UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

# Decodificação de Códigos Não Sistemáticos de Reed-Solomon

por

Douglas Goulart Campelo

Dissertação submetida como requisito parcial
para a obtenção do grau de

Mestre em Matemática Aplicada

Prof. Dr. Vilmar Trevisan
Orientador

Porto Alegre, Novembro de 2012.

## CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Campelo, Douglas Goulart

Decodificação de Códigos Não Sistemáticos de Reed-Solomon / Douglas Goulart Campelo.—Porto Alegre: PPGMAp da UFRGS, 2012.

83 p.: il.

Dissertação (mestrado) —Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Matemática Aplicada, Porto Alegre, 2012.
Orientador: Trevisan, Vilmar

Dissertação: Matemática Aplicada
Códigos de Reed-Solomon, Algoritmo de Berlekamp, Algoritmo de Shiozaki-Gao, Algoritmo de Euclides Estendido, Transformada de Fourier em Corpos de Galois, Códigos Corretores de Erro (Teoria da Informação)

# Decodificação de Códigos Não Sistemáticos de Reed-Solomon

por

Douglas Goulart Campelo

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de

## Mestre em Matemática Aplicada

Linha de Pesquisa: Análise Numérica e Computação Científica

Orientador: Prof. Dr. Vilmar Trevisan

Banca examinadora:

Profª Drª Renata Raposo Del-Vecchio
UFF

Profª Drª Catia Maria dos Santos Machado
FURG

Prof. Dr. Luiz Emilio Allem
IM - UFRGS

Dissertação apresentada e aprovada em 23/11/2012.

Profª Drª Maria Cristina Varrialle
Coordenadora

# Sumário

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Nesta dissertação de mestrado estudamos códigos de Reed-Solomon. Começamos fazendo uma revisão sobre extensões de corpos finitos, focando na maneira de representar e operar com os seus elementos, e também sobre teoria de códigos, explorando os códigos lineares e os códigos cíclicos. Apresentamos as duas construções dos códigos de Reed-Solomon, a original, como a imagem de uma função polinomial, e a descoberta por Gorenstein e Zierler, como o ideal gerado por um polinômio gerador. Terminamos mostrando um algoritmo devido a Gao que mostra como decodificar palavras código de Reed-Solomon codificadas de maneira não sistemática.

# ABSTRACT

In this dissertation we study Reed-Solomon codes. We begin with a review about extensios of finite fields, focusing on the way to represent and operate with its elements, and also about the theory of codes, exploting a few properties of linear codes and codes cyclic. We present two constructions of Reed-Solomon codes, the original, as the image of a polynomial function, and the discovery by Gorenstein and Zierler, as the ideal generated by a polynomial generator. Finished showing an algorithm due to Gao that shows how to decode Reed-Solomon code words coded in a nonsystematic way.

# AGRADECIMENTOS

À minha família. Ao meu irmão Rodrigo Campelo por todos estes anos de carinho, de incentivo, de compreensão e de exemplos para a minha vida. A minha mãe Nadia Goulart e ao meu pai Ubirajara Campelo que, de uma forma ou de outra, nunca me deixaram desanimar e sempre me ajudaram. Amo muito vocês.

Ao Professor e amigo Vilmar Trevisan que desde a graduação, em que me incentivou e ajudou a seguir em frente em momentos muito difíceis pra mim, até agora, na reta final, sempre me estimulou, acompanhou e esteve junto quando eu precisava. Tenho certeza que não teria chegado aqui sem o seu apoio. Do fundo do meu coração, muito obrigado!

Ao Professor e amigo Luiz Emilio Allem por toda a ajuda, desde a sugestão do problema deste trabalho até agora e por todas as sábias dicas que sempre me deu, obrigado!

Aos meus amigos Everson Gomes e Julio Cesar Lombaldo Fernandes, por quem tenho um carinho muito grande e só tenho a agradecer por todo o apoio que me deram agora e desde que entramos na UFRGS.

À minha namorada Caroline Corrêa, por toda a paciência, companheirismo e ajuda prestada nesta reta final. Te amo!

À Banca examinadora pelas contribuições a este trabalho e aos professores do IM pela formação e pelo carinho dados desde a minha graduação.

Ao IM e ao PPGMAP pelos laboratórios e salas disponibilizados, assim como pela ajuda e prestatividade de seus funcionários.

À UFRGS pelo ensino público, gratuito e de qualidade.

Ao CNPQ pelo apoio financeiro.

# 1 INTRODUÇÃO

Durante a transmissão de dados às vezes ocorrem problemas e a informação transmitida acaba sofrendo a ação dos chamados **ruídos** ou **erros**, causando assim uma discrepância entre o objeto enviado e o recebido. O objetivo da Teoria de Códigos é estudar formas de corrigir estes problemas. Esta teoria teve início nos anos 40 do século passado e nela o obejeto transmitido é chamado de **palavra** ou **mensagem** (uma sequência $m_1, \ldots, m_n$, de comprimento $n$, formada por símbolos tomados em um conjunto $A$ com $q$ elementos, chamado **alfabeto**) e o tratamento do erro ocorre pelo uso de um **código** (um conjunto $C$ formado por sequências $c_1, \ldots, c_n, \ldots, c_i \in A$ [7]) junto à transmissão, de maneira que o objeto transmitido não é a palavra, mas sim a chamada **palavra código**, que é um elemento do código. Esta situação é ilustrada no seguinte esquema

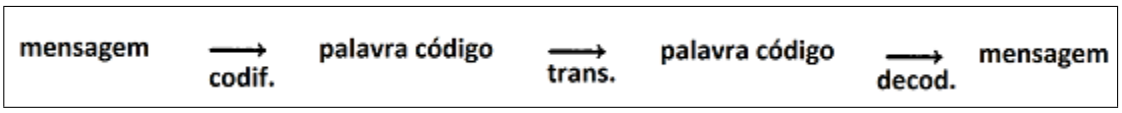


Figura 1.1: Esquema de transmissão

Golay, Hamming e Shannon foram os grandes pioneiros que iniciaram os trabalhos com este assunto e desenvolveram estudos e idéias que são usadas até hoje nas mais diversas situações do armazenamento e da transmissão da informação.

Em 1947, Richard W. Hamming, irritado com a limitação dos computadores da época em apenas detectar os erros dos cartões perfurados, sem corrigí-los [21], desenvolveu um código corretor de único erro e códigos que detectam até dois erros e corrigem um único erro [12]. Golay deu continuidade aos estudos de Hamming generalizando o código binário e perfeito de Hamming, ele ainda criou uma classe de códigos que leva o seu nome. Em outubro de 1948, C. E. Shannon no artigo *A Mathematical Theory of Communication* [28] mostrou que, sobre uma certa

condição, a transmissão pode ser tão confiável quanto se queira. Esta publicação marcou o início de um desenvolvimento contínuo da teoria de códigos.

Em junho de 1960, Irving Reed e Gustave Solomon publicaram no artigo *Polynomial Codes over Certain Finite Fields* [27] um código baseado no que hoje conhecemos como **transformada de Fourier em corpos de Galois**. Os códigos RS (Reed-Solomon) possuem diversas características interessantes que fizeram com que o seu uso fosse extramamente difundido. Além da **distância mínima** desses códigos ser a melhor possível sua estrutura não binária permite tratar um tipo peculiar de erros, as chamadas **rajadas de erros**. O tratamento deste tipo de erros fez dos códigos RS um ponto chave no sistema brasileiro de televisão digital [20], além de ser amplamente utilizado em modens de altas velocidades, como ADSL e CDSL, assim como em sistemas de comunicação móvel, como em telefones celulares e tablets. O mesmo ainda é utilizado nos mais diversos tipos de armazenamento de dados, da fita magnética de um cartão de crédito, passando pelos discos compactos (CDs) e os discos digitais versáteis (DVDs), até os mais robustos discos rígidos (HDs).

Em 1959, o matemático Alexis Hocquenghem apresentou um código que, ainda em 60, seria redescoberto por R. C. Bose e D. K. Ray-Chaudhuri de maneira independente e viria a se chamar **código BCH** (Bose-Chaudhuri-Hocquenghem). No ano seguinte, Gorenstein e Zierler [11] apresentaram uma generalização não binária dos códigos BCH que incluia, como um caso particular, um código equivalente aos códigos RS. Mostrando, assim, que os códigos RS são na verdade subcódigos dos códigos BCH.

A maneira como os códigos RS se difundiram não seguiu a abordagem original da transformada de Fourier, ela foi baseada no que chamamos de **polinômio gerador** seguindo uma codificação chamada de **sistemática** e é devida ao trabalho de Berlekamp, que foi o primeiro a fornecer um método aplicável de decodificação para estes códigos em 1967. Em 1988, Shiozaki inventou um algoritmo de decodifi-

cação para a abordagem original (não sistemática) dos códigos RS que é comparável ao algoritmo de Berlekamp. Recentemente, em 2003, Gao reinventou o algoritmo de Shiozaki de maneira independente.

O objetivo deste trabalho é apresentar algumas propriedades dos códigos de Reed-Solomon e mostrar o algoritmo de Shiozaki-Gao para a decodificação de palavras código codificadas de maneira não sistemática.

Este texto está organizado da seguinte maneira: no capítulo 1, exibimos um breve resumo histórico dos códigos de Reed-Solomon. No capítulo 2, apresentamos as ferramentas algébricas necessárias para o entendimento da teoria de códigos. No capítulo 3, uma introdução a teoria de códigos é apresentada. No capítulo 4, descrevemos os códigos de Reed-Solomon. No capítulo 5, explicamos o problema de decodificação para os códigos de Reed-Solomon.

# 2 FUNDAMENTOS ALGÉBRICOS

Neste capítulo apresentamos alguns resultados da teoria de corpos finitos necessários ao entendimento da teoria de códigos e em particular dos códigos de Reed-Solomon.

## 2.1 Elementos da Teoria de Corpos Finitos

Nesta seção mostramos alguns resultados da teoria de corpos finitos que permitem representar e efetuar operações aritméticas com os elementos desta estrutura algébrica. Começamos com algumas definições e resultados que se aplicam a anéis. No entanto, nosso interesse estará nas implicações destes aos corpos finitos.

A primeira parte mostra alguns elementos da teoria de anéis e serve como base para os resultados sobre extensões de corpos e corpos finitos. Esta seção é baseado em [15], [16] e [25] onde podem ser encontradas as demonstrações omitidas aqui.

### 2.1.1 Anéis e Corpos

**Definição 2.1.1.** *Um* ***anel*** $(R, +, \cdot)$ *é um conjunto* $R$ *munido de duas operações binárias, chamadas adição e multiplicação, satisfazendo para quaisquer* $a, b, c \in R$

*(1)* $(R, +)$ *é um grupo abeliano;*

*(2)* $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ *(Associatividade da multiplicão);*

*(3)* $a \cdot (b + c) = (a \cdot b) + (a \cdot c), a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ *(Distributividade da multiplicação em relação à adição).*

No lugar de escrever $a \cdot b$ é comum escrever $ab$. Também costuma-se utilizar o símbolo 0 para o elemento neutro da adição e $-a$ para o inverso aditivo de $a$.

**Exemplo 2.1.1.** $(\mathbb{Z}, +, \cdot)$ *é um anel, cujas operações* $+$ *e* $\cdot$ *são, respectivamente, as operações de adição e multiplicação usuais dos inteiros.*

**Exemplo 2.1.2.** *Para cada* $n \geq 0$ *o conjunto* $n\mathbb{Z} = \{na | a \in \mathbb{Z}\}$*, cujas operações* $+$ *e* $\cdot$ *são induzidas pelas operações em* $\mathbb{Z}$*, é um anel.*

**Exemplo 2.1.3.** $(R, +, \cdot)$*, onde* $R = \{f : \mathbb{R} \to \mathbb{R} | f$ *é função*$\}$*, é um anel. Para* $f, g \in R$ *as operações* $f + g$ *e* $f \cdot g$ *são definidas por*

*(1)* $(f + g)(x) = f(x) + g(x)$, $\forall x \in \mathbb{R}$,

*(2)* $(f \cdot g)(x) = f(x) \cdot g(x)$, $\forall x \in \mathbb{R}$.

**Exemplo 2.1.4.** *Seja* $R$ *um anel. O conjunto* $R[x]$*, formado pelos polinômios com coeficientes em* $R$*, possui as operações de soma e multiplicação para* $f = \sum_{i=0}^{n} a_i x^i, g = \sum_{j=0}^{m} b_i x^j \in R[x]$*, com* $m, n \geq 0$*, dadas por*

*(1)* $f + g = \sum_{i=0}^{\max m,n} (a_i + b_i) x^i$,

*(2)* $f \cdot g = \sum_{l=0}^{m+n} (a_0 b_l + a_1 b_{l-1} + \cdots + a_l b_0) x^l$.

$(R[x], +, \cdot)$ *é um anel.*

Alguns anéis possuem propriedades específicas.

**Definição 2.1.2.** *Um* ***anel comutativo*** *é um anel em que a multiplicação é comutativa.*

**Definição 2.1.3.** *Um* ***anel com identidade*** *é um anel em que a multiplicação possui um elemento neutro.*

Costuma-se denotar este elemento por 1.

**Exemplo 2.1.5.** $(\mathbb{Z}_n, +, \cdot)$ *é um anel comutativo, onde* $n \geq 0$ *e* $\mathbb{Z}_n = \{\overline{0}, \overline{1}, \ldots, \overline{n-1}\}$. *Para* $\overline{a}, \overline{b} \in \mathbb{Z}_n$ *as operações* $+$ *e* $\cdot$ *são definidas por*

*(1)* $\overline{a} + \overline{b} = \overline{a+b}$,

*(2)* $\overline{a} \cdot \overline{b} = \overline{ab}$.

*a operação* $\cdot$ *possui elemento neutro* $\overline{1}$.

**Definição 2.1.4.** *Um* ***anel com divisão*** *é um anel em que os seus elementos não nulos formam um grupo sob a multiplicação.*

**Exemplo 2.1.6.** $\mathbb{Z}_p$ *com* $p$ *primo é um anel com divisão.*

**Definição 2.1.5.** *Um* ***domínio****, ou um* ***anel, de integridade****, é um anel comutativo com identidade em que,* $ab = 0$ *implica que* $a = 0$ *ou* $b = 0$, *para quaisquer* $a, b$ *no anel.*

**Exemplo 2.1.7.** *Seja* $R$ *é um domínio de integridade. Então, para* $f, g \in R[x]$, $\text{grau}(fg) = \text{grau}(f) + \text{grau}(g)$. *Assim, se* $f$ *e* $g$ *são não nulos, então* $fg$ *também é não nulo. Neste caso, então,* $R[x]$ *também é um domínio de integridade.*

**Definição 2.1.6.** *Um* ***corpo*** *é um anel comutativo não trivial em que cada elemento não nulo é invertível.*

**Exemplo** Os números racionais $\mathbb{Q}$. Dados $a, b \in \mathbb{Z}$ não nulos, temos $a/b \in \mathbb{Q}$ cujo inverso é dado por $(a/b)^{-1} = b/a$.

Sendo $R$ um anel, podemos definir a multiplicação entre um inteiro não negativo $n$ e um elemento $a \in R$ por $n \cdot a = a + \cdots + a$ ($n$ vezes) se $n > 0$ e $n \cdot a = 0$ se $n = 0$.

**Definição 2.1.7.** *Para um anel* $R$ *em que existe um inteiro positivo* $n$ *tal que* $n \cdot 1 = 0$. *Chamamos* $n$ *de* ***característica*** *de* $R$, *denotada por* $car(R)$. *Se tal* $n$ *não existe, diz-se que* $R$ *possui característica zero.*

**Definição 2.1.8.** *Um subgrupo aditivo $H$ de um anel $R$ é um* ***ideal*** *de $R$, se $ah \in H$ para quaisquer $h \in H$ e $a \in R$.*

Assim como os anéis, alguns ideais se destacam por suas propriedades.

**Definição 2.1.9.** *Seja $R$ um anel. Um ideal $H$ de $R$ é chamado de* ***principal*** *ou de* ***ideal principal gerado por a*** *quando existe $a \in R$ tal que $H = (a)$.*

**Exemplo 2.1.8.** *Os ideais $(n) = n\mathbb{Z}$ são todos principais.*

**Definição 2.1.10.** *Seja $R$ um anel. Um ideal $I$ de $R$ é chamado de* ***primo*** *quando $I \neq R$ e $ab \in I$, implica que $a \in I$ ou $b \in I$.*

**Definição 2.1.11.** *Um ideal $M$ é de um anel $R$ dito* ***maximal*** *se $M \neq R$ e se os únicos ideais $I$ de $R$ tais que $M \subseteq I \subseteq R$ são $I = M$ e $I = R$.*

**Exemplo 2.1.9.** *Em $\mathbb{Z}$, o ideal $(7)$ é maximal. $(7) \subseteq (m) \subseteq \mathbb{Z} \Leftrightarrow m|7 \Rightarrow m = 1$ ou $m = 7 \Leftrightarrow (m) = \mathbb{Z}$ ou $(m) = (7)$.*

**Definição 2.1.12.** *Um anel $R$ é um* ***domínio de ideais principais*** *se $R$ é um domínio de integridade e se cada ideal $I$ de $R$ é principal.*

Uma **classe lateral** de um ideal $H$ em um anel $R$ é o conjunto $a+H = \{a + h | h \in H\}$ com $a \in R$. O conjunto de todas essas classes laterais forma o chamado **anel quociente** de $R$ por $H$ e é denotado por $R/H$.

**Teorema 2.1.** *Seja $R$ um anel. Se $R$ é um domínio de ideais principais, então $R/(p)$ é um corpo se e somente se $p$ é irredutível em $R$.*

**Prova**. Sabe-se da teoria de anéis que $R/(p)$ é corpo se e somente se $(p)$ é maximal. Usando este resultado, suponhamos que o ideal $(p)$ é maximal e que $p = ab$, sendo $a, b \in R$. Como $p = ab$, $(p) \subseteq (a)$, mas como $(p)$ é maximal, temos duas possibilidade ou $(a) = (p)$ ou $(a) = R$. Se $(a) = (p)$, existe um certo $r \in R$ tal que $a = rp$, ou seja, $p = ab = rpb$ o que permite que escrevamos $p(1 - rb) = 0$, mas como $R$ é

domínio de integridade e $p \neq 0$, então $rb = 1$ e portanto $b$ é invertível. Suponhamos agora que $p$ é irredutível e que $(p) \subseteq (a)$ para algum $a \in R$. Segue que $p = ar$ sendo $r \in R$. Como $p$ é irredutível, então $a$ ou $r$ é invertível e por consequência $(a) = R$ ou $(a) = (p)$. Assim, $(p)$ é maximal.

**Exemplo 2.1.10.** $\mathbb{Z}/(5), \mathbb{Z}/(7), \mathbb{Z}/(11)$ *são corpos.*

**Teorema 2.2.** *Um subconjunto $K$ de um corpo $F$ é um subcorpo se e somente se as seguintes condições são satisfeitas*

*(1) Existe algum elemento não nulo em $K$;*

*(2) Se $a, b \in K$, $a - b \in K$;*

*(3) Se $a, b \in K$ e $b \neq 0$ então $ab^{-1} \in K$.*

**Definição 2.1.13.** *Sejam $K$ e $F$ corpos e $M$ um conjunto, sendo $M$ e $K$ subconjuntos de $F$. O corpo $K(M)$ é a intersecção de todos os subcorpos de $F$ que contém $K$ e $M$, o chamamos de o* ***corpo de extensão*** *de $K$ obtido pela adjunção dos elementos de $M$. Quando $M$ possui apenas um elemento $\alpha$, dizemos que $K(\alpha)$ é uma* ***extensão simples*** *de $K$.*

Um corpo de extensão de $K$ é um espaço vetorial. Seja $V$ tal extensão, a multiplicação de um elemento de $V$ por um escalar de $K$ é fechada e satisfaz

$$\begin{aligned} v(\alpha + \beta) &= v\alpha + v\beta \\ 1_K\alpha &= \alpha \\ (vt)\alpha &= v(t\alpha) \\ (v + t)\alpha &= v\alpha + t\alpha \end{aligned}$$

para quaisquer $\alpha, \beta \in V$ e $v, t \in K$ e onde $1_K$ denota o neutro multiplicativo em $K$. Além disso, $(L, +)$ é um grupo abeliano.

**Teorema 2.3.** *Seja $F$ um corpo. Então $F[x]$ é um domínio de ideais principais. Mais precisamente, para qualquer ideal não nulo $I$ de $F[x]$, existe um único polinômio mônico $f \in F[x]$ tal que $I = (f)$.*

**Prova**. Pelo exemplo 2.1.7, $F[x]$ é um domínio de integridade. Seja $I$ um ideal não nulo de $F[x]$. Seja $f$ um polinômio mônico não constante de menor grau em $I$. Claramente $(f) \subseteq I$. Para a outra inclusão, seja $g \in I$. Pelo algoritmo da divisão, escrevemos $g = fh + r$ onde $h, r \in F[x]$ e $grau(r) < grau(f)$. Como $r = g - fh$ está em $I$, a minimalidade do grau de $f$ implica que $r = 0$. Portanto $g = fh$ e assim $I = (f)$. Para a unicidade, tomemos $l$ um polinômio mônico sobre $F$ e suponhamos que $I = (l)$. Então $f|l$ e $l|f$, mas como $f, l$ são ambos mônicos, obtemos $f = l$.

**Teorema 2.4.** *Sejam $F$ um corpo e $f$ um polinômio mônico de grau positivo sobre $F$. Então $F[x]/(f)$ é um corpo se e somente se $f$ é irredutível sobre $F$.*

**Teorema 2.5.** *Sejam $F$ um corpo e $f$ um polinômio mônico de grau positivo $n$ sobre $F$. Então o anel quociente $F[x]/(f)$ pode ser descrito como*

$$\{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} | a_0, a_1, \ldots, a_{n-1} \in F \text{ e } f(\alpha) = 0\}$$

**Definição 2.1.14.** *Sejam $K$ e $F$ corpos, sendo $K$ uma extensão de $F$. Dizemos que $K$ é um* ***corpo de decomposição*** *do polinômio $f \in F[x]$, se $f$ é um produto de fatores lineares em $K[x]$ e se $f$ não é um produto de fatores lineares em nenhum subcorpo próprio de $K$ contendo $F$.*

Assim, o corpo de decomposição de um polinômio sobre $F$ é o menor corpo que contém $F$ e as raízes deste polinômio. Para um dado polinômio $f$ sobre $F$ existe apenas um corpo de decomposição, exceto por isomorfismos.

**Teorema 2.6.** *Sejam $F$ um corpo e $f$ um polinômio sobre $F$. Existe uma extensão $K$ de $F$ que é um corpo de decomposição de $f$. Além disso, quaisquer dois corpos de decomposição de $f$ sobre $F$ são isomorfos.*

### 2.1.2 Extensões de Corpos Finitos

Quando um corpo $F$ possui uma quantidade finita $q$ de elementos, este é chamado de **corpo finito** e a quantidade $q$ recebe o nome de **ordem** de $F$. O número $q$ pode ser expresso como uma potência de algum primo, isto é, para algum primo $p$ e algum inteiro $m$, $q = p^m$. Além disso, todos os corpos com $q$ elementos são isomorfos. Por isso costuma-se falar no corpo finito com $p^m$ elementos, como se houvesse apenas um. Este corpo é denotado por $GF(p^m)$. Esta notação $GF(p^m)$ vem da expressão em inglês *Galois field* que, em português, significa *corpo de Galois*, em homenagem ao matemático francês Évariste Galois.

Se $L$ é um corpo de extensão de $F$, já vimos que $L$ é um espaço vetorial. Quando $L$ possui dimensão finita, esta dimensão é chamada de **grau** de $L$ sobre $F$ e é denotada por $[L : F]$. Neste caso, dizemos que $L$ é uma **extensão finita** de $F$.

**Teorema 2.7.** *Sejam $F$ um corpo finito de ordem $q$ e $L$ uma extensão finita de $F$ de grau $n$. Então, a ordem de $L$ é $q^n$. Em particular, se $L$ é um corpo finito de ordem $q$ e característica $p$, então $q = p^n$ onde $n$ é o grau da extensão de $F$ sobre $GF(p)$.*

**Prova**. Para cada um dos $n$ elementos da base $\{b_1, \ldots, b_n\}$ de $L$ existem $q$ possíveis coeficiente na expressão de um elemento de $L$: $l_1b_1 + \cdots + l_nb_n$. Assim, $L$ possui $q^n$ elementos. No caso de $L$ finito, sabemos que a sua característica é prima e portanto contém o corpo primo $GF(p)$.

**Lema 2.1.1.** *Em $GF(q)$ $a^q = a$ para qualquer $a \in GF(q)$.*

**Prova**. Se $a = 0$, é imediato. Como $GF(q) \setminus \{0\}$ é um grupo multiplicativo de ordem $q - 1$, então $a^{q-1} = 1$ para todo $a \neq 0$.

**Lema 2.1.2.** *Seja $F$ um corpo de ordem $q$ e caraterística $p$. Então $F$ é um corpo de decomposição de $x^q - x$ sobre $GF(p)$ e o polinômio $x^q - x$ fatora-se em fatores lineares em $F[x]$.*

**Prova**. Como, pelo lema 2.1.1, qualquer $a \in F$ é raiz de $x^q - x$ e como este possui no máximo $q$ raízes em $F$, então $x^q - x$ fatora-se sobre $F$.

**Lema 2.1.3.** *Em um corpo de característica $p$, dado um inteiro $n \geq 0$, temos que*

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}.$$

**Prova**. Por indução: para $n = 1$ a expressão torna-se o binômio de Newton $(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i} b^i$ cujos coeficientes são todos zero mod $p$.

Da hipótese de indução,

$$(a+b)^{p^{n+1}} = ((a+b)^{p^n})^p = (a^{p^n} + b^{p^n})^p = a^{p^{n+1}} + b^{p^{n+1}}.$$

Logo, $(a+b)^{p^n} = a^{p^n} + b^{p^n}$ para todo inteiro $n \geq 0$.

**Teorema 2.8.** *Para qualquer primo $p$ e qualquer inteiro $n$, com $n > 0$, existe um corpo finito com $p^n$ elementos e qualquer corpo finito com $p^n$ elementos é isomorfo ao corpo de decomposição de $x^{p^n} - x$ sobre $GF(p)$.*

**Prova**. Nesta demonstração fazemos uso do seguinte resultado, se uma dado polinômio $g$ possui raízes repetidas, então o $mdc(g, g') \neq 1$, ver [16].

Seja $F$ o corpo de decomposição de $x^q - x$ sobre $GF(p)$, onde $q = p^n$. Como $mdc(x^q - x, qx^{q-1} - 1) = mdc(x^q - x, -1) = 1$, então todas as raízes de $x^q - x$ são distintas. Seja o conjunto $S = \{a \in F | a^q = a\}$. Podemos ver que $S \subseteq F$ e que $0, 1 \in S$. Dados $a, b \in S$, pelo lema 2.1.3 $(a-b)^q = a^q + (-b^q) = a^q - b^q = a - b$, logo $a - b \in S$. Supondo $b \neq 0$, temos que $(ab^{-1})^q = a^q (b^q)^{-1} = ab^{-1}$, assim $ab^{-1} \in S$. Pelo teorema 2.2, $S$ é um subcorpo. Como $S$ possui todas as raízes de $x^q - x$, $S = F$ e $F$ é um corpo finito com $q$ elementos.

Seja $H$ um corpo com $q = p^n$ elementos. $H$ possui característica $p$ e contém $GF(p)$ como um subcorpo primo. Pelo lema 2.1.2, $H$ é um corpo de decomposição de $x^q - x$. Aplicando o teorema 2.6 vemos que $H$ e $F$ são isomorfos.

Como consequência da unicidade de corpos finitos e do teorema 2.4 temos o seguinte resultado.

**Teorema 2.9.** *Seja $f$ um polinômio de grau $n$ irredutível sobre $GF(q)$, onde $q = p^n$, então $GF(q) \cong GF(p)[x]/(f)$.*

**Teorema 2.10.** *Seja $m(x)$ um polinômio irredutível sobre $F$. Então $F/(m(x))$ é uma extensão de $F$ em que $m(x)$ possui uma raíz.*

**Lema 2.1.4.** *Sejam $R$ um anel e $r \in R$. Suponhamos que $m$ e $n$ sejam inteiros positivos tais que $m|n$. Então $(r^m - 1)|(r^n - 1)$.*

**Prova**. Seja $s$ um inteiro tal que $n = sm$, então

$$r^n - 1 = (r^m)^s - 1 = (r^m - 1)((r^m)^{l-1} + \cdots + 1). \tag{2.1}$$

**Teorema 2.11.** *Seja $GF(q)$ com $q = p^n$. Então todo subcorpo de $GF(q)$ possui ordem $p^m$, onde $m$ é um divisor positivo de $n$. Reciprocamante, Se $m$ é um divisor positivo de $n$, então existe um subcorpo de $GF(q)$ com $p^m$ elementos.*

**Prova**. Se $q = p^n$, então qualquer subcorpo $F$ de $GF(q)$ possui ordem $p^m$ com $0 \leq m \leq n$. Se $[GF(q) : F] = l$, então $p^n = (p^m)^l = p^{ml}$, pelo teorema 2.7, e assim $m|n$.

Por outro lado, se $m$ é um divisor positivo de $n$, pelo lema 2.1.4, obtemos que $(p^m - 1)|(p^n - 1)$

$$(x^{p^m-1} - 1)|(x^{p^n-1} - 1),$$

pela aplicação do lema 2.1.4 no anel $GF(p)[x]$. Deduzimos que toda raiz de $(x^{p^m} - x)$ é uma raiz de $x^{p^n} - x = x^q - x$. Assim, toda raiz de $x^{p^m} - x$ pertence a $GF(q)$. Segue que $GF(q)$ deve conter um corpo de decomposição de $x^{p^m} - x$ sobre $GF(q)$. O teorema 2.8 implica que este corpo de decomposição possui $p^m$ elementos. Portanto,

existe exatamente um subcorpo com $p^m$ elementos caracterizado pelas raízes do polinômio $x^{p^m} - x$ em $GF(q)$.

#### *2.1.2.1 Polinômio Mínimo*

Seja $\alpha \in E$ algébrico sobre $F$. Definimos o polinômio mínimo de $\alpha$ sobre $F$, denotado por $m_\alpha(x)$, como o polinômio mônico de menor grau em $F[x]$ tendo $\alpha$ como uma raíz. Este polinômio possui algumas propriedades importantes para o estudo de extensões de corpos e corpos quocientes apresentadas nas seguintes proposições. Em particular, para o quociente $F[x]/(m_\alpha(x))$, $\alpha = x + (m_\alpha(x))$.

**Proposição 2.1.1.** *Seja $\alpha \in E$ com polinômio mínimo $m_\alpha(x)$ sobre $F$.*

**(1)** *$m_\alpha(x)$ é único;*

**(2)** *$m_\alpha(x)$ é irredutível sobre $F$;*

**(3)** *Para qualquer $f(x) \in F[x]$*

$$f(\alpha) = 0 \Leftrightarrow m_\alpha(x) | f(x).$$

**Proposição 2.1.2.** *Se $\alpha \in E$ é a raiz de um polinômio mônico irredutível $m(x)$ sobre $F$, então $m(x)$ é o polinômio mínimo de $\alpha$ sobre $F$.*

**Teorema 2.12.** *Seja $\alpha \in E$ algébrico sobre $F \leq E$ com polinômio mínimo $m_\alpha$ sobre $F$. Então,*

$$F(\alpha) \cong F[x]/(m_\alpha(x)).$$

O isomorfismo que existe entre o corpo quociente $F[x]/(m_\alpha(x))$ e o corpo extensão $F(\alpha)$ dado por

$$\psi : a(x) + (m_\alpha(x)) \mapsto a(\alpha)$$

garante-nos uma representação polinomial igual a que apresentamos para o quociente $F[x]/(m(x))$ no teorema 2.5. Como cada classe lateral de $F[x]/(m_\alpha(x))$ pode ser expressa na forma $a(x) + (m_\alpha(x))$ com $grau(a(x)) < n$ e como $\psi$ é sobrejetora, tal representação é possível para cada elemento da extensão $F(\alpha)$. A injeção da mesma deixa claro que tal representação é única.

**Corolário 2.1.1.** *Seja $\alpha \in E$ com polinômio mínimo $m_\alpha$, de grau $n$, sobre $F$. Então cada $\beta \in F(\alpha)$ pode ser representado de maneira única na forma*

$$\beta = a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} \ (a_i \in F). \tag{2.2}$$

Como $m(\alpha) = 0$ em $F(\alpha)$, a aritmética em $F(\alpha)$ é a aritmética polinomial módulo $m_\alpha(\alpha)$, tratando $\alpha$ como se fosse uma incógnita (apesar de $\alpha$ não ser desconhecido).

**Exemplo 2.1.11.** *O polinômio $x^4 + x + 1$ é irredutível sobre $\mathbb{Z}_2$, o que pode ser verificado por exaustão.*

*Pelo teorema 2.10 $x^4 + x + 1$ possui uma raiz $\alpha$ em uma extensão de $\mathbb{Z}_2$, a saber $\mathbb{Z}_2[x]/(x^4 + x + 1)$. Além disso, pela proposição 2.1.2, $x^4 + x + 1$, é o polinômio mínimo de $\alpha$ sobre $\mathbb{Z}_2$. Pelo corolário 2.1.1, cada elemento de $\mathbb{Z}_2(\alpha)$ pode ser unicamente representado na forma*

$$a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 \ (a_i \in \mathbb{Z}_2).$$

*Portanto, $\mathbb{Z}_2(\alpha)$ é $GF(2^4)$, uma extensão de $\mathbb{Z}_2$ com 16 elementos.*

#### *2.1.2.2 Representando Elementos*

A representação dos elementos da extensão $GF(q)$, com $q = p^n$ e $p$ primo, depende da escolha de um polinômio irredutível $m(x)$ (de grau $n$) sobre o

corpo base $GF(p)$. Escolhido o irredutível $m(x)$ podemos representar os elementos de $GF(q)$ por meio dos polinômios de grau menor que $n$. Esta representação é chamada de **representação polinomial**.

**Exemplo 2.1.12.** *Usando o irredutível $m(x) = x^2 + x + 1$ podemos usar os polinômios de grau menor que 2 para representar $GF(2^2)$.*

| *Representação Polinomial* |
|---|
| $0$ |
| $1$ |
| $x$ |
| $x+1$ |

*Tabela 2.1: $GF(2^2)$ - Representação Polinomial*

Podemos usar a representação polinomial substituindo $x$ pela raiz $\alpha$ de $m(x)$. Ver exemplo 2.1.11.

Associada a representação polinomial temos a **representação vetorial**. Nesta os elementos do corpo $GF(q)$ são expressos como vetores cujas entradas são os coeficientes da representação polinomial.

**Exemplo 2.1.13.** *Usando o irredutível $m(x) = x^3 + x + 1$ podemos usar os polinômios de grau menor que 3 para representar $GF(2^3)$.*

| *Representação Polinomial* | *Representação Vetorial* |
|---|---|
| $0$ | *000* |
| $1$ | *001* |
| $x$ | *010* |
| $x^2$ | *100* |
| $x+1$ | *011* |
| $x^2+x$ | *110* |
| $x^2+x+1$ | *111* |
| $x^2+1$ | *101* |

*Tabela 2.2: $GF(2^3)$ - Representações Polinomial e Vetorial*

Outra representação utilizada é a chamada **representação em série**, baseada no gerador do grupo multiplicativo do corpo. Isolamos o termo de maior grau do polinômio irredutível, em $\alpha$, e vamos construindo o corpo por meio de multiplicações deste termo conhecido por $\alpha$.

**Exemplo 2.1.14.** *Usando o irredutível* $m(x) = x^3 + x + 1$ *podemos usar os polinômios de grau menor que 3 para representar* $GF(2^3)$.

| *Representação em Série* | *Representação Polinomial* | *Representação Vetorial* |
|---|---|---|
| 0 | 0 | *000* |
| 1 | 1 | *001* |
| $\alpha$ | $x$ | *010* |
| $\alpha^2$ | $x^2$ | *100* |
| $\alpha^3$ | $x+1$ | *011* |
| $\alpha^4$ | $x^2+x$ | *110* |
| $\alpha^5$ | $x^2+x+1$ | *111* |
| $\alpha^6$ | $x^2+1$ | *101* |

*Tabela 2.3:* $GF(2^3)$ *- Representações em Série, Polinomial e Vetorial*

#### *2.1.2.3 Aritmética*

As operações aritméticas em um corpo extensão $F/(m(x))$ são calculadas tomando o resto da divisão pelo polinômio irredutível $m(x)$.

**Exemplo 2.1.15.** *Em* $GF(2^3)$ *usando o irredutível* $m(x) = x^3 + x + 1$ *calculamos*

$$x^2(x^2+1) \mod m(x) = x^4 + x^2 \mod m(x) = x.$$

É comum utilizar-se de tabelas para representar as operações nestes corpos. Cada tabela pode ser representada de uma forma diferente, dependendo da maneira adotada para representar os elementos do corpo e do polinômio irredutível utilizado.

**Exemplo 2.1.16** ($GF(2^2)$)**.** *Podemos definir* $GF(2^2)$ *como sendo o conjunto dos polinômios com coeficientes em* $GF(2)$ *módulo algum polinômio irredutível sobre* $GF(2)$ *de grau 2. Neste caso é realmente simples encontrar tal irredutível. São poucos os polinômios de grau 2 em* $GF(2)$, $x^2$, $x^2+1$, $x^2+x$ *e* $x^2+x+1$, *destes podemos descartar três explicitando suas fatorações,* $x^2 = x \cdot x$, $x^2+1 = (x+1)^2$ *e* $x^2+x = x(x+1)$. *O único candidato a irredutível que resta é* $x^2+x+1$ *e como basta testar dois elementos como possíveis raízes, vemos que* $x^2+x+1$ *é o único polinômio irredutível de grau 2 sobre* $GF(2)$. *Portanto,* $GF(2^2) \cong \mathbb{Z}_2[x]/(x^2+x+1) = \{0, 1, x, x+1\}$ *com adição e multiplicação módulo* $x^2+x+1$ *representadas nas seguintes tabelas em uma representação polinomial.*

| $+$ | $0$ | $1$ | $x$ | $x+1$ |
|---|---|---|---|---|
| $0$ | $0$ | $1$ | $x$ | $x+1$ |
| $1$ | $1$ | $0$ | $x+1$ | $x$ |
| $x$ | $x$ | $x$ | $0$ | $1$ |
| $x+1$ | $x+1$ | $x+1$ | $1$ | $0$ |

*Tabela 2.4: Adição em* $GF(2^2)$

| $\times$ | $0$ | $1$ | $x$ | $x+1$ |
|---|---|---|---|---|
| $0$ | $0$ | $0$ | $0$ | $0$ |
| $1$ | $0$ | $1$ | $x$ | $x+1$ |
| $x$ | $0$ | $x$ | $x+1$ | $1$ |
| $x+1$ | $0$ | $x+1$ | $1$ | $x$ |

*Tabela 2.5: Multiplicação em* $GF(2^2)$

## 2.2 A Transformada de Fourier

### 2.2.1 Multiplicação Polinomial

Multiplicar dois polinômios reais $a(x)$ e $b(x)$ cujo produto possui grau menor do que $n$, multiplicando cada coeficiente de $a$ por um coeficiente de $b$ e somando-os, exige operações aritméticas com uma complexidade $O(n^2)$. Existem outras maneiras de se fazer isto, a que vamos investigar consiste em três passos: resolver um problema de avaliação polinomial obtendo assim dois vetores, multiplicar

estes vetores ponto a ponto resultando em um outro vetor $c$ com o qual pode-se obter o polinômio procurado resolvendo um problema de interpolação. Esta situação se resume na figura abaixo.

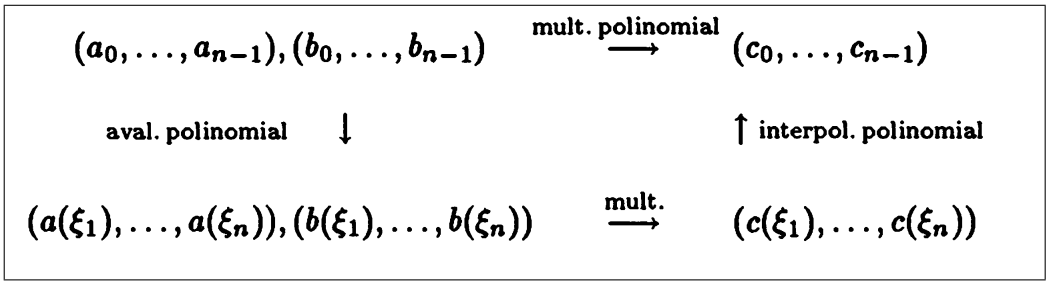


Figura 2.1: Esquema de multiplicação polinomial

### 2.2.2 Álgebras e Morfismos

Uma $\mathbb{C}$ álgebra é um espaço vetorial complexo $A$ munido de uma multiplicação binária $*$ sobre $A$ de modo que $(A, +, *)$ é um anel com 1 e $*$ comuta com a multiplicação por escalar.

Sejam $A$ e $B$ $\mathbb{C}$-álgebras. $\phi : A \to B$ um morfismo. A imagem $\phi(A) \subseteq B$ é uma subálgebra de B, $\mathbb{C}$-subespaço que também é um subanel de B. O núcleo $K = \ker \phi := \phi^{-1}(0)$ de um morfismo $\phi : A \to B$ é um ideal bi-lateral, isto é, se $k \in K$ então $akb \in K \ \forall a, b \in A$. O espaço quociente torna-se uma $\mathbb{C}$-álgebra por meio de $(a + \ker \phi) \cdot (b + \ker \phi) := ab + \ker \phi$. Existe um isomorfismo natural entre esta álgebra quociente $A/\ker \phi$ e a imagem $\phi(A)$.

Sejam $\xi_1, \ldots, \xi_n$ $n$ distintos pontos de interpolação, a transformação de avaliação $\phi : \mathbb{C}[x] \to \mathbb{C}^n$ definida por $a(x) \mapsto (a(\xi_1), \ldots, a(\xi_n))$ é claramente um morfismo de $\mathbb{C}$-álgebras. Em particular, multiplicação polinomial em $\mathbb{C}[x]$ corresponde à multiplicação ponto a ponto em $\mathbb{C}^n$. O núcleo de $\phi$ é o ideal gerado pelo polinômio $\prod_{i \leq n}(x - \xi_i)$ de grau $n$. Assim, temos um isomorfismo $\mathbb{C}[x]/(\prod_{i \leq n}(x - \xi_i)) \simeq \phi(\mathbb{C}[x])$. A álgebra $\mathbb{C}[x]/(\prod_{i \leq n}(x - \xi_i))$ consiste em todos os polinômios com grau menor que

$n$ com multiplicação módulo $\prod_{i\leq n}(x-\xi_i)$. Como este é $n$-dimensional, o mesmo é verdade para $\phi(\mathbb{C}[x])$, então $\phi$ é sobrejetora. Ao todo, vimos que

$$\mathbb{C}[x]/(\prod_{i\leq n}(x-\xi_i)) \simeq \phi(\mathbb{C}[x]).$$

Assim, o esquema de avaliação e interpolação calcula o produto de dois polinômios corretamente se este produto possui grau menor que $n$. Do contrário obtemos o produto módulo $\prod_{i\leq n}(x-\xi_i)$.

### 2.2.3 DFT e FFT no Corpo Complexo

Vamos agora nos concentrar na busca dos pontos de interpolação $\xi_1, \ldots, \xi_n$ que permitem executar o esquema de multiplicação em um tempo de ordem menor que $o(n^2)$ operações. Suponhamos que $\xi_k = \xi_{-j} \neq 0$ para algum $j \neq k$. Com isto podemos armazenar simultaneamente multiplicações quando avaliamos $a(\xi_j)$ e $a(\xi_k) = a(\xi_{-j})$. Assumindo que $n$ é par, podemos separar $a(x)$ em

$$a(\xi_i) = \sum_{k<n/2} a_{2k}(\xi_j^2)^k + \xi_j \sum_{k<n/2} a_{2k+1}(\xi_j^2)^k = e(\xi_j^2) + \xi_j o(\xi_j^2)$$

e

$$a(-\xi_i) = \sum_{k<n/2} a_{2k}(\xi_j^2)^k - \xi_j \sum_{k<n/2} a_{2k+1}(\xi_j^2)^k = e(\xi_j^2) - \xi_j o(\xi_j^2)$$

onde $e(x) := \sum_{k<n/2} a_{2k}x^k$ e $o(x) := \sum_{k<n/2} a_{2k+1}x^k$ são polinômios de grau $n/2$ cujos coeficientes são, respectivamente, os coeficientes de índice par e ímpar de $a(x)$. Esta separação reduz o problema de calcular $a(\xi_j)$ e $a(-\xi_j)$ para a avaliação de dois polinômios $e(x)$ e $o(x)$ de metade do grau nos pontos $\xi_j^2 = \xi_k^2$ seguido por

uma multiplicação, uma adição e uma subtração. Todos os $n$ pontos de interpolação podem ser escolhidos desta forma, isto é, $\xi_1, \ldots, \xi_{n/2}, \xi_{n/2+1} = -\xi_1, \ldots, -\xi_n = -\xi_{n/2}$. Com isto, os valores $a(\xi_1), \ldots, a(\xi_n)$ podem ser calculados a partir dos valores de $e(\xi_1^2), \ldots, e(\xi_{n/2}^2)$ e $o(\xi_1^2), \ldots, o(\xi_{n/2}^2)$ com $3 \cdot n/2$ operações a mais.

Podemos continuar com este procedimento recursivo, assumnido que $n$ é uma potência de 2. No próximo nível da recursão, nossos pontos de interpolação são $\xi_1^2, \ldots, \xi_{n/2}^2$. Como antes, queremos escolher estes pontos de tal forma que $\xi_{n/4+j}^2 = -\xi_j^2$ para todo $1 \leq j \leq n/4$. Isto é impossível sobre os reais, mas pode ser obtido sobre $\mathbb{C}$ definindo $\xi_{n/4+j} := i\xi_j$. Assim, podemos usar o mesmo procedimento como acima para separar cada problema de tamanho $n/2$ em dois subproblemas de tamanho $n/4$ mais $3 \cdot n/4$ operações adicionais.

No próximo passo de recursão, usamos uma raiz quadrada de $i$, um número $z$ para o qual $z^8 = 1$. Para nossos pontos serem distintos, $z$ precisa satisfazer $z^j \neq 1$ para $0 < j < 8$. Desta forma, para terminar a recursão em problemas de grau um, é necessário um número satisfazendo.

$$\omega^n = 1 \text{ e } \omega^j \neq 1$$

para $0 < j < n$. Um $\omega$ desta forma é chamado de uma **n-ésima raiz primitiva da unidade**. Por exemplo, podemos escolher $\omega = e^{2\phi i/n}$. Assim, o procedimento recursivo que descrevemos acima utiliza os distintos pontos $1, \omega, \omega^2, \ldots, \omega^{n-1}$ como pontos de interpolação.

Vamos analizar o custo aritmético deste algoritmo. Se $T(n)$ denota o número de operações aritméticas complexas para avaliar um polinômio $a(x)$ de grau menor que $n$ em $1, \omega, \omega^2, \ldots, \omega^{n-1}$, vimos acima que $T(n) \leq 2T(n/2) + 3n/2$. Obviamente, $T(1) = 0$. Como assumimos que $n$ é uma potência de 2, esta recursão fornece a cota superior

$$T(n) \leq 1,5n \log_2 n.$$

Então, usando este conjunto especial de pontos, podemos avaliar muito mais rápido que no caso geral. Se pudéssemos também interpolar em tempo $(n \log n)$, teriamos um algoritmo $(n \log n)$ para multiplicação polinomial.

A avaliação de $a(x) = \sum_{i<n} a_i x^i$ em $1, \omega, \omega^2, \ldots, \omega^{n-1}$ pode ser escrita como uma multiplicação da matriz abaixo pelo vetor dos coeficientes de $a(x)$:

$$\begin{pmatrix} a(1) \\ a(\omega) \\ a(\omega^2) \\ a(\omega^3) \\ \vdots \\ a(\omega^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Esta transformação linear é chamada a **transformada discreta de Fourier** (DFT) de ordem $n$. A matriz $DFT_n := (\omega^{ij})_{i,j<n}$ é chamada a matriz $DFT$. Esta matriz é essencialmente auto-inversível, tomando os expoentes módulo $n$, podemos escrever $(DFT_n)^2$ como:

$$\begin{pmatrix} \sum 1 & \sum \omega^j & \sum \omega^{2j} & \sum \omega^{3j} & \cdots & \sum \omega^{(n-1)j} \\ \sum \omega^j & \sum \omega^{2j} & \sum \omega^{3j} & \omega^{4j} & \cdots & \sum \omega^{nj} \\ \sum \omega^{2j} & \sum \omega^{3j} & \sum \omega^{4j} & \omega^{5j} & \cdots & \sum \omega^j \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum \omega^{(n-1)j} & \sum \omega^{nj} & \sum \omega^j & \omega^{2j} & \cdots & \sum \omega^{(n-2)j} \end{pmatrix} \tag{2.3}$$

onde os somatórios da matriz são considerados para $0 \leq j < n$.

Como $\omega$ é uma $n$-ésima raiz da unidade, então $1^k = (\omega^n)^k = \omega^{kn} = (\omega^k)^n$, mas em qualquer corpo finito $x^n - 1 = (x-1)\left(\sum_{i=0}^{n-1} x^i\right)$. Juntando estes resultados, podemos escrever

$$0 = \omega^{kn} - 1 = (\omega^k - 1)\sum_{j=0}^{n-1} \omega^{kj}.$$

Mas como $\omega^k \neq 1$ para $0 < k < n$, então o somatório à direita é igual a zero, isto é,

$$\sum_{j=0}^{n-1} \omega^{kj} = 0 \text{ para } 0 < k < n.$$

Usando esta última equação podemos calcular as entradas da matriz $(DFT_n)^n$ em (2.3),

$$(DFT_n)^2 = n \cdot \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \iddots & \vdots \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$$

Em outras palavras, $DFT_n^{-1}$ é a própria $DFT_n$ seguinda de uma multiplicação por $n^{-1}$ e uma permutação de suas entradas. Desta forma a interpolação pode ser efetuada com no máximo $1,5n\log n + n$ operações aritméticas utilizando o procedimento descrito acima. Juntando os passos do nosso esquema de multiplicação, avaliação, multiplicação ponto a ponto e interpolação obtemos um algoritmo para multiplicar polinômios com custo aritmético em no máximo $4,5n\log n + 2n = (n\log n)$.

O algoritmo de avaliação recursivo que descrevemos, capaz de efetuar a multiplicação matriz-vetor em $(n\log n)$ operações é chamado o algoritmo da **trans-**

**formada rápida de Fourier** (FFT) e foi descoberto por Cooley e Tukey [6] em 1965.

### 2.2.4 DFT e FFT em Corpos Finitos

Com base no que vimos sobre $DFT_n$ e o algoritmo $FFT$ definidos sobre o corpo dos números complexos vamos trazer estas idéias para os corpos de interesse neste texto, os corpos finitos. O ponto essencial da discussão para se chegar a $FFT$ foi a existência de uma $n$-ésima raiz primitiva da unidade em $\mathbb{C}$. Desta forma, dado $n$, podemos definir $DFT_n$ e o algoritmo $FFT$ sobre qualquer anel comutativo $R$ com um elemento unidade 1 contendo uma $n$-ésima raiz primitiva $\omega$. Neste sentido, a verificação que fizemos de $DFT_n$ ser invertível continua válida se observarmos ainda que

**(1)** Para $0 < k < n, \omega^k - 1$ não é um divisor de zero em $R$;

**(2)** $n \cdot 1 := 1 + \ldots + 1$ ($n$ vezes) é uma unidade em $R$.

No corpo complexo, a transformada discreta de Fourier de um vetor $v = (v_0, v_1, \ldots, v_{n-1})$ com componentes complexas, é um vetor $V = (V_0, V_1, \ldots, V_{n-1})$, dado por

$$V_k = \sum e^{-j2\phi n^{-1} ik} v_i, \ k = 0, \ldots, n-1$$

onde $j = \sqrt{-1}$. O núcleo de Fourier $e^{-j2\phi/n}$ é a $n$-ésima raiz da unidade no corpo complexo. No corpo $GF(q)$, um elemento $\omega$ de ordem $n$ é a $n$-ésima raiz da unidade. Por uma analogia entre $e^{-j2\phi/n}$ e $\omega$, temos a seguinte definição.

**Definição 2.2.1.** *Seja $v = (v_0, v_1, \ldots, v_{n-1})$ um vetor sobre $GF(q)$, e seja $\omega \in GF(q)$ de ordem $n$. A transformada (discreta) de Fourier do vetor $v$ é o vetor*

$V = (V_0, V_1, \ldots, V_{n-1})$ *com componenetes dadas por*

$$V_j = \sum \omega^{ij} v_i, \ j = 0, \ldots, n-1.$$

Alguns autores chamam $i$ de o índice do **tempo**, tomando valores no **eixo do tempo** $\{0, 1, \ldots, n-1\}$, e $v$ de a **função do domínio-tempo**. Sendo $j$ o índice **frequência**, assumindo valores no **eixo da frequência** $\{0, 1, \ldots, n-1\}$, e $V$ a **função do domínio-frequência** ou **spectro**.

A transformada de Fourier em um corpo de Galois é muito parecida com a transformada de Fourier no corpo complexo com uma importante diferença. No corpo complexo um elemento $\omega$ de ordem $n$ existe para cada valor de $n$. Contudo, no corpo $GF(q)$ um tal $\omega$ existe apenas se $n$ divide $q-1$. Além disso, se para algum valor de $m$, $n$ divide $q^m - 1$, então um vetor com **comprimento de bloco** $n$ possui uma transformada de Fourier no corpo extensão $GF(q^m)$. Por esta razão, um vetor $v$ de comprimento $n$ sobre $GF(q)$ pode também ser tratado como um vetor sobre $GF(q^m)$; este possui uma transformada de Fourier de comprimento $n$ sobre $GF(q^m)$. Isto é completamente análogo à transformada de Fourier de um vetor sobre $\mathbb{R}$, esta transformada possui componentes no corpo complexo.

**Teorema 2.13** (Transformada Inversa de Fourier)**.** *Seja* $v = (v_0, v_1, \ldots, v_{n-1})$ *um vetor sobre* $GF(q)$*. Interpretando* $n$ *como um inteiro do corpo, podemos escrever*

$$v_i = \frac{1}{n} \sum_{j=0}^{n-1} \omega^{-ij} V_j$$

*onde* $V_j$ *são as componentes de* $V$*, a transformada de Fourier de* $v$*.*

**Exemplo 2.2.1.** *Em* $\mathbf{Z}_{13}$*,* $8$ *é uma quarta raiz primitiva da unidade e* $8^{-1} = 5$*. Assim, montamos as matrizes*

$$V(8) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 8 & 12 & 5 \\ 1 & 12 & 1 & 12 \\ 1 & 5 & 12 & 8 \end{pmatrix}$$

*e*

$$V(5) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 12 & 8 \\ 1 & 12 & 1 & 12 \\ 1 & 8 & 12 & 5 \end{pmatrix}$$

*cujo produto resulta em*

$$V(8)V(5) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 8 & 12 & 5 \\ 1 & 12 & 1 & 12 \\ 1 & 5 & 12 & 8 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 12 & 8 \\ 1 & 12 & 1 & 12 \\ 1 & 8 & 12 & 5 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}.$$

*Considere agora*

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.4}$$

*podemos calcular*

$$
\begin{aligned}
c &= V(8)u \\
&= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 8 & 12 & 5 \\ 1 & 12 & 1 & 12 \\ 1 & 5 & 12 & 8 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\end{aligned}
$$

*e podemos obter* $u$ *novamente partindo de* $c$*,*

$$
\begin{aligned}
u &= 4^{-1}[V(5)c] \\
&= 4^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 12 & 8 \\ 1 & 12 & 1 & 12 \\ 1 & 8 & 12 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\
&= 4^{-1} \begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.
\end{aligned}
$$

**Exemplo 2.2.2.** *Sendo* $a(x) = b(x) = x + 1 \in \mathbf{Z}_{13}[x]$ *vamos calcular o produto* $c(x) = a(x)b(x)$. *Para isso vamos considerar os vetores* $a$ *e* $b$ *formados pelos coeficientes dos polinômios* $a(x)$ *e* $b(x)$ *respectivamente, ambos iguais a*

$$\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

*Utilizando a matriz* $V(8)$ *do exemplo 2.2.1 podemos representar o passo de avaliação polinomial da figura 2.1. O vetor* $\tilde{a}$ *resultante da avaliação de* $a(x)$ *em* $8^0, 8^1, 8^2$ *e* $8^3$ *pode ser expresso como*

$$\begin{aligned} \tilde{a} &= V(8)a \\ &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 8 & 12 & 5 \\ 1 & 12 & 1 & 12 \\ 1 & 5 & 12 & 8 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 9 \\ 0 \\ 6 \end{pmatrix} \end{aligned}$$

*de maneira inteiramente análoga definimos o vetor* $\tilde{b}$ *que, convenientemente, é igual ao vetor* $\tilde{a}$.

*O segundo passo do esquema apresentado na figura 2.1 é a multiplicação ponto a ponto das avaliações* $\tilde{a}$ *e* $\tilde{b}$ *que acabamos de calcular. Vamos representar este produto pelo vetor* $\tilde{c}$,

$$\begin{aligned}\tilde{c} &= \begin{pmatrix} 2^2 & \bmod 13 \\ 9^2 & \bmod 13 \\ & 0 \\ 6^2 & \bmod 13 \end{pmatrix} \\ &= \begin{pmatrix} 4 \\ 3 \\ 0 \\ 10 \end{pmatrix}.\end{aligned}$$

*O último passo no esquema de multiplicação é a interpolação das entradas do vetor $\tilde{c}$ para obter o vetor $c$. Isto é feito com o auxílio da matriz $V(5)$ construída no exemplo 2.2.1,*

$$\begin{aligned}c &= 4^{-1}V(5)\tilde{c} \\ &= 10\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 12 & 8 \\ 1 & 12 & 1 & 12 \\ 1 & 8 & 12 & 5 \end{pmatrix}\begin{pmatrix} 4 \\ 3 \\ 0 \\ 10 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 2 \\ 1 \\ 0 \end{pmatrix}.\end{aligned}$$

*Assim,* $c(x) = x^2 + 2x + 1$.

# 3 INTRODUÇÃO À TEORIA DE CÓDIGOS

Neste capítulo apresentamos alguns conceitos fundamentais da Teoria de Códigos, que permitem-nos estudar os problemas de codificação e de decodificação. Também mostramos os chamados **códigos lineares**, que utilizam a estrutura de um espaço vetorial. Os resultados deste capítulo foram retirados e adaptados de [19], [25], [30] e [24].

## 3.1 Conceitos Elementares

Seja $A$ um conjunto finito com $q$ elementos, chamado alfabeto. Os elementos de $A$ são denominamos de letras, caracteres ou dígitos. Uma sequência de $n$ elementos de $A$ é chamada de palavra de comprimento $n$. O conjunto $A^n$ é o conjunto de todas as palavras de comprimento $n$ sobre $A$, isto é,

$$A^n = \{(m_0, m_1, \ldots, m_{n-1}) | m_i \in A, 0 \leq i \leq n-1\}.$$

Um dos objetivos da teoria de códigos é detectar os possíveis erros que podem somar-se à palavra transmitida. Esta detecção é baseada na idéia de semelhança entre palavras, para entendermos melhor esta idéia é necessário aprofundar o conceito de proximidade entre palavras.

Dado $x \in A^n$, denotamos por $C_x$ o conjunto formado pelas componentes não nulas de $x$.

**Definição 3.1.1** (**Norma de Hamming**). *Seja $x \in A^n$, a norma de Hamming de $x$, denotada por $|x|$, é dada por*

$$|x| = \sum_{i=0}^{n-1} \mathcal{C}_{C_x}(x_i) \tag{3.1}$$

*onde $\mathcal{C}_{C_x}$ é a função característica de $C_x$.*

**Exemplo 3.1.1.** *Tomando $A = GF(8)$, considere $c = (0, 0, 1, 1, 0, 1)$. O conjunto das componentes não nulas de $c$ é*

$$C_c = \{1\}$$

*para o qual*

$$\mathcal{C}_{C_c}(c_0) = \mathcal{C}_{C_c}(c_1) = \mathcal{C}_{C_c}(c_4) = 0$$

*e*

$$\mathcal{C}_{C_c}(c_2) \;\; = \;\; \mathcal{C}_{C_c}(c_3) = \mathcal{C}_{C_c}(c_5) = 1$$

*de onde $|c| = \sum_{i=0}^{5} \mathcal{C}_{C_c}(c_i) = 1 + 1 + 1 = 3$.*

Seja $I_n = \{0, 1, ..., n\}$. A função $|\cdot| : GF(q)^n \to I_n$ definida acima é uma norma. De fato, sejam $x, y \in GF(q)^n$ e $\lambda \in GF(q)$ temos

(1) $|x| = 0$ se e somente se $x = 0$. Bem, para que tenha-se $|x| = 0$ é preciso que $x_i \notin C_x$, para $i = 0, ...n - 1$, mas isto só ocorre se tivermos $x = 0$.

(2) $|\lambda x| = |x|$, se $\lambda \neq 0$. Como a função $\lambda(\cdot) : C_x \to C_{\lambda x}$ é uma bijeção, $\#C_x = \#C_{\lambda x}$ e como $\lambda s \neq 0$, se $s \neq 0$. Então, $|\lambda x| = |x|$.

(3) $|x+y| \leq |x| + |y|$. Para cada $z \in C_{x+y}$ existe $a \in C_x$ ou $b \in C_y$, com os quais podemos escrever $z$ na forma $z = a + b$, mas $ab \neq 0$. Então $\#C_{x+y} \leq \#C_x + \#C_y$, logo $|x+y| \leq |x| + |y|$.

A norma de Hamming também é chamada de função **peso** e neste caso costuma-se usar a notação $w(\cdot)$ em vez de $|\cdot|$.

**Definição 3.1.2** (**Distância de Hamming**). *Sejam $c$ e $d \in A^n$. Definimos a* ***distância de Hamming*** *entre as palavras $c$ e $d$ como sendo a quantidade de respectivas entradas diferentes entre $c$ e $d$ e pode ser dada por*

$$\mathrm{d}(c,d) = |c-d|. \tag{3.2}$$

**Exemplo 3.1.2.** *Se $c = (0,1,2)$ e $d = (1,2,0)$ em $GF(3)$, então $\mathrm{d}(c,d) = 2$.*

Para os nossos estudos consideramos que o alfabeto utilizado é um corpo finito $GF(q)$ com $q$ elementos e que as palavras do código formam um subespaço vetorial não trivial de $GF(q)^n$. Neste caso, o código é chamado de código linear.

**Definição 3.1.3.** *A* ***distância mínima*** *de um código $C$ é definida como o número*

$$d(C) = min\{d(x,y) | x, y \in C, x \neq y\}. \tag{3.3}$$

Ou seja, a distância mínima $d(C)$ de $C$ é a menor distância de Hamming entre palavras diferentes do código $C$.

**Teorema 3.1.** *A função distância de Hamming é uma métrica sobre o conjunto $A^n$, isto é, para todos $x$, $y$ e $z$ em $A^n$, temos*

*(1) (positiva definida)*

$$\mathrm{d}(x,y) \geq 0 \text{ e } \mathrm{d}(x,y) = 0 \text{ se e somente se } x = y.$$

*(2) (simétrica)*

$$\mathrm{d}(x, y) = \mathrm{d}(y, x).$$

*(3) (desigualdade triangular)*

$$\mathrm{d}(x, z) \leq \mathrm{d}(x, y) + \mathrm{d}(y, z).$$

**Prova**. (1) é direto propriedade 1 da norma de Hamming. (2) também é imediato da propriedade 2 da norma de Hamming usando $\lambda = -1$. Para provar (3) observe que $|x - z| = |x - y + y - z| \leq |x - y| + |y - z|$.

Como a distância de Hamming é uma métrica, o par $(A^n, d)$ é chamado de espaço métrico [26].

**Definição 3.1.4.** *O peso de um código linear $C$ como sendo o inteiro*

$$\mathrm{w}(C) = min\{\mathrm{w}(c)|c \in C/\{0\}\}.$$

**Proposição 3.1.1.** *Seja $C$ um código linear em $GF(q)^n$ com distância mínima* d.

*(1) Para todo $x$, $y \in C$,* $\mathrm{d}(x, y) = \mathrm{w}(x - y)$.

*(2)* $\mathrm{d} = \mathrm{w}(C)$.

## 3.2 Códigos Lineares

Neste trabalho consideramos os chamados códigos lineares, que são aqueles que formam um espaço vetorial sobre algum corpo finito $GF(q)$. Em particular, um código linear $C$ é dito **cíclico** se para cada palavra $c_1, \ldots, c_n$ em $C$, a palavra $c_n, c_1, \ldots, c_{n-1}$ também está em $C$.

**Definição 3.2.1 (Código linear).** *Um **código linear** é um espaço vetorial sobre um corpo finito.*

Ao longo deste texto são utilizadas três maneiras de representar uma palavra de comprimento $n$

(1) Como um vetor com a ordem das coordenadas invertidas, $(F_{n-1}, \ldots, F_1, F_0)$;

(2) Como uma sequência ordenada, $F_{n-1} \ldots F_1 F_0$;

(3) Como um polinômio de grau $n-1$, $F_{n-1}x^{n-1} + \cdots + F_1 x + F_0$.

onde $F_0, F_1, \ldots, F_{n-1} \in GF(q)$.

Uma função que opera sobre uma palavra $F \in GF(q)^k$ produzindo um resultado com comprimento $m \in \mathbb{Z}$ é chamada de função de codificação.

**Definição 3.2.2** (Codificação)**.** *Seja $f : GF(q)^m \to GF(q)^k$ com $m \leq k$, chamamos $f$ de* ***função de codificação****.*

Associada a cada codificação podemos ter uma função inversa chamada de **função de decodificação**.

**Definição 3.2.3** (Decodificação)**.** *Seja $f^{-1} : GF(q)^k \to GF(q)^m$ com $k \geq m$, chamamos $f^{-1}$ de função de decodificação.*

Estas funções não são o foco das discussões sobre códigos corretores e/ou detectores de erro. Para o discutido neste texto sua importância reside em conhecermos o tamanho de bloco $k$, pois este é o tamanho das palavras sobre as quais o **processador de erro** opera, função responsável por tentar detectar e corrigir erros antes de utilizarmos o decodificador.

**Definição 3.2.4.** *Um processador de erro $P$ para $C$ é uma transformação que aceita uma palavra $v$ de tamanho $n$ e produz um par $(a, u)$ onde $a$ assume dois valores, 'bom' ou 'mau', e $u$ é uma palavra de comprimento $n$. $a$ tem valor 'bom' quando $u$ é uma palavra código, caso acontrário tem valor 'mau'.*

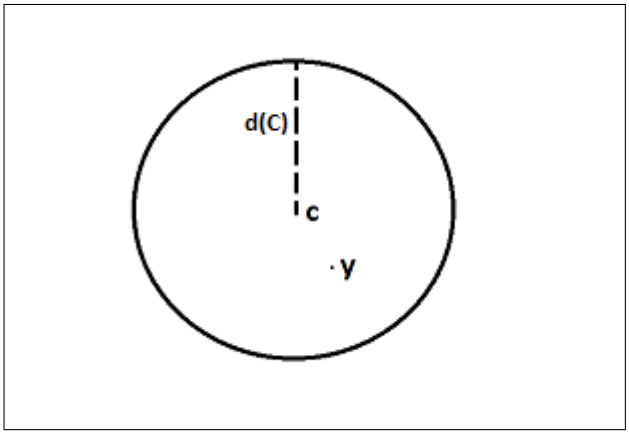


Figura 3.1: Palavras com distância menor do que a distância mínima.

Um processador de erro que nunca modifica a palavra recebida é chamado um **detector de erro** e um processador de erro que sempre retorna uma palavra código é dito **perfeito**.

Como a distância mínima $d(C)$ é a menor distância entre duas palavras distintas do código, modificar uma palavra do código em uma quantidade de posições menor do que $d(C)$ transforma esta em uma palavra fora do código, portanto detectável.

**Proposição 3.2.1.** *Seja $C$ um código. Então é possível a um processador de erro para $C$ detectar todos erros de peso $\leq s$ se e somente se* $\mathrm{d}(C) \geq s+1$.

**Prova**. Dados $c \in C$ e $e \in A^n$ com $w(e) \leq s$, então $y = c + e$ e $d(c, y) = w(e) \leq s$. $y \notin C$ se e somente se como $d(C) > s$.

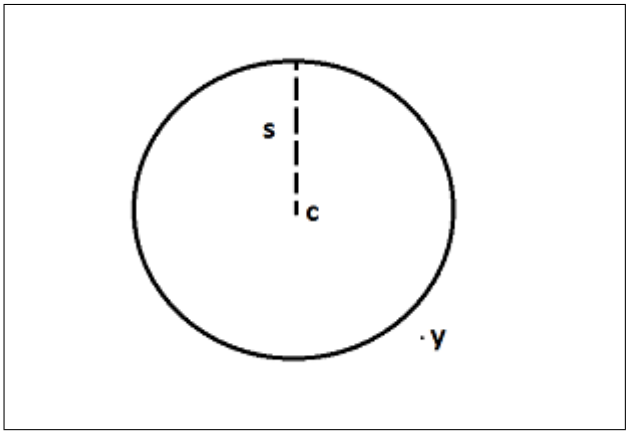


Figura 3.2: Palavras com distância maior do que s.

**Proposição 3.2.2.** *Existe um processador de erro para o código $C$ que corrige todos os erros de peso até $t$ se e somente se $C$ possui distância mínima $2t+1$.*

**Prova**. Suponha que o código possui duas palavras código $u$ e $v$ com distância de no máximo $2t$. Seja $w$ uma palavra que coincide com $u$ em todas as posições em que $u$ e $v$ coincidem. Deixe ainda $w$ e $u$ coincidirem nos primeiros $t$ locais em que $u$ e $v$ não coincidem. Então $d(u,w) \leq t$ e $d(v,w) \leq t$. Agora suponha que $w$ é recebida junto com a informação que no máximo $t$ erros ocorreram. Então ou $u$ ou $v$ pode ter sido transmitida (e possivelmente até mesmo uma palavra código diferente). Não existe maneira de a partir das informação dadas um processador de erro decidir com certeza que uma palavra código foi transmitida. Reciprocamente, suponha que o código possui distância mínima $2t+1$ e uma palavra $w$ é recebida, junto com a informação que um erro de peso no máximo $t$ ocorreu. Se houverem duas palavras $u$ e $v$ afastadas no máximo $t$ de $w$, então, pela desigualdade triangular, $d(u,v) \leq 2t$, contradizendo nossa hipótese. Portanto, existe uma única palavra código $u$ distante no máximo $t$ de $w$ e concluímos que $u$ precisa ter sido transmitida.

### 3.2.1 Representando Códigos Lineares

Os códigos lineares, por sua estrutura de espaço vetorial, permitem-nos usar as ferramentas da Álgebra Linear para representá-los. A seguir apresentamos duas maneiras de representar um código linear.

Seja $\{f_1, f_2, \ldots, f_n\}$ a base canônica de $GF(q)^n$, de modo que todo vetor $v$ deste espaço pode ser escrito como

$$v = v_1 f_1 + v_2 f_2 + \ldots + v_n f_n,$$

onde $v_i \in GF(q)$ para $1 \leq i \leq n$.

Seja $C$ um código linear de dimensão $k$ sobre $GF(q)$. Então, $C$ é isomorfo a $GF(q)^k$ e podemos definir uma aplicação injetora $\nu$ de $GF(q)^k$ em $GF(q)^n$ cuja imagem é exatamente $C$. Para isso, se $\{c_1, c_2, \ldots, c_k\}$ é uma base de $C$ e $\{e_1, e_2, \ldots, e_k\}$ é a base canônica de $GF(q)^k$, basta definirmos $\nu(e_i) = c_i$ para $1 \leq i \leq k$.

Cada elemento $c_j$ da base de $C$ pode ser escrito como

$$c_j = \sum_{i=1}^{n} b_{ij} f_i$$

onde $b_{ij} \in GF(q)$ para $1 \leq j \leq k$. De forma que a matriz $G \in \mathbb{M}_{n \times k}(GF(q))$ formada pelos elementos $b_{ij}$ representa a aplicação linear $\nu$ nas bases canônicas de $GF(q)^k$ e $GF(q)^n$.

Como $C = Im(\nu)$, cada vetor coluna de $G$ pertence ao código $C$, ou seja, $C$ é o subespaço gerado pelas colunas de $G$, que formam uma base para o código. Os elementos de $C$ são todas as palavras $w$ da forma $w = \nu(v)$, para $v \in GF(q)^k$.

**Definição 3.2.5.** *Uma matriz $G \in \mathbb{M}_{n\times k}(GF(q))$ cujas colunas formam uma base para o código $C$ é chamada de* ***matriz geradora*** *do código.*

**Definição 3.2.6 (Código em Bloco).** *Um $(n,k)$-código $C$, também chamado de $(n,k)$-código em bloco, sobre $GF(q)$ é um conjunto de $q^k$ palavras código em $GF(q)^n$.*

Chamamos $n$ de **comprimento de bloco** do código, $k$ de **posto** do código e a fração $k/n$ de **taxa** do código. A notação $(n,k,d)$-código é usada quando se deseja dar atenção a distância mínima do código. Como um código de posto $k$ possui $q^k$ palavras, quando se pretende estudar esta quantidade costuma-se utilizar a notação $(n,M,d)$-código, onde $n$ é o mesmo da definição acima, $M = |C|$ é o número de palavras do código e $d$ é a distância mínima de $C$.

**Proposição 3.2.3.** *Sejam $C$ um $(n,k)$-código linear, $G$ uma matriz $n \times k$. Então $G$ é uma matriz geradora para $C$ se e somente se o posto de $G$ é $k$ e suas colunas são palavras código.*

Outra forma de descrevermos um código linear $C$ é utilizando uma transformação linear sobrejetora $\phi : GF(q)^n \to GF(q)^{n-k}$ tal que $\ker(\phi) = C$. Para isso, consideremos uma base qualquer $\{c_1, c_2, \ldots, c_k\}$ de $C$ e a ampliemos a uma base $\{c_1, c_2, \ldots, c_k, v_1, \ldots, v_{n-k}\}$ de $GF(q)^n$, de modo que um vetor $GF(q)^n$ possa ser escrito como

$$v = \lambda_1 c_1 + \lambda_2 c_2 + \ldots + \lambda_k c_k + \lambda_{k+1} v_1 + \ldots + \lambda_n v_{n-k}$$

onde $\lambda_i \in GF(q)$ para $1 \leq i \leq n$.

Assim, definimos $\phi : GF(q)^n \to GF(q)^{n-k}$ por

$$v \mapsto v' = \lambda_{k+1} v_1 + \ldots + \lambda_n v_{n-k}.$$

A matriz $H \in \mathbb{M}_{(n-k)\times n}(GF(q))$ de posto $n-k$ que representa esta transformação linear nas bases canônicas de $GF(q)^n$ e $GF(q)^k$ é chamada de **matriz de verificação** de $C$

Como $\ker(\phi) = C$, segue que o conjunto de todas as palavras $w$ de $GF(q)^n$ que satisfazem $Hw = 0$ é o código linear $C$.

**Teorema 3.2.** *Seja $C$ um código linear com matriz de verificação $H$. Então $C$ possui distância mínima maior do que $d$ se e somente se nenhum conjunto de $d$ colunas de $H$ é linearmente dependente.*

**Prova**. Suponhamos que $D_{i_1}, D_{i_2}, \ldots, D_{i_d}$ são $d$ colunas linearmente dependentes de $H$. Desta forma, podemos escrever $\sum_{j=1}^{d} \alpha_{i_j} D_{i_j} = 0$ para $\alpha_{i_j}$ constantes não todas nulas. Seja $c \in GF(q)^n$ satisfazendo $c_{i_j} = \alpha_{i_j}$ , para $j = 1, 2, \ldots, d$ e todas as outras coordenadas de $c$ são nulas. Temos que $Hc^T = \sum_{j=1}^{d} \alpha_{i_j} D_{i_j} = 0$. Logo $c \in C$. Além disso, $c \neq 0$ e $w(c) \leq d$, ou seja $d(C) \leq d$.

Por outro lado, se qualquer conjunto de $d$ colunas de $H$ é linearmente independentes, então não existe $c \in C$ não nulo de peso no máximo $d$. De fato, se existisse uma palavra-código $c \neq 0$ com $w(c) \leq d$, então $\sum_{j=1}^{d} c_{i_j} H_{i_j} = 0$ implica que $c_{i_j} = 0$ para todo $j = 1, 2, \ldots, d$. Assim, o único $c \in C$ com $w(c) \leq d$ é $c = 0$.

O que a teoria de códigos busca é construir um bom código, mas o que é um bom código? É um código que permita transmitir uma quantidade alta de informação com uma elevada capacidade de correção de erros. Isto significa procurar por um código com bastante palavras e com uma distância mínima relativamente grande.

Dados $d$, $n$ e $q$ inteiros, na tentativa de se encontrar o maior código de distância mínima $d$ e comprimento de bloco $n$ sobre $GF(q)$, defini-se o número

$$A_q(n, d) = \max\{M | \exists (n, M, d)\text{-código} \subset GF(q)^n\}$$

Com este número formamos a idéia de código **ótimo**. Um código é chamado assim quando este possui tamanho máximo entre os códigos de $GF(q)^n$ com distância mínima $d$.

**Definição 3.2.7.** *(n,M,d)-código se diz ótimo quando* $M = A_q(n, d)$

Não se conhece uma fórmula algébrica que determine este número. No entanto, existem diversas cotas para $A_q(n, d)$, uma destas é a chamada **Cota de Singleton**.

**Teorema 3.3** (**Cota de Singleton**)**.**

$$A_q(n, d) \leq q^{n-d+1} \tag{3.4}$$

**Prova**. Seja $C$ um $(n, M, d) -$ código ótimo. Afirmamos que se $a$, $b \in C$ e $a \neq b$, então as palavras $a'$ e $b'$ que resultam destas anulando as últimas $d - 1$ posições também devem satisfazer $a' \neq b'$. Se $a' = b'$, $a$ e $b$ podem diferir apenas nas $d - 1$ posições descartadas, mas isto significa que $d(a, b) \leq d - 1$, contradição. Seja $C'$ o código de comprimento $n - d + 1$ que resulta de $C$ por anulação das últimas $d - 1$ posições de cada palavra. O argumento acima mostra que $|C'| = |C|$. Como $C' \subset GF(q)^{n-d+1}$ temos imediatamente que

$$A_q(n, d) \leq q^{n-d+1} \tag{3.5}$$

**Exemplo 3.2.1.** *Para* $A_q(4, 3)$ *a cota de Singleton é*

$$A_q(4, 3) \leq q^2 \tag{3.6}$$

**Exemplo 3.2.2.** *Seja* $C$ *o código sobre* $GF(2)$ *cuja matriz de verificação é*

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{3.7}$$

*Duas linhas de $H$ são linearmente dependentes se e somente se são iguais. Assim, vemos que quaisquer duas linhas de $H$ são linearmente independentes e a terceira é a soma das duas primeiras e concluímos que $C$ possui distância mínima $d = 3$. Pelas dimensões de $H$ é claro que $n = 6$. Logo, $A_2(6,3) \leq 2^{6-3+1} = 2^4$.*

### 3.2.2 Decodificação de Códigos Lineares

Consideremos agora o problema de decodificar a palavra recebida $y$. Uma maneira simples de fazer isto é comparando a distância da palavra recebida $y$ a cada palavra $x$ do código e escolher como valor da decodificação o $x$ de menor distância à $y$. Para um $(n,k)$-código linear sobre $GF(q)$ isto exige $q^k$ comparações, o que, para $k$ grande, é impraticável.

Sejam $0 = c^{(1)}, c^{(2)}, \ldots, c^{(q^k)}$ as $q^k$ palavras do código $C$. Observamos que $C$ é um ideal de $GF(q)^n$. O anel quociente $GF(q)^n/C$ é formado pelas classes laterais $a + C = \{a + c | c \in C\}$, onde $a \in GF(q)^n$. Desta forma particionamos $GF(q)^n$ em $q^{n-m}$ classes laterais

$$GF(q)^n = (a^{(0)} + C) \uplus (a^{(1)} + C) \uplus \cdots \uplus (a^{(q^{n-m}-1)} + C) \tag{3.8}$$

tendo cada classe lateral um número de $|C| = q^k$ elementos. Sejam $c, y \in GF(q)^n$ as palavras transmitida e recebida respctivamente, digamos $y = a^{(i)} + c^{(j)}$ com

$0 \leq i \leq q^{n-m} - 1$ e $1 \leq j \leq q^k$. O erro $e = y - c = a^{(i)} + c^{(j)} - c$ pertence à mesma classe lateral que a mensagem recebida, pois o espaço vetorial $C$ é fechado em relação à adição.

**Definição 3.2.8.** *Sejam $H$ uma matriz de verificação de um $(n, k)$-código linear $C$ e $y \in GF(q)^n$. O vetor $S(y) = Hy^T \in GF(q)^{n-k}$ é chamado de* ***síndrome*** *de $y$.*

**Teorema 3.4.** *Para $y, z \in GF(q)^n$, temos que*

*(1) $S(y) = 0$ se e somente se $y \in C$,*

*(2) $S(y) = S(z)$ se e somente se $y + C = z + C$.*

**Prova**. A afirmação (1) segue da definição de síndrome. Observe que $S(y) = S(z)$ se e somente se $H(y - z)^T = 0$, então $y - z \in C$ e $y + C = z + C$.

**Definição 3.2.9.** *Seja $C$ um $(n, k)$-código linear sobre $GF(q)$. Um elemento de peso mínimo numa classe lateral $a + C \in GF(q)^n/C$ é chamado um representante da classe lateral. Se há mais de um vetor com peso mínimo na classe, escolhemos qualquer um deles para ser o representante.*

Suponhamos que os elementos $a^{(1)}, \ldots, a^{(q^{n-k}-1)}$ são os representantes das classes laterais, podemos escrever os elementos de $GF(q)^n$, distribuídos nas classes laterais, usando a representação vetorial

| $00\ldots0$ | $00\ldots01$ | $\ldots$ | $(q-1)\ldots(q-1)$ |
|---|---|---|---|
| $c^{(1)}$ | $c^{(2)}$ | $\ldots$ | $c^{(q^k)}$ |
| $a^{(1)} + c^{(1)}$ | $a^{(1)} + c^{(2)}$ | $\ldots$ | $a^{(1)} + c^{(q^k)}$ |
| $a^{(2)} + c^{(1)}$ | $a^{(2)} + c^{(2)}$ | $\ldots$ | $a^{(2)} + c^{(q^k)}$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $a^{(s)} + c^{(1)}$ | $a^{(s)} + c^{(2)}$ | $\ldots$ | $a^{(s)} + c^{(q^k)}$ |

Como o erro é o representante da classe lateral de $y$, temos que $e = a^{(i)}$ e a mensagem transmitida é

$$x = y - e = a^{(i)} + c^{(j)} - a^{(i)} = c^{(j)}.$$

A noção de síndrome pode ser usada para resolver o problema de encontrarmos um representante para uma classe lateral de $GF(q)^n/C$. Ao recebermos a mensagem $y$, calculamos a síndrome $S(y)$ e encontramos os representantes de $a^{(i)}$ com síndrome igual a $S(y)$. Então decodificamos $y$ através de $x = y - a^{(i)}$, onde $x$ é a palavra-código que está a uma distância de $y$ igual à distância mínima do código.

**Exemplo 3.2.3.** *Seja $C(4,2)$ um código binário linear com matriz geradora*

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

*e matriz de verificação*

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

*A seguinte tabela contém os elementos de $GF(2^4)$ particionados em classe conforme o código $C$.*

| 00 | 01 | 10 | 11 |
|---|---|---|---|
| 0000 | 0110 | 1011 | 1101 |
| 0001 | 0111 | 1010 | 1100 |
| 0010 | 0100 | 1001 | 1111 |
| 1000 | 1110 | 0011 | 0101 |

*Usando a matriz geradora G podemos calcular as palavras c do código. Os elementos das classes laterais são obtidos tomando o representante de cada classe lateral e adicionando a este cada palavra do código. Se a palavra recebida é* $y = 0101$*, tomamos o representante da classe como sendo o erro e a mensagem transmitida é*

$$x = y - e = 0101 - 1000 = 1101.$$

*A palavra transmitida é* 1101.

### 3.2.3 Códigos Cíclicos

Um código linear $C$ é chamado de cíclico quando $(c_{n-1}, c_0, \ldots, c_{n-2}) \in C$ sempre que $(c_0, \ldots, c_{n-1}) \in C$.

Seja $A = GF(q)[x]/(x^n - 1)$ o anel quociente de $GF(q)[x]$ pelo ideal gerado por $x^n - 1$. Vamos identificar $GF(q)^n$ com $A$ identificando $a = (a_0, \ldots, a_{n-1})$ com $a_0 + \ldots + a_{n-1}x^{n-1} = a(x)$

Cada elemento de $A$ pode ser representado de maneira única por um polinômio de grau menor que $n$. Assim, identificamos $a = (a_0, ..., a_{n-1})$ com $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. Note que se $b = (a_{n-1}, a_0, ..., a_{n-2})$, então $b(x) \equiv xa(x) \mod (x^n - 1)$. De fato $xa(x) \equiv a_{n-1} + a_0x + \cdots + a_{n-2}x^{n-1} \mod (x^n - 1)$. Então os códigos cíclicos podem ser caracterizados como os subespaços $C$ de $A$ tais que $c(x) \in C \Rightarrow xc(x) \in C$.

Observe ainda que, para um código cíclico $C$, se $c(x) \in C$ então $xc(x) \in C$, $x^2c(x) \in C$ e assim por diante, logo $b(x)a(x) \in C$ para todo $b(x) \in A$. Isto mostra que $C$ é um ideal de $A$. Reciprocamente se $C$ é um ideal de $A$ então $C$ é um subespaço de $A$ e $xc(x) \in C$ sempre que $c(x) \in C$, logo $C$ é um código cíclico. Ou seja, os códigos cíclicos são os ideais de $A$. Os ideais de A podem ser caracterizados pela seguinte proposição

**Proposição 3.2.4.** *Todo ideal de $A$ é da forma $(g)$ onde $g$ é um divisor mônico de $x^n - 1$. Além disso tal $g$ é unicamente determinado pelo ideal.*

**Prova**. Seja $\phi : GF(q)[x] \to A$ a aplicação natural. Se $C$ é um ideal de $A$, então $\phi^{-1}(C)$ é um ideal de $GF(q)[x]$. Como $GF(q)[x]$ é euclideano, $\phi^{-1}(C)$ é principal, $\phi^{-1}(C) = (h)$. Logo, $C$ é gerado por $h$ em $A$. Escrevemos $h = fg$ onde $g$ é um divisor mônico de $x^n - 1$ e $(f, x^n - 1) = 1$, isto é, $g$ é o $MDC$ de $h$ e $x^n - 1$. Mostraremos que $C = (g)$. Como $f$ e $x^n - 1$ são coprimos, existe $\tilde{f}$ tal que $f\tilde{f} \equiv 1 \mod (x^n - 1)$. Seja $c \in C$, sabemos que $c = ah = afg$, logo $c \in (g)$, isto é, $C \subseteq (g)$. Se mostrarmos que $g \in (h)$ então teremos $C = (h) \supseteq (g)$ e consequentemente $C = (g)$. De fato, temos que $g \in (h)$, pois $\tilde{f}h = \tilde{f}fg \equiv g \mod (x^n - 1)$, logo $g = \tilde{f}h$ em $A$.

Para a unicidade suponha que $(g) = (g')$, então $g' = fg$ e $g = f'g'$, logo $ff' = 1$ e $f$ é invertível em $A$. Isso implica que $f$ é coprimo com $x^n - 1$. Como $g'$ divide $x^n - 1$, devemos ter $f \in GF(q)$ e como $g$ e $g'$ são mônicos, $f = 1$, logo $g = g'$.

Se $C = (g)$ é um código cíclico, então $g$ é chamado o **polinômio gerador** de $C$ e o polinômio $h = x^n - 1/g$ é chamado de **polinômio verificador** de $C$. O nome verificador vem do fato que $c(x) \in C$ se e só se $h(x)c(x) \equiv 0 \mod (x^n - 1)$.

Lembramos agora que se $a(x) \in A$, sendo $a(x) = a_0 + \ldots + a_{n-1}x^{n-1}$, então existem polinômios $b(x)$ e $r(x)$ tais que $a(x) = b(x)g(x) + r(x)$, *grau* $r(x) \leq$ *grau* $g(x)$. Mais ainda, *grau* $b(x) <$ $n -$ *grau* $g(x)$. Seja $d$ o grau de $g$. Então, para todo elemento $a(x) \in C$, devemos ter $r(x) = 0$ , logo $a(x) = b(x)g(x)$ onde *grau* $b(x) < n - d$. Daí segue-se que $\dim C = n - d$.

Vemos também que $g(x), xg(x), ..., x^{n-d-1}g(x)$ é uma base de $C$ sobre $GF(q)$, logo, se $g = g_0 + g_1x + \cdots + g_dx^d$, então

$$G = \begin{bmatrix} g_0 & g_1 & \dots & g_d & 0 & \dots & 0 \\ 0 & g_0 & \dots & g_d & 0 & \dots & 0 \\ 0 & 0 & g_0 & \dots & g_d & \dots & 0 \\ \vdots & \vdots & \vdots & g_0 & 0 & \dots & \vdots \\ 0 & \dots & 0 & \dots & g_0 & \dots & g_d \end{bmatrix} \tag{3.9}$$

é uma matriz geradora de $G$.

**Lema 3.2.1.** *Seja $C$ um código cíclico sobre $GF(q)$ de comprimento $n$ com polinômio gerador $g(x)$. Suponha que $\alpha_1, \dots, \alpha_r$ são todas as raízes de $g(x)$ e este não possui raízes repetidas. Então um elemento $c(x) \in GF(q)[x]/(x^n - 1)$ é uma palavra código de $C$ se e somente se $c(\alpha_i) = 0$ para todo $i = 1, \dots, r$.*

**Prova**. Se $c(x)$ é uma palavra código de $C$, então existe $q(x) \in GF(q)[x]$ tal que $c(x) = q(x)g(x)$. Assim, $c(\alpha_i) = q(\alpha_i)g(\alpha_i) = 0$ para todo $i = 1, \dots, r$. Agora, se $c(\alpha_i) = 0$ para todo $i = 1, \dots, r$, então $g(x)|c(x)$, pois $g(x)$ não tem raízes multiplas. Portanto, $c(x) \in C$.

# 4 CÓDIGOS DE REED-SOLOMON

Os códigos de Reed-Solomon formam uma das famílias de códigos mais utilizadas na trasmissão de dados, com usos em telefônia, discos compactos, transmissões de satélites e outros. Um dos principais motivos para estes códigos serem tão empregados é a capacidade que estes possuem de corrigir erros agrupados, as chamadas rajadas de erros. Antes dos códigos RS receberem o nome de seus autores eram conhecidos como códigos polinomiais em $p^m$ símbolos. Este nome já indicava uma das principais características destes códigos: trabalhar com corpos base diferentes de $GF(2)$, o que na prática significa corrigir grupos de bits, grupos de erro.

Outra propriedade notável destes códigos é que eles são códigos $MDS$ (da expressão em inglês *Maximum Distance Separable*). Estes códigos conseguem atingir a mais alta possível distância mínima.

Estas e outras características são explicadas e identificadas pelos parâmetros do código, este capítulo é voltado para entender estes parâmetros e apresentar o processo de codificação que utilizamos ao longo deste texto, a codificação não sistemática dos códigos RS.

## 4.1 Códigos RS

Os códigos de Reed-Solomon foram originalmente apresentados como a imagem de uma função polinomial.

Em 1961 Gorenstein e Zierler [11] publicaram um resultado importante para a teoria de códigos. Mostraram que a formulação original apresentada por Reed e Solomon é **equivalente** a um caso particular de uma certa família de códigos cíclicos e não binários. A demonstração desta equivalência pode ser encontrada em [22].

### 4.1.1 A Formulação Original

**Definição 4.1.1.** *Seja $\alpha$ um elemento primitivo em $GF(q^m)$ e seja $n = q^m$. A palavra código de uma palavra mensagem $F = (f_0, f_1, \ldots, f_{k-1}) \in GF(q^m)^k$, cujo polinômio associado é $F(x) = f_0 + f_1 x + \cdots + f_{k-1}x^{k-1} \in GF(q^m)[x]$, é definida pela transformação $\rho : F(x) \mapsto c$ por*

$$(c_0, c_1, c_2, \ldots, c_{n-1}) = (F(0), F(1), F(\alpha), \ldots, F(\alpha^{n-2})). \tag{4.1}$$

Desta forma, o código de Reed-Solomon de comprimento $n$ e dimensão $k$ sobre $GF(q^m)$ é a imagem, através de $\rho$, de todos os polinômios em $GF(q^m)[x]$ de grau menor ou igual a $k-1$. Resultando num total de $n^k$ palavras código.

Como a soma de dois polinômios em $GF(q^m)[x]$ de grau menor ou igual a $k-1$ é também um polinômio em $GF(q^m)[x]$ e de grau menor ou igual a $k-1$, então o código descrito acima é linear.

Nota-se que 4.1 é de fato uma transformada de Fourier em corpos de Galois.

**Exemplo 4.1.1.** *Considere o código RS sobre $GF(2^3)$ usando o polinômio irredutível $m(x) = x^3 + x + 1$. Sejam $n = 8$ e $k = 4$. Considere a palavra mensagem $F = (\alpha, \alpha^2, \alpha^5, \alpha^4)$ onde $\alpha$ é uma raiz de $m(x)$. O polinômio da palavra código $c$ é*

$$c(x) = \alpha + \alpha^5 x + x^2 + \alpha^3 x^3 + \alpha^4 x^4 + \alpha^4 x^5 + \alpha^4 x^6 + x^7.$$

Observe que no exemplo acima a palavra mensagem não aparece na palavra código, isto é, esta codificação é não sistemátia.

## 4.2 Propriedades dos Códigos RS

**Definição 4.2.1.** *Dados r polinômios não nulos* $f_1(x), ..., f_r(x) \in GF(q)[x]$*, o* ***mínimo múltiplo comum*** *de* $f_1(x), ..., f_r(x)$ *é o polinômio mônico de menor grau que é um múltiplo de todos estes r polinômios, denotado por* $mmc(f_1(x), ..., f_r(x))$*.*

Pode-se mostrar que $mmc(mmc(f_1(x), ..., f_{r-1}(x)), f_r(x)) = mmc(f_1(x), ..., f_r(x))$ [14].

Se $f_1(x), ..., f_r(x) \in GF(q)[x]$ fatorados da seguinte maneira

$$\begin{aligned} f_1(x) &= a_1(x)p_1(x)^{e_{1,1}} \dots p_n(x)^{e_{1,n}} \\ &\vdots \\ f_r(x) &= a_r(x)p_1(x)^{e_{r,1}} \dots p_n(x)^{e_{r,n}} \end{aligned}$$

com $a_1, \dots, a_r \in GF(q)$, $e_{i,j} \geq 0$ e $p_i(x)$ são distintos polinômios mônicos irredutíveis sobre $GF(q)$, então

$$mmc(f_1(x), ..., f_r(x)) = p_1(x)^{\max\{e_{1,1},...,e_{r,1}\}} \dots p_r(x)^{\max\{e_{1,n},...,e_{r,n}\}}$$

**Lema 4.2.1.** *Sejam* $f(x), f_1(x), ..., f_r(x)$ *polinômios sobre* $GF(q)$*. Se* $f_i(x)|f(x)$*, para cada* $i = 1, \dots, r$*, então* $mmc(f_1(x), ..., f_r(x))|f(x)$*.*

**Prova**. Seja $m(x) = mmc(f_1(x), ..., f_r(x))$. Dividindo $f(x)$ por $m(x)$, pelo algoritmo da divisão, existem dois polinômios $q(x)$, $r(x) \in GF(q)[x]$ tais que

$$f(x) = q(x)m(x) + r(x), \text{ com } grau\ (r(x)) < grau\ (g(x)).$$

Portanto, $r(x) = f(x) - q(x)m(x)$, e portanto $r(x)$ é divísivel por todos $f_i(x)$. Como $m(x)$ possui o menor grau, $r(x) = 0$.

**Definição 4.2.2 (BCH).** *Sejam $\alpha$ um elemento primitivo de $GF(q^m)$ e $M^{(i)}(x)$ o polinômio minimal de $\alpha^i$ com relação a $GF(q)$. Um código BCH* ***primitivo*** *sobre $GF(q)$ que possui comprimento $n = q^m - 1$ e* ***distância projetada*** *$\delta$ é um código cíclico sobre $GF(q)$ com polinômio gerador $g(x) = mmc(M^{(a)}(x), M^{(a+1)}(x), \ldots, M^{(a+\delta-2)}(x))$ para algum inteiro $a$. Além disso, quando $a = 1$ o código é chamado de* ***sentido estrito****.*

**Exemplo 4.2.1.** *Seja $\alpha \in GF(8)$ raiz de $1 + x + x^3$, este é um elemento primitivo de $GF(8)$ e para este temos $M^{(1)}(x) = M^{(2)}(x) = 1 + x + x^3$. Assim, um código BCH binário de sentido estrito, com comprimento 7 e com $\delta = 3$ é gerado por $mmc(M^{(1)}(x), M^{(2)}) = 1 + x + x^3$ e é um $[7, 4]$-código.*

**Teorema 4.1.** *Um código BCH com distância projetada $\delta$ possui distância mínima de pelo menos $\delta$.*

**Prova**. Sejam $\alpha$ um elemento primitivo de $GF(q^m)$ e seja $C$ um código $BCH$ gerado por $g(x) = mmc(M^a(x), M^{a+1}(x), \ldots, M^{a+\delta-2}(x))$. Está claro que os elementos $\alpha^a, \ldots, \alpha^{a+\delta-2}$ são raízes de $g(x)$.

Suponha que $d(C) < \delta$. Então, existe uma palavra código não nula $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ tal que $w(c(x)) = d < \delta$. Pelo lema 3.2.1, $c(\alpha^i) = 0$, para cada $i = a, \ldots, a + \delta - 2$, que na forma matricial pode ser escrito como

$$\begin{bmatrix} 1 & \alpha^a & \ldots & (\alpha^a)^{n-1} \\ 1 & \alpha^{a+1} & \ldots & (\alpha^{a+1})^{n-1} \\ \vdots & \vdots & \ldots & \vdots \\ 1 & \alpha^{a+\delta-2} & \ldots & (\alpha^{a+\delta-2})^{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = 0. \tag{4.2}$$

Assumindo que $c(x) \neq 0$ se e somente $j \in R = \{1, \ldots, i_d\}$. Então

$$\begin{bmatrix} (\alpha^a)^{i_1} & (\alpha^a)^{i_2} & \dots & (\alpha^a)^{i_d} \\ (\alpha^{a+1})^{i_1} & (\alpha^{a+1})^{i_2} & \dots & (\alpha^{a+1})^{i_d} \\ \vdots & \vdots & \dots & \vdots \\ (\alpha^{a+\delta-2})^{i_1} & (\alpha^{a+\delta-2})^{i_2} & \dots & (\alpha^{a+\delta-2})^{i_d} \end{bmatrix} \begin{bmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_d} \end{bmatrix} = 0. \tag{4.3}$$

Como $d < \delta$, podemos tomar as primeiras $d$ equações do sistema acima e formar o seguinte sistema

$$\begin{bmatrix} (\alpha^a)^{i_1} & (\alpha^a)^{i_2} & \dots & (\alpha^a)^{i_d} \\ (\alpha^{a+1})^{i_1} & (\alpha^{a+1})^{i_2} & \dots & (\alpha^{a+1})^{i_d} \\ \vdots & \vdots & \dots & \vdots \\ (\alpha^{a+d-1})^{i_1} & (\alpha^{a+d-1})^{i_2} & \dots & (\alpha^{a+d-1})^{i_d} \end{bmatrix} \begin{bmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_d} \end{bmatrix} = 0. \tag{4.4}$$

Podemos escrever o determinante da matriz acima como

$$\prod_{j=1}^{d} (\alpha^a)^{i_j} det \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \dots & \alpha^{i_d} \\ \vdots & \vdots & \dots & \vdots \\ (\alpha^{d-1})^{i_1} & (\alpha^{d-1})^{i_2} & \dots & (\alpha^{d-1})^{i_d} \end{bmatrix}$$
$$= \prod_{j=1}^{d} (\alpha^a)^{i_j} \prod_{k>l} (\alpha^{i_k} - \alpha^{i_l}) \neq 0. \tag{4.5}$$

De onde obtemos que $(c_{i_1}, \dots, c_{i_d}) = 0$. Contradição.

Seja $C$ um código $BCH$ sobre $GF(q)$ de comprimento $q^m - 1$ gerado por $g(x) = mmc(M^{(a)}(x), M^{(a+1)}(x), \dots, M^{(a+\delta-2)}(x))$ onde $M^{(i)}(x)$ é o polinômio minimal de $\alpha^i$ com relação a $GF(q)$ para um elemento primitivo $\alpha$ de $GF(q^m)$. Se $m = 1$, o código tem comprimento $q - 1$. Neste caso, $\alpha$ é um elemento primitivo de $GF(q)$ e $M^{(i)}(x) = x - \alpha^i$. Assim, para $\delta < q$ e como $\alpha^a, \alpha^{a+1}, \dots, \alpha^{a+\delta-2}$ são dois a dois distintos, o polinômio gerador torna-se

$$\begin{aligned} g(x) &= mmc(x-\alpha^a, x-\alpha^{a+1}, \ldots, x-\alpha^{a+\delta-2}) \\ &= (x-\alpha^a)(x-\alpha^{a+1})\ldots(x-\alpha^{a+\delta-2}). \end{aligned}$$

**Definição 4.2.3 (Código RS).** *Um código de Reed Solomon (RS) é um código BCH sobre $GF(q)$ de comprimento $q-1$ gerado por*

$$g(x) = (x-\alpha^{a+1})(x-\alpha^{a+2})\ldots(x-\alpha^{a+\delta-1}).$$

*com $a \geq 0$ e $2 \leq \delta \leq q-1$, onde $\alpha$ é um elemento primitivo de $GF(q)$.*

**Exemplo 4.2.2.** *Um código RS sobre $GF(2^3)$ de comprimento 7 com polinômio gerador $g(x) = (x-\alpha)(x-\alpha^2) = 1+\alpha+(\alpha+\alpha^2)x+x^2$, onde $\alpha$ é raiz de $1+x+x^3 \in GF(2)[x]$. Este é um $[7,5]$-código sobre $GF(2^3)$.*

### 4.2.1 RS e MDS

Como vimos no teorema 3.2.1, o número máximo de palavras em um código de comprimento de bloco $n$ e distância mínima $d$ sobre $GF(q)$ é

$$A_q(n,d) \leq q^{n-d+1}. \tag{4.6}$$

Para um código $C$ com dimensão $m$ a quantidade de palavras no código é $q^m$. Logo $q^m \leq q^{n-d+1}$, de onde vem que

$$m \leq n-d+1. \tag{4.7}$$

Portanto, dados o comprimento de bloco e a distância mínima temos uma cota para a dimensão do código.

**Definição 4.2.4 (MDS).** *Um código é dito separável pela distância máxima, ou MDS, se*

$$m = n - d + 1. \tag{4.8}$$

**Exemplo 4.2.3.** *Seja $C$ o código sobre $GF(2)$ cuja matriz de verificação é*

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.9}$$

*Como vimos no exemplo 3.2.2 $C$ possui distância mínima $d = 3$, comprimento de bloco $n = 6$ e dimensão $m = 4$. Logo, $n - d + 1 = 6 - 3 + 1 = 4 = m$. Portanto, $C$ é $MDS$.*

**Teorema 4.2 (MDS).** *Códigos de Reed-Solomon são $MDS$. Isto é, um código de Reed-Solomon de comprimento $q - 1$ gerado por $g(x) = \prod_{i=a+1}^{a+\delta-1}(x - \alpha^i)$ é um $[q - 1, q - \delta]$-código com distância mínima igual a $\delta$ para qualquer $2 \leq \delta \leq q - 1$.*

**Prova**. Como o grau de $g(x)$ é $\delta - 1$, a dimensão do código é $k = q - 1 - (\delta - 1) = q - \delta$. Pelo Teorema 4.1 $d(C) \geq \delta$. Por outro lado, pela cota de Singleton, $d(C) \leq (q - 1) + 1 - k = \delta$.

Códigos de Reed-Solomon são $MDS$ e consequentemente têm parâmetros muito bons. Entretanto, códigos $RS$ são não binários, enquanto aplicações práticas exigem códigos binários. Na prática, técnicas de concatenação são utilizadas para se produzir códigos binários a partir de códigos $RS$ sobre extensões de $GF(2)$.

O resultado que permite isto é o chamado **Teorema dos Códigos Concatenados** apresentado abaixo e cuja demonstração pode ser encontrada em [14].

**Teorema 4.3 (Códigos Concatenados).** *Seja $A$ um $[N, K, D]$-código sobre $GF(q^m)$. Então, existe um $[nN, mK, d']$-código $C$ sobre $GF(q)$ com $d' = d(C) \geq dD$, contanto que exista um $[n, m, d]$-código $B$ sobre $GF(q)$. Além disso, um $[nN, mK, dD]$-código sobre $GF(q)$ pode ser obtido.*

Seja $C$ um código $[n, k]$-*RS* sobre $GF(2^m)$, onde $n = 2^m - 1$. Aplicando o teorema 4.3, concatenamos $C$ com o código trivial $GF(2^m)$.

Seja $\alpha_1, \ldots, \alpha_m$ uma $GF(2)$-base de $GF(2^m)$ e considere a transformação $\phi : GF(2^m) \to GF(2)^m$

$$\sum_{i=1}^{m} u_i \alpha_i \mapsto (u_1, \ldots, u_m).$$

Então, pelo Teorema 4.3, temos o seguinte resultado.

**Teorema 4.4.** *Seja $C$ um código $[n, k]$-RS sobre $GF(2^m)$, onde $n = 2^m - 1$. Então*

$$\phi^*(C) = \{(\phi(c_0), \ldots, \phi(c_{n-1}))|(c_0, \ldots, c_{n-1}) \in C\}$$

*é um $[mn, mk]$-código binário com distância mínima de pelo menos $n - k + 1$.*

### 4.2.2 RS e Rajadas de Erros

Em um primeiro momento havia a preocupação de obter-se códigos capazes de corrigir erros aleatórios, no entanto, alguns sistemas de armazenamento e certos canais de comunicação são afetados por erros em pequenos intervalos ao invés de erros aleatórios. Estes erros são chamados de **rajadas de erros**.

**Definição 4.2.5.** *Uma rajada de comprimento $l > 1$ é um vetor binário cujas componentes não nulas estão confinadas a $l$ posições consecutivas. Sendo a primeira e a última diferentes de zero.*

Para entender a capacidade de correção de erros em rajada pense em um código $RS(255, 247)$ com $m = 8$. Neste código os símbolos estão agrupados em blocos de 8 bits cada e podem ser corrigidos até 4 símbolos, mas como o código não é binário, os códigos $RS$ acabam tendo vantagem sobre outros códigos para corrigir este tipo de erro, pois o símbolo todo de 8 bits é corrigido.

**Teorema 4.5.** *Um código linear corrige todas as rajadas de comprimento $s$ ou menos se e somente se todas as rajadas de erros de comprimento $s$ ou menor estão em classes laterais distintas.*

**Prova**. Se todas as rajadas de comprimento $s$ ou menores estão em classes laterais distintas, cada rajada é determinada por sua síndrome. Erro pode então ser corrigido por meio de síndromes. Por outro lado, se duas rajadas $r_1$ e $r_2$ de comprimento menor ou igual a $s$ estão na mesma classe lateral, a diferença $c = r_1 - r_2$ é uma palavra código. Assim, se $r_1$ é recebida, $r_1$ pode ser decodificada para ambos $c$ e 0.

**Teorema 4.6.** *Seja $C \subseteq GF(q)^n$ um código com distância mínima $d$. Um elemento $x \in GF(q)^n$ é o único líder da classe lateral $x + C$ se* $\mathrm{w}(x) \leq \lfloor \frac{d-1}{2} \rfloor$.

**Teorema 4.7.** *Seja $C$ um $[2^m - 1, k]$-RS código sobre $GF(2^m)$. Então, o código $\phi^*(C)$ pode corrigir $\lfloor (n-k)/2 \rfloor - m + 1$ rajadas de erros, onde $n = 2^m - 1$ é o comprimento do código.*

**Prova**. Seja $s = m\lfloor (n-k)/2 \rfloor - m + 1$. Pelo Teorema 4.5 basta mostrarmos que todas as rajadas de erros de comprimento $s$ ou menor estão em classes laterais distintas.

Sejam $e_1$, $e_2 \in GF(2)^{mn}$ duas rajadas de erros de comprimento $s$ ou menos que estão na mesma classe lateral de $\phi^*(C)$. Seja, ainda, $c_i$ tal que $\phi^*(c_i) = e_i$ para $i = 1, 2$. Então, como $C$ é $MDS$, para $i = 1, 2$,

$$\begin{aligned} w(c_i) &\leq \left\lceil \frac{s-1}{m} + 1 \right\rceil \\ &= \left\lfloor \frac{n-k}{2} \right\rfloor \\ &= \left\lfloor \frac{d(C)-1}{2} \right\rfloor \end{aligned}$$

Assim, $c_1, c_2$ estão na mesma classe lateral. Por (4.6), $c_1 = c_2$, o que significa que $e_1 = e_2$ pois $\phi^*$ é injetora.

**Exemplo 4.2.4.** *Para um código $C$ $[7,5]$-RS com $d(C) = 3$, o código $\phi^*(C)$ é um $[21,9]$-código binário que pode corrigir até*

$$s = 3\lfloor \frac{7-3}{2} \rfloor - 3 + 1 = 4$$

*rajadas de erros.*

# 5 DECODIFICANDO CÓDIGOS DE REED-SOLOMON

A teoria de códigos como é conhecida, com diversos códigos e métodos otimizados de codificação e de decodificação, teve um início bem antagônico. Poucos códigos eram conhecidos e as relações entre estes ainda não haviam sido exploradas, os métodos conhecidos fugiam da praticidade e estavam longe de alguma melhora. No caso dos códigos RS não foi diferente.

Reed e Solomon propuseram, ao descreverem o código, um algoritmo de decodificação impraticável e as técnicas de decodificação conhecidas eram pouco úteis. Por exemplo, alguns códigos podiam ser decodificados por meio de tabelas com padrões de erro. Neste método, a síndrome para a palavra recebida é calculada e o padrão de erro com um peso mínimo correspondente é procurado na tabela. Este padrão de erro é subtraído da palavra recebida, produzindo uma palavra código. Usar esta técnica para um código $(63, 53) - RS$ com capacidade para corrigir até 5 erros, exige a construção de uma tabela com aproximadamente $10^{20}$ síndromes.

A busca por um algoritmo de decodificação útil para os códigos RS foi intensa. Peterson [24], Chien [4], Forney [8], Gorestein e Zierler [11] foram os principais pesquisadores do assunto no início dos anos 60 e seus esforços foram muito produtivos. No entanto, ainda não se conseguia utilizar códigos RS capazes de corrigir mais do que sete erros de maneira eficiente. Muitos afirmavam que estes códigos eram apenas uma curiosidade matemática e nunca teriam um uso prático, mas estas pessoas estavam erradas [1].

Em 1967 Berlekamp apresentou o seu algoritmo de decodificação para os códigos RS que permitiu que dezenas de erros fossem corrigidos de uma vez. Um ano depois, Massey chegou em um algoritmo equivalente ao de Berlekamp. Este

método de decodificação ficou conhecido como o algoritmo de Berlekamp-Massey e se tornou um protótipo para a decodificação de vários outros códigos lineares.

Este capítulo é voltado a apresentar o algoritmo publicado por Shiozaki [29] em 1988 e reinventado, independentemente, por Gao [9] 15 anos depois. Primeiramente descrevemos o método de decodificação por síndromes, exibindo como este foi modificado no algoritmo de Berlekamp-Massey. Por fim, apresentamos o algoritmo de Shiozaki-Gao.

## 5.1 Decodificação por Síndromes

Existem diversas abordagens para se resolver o problema de decodificação de uma dada palavra código, aqui apresentamos a técnica mais adotada quando se fala em códigos de Reed-Solomon, a **decodificação baseada em síndromes**.

Sendo $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ o polinômio código e $e(x) = e_0 + e_1x + \cdots + e_{n-1}x^{n-1}$ o polinômio erro, a palavra recebida $r$ na entrada do decodificador é dada pelo polinômio $r(x) = c(x) + e(x) = r_0 + r_1x + \cdots + r_{n-1}x^{n-1}$. Calculando o valor deste polinômio nas raízes do polinômio gerador $g(x)$ obtemos as síndromes

$$\begin{aligned} S_j &= c(\alpha^j) + e(\alpha^j) \\ &= e(\alpha^j) \\ &= \sum_{i=0}^{n-1} e_i\alpha^{ij}, \text{ j} = 1, 2, \ldots, 2\text{t} \end{aligned} \tag{5.1}$$

onde $t$ é a capacidade de correção de erros do código.

Após o cálculo das síndromes, as componentes do padrão de erro podem ser determinadas. Seja $\mu$ a quantidade de erros, $0 \leq \mu \leq t$, que ocorrem nos locais $i_1, i_2, \ldots, i_\mu$. Então o polinômio erro pode ser escrito como

$$e(x) = e_{i_1}\alpha^{i_1} + e_{i_2}\alpha^{i_2} + \cdots + e_{i_\mu}\alpha^{i_\mu}.$$

Avaliando o polinômio recebido em cada uma das raízes do polinômio gerador, as quais são $\alpha, \alpha^2, \ldots, \alpha^{2t}$, temos o seguinte sistema de $2t$ equações

$$S_j = Y_1 X_1^j + Y_2 X_2^j + \cdots + Y_\mu X_\mu^j \tag{5.2}$$

para $j = 1, 2, \ldots, 2t$ e onde $Y_l = e_{i_l}$ são os valores de erro e $X_l = \alpha^{i_l}$ são os números localizadores de erro, para $l = 1, 2, \ldots, \mu$. Este conjunto de equações possui no mínimo uma solução por causa da maneira como as síndromes são definidas. Pode-se mostrar que a solução é única para $0 \leq \mu \leq t$ [3].

Podemos, inicialmente, resumir o processo de decodificação em dois passos:

1. Cálculo das síndromes utilizando a equação 5.1.
2. Resolver o sistema de $2t$ equações não-lineares em 5.2 para encontrar os localizadores de erro e os valores de erro.

Uma abordagem que evite resolver o sistema não-linear, que é difícil de resolver exceto para valores pequenos de $t$, exige a inclusão de passos intermediários neste processo. Com esta finalidade, o polinômio localizador de erro $\Lambda(x)$ é introduzido

$$\Lambda(x) = 1 + \Lambda_1 x + \cdots + \Lambda_\mu x^\mu. \tag{5.3}$$

Uma maneira tradicional de se definir este polinômio é fazer com que suas raízes sejam os inversos dos números localizadores de erro, isto é

$$\Lambda(x) = \prod_{l=1}^{\mu}(1 - xX_l),\ X_l = \alpha^{i_l} \tag{5.4}$$

os números localizadores de erro $X_l$ indicam os erros nos locais $i_l$ para $l = 1, 2, \ldots, \mu$. Multiplicando a equação 5.3 por $Y_l X_l^{j+\mu}$ e usando $x = X_l^{-1}$, chegamos em

$$Y_l(X_l^{j+\mu} + \Lambda_1 X_l^{j+\mu-1} + \cdots + \Lambda_\mu X_l^j) = 0 \tag{5.5}$$

esta equação é válida para cada $l$ e $j$ [3]. A soma destas equações fornece

$$\sum_{l=1}^{\mu} Y_l(X_l^{j+\mu} + \Lambda_1 \sum_{l=1}^{\mu} X_l^{j+\mu-1} + \cdots + \Lambda_\mu \sum_{l=1}^{\mu} X_l^j) = 0. \tag{5.6}$$

Combinando as equações 5.6 e 5.2 obtemos

$$\Lambda_1 S_{j+\mu-1} + \Lambda_2 S_{j+\mu-2} + \cdots + \Lambda_\mu S_j = -S_{j+\mu},\ j = 1, 2, \ldots, \mu \tag{5.7}$$

o que pode ser escrito como

$$\begin{bmatrix} S_1 & S_2 & \ldots & S_\mu \\ S_2 & S_3 & \ldots & S_{\mu+1} \\ \vdots & \vdots & \ldots & \vdots \\ S_\mu & S_{\mu+1} & \ldots & S_{2\mu-1} \end{bmatrix} \begin{bmatrix} \Lambda_\mu \\ \Lambda_{\mu-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{\mu+1} \\ -S_{\mu+2} \\ \vdots \\ -S_{2\mu} \end{bmatrix}. \tag{5.8}$$

**Exemplo 5.1.1.** *Seja* $F(x) = \alpha + \alpha^3 x + \alpha^5 x^2$ *a mensagem que vamos codificar com um código* $RS(7, 3)$ *capaz de corrigir 2 erros com símbolos em* $GF(2^3)$ *obtidos através do polinômio irredutível* $1 + x + x^3$*. O polinômio gerador é*

$$g(x) = \alpha^3 + \alpha x + x^2 + \alpha^3 x^3 + x^4. \tag{5.9}$$

*Sendo* $h(x) = x^4(\alpha + \alpha^3 x + \alpha^5 x^2) \mod g(x)$ *a palavra código é dada por*

$$\begin{aligned} c(x) &= x^4(\alpha + \alpha^3 x + \alpha^5 x^2) + h(x) && (5.10) \\ &= 1 + \alpha^2 x + \alpha^4 x^2 + \alpha^6 x^3 + \alpha^1 x^4 + \alpha^3 x^5 + \alpha^5 x^6. && (5.11) \end{aligned}$$

*Bem, supondo que tenham ocorrido dois erros na transmissão da palavra código* $c(x)$ *chegando ao decodificador o polinômio*

$$r(x) = 1 + \alpha^2 x + \alpha^4 x^2 + x^3 + \alpha^1 x^4 + \alpha^2 x^5 + \alpha^5 x^6. \quad (5.12)$$

*Para este exemplo devemos encontrar 4 síndromes. Como as raízes do polinômio gerador* $g(x)$ *também são raízes da palavra código* $c(x)$*, o erro introduzido alterou as raízes da palavra recebida* $r(x)$*. Assim, podemos utilizar* $r(x)$ *e as raízes de* $g(x)$ *para obter as síndromes.*

$$\begin{aligned} S_1 &= r(\alpha) \\ &= 1 + \alpha^2\alpha + \alpha^4\alpha^2 + \alpha^3 + \alpha\alpha^4 + \alpha^5\alpha^5 + \alpha^5\alpha^6 \\ &= \alpha^2 && (5.13) \end{aligned}$$

$$\begin{aligned} S_2 &= r(\alpha^2) \\ &= 1 + \alpha^2\alpha^2 + \alpha^4\alpha^4 + \alpha^6 + \alpha\alpha^8 + \alpha^5\alpha^{10} + \alpha^5\alpha^{12} \\ &= 0 && (5.14) \end{aligned}$$

$$\begin{aligned} S_3 &= r(\alpha^3) \\ &= 1 + \alpha^2\alpha^3 + \alpha^4\alpha^6 + \alpha^9 + \alpha\alpha^{12} + \alpha^5\alpha^{15} + \alpha^5\alpha^{18} \\ &= \alpha^3 \end{aligned} \tag{5.15}$$

$$\begin{aligned} S_4 &= r(\alpha^4) \\ &= 1 + \alpha^2\alpha^4 + \alpha^4\alpha^8 + \alpha^{12} + \alpha\alpha^{16} + \alpha^2\alpha^{20} + \alpha^5\alpha^{24} \\ &= \alpha^5 \end{aligned} \tag{5.16}$$

*Neste caso, em que 2 erros ocorreram, a equação matricial (5.8) torna-se*

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix}. \tag{5.17}$$

*Substituindo as síndromes calculadas nesta equação podemos resolvê-la invertendo a matriz*

$$\begin{bmatrix} \alpha^2 & 0 \\ 0 & \alpha^3 \end{bmatrix} \tag{5.18}$$

*como esta matriz é diagonal, para encontrarmos a sua inversa basta invertermos os seus elementos*

$$\begin{bmatrix} \alpha^2 & 0 \\ 0 & \alpha^3 \end{bmatrix}^{-1} = \begin{bmatrix} \alpha^{-2} & 0 \\ 0 & \alpha^{-3} \end{bmatrix} = \begin{bmatrix} \alpha^5 & 0 \\ 0 & \alpha^4 \end{bmatrix}. \tag{5.19}$$

*Assim, podemos determinar os coeficientes do polinômio localizador de erro*

$$\begin{aligned}\begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} &= \begin{bmatrix} \alpha^5 & 0 \\ 0 & \alpha^4 \end{bmatrix} \begin{bmatrix} \alpha^3 \\ \alpha^5 \end{bmatrix} \\ &= \begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix} \end{aligned} \tag{5.20}$$

*utilizando a equação (5.3) podemos montar o polinômio localizador de erro*

$$\Lambda(x) = 1 + \alpha^2 x + \alpha x^2. \tag{5.21}$$

*as raízes de $\Lambda(x)$ podem ser obtidas calculando os valores deste polinômio nos pontos diferentes de zero do corpo.*

$$\begin{aligned}\Lambda(1) &= \alpha^5 \\ \Lambda(\alpha) &= 1 \\ \Lambda(\alpha^2) &= 0 \\ \Lambda(\alpha^3) &= \alpha^5 \\ \Lambda(\alpha^4) &= 0 \\ \Lambda(\alpha^5) &= \alpha^4 \\ \Lambda(\alpha^6) &= \alpha^4\end{aligned}$$

*desta forma vemos que $\Lambda(x)$ possui duas raízes, $\alpha^2$ e $\alpha^4$, e com estas podemos determinar os locais de erro: $\alpha^{-2} = \alpha^5$ e $\alpha^{-4} = \alpha^3$. As posições de erro são as potências 3 e 5. Logo, o polinômio erro é da forma $e(x) = e_3x^3 + e_5x^5$ e calculando este nos pontos $\alpha$ e $\alpha^2$ podemos utilizar as síndromes obtidas, $S_1$ e $S_2$, para determinar os valores do erro por meio da solução de*

$$\begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^6 & \alpha^{10} \end{bmatrix} \begin{bmatrix} e_3 \\ e_5 \end{bmatrix} = \begin{bmatrix} \alpha^2 \\ 0 \end{bmatrix}, \tag{5.22}$$

*obtida pela inversão da matriz 2 por 2 acima. Resultando em*

$$\begin{aligned} \begin{bmatrix} e_3 \\ e_5 \end{bmatrix} &= \begin{bmatrix} \alpha^3 & \alpha^5 \\ \alpha^6 & \alpha^{10} \end{bmatrix}^{-1} \begin{bmatrix} \alpha^2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & \alpha^2 \\ \alpha^3 & 1 \end{bmatrix} \begin{bmatrix} \alpha^2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \alpha^2 \\ \alpha^5 \end{bmatrix}. \end{aligned} \tag{5.23}$$

*Finalmente obtemos o polinômio erro* $e(x) = \alpha^2 x^3 + \alpha^5 x^5$. *Em posse do erro, basta corrigir a palavra recebida* $r(x)$ *obtendo-se a palavra código* $c(x) = r(x) + e(x) = 1 + \alpha^2 x + \alpha^4 x^2 + \alpha^6 x^3 + \alpha^1 x^4 + \alpha^3 x^5 + \alpha^5 x^6$.

Uma maneira de resolver o sistema (5.7) seria encontrando a inversa da matriz em (5.8). Entretanto, o valor do número de erros $\mu$ não é conhecido ainda. Este pode ser encontrado da seguinte maneira. Suponha que $\mu = t$, onde $t$ é a capacidade de correção do código, e calcule o determinante da matriz na equação (5.8). Se o valor for nulo, reduza um do valor de tentativa para $\mu$ e repita o teste. Continue este processo até um determinante não nulo ser obtido. A verdadeira quantidade de erros pode ser obtida a partir do tamanho da matriz. Assim, podemos calcular os coeficientes de $\Lambda(x)$ usando o valor de $\mu$ na equação (5.8). Entretanto, este método é muito ineficiente, principalmente para códigos com alta capacidade de correção de erros. O algoritmo de Berlekamp-Massey utiliza uma alternativa para resolver este problema. Esta foi feita para encontrar o polinômio mínimo de uma **sequência de recorrência linear** e pode ser resumido na aplicação do seguinte conjunto de equações recursivas

$$\nabla_i = \sum_{j=0}^{i-1} \Lambda_j^{(i-1)} S_{i-j}, \tag{5.24}$$

$$L_i = \delta(i - L_{i-1}) + (1-\delta)L_{i-1}, \tag{5.25}$$

para $i = 1, 2, \ldots, 2t$. As condições iniciais são $\Lambda_i^{(0)}(x) = 1, B^{(0)}(x) = 1, L_0 = 0$, e $\delta = 1$ se ambos $\nabla_i \neq 0$ e $2L_{i-1} \leq i - 1$, de outra forma $\delta = 0$. Então $\Lambda^{(2t)}(x)$ é o polinômio de menor grau com a propriedade que $\Lambda_0^{(2t)} = 1$ e

$$S_i = \sum_{j=1}^{i-1} \Lambda_j^{(2t)} S_{i-j} = 0, i = L_{2t+1}, \ldots, 2t. \tag{5.26}$$

O algoritmo de Berlekamp-Massey deu origem aos chamados **métodos de decodificação baseados em síndromes**. Para estes métodos, de maneira geral, podemos reescrever o processo de decodificação apresentado no início desta seção da seguinte maneira:

1. Cálculo das síndromes.

2. Determinação de um polinômio localizador de erro. Isto pode ser feito de diferentes maneiras, alguns destes métodos incluem o **Algoritmo de Peterson-Gorenstein-Zierler**, o **Algoritmo de Berlekamp-Massey** e o Algoritmo de Euclides.

3. Busca pelas raízes do polinômio localizador de erro. Costuma-se usar uma busca exaustiva sobre os elementos do corpo, a **Busca Chinesa**.

4. Determinação dos valores do erro, isto normalmente é feito por meio do chamado **Algoritmo de Forney**.

Estas técnicas são explorados e exemplificados por Moon [22].

## 5.2 O Algoritmo de Shiozaki-Gao para o Caso Não Sistemático

O método de Shiozaki-Gao, apresentado a seguir, diferencia-se da abordagem de Berlekamp-Massey no sentido que calcula os símbolos mensagem diretamente sem explicitamente encontrar os locais e as magnitudes de erro. Esta seção foi baseada em [9], [29].

Estamos considerando um código RS não cíclico sobre um corpo extensão $GF(2^m)$, $m \in \mathbb{Z}$. Representamos a palavra mensagem pelo polinômio

$$F(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}. \tag{5.27}$$

A palavra código $c$ é dada por

$$c(x) = F(0) + F(1)x + F(\alpha)x^2 + \cdots + F(\alpha^{n-2})x^{n-3} \tag{5.28}$$

onde $\alpha \in GF(2^m)$ é um elemento primitivo.

A palavra recebida $F$ pode ser obtida a partir de $c$ por meio de

$$F(x) = \sum_{\beta \in GF(2^m)} F(\beta) \frac{G(x)}{x - \beta} \tag{5.29}$$

pois $G(x)x - \beta = 1$ quando $x = \beta$ para cada $\beta \in GF(2^m)$, sendo $G(x) = \prod_{\beta \in GF(2^m)}(x - \beta)$.

A palavra recebida $r$ é dada por $r = c + e$, onde $e$ é a palavra erro.

Sejam

$$\begin{aligned}\sigma(x) &= \prod_{\beta\in B}(x-\beta) \text{ polinômio localizador de erro} \\ \omega(x) &= \sum_{\beta\in B} e_\beta \prod_{\gamma\in B,\gamma\neq\beta}(x-\gamma) \text{ polinômio avaliador de erro} \end{aligned} \tag{5.30}$$

onde $B$ é o conjunto das posições em que o vetor $e$ possui coordenadas não nulas.

Seja $R(x)$ o polinômio determinado pela interpolação, de maneira análoga à 5.29, dos coeficientes de $r(x)$,

$$R(x) = (r_0)\frac{G(x)}{x} + \sum_{1\leq i\leq n-2)}(r_i)\frac{G(x)}{x-\alpha^i}. \tag{5.31}$$

Usando 5.28 - 5.31 podemos escrever $R(x)$ como

$$R(x) = F(x) + \frac{\omega(x)G(x)}{\sigma(x)}. \tag{5.32}$$

de onde

$$\sigma(x)R(x) = \sigma(x)F(x) + \omega(x)G(x) \tag{5.33}$$

ou

$$\sigma(x)R(x) \equiv \sigma(x)F(x) \mod G(x). \tag{5.34}$$

Seja

$$s_n(x)a(x) + t_n(x)b(x) = r_n(x) \tag{5.35}$$

a equação obtida no n-ésimo passo do algoritmo de Euclides para a obtenção do $mdc(a(x), b(x))$. Então, temos os seguinte

**Lema 5.2.1.** *Se* $\nu \geq \text{grau}[\text{mdc}(a(x), b(x))]$, $\mu + \nu = \text{grau}[a(x)] - 1$, *então existe um único índice* $j$ *no algorítmo tal que*

$$\text{grau}[t_j(x)] \leq \mu \text{ e } \text{grau}[r_j(x)] \leq \nu. \tag{5.36}$$

**Teorema 5.1.** *Se* $t(x)$ *e* $r(x)$ *são não nulos e*

$$t(x)b(x) \equiv r(x) \mod a(x) \tag{5.37}$$

$$\text{grau}[t(x)] + \text{grau}[r(x)] < \text{grau}[a(x)], \tag{5.38}$$

*então* $t(x) = \lambda(x)t_j(x)$, $r(x) = \lambda(x)r_j(x)$ *com* $j$ *como no lema.*

Se $F(x) = 0$ então $w(r) \leq t$ e assumimos que $c(x) = 0$. Caso contrário, temos

$$grau[mdc(G(x), R(x))] = n - w(r) \leq n - t - 1 = k + t - 1. \tag{5.39}$$

Aplicando o lema com $a(x) = G(x)$, $b(x) = R(x)$, $\nu = k + t - 1$, $\mu = t$ e aplicando o teorema com $t(x) = \sigma(x)$ e $r(x) = \sigma(x)F(x)$, a condição sobre os graus é satisfeita. Assim, determinamos o polinômio

$$F(x) = \frac{\sigma(x)F(x)}{\sigma(x)} = \frac{r(x)}{t(x)} = \frac{r_j(x)}{t_j(x)}. \tag{5.40}$$

onde $j$ é o primeiro índice para o qual $grau[r_j(x)] < k + t$. Caso $grau[r_j(x)] - grau[t_j(x)] \geq k$ ou, ainda, se $t_j(x)$ não dividir $r_j(x)$, então mais do que $t$ erros ocorreram e a impossibilidade de decodificação é detectada.

**Exemplo 5.2.1.** *Vamos agora dar continuidade ao exemplo 4.1.1. Lembrando que o código RS é considerado sobre* $GF(2^3)$ *usando o irredutível* $p(x) = x^3 + x + 1$, *com* $n = 8$, $k = 4$ *e a palavra mensagem é dada por* $F = (\alpha, \alpha^2, \alpha^5, \alpha^4)$ *onde* $\alpha$ *é uma raiz de* $p(x)$. *O polinômio da palavra código* $c$ *é*

$$c(x) = \alpha + \alpha^5 x + x^2 + \alpha^3 x^3 + \alpha^4 x^4 + \alpha^4 x^5 + \alpha^4 x^6 + x^7. \tag{5.41}$$

*Considere ainda que* $t = 2$, $G(x) = x^8 - x$ *e que o polinômio da palavra recebida* $r$ *é dado por*

$$r(x) = \alpha + \alpha^5 x + \alpha^6 x^2 + \alpha^3 x^3 + \alpha^4 x^4 + \alpha^4 x^5 + \alpha^3 x^6 + x^7. \tag{5.42}$$

*Então*

$$\begin{aligned} R(x) &= \alpha\frac{x^8 - x}{x} + \alpha^5\frac{x^8 - x}{x - 1} + \alpha^6\frac{x^8 - x}{x - \alpha} + \cdots + \alpha\frac{x^8 - x}{x - \alpha^6} \\ &= x^7 + \alpha^6 x^6 + \alpha x^5 + \alpha^4 x^4 + \alpha^2 x^3 + \alpha^6 x^2 + \alpha^2 x + \alpha \end{aligned}$$

*Do algoritmo de Euclides para o cálculo do* $MDC(R(x), G(x))$

$$\begin{aligned} t_2(x) &= \alpha^2 + x + 1, \\ r_2(x) &= \alpha^5 x^5 + \alpha^3 x^4 + \alpha x^3 + \alpha^5 x^2 + \alpha^4 x + \alpha. \end{aligned} \tag{5.43}$$

*Note que* $grau[r_2(x)] < k + t = 6$.

*Assim, pela equação 5.40 o polinômio da palavra* $F$ *é dado por,*

$$F(x) = \frac{r_2(x)}{t_2(x)} = \alpha^4 x^3 + \alpha^5 x^2 + \alpha^2 x + \alpha. \tag{5.44}$$

.

A operação predominante no algoritmo acima é o passo do algoritmo de Euclides, que usa $\mathcal{O}(n^2)$ operações para o cálculo do MDC. Além de reinventar o algoritmo de decodificação apresentado acima, Gao apresentou também uma versão melhorada deste. Neste algoritmo melhorado Gao diminuiu o esforço computacional do passo do $MDC$ usando o princípio de funcionamento do então chamado "Algoritmo Euclideano Rápido". No segundo passo do algoritmo de Euclides, Gao utiliza apenas os coeficientes dos termos de maior grau do divisor e do dividendo [9],[18].

Gao indicou como aprimorar mais o algoritmo. Utilizar FFT para efetuar os passos de MDC, avaliação, interpolação, divisão e multiplicação polinomiais. Sendo necessário que o corpo utilizado tenha uma n-ésima raiz primitiva da unidade.

Chen e Yan [23] analisaram o algoritmo aqui apresentado e concluiram que este possui um desempenho pior do que os algoritmos de decodificação tradicionais de Reed-Solomon para quase todos os casos práticos. Recentemente, Matter afirmou que fez a mesma análise e chegou aos mesmos resultados. Usando um resultado semelhante à regra de L'Hopital de cursos de cálculo ele apresentou uma maneira de contornar este problema de performance fazendo um novo algoritmo melhorado tão eficiente quanto os tais métodos tradicionais de decodificação, exigindo $O(n(\log n)^2)$ operações.

# 6 CONCLUSÃO

Apresentamos um algoritmo de decodificação de palavras código de um código RS codificadas de maneira não sistemática. Este algoritmo, ao contrário do algoritmo de Berlekamp-Massey, não exige que o código seja cíclico. O algoritmo apresentado diferencia-se, ainda, da solução de Berlekamp-Massey por não calcular explicitamente os locais e os valores de erro, calculando diretamente a palavra mensagem.

Mostramos ainda um resumo dos últimos acontecimentos sobre os estudos e melhorias na performance do algoritmo apresentado aqui, concluindo que o este algoritmo pode ser melhorado para ser tão eficiente quanto outros métodos de decodificação. Como Mateer [18] observou, estas comparações assintóticas são de pouca impotância em aplicações práticas, pois elas não exigem valores grandes de $n$, e a decisão pelo uso de um algoritmo ou outro deve ser feita com muito cuidado, comparando com outros algoritmos existentes em seu próprio computador.

Uma possível continuação a este trabalho, seria buscar melhorias de performance partindo dos recentes resultados de Mateer [17] e [18], procurando por maneiras mais vantajosas de calcular $FFT$. Outra possibilidade seria tentar adaptar este algoritmo para códigos $BCH$ e generalizar este para decodificação de códigos de geometria algébrica por meio de bases de Gröbner, como indicado por [9].

# Referências Bibliográficas


[1] BHARGAVA, V.; WICKER, S. **Reed-Solomon Codes and Their Applications**. New York: IEEE Press., 1994.

[2] BLAHUT, R. E. **Algebraic Codes for Data Transmission**. New York: Cambridge University Press., 2003.

[3] BLAHUT, R. E. **Theory and Practice of Error Control Codes**. Boston: Addison-Wesley, 1984.

[4] CHIEN, R. T. Cyclic Decoding Procedure for the Bouse-Chaudhuri-Hocquenghem Codes. **IEEE Transactions on Information Theory**, New York: IEEE Information Theory Society, v. 10, p.357-363, october, 1964.

[5] CLAUSEN, M.; BAUM, U. **Fast Fourier Transforms**. Mannhein: BI-Wissenschaftsverlag, 1993.

[6] COOLEY, J. W.; TUKEY, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. **Mathematics of Computation**, Murray Hill: Bell Telephone Laboratories, v. 19, p. 297-301, 1965.

[7] DUTRA, F. S. **Sobre Códigos Diedrais e Quatérnios**. Tese de Doutorado-Departamento de Matemática da Universidade Federal de Minas Gerais, Belo Horizonte, 2006.

[8] FORNEY, G. D. On Decode BCH Codes. **IEEE Transactions on Information Theory**, New York: IEEE Information Theory Society, v. 11, p.549-557, october, 1965.

[9] GAO, S. A new algorithm for decoding Reed-Solomon codes. **Communications, Information and Network Security**, New York: Springer, p.

55-68, 2003.(The Springer International Series in Engineering and Computer Science, v. 712)

[10] GOLAY, M. J. E. Notes on Digital Coding. **Proceedings of the Institute of Radio Engineers**, New York: Institute of Radio Engineers, v. 37, p. 657, july, 1948.

[11] GORENSTEIN, D.; ZIERLER, N. A Class of Error Correcting Codes in $p^m$ Symbols. **Journal of the Society of Industrial and Applied Mathematics**, Philadelphia: Society of Industrial and Applied Mathematics, v. 9, p.207-214, june, 1961.

[12] HAMMING, R. W. Error Detecting and Error Correcting Codes. **The Bell System Technical Journal**, New York: American Telephone and Telegraph Co., v. 29, n. 2, p. 147-160, 1950.

[13] HEFEZ, A.; VILLELA, M. L. T. **Códigos Corretores de Erros**. Rio de Janeiro: IMPA, 2002.

[14] LING, S.; XING, C. **Coding Theory a First Course**. New York: Cambridge University Press., 2004.

[15] LIPSON, J. D. **Elements of Algebra and Algebraic Computing**. Toronto: Benjamin/Cummings Publishing Company, 1981.

[16] MASUDA, A. **Tópicos de Corpos Finitos com Aplicações em Criptografia e Teoria dos Códigos**. Rio de Janeiro: IMPA, 2007.

[17] MATEER, T. **Fast Fourier Transform Algorithms with Applications**. 2008. 328 f. PhD Dissertation-Graduate School of Clemson University, Clemson, 2008.

[18] MATEER, T. Simple algorithms for decoding systematic Reed-Solomon codes. **Designs, Codes and Cryptography**, New York: Springer-USA, v. 65, p. 1-15, july, 2012.

[19] MCELIECE, R. J. **The Theory of Information and Coding**. Boston: Addison-Wesley, 1977.

[20] MENDES, L. L. Uma Visão sobre a TV Digital no Brasil. **T&C Amazônia**, Manaus: Fundação Centro de Análise, Pesquisa e Inovação Tecnológica, ano V, n. 12, outubro, 2007.

[21] MILIES, C. P. **Breve Introdução à Teoria de Códigos Corretores de Erros**. Campo Grande: UFMS, 2009.

[22] MOON, T. K. **Error Correction Coding**: Mathematical Methods and Algorithms. New Jersey: John Wiley & Sons, Inc., 2005.

[23] NING, C.; ZHIYUAN, Y. Complexity analysis of Reed-Solomon decoding over $GF(2^m)$ without using syndromes. **EURASIP Journal on Wireless Communications and Networking, Advances in Error Control Coding Techniques**, New York: Hindawi Publishing Corporation, 2008.

[24] PETERSON, W. Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes. **IRE Transactions on Information Theory**, New York: Institute of Radio Engineers, v. 6, p. 459-470, september, 1960.

[25] PRETZEL, O. **Error-Correcting Codes and Finite Fields**. New York: Clarendon Press, Oxford University Press., 1992.

[26] ROMAN, S. **Introduction to Coding and Information Theory**. New York: Springer, 1997.

[27] REED, I. S.; SOLOMON, G. Polynomial Codes Over Certain Finite Fields. **Journal of the Society for Industrial and Applied Mathematics**, Philadelphia: Society of Industrial and Applied Mathematics, v. 8, n. 2, p. 300-304, june, 1960.

[28] SHANNON, C. E. A Mathematical Theory of Communication. **The Bell System Technical Journal**, New York: American Telephone and Telegraph Co., v. 27, p. 379-423, 623-656, july, october, 1948.

[29] SHIOZAKI, A. Decoding of Redundant Residue Polynomial Codes Using Euclid's Algorithm. **IEEE Transactions on Information Theory**, New York: IEEE Information Theory Society, v. 34, n. 5, p. 1351-1354, september, 1988.

[30] VOLOCH, J. F. **Códigos Corretores de Erros**. Rio de Janeiro: IMPA, 1987.

# Apêndice A O ALGORITMO DE EUCLIDES

Apresentamos aqui uma técnica que desempenha um papel essencial nas discussões desenvolvidas ao longo deste texto, o algoritmo de Euclides. Este, com aproximadamente 2000 anos de seu invento, foi criado para encontrar o máximo divisor comum de inteiros sem ter que separá-los em fatores primos. Além de sua utilidade na construção de corpos finitos ainda fornece um método para o processamento de erros nos códigos de Reed-Solomon. Este capítulo é baseado em [25] e [15] onde podem ser encontradas as demonstrações omitidas aqui.

## A.1 O Máximo Divisor Comum

A maneira como aprendemos na escola a calcular o máximo divisor comum ($mdc$) de dois inteiros é por meio de uma comparação entre os conjuntos de divisores destes números. Por exemplo, se quisermos calcular o $mdc$ de 18 e 12 começamos obtendo os seus divisores

$$\begin{aligned} D_{18} &= \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{6}, 9, 18\}, \\ D_{12} &= \{\mathbf{1}, \mathbf{2}, \mathbf{3}, 4, \mathbf{6}, 12\} \end{aligned}$$

de onde podemos ver que os fatores comuns de 12 e 18 são 1, 2, 3 e 6, e o máximo divisor comum de 12 e 18 é 6

$$mdc(18, 12) = 6.$$

Assim, dados $a$ e $b$ em $\mathbb{Z}$ aprendemos a formar dois conjuntos. O conjunto $D_a$, dos divisores de $a$, e o conjunto $D_b$, dos divisores de $b$, e a tomar o $mdc(a,b)$ em função dos elementos da intersecção entre estes dois conjuntos. Isto nos leva a seguinte

**Definição A.1.1** (**Máximo divisor comum de dois inteiros**). *Sejam $a$ e $b$ em $\mathbb{Z}$, com $D_a$ e $D_b$ os seus respectivos conjuntos de divisores. O máximo divisor comum de $a$ e $b$ é dado por*

$$mdc(a,b) = \max D_a \bigcap D_b$$

A maneira como podemos definir o máximo divisor comum de dois números inteiros não é única. Outra possível definição para o $mdc$ é como sendo o produto dos elementos da intersecção entre os conjuntos de fatores primos de cada um dos número considerados. Isto é, dados $a$ e $b$ em $\mathbb{Z}$ se $F_a$ é o conjunto dos fatores primos de $a$ e $F_b$ é o conjunto dos fatores primos de $b$, podemos escrever $mdc(a,b) = \prod_{f \in F} f$, onde $F = F_a \bigcap F_b$. Note que costumamos assumir que o $mdc$ é positivo, isto serve para garantirmos unicidade sem termos qualquer preocupação com sinais.

Com a intenção de estender o conceito de $mdc$ a um domínio $D$ e deixar claro o motivo do sinal não influenciar na escolha do $mdc$ é que definimos a seguinte relação de equivalência

**Definição A.1.2** (**Associados**). *Sejam $a, b \in D^*$, com $D$ um domínio. Dizemos que $a$ e $b$ são associados se $a = ub$, onde $u \neq 0$ é invertível em $D$.*

Notação: $a \sim b$.

A seguinte proposição nos diz que podemos trocar um elemento por qualquer um de seus associados sem que hajam mudanças em suas relações de divisibilidade.

**Proposição A.1.1.** *Sejam $a \sim b$ e $c \sim d$. Se $a|c \Rightarrow b|d$*

De fato não requer muito esforço vericarmos que $\sim$ é uma relação de equivalência sobre $D^*$. Claramente $\sim$ é simétrica, pois se tivermos $a = ub$ com $u$ invertível, também temos $b = ga$ com $g = u^{-1}$ invertível. Para percebermos que $\sim$ é reflexiva basta tomarmos $u = 1$. Esta relação é também transitiva, pois se tivermos três elementos $a, b$ e $c$ de $D$ em que verifique-se $a = ub$ e $b = gc$ com $u$ e $g$ invertíveis, claramente podemos escrever $a = (ug)c$.

Sendo $\sim$ uma relação de equivalência sobre $D^*$, cada elemento $d \in D^*$ está contido na classe formada pelos produtos de $d$ com os elementos não-nulos invertíveis de $D$. Isto nos permite falar em unicidade para o *mdc*, assim quando falarmos *no mdc* estaremos falando do representante da classe em que o *mdc* está contido. Para isto temos que determinar quais são os representantes das classes de associados no domínio em questão.

**Exemplo A.1.1.**

*Em $\mathbb{Z}$, como os invertíveis são $1$ e $-1$, dado $d \in \mathbb{Z}$, $[d] = \{-d, d\}$. Podemos tomar como representante desta classe o elemento positivo $d$.*

**Exemplo A.1.2.**

*Em $F[x]$, como os invertíveis são as constantes não-nulas, dado $d(x) \in F[x]$, $[d(x)] = \{fd(x) | f \in F^*\}$. Podemos tomar como representante o único associado mônico (Se $d(x) = d_n x^n +$ termos de grau menor $(d_n \neq 0)$, o único associado mônico é $d_n^{-1} d(x)$).*

De maneira mais geral, *um* máximo divisor comum $d$ de $a$ e $b$ em um domínio $D$ é caracterizado por duas propriedades. A primeira diz que $d$ é um fator comum de $a$ e $b$, a segunda diz que qualquer outro fator comum de $a$ e $b$ é também um fator de $d$:

**MDC1.** $d|a$ e $d|b$

**MDC2.** $c|a$ e $c|b \Rightarrow c|d$

### A.1.1 Obtendo o mdc

A maneira de se obter o *mdc* de dois inteiros, apresentada na seção anterior, pode ser muito trabalhosa. Se os números forem grandes os conjuntos de divisores também serão grandes. Com a intenção de simplificar este problema o algoritmo de Euclides é construido, a idéia chave deste é recair sobre um problema equivalente ao original, mas menos trabalhoso. Para chegarmos a este problema equivalente enunciamos as seguintes propriedade do *mdc*

**Proposição A.1.2.** *Em um domínio Euclideano D*

**(1)** $mdc(a, 0) = a$,

**(2)** $mdc(a, b) = mdc(b, a \mod b)$, *para* $b \neq 0$.

**Exemplo A.1.3.** *Usando as propriedades acima obtemos*

$$\begin{aligned} mdc(18, 12) &= mdc(12, 6) \\ &= mdc(6, 0) \\ &= 6. \end{aligned}$$

*Isto nos diz que* 6 *é um mdc de* 18 *e* 12*, sabemos que* $[6] = \{-6, 6\}$ *e escolhendo* 6 *como o representante desta classe dizemos que o mdc de* 18 *e* 12 *é* 6*.*

A cada vez que simplificamos o problema da obtenção do *mdc* estamos executando uma divisão e voltando as nossas atenções para o resto gerado apartir desta divisão. Podemos representar este processo de obtenção do *mdc* por uma tabela com duas colunas, uma para o quociente e a outra para o resto da divisão.

| $Q$ | $R$ |
|---|---|
| | 18 |
| 1 | 12 |
| 2 | 6 |
| | 0 |

### A.1.2 Cofatores e a tabela de 4 colunas

Do método visto na seção anterior podemos observar que é possível expressar o *mdc* de 18 e 12 como uma combinação destes dois números: $6 = 18 - 1 \times 12$. Este é o nosso objetivo nesta seção. Colocar o *mdc* de dois elementos $a, b \in D$ na forma $mdc(a, b) = ua + vb$. Os coeficientes $u$ e $v$ são chamados de *cofatores* e o que vamos fazer é acrescentar duas colunas na nossa tabela para que em cada linha tenhamos uma expressão como esta. Começamos agora a descrever o algoritmo de Euclides. A nossa tabela passa a ser formada por quatro colunas: a do quociente, $Q$, a do resto, $R$, e as duas acrescentadas para os cofatores, $U$ e $V$. Vamos começar a contar as linhas partindo a contagem de $-1$.

Passo 1:
Iniciamos a linha $-1$ assim, supondo $grau(a) \geq grau(\text{b})$

$$r_{-1} = a,\ u_{-1} = 1,\ v_{-1} = 0$$

e a linha 0 iniciamos assim

$$r_0 = b,\ u_0 = 0,\ v_0 = 1.$$

Para obtermos a linha 1 dividimos $a$ por $b$, $a = q_1 \times b + r_1$. Tomamos $u_1 = 1$ e $v_1 = -q_1$.

Passo 2:

Na obtenção das outras linhas dividimos $r_{k-1}$ por $r_k$, isto nos fornece $r_{k+1} = r_{k-1}$ mod $r_k(x)$ e $q_{k+1} = r_{k-1} div\, r_k(x)$.

Passo 3:

As entradas das colunas $U$ e $V$ são, respectivamente,

$$\begin{aligned} u_{k+1} &= u_{k-1} - q_{k+1}u_k, \\ v_{k+1} &= v_{k-1} - q_{k+1}v_k. \end{aligned}$$

Passo 4:

Se chegarmos em $r_{k+1} \neq 0$, continuamos do passo 2. Se $r_{k+1} = 0$ devemos parar, neste ponto já temos

$$r_k = u_k a + v_k b$$

este é o *mdc* de $a$ e $b$.

A validade deste algoritmo recai em garantirmos três fatos. O primeiro é que o algoritmo termina em número finito de passos, o segundo é que realmente chegamos ao *mdc* de $a$ e $b$ no último elemento não-nulo da coluna $R$ e o terceiro é que a expressão $r_k = u_k a + v_k b$ vale para cada linha $k$ da tabela. Resumimos estes fatos no seguinte

**Teorema A.1.** *Sejam $a$ e $b$ em um domínio $D$. Suponha que o algoritmo de Euclides foi executado para $a$ e $b$. Denotando as colunas da tabela por $Q, R, U$ e $V$ com*

*entradas em cada linha $k$ dadas, respectivamente, por $q_k, r_k, u_k$ e $v_k$. Então valem as seguintes afirmações.*

**(a)** *O algoritmo termina após um número finito de passos.*

**(b)** *O mdc de $a$ e $b$ é o último elemento não-nulo da coluna $R$.*

**(c)** *Para cada $k$ temos $r_k = u_k a + v_k b$.*

Vamos aplicar o algoritmo em um exemplo simples, para que possamos entender melhor o seu funcionamento.

**Exemplo A.1.4.** *Para calcular o mdc de 12 e 104 começamos executando o passo 1 de iniciação da tabela. No passo 2 devemos dividir 12 por 8 e assim obtemos $q_2 = 1$ e o resto $r_2 = 4$. Seguindo para o passo 3 calculamos $u_2 = 0 - q_2 \times 1 = -1$ e $v_2 = 1 - q_2(-8) = 9$. Voltando para o passo 2 dividimos 8 por 4 e obtemos $r_3 = 0$, por isso sabemos que $mdc(104, 12) = 4$ e pode ser expresso como $4 = -1 \times 104 + 9 \times 12$.*

| *Linha* | *Q* | *R* | *U* | *V* |
|---|---|---|---|---|
| *-1* | - | *104* | *1* | *0* |
| *0* | - | *12* | *0* | *1* |
| *1* | *8* | *8* | *1* | *-8* |
| *2* | *1* | *4* | *-1* | *9* |
| *3* | *2* | *0* | *3* | *-26* |

**Exemplo A.1.5.** *Vamos utilizar o algoritmo de Euclides para obter o mdc entre $f(x) = x^4 + x^3 + 1$ e $g(x) = x^3 + 1$. Inserimos uma linha extra na tabela representando a divisão polinomial de $f(x)$ por $g(x)$. Pela tabela vemos que $mdc(f(x), g(x)) = 1$.*

| *Linha* | $Q$ | $R$ | $U$ | $V$ |
|---|---|---|---|---|
| *-1* | - | $x^4 + x^3 + 1$ | *1* | *0* |
| *0* | - | $x^3 + 1$ | *0* | *1* |
| - | $x$ | $x^3 + x + 1$ | *1* | $x$ |
| *1* | 1 | $x$ | *1* | $x + 1$ |
| *2* | $x^2$ | 1 | $x^2$ | $x^3 + x^2 + 1$ |
| *3* | $x$ | 0 | $x^3 + 1$ | $x^4 + x^3 + 1$ |

*Esta tabela pode ser montada de outra maneira representando os elementos de* $GF(16)$ *na forma binária. Por exemplo o polinômio* $x^3 + 1$ *representa o símbolo* $9 = 1001$ *em sua forma binária. Nas linhas abaixo remontamos a mesma tabela representando* $x^4 + x^3 + 1$ *por* $11001$ *e* $x^3 + 1$ *por* $1001$*. Desta forma poodemos ter uma idéia mais clara dos passos intemediários de divisão polinomial*

| | *Linha* | $Q$ | $R$ | $U$ | $V$ |
|---|---|---|---|---|---|
| | *-1* | - | *11001* | *00001* | *00000* |
| | *0* | - | *01001* | *00000* | *00001* |
| *0,1:* | - | *10* | *01011* | *00001* | *00010* |
| *0,2:* | *1* | *01* | *00010* | *00001* | *00011* |
| | *2* | *100* | *00001* | *00100* | *01101* |
| | *3* | *10* | *00000* | *01001* | *11001* |

*A linha intermediária de divisão polinomial foi incluída. Para mostrar como esta linha antes da linha 1 é calculada considere a tabela para divisão de* $11001$ *por* $01001$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | *1001)* | *1* | *1* | *0* | *0* | *1* | *(11* |
| | | *1* | *0* | *0* | *1* | | |
| (1) | | | *1* | *0* | *1* | *1* | |
| | | | *1* | *0* | *0* | *1* | |
| (2) | | | | | *1* | *0* | |

*As linhas que aparecem na tabela estão marcadas* (1) *e* (2). *Estes são obtidos mudando 1001 para coincidir com o coeficiente de mais alto grau não nulo restante e subtraindo. Na tabela a coluna Q representa a mudança e a regra é a seguinte:*

*Escolha q como uma potência de x de forma que* $q \times r_0$ *coincida com o termo de maior grau de* $r_{-1}$ *(para um corpo geral tomamos* $q = ax^m$ *de modo que os maiores termos coincidam).*

*Agora calcule a próxima linha, como se este fosse o q correto, dando à linha o número 0,1. As entradas desta linha são:*

$$q_{0,1} = q,\ r_{0,1} = r_{-1} - qr_0,\ u_{0,1} = u_{-1} - qu_0,\ v_{0,1} = v_{-1} - qv_0.$$

*A linha resultante continua tendo grau maior que o grau de* $r_0$. *então repita o processo escolhendo q tal que* $q \times r_0$ *coincide com o termo de maior grau de* $r_{0,1}$. *Isto produz* $r_{0,2}$, *o que fornece uma nova linha intermediária com entradas*

$$q_{0,2} = q,\ r_{0,2} = r_{0,1} - qr_0,\ u_{0,2} = u_{0,1} - qu_0,\ v_{0,2} = v_{0,1} - qv_0.$$

*Continue da mesma forma até o grau da entrada da coluna R ficar abaixo do grau de* $r_0$. *Rotule a última linha como linha 1. A entrada da coluna Q não representa o verdadeiro quociente q, para obter este é necessário somar os valores obtidos nestas linhas intermediáris. Agora continue para o próximo passo do algoritmo de Euclides.*

*Tabela A.1: $GF(16)$ baseado em $x^4 + x^3 + 1$*

| *Log* | | − | *0* | *1* | *12* | *2* | *9* | *13* | *7* | *3* | *4* | *10* | *5* | *14* | *11* | *8* | *6* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* |
| *0* | × | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* |
| *1* | + | *1* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* |
| *2* | | *2* | *3* | *4* | *6* | *8* | *10* | *12* | *14* | *9* | *11* | *13* | *15* | *1* | *3* | *5* | *7* |
| *3* | | *3* | *2* | *1* | *5* | *12* | *15* | *10* | *9* | *1* | *2* | *7* | *4* | *13* | *14* | *11* | *8* |
| *4* | | *4* | *5* | *6* | *7* | *9* | *13* | *1* | *5* | *11* | *15* | *3* | *7* | *2* | *6* | *10* | *14* |
| *5* | | *5* | *4* | *7* | *6* | *1* | *8* | *7* | *2* | *3* | *6* | *9* | *12* | *14* | *11* | *4* | *1* |
| *6* | | *6* | *7* | *4* | *5* | *2* | *3* | *13* | *11* | *2* | *4* | *14* | *8* | *3* | *5* | *15* | *9* |
| *7* | | *7* | *6* | *5* | *4* | *3* | *2* | *1* | *12* | *10* | *13* | *4* | *3* | *15* | *8* | *1* | *6* |
| *8* | | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* | *15* | *7* | *6* | *14* | *4* | *12* | *13* | *5* |
| *9* | | *9* | *8* | *11* | *10* | *13* | *12* | *15* | *14* | *1* | *14* | *12* | *5* | *8* | *1* | *3* | *10* |
| *10* | | *10* | *11* | *8* | *9* | *14* | *15* | *12* | *13* | *2* | *3* | *11* | *1* | *5* | *15* | *8* | *2* |
| *11* | | *11* | *10* | *9* | *8* | *15* | *14* | *13* | *12* | *3* | *2* | *1* | *10* | *9* | *2* | *6* | *13* |
| *12* | | *12* | *13* | *14* | *15* | *8* | *9* | *10* | *11* | *4* | *5* | *6* | *7* | *6* | *10* | *7* | *11* |
| *13* | | *13* | *12* | *15* | *14* | *9* | *8* | *11* | *10* | *5* | *4* | *7* | *6* | *1* | *7* | *9* | *4* |
| *14* | | *14* | *15* | *12* | *13* | *10* | *11* | *8* | *9* | *6* | *7* | *4* | *5* | *2* | *3* | *2* | *12* |
| *15* | | *15* | *14* | *13* | *12* | *11* | *10* | *9* | *8* | *7* | *6* | *5* | *4* | *3* | *2* | *1* | *3* |