhecida da matriz S, se $m_i + m_j \leq m_k$.

Defina o conjunto de pares N_k por

$$N_k = \left\{(i,j) \in \mathbb{N}^2 : m_i + m_j = m_{k+1}\right\}$$

e denote por n_k seu número de elementos. As entradas da matriz S com índices $(i,j) \in N_k$ são as primeiras síndromes desconhecidas com relação ao código C_{m_k} a serem determinadas. Uma vez determinada uma entrada $S_{i,j}$, com $(i,j) \in N_k$, então todas as outras entradas $S_{i',j'}$, com $(i',j') \in N_k$, que não são necessariamente iguais, podem ser obtidas. Isto, porque cada uma das funções $g_i g_j$, $g_{i'} g_{j'}$ e g_{k+1} gera o espaço $L\left(m_{k+1} Q\right) \setminus L\left(m_k Q\right)$. Ou seja, existem elementos $\lambda_{i,j}, \lambda_{i,j,r} \in \mathbb{F}_q$, com $\lambda_{i,j} \neq 0$, tais que

$$\begin{aligned} g_i g_j &= \lambda_{i,j} g_{k+1} + \sum_{r \leq k} \lambda_{i,j,r} g_r \Rightarrow \\ S_{i,j} &= \lambda_{i,j} S_{k+1} + \sum_{r \leq k} \lambda_{i,j,r} S_r, \end{aligned}$$

para todo $(i,j) \in N_k$. Além disso, esta relação é a mesma para todos os vetores erro.

Considere agora a matriz $S\left(i,j\right)$, dada por

$$S\left(i,j\right) = \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,j-1} & S_{1,j} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,j-1} & S_{2,j} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ S_{i-1,1} & S_{i-1,2} & \cdots & S_{i-1,j-1} & S_{i-1,j} \\ S_{i,1} & S_{i,2} & \cdots & S_{i,j-1} & S_{i,j} \end{bmatrix}. \tag{3.19}$$

Se $m_i + m_j = m_{k+1}$, então todos os elementos de $S(i,j)$ são conhecidos, exceto $S_{i,j}$. Se $m_i + m_j = m_k$, então $S(i,j)$ equivale à matriz de síndromes determinada no passo 2 do algoritmo básico (algoritmo 10) para o código C_{m_k} (m_k é o parâmetro a naquele algoritmo).

Candidatos e discrepâncias

Considere $(i,j) \in N_k$, ou seja, $m_i + m_j = m_{k+1}$.

Definição 12 (Candidato) *Se as matrizes $S(i-1,j-1)$, $S(i-1,j)$ e $S(i,j-1)$ possuem o mesmo posto, então (i,j) é dito ser um* candidato *com relação ao código C_{m_k}.*

Se (i,j) é um candidato, então existe um único valor $S'_{i,j}$ a ser atribuído à entrada desconhecida $S_{i,j}$, de modo que as matrizes $S(i-1,j-1)$ e $S(i,j)$ possuam o mesmo posto. O elemento $S'_{i,j}$ é dito ser um *valor candidato* ou *predito* da síndrome desconhecida $S_{i,j}$.

Denote o número de candidatos *verdadeiros* ou *corretos*, em que $S'_{i,j} = S_{i,j}$, por T e o número de candidatos *falsos* ou *incorretos*, em que $S'_{i,j} \neq S_{i,j}$, por F.

Definição 13 (Discrepância) *Um índice (i,j) é dito ser uma* discrepância *(não confundir com o conceito de discrepância usado na descrição do algoritmo BMS na subseção 3.2.3), se as matrizes $S(i-1,j-1)$, $S(i-1,j)$ e $S(i,j-1)$ possuem um mesmo posto, que difere do posto de $S(i,j)$.*

Se for aplicado o algoritmo de eliminação de Gauss sem troca de linhas ou colunas à matriz de síndromes bi-dimensionais S, as discrepâncias serão os pivôs da matriz resultante. Portanto, o número total de discrepâncias, denotado por DT, é igual ao posto de S. Além disso, a matriz S pode ser escrita na forma

$$S = XYX^T,$$

em que

$$X = \begin{bmatrix} g_1(P_{u_1}) & g_1(P_{u_2}) & \cdots & g_1(P_{u_t}) \\ g_2(P_{u_1}) & g_2(P_{u_2}) & \cdots & g_2(P_{u_t}) \\ \vdots & \vdots & \ddots & \vdots \\ g_k(P_{u_1}) & g_k(P_{u_2}) & \cdots & g_k(P_{u_t}) \end{bmatrix}$$

e

$$Y = \begin{bmatrix} e_{u_1} & 0 & \cdots & 0 \\ 0 & e_{u_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_{u_t} \end{bmatrix}.$$

Portanto, o número total de discrepâncias é, no máximo, igual ao número de erros ocorridos t.

Tomada de decisão

Considere $\vec{v} = \vec{c} + \vec{e}$ uma palavra recebida com $t \leq \frac{n_r - 1}{2}$ erros com relação ao código C_{m_k}. Então, todas as síndromes bi-dimensionais $S_{i,j}$, com $m_i + m_j \leq m_k$, são conhecidas e as demais síndromes são desconhecidas.

Denote o número de discrepâncias conhecidas por K. Um candidato é incorreto se e só se for uma discrepância. Então,

$$K + F \leq DT \leq t. \tag{3.20}$$

Se (i, j) é uma discrepância conhecida, então todas as entradas (i, j') e (i', j), com $j' > j$ e $i' < i$, não são candidatos. Se $(i, j) \in N_k$ não é um candidato, então existe pelo menos uma discrepância conhecida na mesma linha i ou coluna j. Portanto, o número de pares $(i, j) \in N_k$ que não são candidatos é, no máximo, $2K$.

O número de pares $(i, j) \in N_k$ que são candidatos é igual a $T + F$. Portanto,

$$n_r = \text{n}^\text{o} \text{ de candidatos} + \text{n}^\text{o} \text{ de não candidatos} \leq (T + F) + 2K. \tag{3.21}$$

Como, por hipótese, $w(\vec{e}) = t \leq \frac{n_r - 1}{2}$, tem-se de (3.20) e (3.21) que

$$\begin{gathered} K + F \leq \frac{n_r - 1}{2} \Rightarrow \\ 2K + 2F + 1 \leq n_r \leq T + F + 2K \Rightarrow \\ F < T. \end{gathered}$$

Não há uma forma direta de determinar se um candidato é ou não verdadeiro. Contudo, se for atribuído um valor predito a cada um dos candidatos, então a maioria, T candidatos, terá o mesmo valor, que é, por definição, o valor correto de S_{r+1}.

Esta presente descrição teve por objetivo introduzir a idéia do esquema de decisão por maioria de Feng e Rao. Um algoritmo de decodificação com decisão por maioria na forma apresentada aqui tem uma complexidade $\mathcal{O}(n^3)$, no entanto algoritmos que associam o algoritmo BMS ao esquema de decisão por maioria apresentam complexidades menores, da ordem de $\mathcal{O}\left(n^{\frac{7}{3}}\right)$.

3.2.5 Algoritmo de Porter (segunda abordagem)

A segunda abordagem utilizada no desenvolvimento de esquemas de decodificação para códigos AG é aquela em que as síndromes são definidas como elementos em um anel afim.

O algoritmo apresentado por Porter, Shen e Pellikaan é um exemplo típico desta segunda abordagem [54].

A descrição do algoritmo de Porter, Shen e Pellikaan a seguir será feita de forma simplificada, objetivando apenas transmitir a idéia do processo de decodificação de códigos AG através desta segunda abordagem. O algoritmo de O'Sullivan, para o qual propomos uma arquitetura de implementação e que será discutido em detalhes no capítulo 5, utiliza uma variação deste tipo de abordagem.

O algoritmo de decodificação de Porter pode ser visto como uma generalização da solução da equação chave para códigos de Goppa clássicos pelo algoritmo de Euclides em um anel de polinômios em uma única variável. No caso dos códigos AG, ao invés de um anel de polinômios em uma variável, tem-se o anel de funções racionais em uma curva, denotado por $K_\infty(P)$, definido a seguir.

Anel afim $K_\infty(P)$

Considere uma curva projetiva, não singular e irredutível $\mathcal{X}$ de gênero g, definida sobre o corpo $\mathbb{F}_q$. Considere $\mathbb{F}_q(\mathcal{X})$ o corpo das funções racionais em $\mathcal{X}$. Considere $P, P_1, \ldots, P_n$ pontos racionais (equivalentes a lugares de grau 1) de $\mathcal{X}$[8], e D o divisor $P_1 + \cdots + P_n$. Considere também o divisor G, cujo suporte é disjunto de $\{P_1, \ldots, P_n\}$.

Defina, então, o anel afim $K_\infty(P)$ com relação ao ponto (lugar) P pela equação

$$K_\infty(P) = \{f \in \mathbb{F}_q(\mathcal{X}) : \sup((f)_\infty) \subseteq \{P\}\},$$

ou seja, o conjunto das funções racionais em $\mathcal{X}$ que possuem pólos exclusivamente em P.

Defina o *grau de uma função* $f \in K_\infty(P)$ por

$$\deg(f) = -v_P(f),$$

em que $v_P(f)$ é a função de avaliação discreta de f em P. Observe que, se $f, g \in K_\infty(P)$, então

$$\deg(fg) = \deg(f) + \deg(g),$$

e

$$\deg(f+g) \leq \max[\deg(f), \deg(g)].$$

Também, se $\deg(f) = \deg(g)$, então existe um $\lambda \in \mathbb{F}_q^*$, tal que $\deg(f - \lambda g) < \deg(f)$.

[8]Em todos os algoritmos de decodificação para códigos AG que utilizam esta segunda abordagem, é necessário reservar um dos pontos racionais da curva para a definição de um divisor extra E.

Isometria de códigos

Considere um código linear C. Se $\vec{c} = [x_1 \ \cdots \ x_n]$ é uma palavra código de C, defina $\sigma\vec{c} = \left[x_{\sigma(1)} \ \cdots \ x_{\sigma(n)}\right]$ como sendo uma permutação das posições da palavra $\vec{c}$, e $\sigma C = \{\sigma\vec{c} : \vec{c} \in C\}$. Dois códigos lineares C_1 e C_2 em $\mathbb{F}_q^n$ são ditos equivalentes se $C_1 = \sigma C_2$, para alguma permutação σ. Considere $\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{F}_q^n$ uma n-úpla não nula. Defina, então, $\lambda\vec{c} = [\lambda_1 x_1 \ \cdots \ \lambda_n x_n]$ e $\lambda C = \{\lambda\vec{c} : \vec{c} \in C\}$. Dois códigos lineares C_1 e C_2 em $\mathbb{F}_q^n$ são ditos *isométricos* se existir uma n-úpla λ de elementos não nulos de $\mathbb{F}_q$ e uma permutação σ, tais que $C_1 = \lambda\sigma C_2$. Pode-se ver que este mapeamento $\lambda\sigma$ mantém a métrica de Hamming do código invariante.

Considere C_1 e C_2 dois códigos isométricos em $\mathbb{F}_q^n$, com $C_1 = \lambda\sigma C_2$. Suponha que $A(C_2)$ é um algoritmo de decodificação de C_2 que corrige até t erros. O seguinte procedimento, chamado *algoritmo de decodificação induzido* $\lambda\sigma A(C_2)$, corrige o código C_1 também até t erros:

1. Entrada: $\vec{v}$;
2. $\vec{u} = \sigma^{-1}\left[\frac{x_1}{\lambda_1} \ \ldots \ \frac{x_n}{\lambda_n}\right]$;
3. Execute $A(C_2)$ com o vetor de entrada $\vec{u}$ para obter $\vec{c}' \in C_2$;
4. Saída: $\vec{c} = \lambda\sigma\vec{c}'$.

Portanto, uma vez que um decodificador para um dos códigos de uma classe de isometria é dado, então todos os decodificadores dos códigos desta classe são obtidos através de algoritmos induzidos.

Se m é um inteiro, então existe uma função $h \in K_\infty(P)$ e um inteiro positivo μ, tais que os códigos AG $C^*(D, mP)$ e $C^*(D, (h)_0 - \mu P)$ são isométricos.

Neste processo de decodificação, sem perda de generalidade, portanto, será considerado o código $C^*(D, G)$, em que $G = E - \mu P$ e E é um divisor efetivo.

Resíduos de diferenciais

Considere P um ponto racional de $\mathcal{X}$ disjunto de $\{P_1, \ldots, P_n\}$. Considere E um divisor efetivo e μ um inteiro positivo, tais que E e $D = P_1 + \cdots + P_n$ possuem suportes disjuntos e $\deg(E - \mu P) \geq 2g - 1$ (teorema de Riemman-Roch).

Mostra-se que existem sempre n diferenciais $\varepsilon_1, \ldots, \varepsilon_n \in \Omega(-D - \mu P)$ (espaço de diferenciais gerado pelo divisor $-D - \mu P$) independentes módulo $\Omega(-\mu P)$, tais que $\mathrm{Res}_{P_i}(\varepsilon_j) = 1$, se $i = j$, e $\mathrm{Res}_{P_i}(\varepsilon_j) = 0$, se $i \neq j$ [54]. Se, além disso, $\mu = 1$, então $(\varepsilon_i)_\infty = P_i + P$, para $1 \leq i \leq n$.

Mostra-se também que, para todo diferencial $\omega \in \Omega(E - \mu P - D)$, [54]

$$\omega = \sum_{j=1}^{n} \mathrm{Res}_{P_j}(\omega)\,\varepsilon_j.$$

Defina

$$\varepsilon(\vec{c}) = \sum_{i} c_i \varepsilon_i.$$

Este mapeamento $\varepsilon : \mathbb{F}_q^n \to \Omega_{\mathcal{X}}$ ($\Omega_{\mathcal{X}}$ é o espaço de diferenciais associados à curva $\mathcal{X}$) é, na verdade, um mapeamento inverso de Res_D, uma vez que $\mathrm{Res}_D(\varepsilon(\vec{c})) = \vec{c}$. Além disso, $\varepsilon(\vec{c}) \in \Omega(E - \mu P - D)$ se e só se $\vec{c} \in C^*(D, E - \mu P)$.

As síndromes

Segundo a primeira abordagem na decodificação de códigos AG, as síndromes de uma palavra recebida $\vec{v} \in \mathbb{F}_q^n$ são definidas por um mapeamento S do espaço de funções $L(G)$ para o corpo $\mathbb{F}_q$. Este mapeamento é dado por

$$S(\vec{v}, f) = \sum_{i=1}^{n} v_i f(P_i),$$

em que $f \in L(G)$.

Nesta segunda abordagem, a síndrome de uma palavra recebida $\vec{v} \in \mathbb{F}_q^n$ é definida como um elemento do anel afim $K_\infty(P)$, que consiste numa generalização das síndromes dos códigos de Goppa clássicos. A definição das síndromes que será apresentada é valida para códigos da forma $\vec{C}^*(D, E - \mu P)$, o que não constitui uma restrição, uma vez que qualquer código AG é isométrico a algum código deste tipo.

Suponha que $E = (h)_0$ (divisor dos zeros de h), com $h \in K_\infty(P)$, em que h não possui zeros em qualquer dos pontos $P_1, \ldots, P_n$ de $\mathcal{X}$.

Mostra-se que existe sempre um diferencial η, tal que [54]

$$\begin{aligned} &\sup((\eta)_0) \subseteq \{P\} \Rightarrow \\ &\qquad (\eta)_0 = lP \end{aligned} \tag{3.22}$$

e

$$\sup((\eta)) \cap (\{P_1, \ldots, P_n\} \cup \sup(E)) = \varnothing.$$

Se $\mathcal{X}$ é uma curva de gênero $g > 1$, então $l > 0$.

Definição 14 (Síndrome) *A síndrome de uma palavra recebida $\vec{v} \in \mathbb{F}_q^n$ para um código $C^*(D, E - \mu P)$ é definida como o mapeamento linear $S : \mathbb{F}_q^n \to \mathbb{F}_q(\mathcal{X})$ dado por*

$$S(\vec{v})\,\eta = \sum_{i=1}^{n} v_i \frac{h(P_i) - h}{h(P_i)} \varepsilon_i.$$

O nome síndrome para este mapeamento S é justificado pelo fato de que, se $E = (h)_0$, então

$$\vec{v} \in C^* (D, E - \mu P) \Leftrightarrow S(\vec{v}) \equiv 0 \mod h.$$

Como já foi afirmado, esta síndrome $S(\vec{v})$ constitui um elemento de $K_\infty (P)$.

Decodificação pela solução da equação chave

Por simplicidade, assuma que o diferencial η é tal que $(\eta) = (2g-2) P$.

Considere W um divisor em $\mathcal{X}$, tal que o ponto P não pertence ao suporte de W. Defina o ideal $K_\infty (P, W)$ por

$$K_\infty (P, W) = \{f \in K_\infty (P) : f = 0 \text{ ou } v_Q (f) \geq n_Q (W)\},$$

em que $n_Q (W)$ é o coeficiente do ponto Q no divisor W.

Considere $E = (h)_0$ (suporte disjunto de $\{P_1, \ldots, P_n\}$). Dada a síndrome $S(\vec{v})$ da palavra recebida $\vec{v} \in \mathbb{F}_q^n$, a decodificação de $\vec{v}$ é feita pela solução da *equação chave*

$$\begin{aligned} &fS(\vec{v}) \equiv r \mod h \Rightarrow \\ &fS(\vec{v}) = r + qh, \end{aligned} \tag{3.23}$$

em que $f \in K_\infty (P)$, $r, q \in K_\infty \left(P, (\eta)_{\in fty}\right)$ e $\deg (r) \leq \deg (f) + 2g - 2 + \mu$. O par (f, r) é dito ser uma *solução válida* para a equação chave. Uma solução válida (f, r) é dita ser *mínima* se $\deg (f)$ for o menor entre os graus de todas as funções f', tais que (f', r') também é uma solução válida.

Defina agora o *defeito de Clifford* σ do par (E, P) pela equação

$$\sigma = \max \left\{ \frac{\deg (E - kP)}{2} - (l (E - kP) - 1) : k \in \mathbb{N} \right\}.$$

Mostra-se que $\sigma \leq \frac{g}{2}$ [67].

Segue, então, o chamado teorema da decodificação.

Teorema 15 (Decodificação) *Considere a palavra recebida $\vec{v} = \vec{c} + \vec{e}$, em que $\vec{c} \in C^* (D, E - \mu P)$ é uma palavra código e $\vec{e} \in \mathbb{F}_q^n$ é um vetor de erros ocorridos. Tem-se que:*

1. *(Existência) Existe uma solução válida (f, r) para a equação chave de $\vec{v}$ (*cf. *(3.23)), tal que*

$$\frac{r}{f}\eta \in \Omega (-D - \mu P) \text{ e } \vec{e} = \mathrm{Res}_D \left(\frac{r}{f}\eta\right);$$

2. *(Unicidade) Considere a ocorrência de até* $t = \frac{d-1}{2} - \sigma$ *erros, em que* d *é a distância mínima projetada do código e* σ *o defeito de Clifford. Se* (f, r) *é uma solução válida mínima da equação chave de* $\vec{v}$*, então*

$$\frac{r}{f}\eta \in \Omega\left(-D - \mu P\right) \;\; e \;\; \vec{e} = \mathrm{Res}_D\left(\frac{r}{f}\eta\right).$$

Este teorema estabelece que o problema da decodificação resume-se à obtenção de uma solução válida para a equação chave (*cf.* (3.23)). Esta solução para a equação chave pode ser obtida através do algoritmo proposto por Ba-Zhong Shen (não descrito aqui), que utiliza uma *seqüência de sub-resultantes* e constitui uma generalização do algoritmo de Euclides. A utilização deste algoritmo de Shen associado ao algoritmo de Porter permite a decodificação dos $\frac{d-1}{2} - \sigma$ erros com uma complexidade $\mathcal{O}\left(n^3\right)$.

Ressalta-se que a presente descrição do algoritmo de Porter não teve como objetivo proporcionar uma completa compreensão deste algoritmo, mas apenas ilustrar a idéia existente em torno desta segunda abordagem na decodificação dos códigos AG. Maiores detalhes podem ser obtidos em Porter, Shen e Pellikaan [54].

3.2.6 Algoritmo de Shokrollahi e Wasserman (terceira abordagem)

A terceira abordagem utilizada no desenvolvimento de esquemas de decodificação para códigos AG é a que envolve a chamada decodificação de lista, proposta inicialmente na década de 1950 por Elias [20] e Wozencraft [74] e estendida aos códigos AG em 1999 por Shokrollahi e Wasserman [66]. Nela, ao invés de se buscar a palavra código única correspondente ao vetor recebido situada dentro da esfera de decodificação de raio $t = \frac{d-1}{2}$, sendo d a distância mínima do código e t sua capacidade de correção, considera-se a possibilidade de que ocorra um número $t^{'}$ de erros maior que t e busca-se o conjunto de palavras código contidas dentro da esfera de raio $t^{'}$.

A presente seção apresenta de forma sucinta o algoritmo de decodificação de lista proposto por Shokrollahi e Wasserman. Maiores detalhes sobre o algoritmo e provas dos resultados aqui descritos podem ser obtidas em Shokrollahi e Wasserman [66].

Usualmente, se o número de erros inseridos no vetor recebido durante a transmissão for maior que $\frac{d-1}{2}$, então a decodificação da palavra código única geralmente não é possível. Entretanto, para uma quantidade limitada e de erros, é possível se construir um algoritmo que forneça uma lista com todas as palavras código cujas distâncias de Hamming do vetor recebido sejam menores ou iguais a e. A este tipo de decodificação dá-se o nome de *decodificação de lista*.

Definição 16 *Um código de bloco linear C de comprimento n definido sobre $\mathbb{F}_q$ é dito (e, b)-decodificável se toda esfera de Hamming de raio e em $\mathbb{F}_q^n$ contiver no máximo b palavras código.*

Portanto, um código (e, b)-decodificável permite uma decodificação de lista com listas de tamanho no máximo b. Um mesmo código pode ser (e, b)-decodificável para diferentes valores de e e b. Para dar dois exemplos extremos, observe que qualquer código definido sobre $\mathbb{F}_q$ de comprimento n, dimensão k e distância mínima d (um código (n, k, d)) é $(\lfloor \frac{d-1}{2} \rfloor, 1)$-decodificável, assim como também (n, q^k)-decodificável.

Considere $\mathcal{X}$ uma curva algébrica de gênero g definida sobre $\mathbb{F}_q$, e $\mathbb{F}_q(\mathcal{X})$ seu corpo de funções.

Considere $P_1, P_2, \ldots, P_n$ pontos racionais da curva $\mathcal{X}$, e G um divisor de $\mathcal{X}$ cujo suporte seja disjunto do suporte do divisor $D = P_1 + P_2 + \cdots + P_n$. Denote por α o grau do divisor G e considere $\alpha < n$.

Considere o espaço de funções $L(G)$ do divisor G. O teorema de Riemann (seção C.4) estabelece que a dimensão $l(G)$ do espaço $L(G)$ sobre $\mathbb{F}_q$ é finita e limitada inferiormente por $\alpha - g + 1$.

Os códigos AG utilizados no algoritmo descrito aqui são os construídos por funções (*cf.* seção 2.1.1). Neste caso, os códigos, denotados por $C(D, G)$, são obtidos pela avaliação das funções de $L(G)$ nos pontos $P_1, P_2, \ldots, P_n$ de $\mathcal{X}$. Uma palavra código, portanto, consiste num vetor $(f(P_1), f(P_2), \ldots, f(P_n))$, sendo f uma função de $L(G)$. Observe que, comumente, os códigos AG utilizados nos diversos algoritmos de decodificação propostos na literatura são aqueles construídos por diferenciais (*cf.* seção 2.1.1), diferentemente do que ocorre neste caso. Esta diferença, entretanto, não é essencial, uma vez que qualquer código AG pode ser definido das duas maneiras.

Pela aplicação do teorema de Riemann, prova-se que $C(D, G)$ é um código (n, k, d), em que $k \geq \alpha - g + 1$ e $d \geq n - \alpha$. O valor $d^* = n - \alpha$ é a distância mínima projetada de $C(D, G)$.

O algoritmo 18 de decodificação de lista é resultado do teorema que segue.

Teorema 17 *Considere C um código AG de comprimento n e dimensão k definido por uma curva algébrica $\mathcal{X}$ de gênero g sobre $\mathbb{F}_q$. Então, para um inteiro positivo b, C é um código $(n - \beta - 1, b)$-decodificável, sendo $\beta = \left\lceil \frac{n+1}{b+1} + \frac{b\alpha}{2} + g - 1 \right\rceil$ e $\alpha = k + g - 1$.*

Prova. Veja [66]. ■

Algoritmo 18 (Decodificador de lista)

Entradas:

- *Um vetor recebido* $\vec{v} = [y_1, y_2 \ \cdots \ y_n]$;
- *Um divisor* F *de grau* $\beta - b\alpha = \left\lceil \frac{n+1}{b+1} + \frac{b\alpha}{2} + g - 1 \right\rceil$, *cujo suporte seja disjunto de* D.

Saídas:

- *Uma lista de* b *palavras código* $\vec{x}$ *com distância de Hamming no máximo* $n-\beta-1$ *do vetor recebido* $\vec{v}$.

<<< **Passo 1 – Interpolação** >>>
Encontrar um polinômio diferente de zero

$$H(T) = u_b T^b + \cdots + u_1 T + u_0 \in \mathbb{F}_q(\mathcal{X})[T],$$

em que $u_i \in L(F + (b-j)G)$, *tal que*

$$H(P_i, y_i) = \sum_{j=0}^{b} u_j(P_i) y_i^j$$

é zero para $i = 1, \ldots, n$.

<<< **Passo 2 – Fatoração** >>>
Encontrar todas as raízes ρ *de* $H(T)$ *em* $\mathbb{F}_q(\mathcal{X})[T]$.
Para cada ρ *encontrada, calcular o vetor*

$$x_\rho = [\rho(P_1) \ \cdots \ \rho(P_n)].$$

Se x_ρ *é não definido ou se a distância entre* x_ρ *e o vetor recebido* $\vec{v}$ *for maior que* $n - \beta - 1$, *descartar* x_ρ. *Caso contrário, acrescentar* x_ρ *à lista de palavras código de saída.*

Prova. Para detalhes sobre a prova deste algoritmo, veja [66]. ■

Teorema 19 *Considere* C *um código AG de comprimento* n *e dimensão* k *definido por uma curva algébrica* $\mathcal{X}$ *de gênero* g. *Considere* $\alpha = k + g - 1$ *e* $\beta = \left\lceil \sqrt{2\alpha n} + g - 1 \right\rceil$. *Então,* C *é* $\left(n - \beta - 1, \left\lceil \sqrt{2n/\alpha} \right\rceil\right)$*-decodificável.*

Prova. Veja [66]. ■

O passo 1 do algoritmo 18 consiste basicamente da solução de um sistema de equações lineares. A existência de um polinômio não trivial $H(T)$ no algoritmo segue do fato de que um sistema de equações homogêneo com mais variáveis desconhecidas que equações possui sempre uma solução não trivial.

A complexidade do algoritmo 18 do decodificador de lista é determinada pelo passo 2. A principal tarefa do algoritmo 18 consiste na determinação das raízes em $\mathbb{F}_q(\mathcal{X})$ de $H(T) \in \mathbb{F}_q(\mathcal{X})[T]$. Para completar o decodificador de lista, portanto, é necessário descrever um algoritmo que determine estas raízes. Shokrollahi e Wasserman propõem um algoritmo para fatorar completamente $H(T)$ [66]. Este algoritmo de fatoração, no entanto, não será descrito aqui.

3.2.7 Algoritmo GMD (quarta abordagem)

A quarta abordagem utilizada no desenvolvimento de esquemas de decodificação para códigos AG é a que envolve a utilização de decisão suave. O método padrão de decodificação a decisão suave para códigos de bloco é o algoritmo de distância mínima generalizada ("generalized minimum-distance" – GMD) proposto por G. D. Forney Jr. em 1966.

O algoritmo GMD consiste num esquema genérico que pode ser aplicado a qualquer código de bloco linear. Diversas publicações têm apresentado derivações mais eficientes do decodificador GMD para códigos RS e códigos AG [5,34,36,52]. Na presente subseção, será descrito o algoritmo GMD, que é a base para este tipo de abordagem.

Decodificação a decisão suave

Um sistema de comunicação digital possui no centro de sua topologia um canal analógico. Cabe ao modulador (demodulador) converter este canal analógico num canal digital a ser visto e acessado pelo codificador (decodificador) do sistema (após a codificação, uma palavra código ou uma seqüência de palavras código é mapeada em um conjunto de sinais analógicos pela modulação). Um sistema de codificação para controle de erros pode apresentar melhores resultados se houver algum tipo de iteração entre o codificador (decodificador) e o modulador (demodulador). Historicamente, o estudo dos códigos para controle de erros foi mantido separado do estudo da modulação digital. Aqui, no entanto, o estudo de ambos, modulação e codificação, é feito de forma conjunta. Em abordagens deste tipo, pode-se encarar a codificação para controle de erros como uma das ferramentas usadas no projeto de sistemas de sinalização eficiente para comunicações através de canais ruidosos contínuos no tempo.

Os algoritmos de decodificação a decisão suave, no caso o algoritmo GMD, são esquemas deste tipo que utilizam informações extras obtidas do demodulador. Para um determinado código, a decodificação a decisão suave apresenta um desempenho melhor que a decodificação com decisão brusca (do inglês *hard decision*). No entanto, estes sistemas são mais complexos, sendo úteis em geral apenas para códigos pequenos. Num caso extremo em que a complexidade do sistema possa ser desconsiderada, as tarefas de demodulação e decodificação podem ser feitas simultaneamente.

Algoritmo GMD

Nas descrições que seguem, serão abordados inicialmente os códigos binários. Em seguida, os resultados serão generalizados para códigos sobre corpos finitos em geral.

Considere que o demodulador receba do canal bits binários adicionados a um ruído gaussiano (canal gaussiano aditivo). O i-ésimo bit recebido é, por exemplo, mapeado pelo demodulador num número real $\tilde{v}_i \in [-1, 1]$, que representa o valor do i-ésimo bit e seu nível de confiabilidade. Representando $\tilde{v}_i$ na forma de um número binário com sinal, pode-se ter o bit de sinal representando o valor demodulado e os demais bits representando seu nível de confiabilidade. Há dois casos especiais a serem observados: se $\tilde{v}_i$ assume apenas os valores 1 e -1, então o demodulador é de decisão brusca e o decodificador corrige apenas erros; se $\tilde{v}_i$ assume apenas os valores 0, 1 e -1, então o demodulador é de decisão brusca com apagamento e o decodificador corrige erros e apagamentos. O algoritmo GMD permite que $\tilde{v}_i$ assuma qualquer valor entre -1 e 1, podendo-se usar qualquer regra de demodulação que mapeie $\tilde{v}_i$ neste intervalo.

Considere agora que o canal não introduza qualquer erro e que os bits demodulados apresentem nível máximo de confiabilidade. Então, para uma palavra código $\vec{c}_r$ transmitida, tem-se no demodulador o mapeamento

$$\tilde{\vec{c}}_{ri} = \begin{cases} -1, & \text{se } \vec{c}_{ri} = 0, \\ 1, & \text{se } \vec{c}_{ri} = 1. \end{cases}$$

Este vetor $\tilde{\vec{c}}_r$ é também chamado de palavra código.

A distância euclidiana d_e entre uma palavra recebida $\tilde{\vec{v}}$ e uma palavra código $\tilde{\vec{c}}_r$ é dada por

$$d_e^2(\tilde{\vec{v}}, \tilde{\vec{c}}_r) = \|\tilde{\vec{v}} - \tilde{\vec{c}}_r\|^2 = \sum_{i=0}^{n-1} (\tilde{v}_i - \tilde{c}_{ri})^2.$$

Reescreva agora esta equação em termos do produto interno

$$d_e^2(\tilde{\vec{v}}, \tilde{\vec{c}}_r) = \|\tilde{\vec{v}}\|^2 - 2\tilde{\vec{v}} \cdot \tilde{\vec{c}}_r + \|\tilde{\vec{c}}_r\|^2.$$

Observe que, para um $\tilde{\vec{v}}$ fixo, minimizar $d_e^2(\tilde{\vec{v}}, \tilde{\vec{c}}_r)$ sobre r equivale a maximizar o produto interno $\tilde{\vec{v}} \cdot \tilde{\vec{c}}_r$ sobre r, uma vez que $\|\tilde{\vec{v}}\|^2$ independe de r e $\|\tilde{\vec{c}}_r\|^2 = n$ para todo r. Minimizar a distância euclidiana é um conceito mais intuitivo, no entanto a tarefa de maximizar o produto interno é preferível para os cálculos.

Teorema 20 *Existe no máximo uma palavra código $\tilde{\vec{c}}_r$ em um código binário de comprimento n e distância mínima de Hamming d^*, tal que*

$$\tilde{\vec{v}} \cdot \tilde{\vec{c}}_r > n - d^*$$

sempre que as componentes de $\tilde{\vec{v}}$ pertencerem ao intervalo $[-1, 1]$.

Prova. Considere $\vec{\tilde{c}}_r$ uma palavra código que satisfaça a desigualdade acima, e considere $\vec{\tilde{c}}_{r'}$ qualquer outra palavra código. Então, $\vec{\tilde{c}}_{r'}$ difere de $\vec{\tilde{c}}_r$ em pelo menos d^* posições. Faça

$$S = \{i \mid \tilde{c}_{ri} \neq \tilde{c}_{r'i}\}.$$

Portanto,

$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r = \sum_{i \notin S} \tilde{v}_i \tilde{c}_{ri} + \sum_{i \in S} \tilde{v}_i \tilde{c}_{ri} = A_1 + A_2,$$
$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_{r'} = \sum_{i \notin S} \tilde{v}_i \tilde{c}_{r'i} + \sum_{i \in S} \tilde{v}_i \tilde{c}_{r'i} = A_1 - A_2.$$

No entanto,

$$A_1 = \sum_{i \notin S} \tilde{v}_i \tilde{c}_{ri} \leq n - d^*,$$

uma vez que esta soma possui no máximo $n - d^*$ termos, nenhum dos quais maior que 1. Portanto, como $\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*$, tem-se que $A_2 > 0$ e $A_1 - A_2 \leq n - d^*$, o que completa a prova. ■

O algoritmo GMD (algoritmo 21) fornece a palavra código que satisfaça o teorema 20 se ela existir, caso contrário declara falha de decodificação. Observe, no entanto, que não é prático determinar a palavra código pelo cálculo de todas os 2^k produtos internos. Segue aqui a idéia para reduzir a complexidade desta tarefa. Ela consiste em realizar um laço de processamento em torno de um decodificador de erros e apagamentos.

Algoritmo 21 (Algoritmo GMD)

Entradas:

- *Um vetor recebido $\vec{\tilde{v}}$ de comprimento n;*
- *Um nível de confiabilidade α associado a cada posição do vetor $\vec{\tilde{v}}$.*

Saídas:

- *Uma palavra código decodificada $\vec{\tilde{c}}_r$, ou uma indicação de falha de decodificação.*

<<< **Passo 1** >>>

Determinar as $d^* - 1$ componentes $\tilde{v}_i$ do vetor recebido $\vec{\tilde{v}}$ que apresentam os menores níveis de confiabilidade α, ordenadas segundo estes:

$$\alpha_{i_1} \leq \alpha_{i_2} \leq \alpha_{i_3} \leq \cdots \leq \alpha_{i_{d^*-1}}.$$

Fazer $l = 0$.

<<< **Passo 2** >>>

Introduzir apagamento nas l componentes do vetor recebido $\vec{\tilde{v}}$ de menores níveis de confiabilidade α, ou seja,

$$\tilde{v}_{i_1}, \tilde{v}_{i_2}, \ldots, \tilde{v}_{i_l}.$$

Executar o decodificador de erros e apagamentos para o vetor recebido $\vec{v}$ acrescido dos apagamentos.

<<< **Passo 3** >>>
Verificar se

$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*.$$

Se a desigualdade for verdadeira, concluir que a palavra $\vec{\tilde{c}}_r$ fornecida pelo decodificador de erros e apagamentos é a palavra código procurada e parar o algoritmo. Se a desigualdade for falsa ou se o decodificador de erros e apagamentos não fornecer uma palavra $\vec{\tilde{c}}_r$ decodificada, fazer

$$l = l + 2.$$

<<< **Passo 4** >>>
Verificar se

$$l > d^* - 1.$$

Se a desigualdade for verdadeira, concluir que ocorreu falha na decodificação e parar o algoritmo. Se a desigualdade for falsa, retornar ao passo 2.

Prova. Para maiores detalhes sobre a prova deste algoritmo, veja [6, pp. 464–473]. ■

Baseado nas informações de decisão suave, o algoritmo GMD (algoritmo 21) gera uma série de vetores $\vec{\tilde{v}}^{(l)}$ incluindo apagamentos ao vetor recebido $\vec{\tilde{v}}$. Para cada l, de 0 a $d^* - 1$, ele apaga as l componentes da palavra recebida para as quais o nível de confiabilidade é menor. O vetor $\vec{\tilde{v}}^{(l)}$ é, então, decodificado usando um decodificador de erros e apagamentos. Se o decodificador fornecer uma palavra código $\vec{\tilde{c}}_r$, realiza-se o teste $\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*$. Se este teste for satisfeito, então $\vec{\tilde{c}}_r$ é a palavra código decodificada. Caso contrário, o valor de l é incrementado em dois (isto porque um erro passível de correção equivale a dois apagamentos) e o laço do algoritmo é repetido enquanto $l \leq d^* - 1$. Caso nenhuma palavra código seja encontrada, um falha de decodificação é declarada.

Observe que a complexidade do algoritmo GMD é aproximadamente igual a $\frac{1}{2}(d^* - 1)$ vezes a complexidade do decodificador para erros e apagamentos usado para o mesmo código.

Os teoremas 22 e 24 descritos a seguir demonstram que o algoritmo 21 realmente fornece a palavra código única procurada caso ela exista.

Teorema 22 *Se $\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*$, então pelo menos um vetor $\vec{\tilde{v}}^{(l)}$ satisfaz a desigualdade*

$$\vec{\tilde{v}}^{(l)} \cdot \vec{\tilde{c}}_r > n - d^*,$$

para $l = 0, \ldots, d^* - 1$.

Prova. A prova deste teorema está contida na prova do teorema 24. ■

Considere agora o caso de códigos sobre quaisquer corpos finitos. Tanto o conceito de distância euclidiana quanto o de distância de Hamming são extensíveis aos sinais não binários. Para ambos os conceitos, o problema não binário é tratado como um problema binário apenas pela distinção entre os dois símbolos, se são iguais ou diferentes, desprezando-se seus valores. O algoritmo GMD para o caso não binário, portanto, é praticamente o mesmo do caso binário.

Num corpo $\mathbb{F}_q$, para cada símbolo $\tilde{v}_i$ recebido, o demodulador associa um nível de confiabilidade α_i, sendo $0 \leq \alpha_i \leq 1$. Um $\alpha_i = 1$ indica o mais alto nível de confiabilidade sobre o valor de $\tilde{v}_i$, enquanto um $\alpha_i = 0$ indica o menor nível de confiabilidade sobre o valor de $\tilde{v}_i$.

O produto interno entre dois vetores em $\mathbb{F}_q$ é definido por

$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r = \sum_{i=0}^{n-1} \alpha_i \delta(\tilde{v}_i, \tilde{c}_{ri}),$$

em que

$$\delta(\tilde{v}_i, \tilde{c}_{ri}) = \begin{cases} 1, & \tilde{v}_i = \tilde{c}_{ri}, \\ -1, & \tilde{v}_i \neq \tilde{c}_{ri}. \end{cases}$$

Segue, então, a versão para corpos finitos em geral do teorema 20.

Teorema 23 *Existe no máximo uma palavra código $\vec{\tilde{c}}_r$ em um código definido sobre $\mathbb{F}_q$ de comprimento n e distância mínima de Hamming d^*, tal que*

$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*.$$

Prova. A prova é similar à do teorema 20. ■

O teorema que segue completa a prova do algoritmo GMD.

Teorema 24 *Se $\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r > n - d^*$ para uma palavra código $\vec{\tilde{c}}_r$, então pelos menos um vetor $\vec{\tilde{v}}^{(l)}$ satisfaz a desigualdade*

$$\vec{\tilde{v}}^{(l)} \cdot \vec{\tilde{c}}_r > n - d^*,$$

para $l = 0, \ldots, d^* - 1$.

Prova. É suficiente provar o teorema 24 para l de 0 a n, já que a conclusão claramente não é possível para $l \geq d^*$.

Ordene os níveis de confiabilidade α_i segundo suas magnitudes

$$\alpha_{i_1} \leq \alpha_{i_2} \leq \ldots \leq \alpha_{i_n}$$

e faça

$$\begin{aligned}\lambda_0 &= \alpha_{i_1},\\ \lambda_1 &= \alpha_{i_2} - \alpha_{i_1},\\ \lambda_2 &= \alpha_{i_3} - \alpha_{i_2},\\ &\vdots\\ \lambda_n &= 1 - \alpha_{i_n}.\end{aligned}$$

Assim,

$$0 \leq \lambda_l \leq 1 \quad \text{e} \quad \sum_{l=0}^{n} \lambda_l = 1.$$

Observe que o vetor λ se comporta como uma distribuição de probabilidade. Além disso,

$$\sum_{l=0}^{l'-1} \lambda_l = \alpha_{il'},$$

e, então,

$$\vec{\tilde{v}} = \sum_{l=0}^{n} \lambda_l \vec{\tilde{v}}^{(l)}.$$

Suponha que $\vec{\tilde{v}}^{(l)} \cdot \vec{\tilde{c}}_r \leq n - d^*$, para todo l. Então,

$$\vec{\tilde{v}} \cdot \vec{\tilde{c}}_r = \sum_{l=0}^{n} \lambda_l \vec{\tilde{v}}^{(l)} \cdot \vec{\tilde{c}}_r \leq (n - d^*) \sum_{l=0}^{n} \lambda_l = n - d^*,$$

o que contradiz a hipótese inicial. Portanto, o teorema está provado ■

Decodificador GMD para códigos AG

Basicamente, a decodificação de códigos AG usando o algoritmo GMD é feita incorporando um decodificador de erros e apagamentos para estes códigos. Considere aqui um esquema apresentado por R. Kötter [34].

Considere um corpo $\mathbb{F}_q$ e o espaço vetorial $\mathbb{F}_q^n$ de todas as n-úplas $\vec{u} = [u_0\ u_1\ \cdots\ u_{n-1}]$ sobre $\mathbb{F}_q$. Considere um código linear $C \subset \mathbb{F}_q^n$ de dimensão $k(C)$ e distância mínima de Hamming $d(C)$. Considere também o código dual denotado por $C^{\perp}$. Denote a matriz geradora de C por G_C e sua matriz de paridade por H_C.

Dados dois vetores

$$\vec{u} = [u_0\ u_1\ \cdots\ u_{n-1}]$$

e

$$\vec{v} = [v_0\ v_1\ \cdots\ v_{n-1}],$$

defina seu vetor produto de componentes por

$$\vec{u} * \vec{v} = [u_0 v_0 \ u_1 v_1 \ \cdots \ u_{n-1} v_{n-1}] .$$

Para dois subespaços $U, V \subset \mathbb{F}_q^n$, denote por $U * V$ o conjunto dos vetores $\{\vec{u} * \vec{v} : \vec{u} \in U, \vec{v} \in V\}$.

A definição seguinte é elemento central na solução do problema da localização dos erros.

Definição 25 (Par localizador de t erros) *Considere U, V e C códigos lineares de comprimento n sobre $\mathbb{F}_q$. Denota-se por (U, V) um par localizador de t erros para um código C se as seguintes condições forem válidas:*

$$C * U \subseteq V \tag{3.24}$$

$$k(U) > t \tag{3.25}$$

$$d(V) > t. \tag{3.26}$$

O primeiro objetivo deste algoritmo proposto por Kötter é encontrar um vetor $\vec{u} \in U$, tal que $\vec{u} * \vec{e} = \vec{0}$. Observe que, definido assim, este vetor $\vec{u}$ apresenta zeros nas posições dos erros.

Denote por $\mathrm{diag}(\vec{e})$ a matriz diagonal $n \times n$, cuja diagonal principal possui os elementos do vetor $\vec{e}$.

Teorema 26 *Considere (U, V) um par localizador de t erros para um código C. Considere $\vec{v} = \vec{c} + \vec{e}$ uma palavra em $\mathbb{F}_q^n$, sendo $\vec{c} \in C$ e $\vec{e}$ um vetor de peso no máximo t. Qualquer solução $\boldsymbol{\sigma}$ do sistema de equações lineares dado por*

$$H_V \cdot \mathrm{diag}(\vec{v}) \cdot G_U^T \boldsymbol{\sigma}^T = \vec{0} \tag{3.27}$$

fornece um vetor $\vec{u} = \boldsymbol{\sigma} G_U$ que satisfaz a condição

$$\vec{u} * \vec{e} = \vec{0}.$$

Além disso, existe sempre uma solução $\boldsymbol{\sigma}$ não trivial.

Prova. Considerando $\vec{v} = \vec{c} + \vec{e}$, pode-se reescrever (3.27) como

$$H_V \cdot \mathrm{diag}(\vec{c}) \cdot G_U^T \boldsymbol{\sigma}^T + H_V \cdot \mathrm{diag}(\vec{e}) \cdot G_U^T \boldsymbol{\sigma}^T = \vec{0}.$$

Pela condição 3.24, o primeiro termo da equação acima é um vetor nulo para todo $\boldsymbol{\sigma}$. Assim, o espaço de soluções de (3.27) depende apenas do vetor $\vec{e}$. Para qualquer destas soluções $\boldsymbol{\sigma}$, tem-se um vetor $\vec{u} = \boldsymbol{\sigma} G_U$ tal que

$$\vec{u} * \vec{e} \in V.$$

É fácil ver que o peso do vetor $\vec{u} * \vec{e}$ não pode exceder t, o que, pela condição 3.26, implica

$$\vec{u} * \vec{e} = \vec{0}.$$

Sobre a existência de uma solução não trivial $\boldsymbol{\sigma}$, observe que qualquer vetor não nulo $\vec{u}$ que possui zeros nas posições dos erros fornece uma solução não trivial $\boldsymbol{\sigma}$ para (3.27). Para encontrar este vetor $\vec{u}$, deve-se impor no máximo t condições linearmente independentes a U. Portanto, a existência de um vetor não trivial $\vec{u}$ é garantida pela condição 3.25. ■

Considere agora o caso em que ocorrem erros e apagamentos. Denote por $\mathcal{R}$ o conjunto das posições dos apagamentos[9]. Dado um vetor recebido $\vec{v}$, considere que $y_i = 0$ para todo $i \in \mathcal{R}$.

Também no caso de erros e apagamentos, o objetivo primeiro é encontrar um vetor $\vec{u}$, tal que $\vec{u} * \vec{e} = \vec{0}$. Entretanto, além disso, exige-se que $u_i = 0$ para todo $i \in \mathcal{R}$.

Definição 27 (Par localizador de t erros e ρ apagamentos) *Considere U, V e C códigos lineares de comprimento n sobre $\mathbb{F}_q$. Denota-se por (U, V) um par localizador de t erros e ρ apagamentos para um código C se as seguintes condições forem válidas:*

$$C * U \subseteq V \tag{3.28}$$

$$k(U) > t + \rho \tag{3.29}$$

$$d(V) > t. \tag{3.30}$$

Corolário 28 *Considere (U, V) um par localizador de t erros e ρ apagamentos para um código C. Considere $\vec{v} = \vec{c} + \vec{e}$ uma palavra em $\mathbb{F}_q^n$, sendo $\vec{c} \in C$ e $\mathcal{R}$ um conjunto de posições de erros e apagamentos, com $|\mathcal{R}| \leq \rho$. Considere um vetor erro $\vec{e}$ de peso no máximo t, excetuando-se as posições dos apagamentos. Então, qualquer solução $\boldsymbol{\sigma}$ do sistema de equações lineares dado por*

$$H_V \cdot \mathrm{diag}(\vec{v}) \cdot G_U^T \boldsymbol{\sigma}^T = \vec{0}, \tag{3.31}$$

com $\vec{u} = \boldsymbol{\sigma} G_U$ e

$$u_i = 0, \quad \textit{para todo } i \in \mathcal{R}, \tag{3.32}$$

fornece um vetor $\vec{u}$ que satisfaz

$$\vec{u} * \vec{e} = \vec{0}.$$

[9]Em diversas aplicações práticas, o decodificador tem acesso a informações adicionais acerca da confiabilidade de cada posição do vetor recebido. Estas informações permitem presumir que ocorreram erros nas ρ posições menos confiáveis. Estas posições são chamadas de *posições de apagamento*. Decodificadores de erros e apagamentos de um código C decodificam t erros e ρ apagamentos dado que $2t + \rho < d(C)$. Estes decodificadores buscam uma palavra código tal que minimize o peso de Hamming do vetor erro excetuando-se as posições dos apagamentos.

Prova. Para satisfazer a condição 3.32, considere W o subespaço de U que consiste de todos os vetores de U que possuem componentes zero nas posições de $\mathcal{R}$. Claramente, $k(W) > t + \rho - |\mathcal{R}| > t$. (W, V) é um par localizador de erros para C. Daí, o teorema 26 fornece o corolário. ■

Para a implementação do algoritmo GMD a partir deste esquema proposto por Kötter, resta apenas a escolha dos pares localizadores de t erros e ρ apagamentos (além da determinação dos valores dos erros), que não será descrita aqui. Este algoritmo GMD completo tem complexidade $O(dn^3)$. Kötter apresenta em seu artigo um esquema que reduz esta complexidade para $O(o_1 dn)$, sendo o_1 a primeira anti-lacuna do espaço de funções associado ao código. Maiores detalhes sobre este esquema podem ser obtidos no artigo de Kötter [34].

3.3 Implementação de decodificadores

O desenvolvimento de algoritmos de decodificação associados a arquiteturas de implementação em hardware é de fundamental importância para tornar os códigos AG competitivos quando comparados, por exemplo, com códigos BCH não binários ou códigos concatenados. Desta forma, na implementação em hardware, não apenas a complexidade computacional do algoritmo deve ser considerada, mas também a parte de controle necessária à operacionalização do algoritmo, a interligação de elementos no circuito, além dos requisitos de espaço.

São poucos os artigos publicados tratando da questão da implementação em hardware de decodificadores para códigos AG. Em 1997, M. E. O'Sullivan e S. P. Pope apresentaram resultados sobre algoritmos e arquiteturas de implementação de códigos AG que demonstram a viabilidade dos decodificadores para estes códigos [50].

O principal trabalho relativo a este tema foi publicado em 1998 por R. Kötter, que consiste numa versão do algoritmo BMS voltada à implementação em hardware [35]. A idéia central nesta proposta de Kötter é utilizar uma configuração em paralelo de vários algoritmos modificados de Berlekamp-Massey, de modo a obter um esquema que determine polinômios localizadores de erros para códigos AG usando os mesmos requisitos de tempo que um decodificador de Berlekamp-Massey unidimensional para decodificação de códigos RS.

Em 2001, J. B. Ashbrook et al propuseram uma implementação em circuito integrado CMOS de 3,3 V e 0,35 μm de um decodificador de códigos AG definidos sobre curvas de Hermite a partir do trabalho de Kötter [2]. Esta foi a primeira implementação em hardware de um decodificador para códigos AG. Este decodificador de códigos AG sobre

curvas de Hermite proposto trabalha com códigos de comprimento $n = 4080$, tem capacidade de correção de 60 erros em $\mathbb{F}_{256}$ e proporciona um ganho de codificação de 0,6 dB em relação a um código RS de mesma taxa. A implementação realizada opera a 50 MHz e decodifica a uma taxa de 400 Mb/s.

Ainda em 2001, E. M. Popovici et al publicaram um resumo sobre uma implementação de um decodificador de códigos AG sobre curvas de Hermite também baseado no algoritmo de Kötter [53].

Outro trabalho importante nesta área foi apresentado em 1999 por C.-W. Liu et al [43], que propuseram uma implementação em hardware do algoritmo proposto por Feng e Rao em [21] baseada numa estrutura de arranjo sistólico. Entretanto, em [3] Z. M. Belkoura apontou incorreções nas equações que determinam a estrutura sistólica proposta, comprometendo as vantagens obtidas por esta estrutura. Em [4], apresenta-se uma arquitetura considerando as equações corrigidas.

Além dos esforços para obtenção de arquiteturas para implementação eficiente de decodificadores para códigos AG, é importante destacar também a necessidade de esquemas de implementação eficiente das diversas operações em corpo finito, uma vez que a aritmética de corpo finito é o pano de fundo para todos os algoritmos de codificação / decodificação. As operações aritméticas num corpo finito $\mathbb{F}_{2^m}$ são mais complexas que as operações aritméticas com inteiros representados com o mesmo número m de bits. Os elementos de $\mathbb{F}_{2^m}$ podem ser representados a partir de uma base. Usando esta representação de base, as operações de adição e subtração tornam-se simples, contrariamente às operações de multiplicação e divisão. Diversos algoritmos existem para uma implementação mais eficiente destas operações [15, 27, 45, 51, 73, 75, 78]. No capítulo 4, são descritas as arquiteturas para unidades aritméticas em corpo finito adotadas no presente trabalho.

O principal aspecto a ser observado no que tange o desenvolvimento de decodificadores voltados à implementação em hardware é a necessidade de estruturas que permitam a realização de operações em paralelo, o que proporciona uma redução na complexidade temporal do algoritmo (em detrimento da complexidade espacial). Outro aspecto importante é o desenvolvimento de estruturas simples e regulares. A regularidade da estrutura proporciona uma menor complexidade espacial e uma redução da complexidade de controle e interligação de elementos na implementação.

A seção seguinte descreve o algoritmo proposto por Kötter em [35], de modo a ilustrar estes aspectos.

3.3.1 Decodificador de Kötter

Considere um corpo finito $\mathbb{F}_q$ de q elementos. Considere uma curva algébrica $\mathcal{X}$ de gênero g sobre $\mathbb{F}_q$. Considere ainda um divisor G, cujo suporte é constituído de pontos racionais de $\mathcal{X}$. Considere o espaço de funções $L(G)$ definido pelo divisor G. Considere um divisor D constituído pela soma simples de n pontos racionais $P_i \in \mathcal{X}$, $i = 0, \ldots, n-1$, em $\mathbb{F}_q$, distintos, tal que os suportes de D e G sejam disjuntos.

Considere um código AG denotado por $C_L(D, G)$ e dado por

$$C_L(D, G) = \{(f(P_0), f(P_1), \ldots, f(P_{n-1})) : f \in L(G)\}$$

e seu código dual

$$C_\Omega(D, G) = (C_L(D, G))^\perp$$

de comprimento $n = \deg(D)$, dimensão $k = n - \deg(G) + g - 1$ e distância mínima projetada $d^* = \deg(G) - 2g + 2$, para $n > \deg(G) > 2g - 2$.

Denote por $l(G)$ a dimensão do espaço de funções $L(G)$. O Teorema de Riemann-Roch, descrito na seção C.4, estabelece que

$$l(G) \geq \deg(G) - g + 1,$$

com igualdade se $\deg(G) > 2g - 2$.

É considerado para este algoritmo apenas o caso dos códigos AG obtidos quando $G = mQ$ é um divisor cujo suporte consiste de apenas um ponto racional Q de $\mathcal{X}$. Esta condição é importante, pois resulta em um espaço de funções $L(G)$ de estrutura relativamente regular quando comparada às estruturas resultantes de divisores G arbitrários.

As funções que possuem pólos apenas em um determinado ponto Q racional de $\mathcal{X}$ formam um anel. Por simplicidade, assume-se nesta seção que este anel é gerado em $\mathbb{F}_q$ por duas funções ψ_1 e ψ_2, com respectivos pólos de ordem v_1 e v_2, sendo $v_1 < v_2$ no ponto Q. O valor de v_1 é muito importante para o estudo da complexidade do algoritmo e será denotado por γ.

Considere um vetor recebido $\vec{v}$ de comprimento n e dado por $\vec{v} = \vec{c} + \vec{e}$, em que $\vec{c} \in C_\Omega(D, G)$ e $\vec{e}$ é um vetor erro com peso de Hamming $t < \frac{d^*(C_\Omega)}{2}$. As síndromes correspondentes a este vetor recebido são definidas por

$$S_{i,j} = \sum_{l=0}^{n-1} e_l \psi_1^i(P_l) \psi_2^j(P_l).$$

Observe da definição do código $C_\Omega(D, mQ)$ que estas síndromes podem ser calculadas a partir do vetor $\vec{v}$ se $iv_1 + jv_2 \leq m$.

Considere agora uma função f da forma

$$f = \sum_{(i,j):iv_1+jv_2\leq w} f_{i,j}\psi_1^i\psi_2^j, \tag{3.33}$$

que apresenta pólo de ordem w no ponto Q. Uma função f é chamada função localizadora de erros se ela se anula nos pontos equivalentes às posições dos erros, ou seja, se $f(P_l)\, e_l = 0$, para todo l. A principal observação que permite obter funções f localizadoras de erros é o fato de que estas funções definidas em 3.33 fornecem as relações recursivas lineares

$$\sum_{(i,j):iv_1+jv_2\leq w} f_{i,j}S_{i+i_0,j+j_0} = 0 \ \forall i_0, j_0 \geq 0. \tag{3.34}$$

sobre a matriz de síndromes

$$S = \begin{pmatrix} S_{1,1} & S_{1,2} & \cdots \\ S_{2,1} & S_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Esta função localizadora de erros f pode ser obtida a partir das síndromes conhecidas apenas se o peso de Hamming do vetor erro for menor que $\frac{(m-3g+1)}{2} = \frac{d^*(C_\Omega)-g}{2}$ (veja [32]). Para se atingir a capacidade de correção de erros proporcionada pelo código, faz-se necessário determinar as síndromes desconhecidas $S_{i,j}$, para $m < iv_1 + jv_2 \leq m+g$, o que é feito usando-se o esquema de decisão por maioria de Feng e Rao, desde que o peso do vetor erro ocorrido seja inferior a $\frac{d_{FR}(C_\Omega)}{2}$, em que $d_{FR}(C_\Omega) \geq d^*(C_\Omega)$ denota a distância de Feng-Rao do código $C_\Omega(D, mQ)$ (*cf.* [21]).

Uma vez obtida uma função localizadora de erros, pode-se determinar o vetor erro ocorrido com uma complexidade $O\left((\log_2(q))^2 n^2\right)$ igual à necessária para calcular as síndromes de uma palavra recebida. A complexidade total de um algoritmo de decodificação de códigos AG é normalmente dominada pela complexidade do algoritmo usado para determinar a função localizadora de erros. O algoritmo de Feng e Rao utilizava eliminação gaussiana para resolver o sistema de equações 3.34 e apresentava complexidade $O\left((\log_2(q))^2 n^3\right)$. O algoritmo de Kötter aqui descrito baseia-se no algoritmo BMS, e apresenta uma estrutura paralela simples e complexidade $O\left((\log_2(q))^2 \gamma n^2\right)$, que é essencialmente a complexidade de qualquer algoritmo de decodificação eficiente para códigos AG.

O algoritmo BMS basicamente é um algoritmo iterativo que opera sobre dois conjuntos de funções F_u e G_u. Cada iteração do algoritmo BMS tem por objetivo produzir dois novos conjuntos de funções F_{u+1} e G_{u+1}, que servirão de entrada para a próxima iteração. O conjunto F_u consiste nas funções f de pólos de ordem w que satisfazem a equação 3.34 para todo (i_0, j_0), tal que $i_0v_1 + j_0v_2 + w \leq u$. O conjunto G_u consiste nas funções

g que não satisfizeram a relação da equação 3.34 em iterações anteriores do algoritmo. Resumidamente, são as seguintes as operações executadas em cada iteração do algoritmo BMS:

1. Para cada função $f \in F_u$, verifica-se na equação 3.34 se f é válido como elemento de F_{u+1};

2. A partir das ordens de pólo das funções de F_u e G_u, calcula-se as ordens de pólo das funções de F_{u+1}, de acordo com um certo número de diferentes casos. As funções $f' \in F_{u+1}$ são, então, construídas selecionando e combinando apropriadamente pares de funções (f, g), sendo $f \in F_u$ e $g \in G_u$. Isto consiste numa operação do tipo

$$f' \leftarrow \psi_1^a \psi_2^b f + \delta \psi_1^{a'} \psi_2^{b'} g,$$

   em que a, b, a', b' são determinados através um certo número de diferentes casos;

3. Os elementos de G_{u+1} são escolhidos entre os elementos de G_u e F_u, tal que certos requisitos associados com o conjunto F_{u+1} sejam satisfeitos.

Um dos principais problemas no que tange a eficiência do algoritmo BMS é que, no passo 2 de cada iteração, um par de funções (f, g) necessário para construir uma função $f' \in F_{u+1}$ é impossível de ser previsto a priori. No algoritmo de Kötter, este problema é resolvido.

Matriz de síndromes

O algoritmo de Kötter representa as síndromes do vetor recebido numa forma matricial. Neste caso, as condições lineares impostas pela função localizadora de erros na equação 3.34 são traduzidas numa linguagem matricial.

Denote por ϕ_{o_j} uma função que possui pólo de ordem o_j no ponto Q da curva $\mathcal{X}$, sendo o_j uma anti-lacuna de Q. ϕ_{o_1} é a função não constante de menor ordem de pólo no ponto Q e, por definição, $\gamma = o_1$. Observe que se o_i e o_j forem duas anti-lacunas, então $o_i + o_j$ é também uma anti-lacuna.

Denote por G o conjunto das lacunas de Q, em que $l(o_j Q) = l((o_j - 1)Q)$.

Segue a definição de uma base para o espaço de funções $L(mQ)$, utilizada no algoritmo de Kötter.

Definição 29 *Uma base padrão $B(m) = \{1, \phi_\gamma, \ldots, \phi_m\}$ para para o espaço de funções $L(mQ)$ possui as seguintes propriedades:*

1. *Se ϕ_j é um elemento de $B(m)$ e l é a menor anti-lacuna positiva em $B(m)$, tal que $j-l$ é também uma anti-lacuna, então*

$$\phi_j = \phi_l \phi_{j-1};$$

2. *É possível determinar constantes $\varepsilon_i^{(a,b)}$, tais que o produto entre quaisquer funções ϕ_a e ϕ_b, com $a+b \leq m$, é dado por*

$$\phi_a \phi_b = \phi_{a+b} + \sum_{i=0}^{a+b-1} \varepsilon_i^{(a,b)} \phi_i.$$

Lema 30 *Para um dado ponto Q de uma curva $\mathcal{X}$, é sempre possível construir uma base padrão $B(m) = \{1, \phi_\gamma, \ldots, \phi_m\}$ para o espaço de funções $L(mQ)$.*

A conseqüência mais importante da definição 29 é que, para qualquer $\phi_l \in B(m)$, com $l \leq m - \gamma$, a função $\phi_\gamma \phi_l$ também é um elemento de $B(m)$. Ou seja, $B(m)$ é fechada sob a multiplicação com ϕ_γ, desde que mantida a consistência com a ordem de pólo máxima m.

De forma a obter uma formulação estruturalmente adequada para o algoritmo de Kötter, define-se formalmente $\phi_i = 0$ se $i \in G$. Pode-se, então, associar univocamente qualquer função $f \in L(mQ)$ com o somatório

$$f = \sum_{i=0}^{m} f_i \phi_i, \text{ com } f_i = 0 \ \forall \ i \in G$$

e identificar vetores $\vec{f} = (f_0, f_1, \ldots, f_m)$ com funções. Essa identificação será livremente usada, de modo que uma função possa ser considerada uma solução de um sistema de equações se o vetor de coeficientes correspondente for solução deste sistema.

Considere agora um vetor recebido $\vec{v} = \vec{c} + \vec{e}$ de comprimento n, com $\vec{c} \in C_\Omega(D, mQ)$. Defina, então, os elementos de uma matriz semi-infinita $S = \|S_{i,j}\|$, com $i, j \geq 0$, como

$$S_{i,j} = \sum_{l=0}^{n-1} e_l \phi_i(P_l) \, \phi_j(P_l). \tag{3.35}$$

Observe que as condições lineares que a matriz S impõe sobre um vetor $\boldsymbol{\sigma}$ num sistema de equações lineares $S\boldsymbol{\sigma}^T = \vec{0}$ são uma reformulação das condições lineares descritas pela equação 3.34. Portanto, qualquer solução da equação $S\boldsymbol{\sigma}^T = \vec{0}$ corresponde ao vetor de coeficientes de uma função localizadora de erros.

Rotina geral

Denote por $S^{(a,b)}$ a sub-matriz de S composta pelos elementos das primeiras $a+1$ linhas com as primeiras $b+1$ colunas de S, ou seja, $S^{(a,b)} = \|S_{i,j}\|$, para $0 \leq i \leq a$ e $0 \leq j \leq b$.

Suponha que seja dada uma função

$$\sigma \in L(bQ) \setminus L((b-1)Q)$$

que seja solução para o sistema de equações lineares

$$S^{(a-1,b)}\boldsymbol{\sigma}^T = \vec{0} \qquad\qquad \sum_{i=0}^{b} \sigma_i S_{a,i} = \Delta \neq 0.$$

Neste caso, diz-se que a função σ fornece uma discrepância Δ na posição (a, b). Se uma função $\sigma \in L(bQ) \setminus L((b-1)Q)$ é solução para o sistema de equações lineares $S\boldsymbol{\sigma}^T = \vec{0}$, então diz-se formalmente que σ fornece uma discrepância na posição (∞, b).

A idéia do algoritmo proposto é encontrar iterativamente funções não nulas σ que sejam soluções do sistema $S^{(a,b)}\boldsymbol{\sigma}^T = \vec{0}$, para valores crescentes de $r = a + b$.

O lema que segue fornece a possibilidade de combinar soluções parciais.

Lema 31 *Considere uma função σ que fornece uma discrepância Δ na posição (a, b). Além disso, suponha que seja dada uma função δ que fornece uma discrepância de valor 1 na posição (a, b'), com $b' \leq b$. Portanto, a função $\sigma' = \sigma - \Delta\delta$ fornece uma discrepância na posição (a', b'), sendo $a' > a$.*

O lema que segue é uma das chaves para melhorar a eficiência de um algoritmo tipo BMS.

Lema 32 *Considere uma função σ que fornece uma discrepância Δ na posição (a, b). Então, a função $\phi_\gamma\sigma$ fornece uma discrepância Δ' na posição $(a', b+\gamma)$, sendo*

$$a' = \begin{cases} > a - \gamma, & a - \gamma \in G, \\ a - \gamma, & \textit{caso contrário.} \end{cases}$$

Além disso, se $a - \gamma \notin G$, então $\Delta' = \Delta$.

O lema que segue combina os lemas 31 e 32 para formar o passo de indução básico do algoritmo de Kötter.

Lema 33 *Considere σ e λ duas funções que fornecem discrepâncias nas posições (a, b) e (a', b') com valores Δ e 1, respectivamente. Considere também que os números a, b, a' e*

b' satisfazem as relações $a' + b' < a + b$ e $a' \equiv a \pmod{\gamma}$. É possível se obter uma função σ' que forneça uma discrepância na posição (a'', b''), em que $a'' + b'' > a + b$, fazendo

$$\sigma' = \begin{cases} \sigma - \Delta\phi_\gamma^{(a'-a)/\gamma}\lambda, & a \leq a' \\ \phi_\gamma^{(a-a')/\gamma}\sigma - \Delta\lambda, & a' < a. \end{cases} \tag{3.36}$$

Considere agora todas as funções de ordem de pólo b no ponto Q na forma

$$f = \sum_{i=0}^{b} f_i\phi_i, \quad f_i = 0 \,\forall\, i \in G.$$

A cada uma destas funções, associe um polinômio em uma variável z na forma

$$\bar{f}(z) = \sum_{i=0}^{b} f_i z^{b-i}.$$

O uso de polinômios em uma variável permite uma reformulação concisa do lema 32. No corolário que segue, é feita a unificação dos dois casos do lema 33.

Corolário 34 *Considere as funções σ, λ e σ' como definidas no lema 33. Os polinômios em uma variável correspondentes $\bar{\sigma}$, $\bar{\lambda}$ e $\bar{\sigma}'$ satisfazem a relação*

$$\bar{\sigma}' = \bar{\sigma} - \Delta z^{(a+b)-(a'+b')}\bar{\lambda}.$$

O lema 32 sugere um particionamento de funções σ em classes de equivalência, em que duas funções pertencem à mesma classe se suas ordens de pólo no ponto Q forem equivalentes módulo γ. Da mesma forma, o lema 33 sugere também um particionamento de funções λ em classes de equivalência, em que duas funções pertencem à mesma classe se fornecerem discrepâncias em linhas cujos índices são equivalentes módulo γ. Pode-se definir, portanto, dois conjuntos de funções $\sigma^{(r,i)}$ e $\lambda^{(r,i)}$.

Definição 35 *Conjuntos de funções $\sigma^{(r,i)}$ e $\lambda^{(r,i)}$ são ditos válidos na ordem de pólo r se forem respeitadas as seguintes condições:*

- *Considere $b_\sigma^{(r,i)}$ o menor inteiro, tal que $b_\sigma^{(r,i)} \notin G$ e $b_\sigma^{(r,i)} \equiv i \pmod{\gamma}$. Então, existe uma função que fornece uma discrepância na posição $\left(a,\, b_\sigma^{(r,i)}\right)$, sendo $a + b_\sigma^{(r,i)} > r$. A função $\sigma^{(r,i)}$ é definida como uma função que fornece uma discrepância na posição $\left(a,\, b_\sigma^{(r,i)}\right)$.*
- *Se uma função λ existe e fornece uma discrepância na posição $(a,\, b)$, com $a \equiv j \pmod{\gamma}$ e $a + b \leq r$, considere $a_\lambda^{(r,i)}$ o maior inteiro, tal que $a_\lambda^{(r,i)} \equiv j \pmod{\gamma}$. Então, existe uma função que fornece uma discrepância na posição $\left(a_\lambda^{(r,i)},\, b\right)$, sendo*

$a_\lambda^{(r,i)} + b \leq r$. *A função* $\lambda^{(r,i)}$ *é definida como uma função que fornece uma discrepância de valor 1 na posição* $\left(a_\lambda^{(r,i)}, b\right)$. *Se estas funções não existem, define-se* $\lambda^{(r,i)} = 0$ *e*

$$a_\lambda^{(r,i)} = \max_{l \in G} \{l : l \equiv j \pmod{\gamma}\}.$$

- *Além disso, define-se os números* $a_\sigma^{(r,i)}$, $j^{(r,i)}$ *e* $\Delta^{(r+1,i)}$ *como*

$$\begin{aligned} a_\sigma^{(r,i)} &= r + 1 - b_\sigma^{(r,i)} \\ j^{(r,i)} &= a_\sigma^{(r,i)} \pmod{\gamma} \\ \Delta^{(r+1,i)} &= \begin{cases} \sum_{l=0}^{b_\sigma^{(r,i)}} \sigma_l^{(r,i)} S_{a_\sigma^{(r,i)}, l}, & a_\sigma^{(r,i)} \notin G \\ 0, & a_\sigma^{(r,i)} \in G. \end{cases} \end{aligned}$$

O próximo lema torna clara a relevância da definição de $\sigma^{(r,i)}$ para a decodificação dos códigos AG.

Lema 36 *Considere para um código* $C_\Omega(D, mQ)$ *um vetor erro* $\vec{e}$ *de peso de Hamming* $t < (m - 2g + 2)/2 = d^*(C_\Omega)/2$. *O espaço de funções localizadoras de erros possui uma base* $\left\{\phi_\gamma^{l_i} \sigma^{(m+2g+\gamma-1,i)}\right\}$, *em que* $l_i \geq 0$ *e* $0 \leq i < \gamma$. *Além disso, a função localizadora de erros de menor ordem de pólo no ponto* Q *é a função de menor ordem de pólo do conjunto* $\left\{\sigma^{(m+g,i)}\right\}_{i=0}^{\gamma-1}$.

O lema 33 e a definição 35 são os dois elementos principais na construção do algoritmo de Kötter. Dadas as funções $\sigma^{(r,i)}$ e $\lambda^{(r,j)}$, pode-se determinar as funções $\sigma^{(r+1,i)}$ e $\lambda^{(r+1,j)}$ aplicando-se o lema 33 a pares de funções $\sigma^{(r,i)}$ e $\lambda^{(r,j)}$, em que i e j são escolhidos de modo que $a_\sigma^{(r,i)} \equiv a_\lambda^{(r,j)} \pmod{\gamma}$. Para quaisquer $0 \leq i, j < \gamma$, existem sempre funções $\sigma^{(r,i)}$ e $\lambda^{(r,j)}$, portanto podem ser usados quaisquer pares $\left(\sigma^{(r,i)}, \lambda^{(r,j)}\right)$ necessários. Com relação a isso, a observação importante na construção de um algoritmo paralelo é que dois pares diferentes de funções nunca possuem uma função em comum (não há essa necessidade). Por fim, as funções $\sigma^{(r+1,i)}$ são obtidas do lema 33 e as funções $\lambda^{(r+1,j)}$ são iguais a $\lambda^{(r,j)}$ ou a um múltiplo escalar de $\sigma^{(r,i)}$.

Lema 37 *Considere duas funções* $\sigma^{(r,i)}$ *e* $\lambda^{(r,j)}$, *com os valores associados* $a_\sigma^{(r,i)}$, $a_\lambda^{(r,j)}$ *e* $\Delta^{(r+1,i)}$ *tais que*

$$a_\sigma^{(r,i)} \equiv a_\lambda^{(r,j)} \pmod{\gamma}.$$

Pode-se calcular $\sigma^{(r+1,i)}$ *através de*

$$\sigma^{(r+1,i)} = \begin{cases} \sigma^{(r,i)} - \Delta^{(r+1,i)} \phi_\gamma^{(a_\lambda^{(r,j)} - a_\sigma^{(r,i)})/\gamma} \lambda^{(r,j)}, & a_\sigma^{(r,i)} \leq a_\lambda^{(r,j)} \\ \phi_\gamma^{(a_\sigma^{(r,i)} - a_\lambda^{(r,j)})/\gamma} \sigma^{(r,i)} - \Delta^{(r+1,i)} \lambda^{(r+1,j)}, & \textit{caso contrário} \end{cases}$$

e $\lambda^{(r+1,\,j)}$ *através de*

$$\lambda^{(r+1,\,j)} = \begin{cases} \left(\Delta^{(r+1,\,i)}\right)^{-1} \sigma^{(r,\,i)}, & \left(\Delta^{(r+1,\,i)} \neq 0\right) \wedge \left(a_\lambda^{(r,\,j)} < a_\sigma^{(r,\,i)}\right) \\ \lambda^{(r,\,j)}, & \textit{caso contrário.} \end{cases}$$

De modo a se obter uma notação concisa, procede-se novamente associando as funções $\sigma^{(r,\,i)}$ e $\lambda^{(r,\,j)}$ a polinômios em uma variável z. Considere, então,

$$\bar{\sigma}^{(r,\,i)}(z) = \sum_{l=0}^{b_\sigma^{(r,\,i)}} \sigma_l^{(r,\,i)} z^{b_\sigma^{(r,\,i)} - l}$$

e

$$\bar{\lambda}^{(r,\,j)}(z) = z^{r+1-(a_\sigma^{(r,\,j)} + b_\sigma^{(r,\,j)})} \left(\sum_{l=0}^{b_\sigma^{(r,\,j)}} \lambda_l^{(r,\,j)} z^{b_\lambda^{(r,\,j)} - l} \right)$$

Usando essa notação, do lema 37 obtém-se o seguinte corolário.

Corolário 38 *Considere duas funções* $\sigma^{(r,\,i)}$ *e* $\lambda^{(r,\,j)}$*, com os valores associados* $a_\sigma^{(r,\,i)}$*,* $a_\lambda^{(r,\,j)}$ *e* $\Delta^{(r+1,\,i)}$ *tais que*

$$a_\sigma^{(r,\,i)} \equiv a_\lambda^{(r,\,j)} \pmod{\gamma}.$$

Os polinômios correspondentes $\bar{\sigma}^{(r,\,i)}(z)$*,* $\bar{\lambda}^{(r,\,j)}(z)$*,* $\bar{\sigma}^{(r+1,\,i)}(z)$ *e* $\bar{\lambda}^{(r+1,\,j)}(z)$ *satisfazem as relações*

$$\bar{\sigma}^{(r+1,\,i)}(z) = \bar{\sigma}^{(r,\,i)}(z) - \Delta^{(r+1,\,i)} \bar{\lambda}^{(r,\,j)}(z)$$

e

$$\bar{\lambda}^{(r+1,\,j)}(z) = \begin{cases} \left(\Delta^{(r+1,\,i)}\right)^{-1} z\bar{\sigma}^{(r+1,\,i)}(z), & \left(\Delta^{(r+1,\,i)} \neq 0\right) \wedge \left(a_\lambda^{(r,\,j)} < a_\sigma^{(r,\,i)}\right) \\ z\bar{\lambda}^{(r,\,j)}(z), & \textit{caso contrário.} \end{cases} \tag{3.37}$$

Defina os números l_i como

$$l_i = \min\{l : (l \equiv i \pmod{\gamma}) \wedge (l \notin G)\}.$$

O algoritmo de Kötter é, então, definido como um conjunto de equações recursivas.

Algoritmo 39 (Kötter) *:*

- ***Inicialização:***
  - $\bar{\sigma}^{(-1,\,i)}(z) = 1$*, para* $0 \leq i < \gamma$*;*
  - $a_\sigma^{(-1,\,i)} = -l_i$*, para* $0 \leq i < \gamma$*;*
  - $\bar{\lambda}^{(-1,\,j)}(z) = 0$*, para* $0 \leq j < \gamma$*;*
  - $a_\lambda^{(-1,\,j)} = l_j - \gamma$*, para* $0 \leq j < \gamma$*.*

- ***Iterações:***

  1. *Calcular*

$$j^{(r,i)} = a_\sigma^{(r,i)} \pmod{\gamma}; \tag{3.38}$$

  2. *Calcular os termos*

$$\Delta^{(r+1,i)} = \begin{cases} 0, & a_\sigma^{(r,i)} \in G \\ \sum_{h=0}^{r+1-a_\sigma^{(r,i)}} \bar{\sigma}_h^{(r,i)} S_{a_\sigma^{(r,i)}, r+1-a_\sigma^{(r,i)}-h}, & \textit{caso contrário}, \end{cases} \tag{3.39}$$

$$\delta^{(r,i)} = \begin{cases} 1, & \left(\Delta^{(r+1,i)} \neq 0\right) \wedge \left(a_\lambda^{(r,j^{(r,i)})} < a_\sigma^{(r,i)}\right) \\ 0, & \textit{caso contrário}; \end{cases} \tag{3.40}$$

  3. *Calcular os termos (o quociente $\delta^{(r,i)}/\Delta^{(r+1,i)}$ é considerado zero se $\Delta^{(r+1,i)}$ for igual a zero)*

$$\begin{pmatrix} \bar{\sigma}^{(r+1,i)}(z) \\ \bar{\lambda}^{(r+1,j^{(r,i)})}(z) \end{pmatrix} = \begin{pmatrix} 1 & -\Delta^{(r+1,i)} \\ \delta^{(r,i)}/\Delta^{(r+1,i)} z & \left(1-\delta^{(r,i)}\right) z \end{pmatrix} \begin{pmatrix} \bar{\sigma}^{(r,i)}(z) \\ \bar{\lambda}^{(r,j^{(r,i)})}(z) \end{pmatrix}, \tag{3.41}$$

$$a_\sigma^{(r+1,i)} = \left(1-\delta^{(r,i)}\right) a_\sigma^{(r,i)} + \delta^{(r,i)} a_\lambda^{(r,j^{(r,i)})} + 1, \tag{3.42}$$

$$a_\lambda^{(r+1,j^{(r,i)})} = \left(1-\delta^{(r,i)}\right) a_\lambda^{(r,j^{(r,i)})} + \delta^{(r,i)} a_\sigma^{(r,i)}. \tag{3.43}$$

Prova. Para detalhes sobre a prova deste algoritmo, veja [35]. ■

Teorema 40 *O conjunto das equações recursivas do algoritmo 39 pode ser usado para calcular os polinômios $\bar{\sigma}^{(s,i)}(z)$ e $\bar{\lambda}^{(s,j)}(z)$, com $0 \leq i,j < \gamma$, para qualquer $s \geq 0$. As funções correspondentes $\sigma^{(s,i)}$ e $\lambda^{(s,i)}$ são válidas na ordem de pólo s.*

Os polinômios $\bar{\lambda}^{(r,j)}(z)$ usados na $(r+1)$-ésima iteração do algoritmo 39 para atualizar $\sigma^{(r,i)}$ mudam na iteração seguinte, de acordo com a definição de $j^{(r,i)}$. Entretanto, eles mudam de uma forma bastante regular, como ficará claro no lema seguinte. A regularidade desta mudança será útil para a implementação deste algoritmo em hardware.

Lema 41 *Os números $j^{(r,i)}$ satisfazem a relação*

$$j^{(r+1,i)} = j^{(r,\,(i+1) \pmod{\gamma})}.$$

O algoritmo 39 possui algumas propriedades interessantes. Os cálculos dos γ polinômios $\bar{\sigma}^{(s,i)}(z)$ na $(s+1)$-ésima iteração são inteiramente independentes entre si e podem ser implementados em paralelo. Dadas γ unidades de processamento similares capazes de realizar os cálculos de 3.38 a 3.43, pode-se obter todos os polinômios $\bar{\sigma}^{(s,i)}$ em $s+1$ iterações.

A complexidade de qualquer unidade de processamento em uma iteração é da ordem de $O\left((\log_2(q))^2 s\right)$. Uma vez que é necessário processar s iterações em γ unidades de processamento, tem-se que a complexidade total para encontrar todos os polinômios $\sigma^{(s,\,i)}$ não é maior que $O\left((\log_2(q))^2 \gamma s^2\right)$. Numa implementação em paralelo, os requisitos de tempo são determinados pelos requisitos de tempo de uma única unidade de processamento. Se estas unidades de processamento operassem de modo serial, os requisitos de tempo seriam da ordem de $O\left(s^2\right)$, portanto essencialmente o mesmo que de um algoritmo BMA implementado de modo serial, projetado para corrigir $s/2$ erros em uma palavra código de um código de Reed-Solomon.

Rotina de decisão por maioria

O conjunto das equações recursivas do algoritmo 39 permite determinar as funções $\sigma^{(r,\,i)}$ para todo $r \geq 0$, desde que seja conhecida toda a matriz de síndromes S. Entretanto, apenas as entradas de $S_{a,\,b}$, com $a + b \leq m$ são conhecidas. Em [21], Feng e Rao apresentaram um procedimento para determinar iterativamente as entradas desconhecidas $S_{a,\,b}$, para $a + b > m$, a partir da parte conhecida de S, desde que o peso do vetor erro ocorrido não excedesse a capacidade de correção de erros do código.

Uma observação importante com relação ao esquema de Feng e Rao é que, pela segunda propriedade da definição 29 de uma base padrão, todas as entradas $S_{a,\,b}$, com $a+b = m+w$ e $w \geq 1$, podem ser expressas na forma

$$S_{a,\,b} = S_{m+w,\,0} + \sum_{i=0}^{m+w-1} \varepsilon_i^{(a,\,b)} S_{i,\,0}, \tag{3.44}$$

em que os coeficientes $\varepsilon_i^{(a,\,b)}$ são conhecidos e determinados pela curva algébrica utilizada na definição dos códigos. São tratados aqui os casos em que w é igual a 1. Os casos em que $w > 1$ podem ser tratados iterativamente de maneira similar.

Uma segunda observação importante é o fato de que o posto da matriz S é igual ao peso t do vetor erro ocorrido. Para verificar isso, considere M uma matriz com entradas definidas por $M_{i,\,j} = \phi_i(P_j)$. S pode, então, ser descrita como

$$S = M\,\mathrm{diag}(\vec{e})M^T,$$

em que $\mathrm{diag}(\vec{e})$ é a matriz diagonal contendo o vetor $\vec{e}$ em sua diagonal principal. Observe que $\mathrm{posto}(M) = n$ e $\mathrm{posto}(\mathrm{diag}(\vec{e})) = t$, portanto o posto de S é igual a t.

De modo a descrever o procedimento de Feng e Rao, é necessário introduzir o conceito de *discrepância de matriz*. Diz-se que S é uma matriz de discrepâncias na posição $(a,\, b)$

se as três condições seguintes forem satisfeitas:

$$\begin{aligned}\text{posto}(S^{(a,\,b)}) &= \text{posto}(S^{(a-1,\,b)}) + 1 \\ &= \text{posto}(S^{(a,\,b-1)}) + 1 \\ &= \text{posto}(S^{(a-1,\,b-1)}) + 1.\end{aligned}$$

Lema 42 *O posto de $S^{(a,\,b)}$ é igual ao número de discrepâncias de matriz nas posições $(a',\,b')$, com $a' \leq a$ e $b' \leq b$. Além disso, qualquer coluna ou linha de S contém no máximo uma discrepância de matriz.*

As discrepâncias fornecidas pelas funções do algoritmo 39 e as discrepâncias de matriz são relacionadas pelo lema que segue.

Lema 43 *Considere os dois polinômios $\bar{\sigma}^{(r,\,i)}(z)$ e $\bar{\lambda}^{(r,\,j)}(z)$, sendo $j \equiv a_\sigma^{(r,\,i)} \pmod{\gamma}$ e $\Delta^{(r+1,\,i)} \neq 0$. Então, S possui $\max\left\{0,\, \gamma^{-1}\left(a_\sigma^{(r,\,i)} - a_\lambda^{(r,\,j)}\right)\right\}$ discrepâncias de matriz nas posições*

$$\left(a_\sigma^{(r,\,i)} - l\gamma,\, r+1-a_\sigma^{(r,\,i)} + l\gamma\right), \quad \textit{para } l = 0, 1, \ldots, \gamma^{-1}\left(a_\sigma^{(r,\,i)} - a_\lambda^{(r,\,j)}\right) - 1. \tag{3.45}$$

Além disso, S não possui discrepâncias de matriz nas posições

$$(a,\, r+1-a), \quad \textit{para } a \equiv a_\sigma^{(r,\,i)} \pmod{\gamma},$$

fora as posições dadas em 3.45.

Considere que o algoritmo 39 teve m iterações processadas. De modo a poder processar a iteração seguinte do algoritmo, é necessário conhecer as entradas $S_{a,\,b}$ de S, para $a+b = m+1$. A estratégia para determinar $S_{m+1,\,0}$ e, portanto, todas as $S_{a,\,b}$, com $a+b = m+1$, é minimizar o posto de S ou, equivalentemente, o número de discrepâncias de matriz nas posições $(a,\, b)$, com $a + b = m + 1$. O lema 43 fornece as ferramentas necessárias para isso.

É provado em [21] que mais da metade das discrepâncias de matriz que podem aparecer na matriz S nas posições $(a,\, b)$, com $a+b = r+1 > m$, podem ser removidas. Isto é feito escolhendo $S_{r+1,\,0}$ de modo que a maioria das discrepâncias dadas pelas funções $\sigma^{(r,\,i)}$ seja igual a zero. O procedimento proposto começa pela escolha de uma estimativa $\hat{S}_{r+1,\,0} = 0$, que, de acordo com a equação 3.44, induz ao mesmo tempo todas as outras estimativas $\hat{S}_{a,\,b}$, com $a + b = r + 1$. Então, para todo i, tal que $a_\sigma^{(r,\,i)} \notin G$, calcula-se os números

$$\hat{\Delta}^{(r+1,\,i)} = \hat{S}_{a_\sigma^{(r,\,i)},\, r+1-a_\sigma^{(r,\,i)}} + \sum_{h=1}^{r+1-a_\sigma^{(r,\,i)}} \bar{\sigma}_h^{(r,\,i)} S_{a_\sigma^{(r,\,i)},\, r+1-a_\sigma^{(r,\,i)}-h}. \tag{3.46}$$

Observe que o termo constante nos polinômios $\bar{\sigma}^{(r,i)}(z)$ é sempre igual a 1 quando o algoritmo 39 é inicializado apropriadamente. Comparando 3.46 com o cálculo de $\Delta^{(r,i)}$ em 3.39 e usando a expansão de $S_{a,b}$ nas síndromes $S_{r,0}$, com $0 \leq r \leq a+b$, dada em 3.44, pode-se escrever

$$\hat{\Delta}^{(r+1,i)} = \Delta^{(r+1,i)} - S_{r+1,0}.$$

Segue que $\Delta^{(r+1,i)} = \hat{\Delta}^{(r+1,i)} + S_{r+1,0}$ e pode-se utilizar a decisão por maioria para determinar $S_{r+1,0}$, tal que a maioria dos $\Delta^{(r+1,i)}$ seja igual a zero, em que cada $\Delta^{(r+1,i)}$ é contado com multiplicidade $\max\left\{0,\, a_\sigma^{(r,i)} - a_\lambda^{(r,j)}\right\}$, de acordo com o lema 43. Este procedimento é formalizado no algoritmo que segue.

Algoritmo 44 (Kötter: Decisão por maioria) *:*

1. *Calcular $\hat{\Delta}^{(r+1,i)}$, para todo i em que $a_\sigma^{(r,i)} \notin G$, e considerar D o conjunto dos números $\hat{\Delta}^{(r+1,i)}$;*

2. *Encontrar o elemento $\hat{S} \in D$ que forneça o maior número*

$$\sum_{i:\hat{\Delta}^{(r+1,i)}=\hat{S}} \max\left\{0,\, a_\sigma^{(r,i)} - a_\lambda^{(r,j^{(r,i)})}\right\},$$

   em que cada somatório é feito sobre os i-ésimos elementos de D que apresentam um mesmo valor $\hat{S}$;

3. *Calcular*

$$S_{r+1,0} = -\hat{S} \quad e \quad \Delta^{(r+1,i)} = \hat{\Delta}^{(r+1,i)} + S_{r+1,0}.$$

Prova. Para detalhes sobre a prova deste algoritmo, veja [35]. ■

Teorema 45 *Considere todas as entradas $S_{a,b}$ em S, com $a+b \leq r$, e os polinômios $\bar{\sigma}^{(r,i)}$ e $\bar{\lambda}^{(r,j)}$, com os valores associados $a_\sigma^{(r,i)}$ e $a_\lambda^{(r,j)}$. O algoritmo 44 pode ser usado para determinar $S_{r+1,0}$ e $\Delta^{(r+1,i)}$ se*

$$t < (m - 2g + 2)/2 = d^*(C_\Omega)/2.$$

O teorema 45 abre a possibilidade de calcular o conjunto $\sigma^{(r,i)}$, para um $r \geq 0$ arbitrário, usando o algoritmo 39, uma vez que qualquer entrada de S pode ser determinada iterativamente quando necessária. Para isso, deve-se substituir 3.39 no algoritmo 39 pelo algoritmo 44.

A complexidade do algoritmo de decisão por maioria é determinado principalmente pela procura do máximo num conjunto de até γ elementos. Isto não afeta a complexidade geral do algoritmo de Kötter, que é ainda limitado por $O\left((\log_2(q))^2\, \gamma m^2\right)$.

Implementação

O conjunto das equações do algoritmo 39 é similar ao conjunto das equações do algoritmo BMS unidimensional. Isto torna possível projetar uma implementação baseada em γ cópias de um decodificador BMA modificado. Para ilustrar isso, pode-se observar a implementação serial do decodificador BMA proposta por Blahut em [6, p. 189].

Na descrição da implementação do algoritmo de Kötter que segue, são considerados os códigos AG definidos sobre curvas de Hermite em corpos de característica 2. As modificações necessárias para implementação utilizando outras curvas serão discutidas no final da seção.

Para as curvas de Hermite, tem-se que $\gamma = q$. No restante do capítulo, será escrito γ se a proposição for válida para qualquer curva algébrica e q se o valor for específico para as curvas de Hermite.

É desejável que se evite utilizar a matriz S completa. Isto é possível, uma vez que apenas as entradas $S_{a,\,0}$ e o conhecimento da curva $\mathcal{X}$ são necessários para reconstruir a parte desejada de S a cada passo do algoritmo. Para a curva de Hermite, tem-se o seguinte lema.

Lema 46 *Considere a curva $\mathcal{X}$ definida pela equação 2.9 (seção 2.2). As entradas $S_{a,\,b}$ da matriz S satisfazem as relações*

$$S_{a,\,b} = \begin{cases} 0, & (a \in G) \vee (b \in G) \\ S_{a+b,\,0}, & (a \pmod q) + (b \pmod q) < q \\ S_{a+b,\,0} - S_{a+b-q^2+1,\,0}, & \textit{caso contrário.} \end{cases}$$

Defina uma seqüência $h(a,\,b)$ de período $2q$ da forma

$$h(a,\,b) = \begin{cases} 0, & (a \pmod q) + (b \pmod q) < q \\ 1, & \text{caso contrário.} \end{cases}$$

O cálculo de $\Delta^{(r+1,\,i)}$ no teorema 40 pode ser reescrito como

$$\Delta^{(r+1,\,i)} = \sum_{l=0}^{r+1-a_\sigma^{(r,\,i)}} \bar{\sigma}_l^{(r,\,i)} \left(S_{r+1-l,\,0} - h\left(a_\sigma^{(r,\,i)},\, b_\sigma^{(r,\,i)} - l\right) S_{r-q^2+2-l,\,0} \right). \tag{3.47}$$

Observe que nem todo $S_{r-q^2+2-l,\,0}$ é definido. Entretanto, se $r-q^2+2-l$ for negativo, então $h\left(a_\sigma^{(r,\,i)},\, b_\sigma^{(r,\,i)} - l\right)$ ou $\bar{\sigma}_l^{(r,\,i)}$ é igual a zero. A equação 3.47 pode ser implementada pelo circuito registrador de deslocamento esboçado na figura 3.3.1, em que a chave comutadora é controlada pela seqüência

$$h\left(a_\sigma^{(r,\,i)},\, b_\sigma^{(r,\,i)} - l\right), \quad \text{com } l = 0, 1, \ldots, b_\sigma^{(r,\,i)} = r + 1 - a_\sigma^{(r,\,i)}.$$

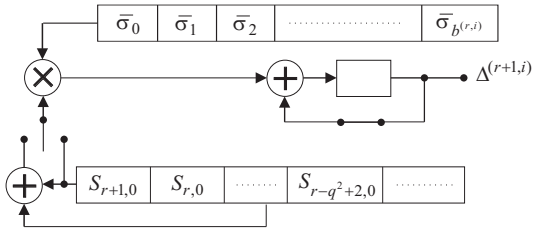

Figura 3.1: Circuito registrador de deslocamento para calcular $\Delta^{(r+1,\,i)}$.

O mesmo circuito é usado para calcular $\hat{\Delta}^{(r+1,\,i)}$ no teorema 45. A única diferença com o cálculo de $\Delta^{(r+1,\,i)}$ é que inicialmente $S_{r+1,\,0}$ é feito igual a zero.

O lema 41 implica que os polinômios $\bar{\lambda}^{(r,\,j)}$ são ciclicamente passados entre as unidades de processamento para o cálculo de $\bar{\sigma}^{(r,\,i)}$. A figura 3.3.1 apresenta um esboço de um circuito que pode ser usado para calcular os polinômios $\bar{\sigma}^{(2\ell-1,\,i)}$. Neste esboço, assume-se que se conheça $S_{a,\,0}$, para $0 \leq a \leq 2\ell-1$. O registrador de síndromes nesta implementação possui comprimento $2\ell+1$ e é inicializado com a seqüência

$$\{S_{0,\,0},\ S_{2\ell-1,\,0},\ S_{2\ell-2,\,0},\ldots,\ S_{1,\,0}\}.$$

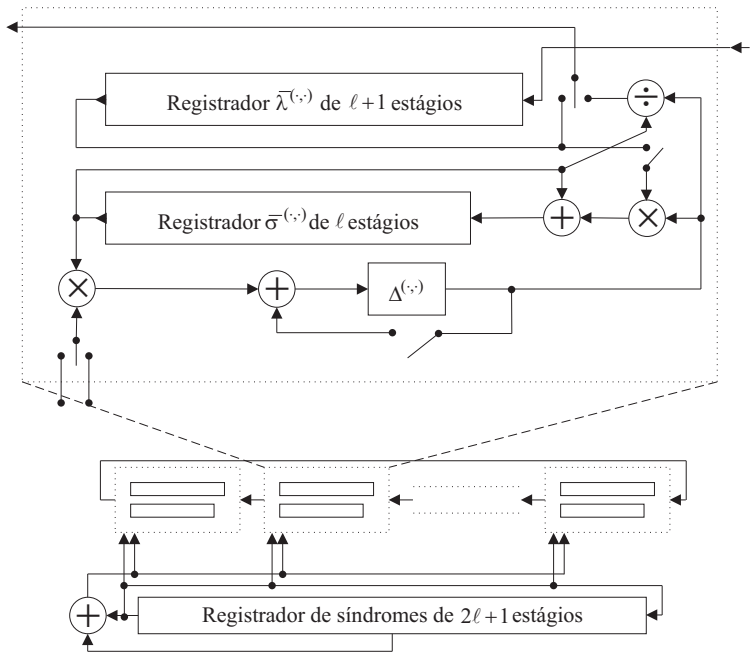

Figura 3.2: Implementação de um algoritmo tipo Berlekamp-Massey para códigos AG sobre curvas de Hermite.

Os registradores para os coeficientes de $\bar{\sigma}^{(r,\,i)}$ possuem comprimento ℓ e são inicializados com um simples 1 na posição mais à esquerda. Os registradores para os coeficientes de

$\bar{\lambda}^{(r,j)}$ possuem comprimento $\ell+1$ e são inicializados com zeros. Por simplicidade, assume-se que o comprimento de todos os registradores é suficiente para armazenar todos os coeficientes dos polinômios $\bar{\sigma}^{(r,i)}$ e $\bar{\lambda}^{(r,j)}$.

Uma iteração típica se inicia pelo cálculo de todos os $\Delta^{(r+1,i)}$. Isto requer ℓ ciclos de relógio. Durante os ℓ ciclos de relógio seguintes, é feita a atualização de $\bar{\sigma}^{(r,i)}$. Ao mesmo tempo, de acordo com o lema 41, os conteúdos dos registradores de $\bar{\lambda}^{(r,j)}$ são substituídos ciclicamente. A multiplicação de $\bar{\lambda}^{(r,j)}$, respectivamente $\bar{\sigma}^{(r,i)}$, por z é realizada simultaneamente, uma vez que os registradores $\bar{\lambda}^{(r,j)}$ possuem comprimento $\ell+1$. De forma similar à aplicada aos registradores de síndromes de comprimento $2\ell+1$, após 2ℓ ciclos de relógio, os conteúdos dos registradores estão no estado apropriado para iniciar uma nova iteração.

De modo a incluir o algoritmo de decisão por maioria do teorema 45, o circuito da figura 3.3.1 tem de ser alterado como mostrado na figura 3.3.1. Deseja-se determinar os polinômios $\bar{\sigma}^{(2\ell-1,i)}$, com o conhecimento inicial de $S_{a,0}$, para $0 \leq a \leq m$ e $m < 2\ell-1$. O registrador de síndromes é inicializado com todos os $S_{a,0}$ e $2\ell-m$ zeros adicionais, ou seja, com a seqüência

$$\{S_{0,0},\, 0,\, 0, \ldots,\, 0,\, S_{m,0},\, S_{m-1,0}, \ldots,\, S_{1,0}\}.$$

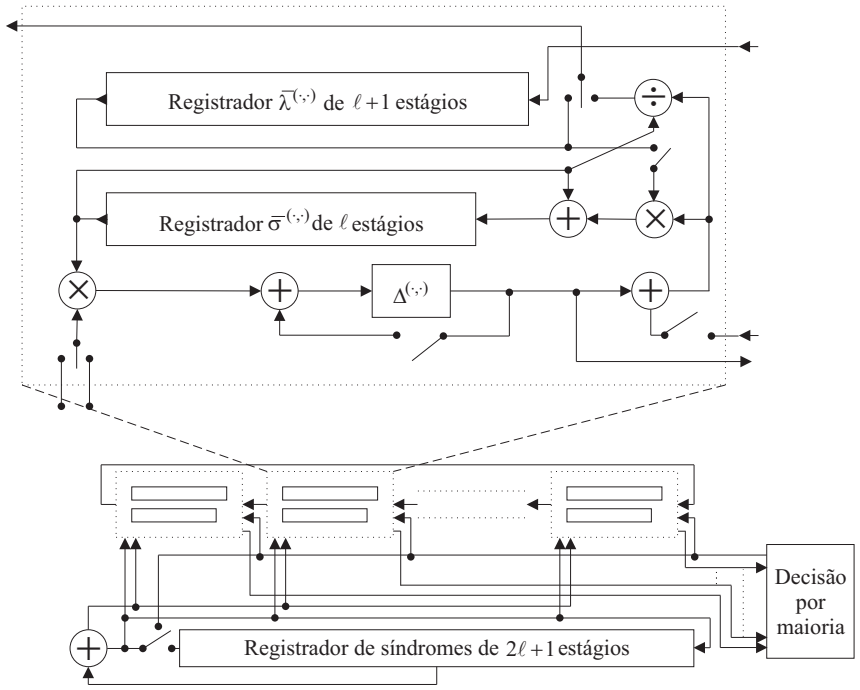

Figura 3.3: Implementação de um algoritmo tipo Berlekamp-Massey para códigos AG sobre curvas de Hermite com decisão por maioria.

Durante as primeiras m iterações, o circuito trabalha exatamente como o circuito da

figura 3.3.1. Na $(m+1)$-ésima iteração, o conjunto de $\hat{\Delta}^{(m+1,\,i)}$ é calculado automaticamente, alimentando um dispositivo de decisão por maioria, que calcula e armazena $S_{m+1,\,0}$. Os $\hat{\Delta}^{(m+1,\,i)}$ são, então, modificados de acordo com a equação

$$\Delta^{(m+1,\,i)} = \hat{\Delta}^{(m+1,\,i)} + S_{m+1,\,0}$$

e a atualização é feita como antes. Além disso, o valor correto de $S_{m+1,\,0}$ é fornecido para o cálculo de $\hat{\Delta}^{(m+2,\,i)}$ quando necessário, alimentando o registrador de síndromes em seguida.

Os comprimentos dos registradores das implementações das figuras 3.3.1 e 3.3.1 são escolhidos de modo a simplificar o "timing" do circuito. Por outro lado, sabe-se que as síndromes $S_{a,\,0}$ são zero para todo $a \in G$. Às custas de um "timing" mais complicado, pode ser usado um registrador de síndromes menor. Considerações similares permitem utilizar registradores mais curtos para os coeficientes dos polinômios $\bar{\sigma}^{(r,\,i)}$ e $\bar{\lambda}^{(r,\,j)}$.

Os circuitos esboçados descrevem uma implementação para códigos AG definidos sobre curvas de Hermite. As diferenças em implementações usando outras curvas são relativamente pequenas. Tem-se uma inicialização diferente para $a_{\sigma}^{(-1,\,i)}$ e $a_{\lambda}^{(-1,\,j)}$, devido ao diferente conjunto de lacunas G. A implementação é também afetada no número de unidades de processamento paralelo. Difere também o número de relações necessárias para descrever a matriz S, dadas as entradas $S_{a,\,0}$, com $a \geq 0$. No caso dos códigos AG sobre curvas de Hermite, existe apenas uma relação trivial dada pelo lema 46. No caso geral, tem-se no máximo $\gamma(\gamma-1)/2$ (normalmente muito menos) relações diferentes entre as entradas em $S_{a,\,b}$.

Exemplo usando código de Hermite

O exemplo apresentado aqui é simples e tem por objetivo apenas ilustrar o funcionamento do algoritmo de Kötter.

Considere a curva de Hermite dada pela equação

$$\mathcal{X} : y^2 + y = x^3$$

de gênero $g = 1$, definida sobre o corpo finito $\mathbb{F}_4$. Esta curva apresenta o ponto projetivo $Q = (0 : 1 : 0)$ no infinito e os 8 pontos afins

$$P_i \in \{(0,\,0), (1,\,\alpha), (\alpha,\,\alpha), (\alpha^2,\,\alpha), (1,\,\alpha^2), (\alpha,\,\alpha^2), (\alpha^2,\,\alpha^2), (0,\,1)\}.$$

Uma observação importante é que, da forma como é inicializado o algoritmo de Kötter, todos os polinômios localizadores de erros possuem sempre um termo independente 1. Portanto, o ponto $(0,\,0)$ jamais é solução do sistema. Deve-se, então, excluir o ponto

(0, 0) da definição do código usado. No caso do código de Hermite, seu comprimento será $n = q^3 - 1$.

Escolhendo $m = 3$, obtém-se o conjunto de lacunas $G = \{1\}$ e a seguinte base para o espaço de funções $L(3Q)$

$$B(3) = \{\phi_0 = 1, \phi_2 = x, \phi_3 = y\}.$$

O código usado $C_\Omega(D, 3Q)$, então, possui comprimento 7, dimensão 3 e distância mínima 4. Sua capacidade de correção é de apenas 1 erro.

Considere uma palavra recebida $\vec{v}$ e as as seguintes síndromes calculadas a partir dela

$$S_{0,0} = \alpha, \quad S_{2,0} = \alpha^2 \quad \text{e} \quad S_{3,0} = \alpha^2.$$

Estas síndromes correspondem a um vetor erro de peso 1 com erro na posição (α, α) de valor α. A parte conhecida da matriz de síndromes, com $a + b < 3$, pode ser calculada usando o lema 46, em que

$$S_{a,b} = \begin{cases} 0, & (a \in G) \vee (b \in G) \\ S_{a+b,0}, & (a \pmod 2) + (b \pmod 2) < 2 \\ S_{a+b,0} - S_{a+b-3,0}, & \text{caso contrário.} \end{cases}$$

Obtém-se, então, a matriz

$$S = \begin{pmatrix} \alpha & 0 & \alpha^2 & \alpha^2 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \alpha^2 & 0 & ? & ? & \cdots \\ \alpha^2 & 0 & ? & ? & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

A tabela 3.3.1 apresenta os resultados intermediários do processamento do algoritmo de Kötter sobre esta matriz S.

As 3 primeiras iterações do algoritmo foram processadas usando a rotina geral, já que eram conhecidos previamente as 3 primeiras síndromes. Para processar a quarta e última iteração, teve de se usar o esquema de decisão por maioria de modo a determinar a síndrome $S_{4,0}$ desconhecida e, daí, obter os valores das discrepâncias $\Delta^{(4,i)}$.

Como $a_\sigma^{(3,1)} = 1 \in G$, apenas o $\hat{\Delta}^{(4,0)} = 1$ foi calculado. Portanto, obteve-se $S_{4,0} = 1$ e $\Delta^{(4,0)} = 0$ (o valor de $\Delta^{(4,1)}$ é 0, já que $a_\sigma^{(3,1)} \in G$).

As funções $\sigma^{(4,i)}$ são os polinômios localizadores de erros procurados. Tem-se, então, o sistema

$$\begin{cases} \sigma^{(4,0)} = \alpha z^2 + 1 \Leftrightarrow \sigma^{(4,0)} = \phi_2 + \alpha \\ \sigma^{(4,1)} = \alpha z^3 + 1 \Leftrightarrow \sigma^{(4,1)} = \phi_3 + \alpha \end{cases} \Rightarrow \begin{cases} x + \alpha \\ y + \alpha \end{cases}$$

Tabela 3.2: Resultados intermediários do processamento do algoritmo de Kötter para o exemplo da seção 3.3.1.

r	$a_\sigma^{(r,i)}$	$a_\lambda^{(r,i)}$	$\sigma^{(r,i)}$	$\lambda^{(r,i)}$	$\Delta^{(r+1,i)}$
-1	$[0, -3]$	$[-2, 1]$	$[1, 1]$	$[0, 0]$	$[\alpha, 0]$
0	$[-1, -2]$	$[0, 1]$	$[1, 1]$	$[\alpha^2 z, 0]$	$[0, 0]$
1	$[0, -1]$	$[0, 1]$	$[1, 1]$	$[\alpha^2 z^2, 0]$	$[\alpha^2, 0]$
2	$[1, 0]$	$[0, 1]$	$[\alpha z^2 + 1, 1]$	$[\alpha^2 z^3, 0]$	$[0, \alpha^2]$
3	$[2, 1]$	$[0, 1]$	$[\alpha z^2 + 1, \alpha z^3 + 1]$	$[\alpha^2 z^4, 0]$	$[0, 0]$
4	$[3, 2]$	$[0, 1]$	$[\alpha z^2 + 1, \alpha z^3 + 1]$	$[\alpha^2 z^5, 0]$	$[?, ?]$

Observa-se facilmente que o único zero deste sistema é o ponto (α, α). O valor do erro é obtido facilmente por um sistema semelhante ao do calculo das síndromes (transformada inversa).

3.4 Conclusões

No presente capítulo, foram apresentadas as principais abordagens na decodificação dos códigos AG, a saber: do algoritmo básico, do algoritmo de Porter (abordagem do algoritmo de Euclides), da decodificação de lista e da decodificação a decisão suave. Foi ainda descrito o estado da arte sobre implementações de decodificadores para os códigos AG, com uma descrição detalhada do algoritmo de Kötter, principal trabalho nesta área.

O objetivo do presente capítulo foi expor sucintamente as diferentes abordagens de decodificação dos códigos AG e a questão da implementação destes decodificadores, de modo a contextualizar e fundamentar a discussão do algoritmo de decodificação de O'Sullivan apresentado no capítulo 5, para o qual é descrita uma nova arquitetura de implementação em hardware.

O capítulo que segue aborda a questão da implementação em hardware de unidades aritméticas em corpo finito, necessárias à implementação do decodificador de O'Sullivan, uma vez que num processor de codificação / decodificação de códigos AG, todas as operações são realizadas em corpo finito.

Capítulo 4

Unidades Aritméticas em Corpo Finito

Este capítulo é o resultado do estudo, análise e implementação em VHDL[1] de arquiteturas para unidades aritméticas em corpos finitos de característica 2 [3, 45, 51, 57, 73]. Estas unidades são a base para a implementação de sistemas de codificação / decodificação de canal, uma vez que todas as operações aritméticas nestes sistemas são feitas em corpos finitos de característica 2. A restrição para corpos de característica 2 se deve à natureza binária dos circuitos digitais.

Uma arquitetura para unidades aritméticas em corpo finito pode ser serial ou paralela. Uma arquitetura paralela produz as saídas a partir de dados de entrada normalmente no período de um único ciclo de relógio. São circuitos tipicamente combinatórios, que favorecem o tempo de processamento em detrimento da complexidade espacial (recursos materiais). Por outro lado, uma arquitetura serial utiliza em geral menos recursos materiais, mas necessita de vários ciclos de relógio para produzir as saídas a partir das entradas. Como conseqüência, em uma arquitetura serial existem sempre elementos de memória, como flip-flops e registradores, o que implica a necessidade de um controle mais complexo em comparação com os circuitos paralelos.

Uma vez que o objetivo aqui é produzir uma biblioteca de base para implementações de codificadores e decodificadores de canal, optou-se pela utilização exclusiva de arquiteturas paralelas para unidades aritméticas em corpo finito de modo a responder às altas exigências em potência de cálculo características deste tipo de processamento. Além disso, buscou-se sempre que possível arquiteturas que permitissem uma descrição genérica, que pudesse ser utilizada para qualquer corpo finito $\mathbb{F}_{2^m}$, em que $m \geq 2$.

Este capítulo descreve as arquiteturas genéricas para operadores de adição, subtração,

[1] Todos os operadores aritméticos foram descritos em VHDL, todavia um operador menos complexo não genérico de inversão utilizando o esquema de inversão direta foi descrito em Verilog.

multiplicação, divisão e inversão para corpos finitos $\mathbb{F}_{2^m}$ quaisquer, sendo m um inteiro positivo, além de apresentar como exemplo arquiteturas para o caso particular do corpo $\mathbb{F}_{16}$, que foram utilizadas na implementação da arquitetura descrita no capítulo seguinte.

Para facilitar o acesso do leitor à discussão das arquiteturas para unidades aritméticas em corpo finito apresentada neste capítulo, o apêndice A apresenta de forma resumida conceitos básicos da álgebra de corpos finitos necessários ao entendimento desta discussão e dos esquemas de codificação e decodificação, objetos principais deste trabalho.

4.1 Operadores de adição e subtração

Considere um corpo finito $\mathbb{F}_{2^m}$ e uma base $\{\gamma_0, \gamma_1, \ldots, \gamma_{m-1}\}$ para este corpo. A operação de adição de dois elementos A, $B \in \mathbb{F}_{2^m}$, em que $A = \alpha_0\gamma_0 + \alpha_1\gamma_1 + \cdots + \alpha_{m-1}\gamma_{m-1}$ e $B = \beta_0\gamma_0 + \beta_1\gamma_1 + \cdots + \beta_{m-1}\gamma_{m-1}$, é feita simplesmente pela adição de seus coeficientes um a um. Em outras palavras,

$$A + B = (\alpha_0 + \beta_0)\gamma_0 + (\alpha_1 + \beta_1)\gamma_1 + \cdots + (\alpha_{m-1} + \beta_{m-1})\gamma_{m-1}, \tag{4.1}$$

sendo $\alpha_i, \beta_i \in \mathbb{F}_2$.

A adição de elementos em $\mathbb{F}_2$ obedece a tabela da verdade descrita na tabela 4.1. Esta lógica pode ser implementada por uma única porta XOR. Conseqüentemente, a adição de A e B (*cf.* (4.1)) pode ser implementada por um conjunto de m portas XOR, segundo a figura 4.1.

+	0	1
0	0	1
1	1	0

Tabela 4.1: Tabela da verdade relativa à operação de adição no corpo finito $\mathbb{F}_2$.

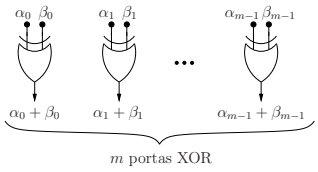

Figura 4.1: Arquitetura para um operador de adição de dois elementos A e B em $\mathbb{F}_{2^m}$.

A figura 4.2 apresenta o módulo GFADD, operador de adição para o caso particular de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5.

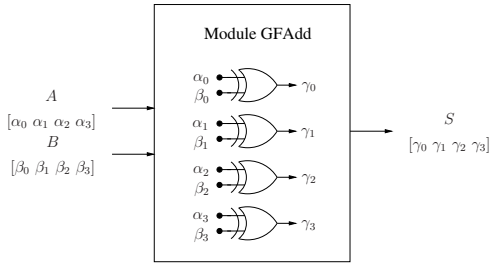

Figura 4.2: Arquitetura para o módulo GFADD, operador de adição de dois elementos A e B para o caso particular de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5.

No que diz respeito à operação de subtração em corpos finitos de característica 2, elas são equivalentes à operação de adição, uma vez que a inversa aditiva de um elemento $\alpha \in \mathbb{F}_{2^m}$ é ele mesmo, ou seja, $-\alpha = \alpha$ (*cf.* apêndice A).

4.2 Operador de multiplicação

A operação de multiplicação é a mais explorada no que se refere ao desenvolvimento de arquiteturas para unidades aritméticas em corpos finitos. Além disso, esta operação é habitualmente considerada predominante na determinação da complexidade algorítmica dos decodificadores. Na verdade, esta complexidade é normalmente indicada como o número de multiplicações efetuadas pelo algoritmo necessárias para a obtenção do resultado desejado.

As arquiteturas para multiplicação em corpos finitos encontradas na literatura podem ser classificadas em três categorias, segundo a base utilizada para representar os elementos do corpo finito:

- Multiplicadores de base canônica;
- Multiplicadores de base normal;
- Multiplicadores de base dual.

No que se refere aos circuitos multiplicadores paralelos, algumas arquiteturas foram propostas recentemente [27, 45, 51, 57, 75, 78]. Para escolher a arquitetura a implementar, foram levadas em consideração a *complexidade material* (número total de portas) e o *caminho crítico* (número máximo de portas a serem percorridas pelos dados de entrada) do circuito. Além disso, foi considerada ainda a eventual necessidade de introduzir etapas de conversão de base, por exemplo caso sejam adotados circuitos inversores ou divisores de uma base diferente à utilizada pelo multiplicador. Observando todos estes fatores,

optamos por adotar o multiplicador paralelo de base canônica de Mastrovito, que consiste numa paralelização do multiplicador serial MSR (do inglês *modified shift register*) proposto também por ele [27, 45, 78].

4.2.1 Multiplicador de Mastrovito

Considere um corpo finito $\mathbb{F}_{2^m}$ gerado por um polinômio primitivo $P(x) = p_0 + p_1 x + \cdots + p_{m-1}x^{m-1} + x^m$ e uma base canônica para este corpo finito. Considere $A(x) = \alpha_0 + \alpha_1 x + \cdots + \alpha_{m-1}x^{m-1}$ e $B(x) = \beta_0 + \beta_1 x + \cdots + \beta_{m-1}x^{m-1}$ dois elementos de $\mathbb{F}_{2^m}$.

Considere $C(x) = \gamma_0 + \gamma_1 x + \cdots + \gamma_{m-1}x^{m-1}$ o produto de $A(x)$ e $B(x)$, tal que

$$\begin{aligned} C(x) &= A(x)B(x) \mod P(x) \\ &= \left[\beta_0 A(x) + \beta_1 x A(x) + \cdots + \beta_{m-1}x^{m-1}A(x)\right] \mod P(x) \\ &= \beta_0 \left[A(x) \mod P(x)\right] + \beta_1 \left[xA(x) \mod P(x)\right] + \cdots \\ &\quad + \beta_{m-1}\left[x^{m-1}A(x) \mod P(x)\right]. \end{aligned} \tag{4.2}$$

Defina agora os polinômios

$$\begin{aligned} Z_j(x) &= x^j A(x) \mod P(x) \\ &= \sum_{i=0}^{m-1} f_{i,j}x^i, \quad j = 0, 1, \ldots, m-1, \end{aligned} \tag{4.3}$$

em que $f_{i,j}x^i \in \mathbb{F}_2$. Conseqüentemente, pode-se escrever (4.2) como sendo

$$C(x) = \beta_0 Z_0(x) + \beta_1 Z_1(x) + \cdots + \beta_{m-1}Z_{m-1}(x).$$

Em notação matricial, obtém-se

$$C = \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{m-1} \end{pmatrix} = \begin{pmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,m-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m-1,0} & f_{m-1,1} & \cdots & f_{m-1,m-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{m-1} \end{pmatrix} = ZB. \tag{4.4}$$

Denota-se por Z a *matriz de produto*, que depende unicamente do polinômio $A(x)$ e do polinômio gerador $P(x)$.

Para obter os coeficientes $f_{i,j}$, considere inicialmente a *matriz de redução* Q dada por

$$\begin{pmatrix} x^m \\ x^{m+1} \\ \vdots \\ x^{2m-2} \end{pmatrix} \mod P(x) = Q \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{m-1} \end{pmatrix} = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,m-1} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m-2,0} & q_{m-2,1} & \cdots & q_{m-2,m-1} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ \vdots \\ x^{m-1} \end{pmatrix}.$$

Mostra-se que os coeficientes $f_{i,j}$ de Z (*cf.* 4.4) são dados pelas equações

$$f_{i,0} = \alpha_i, \quad i = 0, 1, \ldots, m-1, \tag{4.5}$$

e

$$f_{i,j} = \sigma(i-j)\alpha_{i-j} + \sum_{t=0}^{j-1} q_{j-1-t}\alpha_{m-1-t}, \quad i = 0, \ldots, m-1,\ j = 1, \ldots, m-1, \tag{4.6}$$

em que $\sigma(k)$ é a função grau definida por

$$\sigma(k) = \begin{cases} 1, & k \geq 0, \\ 0, & k < 0. \end{cases}$$

4.2.2 Realização do multiplicador de Mastrovito

Pode-se dizer que o produto $C(x) = A(x)B(x)$ implementado pelo multiplicador de Mastrovito é realizado por dois subsistemas (*cf.* figure 4.3). O primeiro subsistema, chamado *rede* f, calcula cada coeficiente $f_{i,j}$ a partir dos coeficientes α_i de $A(x)$ e p_i de $P(x)$. O segundo subsistema, chamado *rede IP*, calcula cada bit γ_i de $C(x)$ através do produto interno (*cf.* (4.4))

$$\gamma_i = \beta_0 f_{i,0} + \beta_1 f_{i,1} + \cdots + \beta_{m-1} f_{i,m-1}, \quad i = 0, 1, \ldots, m-1. \tag{4.7}$$

Uma rede IP consiste num conjunto de m células idênticas, nas quais m portas AND e uma árvore de portas XOR (soma de produtos) realizam o produto interno descrito em (4.7) (*cf.* figura 4.4).

No que concerne a rede f, da definição dos coeficientes $f_{i,j}$, observa-se que cada coluna da matriz Z corresponde aos coeficientes de $x^j A(x) \mod P(x)$ (*cf.* (4.3) e (4.4)). Na

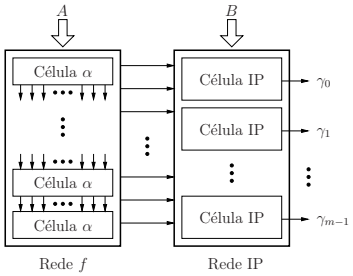

Figura 4.3: Arquitetura geral para um operador paralelo de multiplicação de dois elementos A e B em $\mathbb{F}_{2^m}$, conhecido como multiplicador de Mastrovito.

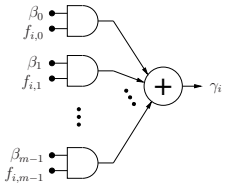

Figura 4.4: Arquitetura geral para células IP utilizadas no multiplicador paralelo de Mastrovito para multiplicação de dois elementos A e B em $\mathbb{F}_{2^m}$.

coluna $j = 0$, observa-se os coeficientes de $A(x)$. Em $j = 1$, obtém-se

$$\begin{aligned} xA(x) \mod P(x) &= \left(x\alpha_0 + x^2\alpha_1 + \cdots + x^{m-1}\alpha_{m-2} + x^m\alpha_{m-1}\right) \mod P(x) \\ &= \left(x\alpha_0 + x^2\alpha_1 + \cdots + x^{m-1}\alpha_{m-2}\right) + \alpha_{m-1}\left(1 + xp_1 + \cdots + x^{m-1}p_{m-1}\right) \\ &= \alpha_{m-1} + \sum_{i=1}^{m-1} x^i \left(\alpha_{m-1}p_i + \alpha_{i-1}\right). \end{aligned}$$

Portanto, os coeficientes de $xA(x) \mod P(x)$ são obtidos por uma cadeia, chamada célula α, de portas XOR interligadas segundo o polinômio $P(x)$ utilizado: uma porta XOR associada a cada ocorrência de um bit $p_i = 1$ para realizar a adição $\alpha_{m-1} + \alpha_{i-1}$. Observa-se que todos os coeficientes de $x^jA(x) \mod P(x)$ são obtidos da mesma forma, uma vez que $x^2A(x) \mod P(x) = x(xA(x) \mod P(x)) \mod P(x)$, e assim por diante. Conseqüentemente, uma rede f pode ser realizada por arranjo de $m-1$ células α (*cf.* figura 4.5).

O multiplicador de Mastrovito descrito apresenta uma complexidade material C_M que depende diretamente do polinômio gerador $P(x)$ adotado e obedece os limitantes

$$2m^2 - 1 \leq C_M \leq 3m^2 - 3m + 1, \tag{4.8}$$

em que a complexidade da rede IP é sempre $m(2m-1)$ e a complexidade devida à rede f varia de $m-1$ a $(m-1)(m-1)$, de acordo com $P(x)$.

O comprimento do caminho crítico L_C depende também de $P(x)$ e obedece os limitantes

$$2 + \lceil \log_2 m \rceil \leq L_C \leq m + \lceil \log_2 m \rceil, \tag{4.9}$$

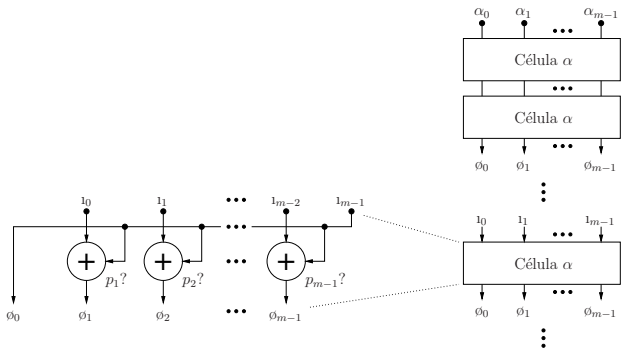

Figura 4.5: Arquitetura geral para células α utilizadas no multiplicador paralelo de Mastrovito para multiplicação de dois elementos A e B em $\mathbb{F}_{2^m}$. Cada adição é implementada segundo os coeficientes do polinômio gerador $P(x)$ ($p_i = 1$ ou 0).

em que o caminho crítico da rede IP é sempre igual a $1 + \lceil \log_2 m \rceil$ e o comprimento do caminho crítico devido à rede f varia de 1 a $m - 1$.

Eventualmente, algumas saídas da rede f são iguais. Mastrovito considerou este fato e eliminou os elementos que geravam saídas repetidas. Contudo, uma vez que optou-se por descrever genericamente as unidades aritméticas, ou seja, uma vez que optou-se por obter códigos VHDL que implementam qualquer corpo $\mathbb{F}_{2^m}$ dado m como parâmetro de projeto, não se pôde tomar proveito dessas otimizações. Por esta razão, o limite inferior de (4.8) e o limite superior de (4.9) diferem algumas vezes dos limites apresentados originalmente por Mastrovito em [45]. Apesar disso, este mesmo procedimento de otimização pode ser facilmente adotado para gerar operadores menos complexos para casos particulares de $\mathbb{F}_{2^m}$.

A figura 4.6 apresenta o módulo GFMULTIPLY, operador de multiplicação para o caso particular de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5. Pode-se identificar à esquerda do circuito 3 portas XOR que representam a rede f do circuito. A rede IP consiste nos 4 circuitos de soma de produtos à direita.

Observe que o desempenho e a complexidade do multiplicador descrito dependem fortemente da escolha de $P(x)$. Mostra-se em [27] que os trinômios

$$x^m + x^k + 1$$

e os polinômios igualmente espaçados

$$x^{ns} + x^{ns-1} + \cdots + x^{2s} + x^s + 1$$

proporcionam os melhores desempenhos e menores complexidades do multiplicador de Mastrovito. Infelizmente, estes polinômios só existem para alguns corpos finitos.

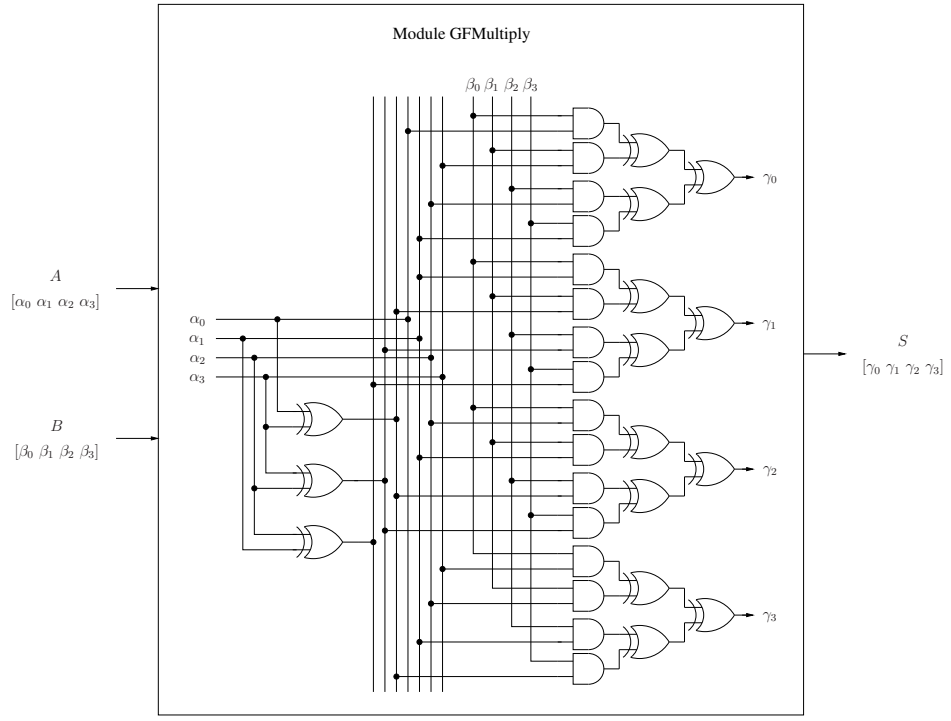

Figura 4.6: Arquitetura para o módulo GFMULTIPLY, operador de multiplicação de dois elementos A e B para o caso particular de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5.

O apêndice D apresenta polinômios primitivos de peso mínimo para $2 \leq m \leq 202$, que podem ser usados como polinômios $P(x)$ no projeto de multiplicadores. Estes polinômios proporcionam bons resultados em termos de complexidade e desempenho.

4.3 Operador de divisão

A operação de divisão de um elemento α por um elemento β em um corpo finito $\mathbb{F}_{2^m}$ é habitualmente visto como uma multiplicação de α pelo inverso multiplicativo de β (este inverso multiplicativo é único), ou seja,

$$\frac{\alpha}{\beta} = \alpha\beta^{-1}. \tag{4.10}$$

Materialmente, a operação de divisão é realizada sob a forma de (4.10).

A operação de inversão em um corpo finito constitui uma tarefa relativamente complexa. Em geral, ela é restrita às aplicações de codificação de canal, nas quais os corpos finitos utilizados são pequenos.

4.4 Operador de inversão

Considere um corpo finito $\mathbb{F}_{2^m}$. A implementação da operação de inversão de um elemento $\alpha \in \mathbb{F}_{2^m}$ é habitualmente realizada tomando-se proveito da seguinte propriedade:

$$\begin{aligned} \alpha &= \alpha^{2^m} \Rightarrow \\ \alpha^{-1} &= \alpha^{2^m-2}. \end{aligned} \tag{4.11}$$

Observa-se na literatura algumas arquiteturas baseadas no resultado apontado por (4.11) [45, 51]. A este tipo de operador dá-se o nome de *inversor por exponenciação*. O inversor por exponenciação apresenta alta complexidade algorítmica, uma vez que ele normalmente faz uso de cadeias em seqüência de multiplicadores.

Uma inversão otimizada, de complexidade relativamente baixa, pode ser obtida pela chamada *inversão direta*. Entretanto, esta arquitetura não apresenta estrutura regular e só pode ser descrita para um corpo finito específico, ou seja, ela não pode ser codificada genericamente, para qualquer m, como no caso do multiplicador de Mastrovito.

Observou-se, contudo, que o VHDL, linguagem adotada para implementação de todo o projeto, não permite implementar diferentes versões do inversor direto para uma única entidade GFINVERT e escolher qual versão utilizar baseado num parâmetro constante m. Esta impossibilidade deve-se diretamente à incompatibilidade do número de bits das portas de entrada e saída da entidade, parametrizado por m, com as diferentes implementações do inversor direto, que apresentam diferentes números de bits. Uma solução para este problema foi a adoção do Verilog em lugar do VHDL para a implementação do inversor direto para alguns casos mais usados. O Verilog apresenta diretivas de compilação `` `ifdef `` que permitem selecionar a parte do código a ser compilada e sintetizada.

A próxima seção descreve o inversor por exponenciação, implementado como caso genérico. Em seguida, é descrito o inversor direto, implementado para alguns valores mais utilizados de m.

4.4.1 Inversor por exponenciação

De (4.11), conclui-se que a operação de inversão α^{-1} é equivalente à exponenciação pelo expoente $2^m - 2$. A representação binária de um inteiro $2^m - 1$ é [1 1 ... 1] (vetor de m bits '1'). Em outras palavras,

$$\begin{aligned} 2^m - 1 &= 2^0 + 2^1 + 2^2 + \cdots + 2^{m-1} \Rightarrow \\ 2^m - 2 &= 2 + 2^2 + \cdots + 2^{m-1}. \end{aligned}$$

Portanto, considerando (4.11),

$$\alpha^{-1} = \alpha^2\alpha^{2^2}\alpha^{2^3}\dots\alpha^{2^{m-1}} \Rightarrow$$
$$\alpha^{-1} = (\dots(((\alpha)^2\alpha)^2\alpha)^2\dots\alpha)^2, \tag{4.12}$$

em que α aparece $m-1$ vezes.

Por (4.12), a inversa de α pode ser realizada por uma cadeia de quadrados e multiplicações (*cf.* figure 4.7).

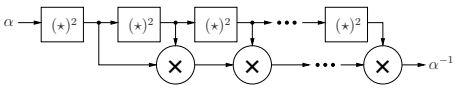

Figura 4.7: Arquitetura geral para um operador de inversão de um elemento α em $\mathbb{F}_{2^m}$, conhecido como inversor por exponenciação.

A operação de quadrado $(\star)^2$ foi implementada na forma de uma simples multiplicação de um elemento por ele mesmo. Existem arquiteturas para a operação quadrado para alguns casos particulares, entretanto a única razão para implementar um inversor por exponenciação, que apresenta alta complexidade, como já foi afirmado, é de fazê-lo genérico para que sirva como opção ao inversor direto, que apresenta complexidade menor e é descrito a seguir.

4.4.2 Inversor direto

Considere a matriz de produto Z de (4.4), derivada do elemento $A \in \mathbb{F}_{2^m}$. Observe que a primeira coluna de Z é composta pelos coeficientes de A.

A matriz Z é inversível. Sua inversa Z^{-1} corresponde à operação de multiplicação pelo elemento A^{-1} e conseqüentemente apresenta na primeira coluna os coeficientes de A^{-1}. Portanto, encontrando-se a primeira coluna à esquerda de Z^{-1}, encontra-se as equações booleanas pelas quais calcula-se A^{-1}, uma equação para cada bit de A^{-1}.

O inversor direto apresenta complexidade material relativamente baixa, contudo também apresenta baixa regularidade. Ele consiste na realização das m equações booleanas (uma para cada bit de A^{-1}) obtidas pela inversão de Z considerando o elemento A uma incógnita. Estas equações booleanas são sempre do tipo $\alpha_i\dots\alpha_j \oplus \dots \oplus \alpha_u\dots\alpha_v$ (*cf.* figura 4.8), e, sendo $\alpha_i \in \mathbb{F}_2$ os coeficientes de A e $0 \leq i, j, u, v \leq m-1$.

A figura 4.9 apresenta o módulo GFINVERT, operador de inversão direta para o caso de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5.

A inversão da matriz Z necessária para determinação das equações booleanas que implementam a operação A^{-1} foi realizada com o uso da ferramenta Macaulay 2, que suporta

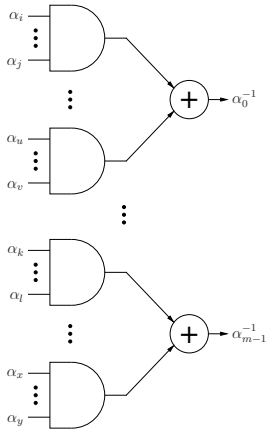

Figura 4.8: Arquitetura geral para um operador de inversão de um elemento A em $\mathbb{F}_{2^m}$, conhecido como inversor direto. Ela consiste basicamente numa cadeia de m somas de produtos.

cálculos da geometria algébrica e da álgebra comutativa (*cf.* http://www.math.uiuc.edu/macaulay2).

4.5 Conclusões

O presente capítulo descreveu as arquiteturas adotadas para as unidades aritméticas em corpo finito de característica 2. Foram descritas arquiteturas para operadores de adição, multiplicação, divisão e inversão, adaptadas a qualquer corpo finito $\mathbb{F}_{2^m}$, em que $m \geq 2$.

Estas unidades aritméticas são necessárias para implementações de sistemas de codificação / decodificação de canal. Desta forma, este trabalho teve por objetivo produzir uma biblioteca de unidades aritméticas de base para implementações de codificadores e decodificadores de canal. Optou-se pela utilização exclusiva de arquiteturas paralelas para as unidades aritméticas de modo a responder às altas exigências em potência de cálculo características dos decodificadores para códigos AG. Além disso, buscou-se sempre que possível arquiteturas que permitissem uma descrição genérica, que pudesse ser utilizada para qualquer corpo finito $\mathbb{F}_{2^m}$.

A arquitetura escolhida para uma descrição genérica do operador de multiplicação foi o multiplicador paralelo de Mastrovito [45]. O multiplicador descrito difere ligeiramente da proposta original de Mastrovito. Ele apresenta complexidade e caminho crítico que podem ser ligeiramente maiores que no multiplicador original. Mastrovito considera o fato de que algumas saídas da rede f são eventualmente iguais e elimina os elementos que geram saídas repetidas. No multiplicador descrito aqui, optou-se por obter códigos VHDL que implementam qualquer corpo $\mathbb{F}_{2^m}$ dado m como parâmetro de projeto, não se

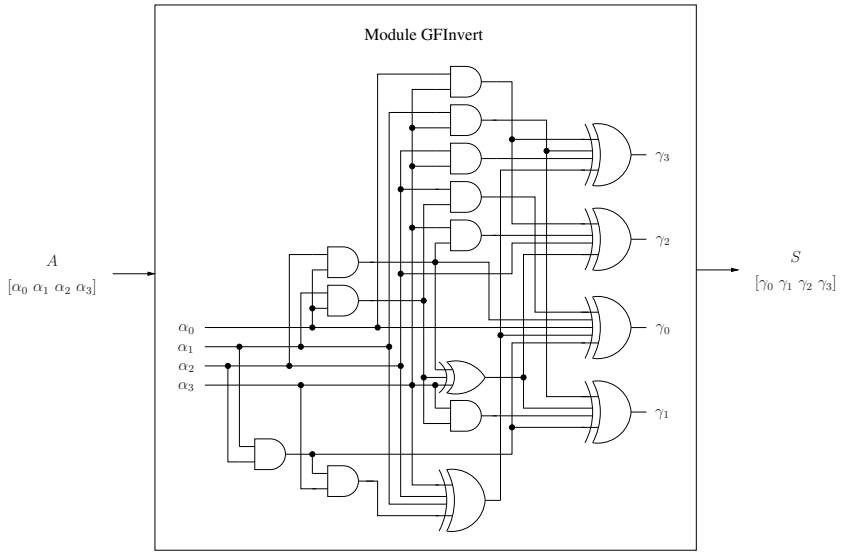

Figura 4.9: Arquitetura para o módulo GFINVERT, operador de inversão direta de um elemento A para o caso de $\mathbb{F}_{16}$, que foi utilizado na implementação da arquitetura do decodificador proposta no capítulo 5.

podendo tomar proveito dessas otimizações. Apesar disso, este mesmo procedimento de otimização pode ser facilmente adotado para gerar operadores menos complexos para casos particulares de $\mathbb{F}_{2^m}$. Esta descrição genérica para o multiplicador, de custo eventualmente mais alto (não sempre), mas de síntese automática, constitui uma alternativa à solução *ad hoc* de Mastrovito, de custo algumas vezes mais baixo, mas que exige uma concepção caso a caso.

O operador de inversão escolhido para uma descrição genérica foi o inversor por exponenciação proposto por Mastrovito em [45]. Este inversor apresenta alta complexidade e foi adotado apenas como opção ao chamado inversor direto. O inversor direto, por sua vez, apresenta complexidade relativamente pequena, mas não pode ser descrito de forma genérica. Ele foi descrito em Verilog para os casos particulares de corpos finitos $\mathbb{F}_{2^m}$, em que $2 \leq m \leq 10$.

O apêndice D fornece alguns polinômios primitivos de peso mínimo a serem utilizados como polinômios geradores na definição dos parâmetros de projeto das unidades aritméticas implementadas.

O capítulo seguinte apresenta o algoritmo de decodificação de O'Sullivan para códigos de Hermite. Em seguida, ele descreve uma nova arquitetura para este decodificador.

Capítulo 5

Decodificador de O'Sullivan

O algoritmo de decodificação de códigos AG de O'Sullivan proposto inicialmente em 1995 baseia-se na solução de uma equação chave, constituindo a segunda abordagem de decodificação (*cf.* seção 3.2.5) [48]. Esta primeira versão do algoritmo trata iterativamente um conjunto de funções localizadoras de erros, soluções para uma equação chave. Em 2000, O'Sullivan propôs uma versão melhorada deste algoritmo, que trata iterativamente não apenas funções localizadoras, mas também funções avaliadoras de erros [49]. Esta abordagem proporciona uma forma mais simples de determinar os valores dos erros ocorridos.

O presente capítulo apresenta o trabalho realizado de estudo, desenvolvimento e implementação de uma arquitetura para o decodificador de O'Sullivan proposto em [49]. A primeira seção descreve sumariamente o algoritmo de O'Sullivan. Nas seções seguintes, são descritos uma nova arquitetura para este algoritmo, assim como os aspectos relativos à implementação realizada.

5.1 Algoritmo

Um processo completo de decodificação utilizando o algoritmo de O'Sullivan inclui as seguintes tarefas (*cf.* figure 5.1):

a) Cálculo das síndromes (vetor $\vec{S}$) a partir do vetor recebido e da matriz de paridade do código, como descrito na seção B.1.4;

b) Determinação dos polinômios avaliadores ϕ e localizadores f de erros (esta etapa constitui usualmente o algoritmo de decodificação propriamente dito);

c) Determinação do vetor de erros $\vec{e}$ a partir dos polinômios avaliadores e localizadores de erros;

d) Correção do vetor recebido, que consiste basicamente em subtrair deste vetor o vetor erro obtido das etapas anteriores;

e) Mapeamento do vetor corrigido $\vec{c}$ no vetor de informação $\vec{i}$ correspondente.

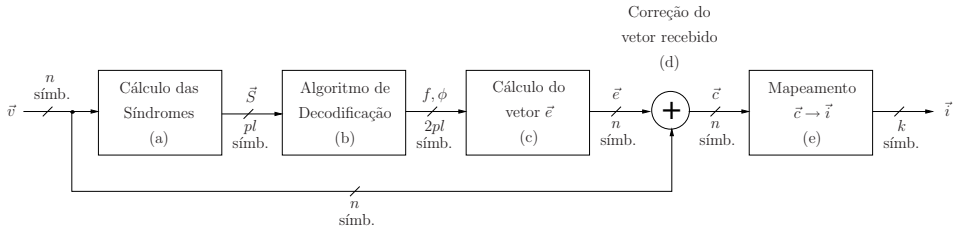

Figura 5.1: Esquema completo de decodificação baseado na abordagem proposta por O'Sullivan [49]. Considera-se um código de bloco (n, k). O comprimento pl de um polinômio é igual ao número máximo de iterações necessárias para processar a decodificação mais 1.

O algoritmo proposto por O'Sullivan em [49] concerne unicamente à tarefa (b) na figura 5.1. Em geral, os algoritmos de decodificação propostos são restritos à tarefa de determinação das posições de ocorrência dos erros, considerada o núcleo de um processo de decodificação de códigos de bloco (*cf.* seção 3.1). As demais tarefas são consideradas problemas de soluções conhecidas. Da mesma forma, a arquitetura para este decodificador apresentada neste capítulo corresponde estritamente ao algoritmo de O'Sullivan e, portanto, à tarefa (b) da figura 5.1.

Considere um corpo finito $\mathbb{F}_q$, no qual $q = r^2$ e r é um inteiro positivo. Na prática, para uma implementação em hardware, considera-se sempre r como sendo uma potência de 2, de modo que $\mathbb{F}_q$ seja um corpo de característica 2. A necessidade por corpos finitos de característica 2 deve-se à natureza binária dos circuitos digitais.

Considere um código de Hermite de comprimento n definido pela curva de Hermite dada pela equação $y^r + y = x^{r+1}$ de gênero g, como descrito na seção 2.2. Considere $P_1, \ldots, P_n$ os r^3 pontos racionais da curva e Q o único ponto no infinito no plano projetivo. Considere Λ o conjunto das anti-lacunas em Q (*cf.* (2.13)) e $\theta_k = x^{\mu(k)} y^{\nu(k)}$ um monômio associado à uma ordem de pólo $k = \mu(k)r + \nu(k)(r+1)$ em Q.

Se $R = \mathbb{F}_q[x, y] / (y^r + y = x^{r+1})$ é um anel de polinômios com pólos unicamente nos pontos da curva, uma síndrome $S(f)$ de uma função $f = f_{pl-1}\theta_{pl-1} + \cdots + f_1\theta_1 + f_0 \in R$

qualquer é calculada da forma

$$\begin{aligned} S(f) &= \sum_{k=1}^{n} v_k f(P_k) \\ &= \sum_{k=1}^{n} v_k \sum_{i=0}^{pl-1} f_i \theta_i(P_k) \\ &= \sum_{i=0}^{pl-1} f_i \sum_{k=1}^{n} v_k \theta_i(P_k) \\ &= \sum_{i=0}^{pl-1} f_i \vec{S} \end{aligned} \tag{5.1}$$

em que $\vec{v}$ é o vetor recebido. Observe que $\vec{S} = \sum_{k=1}^{n} v_k \theta_i(P_k)$ constitui a definição convencional do vetor de síndromes calculadas a partir do vetor recebido apresentada na seção B.1.4.

O cálculo iterativo dos polinômios localizadores e avaliadores de erros pelo algoritmo de O'Sullivan é baseado na solução de uma equação chave (*cf.* (5.2)) derivada de uma série de síndromes h_e definida por

$$h_e = \frac{x^r}{x} \sum_{i=0}^{r} \sum_{j=0}^{\infty} s_{ij} x^{-i} y^{-j},$$

em que $0 \leq i \leq r$, $j \geq 0$ e $s_{ij} = S(x^i y^j)$ é uma síndrome calculada por (5.1) considerando $f = x^i y^j$.

O núcleo do algoritmo de O'Sullivan é o corolário 3.11 de [49] que determina que f é uma função localizadora de erros (função que possui zeros nos pontos correspondentes às posições do vetor recebido em que os erros aconteceram) se e só se $fh_e \in R$. Com base neste resultado, o algoritmo procura iterativamente pares de funções $f, \phi \in R$ (funções localizadoras e avaliadoras de erros, respectivamente) que satisfazem a equação chave

$$fh_e = \phi. \tag{5.2}$$

A cada iteração, o algoritmo incrementa a dimensão do espaço de busca por funções e verifica se os pares de funções produzidos na última iteração $\dot{f}$ e $\dot{\phi}$ (um ponto sobre uma função indica que ela foi produzida na última iteração) satisfazem ainda a equação chave (5.2). Se um par $\dot{f}$ e $\dot{\phi}$ não mais satisfaz (5.2), um novo par de funções f e ϕ é calculado a partir de $\dot{f}$ e $\dot{\phi}$ e de funções auxiliares g e ψ. Estas funções auxiliares são na verdade funções f e ϕ, respectivamente, que não satisfizeram a equação chave numa iteração precedente. Elas são úteis porque representam o espaço ortogonal ao de busca das funções desejadas. A atualização das funções, de modo que elas passem a satisfazer

(5.2), constitui o terceiro passo do algoritmo de O'Sullivan (algoritmo 47 descrito nos próximos parágrafos).

Da proposição 4.1 em [49], o vetor de erros $\vec{e}$ pode ser determinado pela equação

$$e_i = \frac{\phi}{f'}(P_i), \tag{5.3}$$

em que f' denota a derivada primeira de f em relação a x, ou seja, df/dx. Esta derivada sobre corpos finitos de característica 2 é calculada seguindo as regras normais de aplicação da derivada e considerando que $dx^k/dx = 0$ se k for par e $dx^k/dx = x^{k-1}$ se k for ímpar. Além disso, da curva de Hermite tem-se que

$$d(y^r + y)/dx = d(x^{r+1})/dx \Rightarrow$$
$$dy/dx = x^r,$$

já que r como potência de 2 é sempre um número par.

Após um número suficiente de iterações (it_{max}), garante-se que entre as funções localizadoras fornecidas pelo algoritmo existe ao menos uma função f tal que f' não se anula num ponto associado a uma posição de um erro. Desta forma, a determinação do vetor erro $\vec{e}$ a partir das funções f e ϕ fornecidas pelo algoritmo (tarefa (c) na figura 5.1) é feita da seguinte forma:

1. Avaliação das funções f para determinar quais as possíveis posições de ocorrências dos erros (posições associadas aos zeros das funções);

2. Determinação das derivadas primeiras f' (materialmente, esta tarefa não apresenta qualquer complexidade para corpos finitos de característica 2);

3. Avaliação das funções $\frac{\phi}{f'}$, para cada par f e ϕ, nos pontos em que f se anula, mas f' não se anula;

4. Os coeficientes e_i do vetor erro $\vec{e}$ consistem na união dos resultados obtidos para cada par f e ϕ.

O algoritmo 47 descrito a seguir apresenta o algoritmo de decodificação de O'Sullivan. Numa iteração it, denota-se por α, $\beta \in \mathbb{F}_q$ valores de síndromes de funções (*cf.* (5.1)), por σ, $\delta \subset \Lambda$ conjuntos de parâmetros de controle (veja parágrafo seguinte), e por f, g, ϕ, $\psi \in R$ as funções localizadoras e avaliadoras de erros e suas respectivas funções auxiliares.

Considere aqui $f \in R$ uma função que não satisfaz a equação chave e, portanto, não localiza os erros ocorridos. Defina

$$\text{span}(f) = \min\{-\nu_Q(g) : S(fg) \neq 0\}$$

como a menor ordem de pólo de uma função g qualquer, tal que $S(fg) \neq 0$, e

$$\text{fail}(f) = -\nu_Q(f) + \text{span}(f)$$

como a menor ordem de pólo de uma função qualquer, maior ou igual que a ordem de pólo de f, cuja síndrome seja não nula.

Numa dada iteração it, define-se um conjunto Σ_{it} na forma

$$\Sigma_{it} = \{-\nu_Q(f) \in \Lambda : \text{fail}(f) > it\}$$

como sendo uma partição do conjunto Λ cujas funções que possuem estas ordens de pólo em Q apresentam $\text{fail}(f) > it$. Em outras palavras, Σ_{it} constitui um conjunto de ordens de pólo de funções f, tal que as menores ordens de pólo das funções múltiplas de f não localizadoras de erros são maiores que it.

Define-se ainda um conjunto Δ_{it} na forma

$$\Delta_{it} = \{\text{span}(g) \in \Lambda : \text{fail}(g) \leq it\}$$

como sendo outra partição de Λ cujas funções g que apresentam estes valores de $\text{span}(g)$ apresentam $\text{fail}(g) \leq it$. Em outras palavras, Δ_{it} constitui o conjunto dos $\text{span}(g)$, tal que as menores ordens de pólo das funções múltiplas de g não localizadoras de erros são menores ou iguais a it.

Os conjuntos Σ_{it} e Δ_{it} particionam o conjunto de anti-lacunas Λ, ou seja,

$$\Delta_{it} = \Lambda \oplus \Sigma_{it}.$$

Os conjuntos σ e δ consistem em mínimos e máximos de Σ e Δ, respectivamente, segundo uma ordenação parcial denotada por $\preceq$, em que $a \preceq b$ se $b - a \in \Lambda$ (*cf.* [49]). Em uma iteração it, a cada função localizadora de erros f está associado um elemento de σ, que representa a ordem de pólo desta função. Já os elementos de δ correspondem às ordens de pólo de cada uma das funções auxiliares g numa dada iteração it.

A cada iteração, síndromes $\alpha = S(\dot{f})$ são calculadas. Com base nos valores destas síndromes, os parâmetros de controle σ e δ são atualizados. Em seguida, utilizando os novos parâmetros de controle, as funções f, g, ϕ e ψ são atualizadas.

Denota-se por $\widetilde{f}$ a parte R de f, ou seja, a função f excluindo-se os termos θ_i nos quais i é uma lacuna. No algoritmo 47, as funções $\max\{\star\}$ e $\min\{\star\}$ correspondem aos máximos e mínimos com relação à ordenação parcial $\preceq$.

Algoritmo 47 (Decodificador de O'Sullivan)

Entradas:

- *O vetor de síndromes* $\vec{S} = [S(\theta_{n-k}) \cdots S(\theta_0)]$ *do vetor recebido* $\vec{v}$*, em que* θ_i *são as funções de base do espaço ortogonal ao código. Este vetor é obtido pelo produto de* $\vec{v}$ *pela matriz de paridade do código.* $S(\theta_i) = 0$ *se* i *é uma lacuna.*

Saídas:

- *As funções* f *e* ϕ *localizadoras e avaliadoras de erros, respectivamente.*

Inicialização:

- $it = -1$*;*
- $\sigma_{-1} = \{0\}$*;*
- $f(0) = 1$*;*
- $\phi(0) = 0$.

<<< **Passo 1: Cálculo das síndromes** $\alpha = S(f)$ >>>

- *Incrementar* $it = it + 1$*. Se* $it > it_{max}$*, encerrar o algoritmo;*
- *Salvar os valores obtidos na última iteração* $\dot{f} = f$, $\dot{\phi} = \phi$, $\dot{g} = g$, $\dot{\psi} = \psi$, $\dot{\beta} = \beta$, $\dot{\delta} = \delta$, $\dot{\sigma} = \sigma$*;*
- *Para cada* $s \in \dot{\sigma}$*, calcular* $\widetilde{\alpha(s)} = S(\widetilde{\dot{f}\theta_{it-s}})$*. Se* $S(\theta_{it})$ *(necessário a cada iteração) não está disponível, calcular* $S(\widetilde{\dot{f}\theta_{it-s}} - \theta_{it})$ *e empregar o algoritmo de decisão por maioria para determinar* $S(\theta_{it})$*. Em seguida, calcular* $\alpha(s) = S(\widetilde{\dot{f}\theta_{it-s}} - \theta_{it}) + S(\theta_{it})$.
- *O algoritmo de decisão por maioria consiste em se calcular o conjunto* $\Gamma = \dot{\Sigma} \cap \left(it - \dot{\Sigma}\right)$ *e, para cada* $a \in \Gamma$*, encontrar um* $s \in \dot{\sigma}$ *tal que* $s \preceq a$*. Em seguida, escolher* $\zeta \in \mathbb{F}_{r^2}$ *tal que* $g = \theta_{it} + \zeta \dot{f}(s)\theta_{it-s}$ *apresenta grau menor que* it*. O valor de* $S(\theta_{it})$ *consiste na maioria dos valores* $S(g)$ *calculados por cada* s *escolhido.*

<<< **Passo 2 : Determinação dos parâmetros de controle** >>>

- *Calcular* $\delta' = \{it - s : s \in \dot{\sigma},\ it - s \in \Lambda \text{ et } \alpha(s) \neq 0\}$*;*
- *Calcular* $\delta = \max\{\delta' \cup \dot{\delta}\}$*;*
- *Calcular* $\sigma = \min\{t : \nexists\, c \in \delta \text{ with } t \prec c\}$.

<<< **Passo 3 : Atualização das funções** >>>

- *Para cada* $c \in \delta$*, fazer*
  - $g(c) = \begin{cases} \dot{g}(c), & si\ c \in \dot{\delta}, \\ \dot{f}(it - c), & autrement; \end{cases}$
  - $\psi(c) = \begin{cases} \dot{\psi}(c), & si\ c \in \dot{\delta}, \\ \dot{\phi}(it - c), & autrement; \end{cases}$
  - $\beta(c) = \begin{cases} \dot{\beta}(c), & si\ c \in \dot{\delta}, \\ \alpha(it - c), & autrement; \end{cases}$

- *Para cada $t \in \sigma$, encontrar $s \in \dot{\sigma}$ e $a \in \Lambda$ tais que $s + a = t$. Denota-se por $\overline{\phi}$ a função constituída pelos termos de ϕ com valorização discreta máxima $it - t - 2g + 1$.*
  - *Se $it - t$ é uma lacuna, calcular*
    - $f(t) = \dot{f}(s)\theta_a$;
    - $\phi(t) = \overline{\dot{\phi}(s)\theta_a} + \alpha(s)\theta_{2g-1-(it-t)}$;
  - *Se $it-t$ é uma anti-lacuna e $\alpha(s) \neq 0$, encontrar $c \in \dot{\delta}$ e $b \in \Lambda$ tais que $c-s = it-t$. Se $it-t$ é uma anti-lacuna e $\alpha(s) = 0$, então $t = s$ e pode-se escolher b e c quaisquer. Considerando $\gamma = \alpha(s)/\dot{\beta}(c)$, calcular*
    - $f(t) = \dot{f}(s)\theta_a - \gamma\dot{g}(c)\theta_b$;
    - $\phi(t) = \overline{\dot{\phi}(s)\theta_a} - \gamma\overline{\dot{\psi}(c)\theta_b}$;

5.1.1 Exemplo

Considere como exemplo um código de Hermite $(64, 54)$ corretor de 2 erros, definido por uma curva de Hermite $y^4 + y = x^5$ de gênero $g = 6$ sobre $\mathbb{F}_{16}$, portanto de parâmetros $r = 4$ e $m = 59$. Esta curva apresenta 64 pontos racionais mais um ponto Q no infinito do plano projetivo, razão pela qual o código apresenta um comprimento $n = 64$. Este código $(64, 54)$ é definido por um espaço de funções $L(59Q)$. O espaço ortogonal a este código, utilizado na decodificação, é o espaço $L((r^3 + r^2 - r - 2 - m)Q) = L(15Q)$. Este código apresenta um conjunto de lacunas $\{1, 2, 3, 6, 7, 11\}$ de 6 elementos. Ele apresenta uma matriz de paridade de 10 linhas e, portanto, possui 10 síndromes a calcular a partir do vetor recebido.

Considere um vetor recebido $\vec{v}$ ao qual foram adicionados erros de valor:

- α^5 na posição associada ao ponto (α^8, α^3);
- e α^{13} na posição associada ao ponto (α^7, α^{13}).

Desta forma, é obtido o vetor de síndromes

$$\vec{S} = [\alpha\ 1\ \alpha^{13}\ \alpha^9\ 0\ \alpha^2\ \alpha^9\ \alpha^4\ 0\ 0\ \alpha^7\ \alpha^7\ 0\ 0\ 0\ \alpha^7]$$

que apresenta 10 síndromes calculadas a partir de $\vec{v}$ nas posições das anti-lacunas e zeros nas posições das 6 lacunas.

A tabela 5.1 descreve os valores resultados da decodificação a cada iteração do algoritmo 47.

Tabela 5.1: Resultados da decodificação do exemplo descrito na seção 5.1.1. São mostrados os resultados apenas das iterações em que ocorreram mudanças.

it	σ e δ	β	f e g	ϕ e ψ
-1	$\sigma : 0$		1	0
	$\delta : \emptyset$			
0	$\sigma : 4$		x	$\alpha^7 y^3$
	$\sigma : 5$		y	$\alpha^7 x^4$
	$\delta : 0$	α^7	1	0
4	$\sigma : 4$		$x+1$	$\alpha^7 y^3$
	$\sigma : 5$		y	$\alpha^7 x^4 + \alpha^7 x^3$
	$\delta : 0$	α^7	1	0
5	$\sigma : 4$		$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
	$\sigma : 5$		$y+1$	$\alpha^7 x^4 + \alpha^7 x^3$
	$\delta : 0$	α^7	1	0
8	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2$
	$\sigma : 5$		$y+1$	$\alpha^7 x^4 + \alpha^7 x^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
9	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
10	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
11	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^7 xy^2 + y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
12	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
14	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$
	$\delta : 4$	α^3	$x+1$	$\alpha^7 y^3 + \alpha^7 y^2$
15	$\sigma : 8$		$x^2 + x + \alpha^{11}$	$\alpha^7 xy^3 + \alpha^4 y^3 + \alpha^7 xy^2 + \alpha y^2 \alpha^2 xy + \alpha^6 y + \alpha^{14} x$
	$\sigma : 5$		$y + \alpha^{12} + \alpha^{11}$	$\alpha^7 x^4 + \alpha^4 y^3 + \alpha^7 x^3 + \alpha^4 y^3 + \alpha^4 x^2 + \alpha^{14} y + \alpha^9 x$

Tabela 5.1: Resultados da decodificação do exemplo descrito na seção 5.1.1. São mostrados os resultados apenas das iterações em que ocorreram mudanças.

it	σ e δ	β	f e g	ϕ e ψ
	$\delta : 4$	α^3	$x+1$	$\alpha^7y^3+\alpha^7y^2$
16	$\sigma : 8$		$x^2+x+\alpha^{11}$	$\alpha^7xy^3+\alpha^4y^3+\alpha^7xy^2+\alpha y^2\alpha^2xy+\alpha^6y+\alpha^{14}x$
	$\sigma : 5$		$y+\alpha^{12}+\alpha^{11}$	$\alpha^7x^4+\alpha^4y^3+\alpha^7x^3+\alpha^4y^3+\alpha^4x^2+\alpha^{14}y+\alpha^9x+\alpha^7$
	$\delta : 4$	α^3	$x+1$	$\alpha^7y^3+\alpha^7y^2$
19	$\sigma : 8$		$x^2+x+\alpha^{11}$	$\alpha^7xy^3+\alpha^4y^3+\alpha^7xy^2+\alpha y^2\alpha^2xy+\alpha^6y+\alpha^{14}x+\alpha^{11}$
	$\sigma : 5$		$y+\alpha^{12}+\alpha^{11}$	$\alpha^7x^4+\alpha^4y^3+\alpha^7x^3+\alpha^4y^3+\alpha^4x^2+\alpha^{14}y+\alpha^9x+\alpha^7$
	$\delta : 4$	α^3	$x+1$	$\alpha^7y^3+\alpha^7y^2$

Ao fim da iteração $it = 21$, o algoritmo produz dois pares de funções f e ϕ:

- $f_1 = x^2 + x + \alpha^{11}$;
- $\phi_1 = \alpha^7xy^3 + \alpha^4y^3 + \alpha^7xy^2 + \alpha y^2\alpha^2xy + \alpha^6y + \alpha^{14}x + \alpha^{11}$;
- $f_2 = y + \alpha^{12} + \alpha^{11}$;
- $\phi_2 = \alpha^7x^4 + \alpha^4y^3 + \alpha^7x^3 + \alpha^4y^3 + \alpha^4x^2 + \alpha^{14}y + \alpha^9x + \alpha^7$.

A função f_1 apresenta zeros nas posições associadas aos pontos (α^8, α^3), (α^8, α^{14}), (α^7, α^9), (α^7, α^7), (α^7, α^6), (α^7, α^{13}), (α^8, α^{11}) e (α^8, α^{12}). Aplicando (5.3), obtém-se um valor de erro α^5 para o ponto (α^8, α^3), α^{13} para o ponto (α^7, α^{13}) e 0 para os demais pontos.

Já a função f_2 apresenta zeros nas posições associadas aos pontos (α^9, α), (α^{12}, α^2), (α^8, α^3), (α^4, α^6) e (α^7, α^{13}). Aplicando (5.3), obtém-se novamente um valor de erro α^5 para o ponto (α^8, α^3), α^{13} para o ponto (α^7, α^{13}) e 0 para os demais pontos.

5.2 Arquitetura

Através de uma análise simples, pode-se concluir que o decodificador de O'Sullivan apresenta uma complexidade mais importante concentrada na parte de controle do algoritmo (passo 2 e determinação das anti-lacunas a e b no passo 3 d algoritmo 47). Esta característica difere do que se observa em outros algoritmos de decodificação para códigos AG. Em geral, estes algoritmos apresentam uma complexidade concentrada sobretudo no

cálculo das síndromes e na atualização das funções. Foi esta característica deste algoritmo nos motivou e conduziu em direção à sua exploração.

Em [49], O'Sullivan não apresenta uma discussão sobre a complexidade de seu algoritmo, como é comum em artigos deste gênero. Ele não fornece sequer certas informações necessárias à determinação desta complexidade, como o comprimento dos registradores das funções. Nós deduzimos todos os parâmetros que faltavam para a determinação da complexidade e para a implementação deste decodificador.

Nesta seção, uma nova arquitetura de implementação em hardware é apresentada para este algoritmo. A seção seguinte apresenta alguns resultados obtidos após a implementação do caso particular de um código de Hermite (64, 54) sobre $\mathbb{F}_{16}$.

5.2.1 Unidades aritméticas

No decodificador de O'Sullivan, todas as operações aritméticas sobre valores de síndromes e coeficientes de funções em R, símbolos do corpo finito $\mathbb{F}_q$, são realizadas utilizando unidades aritméticas desenvolvidas para este corpo. Estas unidades aritméticas e suas arquiteturas estão descritas em detalhes no capítulo 4. As figuras 4.2, 4.6 e 4.9 apresentam as arquiteturas dos operadores de adição, multiplicação e inversão que foram implementadas para o corpo finito $\mathbb{F}_{16}$.

Na arquitetura a ser apresentada para o decodificador do algoritmo 47, um símbolo de $\mathbb{F}_{r^2}$ é representado por $\log_2 r^2$ bits. Uma função de R consiste num vetor de pl símbolos, em que pl é o número máximo de iterações do algoritmo necessárias à obtenção das funções desejadas mais 1 (observa-se que numa iteração it a ordem de pólo de qualquer função no algoritmo jamais é maior que it), ou seja,

$$pl = it_{max} + 1.$$

Todos os números inteiros no circuito são representados por il bits.

5.2.2 Cálculos mais freqüentes

Dentre os cálculos efetuados no algoritmo, pode-se destacar alguns mais utilizados e com maior impacto sobre o desempenho da implementação e para os quais são propostas arquiteturas particulares:

Operador para o produto $f.\theta_c$

Considere f uma função e θ_c um monômio em R. A operação $f.\theta_c$ é sistematicamente efetuada nos cálculos de síndrome e nas atualizações das funções (etapas 1 e 3 do algoritmo). Este produto de funções em R é feito módulo a equação da curva de Hermite e,

portanto, implica a substituição $x^{r+1} = y^r + y$. Portanto, este produto de funções não se traduz numa mera convolução de vetores, no caso um simples deslocamento de um vetor, mas exige elementos adicionais que implementem a substituição descrita.

Esta substituição se opera da seguinte forma. Considere $\theta_c = x^{\mu(c)}y^{\nu(c)}$. Multiplicando-se um monômio qualquer $f_k x^{\mu(k)}y^{\nu(k)}$ por θ_c, obtém-se $f_k x^{\mu(k)+\mu(c)}y^{\nu(k)+\nu(c)}$ de ordem de pólo $k+c$ em Q. No entanto, se $\mu(k)+\mu(c) > r$, tem-se que

$$x^{\mu(k)+\mu(c)}y^{\nu(k)+\nu(c)}\frac{(y^r+y)}{x^{r+1}} = x^{\mu(k)+\mu(c)-r-1}(y^{\nu(k)+\nu(c)+1} + y^{\nu(k)+\nu(c)+r}).$$

Observe que $x^{\mu(k)+\mu(c)-r-1}y^{\nu(k)+\nu(c)+r}$ apresenta uma ordem de pólo em Q igual a

$$(\mu(k)+\mu(c)-r-1)r + (\nu(k)+\nu(c)+r)(r+1) = k+c$$

e $x^{\mu(k)+\mu(c)-r-1}y^{\nu(k)+\nu(c)+1}$ apresenta uma ordem de pólo em Q igual a

$$(\mu(k)+\mu(c)-r-1)r + (\nu(k)+\nu(c)+1)(r+1) = k+c-r^2+1.$$

Portanto, o produto de uma função f por um monômio θ_c em R corresponde na verdade ao deslocamento de f em c posições, sendo que às posições de ordem k da função em que $\mu(k)+\mu(c) > r$, adiciona-se o coeficiente da posição de ordem $k+c-r^2+1$.

O'Sullivan em [49] observou este fato e propôs a implementação deste produto pelo uso de dois registradores, em que o primeiro armazena os coeficientes de f e o segundo armazena os coeficientes de f em que $\mu(k)+\mu(c) > r$. O primeiro registrador é deslocado em c posições (a direção do deslocamento depende do sinal de c) e o segundo registrador é deslocado em $c-r^2+1$ posições. Em seguida, os dois registradores são somados para gerar a função resultado do produto $f.\theta_c$.

Uma arquitetura menos complexa e mais eficiente para este operador é apresentada na figura 5.2. Ela consiste num único registrador de deslocamento modificado para incluir elementos de decisão e operadores de adição entre células de coeficientes da função. Este módulo recebe uma função f e valores associados a um inteiro c (sinais de controle determinados a partir de c no módulo controlador) e fornece uma função, resultado do produto $f.\theta_c$.

Toda a operação deste módulo é determinada pelo sinal de controle D, gerado pelo módulo controlador do decodificador. Um sinal D é constituído por 3 bits, que determinam os 5 diferentes estados de operação em um módulo $f.\theta_c$, a saber ($\star$ indica indiferença quanto ao bit ser '0' ou '1'):

Estado $\star$00 Indica que o módulo deve carregar-se com a função de entrada f';

Estado $\star$01 Indica que o módulo deve efetuar a adição dos componentes $f_k + f_{k-r^2+1}$, o que corresponde à operação módulo equação da curva de Hermite;

Estado 010 Indica a operação de deslocamento do registrador à esquerda;

Estado 110 Indica a operação de deslocamento do registrador à direita;

Estado $\star$11 Indica o estado de suspensão, no qual os valores do registrador são preservados.

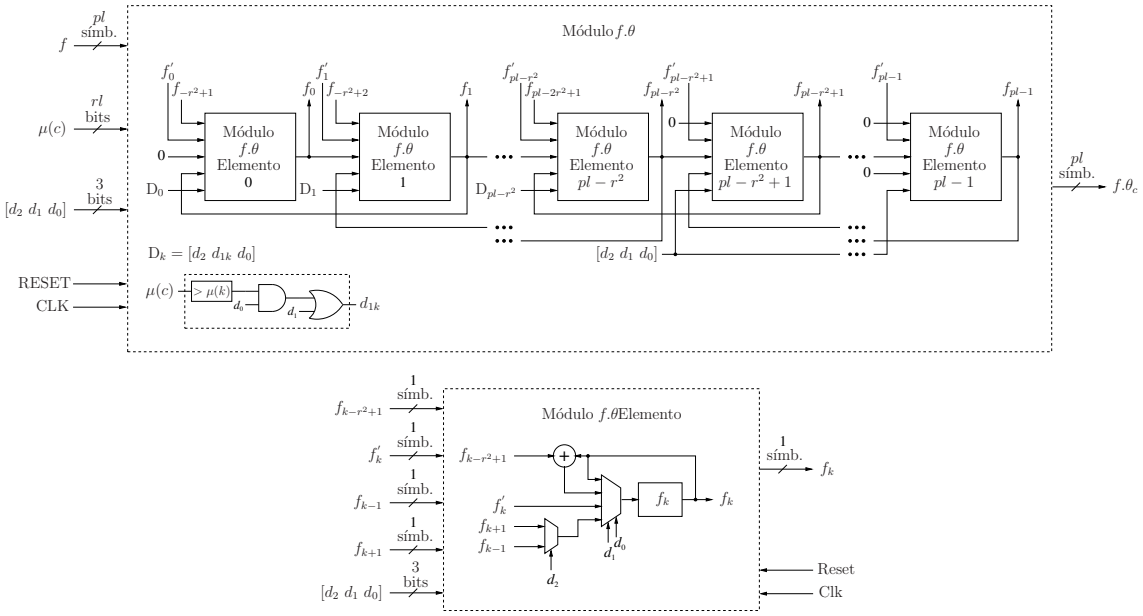

Figura 5.2: Arquitetura para a implementação da operação $f.\theta_c$. Num sub-módulo k do operador, considera-se como entrada da célula do registrador: $f_k^{'}$ se $d_{k1} = d_{k2} = 0$ (carregando a função de entrada $f^{'}$), $f_k + f_{k-r^2+1}$ se $d_{k1} = \overline{d_{k2}}$ e $\mu(c) \geq r - \mu(k)$, f_k se $d_{k1} = \overline{d_{k2}}$ e $\mu(c) < r - \mu(k)$, e f_{k+1} ou f_{k-1} se $d_{k1} = d_{k2} = 1$ e segundo o bit sinal de c (operação de deslocamento).

Operador para a função $\mu(c) = i$

Considere uma função $\mu(c) = i$, tal que $ir + j(r+1) = c$. Esta função é utilizada como sub-módulo do operador $f.\theta_c$ para determinar se o coeficiente f_k do k-ésimo monômio do registrador modificado de $f.\theta_c$ deve ser adicionado ao coeficiente f_{k-r^2+1} (*cf.* figura 5.2). Isto equivale à realização do módulo equação da curva de Hermite.

Em (2.17), observa-se que esta função $\mu(c)$ pode ser realizada por um operador de módulo $(r+1)$. A operação módulo $(r+1)$ pode ser implementada por uma cadeia de até

$$il^2 - il - \lceil \log(r+1) \rceil^2 + \lceil \log(r+1) \rceil$$

somadores completos (na análise de cada caso particular, esta complexidade pode ser bastante reduzida).

Um exemplo de uma arquitetura deste tipo para uma operação módulo 5 sobre um inteiro de 7 bits, utilizada na implementação do decodificador para o código $(64, 54)$, é apresentada na figura 5.3.

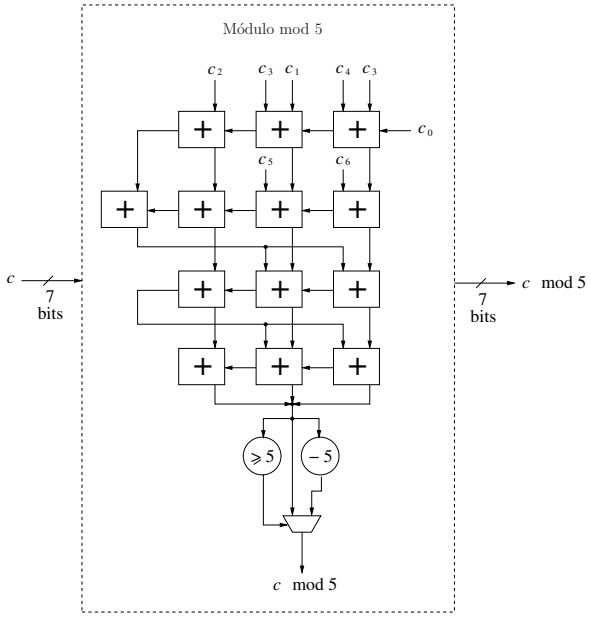

Figura 5.3: Arquitetura para um operador módulo 5 para um inteiro c de 7 bits de entrada (bit de sinal incluído).

Operador para o teste de pertença a Λ

Quando de quase todas as decisões no algoritmo de decodificação, evidencia-se a necessidade de realizar um teste do tipo $c \in \Lambda$, para um dado inteiro c.

Este tipo de teste pode ser realizado por uma busca em tabela verificando-se sua negativa, ou seja, se c pertence ao conjunto das anti-lacunas, que em geral é menor, ou se é negativo. Além disso, este teste pode ser realizado pelo uso da função $\mu(c)$. Observe que $c \in \Lambda$ se e só se $\mu(c)r \leq c$.

Sobre o espaço de funções relativamente pequeno do código $(64, 54)$ (existem apenas seis lacunas: 1, 2, 3, 6, 7 e 11), optou-se por empregar o método de busca em tabela.

5.2.3 Módulos operacionais

A figura 5.4 apresenta a arquitetura geral para o decodificador que emprega o algoritmo de O'Sullivan. Ela inclui r blocos MP (processadores principais), r blocos AP (processadores auxiliares), um buffer para as síndromes de entrada e um módulo controlador que fornece os sinais de controle para as seqüências de operação. Os blocos MP e AP são interconectados por intermédio de um barramento bidirecional.

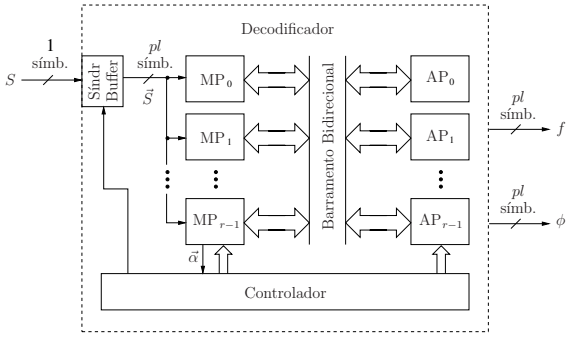

Figura 5.4: Arquitetura geral proposta para o decodificador de O'Sullivan [49].

Dependendo do código sobre o qual se trabalha (comprimento e capacidade de correção), o número de blocos MP e AP necessários pode ser menor que r. Não há uma regra geral para determinar o número de blocos MP e AP necessários, caso possa ser menor que r, mas se o código é capaz de corrigir $\tau < r$ erros, então τ blocos MP e $\tau - 1$ blocos AP são suficientes. Isto se deve ao fato de que o máximo elemento do conjunto δ não pode ser maior que a τ-ésima anti-lacuna. Por exemplo, no caso do código AG $(64, 54)$ de nosso exemplo, corretor de 2 erros, apenas são necessários 2 módulos MP e 1 módulo AP.

O decodificador recebe um vetor de $n - k$ síndromes $\vec{S}$ e fornece um conjunto de r funções localizadoras f e r funções avaliadoras de erros ϕ.

Uma arquitetura para a unidade MP é apresentada na figura 5.5. Cada módulo MP opera sobre uma função localizadora de erros f e uma função avaliadora de erros ϕ. Ele implementa o cálculo de uma síndrome α a partir de uma função f e todas as operações de atualização das funções f e ϕ.

Este módulo emprega dois operadores $f.\theta_c$:

- um para o cálculo de α e para a atualização de f,
- e outro para a atualização de ϕ.

Ele emprega ainda:

- um registrador para memorizar α;
- dois registradores para memorizar f e ϕ;
- e dois registradores para memorizar as funções $g\theta_b$ e $\psi\theta_b$ recebidas dos módulos AP (os produtos $g\theta_b$ e $\psi\theta_b$ são realizados pelos módulos AP).

Nos módulos MP, um módulo decodificador chamado DEC1 gera um vetor de bits na forma $0 \ldots 010 \ldots 0$ a partir de um inteiro tr, no qual o único bit '1' ocupa a tr-ésima posição do vetor de bits. Este vetor é utilizado para implementar a operação de

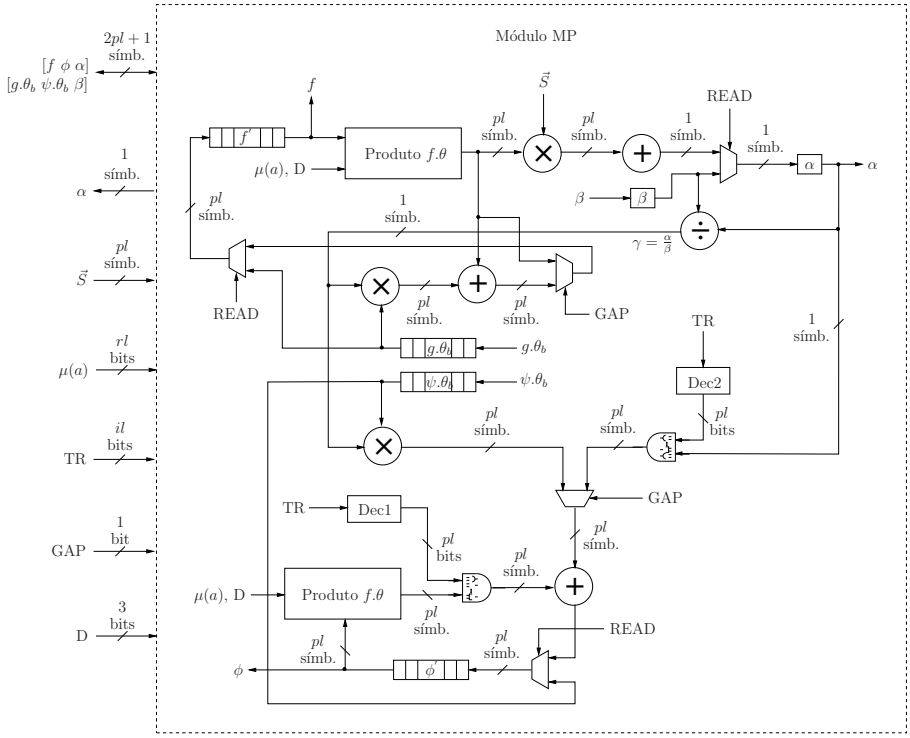

Figura 5.5: Arquitetura para blocos MP do decodificador, responsáveis pelo cálculo das síndromes α e pela atualização das funções localizadoras e avaliadoras de erros f e ϕ.

adição de uma função por um monômio θ_c na atualização de funções ϕ. Um módulo decodificador chamado DEC2 opera de modo análogo, fornecendo um vetor de bits na forma $1 \ldots 10 \ldots 0$, com bits '1' até a tr-ésima posição do vetor. Este vetor é utilizado para truncar uma função.

Os detalhes da arquitetura para unidades AP são apresentados na figura 5.6. Este módulo emprega:

- dois operadores $f.\theta_c$ para calcular os produtos $g.\theta_b$ e $psi.\theta_b$ utilizados na atualização das funções f e ϕ de alguma unidade MP;
- registradores para memorizar uma síndrome β (ela guarda uma síndrome α relativa à função f associada a g), assim como $\dot{\beta}$, $\dot{g}$ e $\dot{\psi}$ (valores produzidos na última iteração).

5.2.4 Módulo controlador

O circuito controlador do decodificador gera todos os sinais de controle para os módulos MP e AP, assim como para o buffer das síndromes de entrada, e para a saída do decodificador. Ele é também responsável por gerar internamente os conjuntos σ e δ e, por conseguinte, tomar todas as decisões relativas ao funcionamento do decodificador. É ele

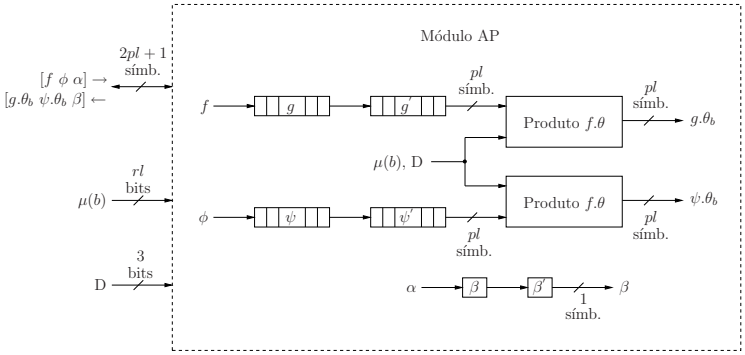

Figura 5.6: Arquitetura para blocos AP do decodificador, responsáveis pela atualização das funções auxiliares e pelo cálculo dos produtos $g.\theta_b$ e $\psi.\theta_b$.

que determina, por exemplo, se uma função deve ser atualizada, por que tipo de operação e utilizando quais outras funções.

O controlador implementa ainda o procedimento de decisão por maioria. Numa iteração na qual $S(\theta_{it})$ é desconhecida, os módulos MP calculam os valores estimados de α, uma vez que $S(\theta_{it}) = 0$ no registrador de síndromes localizado no módulo buffer das síndromes de entrada. O módulo de decisão por maioria interno ao controlador calcula o conjunto Γ e determina os valores de σ associados. Em seguida, ele escolhe dentre os valores de α estimados qual deles corresponde a $S(\theta_{it})$. Este valor é, então, copiado no registrador de síndromes e adicionado aos valores de α em todos os módulos MP.

Dentre os sinais gerados pelo módulo controlador, destacam-se os sinais D, que controlam todo o funcionamento dos módulos $f.\theta_c$ (*cf.* figura 5.2). Uma descrição destes sinais D é apresentada na seção 5.2.2.

Dentre as decisões tomadas pelo controlador, destaca-se a determinação das antilacunas a e b utilizadas no cálculo dos produtos $f.\theta_a$, $\phi.\theta_a$, $g.\theta_b$ e $\psi.\theta_b$, assim como para o endereçamento dos valores armazenados nos módulos MP para os módulos AP, e vice-versa. Dois sub-módulos do controlador são responsáveis por estes cálculos e estas tomadas de decisão: os módulos NONGAPMP e NONGAPAP (*cf.* figuras 5.7 e 5.8).

5.2.5 Complexidade

Como já foi afirmado, em [49] O'Sullivan não trata da questão da complexidade de seu algoritmo. Ele afirma que a relação entre a capacidade de correção e o número de iterações necessárias para decodificar uma palavra recebida não é trivial. Ele também não determina o número de funções f e ϕ calculadas pelo algoritmo.

De modo a possibilitar a implementação deste decodificador, fez-se necessário determinar invariavelmente estes parâmetros. No que diz respeito ao número de iterações

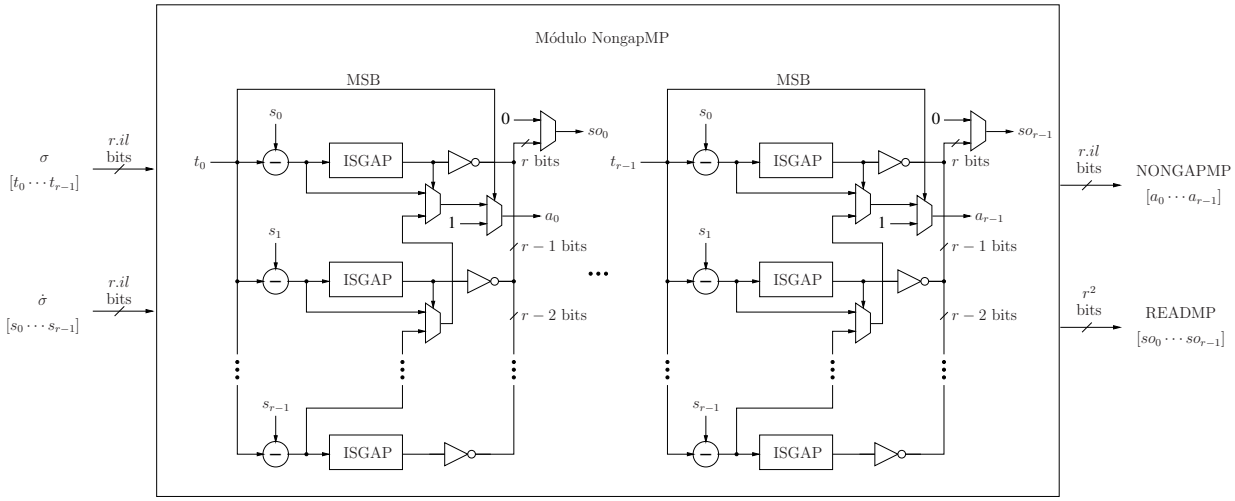

Figura 5.7: Arquitetura para um módulo NONGAPMP responsável pela determinação das anti-lacunas a utilizadas na atualização de funções, assim como pelo endereçamento dos valores armazenados nos módulos MP para os módulos AP específicos.

necessárias para corrigir até metade da distância mínima do código, considerou-se aqui o trabalho apresentado por Feng e Rao em [21]. Considerando que $m = n-k$ é o número de linhas da matriz de paridade do código (número de síndromes conhecidas a partir do vetor recebido) e g é o gênero da curva, Feng e Rao afirmam que em $it_{max} = m + g$ iterações a decodificação utilizando o esquema de decisão por maioria por eles proposto corrige até metade da distância mínima do código.

Pode-se verificar que numa dada iteração it uma operação $f.\theta_c$ fornece uma função que apresenta valorização discreta em Q máxima $-it$. Conseqüentemente, a valorização discreta em Q máxima de uma função qualquer do decodificador numa iteração it é sempre $-it$. Portanto, necessita-se implementar módulos $f.\theta_c$ e registradores para funções de comprimento $pl = it_{max} + 1$.

Em termos de uso de recursos materiais, um módulo $f.\theta_c$ emprega:

- pl operadores de adição em $\mathbb{F}_q$;
- $pl.r$ flip-flops;
- $pl.r$ demultiplexadores 2×1;
- $pl.r$ demultiplexadores 4×1;
- $pl - r^2 + 1$ comparadores de inteiros de rl bits (número de bits necessários para representar o valor de r);
- $pl - r^2 + 1$ portas AND;

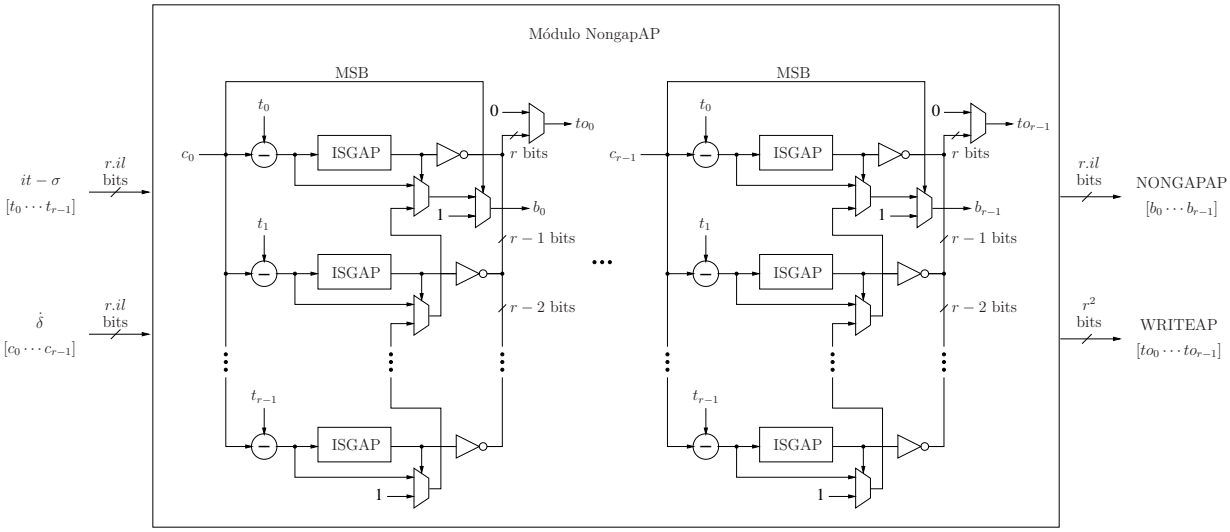

Figura 5.8: Arquitetura para um módulo NONGAPAP responsável pela determinação das anti-lacunas b utilizadas na atualização de funções, assim como pelo endereçamento dos valores armazenados nos módulos AP para os módulos MP específicos.

- $pl - r^2 + 1$ portas OR.

Um módulo AP emprega:

- 2 módulos $f.\theta_c$;
- $(4pl + 2).r$ flip-flops.

Já um módulo MP, responsável pela quase totalidade dos cálculos da decodificação, emprega:

- 2 módulos $f.\theta_c$;
- $(4pl + 2).r$ flip-flops;
- $(4pl + 1).r$ demultiplexadores 2×1;
- $3pl$ operadores de multiplicação em $\mathbb{F}_q$;
- $3pl - 1$ operadores de adição em $\mathbb{F}_q$;
- $(pl + 1).r$ portas AND;
- 1 decodificador DEC1;
- 1 decodificador DEC2.

5.3 Implementação

A arquitetura apresentada foi descrita em VHDL de forma genérica para permitir a síntese sobre quaisquer parâmetros r e m de um código AG (n, k) gerado por um espaço de funções $L(mQ)$.

Um decodificador para o caso do exemplo de um código AG (64, 54) em $\mathbb{F}_{16}$ foi sintetizado para um FPGA EP1C20F324C6 da família Cyclone da Altera utilizando o software Quartus II da própria Altera. Foram utilizados 15.225 elementos lógicos, o que representa 76% dos recursos deste dispositivo. Neste caso, o decodificador pode operar em 50 MHz, desde que 4 ciclos de relógio sejam reservados para o processamento do módulo MAJVOTING, responsável pela decisão por maioria, e 4 ciclos sejam reservados para o módulo UPDDELTASIGMA, responsável pelo cálculo dos novos conjuntos δ e σ. No total, são gastos 1459 ciclos de relógio para decodificar um vetor recebido, compreendidos em 22 iterações de 66 ciclos cada mais 7 ciclos de relógio para a inicialização e a finalização do processo de decodificação. Numa freqüência de 50 MHz, isto implica uma taxa máxima de decodificação de 34270 palavras código por segundo, ou uma taxa de 8,8 Mb/s (bits de código), ou 7,4 Mb/s (bits de informação).

A descrição VHDL feita permite regular o número de ciclos de relógio reservados para as operações dos módulos MAJVOTING e UPDDELTASIGMA, assim como para o cálculo de α, para a atualização das funções e para todas as demais operações do decodificador. Esta regulagem é feita através das constantes cycleestalph, cyclealpha, cyclempap, cyclefcalc, cycleapmp, cycleiterat e period, especificadas no arquivo 'HermConstants.vhd'. Os resultados apresentados no último parágrafo são devidos à tecnologia utilizada: um FPGA Altera EP1C20F324C6 da família Cyclone. No caso de uma tecnologia diferente, em cuja síntese outros tempos de propagação e outros caminhos críticos sejam obtidos, nova regulagem das constantes descritas pode ser necessária.

5.4 Conclusões

O presente capítulo descreveu e exemplificou a operação do algoritmo de decodificação proposto por O'Sullivan em [49]. Parâmetros necessários à implementação do decodificador, que não haviam sido determinados por O'Sullivan, como a ordem de pólo máxima das funções e o número de funções calculadas pelo algoritmo, foram determinados. Em seguida, foi apresentada uma nova arquitetura para este decodificador com vistas à decodificação de códigos de Hermite.

A arquitetura proposta consiste em r unidades MP paralelas e r unidades AP paralelas responsáveis pela atualização iterativa de funções localizadoras e avaliadoras de erros. Es-

tas unidades são interligadas por um barramento bidirecional. Uma unidade MP emprega 2 módulos $f.\theta_c$, $(4pl+2).r$ flip-flops, $(4pl+1).r$ demultiplexadores 2×1, $3pl$ operadores de multiplicação em $\mathbb{F}_q$, $3pl-1$ operadores de adição em $\mathbb{F}_q$, $(pl+1).r$ portas AND e 2 sub-módulos decodificadores de pl bits. Uma unidade AP emprega $(4pl+2).r$ flip-flops e 2 módulos $f.\theta$. Cada módulo $f.\theta$ emprega pl operadores de adição em $\mathbb{F}_q$, $pl.r$ flip-flops, $pl.r$ demultiplexadores 2×1, $pl.r$ demultiplexadores 4×1, $pl - r^2 + 1$ comparadores de inteiros de rl bits (número de bits necessários para representar o valor de r), $pl - r^2 + 1$ portas AND e $pl - r^2 + 1$ portas OR. A arquitetura apresenta ainda um buffer para as síndromes de entrada e um circuito controlador responsável por gerar todos os sinais de controle para o circuito.

Foram ainda propostos operadores otimizados para implementar os cálculos mais freqüentes, tais como os produtos $f.\theta_c$.

A complexidade de um módulo MP assemelha-se à de uma arquitetura de Berlekamp-Massey para decodificação de códigos RS (ou a um dos r módulos do decodificador proposto por Kötter em [34]). O que se pode concluir a partir desta arquitetura descrita sobre a complexidade de decodificadores para códigos de Hermite em comparação com códigos RS é que os decodificadores para códigos de Hermite apresentam complexidade maior que decodificadores para códigos RS numa ordem de r vezes. No entanto, códigos RS necessitam utilizar corpos finitos muito maiores, portanto uma aritmética muito mais complexa, para corrigir vetores de mesmo comprimento. Desta forma, estima-se que a complexidade geral para comprimentos de código iguais (em número de bits) seja equivalente para ambos os códigos de Hermite e RS. Um trabalho de análise e comparação mais preciso neste sentido constitui uma perspectiva de continuação do presente trabalho.

O capítulo seguinte apresenta as conclusões para o presente documento de tese.

Capítulo 6

Conclusões

A presente tese descreveu o trabalho de doutorado realizado no desenvolvimento de uma arquitetura para um decodificador de códigos AG baseados em curvas de Hermite. Este trabalho compreendeu duas competências distintas que se complementaram aqui: o estudo dos algoritmos de decodificação para códigos AG e o desenvolvimento de arquiteturas de implementação em hardware para estes decodificadores.

Para viabilizar a implementação em hardware de sistemas codificadores / decodificadores para códigos AG, assim como para qualquer código de bloco, fez-se necessário o estudo e implementação de unidades aritméticas para corpos finitos de característica 2. Operadores de adição, multiplicação e inversão foram implementados. O capítulo 4 tratou exclusivamente deste assunto.

O capítulo 5 descreveu o algoritmo de decodificação de códigos AG, objeto central deste trabalho, que busca iterativamente funções localizadoras e avaliadoras de erros que satisfaçam um critério de equação chave. Foram determinados parâmetros necessários à implementação deste decodificador que não haviam sido antes determinados. Em seguida, foi apresentada uma nova arquitetura para este decodificador com vistas à decodificação hardware de códigos de Hermite.

A arquitetura proposta consiste em r unidades MP paralelas e r unidades AP paralelas responsáveis pela atualização iterativa de funções localizadoras e avaliadoras de erros. Estas unidades são interligadas por um barramento bidirecional. Uma unidade MP emprega 2 módulos $f.\theta_c$, $(4pl+2).r$ flip-flops, $(4pl+1).r$ demultiplexadores 2×1, $3pl$ operadores de multiplicação em $\mathbb{F}_q$, $3pl-1$ operadores de adição em $\mathbb{F}_q$, $(pl+1).r$ portas AND e 2 sub-módulos decodificadores de pl bits. Uma unidade AP emprega $(4pl+2).r$ flip-flops e 2 módulos $f.\theta$. Cada módulo $f.\theta$ emprega pl operadores de adição em $\mathbb{F}_q$, $pl.r$ flip-flops, $pl.r$ demultiplexadores 2×1, $pl.r$ demultiplexadores 4×1, $pl - r^2 + 1$ comparadores de inteiros de rl bits (número de bits necessários para representar o valor de r), $pl - r^2 + 1$ portas AND e $pl - r^2 + 1$ portas OR. A arquitetura apresenta ainda um buffer para as

síndromes de entrada e um circuito controlador responsável por gerar todos os sinais de controle para o circuito.

Foram ainda propostos operadores otimizados para implementar os cálculos mais freqüentes, tais como os produtos $f.\theta_c$.

Este trabalho constitui uma contribuição na área do desenvolvimento dos sistemas de codificação de canal usando códigos AG. Apesar destes códigos apresentarem melhores parâmetros que códigos de uso geral como os RS, eles enfrentam uma barreira à sua utilização imposta pela complexidade de seus algoritmos de decodificação. O que se pode concluir a partir desta arquitetura descrita é que os decodificadores para códigos de Hermite apresentam complexidade maior que decodificadores para códigos RS numa ordem de r vezes. No entanto, códigos RS necessitam utilizar corpos finitos maiores ($\sqrt{q}$ vezes em relação a códigos de Hermite), portanto uma aritmética mais complexa, para corrigir vetores de mesmo comprimento. Desta forma, estima-se que a complexidade geral para comprimentos de código iguais (em número de bits) seja equivalente para ambos os códigos de Hermite e RS. Sugere-se, portanto, a utilização preferencial dos códigos AG em relação ao códigos RS em aplicações que priorizem códigos de comprimento grande e boa capacidade de correção em detrimento do custo de implementação do sistema.

Um trabalho a ser proposto nesta linha consiste numa comparação dos sistemas de codificação de canal usando códigos AG com sistemas usando códigos RS, ou possivelmente códigos turbo no caso de uma abordagem por decisão suave. Esta comparação seria não apenas do ponto de vista de desempenho (ganho de codificação e relação capacidade de correção versus taxa do código), mas uma análise do custo / benefício levando-se em consideração o custo de implementação associado ao uso de cada código.

Outra sugestão de trabalho futuro consiste na busca por uma estruturação do fluxo de dados no decodificador, semelhante ao feito por Kötter em sua versão do algoritmo BMS. Esta modificação algorítmica proporcionaria uma boa simplificação na parte de controle do decodificador.

Apêndice A

Álgebra de Corpos Finitos

Os códigos de bloco, em geral, baseiam-se em estruturas de anéis de polinômios e nos sistemas aritméticos dos corpos finitos. Estes e outros conceitos básicos da álgebra necessários ao entendimento dos esquemas de codificação e decodificação são descritos resumidamente neste apêndice.

Não é objetivo deste apêndice descrever de forma pormenorizada a teoria dos corpos finitos. Para tanto, existem diversas referências importantes sobre este assunto [1, 6, 38, 39, 47].

A.1 Propriedades básicas

A álgebra apresenta três estruturas básicas, chamadas grupo, anel e corpo, a partir das quais toda a teoria é desenvolvida. Estas estruturas consistem em conjuntos de objetos matemáticos (como o dos números reais, dos números inteiros, etc.) associados a regras de relação entre seus elementos.

Definição 48 (Grupo) *Um conjunto G associado a uma operação binária $G + G \rightarrow G$ é denominado um* grupo *se as seguintes condições forem satisfeitas:*

- *A operação binária '+' é associativa, ou seja, $(a + b) + c = a + (b + c)$, para todo $a, b, c \in G$.*
- *Existe um único elemento identidade $e \in G$, tal que $a + e = e + a = a$, para todo $a \in G$.*
- *Existe um único elemento inverso $a^{'} \in G$, tal que $a + a^{'} = a^{'} + a = e$, para todo elemento $a \in G$.*

Se um grupo satisfaz ainda a condição de que $a + b = b + a$, para todo $a, b \in G$, diz-se que o grupo é *comutativo* ou *abeliano*.

Definição 49 (Anel) *Um conjunto R associado a duas operações binárias $R + R \to R$ e $R \times R \to R$ é denominado um* anel *se as seguintes condições forem satisfeitas:*

- *R constitui um grupo abeliano sob a operação binária '+', chamada adição. O elemento identidade com relação à adição é o '0', chamado elemento zero.*
- *A operação binária '$\times$', chamada multiplicação, é associativa, ou seja, $a \times (b \times c) = (a \times b) \times c$, para todo $a, b, c \in R$;*
- *A operação binária '$\times$' obedecem à propriedade da distributividade, ou seja, $a \times (b + c) = a \times b + a \times c$ e $(b + c) \times a = b \times a + c \times a$, para todo $a, b, c \in R$;*

Um anel é *comutativo* quando a multiplicação é comutativa, ou seja, quando $a \times b = b \times a$, para todo $a, b \in R$.

Um anel não necessariamente possui identidade na multiplicação, assim como inversas de seus elementos. Caso um anel tenha identidade na multiplicação, esta identidade é única. Além disso, se $a \times b = 1$ e $c \times a = 1$, então $b = c$ e a é dito ter uma única inversa denotada por a^{-1}.

Um elemento que possua inversa em um anel é dito uma *unidade*.

Segue a definição da estrutura algébrica denominada corpo.

Definição 50 (Corpo) *Um conjunto F associado a duas operações binárias $F + F \to F$ e $F \times F \to F$ é denominado um* corpo *se as seguintes condições forem satisfeitas:*

- *F constitui um grupo abeliano sob a operação binária '+', chamada adição. O elemento identidade com relação à adição é o '0', chamado elemento zero.*
- *O conjunto dos elementos de F excluindo-se o zero forma um grupo sob a operação binária $\times$, chamada multiplicação. O elemento identidade com relação à multiplicação é o '1', chamado elemento unidade.*
- *A operação de multiplicação é distributiva sobre a adição, ou seja, para $\alpha, \beta, \gamma \in F$, tem-se que $\alpha \times (\beta + \gamma) = \alpha \times \beta + \alpha \times \gamma$.*

São exemplos de corpos os conjuntos $\mathbb{R}$ (números reais), $\mathbb{C}$ (números complexos) e $\mathbb{Q}$ (números racionais). Os corpos com número finito de elementos são chamados *corpos finitos* ou *corpos de Galois*. Um corpo finito com q elementos é denotado por $\mathbb{F}_q$.

Um subconjunto de um corpo F é dito *subcorpo* de F se constituir um corpo sob as operações inerentes a F. O corpo F é dito uma *extensão* deste subcorpo.

Um corpo comporta-se como um anel, que permite a divisão ou cancelamento. Em um corpo, se $a \times b = a \times c$ e $a \neq 0$, então $a^{-1} \times a \times b = a^{-1} \times a \times c \Rightarrow b = c$. Existem alguns

anéis, como o anel dos inteiros, que permitem o cancelamento, mesmo não sendo corpos, ou seja, mesmo não existindo a inversa na multiplicação para todos os seus elementos. Um anel comutativo em que $b = c$ sempre que $a \times b = a \times c$, com $a \neq 0$, é chamado de *domínio integral*.

O número de elementos de um corpo finito é dito ser a *ordem do corpo*. A ordem q de um corpo é sempre uma potência de um número primo, ou seja, $q = p^m$, sendo p primo. Além disso, existe sempre um único corpo de ordem p^m, para qualquer primo p e inteiro positivo m.

O menor inteiro positivo λ para o qual $\sum_{i=1}^{\lambda} 1 = 0$ em um corpo é dito ser a *característica do corpo*. Apesar dos algoritmos de codificação e decodificação serem descritos sobre corpos finitos quaisquer, como circuitos digitais operam naturalmente sobre base binária, as arquiteturas de implementação hardware são normalmente baseadas em corpos de característica 2. Estes corpos têm uma propriedade interessante, de que qualquer elemento α do corpo é sua própria inversa aditiva, ou seja, $\alpha + \alpha = \alpha - \alpha = 0$.

Considerando $\alpha \in \mathbb{F}_q$, o menor inteiro positivo s para o qual $\alpha^s = 1$ é dito ser a *ordem do elemento* α. Os elementos que apresentam ordem $s = q - 1$ (a maior ordem possível) são chamados de *elementos primitivos*, ou *raízes primitivas*. As potências de um elemento primitivo α geram todo o grupo multiplicativo $\{1, \alpha, \alpha^2, \alpha^2, \ldots, \alpha^{q-2}\}$ do corpo. Mostra-se que existem elementos primitivos para todos os corpos finitos.

Observe que $\alpha^{q-1} = 1$ para qualquer $\alpha \in \mathbb{F}_q$, e não apenas quando α for um elemento primitivo. Isso é conseqüência do fato da multiplicidade de qualquer elemento $\alpha \in \mathbb{F}_q$ sempre dividir $q - 1$. Várias arquiteturas para inversão de elementos em corpo finito tomam por base esta propriedade, e o fato de que $\alpha^{q-1} = \alpha \times \alpha^{q-2} = 1 \Rightarrow \alpha^{-1} = \alpha^{q-2}$.

Por simplicidade, o operador $\times$ de multiplicação será omitido no resto do texto. Por exemplo, onde for lido $\alpha\beta$, leia-se $\alpha \times \beta$.

A.2 Corpos finitos baseados em anéis de inteiros

O conjunto $\mathbb{Z}$ dos inteiros forma um domínio integral sob as operações de adição e multiplicação usuais, sendo denotado por $\mathbf{Z}$.

Um inteiro s é dito *divisível* pelo inteiro r, ou de modo igual r *divide* s, se $ra = s$ para algum inteiro a. Um inteiro p é dito *primo* se for divisível apenas por $\pm p$ ou ± 1. O *máximo divisor comum* $MDC\,(r, s)$ de dois inteiros r e s é o maior inteiro positivo que divide ambos r e s. O *mínimo múltiplo comum* $MMC\,(r, s)$ de dois inteiros r e s é o menor inteiro positivo que é divisível por ambos r e s.

Em geral, a divisão não é possível em um anel. Pode-se, entretanto, definir uma

divisão com resto e um cancelamento (resto igual a zero), razão pela qual o anel de inteiros constitui um domínio integral. O chamado *algoritmo da divisão* estabelece que, para todo par de inteiros c e d, com $d \neq 0$, existe um único par de inteiros Q (quociente) e s (resto), tal que $c = dQ + s$, em que $0 \leq s < |d|$. O resto s também pode ser escrito como $s = [c]_d$. Outra expressão comum é a de *congruência* $s \equiv c \mod d$. Dizer que s é congruente a c módulo d significa que s e c possuem o mesmo resto na divisão por d, mas s não necessariamente é menor que d.

Considere q um inteiro positivo. O anel dos inteiros módulo q, denotado por $\mathbf{Z}/q$, é o conjunto $\{0, \ldots, q-1\}$ associado à adição e à multiplicação definidos por $a+b = [a+b]_q$ e $ab = [ab]_q$.

Quando q for um inteiro primo, o anel $\mathbf{Z}/q$ constituirá um corpo, sendo denotado por $\mathbb{F}_q$. Portanto, tomando-se q inteiro primo, pode-se obter um corpo $\mathbb{F}_q$ a partir do anel de inteiros $\mathbf{Z}$ associado à operação de módulo q.

A.3 Corpos finitos baseados em anéis de polinômios

Um polinômio em $\mathbb{F}_q$ consiste numa expressão da forma

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \cdots + f_1 x + f_0,$$

em que x é uma variável e $f_{n-1}, \ldots, f_0 \in \mathbb{F}_q$. O polinômio nulo é $f(x) = 0$. Um *polinômio mônico* é um polinômio no qual o coeficiente f_{n-1} é igual a 1. O *grau* $\deg f(x)$ de um polinômio não nulo $f(x)$ é o índice do coeficiente f_{n-1}.

O conjunto de todos os polinômios em $\mathbb{F}_q$ associado à adição e à multiplicação de polinômios (com adição e multiplicação dos coeficientes em $\mathbb{F}_q$) constitui um anel, denotado por $\mathbb{F}_q[x]$.

Um polinômio $s(x)$ é *divisível* por $r(x)$, ou $r(x)$ *divide* $s(x)$, se existir um polinômio $a(x)$, tal que $s(x) = r(x)\,a(x)$. Um polinômio $p(x)$ é dito *irredutível* se for divisível apenas por $\alpha p(x)$ ou α, em que $\alpha \in \mathbb{F}_q$. Um polinômio mônico irredutível não nulo é dito um *polinômio primo*. O *máximo divisor comum* $MDC[r(x), s(x)]$ é o polinômio mônico de maior grau que divide ambos $r(x)$ e $s(x)$. O *mínimo múltiplo comum* $MMC[r(x), s(x)]$ é o polinômio mônico de menor grau que é divisível por ambos $r(x)$ e $s(x)$.

O chamado *algoritmo da divisão de polinômios* determina que, para todo par de polinômios $c(x)$ e $d(x)$, com $d(x) \neq 0$, existe um único par de polinômios $Q(x)$ (quociente) e $s(x)$ (resto), tal que $c(x) = d(x)\,Q(x) + s(x)$, em que $\deg s(x) < \deg d(x)$. O resto $s(x)$ também pode ser escrito como $s(x) = [c(x)]_{d(x)}$. Assim como no caso dos anéis de inteiros, a congruência $s(x) \equiv c(x) \mod d(x)$ indica que $s(x)$ e $c(x)$ possuem o mesmo resto na divisão por $d(x)$.

Assim como em certos casos é útil expressar inteiros como produto de inteiros primos (fatoração), também o é no caso de polinômios. Um polinômio não nulo $p(x)$ em um corpo $\mathbb{F}_q$ possui uma fatoração única (exceto pela ordem dos fatores) em um produto de um escalar (elemento de $\mathbb{F}_q$) e de polinômios primos em $\mathbb{F}_q$.

Um polinômio em $\mathbb{F}_q$ pode ser avaliado em qualquer elemento β de $\mathbb{F}_q$ substituindo-se a variável x por β. Um elemento β é um zero de ordem m de um polinômio $p(x)$ se e só se $(x-\beta)^m$ dividir $p(x)$ e $(x-\beta)^{m+1}$ não dividi-lo. Além disso, um polinômio $p(x)$ de grau n possui no máximo n zeros.

Para qualquer polinômio mônico $p(x)$ em $\mathbb{F}_q$ de grau não nulo, o *anel de polinômios módulo* $p(x)$ é o conjunto de todos os polinômios com grau menor que $\deg p(x)$, associado à adição e multiplicação de polinômios módulo $p(x)$. Este anel é denotado por $\mathbb{F}_q[x]/p(x)$. Quando $p(x)$ for um polinômio primo de grau m, o anel $\mathbb{F}_q[x]/p(x)$ será um *corpo extensão* $\mathbb{F}_{q^m}$ de ordem q^m. O corpo $\mathbb{F}_q$ será, portanto, um *subcorpo* de $\mathbb{F}_{q^m}$. Todos os q^m elementos do corpo extensão podem ser representados por polinômios em $\mathbb{F}_q$ com um grau máximo $m-1$. Estes q^m polinômios são as classes de resíduos módulo $p(x)$ de todos os polinômios em $\mathbb{F}_q$. Conclui-se, portanto, que o polinômio $p(x)$ é parâmetro determinante nos algoritmos das operações aritméticas neste corpo extensão.

Um *elemento primitivo* α do corpo $\mathbb{F}_q$ é tal que todo elemento não nulo de $\mathbb{F}_q$ pode ser expresso como potência de α. Os elementos primitivos são úteis na construção de corpos, já que, conhecido um elemento primitivo, é possível construir-se todo o corpo e sua tabela de multiplicação através das potências deste elemento (pode-se mostrar que todo corpo de Galois possui um elemento primitivo).

Um *polinômio primitivo* $p(x)$ em $\mathbb{F}_q$ consiste num polinômio primo em $\mathbb{F}_q$, tal que o elemento representado por x é primitivo no corpo extensão construído módulo $p(x)$. Existem polinômios primitivos de todos os graus em todos os corpos finitos. Além disso, mostra-se que um elemento primitivo de um corpo é um zero de qualquer polinômio primitivo neste corpo. O apêndice D apresenta uma lista não exaustiva de polinômios primitivos em $\mathbb{F}_2$.

Um corpo $\mathbb{F}_q$ é dito *fechado algebricamente* se todo polinômio $f(x) \in \mathbb{F}_q[x]$ de grau maior ou igual a 1 tiver raízes em $\mathbb{F}_q$. Qualquer corpo $\mathbb{F}_q$ possui uma extensão $\overline{\mathbb{F}_q}$ fechada algebricamente. Esta extensão $\overline{\mathbb{F}_q}$ é dita o *fecho algébrico* de $\mathbb{F}_q$.

A.4 Bases de corpos finitos

Um corpo extensão $\mathbb{F}_{q^m}$ de um corpo $\mathbb{F}_q$ pode ser visto como um espaço vetorial de dimensão m sobre $\mathbb{F}_q$. Cada elemento de $\mathbb{F}_{q^m}$ pode ser representado pela combinação

linear dos m elementos de uma base, em que os coeficientes da combinação linear são elementos de $\mathbb{F}_q$.

Embora, a princípio, existam diversas bases diferentes para um corpo finito, três são consideradas as mais importantes do ponto de vista técnico: as bases polinomiais, as bases normais e as bases duais.

Definição 51 (Base canônica) *O conjunto*

$$\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\},$$

em que α é uma raíz do polinômio irredutível $P(x)$ de grau m em $\mathbb{F}_q$, é chamada base canônica*, ou polinomial, ou padrão.*

Esta base é a mais intuitiva, pois está diretamente relacionada à representação dos elementos do corpo extensão como polinômios definidos num subcorpo. Um elemento $A \in \mathbb{F}_{q^m}$ é representado pelo polinômio $A(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{m-1} x^{m-1}$ definido em $\mathbb{F}_q$, e cada elemento de $\mathbb{F}_{q^m}$ representa uma classe de resíduos módulo $P(x)$. A representação polinomial $A(x)$ é equivalente a $A(\beta) = \alpha_0 + \alpha_1\beta + \alpha_2\beta^2 + \cdots + \alpha_{m-1}\beta^{m-1}$, sendo β uma raíz de $P(x)$.

Definição 52 (Base normal) *O conjunto*

$$\{1, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{m-1}}\},$$

em que α é uma raíz do polinômio irredutível $P(x)$ de grau m em $\mathbb{F}_q$, é chamada base normal *se os m elementos forem linearmente independentes.*

Mostra-se que existem bases normais para todo corpo finito. A representação em base normal é especialmente atraente para aplicações que envolvem a exponenciação em corpo finito, uma vez que a q-ésima potência de um elemento é obtida pela simples rotação dos coeficientes da representação do elemento nesta base.

Defina a função $\mathrm{Tr}(\alpha)$ de um elemento $\alpha \in \mathbb{F}_{q^m}$ relativa ao corpo $\mathbb{F}_q$ como sendo

$$\mathrm{Tr}(\alpha) = \alpha + \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{m-1}}.$$

Mostra-se que $\mathrm{Tr}(\alpha) \in \mathbb{F}_q$.

Definição 53 (Base dual) *Considere o conjunto $\{\beta_0, \beta_1, \ldots, \beta_{m-1}\}$ como sendo uma base para o corpo $\mathbb{F}_{q^m}$. Define-se como* base dual*, o conjunto $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ que satisfaz as condições*

$$\mathrm{Tr}(\beta_i \alpha_j) = \begin{cases} 1, & \text{se } i = j, \\ 0, & \text{se } i \neq j. \end{cases}$$

Mostra-se que existe uma base dual para toda base.

Apêndice B

Codificação para Controle de Erros

Este apêndice tem por objetivo revisar resumidamente os conceitos relacionados à codificação para controle de erros. Para detalhes sobre o assunto, veja [6, 72] ou qualquer outro texto básico sobre teoria da informação e codificação.

Este texto requer conhecimentos prévios de álgebra e aritmética de corpos finitos (*cf.* apêndice A).

B.1 Conceitos básicos

Em sistemas de comunicação digital, a informação a ser transmitida através de um canal sofre os efeitos de sua influência (introdução de ruído, desvanecimento, etc.), ocasionando erros de interpretação na recepção. O objetivo da codificação para controle de erros é adicionar símbolos extras (redundância) ao sinal, de modo a permitir na recepção a detecção e correção dos erros que porventura ocorram. Esta codificação aumenta a relação sinal / ruído ou, em outras palavras, reduz a probabilidade de erro de bit vista através do canal.

B.1.1 Códigos de bloco

Os códigos AG fazem parte de uma classe de códigos chamados códigos de bloco. Considere um corpo finito $\mathbb{F}_q$ de q elementos.

Definição 54 (Código de bloco) *Um* código de bloco C *de tamanho* M*, com símbolos em* $\mathbb{F}_q$*, consiste num conjunto de* M *seqüências* $\vec{c} = [c_0, c_1, \ldots c_{n-1}]$*, com* $c_i \in \mathbb{F}_q$*, de comprimento* n*, chamadas* palavras código.

O processo de codificação para um código de bloco consiste simplesmente no mapeamento bijetivo de palavras de informação de comprimento fixo k em palavras código de

comprimento n.

Se $q = 2$, os símbolos são chamados *bits* e o código é dito *binário*. Usualmente, para algum inteiro k, $M = q^k$ palavras código, sendo o código referido como um código (n, k). Um exemplo elementar é o código binário de repetição $(3, 1)$, no qual

informação		código
0	$\leftrightarrow$	000
1	$\leftrightarrow$	111

.

Na decodificaçao deste código, a decisão na recepção (após introdução de erros pelo canal) sobre qual informação foi transmitida é feita por maioria: se dois ou três bits forem 0, decide-se por 0; se dois ou três bits forem 1, decide-se por 1. Observa-se que este código possibilita a correção de apenas um erro por palavra código. Como será visto, este não é um *bom* código.

A *taxa de informação* R de um código de bloco[1] é definida por

$$R = \frac{k}{n}.$$

A princípio, será sempre melhor utilizar um código de comprimento n grande do que um código com palavras curtas. Isto, porque os erros são de natureza aleatória e ocorrem durante intervalos de tempo variáveis. Em alguns segmentos ocorrem mais erros que a média, em alguns menos. Portanto, para a mesma taxa R, de forma a possibilitar uma transmissão confiável de informação, é preferível utilizar um código de comprimento n grande, que seja capaz de corrigir uma quantidade maior de erros de uma única vez, mesmo que isto implique codificação e decodificação mais complexas. *Bons códigos* são, portanto, aqueles de comprimento n tão grande quanto possível, sem que isto comprometa a taxa do código ($R_{n\to\infty} = \frac{k}{n} = R_0 \neq 0$) ou sua capacidade de correção. O código de repetição $(n, 1)$, com $n \to \infty$, por exemplo, terá $R = 0$ (corrige tudo, mas não transmite nada).

B.1.2 Geometria dos códigos

Define-se como *distância de Hamming* $d(\vec{x}, \vec{y})$ entre duas palavras código $\vec{x}$ e $\vec{y}$ de comprimento n, com símbolos em $\mathbb{F}_q$, o número de posições nas quais elas diferem. Por exemplo, se $\vec{x} = [\alpha^2\ 0\ 1\ 0\ \alpha]$ e $\vec{y} = [\alpha\ 1\ 1\ 0\ \alpha^2]$ são palavras de comprimento $n = 5$ em $\mathbb{F}_4$, então $d(\vec{x}, \vec{y}) = 3$ (elas diferem nas posições 1, 2 e 5).

[1] Esta taxa R é adimensional e não deve ser confundida com a taxa de transmissão de informação R_T, medida em bits por segundo.

Considere $C = \{\vec{c}_i,\, i = 0,\, \ldots, M-1\}$ um código de bloco. A *distância mínima* d de C é a menor distância de Hamming entre duas palavras código de C. Ou seja,

$$d = \min_{\substack{\vec{c}_i, \vec{c}_j \in C \\ i \neq j}} d\left(\vec{c}_i, \vec{c}_j\right).$$

Um código (n, k) de distância mínima d também é referido como um código (n, k, d). Define-se *distância mínima relativa* δ de um código (n, k, d) como sendo

$$\delta = \frac{d}{n}.$$

Este conceito também é importante na avaliação de bons códigos. Por exemplo, um código (n, n), com $n \to \infty$, tem taxa $R = 1$, porém tem distância mínima relativa $\delta = 0$ (transmite tudo, mas não corrige nada).

Definindo uma *esfera* $S\left(\vec{x}, r\right)$ de raio r com centro em $\vec{x}$ como sendo

$$S\left(\vec{x}, r\right) = \left\{\vec{c} \in \mathbb{F}_q^n : d\left(\vec{x}, \vec{c}\right) \leq r\right\},$$

observa-se que é possível associar a todas as palavras de um código C de distância mínima d, esferas de raio $r = \frac{d-1}{2}$ que não se interceptam. A cardinalidade $V(n, r)$ desta esfera é dada por

$$V(n, r) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i. \tag{B.1}$$

Portanto,

$$|C|\, .V(n, r) \leq q^n, \tag{B.2}$$

em que $|C|$ é o número de palavras M do código C. Este resultado (*cf.* (B.2)) é conhecido como *limite de Hamming* para o código C.

O parâmetro $t = \lceil \frac{d-1}{2} \rceil$ denota a chamada *capacidade de correção* de um código de bloco, ou seja, o número de posições de um vetor recebido que um código é capaz de corrigir. Pode-se também ver a capacidade de correção de um código como o valor máximo de r, tal que esferas de raio r centradas nas palavras código jamais se interceptam. De modo similar, $\tau = d - 1$ denota a *capacidade de detecção* do código. Combinando estas duas capacidades, pode-se dizer em relação à capacidade total de um código que

$$2t + \tau \leq d.$$

B.1.3 Códigos lineares

A maioria dos bons códigos conhecidos pertence a uma classe de códigos chamados de códigos lineares. A estrutura imposta por esta classe proporciona meios para a busca de bons códigos e construção de codificadores e decodificadores práticos.

Considere o espaço vetorial $\mathbb{F}_q^n$. Um *código linear* é qualquer subespaço vetorial de $\mathbb{F}_q^n$. Em outras palavras, um código linear é um conjunto não vazio de n-úplas (vetores) em $\mathbb{F}_q$, chamadas palavras código, tal que a soma de duas palavras código é uma palavra código, e o produto de um escalar (elemento de $\mathbb{F}_q$) por uma palavra código também é uma palavra código.

Se um vetor $\vec{c} = [c_1\ c_2\ \cdots\ c_n]$ pertence a um código linear, assim como um vetor $[c_2\ \cdots\ c_n\ c_1]$ resultado da rotação de $\vec{c}$, então diz-se que este é um *código cíclico*. Os códigos AG em geral não são códigos cíclicos.

O *peso de Hamming* $w(\vec{c})$ de uma palavra código $\vec{c}$ é o número de posições não nulas desta palavra. O *peso mínimo* $w_{\min}$ de um código é o menor entre os pesos de Hamming de todas as palavras código não nulas. Por conseguinte, em um código linear,

$$d = w_{\min}.$$

Considere C um código linear em $\mathbb{F}_{q^m}$, em que $m > 1$. O *subcódigo de subcorpo* em $\mathbb{F}_q$ consiste em todas as palavras de C que possuem todas as coordenadas no subcorpo $\mathbb{F}_q$.

B.1.4 Matrizes dos códigos

Qualquer conjunto de k vetores linearmente independentes de uma base para o código C (base para o subespaço vetorial C) pode ser usado como linhas de uma matriz $G_{k\times n}$, chamada *matriz geradora* de C. Por exemplo, para algum código binário $(7, 4)$, pode-se ter

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Qualquer palavra código de C é uma combinação linear das linhas de G. O número de linhas k é a *dimensão* do código C.

As q^k palavras de informação (k-úplas) $\vec{i}$ são mapeadas nas q^k palavras código $\vec{c}$ da seguinte forma:

$$\vec{c} = \vec{i}G.$$

Por exemplo, para o código binário $(7, 4)$ cuja matriz G é dada acima, a palavra $\vec{i} = [1\ 1\ 0\ 1]$ é mapeada na palavra código $\vec{c} = [1\ 1\ 0\ 1\ 0\ 0\ 1]$.

Como C é um subespaço vetorial, ele possui um subespaço ortogonal $C^\perp$, que é o conjunto de todos os vetores ortogonais a C. Este código $C^\perp$ ortogonal é dito o *código dual* de C. Este código tem dimensão $n-k$ e uma matriz geradora $H_{n-k\times n}$. Por exemplo,

para o código binário $(7, 4)$ do exemplo acima, tem-se que

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Uma n-úpla $\vec{c}$ é uma palavra código se for ortogonal às linhas de H, ou seja, se

$$\vec{c}H^T = 0.$$

Por este motivo, H é dita a *matriz de paridade* (verificação de paridade) de C. Desta forma, pode-se definir o código C como sendo

$$C = \left\{ \vec{c} \in \mathbb{F}_q^n : \vec{c}H^T = 0 \in \mathbb{F}_q^{n-k} \right\}.$$

Para um vetor qualquer $\vec{v} \in \mathbb{F}_q^n$, define-se como *síndrome* de $\vec{v}$ os elementos de um vetor $\vec{S} = \vec{c}H^\top$ de comprimento $n-k$. Claramente, $\vec{S} = \vec{0}$ se e só se $\vec{v} \in C$. Em decodificadores a decisão brusca[2], as síndromes são a única informação disponível no receptor sobre os erros que porventura tenham sido adicionados aos sinal durante a transmissão. Para um vetor recebido $\vec{v} = \vec{c} + \vec{e}$, em que $\vec{c} \in C$ e $\vec{e}$ é um vetor de erros adicionado a $\vec{c}$ pelo canal, pode-se verificar que $\vec{S} = (\vec{c} + \vec{e})H^\top = \vec{c}H^\top + \vec{e}H^\top = \vec{e}H^\top$.

Observe, então, que uma palavra código $\vec{c} \in C$ de peso $w(\vec{c})$ implica uma relação de dependência entre as $w(\vec{c})$ colunas da matriz H correspondentes às coordenadas não nulas de $\vec{c}$. Portanto, observa-se que o código C possui distância mínima d se e só se nenhum grupo de $d-1$ colunas de H for linearmente dependente, caso em que haveria uma palavra $\vec{c}$ de peso $d-1$, tal que $\vec{c}H^T = 0$. Como as colunas de H são palavras de comprimento $n-k$, então o número máximo de colunas linearmente independentes de H é $n-k$. Portanto,

$$\begin{aligned} d - 1 \leq n - k \Rightarrow \\ d \leq n - k + 1. \end{aligned} \tag{B.3}$$

Esta desigualdade é conhecida como *limite de Singleton* para códigos lineares (n, k, d). Os códigos que apresentam igualdade nesta relação são ditos *códigos de máxima distância mínima* ("maximum-distance separable" – MDS).

Se G é a matriz geradora de um código MDS, então quaisquer k colunas de G são linearmente independentes. Como G é a matriz de paridade para o código dual, então o código dual possui distância mínima maior que k. Como o código dual possui dimensão $n-k$, sua distância mínima não pode exceder $k+1$. Conclui-se que o dual de um código MDS também é um código MDS.

[2] Além da decodificação a decisão brusca, existe ainda a decodificação a decisão suave, em que se dispõe no receptor de informações acerca da confiabilidade de cada elemento do vetor recebido.

B.1.5 Limites dos códigos

As seqüências de código desejadas (os bons códigos), como já fora explicado, são aquelas com comprimento n tão grande quanto possível, mantendo finitas e não nulas a taxa de informação R e a distância mínima relativa δ (capacidade de correção de erros).

Considere a notação $A(n,d)$ para o valor máximo de $M = |C|$ para o qual um código (n,k,d) (linear ou não) existe. A avaliação de bons códigos é feita através da função *taxa de informação assintótica* $R(\delta)$ definida por

$$R(\delta) = \lim_{n\to\infty} \frac{\log_q A(n, \delta n)}{n}, \tag{B.4}$$

que relaciona a taxa de informação R e a distância mínima relativa δ de um código quando n tende para infinito.

Durante muitos anos, o melhor limite inferior para $R(\delta)$ foi o limite de Gilbert-Varshamov, descrito em seguida.

Teorema 55 (Limite de Gilbert-Varshamov) *Considere* $0 \leq \delta \leq \frac{q-1}{q}$, *então*

$$R(\delta) \geq 1 - H_q(\delta), \tag{B.5}$$

sendo H_q *a função entropia definida por*

$$H_q(x) = \begin{cases} 0 & , x = 0; \\ x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x) & , 0 < x < \frac{q-1}{q}. \end{cases}$$

Prova. A prova deste teorema está descrita com detalhes no apêndice B. ■

No capítulo 2, após descritos os códigos AG, é apresentado um novo limite inferior para $R(\delta)$, que é melhor que o de Gilbert-Varshamov para todo $q \geq 49$ em parte do intervalo $\left[0, \frac{q-1}{q}\right]$.

B.2 Códigos de Reed-Solomon

Convencionalmente, os códigos RS são definidos a partir de um polinômio gerador ou, equivalentemente, a partir de uma matriz geradora (ou de uma matriz de paridade) [6,72]. Além desta, uma construção distinta destes códigos é apresentada em seguida no intuito de facilitar a definição dos códigos AG como extensão destes [8,41,44].

B.2.1 Definição convencional

Considere $\mathbb{F}_q[x]/(x^n - 1)$ o anel dos polinômios em $\mathbb{F}_q$ com grau menor que n. Toda palavra código $[c_0\ c_1\ \ldots\ c_{n-1}] \in \mathbb{F}_q^n$ pode ser vista como um polinômio $c_0 + c_1 x + \cdots +$

$c_{n-1}x^{n-1} \in \mathbb{F}_q[x]/(x^n-1)$. Neste anel, a multiplicação de um polinômio por x equivale ao deslocamento cíclico da palavra equivalente. Observa-se, então, que existe uma correspondência entre um código cíclico e um ideal[3] contido em $\mathbb{F}_q[x]/(x^n-1)$. Mostra-se [9, pp. 37–44] que este ideal (o código cíclico) pode ser gerado por um único polinômio $g(x)$, divisor de x^n-1, conhecido como *polinômio gerador*. O polinômio $g(x)$ é, por definição, o máximo divisor comum – MDC dos polinômios do ideal (ele gera este ideal). Sendo $g(x)$ um polinômio de grau $n-k$, um código cíclico de comprimento n é o resultado da multiplicação de $g(x)$ pelos polinômios $i(x)$ de grau menor que k (informação). Ou seja,

$$c(x) = i(x)g(x).$$

Os códigos RS são códigos cíclicos de comprimento $n = q-1$, em $\mathbb{F}_q$, cujo polinômio gerador é da forma

$$g(x) = \prod_{i=1}^{d'-1} \left(x - \alpha^i\right),$$

em que α é um elemento primitivo de $\mathbb{F}_q$ e d' é a distância mínima projetada do código.

Observe que $g(x)$ é sempre um polinômio de grau $d'-1 = n-k \Rightarrow d' = n-k+1$. Deste resultado e do limite de Singleton (*cf.* (B.3)), tem-se que a distância mínima d do código é

$$d = d' = n-k+1.$$

Este é, então, um código MDS. Sua dimensão é $k = n-d+1$. Este código é capaz de corrigir $d = 2t+1 \Rightarrow t = n-k$ erros.

Pode-se também definir os *códigos RS estendidos* pela adição de uma componente, fazendo $n = q$. Estes códigos, também MDS, diferentemente dos anteriores, não são cíclicos.

B.2.2 Definição geométrica

Observe agora uma formulação diferente para os códigos RS apresentados acima. Primeiramente, considere o conjunto $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ de n elementos distintos de $\mathbb{F}_q$. Denote por $L \subset \mathbb{F}_q[x]$ o conjunto de polinômios de grau menor que $k \leq n$. Defina um código C

[3]Um *ideal* I é um subconjunto de um anel A que satisfaz os seguintes axiomas:

1. I é um grupo abeliano sob a operação de adição;
2. Para todo $a \in I$ e $b \in A$, tem-se que $ab \in I$.

Uma maior discussão sobre este assunto será feita no capítulo seguinte.

como sendo

$$C = \{(f(\alpha_0),\ f(\alpha_1),\ \ldots,\ f(\alpha_{n-1})) : f \in L\}. \tag{B.6}$$

Este código possui comprimento n e dimensão k.

Como um polinômio de grau menor que k possui no máximo $k-1$ zeros, cada palavra código de C (*cf.* (B.6)) possui peso no mínimo $n-(k-1) = n-k+1$. Daí e do limite de Singleton (*cf.* (B.3)), conclui-se que este é um código MDS, ou seja, possui distância mínima $d = n-k+1$. O mesmo código RS cíclico da definição convencional vista na subseção anterior é obtido nesta construção para $n = q-1$, e o código RS estendido para $n = q$.

Existe ainda uma forma mais geral do código dado por (B.6). Considere os vetores $\vec{a} = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ de n elementos distintos de $\mathbb{F}_{q^m}$ e $\vec{v} = (v_0, v_1, \ldots, v_{n-1})$ de elementos não nulos e não necessariamente distintos de $\mathbb{F}_{q^m}$. Considere L o conjunto de polinômios de grau menor que k em $\mathbb{F}_{q^m}[x]$. O código

$$GRS_k(\vec{a}, \vec{v}) = \{(v_0 f(\alpha_0),\ v_1 f(\alpha_1),\ \ldots,\ v_{n-1} f(\alpha_{n-1})) : f \in L\} \tag{B.7}$$

possui os mesmos parâmetros do código anterior e é conhecido como *código RS generalizado*.

Considere agora o conjunto dos pares de elementos (α_1, α_2), com $\alpha_i \in \mathbb{F}_q$. Considere os pares que são múltiplos escalares entre si como sendo de uma mesma classe de equivalência, ou seja, $(\alpha_1, \alpha_2) \equiv (\beta\alpha_1, \beta\alpha_2)$, para todo $\beta \in \mathbb{F}_q^*$. Cada classe de equivalência é representada por um dos pares $(1, \alpha)$, com $\alpha \in \mathbb{F}_q$, ou $Q = (0, 1)$ (Q é chamado ponto no infinito). Este conjunto de classes de equivalência é conhecido como *linha projetiva* $\mathbb{P}^1$ (uma maior explanação sobre o assunto é feita no apêndice C). Por exemplo, considerando o corpo $\mathbb{F}_4$, tem-se que

$$\mathbb{P}^1 = \left\{(1,0), (1,1), (1,\alpha), \left(1,\alpha^2\right), Q\right\}.$$

Uma *função racional* em $\mathbb{P}^1$ é um quociente da forma $\frac{a(x,y)}{b(x,y)}$, em que $a(x,y)$ e $b(x,y)$ são polinômios homogêneos de mesmo grau em $\mathbb{F}_q$ (sem esta restrição o quociente não seria definido em $\mathbb{P}^1$). Um ponto de $\mathbb{P}^1$ é um pólo de uma função racional $\frac{a(x,y)}{b(x,y)}$ se o polinômio $b(x,y)$ for zero neste ponto (e $a(x,y)$ não for zero). Defina, então, L como sendo o espaço vetorial de todas as funções racionais em $\mathbb{P}^1$ que não possuem pólos em $\mathbb{P}^1$, exceto possivelmente pólos de ordem menor que k em Q. É fácil observar que funções racionais da forma $\frac{a(x,y)}{x^l}$, com $l < k$, em que $a(x,y)$ é um polinômio homogêneo de grau l, possuem as propriedades descritas. Segue, então, a definição dos códigos RS.

Definição 56 (Códigos de Reed-Solomon) *Um código RS pode ser descrito pelo conjunto de n-úplas*

$$C = \{(\ f(P_1), f(P_2), \ldots,\ f(P_n)) : f \in L\}, \tag{B.8}$$

em que $P_1, P_2, \ldots, P_n \in \mathbb{P}^1$ *são pontos diferentes de* Q.

Em outras palavras, o que se fez foi tomar uma linha projetiva, um conjunto de n pontos desta linha e um espaço vetorial de funções definido a partir de um ponto da linha projetiva diferente dos n pontos supracitados. O código C fornecido por (B.8) consiste apenas num conjunto de n-úplas de valores destas funções nestes pontos. Este processo de avaliar funções racionais em pontos de uma curva (no caso uma linha) constitui uma das idéias centrais na construção de códigos AG, que será chamada de *construção por funções*. Uma segunda idéia central na construção de códigos AG é derivada da definição dos códigos de Goppa na seção seguinte.

B.3 Códigos de Goppa

Antes de definir os códigos de Goppa, será apresentada uma conhecida classe de códigos cíclicos, os chamados códigos BCH (Bose-Chaudhuri-Hocquenghem) [6, 72].

B.3.1 Códigos BCH

Considere α um elemento de ordem n do corpo $\mathbb{F}_{q^m}$, em que n divide $q^m - 1$. Considere $g(x) \in \mathbb{F}_q[x]$ o polinômio de menor grau cujos zeros sejam

$$\left\{\alpha^i : i = 1, 2, \ldots, 2t\right\},$$

para algum inteiro $t \geq 1$. Faça o grau de $g(x)$, referido como o polinômio gerador do código, igual a $n - k$. Mostra-se que $n - k \leq 2tm$ [8]. Então,

$$C = \{a(x)\, g(x) : \deg[a(x)] < k,\ a(x) \in \mathbb{F}_q[x]\} \tag{B.9}$$

é um *código BCH* de comprimento n, dimensão $k \geq n - 2tm$ e distância mínima $d \geq 2t + 1$.

Observe que, tomando-se o polinômio

$$h(x) = \prod_{i=1}^{2t} \left(x - \alpha^i\right) \in \mathbb{F}_{q^m}[x]$$

no lugar de $g(x)$ e substituindo-se o corpo $\mathbb{F}_q$ por $\mathbb{F}_{q^m}$, o código da equação B.9 torna-se um código RS C' de comprimento n, dimensão k e distância mínima $d = 2t + 1$. Portanto, o código BCH C da equação B.9 consiste num sub-código de subcorpo de C', ou seja,

$$C = C' \cap \mathbb{F}_q^n.$$

B.3.2 Códigos de Goppa

Utilizando a mesma notação da definição dos códigos BCH acima, em que as palavras código são da forma $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$, tal que $c(\alpha^i) = 0$, para $i = 1, \ldots, d-1$, considere o seguinte cálculo:

$$\begin{aligned}(x^n - 1)\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha^{-i}} &= \sum_{i=0}^{n-1} c_i \frac{x^n - 1}{x - \alpha^{-i}} \\ &= \sum_{i=0}^{n-1} c_i \left(x^{n-1} + \alpha^{-i}x^{n-2} + \cdots + \alpha^{-(n-2)i}x + \alpha^{-(n-1)i}\right) \\ &= \sum_{i=0}^{n-1} c_i \sum_{j=0}^{n-1} x^j \left(\alpha^{-i}\right)^{n-1-j} \\ &= \sum_{j=0}^{n-1} x^j \sum_{i=0}^{n-1} c_i \left(\alpha^{j+1}\right)^i .\end{aligned}$$

Para valores de j no intervalo $[0, d-2]$, o somatório $\sum_{i=0}^{n-1} c_i \left(\alpha^{j+1}\right)^i = c\left(\alpha^{j+1}\right)$ é por definição nulo. Portanto, para algum polinômio $f(x)$, tem-se que

$$\begin{aligned}(x^n - 1)\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha^{-i}} &= \sum_{j=d-1}^{n-1} x^j \sum_{i=0}^{n-1} c_i \left(\alpha^{j+1}\right)^i \\ &= x^{d-1} f(x),\end{aligned}$$

ou seja, o somatório $\sum_{i=0}^{n-1} \frac{c_i}{x-\alpha^{-i}}$ é necessariamente divisível por x^{d-1} (x^{d-1} não divide $x^n - 1$). Portanto,

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha^{-i}} \equiv 0 \mod x^{2t}. \tag{B.10}$$

Por conseguinte, um vetor $[c_0\ c_1\ \cdots c_{n-1}]$, com $c_i \in \mathbb{F}_q$, é uma palavra código se satisfizer a equação B.10. Dependendo do corpo utilizado, esta construção acima fornece um código RS ou um código BCH. A passagem destes para os códigos de Goppa agora envolve apenas a substituição da seqüência $\{\alpha^i : i = 1, 2, \ldots, 2t\}$ usada (α é um elemento primitivo de ordem n do corpo $\mathbb{F}_{q^m}$) por um conjunto arbitrário de elementos distintos, e do monômio x^{2t} por um polinômio geral $g(x)$.

Definição 57 (Códigos de Goppa) *Considere $L = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ um conjunto arbitrário de n elementos distintos de $\mathbb{F}_{q^m}$ e $g(x) \in \mathbb{F}_{q^m}[x]$ um polinômio mônico, tal que $g(\alpha_i) \neq 0$, para $i = 0, 1, \ldots, n-1$. O* código de Goppa $\Gamma(L, g)$ *é o conjunto de palavras $(c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$, tal que*

$$\sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \mod g(x). \tag{B.11}$$

O código $\Gamma(L, g)$ definido pela equação B.11 é um subcódigo de subcorpo do dual do código RS generalizado (equação B.7). Para verificar isto, considere $g(x) = \sum_{i=0}^{t} g_i x^i$. Então,

$$\phi(x) = \frac{g(x) - g(a)}{x - a} = \sum_{k+j \leq t-1} g_{k+j+1} a^j x^k$$

é um polinômio de grau menor que t, para qualquer a. Como

$$(x - a)\,\phi(x) \equiv -g(a) \mod g(x),$$

pode-se reescrever a equação B.11 como sendo

$$\sum_{i=0}^{n-1} \frac{c_i}{(x - \alpha_i)\,\phi(x)} \phi(x) \equiv 0 \mod g(x) \Rightarrow$$

$$\sum_{i=0}^{n-1} \frac{c_i}{-g(\alpha_i)} \sum_{k+j \leq t-1} g_{k+j+1} (\alpha_i)^j x^k = 0 \Rightarrow$$

$$\sum_{i=0}^{n-1} c_i h_i \sum_{k+j \leq t-1} g_{k+j+1} (\alpha_i)^j x^k = 0, \tag{B.12}$$

em que $h_i = \frac{1}{g(\alpha_i)}$. Observe que o coeficiente de x^k na equação B.12 é zero para $0 \leq k \leq t - 1$. Isto significa que, se $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ é uma palavra código, então o produto interno de $\mathbf{c}$ com as linhas da matriz

$$\begin{bmatrix} h_0 g_t & \cdots & h_{n-1} g_t \\ h_0 (g_{t-1} + g_t \alpha_0) & \cdots & h_{n-1} (g_{t-1} + g_t \alpha_{n-1}) \\ \vdots & \cdots & \vdots \\ h_0 \sum_{i=1}^{t} g_i \alpha_0^{i-1} & \cdots & h_{n-1} \left(\sum_{i=1}^{t} g_i \alpha_{n-1}^{i-1} \right) \end{bmatrix}$$

deve ser zero. Usando operações elementares nas linhas desta matriz, obtém-se a seguinte matriz de paridade para o código $\Gamma(L, g)$:

$$H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_0 \alpha_0 & h_1 \alpha_1 & \cdots & h_{n-1} \alpha_{n-1} \\ \vdots & \vdots & \cdots & \vdots \\ h_0 \alpha_0^{t-1} & h_1 \alpha_1^{t-1} & \cdots & h_{n-1} \alpha_{n-1}^{t-1} \end{bmatrix}.$$

Compare esta matriz com os códigos da equação B.7, em que $\mathbf{v} = (h_0, h_1, \ldots, h_{n-1})$. Observa-se que H é a matriz geradora do código $GRS_k(\mathbf{a}, \mathbf{v})$ da equação B.7. Portanto, o código de Goppa $\Gamma(L, g)$ é um subcódigo de subcorpo do dual de um código RS generalizado.

Como o posto da matriz H sobre $\mathbb{F}_{q^m}$ é exatamente t, seu posto sobre $\mathbb{F}_q$ é no máximo mt. Portanto, a dimensão do código de Goppa $\Gamma(L, g)$ é $k \geq n - mt$ e sua distância mínima é $d \geq t + 1$, em que t é o grau de $g(x)$.

B.4 Transição para os códigos AG

Agora será dada uma nova formulação aos códigos de Goppa, de forma a colocar em perspectiva sua transição para os códigos AG.

Considere a função

$$f(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} = \frac{\omega(x)}{\lambda(x)}$$

correspondente à palavra código $(c_0, c_1, \ldots, c_{n-1})$, em que

$$\lambda(x) = \prod_i (x - \alpha_i) \in \mathbb{F}_{q^m}[x],$$

$\deg \omega(x) < \deg \lambda(x) = n$ e $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ é um conjunto arbitrário de n elementos distintos de $\mathbb{F}_{q^m}$. Portanto,

$$c_i = f(x)(x - \alpha_i)\big|_{x=\alpha_i}$$

é obtido pelo cancelamento do pólo α_i em $f(x)$ e avaliação no próprio ponto α_i. Em outras palavras, c_i é o resíduo de $f(x)$ em α_i, denotado por $\mathrm{Res}_{\alpha_i}(f)$.

Considere

$$\mathcal{X}_j(x) = \prod_{i=1,\, i\neq j}^{n} (x - \alpha_i) = \frac{\lambda(x)}{x - \alpha_i}$$

e

$$f(x) = \frac{\omega(x)}{\lambda(x)} = \frac{g(x)\, q(x)}{\lambda(x)},$$

uma vez que, por definição, $g(x) | f(x)$ ($g(x)$ é o mesmo da equação B.11). Pode-se, então, expressar o resíduo de $f(x)$ em α_i por

$$\mathrm{Res}_{\alpha_i}(f) = \left.\frac{\omega(x)(x - \alpha_i)}{\lambda(x)}\right|_{x=\alpha_i} = \frac{g(\alpha_i)}{\mathcal{X}(\alpha_i)} q(\alpha_i),$$

que é zero apenas se $q(\alpha_i) = 0$.

Agora, generalizando este conceito, defina um espaço vetorial L de funções racionais, tal que

1. $f(x) \in L$ possui pelo menos os mesmos zeros que $g(x)$, com pelo menos a mesma multiplicidade;

2. $f(x) \in L$ não possui pólos, exceto possivelmente em $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$, caso em que os pólos são de ordem 1.

Defina, então, um código C' de comprimento n em $\mathbb{F}_{q^m}$ como sendo

$$C' = \left\{ \left(\mathrm{Res}_{\alpha_0} f, \mathrm{Res}_{\alpha_1} f, \ldots, \mathrm{Res}_{\alpha_{n-1}} f\right) : f \in L \right\}. \tag{B.13}$$

Basicamente, este código C' fornecido por (B.13) consiste no conjunto das n-úplas dos resíduos das funções do espaço de funções L definido pelos pontos de $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ nestes pontos. Este processo de determinar os resíduos de funções racionais em um conjunto de pontos constitui a segunda idéia central na construção de códigos AG, que será chamada de *construção por diferenciais*.

O código de Goppa definido pela equação B.11 é um subcódigo de subcorpo deste código C' em $\mathbb{F}_q$. Como o código de Goppa é também um subcódigo de subcorpo do dual do código C definido pela equação B.8, observa-se, então, que existe uma relação de dualidade entre as construções por funções e por diferenciais usadas nas definições dos códigos C (equação B.8) e C' (equação B.13), respectivamente.

Do que foi apresentado neste capítulo, dois enfoques usados na definição de códigos são de grande importância na construção dos códigos AG:

- A avaliação de funções racionais em um conjunto fixo de pontos distintos (equação B.8);
- O resíduo de funções racionais em um conjunto fixo de pontos distintos (equação B.13).

A definição dos códigos AG, descrita em detalhes no capítulo 2, utiliza estes dois enfoques, que fornecem códigos duais, diferindo apenas na obtenção do conjunto de pontos distintos: são pontos de uma curva algébrica em um corpo finito. Além disso, o espaço de funções racionais é definido a partir de pontos desta mesma curva algébrica.

Apesar destas abordagens serem simples, a definição dos parâmetros do código dependerá fortemente da teoria das curvas algébricas. O apêndice C dedica-se à apresentação desta teoria matemática, requisito necessário ao entendimento dos códigos AG, definidos no capítulo 2.

Apêndice C

Geometria Algébrica

Este apêndice tem por objetivo descrever resumidamente alguns conceitos da geometria algébrica, como os de variedades afins e projetivas, de bases de Gröbner, corpos de funções, anéis de coordenadas, anéis locais e divisores, necessários ao estudo dos códigos AG e seus esquemas de decodificação.

Não é pretensão do presente texto constituir uma referência completa ou pormenorizada sobre este assunto. Grande parte das provas não são apresentadas aqui. Abordagens mais detalhadas sobre esta teoria e provas dos resultados apresentados podem ser obtidos em diversas publicações [9, 24, 46, 55, 70].

C.1 Ideais e variedades

Os conceitos de ideal e variedade e suas relações são a base para a teoria matemática denominada geometria algébrica. A natureza geométrica provém das variedades, que são as curvas, superfícies e objetos de maior dimensão definidos por equações polinomiais. Os ideais, subestruturas dos anéis polinomiais $\mathbb{F}_q[x_1, \ldots, x_n]$, constituem a "álgebra" relacionada às variedades.

C.1.1 Variedade afim

Defina um *monômio* em $x_1, \ldots, x_n$ por

$$x^{\alpha} = x_1^{\alpha_1} \ldots x_n^{\alpha_n},$$

em que $\alpha_1, \ldots, \alpha_n$ são inteiros não negativos. O grau de x^{α} é dado por $\alpha_1 + \cdots + \alpha_n$.

Considere $\mathbb{F}_p$ um corpo finito com p elementos e $\mathbb{F}_q$ seu fecho algébrico. Um *polinômio* f em $x_1, \ldots, x_n$ com coeficientes em $\mathbb{F}_q$ consiste numa combinação linear de monômios

da forma

$$f = \sum_{\alpha} a_\alpha x^\alpha, \quad a_\alpha \in \mathbb{F}_q,$$

em que a soma é feita sobre um número finito de n-úplas $\alpha = (\alpha_1, \ldots, \alpha_n)$. O anel dos polinômios em $x_1, \ldots, x_n$ com coeficientes em $\mathbb{F}_q$ é denotado por $\mathbb{F}_q[x_1, \ldots, x_n]$.

Defina agora *espaço afim* de dimensão n, sobre $\mathbb{F}_q$, como o conjunto

$$\mathbb{A}^n = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in \mathbb{F}_q\}.$$

Um elemento $P \in \mathbb{A}^n$ é dito ser um *ponto* do espaço $\mathbb{A}^n$. Observe que um polinômio $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ pode ser considerado como um mapeamento $f : \mathbb{A}^n \to \mathbb{F}_q$ dado por

$$f(P) = f(a_1, \ldots, a_n),$$

em que $P = (a_1, \ldots, a_n) \in \mathbb{A}^n$. Se $f(P) = 0$, diz-se que P é um *zero* de f.

Definição 58 (Variedade afim) *Considere $f_1, \ldots, f_s$ polinômios em $\mathbb{F}_q[x_1, \ldots, x_n]$. A* variedade afim $\mathbf{V}(f_1, \ldots, f_s)$ *definida por $f_1, \ldots, f_s$ é, então, dada por*

$$\mathbf{V}(f_1, \ldots, f_s) = \{P \in \mathbb{A}^n : f_i(P) = 0,\ 1 \leq i \leq s\}.$$

Em outras palavras, uma variedade afim $\mathbf{V}(f_1, \ldots, f_s) \subset \mathbb{A}^n$ é o conjunto de soluções para o sistema de equações $f_1(x_1, \ldots, x_n) = \cdots = f_s(x_1, \ldots, x_n) = 0$.

C.1.2 Ideal

Definição 59 (Ideal) *Um subconjunto $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ constitui um* ideal *se satisfizer os seguintes axiomas:*

1. *I é um grupo abeliano sob a adição;*
2. *Se $f \in I$ e $h \in \mathbb{F}_q[x_1, \ldots, x_n]$, então $hf \in I$.*

Considere $f_1, \ldots, f_s$ polinômios pertencentes ao anel $\mathbb{F}_q[x_1, \ldots, x_n]$. O conjunto denotado por $\langle f_1, \ldots, f_s \rangle$ e dado por

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i : h_1, \ldots, h_s \in \mathbb{F}_q[x_1, \ldots, x_n] \right\} \tag{C.1}$$

é um ideal gerado por $f_1, \ldots, f_s$ (isto é facilmente verificado aplicando-se os axiomas da definição 59).

Considere $V \subset \mathbb{A}^n$ uma variedade afim. O conjunto denotado por $\mathbf{I}(V)$ e dado por

$$\mathbf{I}(V) = \{f \in \mathbb{F}_q[x_1, \ldots, x_n] : f(P) = 0,\ P \in V\} \tag{C.2}$$

é um ideal gerado por V (isto é verificado também aplicando-se os axiomas da definição acima).

Todo ideal pode ser gerado por um conjunto finito de funções linearmente independentes denominado *base*. Um ideal que pode ser gerado por uma única função é dito *principal*. Um ideal $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ é dito *primo* se $I \neq \mathbb{F}_q[x_1, \ldots, x_n]$ e se, para $ab \in I$, $a \in I$ ou $b \in I$. Um ideal $I \subset A$ é dito *máximo* em um conjunto A se não existir outro ideal em A que contenha I.

Um ideal I é dito *radical* se o fato de que $f^m \in I$, para $m \geq 1$ inteiro, implicar que $f \in I$. Para uma variedade V qualquer, se $f^m \in \mathbf{I}(V)$, então $f \in \mathbf{I}(V)$, uma vez que $[f(x)]^m = 0 \Rightarrow f(x) = 0$. Portanto, $\mathbf{I}(V)$ é sempre um ideal radical. Por conseguinte, um ideal $I = \langle f_1, \ldots, f_s \rangle$, em que $f^k \in I$ e $f^{k-1} \notin I$, não pode ser representado na forma $\mathbf{I}(V)$, ou seja, não pode ser gerado por qualquer variedade V.

Teorema 60 ("Nullstellensatz" de Hilbert) *Considere o anel de funções $\mathbb{F}_q[x_1, \ldots, x_n]$ e as funções $f, f_1, \ldots, f_s$ pertencentes a este anel . A função $f \in \mathbf{I}(\mathbf{V}(f_1, \ldots, f_s))$ se e só se $f^m \in \langle f_1, \ldots, f_s \rangle$, para algum inteiro $m \geq 1$.*

C.1.3 Bases de Gröbner

A todo monômio $x^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$ pode-se associar uma n-úpla $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_+^n$, e vice-versa. Portanto, uma mesma regra de ordenação $>$ pode ser aplicada tanto a monômios quanto ao espaço $\mathbb{Z}_+^n$, ou seja, se $\alpha > \beta$ segundo a ordenação adotada, então $x^\alpha > x^\beta$, e vice-versa.

Definição 61 (Ordenação de monômios) *Uma* ordenação de monômios *atribuída aos elementos do anel $\mathbb{F}_q[x_1, \ldots, x_n]$ é qualquer relação $>$ sobre as n-úplas $\alpha \in \mathbb{Z}_+^n$, ou sobre os monômios x^α, que satisfaz as condições:*

1. *$>$ é uma ordenação linear, ou seja, aplica-se a todos os vetores de $\mathbb{Z}_+^n$;*
2. *Se $\alpha > \beta$, com $\alpha, \beta, \gamma \in \mathbb{Z}_+^n$, então $\alpha + \gamma > \beta + \gamma$;*
3. *Todo subconjunto finito não vazio de $\mathbb{Z}_+^n$ possui um elemento menor e um elemento maior sob a ordenação $>$.*

Um exemplo de ordenação de monômios é a chamada *ordenação lexicográfica*, denotada por $>_{lex}$, em que $\alpha >_{lex} \beta$, com $\alpha, \beta \in \mathbb{Z}_+^n$, se no vetor $\alpha - \beta \in \mathbb{Z}^n$ o elemento não nulo mais à esquerda for positivo. Por exemplo, para $n = 2$, tem-se que $(0,0) <_{lex} (0,1) <_{lex} \cdots <_{lex} (1,0) <_{lex} (1,1) <_{lex} \cdots <_{lex} (2,0) <_{lex} (2,1) <_{lex} \cdots <_{lex} (3,0) <_{lex} (3,1) <_{lex} \cdots$.

Considere $f = \sum_\alpha a_\alpha x^\alpha$ um polinômio não nulo em $\mathbb{F}_q[x_1, \ldots, x_n]$ e $>$ uma ordenação de monômios. Defina, então, o *grau* de f, denotado por $\deg(f)$, como o maior vetor α, expoente de x, segundo a ordenação $>$. Defina também o *coeficiente líder* de f, denotado por $\mathrm{lc}(f)$, como sendo o coeficiente $a_{\deg(f)}$, e o *termo líder* de f, denotado por $\mathrm{lt}(f)$, como sendo $a_{\deg(f)}x^{\deg(f)}$.

Considere uma ordem de monômios $>$ e um conjunto de funções $f_1, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$ ordenadas segundo a ordenação $>$. Tem-se que toda função $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ pode ser escrita na forma

$$f = q_1 f_1 + \cdots + q_s f_s + r,$$

em que $q_1, \ldots, q_s, r \in \mathbb{F}_q[x_1, \ldots, x_n]$, e, ou $r = 0$, ou r é uma combinação linear de monômios não divisíveis por $\mathrm{lt}(f_1), \ldots, \mathrm{lt}(f_s)$. O algoritmo 62 descreve em detalhes a divisão de polinômios em várias variáveis.

Algoritmo 62 (Divisão de polinômios em várias variáveis)

Entradas:

- *Os polinômios* $f, f_1, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$ *ordenados segundo a ordem* $>$.

Saídas:

- $q_1, \ldots, q_s, r \in \mathbb{F}_q[x_1, \ldots, x_n]$.

Inicialização:

- $q_1 = 0, \ldots, q_s = 0, r = 0$.

<<< **Única iteração** >>>
$p = f$
WHILE $p \neq 0$ ***DO***
  $i = 1$
  $ocorreudivisao = false$
  WHILE $i \leq s$ ***AND*** $ocorreudivisao == false$ ***DO***
    IF $\mathrm{lt}(f_i)$ *divide* $\mathrm{lt}(p)$ ***THEN***
      $q_i = q_i + \mathrm{lt}(p)/\mathrm{lt}(f_i)$
      $p = p - f_i[\mathrm{lt}(p)/\mathrm{lt}(f_i)]$
      $ocorreudivisao = true$
    ELSE
      $i = i + 1$
  IF $ocorreudivisao == false$ ***THEN***
    $r = r + \mathrm{lt}(p)$
    $p = p - \mathrm{lt}(p)$

Diferentemente do caso dos polinômios em uma única variável, no caso geral da divisão de polinômios em n variáveis, os valores dos quocientes e do resto não são únicos e dependem da ordem de monômios adotada. Apesar disto, se uma função f pode ser escrita na forma

$$f = q_1 f_1 + \cdots + q_s f_s,$$

então $f \in \langle f_1, \ldots, f_s \rangle$. Isto implica que $r = 0$ é uma condição suficiente para a pertinência de f ao ideal $\langle f_1, \ldots, f_s \rangle$. Contudo, não é uma condição necessária, pois a mesma divisão pode fornecer um resto $r \neq 0$ para uma ordenação de monômios diferente.

Há, entretanto, a possibilidade de, a partir de um conjunto de funções geradoras de um ideal I, obter-se um conjunto diferente de funções geradoras de I que apresentem boas propriedades, tais como a de serem univocamente determinadas e a de tornarem a condição $r = 0$ suficiente e necessária para a pertinência de uma função f ao ideal I.

Definição 63 (Ideal de monômios) *Um ideal $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ é dito ser um* ideal de monômios *se for constituído de todos os polinômios na forma da soma finita*

$$\sum_{\alpha \in A} h_\alpha x^\alpha,$$

em que $A \subset \mathbb{Z}_+^n$ e $h_\alpha \in \mathbb{F}_q[x_1, \ldots, x_n]$.

Todo ideal de monômios I pode ser escrito na forma $\langle x^{\alpha_1}, \ldots, x^{\alpha_s} \rangle$, em que $\alpha_1, \ldots, \alpha_s \in A$. Em outras palavras, todo ideal de monômios I possui uma base finita de monômios. Observe, então, que um monômio x^β pertence a um ideal de monômios $I = \langle x^\alpha : \alpha \in A \rangle$ se e só se for divisível por algum x^α, para $\alpha \in A$. Como conseqüência, todo polinômio $f \in I$ é necessariamente uma combinação linear de monômios de I, ou seja, todo termo de f pertence também a I.

Considere agora $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ um ideal qualquer diferente de $\{0\}$ e denote por $\mathrm{lt}(I)$ o conjunto dos termos líderes dos elementos de I. Tem-se que o ideal gerado pelos elementos de $\mathrm{lt}(I)$, denotado por $\langle \mathrm{lt}(I) \rangle$, constitui um ideal de monômios. Além disso, existem sempre funções $g_1, \ldots, g_t \in I$, tais que $\langle \mathrm{lt}(I) \rangle = \langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t) \rangle$. Associado a este fato, o chamado *teorema das bases de Hilbert* afirma que todo e qualquer ideal $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ possui sempre um conjunto gerador finito, ou seja, todo ideal I pode ser escrito na forma $\langle g_1, \ldots, g_t \rangle$, para algum $g_1, \ldots, g_t \in I$.

Definição 64 (Bases de Gröbner) *Dada uma ordem de monômios, um subconjunto finito $\mathcal{G} = \{g_1, \ldots, g_t\}$ de um ideal I é dito ser uma* base de Gröbner *(também chamada* base padrão*) se*

$$\langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t) \rangle = \langle \mathrm{lt}(I) \rangle.$$

Ou ainda, um conjunto $\mathcal{G} = \{g_1, \ldots, g_t\} \subset I$ *é dito ser uma base de Gröbner se e só se o termo líder de qualquer elemento de* I *for divisível por algum* $\mathrm{lt}(g_i)$*, com* $g_i \in \mathcal{G}$*.*

Uma base de Gröbner $\mathcal{G}$ de um ideal I é dita *mínima* se $\mathrm{lc}(g) = 1$ e $\mathrm{lt}(g) \notin \langle \mathrm{lt}(\mathcal{G} - \{g\}) \rangle$, para todo $g \in \mathcal{G}$. Uma base de Gröbner $\mathcal{G}$ de um ideal I é dita *reduzida* se for mínima e nenhum monômio de g pertencer a $\langle \mathrm{lt}(\mathcal{G} - \{g\}) \rangle$, para todo $g \in \mathcal{G}$.

Todo ideal $I \neq \{0\}$ possui uma base de Gröbner e, dada uma ordem de monômios, todo ideal $I \neq \{0\}$ possui uma única base de Gröbner reduzida. Além disso, o resto r da divisão de uma função $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ por uma base de Gröbner $\mathcal{G}$ de um ideal $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ é univocamente determinado, independentemente da ordem de monômios adotada.

Observe que o problema da pertinência de uma função f a um ideal I reduz-se à determinação de uma base de Gröbner $\mathcal{G}$ para I. Portanto, tem-se que $f \in I$ se e só se a divisão de f por $\mathcal{G}$ for exata (resto $r = 0$).

O processo de decodificação de códigos AG depende diretamente da divisão por polinômios em várias variáveis e reduz-se ao problema da busca por uma base de Gröbner para um ideal de polinômios localizadores de erros (polinômios que apresentam zeros nos pontos associados às posições onde ocorreram erros na comunicação). Daí a importância do estudo das bases de Gröbner e suas propriedades.

O algoritmo de Buchberger [9] é um exemplo de esquema utilizado na na determinação de uma base de Gröbner para um dado ideal $I = \langle f_1, \ldots, f_s \rangle$.

C.1.4 Anel de coordenadas

Considere um ideal $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ e faça $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$. Diz-se que f e g são *congruentes módulo* I, denotando-se por

$$f \equiv g \mod I,$$

se $f - g \in I$. Esta operação de congruência módulo I estabelece uma relação de equivalência, que subdivide $\mathbb{F}_q[x_1, \ldots, x_n]$ nas chamadas *classes de equivalência*. Para qualquer $f \in \mathbb{F}_q[x_1, \ldots, x_n]$, define-se a classe de f como o conjunto

$$[f] = \{g \in \mathbb{F}_q[x_1, \ldots, x_n] : g \equiv f \mod I\}.$$

Defina agora o quociente de $\mathbb{F}_q[x_1, \ldots, x_n]$ módulo I, denotado por $\mathbb{F}_q[x_1, \ldots, x_n]/I$, como o conjunto de classes de equivalência por congruência módulo I dado por

$$\mathbb{F}_q[x_1, \ldots, x_n]/I = \{[f] : f \in \mathbb{F}_q[x_1, \ldots, x_n]\}.$$

Um quociente $\mathbb{F}_q[x_1, \ldots, x_n]/I$ constitui, na verdade, um anel comutativo sobre as operações $[f] + [g] = [f + g]$ de adição e $[f].[g] = [f.g]$ de multiplicação realizadas entre as classes de equivalência, sendo, por isso, denominado *anel quociente*.

Definição 65 (Anel de coordenadas) *Considere uma variedade afim V. No caso em que $I = \mathbf{I}(V)$ (um ideal radical), o anel quociente dado por*

$$\Gamma(V) = \mathbb{F}_q[x_1, \ldots, x_n]/\mathbf{I}(V)$$

é denominado o anel de coordenadas *de V.*

C.1.5 Corpo de funções

A partir do anel de polinômios $\mathbb{F}_q[x_1, \ldots, x_n]$, pode-se construir um corpo com elementos da forma $\frac{f}{g}$, em que $f, g \in \mathbb{F}_q[x_1, \ldots, x_n]$, denominados *funções racionais*. O corpo denotado por $\mathbb{F}_q(x_1, \ldots, x_n)$ e dado por

$$\mathbb{F}_q(x_1, \ldots, x_n) = \left\{ \frac{f(x_1, \ldots, x_n)}{g(x_1, \ldots, x_n)} : f, g \in \mathbb{F}_q[x_1, \ldots, x_n],\ g \neq 0 \right\}$$

é denominado *corpo de funções*.

Definição 66 (Corpo de funções de uma variedade) *Se V é uma variedade afim e $\Gamma(V)$ seu anel de coordenadas, então o corpo das frações $\frac{f}{g}$, com $f, g \in \Gamma(V)$, é dito o* corpo de funções de V *e denotado por $\mathbb{F}_q(V)$.*

A *dimensão* de uma variedade afim V é o grau de transcendência do corpo $\mathbb{F}_q(V)$ sobre o corpo $\mathbb{F}_q$. Desta forma, define-se uma *curva algébrica* $\mathcal{X} \subseteq \mathbb{A}^n$ como uma variedade de dimensão 1.

Uma curva $\mathcal{X}$ é dita ser *não singular*, ou *regular*, se todos os seus pontos forem não singulares, ou seja, possuírem os mesmos zeros em duas derivadas parciais. Caso contrário, a curva é dita *singular*.

C.1.6 Variedade projetiva

Variedades afins são subconjuntos do espaço afim $\mathbb{A}^n$. A adição a $\mathbb{A}^n$ de "pontos no infinito" permite criar o chamado espaço projetivo $\mathbb{P}^n$ de dimensão n, do qual derivam as variedades projetivas.

Considere a relação de equivalência no conjunto $\mathbb{A}^{n+1} \setminus \{(0, 0, \ldots, 0)\}$, com termos em $\mathbb{F}_q$, dada por

$$(a_0, \ldots, a_n) \equiv (b_0, \ldots, b_n) \Leftrightarrow \exists\, \lambda \in \mathbb{F}_q \setminus \{0\} : b_i = \lambda a_i,\ i = 0, 1, \ldots, n.$$

A classe de equivalência de $(a_0, \ldots, a_n)$ é denotada por $(a_0 : \cdots : a_n)$.

Definição 67 (Espaço projetivo) *O* espaço projetivo $\mathbb{P}^n$ *de dimensão* n *é o conjunto de todas as classes de equivalência* $\{(a_0 : \ldots : a_n) : a_i \in \mathbb{F}_q\}$.

Um elemento $P = (a_0 : \cdots : a_n) \in \mathbb{P}^n$ é chamado *ponto*, e $a_0, \ldots, a_n$ são ditas as coordenadas homogêneas de P.

O conjunto $H = \{(0 : a_1 : \cdots : a_n) \in \mathbb{P}^n\}$ é chamado de *hiperplano no infinito* e os pontos de H, denotados por Q, são chamados *pontos no infinito*. O mapeamento $\varphi : \mathbb{A}^n \to \mathbb{P}^n \backslash H$ é definido por $\varphi(a_1, \ldots, a_n) = (1, a_1, \ldots, a_n)$.

Um polinômio constituído pela soma de monômios de mesmo grau é chamado *polinômio homogêneo*. Um polinômio homogêneo $f \in \mathbb{F}_q[x_0, \ldots, x_n]$ pode ser considerado como um mapeamento $f : \mathbb{P}^n \to \mathbb{F}_q$ dado por

$$f(P) = f(a_0, \ldots, a_n),$$

em que $P = (a_0 : \cdots : a_n) \in \mathbb{P}^n$, com $a_0, \ldots, a_n \in \mathbb{F}_q$. Se $f(P) = 0$, f é dito ter um *zero* no ponto $P = (a_0 : \cdots : a_n) \in \mathbb{P}^n$. Isto faz sentido, uma vez que, se f é homogêneo de grau d,

$$f(\lambda a_0, \cdots, \lambda a_n) = \lambda^d f(a_0, \cdots, a_n).$$

É possível sempre converter qualquer polinômio não homogêneo em um polinômio homogêneo. Para qualquer polinômio $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ de grau d, o polinômio $f' \in \mathbb{F}_q[x_0, \ldots, x_n]$ dado por

$$f'(x_0 : \cdots : x_n) = x_0^d f\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right)$$

é homogêneo de grau d.

Definição 68 (Variedade projetiva) *Considerando* $f_1, \ldots, f_s \in \mathbb{F}_q[x_0, \ldots, x_n]$ *polinômios homogêneos, a* variedade projetiva $\mathbf{V}(f_1, \ldots, f_s)$ *definida por* $f_1, \ldots, f_s$ *é dada por*

$$\mathbf{V}(f_1, \ldots, f_s) = \{P \in \mathbb{P}^n : f_i(P) = 0,\ 1 \leq i \leq s\}.$$

Observe que a soma de dois polinômios homogêneos de diferentes graus não preserva a homogeneidade. Portanto, um ideal $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}_q[x_0, \ldots, x_n]$ gerado por polinômios homogêneos sempre contém polinômios não homogêneos, que não podem ser usados na definição de uma variedade projetiva. Observe também que, apesar disto, todos os polinômios deste ideal I se anulam nos pontos da variedade projetiva $\mathbf{V}(f_1, \ldots, f_s)$. Além disso, mostra-se que, para todo $f \in I$, se f_i é uma componente (parte) homogênea de f, então $f_i \in I$. Estes ideais I gerados por polinômios homogêneos são chamados *ideais homogêneos*.

De forma similar ao caso afim, uma variedade projetiva V pode ser também definida como um subconjunto de $\mathbb{P}^n$, tal que $\mathbf{I}(V)$ é um ideal primo homogêneo.

Define-se, assim como no caso afim, o *anel de coordenadas homogêneo* de uma variedade projetiva $V \subset P^n$ como o anel quociente

$$\Gamma_h(V) = \mathbb{F}_q[x_0, \ldots, x_n] / \mathbf{I}(V).$$

Um elemento $f \in \Gamma_h(V)$ é dito ser uma *forma* de grau d se $f = F + \mathbf{I}(V)$, em que $F \in \mathbb{F}_q[x_0, \ldots, x_n]$ é um polinômio homogêneo de grau d.

O *corpo de funções* de uma variedade projetiva V é definido por

$$\mathbb{F}_q(V) = \left\{ \frac{f}{g} : f, g \in \Gamma_h(V) \text{ são formas de mesmo grau e } g \neq 0 \right\}.$$

A *dimensão* da variedade projetiva V é o grau de transcendência de $\mathbb{F}_q(V)$ sobre $\mathbb{F}_q$. Define-se uma *curva projetiva* $\mathcal{X} \subseteq \mathbb{P}^n$ como uma variedade projetiva de dimensão 1.

C.2 Anel local

Considere a variedade V e um ponto $P \in V$. Considere $f \in \mathbb{F}_q(V)$ uma função racional, então $f = \frac{g}{h}$, sendo $g, h \in \lneqq(V)$ para uma variedade V afim, ou $g, h \in \Gamma_h(V)$ para uma variedade V projetiva. Se $h(P) \neq 0$, então f é dito ser *definido* em P.

Definição 69 (Anel local) *O anel denotado por $O_P(V)$ e dado por*

$$O_P(V) = \{f \in \mathbb{F}_q(V) : f \text{ é definido em } P\}$$

é chamado o anel local *em P.*

Defina o *ideal máximo* $M_P(V)$ de um anel local $O_P(V)$ por

$$M_P(V) = \{f \in O_P(V) : f(P) = 0\}.$$

Definição 70 (Anel de valorização) *Chama-se* anel de valorização *de um corpo de funções $\mathbb{F}_q(V)$ um anel O, tal que $\mathbb{F}_q \subset O \subset \mathbb{F}_q(V)$ e, para todo $z \in \mathbb{F}_q(V)$, $z \in O$ ou $z^{-1} \in O$.*

Este anel de valorização O de $\mathbb{F}_q(V)$ é também um anel local, tendo como único ideal máximo $\mathcal{P} = O \backslash O^*$, em que $O^* = \{z \in O : \exists\, w \in O,\ zw = 1\}$. Este único ideal máximo

$\mathcal{P}$ de O é dito ser um *lugar*[1] de $\mathbb{F}_q(V)$. O conjunto dos lugares de um corpo de funções $\mathbb{F}_q(V)$ é denotado por

$$\mathbb{P} = \{\mathcal{P} : \mathcal{P} \text{ é um lugar de } \mathbb{F}_q(V)\}. \tag{C.3}$$

Observa-se que, se $\mathbb{F}_q$ é um corpo fechado algebricamente, que é o caso aqui considerado, então $\mathcal{P}$ possui *grau* 1 (veja definição de grau de um ponto fechado, ou lugar, na subseção seguinte). Se um lugar $\mathcal{P}$ possuir grau 1, então ele é equivalente a um ponto racional P.

Um anel de valorização O apresenta ainda as seguintes propriedades:

- Sendo $0 \neq z \in \mathbb{F}_q(V)$, então $z \in \mathcal{P} \Leftrightarrow z^{-1} \notin O$;
- $\mathcal{P}$ é um ideal principal;
- Se $\mathcal{P} = tO$, então qualquer $0 \neq z \in \mathbb{F}_q(V)$ possui uma representação única da forma $z = t^n u$, com $u \in O^*$ e $n \in \mathbb{Z}$;
- Se $\mathcal{P} = tO$ e $\{0\} \neq I \subseteq O$ é um ideal, então $I = t^n O$ para algum $n \in \mathbb{N}$.

A função t é chamada *parâmetro local*, *parâmetro de uniformização* ou *elemento primo* de $\mathcal{P}$. Um anel de valorização O, cujo respectivo lugar $\mathcal{P} = tO$, é chamado *anel de valorização discreto*.

Considere O um anel de valorização discreto de $\mathbb{F}_q(V)$ e $\mathcal{P}$ seu único ideal máximo. Tem-se que, para $z \in \mathbb{F}_q(V)$, $z = t^n u$, com $u \in O^*$ e $n \in \mathbb{Z}$. Defina, então, a função $v : \mathbb{F}_q(V) \to \mathbb{Z}$, chamada *função de valorização discreta* de $\mathbb{F}_q(V)$, por

$$v_{\mathcal{P}}(z) = \begin{cases} \infty, & z = 0; \\ n, & z \neq 0. \end{cases} \tag{C.4}$$

Esta função apresenta as seguintes propriedades:

- $v_{\mathcal{P}}(x) = \infty \Leftrightarrow x = 0$;
- $v_{\mathcal{P}}(xy) = v_{\mathcal{P}}(x) + v_{\mathcal{P}}(y)$, para qualquer $x, y \in \mathbb{F}_q(V)$;
- $v_{\mathcal{P}}(x+y) \geq \min[v_{\mathcal{P}}(x), v_{\mathcal{P}}(y)]$, com igualdade se $v_{\mathcal{P}}(x) \neq v_{\mathcal{P}}(y)$;
- Existe um elemento $z \in \mathbb{F}_q(V)$, tal que $v_{\mathcal{P}}(z) = 1$;
- $v_{\mathcal{P}}(a) = 0$, para todo $0 \neq a \in \mathbb{F}_q$.

[1] O conceito de lugar é freqüentemente associado a ou confundido com o conceito de ponto. Observe uma analogia com números reais e complexos. Um número $a \in \mathbb{R}$ constitui um ponto no eixo horizontal do plano $\mathbb{R}^2$. Pode-se dizer que o conjunto dos números complexos $a + bi$, com $b \in \mathbb{R}$, constitui um lugar. Um lugar costuma ser vulgarmente referido como um "ponto gordo".

Sendo $\mathcal{X}$ uma curva (afim ou projetiva) e P um ponto de $\mathcal{X}$, o ponto P é não singular se e só se $O_P(\mathcal{X})$ for um anel de valorização discreto.

Denota-se por $\mathcal{P}_\mathcal{X}$ o conjunto dos pontos fechados de uma curva $\mathcal{X}$.

C.3 Divisores

O conceito de divisor constitui uma forma extremamente elegante de manipular um conjunto de pontos relacionados a uma curva.

Considere $\mathcal{X}$ uma curva projetiva definida sobre $\mathbb{F}_q$.

Definição 71 (Divisor) *Um* divisor D *de* $\mathcal{X}$ *é uma soma formal (um conjunto de pontos dispostos em forma de uma soma ponderada) dada por*

$$D = \sum_{P \in \mathcal{X}} n_P P,$$

em que $n_P \in \mathbb{Z}$*, podendo ser* $n_P = 0$ *para alguns, mas não todos, dos pontos* $P \in \mathcal{X}$*.*

O *suporte* de D é definido por

$$\sup D = \{P \in \mathcal{X} : n_P \neq 0\}.$$

Um divisor D é dito ser *efetivo*, denotando-se por $D \succ 0$, se todo n_P for não negativo.

Os divisores de uma curva $\mathcal{X}$ formam um grupo abeliano, denotado por $Div(\mathcal{X})$, chamado *grupo dos divisores* de $\mathcal{X}$. O *grau* de um divisor D é dado por

$$\deg D = \sum_{P \in \mathcal{X}} n_P,$$

o que constitui um mapeamento $\deg : Div(\mathcal{X}) \to \mathbb{Z}$.

Considere uma função racional $f \in \mathbb{F}_q(\mathcal{X})$. A *ordem* de uma função f em um ponto racional $P \in \mathcal{X}$ (um ponto racional P equivale a um lugar de grau 1) é definida como $v_P(f)$, em que v_P é a função de valorização discreta (*cf.* (C.4)) correspondente ao anel de valorização discreto O do corpo de funções $\mathbb{F}_q(\mathcal{X})$. Se $v_P(f) > 0$, diz-se que f possui um *zero* no ponto P, e se $v_P(f) < 0$, diz-se que f possui um *pólo* em P.

Defina agora o *divisor principal* (f) da função racional $f \in \mathbb{F}_q(\mathcal{X})$ por

$$(f) = \sum_{P \in \mathcal{X}} v_P(f) P, \tag{C.5}$$

o *divisor de zeros* $(f)_0$ de f por

$$(f)_0 = \sum_{v_P(f) > 0} v_P(f) P$$

e o *divisor de pólos* $(f)_{\in fty}$ de f por

$$(f)_{\in fty} = -\sum_{v_P(f)<0} v_P(f)\, P.$$

O grau de um divisor principal é, por definição, nulo, o que implica que

$$\sum_{v_P(f)>0} v_P(f) = -\sum_{v_P(f)<0} v_P(f).$$

Dado um grupo de divisores $Div(\mathcal{X})$, define-se uma ordem entre os elementos deste grupo da forma

$$D_1 = \sum_{P\in\mathcal{P}_\mathcal{X}} n_P P \leq D_2 = \sum_{P\in\mathcal{P}_\mathcal{X}} n'_P P$$

se e só se

$$n_P \leq n'_P,$$

para todo $P \in \mathcal{X}$.

Definição 72 (Espaço de funções de um divisor) *Considere $G \in Div(\mathcal{X})$ um divisor de uma curva $\mathcal{X}$, então*

$$L(G) = \{f \in \mathbb{F}_q(\mathcal{X}) : (f) + G \succ 0\} \cup \{0\}$$

é o espaço vetorial das funções racionais com pólos determinados pelos pontos do divisor G, com multiplicidades determinadas pelas ordens destes pontos.

Observe que o divisor do produto de duas funções $f, h \in \mathbb{F}_q(\mathcal{X})$ é a soma dos respectivos divisores, ou seja, $(fh) = (f) + (h)$. Ocorre também que o divisor da soma destas funções satisfaz a desigualdade $(f+h) \geq \min\{(f),(h)\}$, em que $\min\{(f),(h)\}$ é o divisor formado com os menores coeficientes v_P (*cf.* (C.5)) ponto a ponto.

O espaço vetorial $L(G)$ definido sobre $\mathbb{F}_q$ possui dimensão finita, denotada por $l(G)$. Esta dimensão é determinada pelo teorema de Riemann-Roch, que é tratado na seção seguinte.

C.4 Teorema de Riemann-Roch

De modo a determinar a dimensão $l(G)$ do espaço vetorial $L(G)$, torna-se necessário o uso e entendimento do conceito de *diferencial*.

Na matemática clássica, uma integral

$$\int_C f dx$$

consiste num operador que fornece um número a partir de três entradas:

1. Um caminho C, que costuma ser fechado e simples;

2. Uma função f sem pólos em C;

3. Um diferencial dx.

O diferencial é o mais obscuro dos três conceitos, uma vez que suas propriedades permanecem normalmente implícitas. Observe que se pode ter um diferencial

$$dy = \frac{dy}{dx}dx = fdx.$$

Sendo $f = \frac{dy}{dx}$ uma função, tem-se que diferenciais na matemática clássica compõem um espaço vetorial unidimensional sobre um espaço de funções.

No caso do presente estudo, pode-se imaginar diferenciais como objetos da forma fdh, em que $f, h \in \mathbb{F}_q(\mathcal{X})$ são funções racionais associadas à curva $\mathcal{X}$ e o mapeamento linear que leva de h para dh é denominado *derivação*[2]. Para esta derivação vale a conhecida regra

$$d(h_1h_2) = h_1dh_2 + h_2dh_1.$$

Para cada ponto fechado $(O, \mathcal{P})$ de $\mathcal{X}$, existe um parâmetro local t, em que $v_{\mathcal{P}}(t) = 1$ (*cf.* (C.4)). Também, para cada diferencial denotado por ω, existe uma função f, tal que $\omega = fdt$. Tem-se, então, que a função de valorização discreta $v_{\mathcal{P}}(\omega)$ é, por definição, igual a $v_{\mathcal{P}}(f)$.

Assim como no caso das funções racionais, pode-se falar em pólos e zeros de diferenciais. Diz-se que ω possui um *zero* de ordem ρ, se $\rho = v_{\mathcal{P}}(f) > 0$, e que ω possui um *pólo* de ordem ρ, se $\rho = -v_{\mathcal{P}}(f) > 0$.

Define-se o *divisor* de um diferencial ω por

$$(\omega) = \sum v_{\mathcal{P}}(f)\,\mathcal{P}.$$

O divisor de um diferencial é dito ser *canônico* e possui sempre grau $2g - 2$, em que g é o gênero da curva $\mathcal{X}$ definido em seguida na descrição do teorema de Riemann.

Denote por $\Omega_{\mathcal{X}}$ o conjunto dos diferenciais associados à curva $\mathcal{X}$. Assim como foi definido o espaço de funções $L(G)$ de um divisor G, define-se também o *espaço de diferenciais* denotado por $\Omega(G)$ e dado por

$$\Omega(G) = \{\omega \in \Omega_{\mathcal{X}} : (\omega) - G \succ 2\} \cup \{0\}.$$

A dimensão do espaço $\Omega(G)$ é chamada de *índice de especialidade* de G e denotada por $i(G)$.

[2]Será omitida aqui uma definição formal ou mais adequada deste mapeamento, o que demandaria uma longa discussão envolvendo outros conceitos de menor relevância para o presente trabalho.

Como afirmado acima, se $(O, \mathcal{P})$ é um ponto fechado de $\mathcal{X}$ de grau d e t é um parâmetro local neste ponto, então existe uma função racional f, tal que $\omega = fdt$. Se uma função f apresenta uma expansão em série de Laurent $\sum_{i=\rho}^{\in fty} a_i t^i$, em que $a_i \in \mathbb{F}_q$, $\rho = f_{\mathcal{P}}(\omega)$ e $a_\rho \neq 0$, define-se o *resíduo de f com relação a $\mathcal{P}$ e t* por

$$\mathrm{Res}_{\mathcal{P},t}(f) = a_{-1}.$$

Observe que, se $v_{\mathcal{P}}(f) \geq 0$, então $\mathrm{Res}_{\mathcal{P},t}(f) = 0$. O *resíduo de ω em $\mathcal{P}$*, denotado por $\mathrm{Res}_{\mathcal{P}}(\omega)$, é definido, então, por

$$\mathrm{Res}_{\mathcal{P}}(\omega) = \mathrm{Res}_{\mathcal{P},t}(f).$$

Este resultado é independente da escolha do parâmetro local t.

Dado um diferencial $\omega \in \Omega_{\mathcal{X}}$, o *teorema do resíduo* estabelece que

$$\sum_{\mathcal{P} \in \mathcal{P}_{\mathcal{X}}} \mathrm{Res}_{\mathcal{P}}(\omega) = 0. \tag{C.6}$$

Definido o espaço vetorial de diferenciais, pode-se, então, retornar à análise da dimensão $l(G)$ do espaço de funções $L(G)$.

O chamado *teorema de Riemann* afirma que existe um inteiro não negativo r, tal que, para todo divisor G de uma curva $\mathcal{X}$,

$$l(G) \geq \deg(G) + 1 - r,$$

sendo o valor mínimo de r denominado *gênero* de $\mathcal{X}$ e denotado por g. Se $\mathcal{X}$ é uma curva não singular (todos os seus pontos são não singulares) e m é seu grau, então seu gênero é dado por

$$g = \frac{(m-1)(m-2)}{2}. \tag{C.7}$$

O problema da determinação da dimensão $l(G)$ é resolvido pelo teorema que segue.

Teorema 73 (Riemann-Roch) *Para um divisor G de uma curva de gênero g, tem-se que*

$$l(G) = \deg(G) + 1 - g + i(G).$$

Além disso, o índice de especialidade $i(G)$ (dimensão do espaço $\Omega(G)$) é dado por

$$i(G) = l(K - G)$$

para todos os divisores G e divisores canônicos K.

Uma conseqüência do teorema de Riemann-Roch é que, para qualquer divisor N, com $\deg(G) > 2g - 2$, tem-se que

$$l(G) = \deg(G) + 1 - g. \tag{C.8}$$

C.5 Lacunas e anti-lacunas

Considere um ponto racional Q de uma curva algébrica $\mathcal{X}$ de gênero g. Sendo m um inteiro não negativo, considere divisores do tipo mQ (divisores de um único ponto).

Definição 74 (Lacuna) *Um inteiro não negativo m é dito ser uma* lacuna *("gap") de um ponto Q de uma curva $\mathcal{X}$, se $l(mQ) = l((m-1)Q)$.*

Segundo o teorema de Riemann-Roch, para $m > 2g-2$, tem-se que $l(mQ) = m+1-g$. Portanto, valores de $m > 2g-1$ não constituem lacunas. Para valores de m até $2g-1$, observe que

$$1 = l(0) \leq l(Q) \leq \cdots \leq l((2m-1)Q) = g.$$

Ou seja, existem g valores diferentes de $l(iQ)$, para $0 \leq i \leq 2g-1$. Conclui-se que o número total de lacunas de gm ponto Q de uma curva $\mathcal{X}$ é igual ao gênero g de $\mathcal{X}$.

Definição 75 (Anti-lacuna) *Um inteiro positivo m é dito ser uma* anti-lacuna *("non-gap") de um ponto Q de uma curva $\mathcal{X}$, se $l(mQ) \neq l((m-1)Q)$, ou seja, se e só se existir uma função racional $f \in L(mQ)$, tal que $v_Q(f) = -m$ (possui pólos de ordem m em Q).*

Se m_i é uma anti-lacuna e $m_{i-1} < m_i$, então

$$0 = m_0 < m_1 < \cdots < m_{g-1} < m_g = 2g$$

e $m_i = i + g$, para $i \geq g$. Também, se m_1 e m_2 são anti-lacunas de Q, então $m_1 + m_2$ também é uma anti-lacuna de Q. Portanto, as anti-lacunas de um ponto Q formam um semi-grupo na adição.

Apêndice D

Polinômios Primitivos em Corpos Finitos

Seguem como referência alguns polinômios primitivos de peso mínimo em $\mathbb{F}_2$ que podem ser utilizados na geração de corpos finitos de característica 2. Eles são úteis na escolha dos parâmetros nas unidades aritméticas implementadas neste trabalho.

$\mathbb{F}_{2^2} \rightarrow x^2 + x + 1$
$\mathbb{F}_{2^3} \rightarrow x^3 + x + 1$
$\mathbb{F}_{2^4} \rightarrow x^4 + x + 1$
$\mathbb{F}_{2^5} \rightarrow x^5 + x^2 + 1$
$\mathbb{F}_{2^6} \rightarrow x^6 + x + 1$
$\mathbb{F}_{2^7} \rightarrow x^7 + x + 1$
$\mathbb{F}_{2^8} \rightarrow x^8 + x^4 + x^3 + x^2 + 1$
$\mathbb{F}_{2^9} \rightarrow x^9 + x^4 + 1$
$\mathbb{F}_{2^{10}} \rightarrow x^{10} + x^3 + 1$
$\mathbb{F}_{2^{11}} \rightarrow x^{11} + x^2 + 1$
$\mathbb{F}_{2^{12}} \rightarrow x^{12} + x^6 + x^4 + x + 1$
$\mathbb{F}_{2^{13}} \rightarrow x^{13} + x^4 + x^3 + x + 1$
$\mathbb{F}_{2^{14}} \rightarrow x^{14} + x^5 + x^3 + x + 1$
$\mathbb{F}_{2^{15}} \rightarrow x^{15} + x + 1$
$\mathbb{F}_{2^{16}} \rightarrow x^{16} + x^5 + x^3 + x^2 + 1$
$\mathbb{F}_{2^{17}} \rightarrow x^{17} + x^3 + 1$
$\mathbb{F}_{2^{18}} \rightarrow x^{18} + x^7 + 1$
$\mathbb{F}_{2^{19}} \rightarrow x^{19} + x^5 + x^2 + x + 1$
$\mathbb{F}_{2^{20}} \rightarrow x^{20} + x^3 + 1$
$\mathbb{F}_{2^{21}} \rightarrow x^{21} + x^2 + 1$
$\mathbb{F}_{2^{22}} \rightarrow x^{22} + x + 1$
$\mathbb{F}_{2^{23}} \rightarrow x^{23} + x^5 + 1$
$\mathbb{F}_{2^{24}} \rightarrow x^{24} + x^4 + x^3 + x + 1$
$\mathbb{F}_{2^{25}} \rightarrow x^{25} + x^3 + 1$
$\mathbb{F}_{2^{26}} \rightarrow x^{26} + x^6 + x^2 + x + 1$
$\mathbb{F}_{2^{27}} \rightarrow x^{27} + x^5 + x^2 + x + 1$
$\mathbb{F}_{2^{28}} \rightarrow x^{28} + x^3 + 1$
$\mathbb{F}_{2^{29}} \rightarrow x^{29} + x^2 + 1$
$\mathbb{F}_{2^{30}} \rightarrow x^{30} + x^6 + x^4 + x + 1$
$\mathbb{F}_{2^{31}} \rightarrow x^{31} + x^3 + 1$
$\mathbb{F}_{2^{32}} \rightarrow x^{32} + x^7 + x^6 + x^2 + 1$
$\mathbb{F}_{2^{33}} \rightarrow x^{33} + x^{13} + 1$
$\mathbb{F}_{2^{34}} \rightarrow x^{34} + x^8 + x^4 + x^3 + 1$
$\mathbb{F}_{2^{35}} \rightarrow x^{35} + x^2 + 1$
$\mathbb{F}_{2^{36}} \rightarrow x^{36} + x^{11} + 1$
$\mathbb{F}_{2^{37}} \rightarrow x^{37} + x^6 + x^4 + x + 1$
$\mathbb{F}_{2^{38}} \rightarrow x^{38} + x^6 + x^5 + x + 1$
$\mathbb{F}_{2^{39}} \rightarrow x^{39} + x^4 + 1$
$\mathbb{F}_{2^{40}} \rightarrow x^{40} + x^5 + x^4 + x^3 + 1$
$\mathbb{F}_{2^{41}} \rightarrow x^{41} + x^3 + 1$
$\mathbb{F}_{2^{42}} \rightarrow x^{42} + x^7 + x^4 + x^3 + 1$
$\mathbb{F}_{2^{43}} \rightarrow x^{43} + x^6 + x^4 + x^3 + 1$
$\mathbb{F}_{2^{44}} \rightarrow x^{44} + x^6 + x^5 + x^2 + 1$
$\mathbb{F}_{2^{45}} \rightarrow x^{45} + x^4 + x^3 + x + 1$
$\mathbb{F}_{2^{46}} \rightarrow x^{46} + x^8 + x^7 + x^6 + 1$
$\mathbb{F}_{2^{47}} \rightarrow x^{47} + x^5 + 1$
$\mathbb{F}_{2^{48}} \rightarrow x^{48} + x^9 + x^7 + x^4 + 1$
$\mathbb{F}_{2^{49}} \rightarrow x^{49} + x^9 + 1$
$\mathbb{F}_{2^{50}} \rightarrow x^{50} + x^4 + x^3 + x^2 + 1$
$\mathbb{F}_{2^{51}} \rightarrow x^{51} + x^6 + x^3 + x + 1$
$\mathbb{F}_{2^{52}} \rightarrow x^{52} + x^3 + 1$
$\mathbb{F}_{2^{53}} \rightarrow x^{53} + x^6 + x^2 + x + 1$
$\mathbb{F}_{2^{54}} \rightarrow x^{54} + x^8 + x^6 + x^3 + 1$
$\mathbb{F}_{2^{55}} \rightarrow x^{55} + x^{24} + 1$
$\mathbb{F}_{2^{56}} \rightarrow x^{56} + x^7 + x^4 + x^2 + 1$
$\mathbb{F}_{2^{57}} \rightarrow x^{57} + x^7 + 1$
$\mathbb{F}_{2^{58}} \rightarrow x^{58} + x^{19} + 1$
$\mathbb{F}_{2^{59}} \rightarrow x^{59} + x^7 + x^4 + x^2 + 1$
$\mathbb{F}_{2^{60}} \rightarrow x^{60} + x + 1$
$\mathbb{F}_{2^{61}} \rightarrow x^{61} + x^5 + x^2 + x + 1$
$\mathbb{F}_{2^{62}} \rightarrow x^{62} + x^6 + x^5 + x^3 + 1$
$\mathbb{F}_{2^{63}} \rightarrow x^{63} + x + 1$
$\mathbb{F}_{2^{64}} \rightarrow x^{64} + x^4 + x^3 + x + 1$

$\mathbb{F}_{2^{65}} \rightarrow x^{65} + x^{18} + 1$
$\mathbb{F}_{2^{66}} \rightarrow x^{66} + x^{9} + x^{8} + x^{6} + 1$
$\mathbb{F}_{2^{67}} \rightarrow x^{67} + x^{5} + x^{2} + x + 1$
$\mathbb{F}_{2^{68}} \rightarrow x^{68} + x^{9} + 1$
$\mathbb{F}_{2^{69}} \rightarrow x^{69} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{70}} \rightarrow x^{70} + x^{5} + x^{3} + x + 1$
$\mathbb{F}_{2^{71}} \rightarrow x^{71} + x^{6} + 1$
$\mathbb{F}_{2^{72}} \rightarrow x^{72} + x^{10} + x^{9} + x^{3} + 1$
$\mathbb{F}_{2^{73}} \rightarrow x^{73} + x^{25} + 1$
$\mathbb{F}_{2^{74}} \rightarrow x^{74} + x^{7} + x^{4} + x^{3} + 1$
$\mathbb{F}_{2^{75}} \rightarrow x^{75} + x^{6} + x^{3} + x + 1$
$\mathbb{F}_{2^{76}} \rightarrow x^{76} + x^{5} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{77}} \rightarrow x^{77} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{78}} \rightarrow x^{78} + x^{7} + x^{2} + x + 1$
$\mathbb{F}_{2^{79}} \rightarrow x^{79} + x^{9} + 1$
$\mathbb{F}_{2^{80}} \rightarrow x^{80} + x^{9} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{81}} \rightarrow x^{81} + x^{4} + 1$
$\mathbb{F}_{2^{82}} \rightarrow x^{82} + x^{9} + x^{6} + x^{4} + 1$
$\mathbb{F}_{2^{83}} \rightarrow x^{83} + x^{7} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{84}} \rightarrow x^{84} + x^{13} + 1$
$\mathbb{F}_{2^{85}} \rightarrow x^{85} + x^{8} + x^{2} + x + 1$
$\mathbb{F}_{2^{86}} \rightarrow x^{86} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{87}} \rightarrow x^{87} + x^{13} + 1$
$\mathbb{F}_{2^{88}} \rightarrow x^{88} + x^{11} + x^{9} + x^{8} + 1$
$\mathbb{F}_{2^{89}} \rightarrow x^{89} + x^{38} + 1$
$\mathbb{F}_{2^{90}} \rightarrow x^{90} + x^{5} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{91}} \rightarrow x^{91} + x^{8} + x^{5} + x + 1$
$\mathbb{F}_{2^{92}} \rightarrow x^{92} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{93}} \rightarrow x^{93} + x^{2} + 1$
$\mathbb{F}_{2^{94}} \rightarrow x^{94} + x^{21} + 1$
$\mathbb{F}_{2^{95}} \rightarrow x^{95} + x^{11} + 1$
$\mathbb{F}_{2^{96}} \rightarrow x^{96} + x^{10} + x^{9} + x^{6} + 1$
$\mathbb{F}_{2^{97}} \rightarrow x^{97} + x^{6} + 1$
$\mathbb{F}_{2^{98}} \rightarrow x^{98} + x^{11} + 1$
$\mathbb{F}_{2^{99}} \rightarrow x^{99} + x^{7} + x^{5} + x^{4} + 1$
$\mathbb{F}_{2^{100}} \rightarrow x^{100} + x^{37} + 1$
$\mathbb{F}_{2^{101}} \rightarrow x^{101} + x^{7} + x^{6} + x + 1$
$\mathbb{F}_{2^{102}} \rightarrow x^{102} + x^{6} + x^{5} + x^{3} + 1$
$\mathbb{F}_{2^{103}} \rightarrow x^{103} + x^{9} + 1$
$\mathbb{F}_{2^{104}} \rightarrow x^{104} + x^{11} + x^{10} + x + 1$
$\mathbb{F}_{2^{105}} \rightarrow x^{105} + x^{16} + 1$
$\mathbb{F}_{2^{106}} \rightarrow x^{106} + x^{15} + 1$
$\mathbb{F}_{2^{107}} \rightarrow x^{107} + x^{9} + x^{7} + x^{4} + 1$
$\mathbb{F}_{2^{108}} \rightarrow x^{108} + x^{31} + 1$
$\mathbb{F}_{2^{109}} \rightarrow x^{109} + x^{5} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{110}} \rightarrow x^{110} + x^{6} + x^{4} + x + 1$
$\mathbb{F}_{2^{111}} \rightarrow x^{111} + x^{10} + 1$
$\mathbb{F}_{2^{112}} \rightarrow x^{112} + x^{11} + x^{6} + x^{4} + 1$
$\mathbb{F}_{2^{113}} \rightarrow x^{113} + x^{9} + 1$
$\mathbb{F}_{2^{114}} \rightarrow x^{114} + x^{11} + x^{2} + x + 1$
$\mathbb{F}_{2^{115}} \rightarrow x^{115} + x^{8} + x^{7} + x^{5} + 1$
$\mathbb{F}_{2^{116}} \rightarrow x^{116} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{117}} \rightarrow x^{117} + x^{5} + x^{2} + x + 1$
$\mathbb{F}_{2^{118}} \rightarrow x^{118} + x^{33} + 1$
$\mathbb{F}_{2^{119}} \rightarrow x^{119} + x^{8} + 1$
$\mathbb{F}_{2^{120}} \rightarrow x^{120} + x^{9} + x^{6} + x^{2} + 1$
$\mathbb{F}_{2^{121}} \rightarrow x^{121} + x^{18} + 1$
$\mathbb{F}_{2^{122}} \rightarrow x^{122} + x^{6} + x^{2} + x + 1$
$\mathbb{F}_{2^{123}} \rightarrow x^{123} + x^{2} + 1$
$\mathbb{F}_{2^{124}} \rightarrow x^{124} + x^{37} + 1$
$\mathbb{F}_{2^{125}} \rightarrow x^{125} + x^{7} + x^{6} + x^{5} + 1$
$\mathbb{F}_{2^{126}} \rightarrow x^{126} + x^{7} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{127}} \rightarrow x^{127} + x + 1$
$\mathbb{F}_{2^{128}} \rightarrow x^{128} + x^{7} + x^{2} + x + 1$
$\mathbb{F}_{2^{129}} \rightarrow x^{129} + x^{5} + 1$
$\mathbb{F}_{2^{130}} \rightarrow x^{130} + x^{3} + 1$
$\mathbb{F}_{2^{131}} \rightarrow x^{131} + x^{8} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{132}} \rightarrow x^{132} + x^{29} + 1$
$\mathbb{F}_{2^{133}} \rightarrow x^{133} + x^{9} + x^{8} + x^{2} + 1$
$\mathbb{F}_{2^{134}} \rightarrow x^{134} + x^{57} + 1$
$\mathbb{F}_{2^{135}} \rightarrow x^{135} + x^{11} + 1$
$\mathbb{F}_{2^{136}} \rightarrow x^{136} + x^{8} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{137}} \rightarrow x^{137} + x^{21} + 1$
$\mathbb{F}_{2^{138}} \rightarrow x^{138} + x^{8} + x^{7} + x + 1$
$\mathbb{F}_{2^{139}} \rightarrow x^{139} + x^{8} + x^{5} + x^{3} + 1$
$\mathbb{F}_{2^{140}} \rightarrow x^{140} + x^{29} + 1$
$\mathbb{F}_{2^{141}} \rightarrow x^{141} + x^{13} + x^{6} + x + 1$
$\mathbb{F}_{2^{142}} \rightarrow x^{142} + x^{21} + 1$
$\mathbb{F}_{2^{143}} \rightarrow x^{143} + x^{5} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{144}} \rightarrow x^{144} + x^{7} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{145}} \rightarrow x^{145} + x^{52} + 1$
$\mathbb{F}_{2^{146}} \rightarrow x^{146} + x^{5} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{147}} \rightarrow x^{147} + x^{11} + x^{4} + x^{2} + 1$
$\mathbb{F}_{2^{148}} \rightarrow x^{148} + x^{27} + 1$
$\mathbb{F}_{2^{149}} \rightarrow x^{149} + x^{10} + x^{9} + x^{7} + 1$
$\mathbb{F}_{2^{150}} \rightarrow x^{150} + x^{53} + 1$
$\mathbb{F}_{2^{151}} \rightarrow x^{151} + x^{3} + 1$
$\mathbb{F}_{2^{152}} \rightarrow x^{152} + x^{6} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{153}} \rightarrow x^{153} + x + 1$
$\mathbb{F}_{2^{154}} \rightarrow x^{154} + x^{9} + x^{5} + x + 1$
$\mathbb{F}_{2^{155}} \rightarrow x^{155} + x^{7} + x^{5} + x^{4} + 1$
$\mathbb{F}_{2^{156}} \rightarrow x^{156} + x^{9} + x^{5} + x^{3} + 1$
$\mathbb{F}_{2^{157}} \rightarrow x^{157} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{158}} \rightarrow x^{158} + x^{8} + x^{6} + x^{5} + 1$
$\mathbb{F}_{2^{159}} \rightarrow x^{159} + x^{31} + 1$
$\mathbb{F}_{2^{160}} \rightarrow x^{160} + x^{5} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{161}} \rightarrow x^{161} + x^{18} + 1$
$\mathbb{F}_{2^{162}} \rightarrow x^{162} + x^{8} + x^{7} + x^{4} + 1$
$\mathbb{F}_{2^{163}} \rightarrow x^{163} + x^{7} + x^{6} + x^{3} + 1$
$\mathbb{F}_{2^{164}} \rightarrow x^{164} + x^{12} + x^{6} + x^{5} + 1$
$\mathbb{F}_{2^{165}} \rightarrow x^{165} + x^{9} + x^{8} + x^{3} + 1$
$\mathbb{F}_{2^{166}} \rightarrow x^{166} + x^{10} + x^{3} + x^{2} + 1$
$\mathbb{F}_{2^{167}} \rightarrow x^{167} + x^{6} + 1$
$\mathbb{F}_{2^{168}} \rightarrow x^{168} + x^{16} + x^{9} + x^{6} + 1$
$\mathbb{F}_{2^{169}} \rightarrow x^{169} + x^{34} + 1$
$\mathbb{F}_{2^{170}} \rightarrow x^{170} + x^{23} + 1$
$\mathbb{F}_{2^{171}} \rightarrow x^{171} + x^{6} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{172}} \rightarrow x^{172} + x^{7} + 1$
$\mathbb{F}_{2^{173}} \rightarrow x^{173} + x^{8} + x^{5} + x^{2} + 1$
$\mathbb{F}_{2^{174}} \rightarrow x^{174} + x^{13} + 1$
$\mathbb{F}_{2^{175}} \rightarrow x^{175} + x^{6} + 1$
$\mathbb{F}_{2^{176}} \rightarrow x^{176} + x^{12} + x^{11} + x^{9} + 1$
$\mathbb{F}_{2^{177}} \rightarrow x^{177} + x^{8} + 1$
$\mathbb{F}_{2^{178}} \rightarrow x^{178} + x^{87} + 1$
$\mathbb{F}_{2^{179}} \rightarrow x^{179} + x^{4} + x^{2} + x + 1$
$\mathbb{F}_{2^{180}} \rightarrow x^{180} + x^{12} + x^{10} + x^{7} + 1$
$\mathbb{F}_{2^{181}} \rightarrow x^{181} + x^{7} + x^{6} + x + 1$

$\mathbb{F}_{2^{182}} \to x^{182}+x^8+x^6+x+1$
$\mathbb{F}_{2^{183}} \to x^{183}+x^{56}+1$
$\mathbb{F}_{2^{184}} \to x^{184}+x^9+x^8+x^7+1$
$\mathbb{F}_{2^{185}} \to x^{185}+x^{24}+1$
$\mathbb{F}_{2^{186}} \to x^{186}+x^9+x^8+x^6+1$
$\mathbb{F}_{2^{187}} \to x^{187}+x^7+x^6+x^5+1$
$\mathbb{F}_{2^{188}} \to x^{188}+x^6+x^5+x^2+1$
$\mathbb{F}_{2^{189}} \to x^{189}+x^6+x^5+x^2+1$
$\mathbb{F}_{2^{190}} \to x^{190}+x^{13}+x^6+x^2+1$
$\mathbb{F}_{2^{191}} \to x^{191}+x^9+1$
$\mathbb{F}_{2^{192}} \to x^{192}+x^{15}+x^{11}+x^5+1$
$\mathbb{F}_{2^{193}} \to x^{193}+x^{15}+1$
$\mathbb{F}_{2^{194}} \to x^{194}+x^{87}+1$
$\mathbb{F}_{2^{195}} \to x^{195}+x^8+x^3+x^2+1$
$\mathbb{F}_{2^{196}} \to x^{196}+x^{11}+x^9+x^2+1$
$\mathbb{F}_{2^{197}} \to x^{197}+x^9+x^4+x^2+1$
$\mathbb{F}_{2^{198}} \to x^{198}+x^{65}+1$
$\mathbb{F}_{2^{199}} \to x^{199}+x^{34}+1$
$\mathbb{F}_{2^{200}} \to x^{200}+x^5+x^3+x^2+1$
$\mathbb{F}_{2^{201}} \to x^{201}+x^{14}+1$
$\mathbb{F}_{2^{202}} \to x^{202}+x^{55}+1$

Referências Bibliográficas

[1] David W. Ash, Ian F. Blake, e Scott A. Vanstone. Low Complexity Normal Bases. In *Discrete Applied Mathematics*, volume 25, pp. 191–210. North-Holland: Elsevier Science Publishers, 1989.

[2] J. B. Ashbrook, N. R. Shanbhag, Ralf Kötter, e Richard E. Blahut. Implementation of a Hermitian decoder IC in 0.35μm CMOS. In *IEEE Custom Integrated Circuits Conference Proceedings*, pp. 297–300, 2001.

[3] Zouhair M. Belkoura. On Hardware Implementation of Decoding for Error Correcting Codes Based on Algebraic Geometry. Final year project, Ecole Nationale Supérieure des Télécommunications, 2002.

[4] Zouhair M. Belkoura e Lírida Alves de Barros Naviner. Hardware implementation issues of a BMS decoding approach for AG based codes. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, pp. 448–453, 2003.

[5] Elwyn R. Berlekamp. Bounded Distance + 1 Soft-Decision Reed-Solomon Decoding. *IEEE Transactions on Information Theory*, 42(3):704–720, 1996.

[6] Richard E. Blahut. *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

[7] Richard E. Blahut. Encoding of Codes on Curves. In *IEEE International Symposium on Information Theory*, 1994.

[8] Ian Blake, Chris Heegard, Tom Høholdt, e Victor Wei. Algebraic-Geometry Codes. *IEEE Transactions on Information Theory*, 44(6):2596–2618, October 1998.

[9] David Cox, John Little, e Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. New York: Springer-Verlag, 1992.

[10] David A. Cox e Bernd Sturmfels, editors. *Applications of Computational Algebraic Geometry*. American Mathematical Society, 1997.

[11] D. Cunsheng, H. Niederreiter, e X. Chaoping. Some New Codes from Algebraic Curves. *IEEE Transactions on Information Theory*, 46:2638–2642, 2000.

[12] Leocarlos Bezerra da Silva Lima. Análise dos Algoritmos de Decodificação para Códigos de Geometria Algébrica Sobre Curvas de Hermite. Dissertação de Mestrado, Universidade Federal da Paraíba – UFPB, August 1999.

[13] Leocarlos Bezerra da Silva Lima e Francisco Marcos de Assis. Decoding algorithm for Reed-Solomon codes using the method of Gröebner basis. In *Proc. 2nd Conference on Telecommunications – ConfTele99*, pp. 350–353, 1999.

[14] Leocarlos Bezerra da Silva Lima, Francisco Marcos de Assis, e Lirida Alves de Barros Naviner. Decodificação de Códigos Algébrico-Geométricos (in english, Decoding of Algebraic-Geometric Codes). *Revista da Sociedade Brasileira de Telecomunicações*, 18(3):in press, December 2003.

[15] Leocarlos Bezerra da Silva Lima, Lírida Alves de Barros Naviner, e Francisco Marcos de Assis. Implantation matérielle d'unités arithmétiques en corps finis pour le codage de canal. In *JNRDM Proceedings*, pp. 397–399, 2003.

[16] Leocarlos Bezerra da Silva Lima, Lírida Alves de Barros Naviner, Jocelyn Jaubert, e Francisco Marcos de Assis. Architecture for a decoder of algebraic-geometric codes based on Hermitian curves. In *MWSCAS Proceedings*, 2003.

[17] Leocarlos Bezerra da Silva Lima, Lírida Alves de Barros Naviner, Jocelyn Jaubert, e Francisco Marcos de Assis. Architecture pour un décodeur de codes algébriques basés sur les courbes d'Hermite. In *JSF Proceedings*, pp. 226–230, 2003.

[18] Francisco Marcos de Assis. Hit Probability between Frequency Hopping Sequences Generated by Reed-Solomon and Hermitian Codes. *Electronics Letters*, 32(11):962–963, May 1996.

[19] M. Elia, E. Viterbo, e G. Bertinetti. Decoding of Binary Separable Goppa Codes Using Berlekamp-Massey Algorithm. *Electronics Letters*, 35:1720–1721, 1999.

[20] Peter Elias. List decoding for noisy channels. *Technical Report 335, Research Laboratory of Electronics, MIT*, 1957.

[21] Gui-Liang Feng e T. R. N. Rao. Decoding of Algebraic Geometric Codes up to the Designed Minimum Distance. *IEEE Transactions on Information Theory*, 39(1):37–45, January 1993.

[22] Gui-Liang Feng, Victor K. Wei, T. R. N. Rao, e Kenneth K. Tzeng. Simplified Understanding and Efficient Decoding of a Class of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 40(4):981–1002, July 1994.

[23] Patrick Fitzpatrick. On the Key Equation. *IEEE Transactions on Information Theory*, 41(5):1290–1302, September 1995.

[24] William Fulton. *Algebraic Curves: An Introduction to Algebraic Geometry*. Reading, MA: W. A. Benjamin, 1969.

[25] V. D. Goppa. A New Class of Linear Error-Correcting Codes. *Problems of Information Theory*, 6:207–212, 1970.

[26] Venkatesan Guruswami e Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:432–437, March 1999.

[27] A. Halbutogullari e C. K. Koc. Mastrovito Multiplier for General Irreducible Polynomials. *IEEE Transactions on Computers*, 49(5):503–518, May 2000.

[28] Walaa A. Hamouda e Peter J. McLane. Space-time MMSE Multiuser Detection in multipath channels with RS Coding. In *IEEE International Conference on Communications*, 2001.

[29] Tom Høholdt e Ruud Pellikaan. On the Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 41(6):1589–1614, November 1995.

[30] C. D. Jensen. Fast Decoding of Codes from Algebraic Geometry. *IEEE Transactions on Information Theory*, 40:223–230, January 1994.

[31] G. D. Forney Jr. Generalized Minimum Distance Decoding. *IEEE Transactions on Information Theory*, IT-12:125–131, April 1966.

[32] Jørn Justesen, Knud J. Larsen, H. Elbrønd Jensen, Allan Havemose, e Tom Høholdt. Construction and Decoding of a Class of Algebraic Geometric Codes. *IEEE Transactions on Information Theory*, 35(4):811–821, July 1989.

[33] Jørn Justesen, Knud J. Larsen, H. Elbrønd Jensen, e Tom Høholdt. Fast Decoding of Codes from Algebraic Plane Curves. *IEEE Transactions on Information Theory*, 38(1):111–119, January 1992.

[34] Ralf Kötter. Fast Generalized Minimum-Distance Decoding of Algebraic-Geometry and Reed-Solomon Codes. *IEEE Transactions on Information Theory*, 42(3):721–737, May 1996.

[35] Ralf Kötter. A Fast Parallel Implementation of a Berlekamp-Massey Algorithm for Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 44:1353–1368, July 1998.

[36] Ralf Kötter e A. Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. In *ISIT Proceedings*, 2000.

[37] M. Kurihata e Shajiro Sakata. A fast parallel decoding algorithm for general one-point AG codes with a systolic array architecture. In *ISIT Proceedings*, p. 99, 1995.

[38] Rudolf Lidl e Harald Niederreiter. Finite Fields. In *Encyclopedia of Mathematics and Its Applications*. Cambridge: University Press, 1997.

[39] Shu Lin e Daniel J. Costello. *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[40] C. X. S. Ling. A Class of Linear Codes with Good Parameters from Algebraic Curves. *IEEE Transactions on Information Theory*, 46:1527–1532, 2000.

[41] Jacobus H. Van Lint. Algebraic Geometric Codes. In *Coding Theory and Design Theory (Part I)*, volume 21 of *IMA Volumes Math. Appl.*, pp. 137–162. Berlin: Springer-Verlag, 1990.

[42] Chih-Wei Liu, Kuo-Tai Huang, e Chung-Chin Lu. A fast parallel implementation of Feng-Rao algorithm with systolic array structure. In *ISIT Proceedings*, 1997.

[43] Chih-Wei Liu, Kuo-Tai Huang, e Chung-Chin Lu. A systolic array implementation of the Feng-Rao algorithm. *IEEE Transactions on Computers*, 48:690–706, 1999.

[44] Jacobus H. van Lint e T. A. Springer. Generalized Reed-Solomon Codes from Algebraic Geometry. *IEEE Transactions on Information Theory*, 33(3):305–309, May 1987.

[45] Edoardo D. Mastrovito. *VLSI Architectures for Computations in Galois Fields*. Tese de Doutorado, Linköping University, Sweden, 1991.

[46] A. J. Meneses, editor. *Applications of finite fields*. Boston: Kluwer Academic Publishers, 1993.

[47] Alfred J. Menezes, editor. *Applications of Finite Fields*. Boston: Kluwer Academic Publishers, 1993.

[48] Michael E. O'Sullivan. Decoding of Codes Defined by a Single Point on a Curve. *IEEE Transactions on Information Theory*, 41(6):1709–1719, November 1995.

[49] Michael E. O'Sullivan. Decoding of Hermitian Codes: The Key Equation and Efficient Error Evaluation. *IEEE Transactions on Information Theory*, 46(2):512–523, March 2000.

[50] Michael E. O'Sullivan e S. P. Pope. VLSI architecture for a decoder for Hermitian codes. In *ISIT Proceedings*, p. 376, 1997.

[51] Christof Paar. *Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields*. Tese de Doutorado, Fortschritt-Berichte VDI, Germany, 1994.

[52] V. Ponnampalam e B. Vucetic. Soft decision decoding of Reed-Solomon codes. In *ISIT Proceedings*, 2000.

[53] E. M. Popovici, Michael E. O'Sullivan, Patrick Fitzpatrick, e Ralf Kötter. Implementation of a Hermitian decoder. In *ISIT Proceedings*, p. 311, 2001.

[54] S. C. Porter, Ba-Zhong Shen, e Ruud Pellikaan. Decoding Geometric Goppa Codes Using an Extra Place. *IEEE Transactions on Information Theory*, 38(6):1663–1676, November 1992.

[55] Oliver Pretzel. *Codes and Algebraic Curves*. New York: Oxford University Press, 1998.

[56] John G. Proakis. *Digital Communications*. New York: McGraw-Hill, 1995.

[57] Arash Reyhani-Masoleh e M. Anwar Hasan. A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$. *IEEE Transactions on Computers*, 51(5):511–520, May 2002.

[58] Keith Saints e Chris Heegard. Algebraic-Geometric Codes and Multidimensional Cyclic Codes: A Unified Theory and Algorithms for Decoding Using Gröbner Bases. *IEEE Transactions on Information Theory*, 41(6):1733–1751, November 1995.

[59] S. Sakata, Y. Numakani, e M. Fujisawa. A fast interpolation method for list decoding of RS and algebraic-geometric codes. In *ISIT Proceedings*, p. 479, 2000.

[60] Shajiro Sakata. Extension of the Berlekamp-Massey Algorithm to N Dimensions. *Informat. Comput.*, 84:207–239, February 1990.

[61] Shajiro Sakata, Jørn Justesen, Y. Madelung, H. Elbrønd Jensen, e Tom Høholdt. Fast Decoding of Algebraic-Geometric Codes up to the Designed Minimum Distance. *IEEE Transactions on Information Theory*, 41(5):1672–1677, September 1995.

[62] Shajiro Sakata e M. Kurihata. A systolic array architecture for implementing a fast parallel decoding algorithm for one-point AG codes. In *ISIT Proceedings*, p. 378, 1997.

[63] Shojiro Sakata, Helge Elbrond Jensen, e Tom Høholdt. Generalized Berlekamp-Massey Decoding of Algebraic Geometric Codes Up to Half the Feng-Rao Bound. *IEEE Transactions on Information Theory*, 41:1762–1768, November 1995.

[64] Shojiro Sakata, Douglas A. Leonard, Helge Elbrond Jensen, e Tom Høholdt. Fast Erasure-and-Error Decoding of Algebraic Geometry Codes Up to the Feng-Rao Bound. *IEEE Transactions on Information Theory*, 44:1558–1565, July 1998.

[65] Ba-Zhong Shen e Kenneth K. Tzeng. Decoding Geometric Goppa Codes up to Designed Minimum Distance by Solving a Key Equation in a Ring. *IEEE Transactions on Information Theory*, 41(6):1694–1702, November 1994.

[66] M. Amin Shokrollahi e Hal Wasserman. List Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 45:432–437, March 1999.

[67] Alexei N. Skorobogatov e Sergei G. Vlăduţ. On the Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 36(5):1051–1060, September 1990.

[68] W. Stallings. *Cryptography and Network Security: Principles and Practice.* New Jersey: Prentice-Hall, 1999.

[69] Henning Stichtenoth. A Note on Hermitian Codes over $GF\,(q^2)$. *IEEE Transactions on Information Theory*, 34(5):1345–1348, September 1988.

[70] Henning Stichtenoth. *Algebraic Function Fields and Codes.* Berlin: Springer-Verlag, 1993.

[71] M. A. Tsfasman, S. G. Vlăduţ, e T. Zink. Modular Curves, Shimura Curves and Goppa Codes, Better Than Varshamov-Gilbert Bound. *Math. Nachr.*, 104:13–28, 1982.

[72] S. B. Wicker. *Error control systems for digital communications and storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.

[73] Stephen B. Wicker e Vijay K. Bhargava. *Reed-Solomon codes and their applications*. Piscataway, NJ, USA: IEEE Press, 1994.

[74] John M. Wozencraft. List decoding. *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90–95, 1958.

[75] Huapeng Wu, Anwarul Hasan, e Ian F. Blake. New Low-Complexity Bit-Parallel Finite Field Multipliers Using Weakly Dual Bases. *IEEE Transactions on Computers*, 47(11):1223–1234, November 1998.

[76] Xin-Wen Wu e Paul H. Siegel. Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes. *IEEE Transactions on Information Theory*, 47(6):2579–2587, September 2001.

[77] Chaoping Xing. Algebraic-Geometry Codes with Asymptotic Parameters Better than the Gilbert-Varshamov and the Tsfasman-Vlăduţ-Zink Bounds. *IEEE Transactions on Information Theory*, 47:347–352, January 2001.

[78] Tong Zhang e Keshab K. Parhi. Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials. *IEEE Transactions on Computers*, 50(7):734–749, July 2001.

Índice Remissivo