Paulo Cesar dos Santos

Códigos verificadores e corretores de erros

Santo André, 2018

Universidade Federal do ABC

Centro de Matemática, Computação e Cognição

Paulo Cesar dos Santos

Códigos verificadores e corretores de erros

Orientador: Prof. Dr. Armando Caputi

Dissertação de mestrado apresentada ao Centro de Matemática, Computação e Cognição para obtenção do título de Mestre

Este exemplar corresponde a versão final da dissertação
defendida pelo aluno Paulo Cesar dos Santos,
e orientada pelo Prof. Dr. Armando Caputi.

Santo André, 2018

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Santos, Paulo Cesar
Códigos verificadores e corretores de erros / Paulo Cesar Santos. — 2018.

76 fls.

Orientador: Armando Caputi

Dissertação (Mestrado) — Universidade Federal do ABC, Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, Santo André, 2018.

1. identificação de erros. 2. sistemas modulares. 3. códigos corretores de erros. 4. códigos lineares. I. Caputi, Armando. II. Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, 2018. III. Título.

Este exemplar foi revisado e alterado em relação à versão original, de acordo com as observações levantadas pela banca no dia da defesa, sob responsabilidade única do autor e com a anuência de seu orientador.

Santo André, 17 **de** maio **de 20**18**.**

Assinatura do autor: Paulo Cesar dos Santos

Assinatura do orientador:

MINISTÉRIO DA EDUCAÇÃO
Fundação Universidade Federal do ABC
Programa de Pós-Graduação em Mestrado Profissional em Matemática em Rede Nacional
Avenida dos Estados, 5001 – Bairro Santa Terezinha – Santo André – SP
CEP 09210-580 · Fone: (11) 4996-0017
profmat@ufabc.edu.br

FOLHA DE ASSINATURAS

Assinaturas dos membros da Banca Examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Paulo Cesar dos Santos, realizada em 28 de fevereiro de 2018:

Prof.(a) Dr.(a) **Armando Caputi** (Universidade Federal do ABC) – Presidente

Prof.(a) Dr.(a) **Lucia Satie Ikemoto Murakami** (Universidade de São Paulo) – Membro Titular

Prof.(a) Dr.(a) **Jerônimo Cordoni Pellegrini** (Universidade Federal do ABC) – Membro Titular

Prof.(a) Dr.(a) **Rafael de Mattos Grisi** (Universidade Federal do ABC) – Membro Suplente

Prof.(a) Dr.(a) **Barbara Corominas Valerio** (Universidade de São Paulo) – Membro Suplente

Dedico este trabalho ao meu filho Augusto Seabra dos Santos e aos meus pais.

AGRADECIMENTOS

Agradeço a Deus por esta conquista e por me dar força nessa trajetória.

À minha esposa Priscila e ao meu filho Augusto, por compreender minha ausência nesses anos de estudos.

À minha irmã Luciana, por sempre me motivar.

Aos meus pais, por sempre incentivar meus estudos e sempre fazer o possível para ajudar.

À CAPES, pelo apoio financeiro.

Aos professores do PROFMAT da UFABC.

Aos colegas da turma.

À diretora Ana Paula e coordenadora Evelise da EE Profª Ruth Sá, por sempre ajustar meus horários e incentivar a continuidade de meus estudos.

Em especial, ao professor Armando Caputi, por sua paciência e dedicação às orientações.

"É impossível progredir sem mudança, e aqueles que não mudam suas mentes não podem mudar nada."

(George Bernard Shaw)

RESUMO

Nesta dissertação, apresentamos os dígitos que verificam erros em códigos numéricos e os códigos que identificam e corrigem erros em informações transmitidas ou armazenadas. Em relação aos dígitos que verificam erros, enfocamos os sistemas modulares, os principais erros que podem ser cometidos na digitação de sequências numéricas e a eficiência do dígito para detectar sua presença. Já em códigos corretores de erros, denotamos os conceitos de um código e sua utilidade em um sistema de comunicação, mostramos como codificar uma informação e depois como decodificá-la corrigindo erros, caso existam. Finalmente, explicitamos algumas aplicações dos dígitos verificadores e códigos corretores de erros para o Ensino Fundamental e Médio.

Palavras-chave: identificação de erros, sistemas modulares, códigos corretores de erros, códigos lineares

ABSTRACT

In this dissertation, we present the digits that verify errors in numerical codes and the codes that identify and correct errors in transmitted or stored information. Regarding the digits that verify errors, we focus on the modular systems, the main errors that can be presented during the typing of numerical sequences and the efficiency of the digit to detect their presence. As for theerror-correcting codes, we denote the concepts of a code and its use in a communication system, we show how to code pieces of information and then how to decode it correcting errors, in case they exist. Finally, we described in details some applications of the verifier digits and error correction codes for Elementary and Middle School.

Keywords: error identification, modular systems, error-correcting codes, linear codes

CONTEÚDO

INTRODUÇÃO

Nesta dissertação, apresentamos e demonstramos os conceitos dos dígitos verificadores de erros em códigos numéricos e códigos que identificam e corrigem erros em informações transmitidas ou armazenadas.

Os códigos numéricos estão presentes em diversas situações nos nossos hábitos corriqueiros do dia a dia, como por exemplo em sequências numéricas de códigos de barras, CPF, RG, contas bancárias e em diversas outras situações.

Os dígitos desses códigos numéricos utilizam uma congruência módulo k e um vetor de pesos estabelecido pelo sistema para verificar a presença de erros em sequências numéricas. Os dígitos conseguem identificar os erros mais comuns cometidos por falha humana e apresentam um resultado mais eficiente quando o valor de k é um número primo.

Utilizamos os códigos corretores toda vez que assistimos a um filme, enviamos um e-mail etc. Os códigos corretores estão presentes em toda informação transmitida ou armazenada por algum meio tecnológico e seu objetivo é identificar e corrigir erros causados por alguma interferência.

Os códigos corretores de erros adicionam uma redundância em uma informação de modo que seja possível decodificá-la corrigindo erros. Usamos uma transformação linear para codificar o vetor informação e depois usamos uma matriz de paridade para verificar se a informação foi recebida com erro e, caso tenha, por meio de um algoritmo conseguimos fazer a correção.

Podemos perceber que a Matemática está presente em diversas situações em que não a notamos. Pensando nisso, esta dissertação tem como objetivo mostrar essas aplicações da Matemática para que sejam usadas na sala de aula do Ensino Fundamental e Médio.

Infelizmente, é muito comum ouvir dos alunos do Ensino Fundamental e Médio que não conseguem ver aplicações para os conteúdos do currículo da Matemática, que a acham difícil e desestimulante. Mostrar uma situação em que o aluno possa aplicar e contextualizar o conteúdo da aula tem um resultado mais satisfatório e significativo no processo ensino-aprendizagem.

Esta dissertação foi dividida como segue:

No primeiro capítulo, apresentamos alguns conceitos preliminares como: Divisibilidade, Equações Diofantinas, Congruência Módulo k, Congruências Lineares, Anel e Corpo. Tais conceitos foram utilizados com ferramentas para demonstrar nos demais capítulos teoremas e proposições.

No segundo capítulo, apresentamos os dígitos que verificam erros em códigos numéricos, como por exemplo: número do código de barras, números de CPF e RG e código ISBN. Em seguida, analisamos os principais erros que podem ser cometidos por falha humana na digitação de uma sequência numérica e fizemos também uma análise para verificar a eficiência dos dígitos nos casos dos erros mais comuns.

No terceiro capítulo, apresentamos os códigos que fazem a identificação e correção dos erros que ocorrem em informações transmitidas ou armazenadas. Demonstramos os Códigos Lineares – que é a classe de códigos mais utilizada –, Distância Hamming e algoritmos para fazer a decodificação de mensagens e corrigir erros, caso existam.

Por último, no quarto capítulo, apresentamos uma sequência de etapas que permite contextualizar o ensino da Matemática no Ensino Fundamental e Médio. O objetivo desta etapa do trabalho é mostrar a Matemática presente em diversas situações no nosso cotidiano.

1

CONCEITOS PRELIMINARES

Este capítulo será destinado a definição e demonstração de conceitos que serão usados como ferramentas para embasar os códigos identificadores e corretores de erros. Um estudo mais detalhado dos assuntos deste capítulo podem ser encontradas em [1], [2], [5].

1.1 DIVISIBILIDADE

Definição 1.1. *Dados dois inteiros $a \neq 0$ e b, dizemos que b é divisível por a escrevendo $a \mid b$, quando existir $c \in \mathbb{Z}$ tal que $b = ca$. Nesse caso, podemos dizer também que a é divisor ou um fator de b, e b é múltiplo ou divisível por a. Quando b não for divisível por a, usaremos a seguinte notação: $a \nmid b$.*

Quando um inteiro a, diferente de zero, não divide um inteiro b, Euclides, nos seus *Elementos*, utiliza, sem enunciá-lo, o fato de que sempre é possível efetuar a divisão de b por a, com resto.

Teorema 1.2 (Teorema de Euclides)**.** *Sejam a e b dois inteiros com $b \neq 0$. Existem dois números inteiros q e r tais que:*

$$a = bq + r \ , \ com \ 0 \ \leq \ r \ < \ |b| \tag{1.1}$$

Nas condições do teorema acima, temos que os número r e q são chamados, respectivamente, de resto e quociente da divisão de a por b. A demonstração do Teorema de Euclides está disponível no livro Hefez [1].

1.2 MÁXIMO DIVISOR COMUM E EQUAÇÕES DIOFANTINAS

Definição 1.3. *Um número inteiro d será chamado de divisor comum de inteiros a e b se* $d \mid a$ *e* $d \mid b$.

Definição 1.4. *Seja d um inteiro positivo, dizemos que d é o máximo divisor comum de a e b, usaremos a notação* (a, b)*, se possuir as seguintes propriedades:*

i- d é um divisor comum de a e b, e

ii- d é divisível por todo divisor comum de a e b.

Apresentaremos agora, algumas propriedades do Máximo Divisor Comum de dois números inteiros. Para isso, será definido o conjunto de todas as combinações inteiras dos inteiros a e b:

$$I = \{ma + nb; m, n \in \mathbb{Z}\}$$

Teorema 1.5 (Teorema de Bézout)**.** *Se* $d = (a, b)$*, então existem* $m, n \in \mathbb{Z}$ *tais que* $am + bn = d$.

Demonstração. Seja p o menor elemento positivo de I, $p = ma + nb$. Vamos considerar o resto e quociente da divisão de a por p como r e q respectivamente:

$$\begin{aligned} a = pq + r, \ com \ 0 \leq r < p \\ r = a - pq \\ r = a - (am + nb)q \\ r = a(1 - mq) + b(-nq) \end{aligned}$$

Se fosse $r > 0$, teríamos $r \in I$, contradizendo a hipótese de p ser o menor elemento do elemento do conjunto I. Logo $r = 0$ e $p \mid a$. De modo análogo, $p \mid b$ e, pela Definição 1.3, $p \mid d$.

Agora, vamos provar que $d \mid p$. Pela Definição 1.3, temos que $d \mid a$ e $d \mid b$, neste caso podemos escrever $a = a_1 d$ e $b = b_1 d$ para $a_1, b_1 \in \mathbb{Z}$. E

$$\begin{aligned} p = ma + nb \\ p = ma_1 d + nb_1 d \\ p = d(a_1 m + b_1 n) \\ d \mid p \end{aligned}$$

Concluímos então que $p = d$. □

Observe que da demonstração acima, segue que se d é o menor elemento positivo de I, então $d = (a, b)$.

Definição 1.6. *Dados dois inteiros a e b, dizemos que eles são primos entre si se* $(a, b) = 1$.

A seguir apresentaremos as equações diofantinas lineares que ajudam a resolver vários problemas de aritmética e será usada na próxima seção para demonstrar o Teorema 1.15 que vai ser fundamental do Capítulo 2.

Proposição 1.7. *A equação* $aX + bY = c$, *com* $a, b, c \in \mathbb{Z}$, *admite solução em inteiros se, somente se,* $(a, b) \mid c$.

Demonstração. Vamos supor que (x_0, y_0) seja uma solução da equação e $(a, b) = d$, sabemos que $a = a_1 d$ e $b = b_1 d$, logo:

$$\begin{aligned} ax_0 + by_0 &= c \\ a_1 x_0 d + b_1 y_0 d &= c \\ d(a_1 x_0 + b_1 y_0) &= c \\ d &\mid c \end{aligned}$$

Reciprocamente, se $d \mid c$, então existe $c_1 \in \mathbb{Z}$ tal que $c = dc_1$. O Teorema 1.5 garante a existência de um par (x_0, y_0) tal que

$$ax_0 + by_0 = d$$

Multiplicando ambos os lados da equação acima por c_1

$$\begin{aligned} c_1 a x_0 + c_1 b y_0 &= d c_1 \\ c_1 a x_0 + c_1 b y_0 &= c_1 d = c \end{aligned}$$

donde obtemos a solução $(c_1 x_0, c_1 y_0)$ da equação diofantina. □

Podemos estender os resultados acima para n variáveis. Para iniciar, generalizamos a definição do mdc para n números inteiros:

Definição 1.8. *Seja d um inteiro positivo, dizemos que d é o máximo divisor comum de $a_1, ..., a_n$, se:*

i- d é um divisor comum de $a_1, ..., a_n$;

ii- d é divisível por todo divisor comum de $a_1, ..., a_n$.

Iremos indicar o mdc de n inteiros com a seguinte notação $(a_1, ..., a_n)$. Segue facilmente da definição que

$$(a_1, \ldots, a_{n-1}, a_n) = (a_1, \ldots (a_{n-1}, a_n))$$

Da igualdade acima, usando indutivamente a demonstração do Teorema de Bezout, obtemos a versão deste teorema para n inteiros:

Teorema 1.9 (Teorema de Bézout (generalizado))**.** *Se $d = (a_1, a_2, \ldots, a_n)$, então existem $m_1, m_2, \ldots, m_n \in \mathbb{Z}$ tais que $m_1a_1 + m_2a_2 + \cdots + m_na_n = d$.*

Por fim, para equações diofantinas de n variáveis temos o seguinte resultado:

Proposição 1.10. *A equação diofantina $a_1x_1 + ... + a_nx_n = c$, $a_1 \in \mathbb{Z}$ e $c \in \mathbb{Z}$, admite solução se, e somente se, $(a_1, ..., a_n)$ divide c.*

A demonstração é análoga à do caso de duas variáveis.

1.3 CONGRUÊNCIA

A congruência módulo m é uma das ferramentas mais importantes na Teoria dos Números, que chamamos de Matemática Modular ou Aritmética dos Restos. Uma congruência é uma relação entre dois inteiros que quando divididos por um terceiro deixam o mesmo resto, este terceiro inteiro é chamado de módulo da congruência. Nesta seção, será apresentado sua definição e propriedades de acordo com a necessidade de elucidar suas aplicações, para oportunizar melhor entendimento no próximo capítulo, onde as aplicaremos.

A relação de congruência módulo m entre dois inteiros está relacionada com a divisão euclidiana desses por um terceiro número fixo, e foi introduzida por Gauss no seu livro *Disquisitiones Arithmeticae* em 1801.

Definição 1.11. *Seja m um número natural maior do que 1, dizemos que dois números inteiros a e b são congruentes módulo m se os restos de sua divisão por m são iguais. Quando dois inteiros são congruentes módulo m, escreve-se $a \equiv b \ mod \ m$.*

Observe o seguinte exemplo: $31 \equiv 21 \ mod \ 10$, pois a divisão de 31 por 10 deixa o mesmo resto que a de 21 por 10.

Para verificar se dois inteiros a e b são congruentes *mod* m, não é necessário fazer a divisão euclidiana de ambos por m, é suficiente usar a seguinte proposição:

Proposição 1.12. *Tem-se $a \equiv b \ mod \ m$ se, somente se, $m \mid (a - b)$.*

Demonstração. Se $a \equiv b \ mod \ m$, então existem f, p e r inteiros, tais que $a = f.m + r$ e $b = p.m + r$. Subtraindo uma da outra $a - b = m.(f - p)$ logo, pela Definição 1.1 temos que $m \mid (a - b)$.

Reciprocamente, suponhamos que $m \mid (a - b)$ pela divisão euclidiana temos que $a = f.m + r$ com $0 \leq r < m$ e $b = m.p + q$ com $0 \leq q < m$. Subtraindo as equações encontramos $a - b = m.(f - p) + (r - q)$, como $m \mid (a - b)$ concluímos que $m \mid (r - q)$ e $r = q$, pois $\mid r - q \mid < m$.

Portanto $a \equiv b \ mod \ m$. □

A seguir, temos algumas propriedades que são de grande utilidade, pois serão usadas para demonstrar algumas aplicações no Capítulo 2.

Proposição 1.13. *Sejam a, b, c, d, m, n inteiros com $m > 1$ e $n \geq 1$. Temos as seguintes propriedades:*

(i) $a \equiv a \ mod \ m$;

(ii) *Se* $a \equiv b \ mod \ m$, *então* $b \equiv a \ mod \ m$;

(iii) *Se* $a \equiv b \ mod \ m$ *e* $b \equiv c \ mod \ m$, *então* $a \equiv c \ mod \ m$;

(iv) *Se* $a \equiv b \ mod \ m$ *e* $c \equiv d \ mod \ m$, *então* $(a + c) \equiv (b + d) \ mod \ m$;

(v) *Se* $a \equiv b \ mod \ m$ *e* $c \equiv d \ mod \ m$, *então* $a.c \equiv b.d \ mod \ m$;

(vi) *Se* $a \equiv b \ mod \ m$, *então* $a^n \equiv b^n \ mod \ m$.

Demonstração. As demostrações de i e ii são triviais.

iii - Se $a \equiv b \ mod \ m$ e $b \equiv c \ mod \ m$, então $m \mid (a-b)$ e $m \mid (b-c)$, logo $m \mid (a-b+b-c)$. Portanto $m \mid (a-c)$, ou seja, $a \equiv c \ mod \ m$.

iv - Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, temos por definição que $m \mid (a-b)$ e $m \mid (c-d)$, usando as propriedades da divisão de Euclides podemos concluir que $m \mid (a-b+c-d)$ ou podemos escrever da seguinte forma $m \mid ((a+c)-(b+d))$ o que leva a concluir que $(a+c) \equiv (b+d) \ mod \ m$.

v - Se $a \equiv b \ mod \ m$ e $c \equiv d \ mod \ m$, temos que $m \mid (a-b)$ e $m \mid (c-d)$. Observe que:

$$a.c - b.d = a(c-d) + d(a-b)$$

Logo, como $m \mid (c-d)$ e $m \mid (a-b)$ podemos conluir que $m \mid (a.c - b.d)$. Portanto $a.c \equiv b.d \ mod \ m$.

vi - Para demonstrar está propriedade será usado indução sobre n.

Seja $P(n)$: $a^n \equiv b^n \ mod \ m$.

$P(n)$ é válida para $n = 1$, pois $a \equiv b \ mod \ m$ por hipótese. Suponha $P(k)$ válida para algum k natural. Temos que:

$$a^k \equiv b^k \ mod \ m.$$

Temos pela hipótese de indução que:

$$a^k \equiv b^k \ mod \ m \tag{1.2}$$

E por hipótese:

$$a \equiv b \ mod \ m \tag{1.3}$$

Aplicando a propriedade v nas equações 1.2 e 1.3 temos:

$$a^{k+1} \equiv b^{k+1} \ mod \ m$$

Portanto, por indução $P(n)$ é válida para todo n natural.

□

Observação 1.14. *De acordo com as propriedades i, ii e iii, podemos concluir que uma relação de congruência módulo m é uma relação de equivalência nos inteiros.*

E agora, apresentaremos dois teoremas que serão fundamentais no próximo capítulo:

Teorema 1.15. *A congruência linear $ax \equiv b \ mod \ m$ admite solução se, somente se, $(a, m) \mid b$.*

Demonstração. Vamos supor que a equação tenha solução, temos que:

$$ax \equiv b \ mod \ m$$
$$m \mid (ax - b)$$

Logo, existe um inteiro t, tal que:

$$mt = ax - b$$

$$b = mt - ax \tag{1.4}$$

Seja o $(a, m) = k$, então $k \mid a$ e $k \mid m$, logo existem $p, d \in \mathbb{Z}$:

$$a = kd \ e \ m = kp \tag{1.5}$$

Substituindo as equações 1.4 e 1.5, temos:

$$b = kpt - kdx$$
$$b = k(pt - dx)$$

E portanto $k \mid b$

Reciprocamente, se $(a, m) \mid b$, a equação $b = mt - ax$ sempre admite solução inteira, por consequência da Proposição 1.7. □

Teorema 1.16. *Sejam a e x números inteiros, temos $ax \not\equiv 0 \ mod \ m$, para todo $x \in \{1, ..., m - 1\}$ se, e somente se, $(a, m) = 1$.*

Demonstração. Suponhamos $(a, m) = d > 1$, logo $d \mid a$ e $d \mid m$ e assim $a = dd_1$ e $m = dd_2$ com $d_2 \in \{1, ..., m-1\}$. Fazendo $d_2 = x$, encontramos o seguintes resultado:

$$ax = ad_2 = d_1 dd_2 = d_1 m \equiv 0 \ mod \ m$$

O que é um absurdo pois $ax \not\equiv 0 \ mod \ m$ e portanto $d = 1$.

Reciprocamente, se $(a, m) = 1$ e seja $x \in \{1, ..., m-1\}$ tal que $ax \equiv 0 \ mod \ m$, temos que $m \mid ax$, portanto $m \mid x$ o que seria um absurdo. Logo $ax \not\equiv 0 \ mod \ m$ para $x \in \{1, ..., m-1\}$. □

1.4 ANÉIS E CORPOS

Nesta seção, serão definidos os conceitos de anel e corpo, que serão usados como ferramentas no capítulo 3.

Definição 1.17. *Um anel é um conjunto A munido das operações adição (+) e multiplicação ($\times$):*

$$\begin{array}{rccccrccc} + : & A \times A & \to & A & e & \times : & A \times A & \to & A \\ & (a, b) & \to & a + b & & & (a, b) & \to & a \times b \end{array}$$

Que satisfazem as seguintes propriedades:

1. *Associatividade da adição:*

$$\forall a,\ b,\ c\ \in\ A,\ (a + b) + c = a + (b + c)$$

2. *Existência de um elemento neutro para a adição, ou seja, existe um elemento chamado zero e denotado por 0, tal que:*

$$\forall\ a\ \in\ A,\ a + 0 = 0 + a = a$$

3. *Existência de um elemento oposto para a adição, ou seja, existe $(-a)\ \in\ A$, tal que:*

$$a + (-a) = (-a) + a = 0$$

4. Comutatividade da adição:

$$\forall\ a,\ b\ \in\ A,\ a+b=b+a$$

5. Associatividade da multiplicação:

$$\forall a,\ b,\ c\ \in\ A,\ (a\times b)\times c=a\times(b\times c)$$

6. Distributividade da multiplicação em relação a adição:

$$\forall a,\ b,\ c\ \in\ A,\ a\times(b+c)=a\times b+a\times c$$

Exemplo 1.1. *O conjunto dos números inteiros $\mathbb{Z}$, racionais $\mathbb{Q}$ e reais $\mathbb{R}$, munido das operações usuais de adição e multiplicação são exemplos de anéis .*

Definição 1.18. *Se um anel A possuir um elemento neutro para a multiplicação, denotado 1, dizemos que A é um anel com unidade.*

Definição 1.19. *Dado um elemento a de um anel com unidade A, dizemos que será invertível se existir um elemento $b \in A$, tal que $a \times b = 1$. Quando isso ocorre, dizemos que b é um inverso de a.*

Definição 1.20. *Um anel com unidade onde todo elemento não nulo é invertível é chamado de corpo.*

Exemplo 1.2. *O conjunto dos números reais $\mathbb{R}$, juntamente com as operações usuais de adição e multiplicação é um exemplo de corpo.*

Exemplo 1.3. *O conjunto $A = \{0, 1\}$ munido das operações da adição e multiplicação definidas abaixo é um corpo.*

$+$	0	1
0	0	1
1	1	0

e

$\times$	0	1
0	0	0
1	0	1

1.5 CLASSES RESIDUAIS

Nesta seção, apresentaremos o conceito de classes residuais de $\mathbb{Z}$ módulo m.

Qualquer relação de congruência módulo m define uma relação de equivalência nos inteiros, e essas classes de equivalência são chamadas de classes de congruência ou classes residuais.

Definição 1.21. *A classe residual módulo m de um elemento $a \in \mathbb{Z}$ é o conjunto:*

$$[a] = \{x \in \mathbb{Z}; x \equiv a \ mod \ m\}$$

O elemento a será chamado de representante da classe residual $[a]$.

O conjunto de todas as classes residuais módulo m, indicado por $\mathbb{Z}_m$, é:

$$\mathbb{Z}_m = \{[0], [1], ..., [m-1]\}$$

Temos que $\mathbb{Z}_m$ é um anel (com unidade) finito com m elementos.

Exemplo 1.4. *Seja $m = 2$. Logo temos que $\mathbb{Z}_2 = \{[0], [1]\}$ com as seguintes operações*

$+$	$[0]$	$[1]$
$[0]$	$[0]$	$[1]$
$[1]$	$[1]$	$[0]$

$\times$	$[0]$	$[1]$
$[0]$	$[0]$	$[0]$
$[1]$	$[0]$	$[1]$

é um anel. Temos que $[1]$ *é o único elemento não nulo de $\mathbb{Z}_2$ e é invertível, logo $\mathbb{Z}_2$ é um corpo.*

Exemplo 1.5. *Seja $m = 3$, temos que $\mathbb{Z}_3 = \{[0], [1], [3]\}$ com as operações e adições da tabela abaixo, como* $[1]$ *e* $[2]$ *são invertíveis, temos que $\mathbb{Z}_3$ é um corpo.*

$+$	$[0]$	$[1]$	$[2]$
$[0]$	$[0]$	$[1]$	$[2]$
$[1]$	$[1]$	$[2]$	$[0]$
$[2]$	$[2]$	$[0]$	$[1]$

$\times$	[0]	[1]	[2]
[0]	[0]	[0]	[0]
[1]	[0]	[1]	[2]
[2]	[0]	[2]	[1]

2

SISTEMAS DE IDENTIFICAÇÃO MODULARES

No nosso cotidiano, quando utilizamos meios tecnológicos para digitar uma informação podem ocorrer erros na digitação ou na transmissão. Neste capítulo será feita uma análise dos dígitos que verificam a existência de erro na digitação de uma sequência numérica e no próximo capítulo apresentaremos os códigos que fazem correção de erros que podem acontecer na transmissão de uma informação.

Os dígitos verificadores, presentes em códigos numéricos, aparecem em diversas situações no nosso dia a dia, como por exemplo, nos números de uma conta bancária, nos algarismos do CPF, RG, título de eleitor, número do cartão de crédito, certidão de nascimento, matrículas de IPTU, códigos de barras e ISBN. Nesses exemplos, usamos os dígitos para identificar incorreções. No decorrer deste capítulo, mostraremos essas aplicações da Matemática Modular ou Matemática dos Restos.

Códigos numéricos que utilizam o dígito verificador de erros apresentam um conjunto de regras para estar correto. Se a sequência do código não estiver de acordo com as regras pré estabelecidas pelo sistema, o dígito verificador indicará que a sequência apresenta números inconsistentes. Importante ressaltar que, nesse caso, não é feita a correção automática e não é indicado qual o algarismo está incorreto, e pode ser necessário a verificação de todos os algarismos informados. A ideia principal do dígito é adicionar um número suplementar na sequência original do código e estabelecer uma relação, que seja única, entre os algarismos da sequência e o dígito.

2.1 ANÁLISE GERAL DOS SISTEMAS DE IDENTIFICAÇÃO MODULARES

Os sistemas de identificação modulares apresentam características semelhantes. Nesta seção, será apresentada uma análise geral desses sistemas que utilizam os dígitos como identificador ou verificador de erros em sequências numéricas.

Todos os códigos numéricos que utilizam a matemática modular para fazer a identificação de erros apresentam o seguinte formato:

$$x_1x_2x_3....x_nD \tag{2.1}$$

Onde D representa o dígito adicional que fará a verificação dos demais algarismos digitados, D é chamado de dígito verificador de erros. Considere os algarismos da sequência $x_1x_2x_3....x_nD$ como o vetor $\alpha = (x_1, x_2, x_3, ..., x_n, D)$.

Para determinar o valor numérico de D, utilizamos um vetor de pesos, pré estabelecido, $\lambda = (p_1, p_2, ..., p_n, p_{n+1})$, e os relacionamos aos vetores α e λ através de um produto escalar congruente a zero módulo k, sendo k um número natural pré estabelecido pelo sistema de verificação e $p_1, ..., p_{n+1} \in \mathbb{Z}_k$

$$\alpha.\lambda = (x_1, x_2, x_3, ..., x_n, D).(p_1, p_2, ...p_n, p_{n+1}) \equiv\ 0\ mod\ k \tag{2.2}$$

A equação acima sempre apresentará solução inteira, a partir de oportuna escolha dos pesos e de k. Como se trata, fundamentalmente, de uma diofantina de duas variáveis

$$Xp_{n+1} + Yk = c(= x_1p_1 + \cdots + x_np_n)$$

pela Proposição 1.7 basta tomar $(p_{n+1}, k) = 1$. Os dígitos que usam este tipo de sistema são chamados de dígitos de verificação módulo k.

2.1.1 *Códigos de barras*

O código de barras é usado mundialmente para identificar produtos, pois oferecem vantagens como maior eficiência para controlar estoque e maior agilidade no atendimento. O código de barras passou por algumas modificações ao longo dos anos, o mais utilizado é o EAN 13, *Europen Article Numbering*, que é uma sequência numérica de treze algarismos, e nos EUA e Canadá, usam o UPC, *Universal Product Code*, uma sequência numérica de doze dígitos.

Nesta seção, será mostrado o funcionamento do dígito verificador de erro, que é o décimo terceiro dígito do código de barras EAN 13, e será apresentada sua estrutura de organização.

O código de barras EAN 13, da esquerda para direita, é dividido em partes que permite identificar o país de origem, a empresa fabricante e o produto. O último algarismo representa o dígito verificador, responsável por validar as informações do código, como na figura 2.1. [1]

Figura 2.1: Exemplo de um código de barras

Para determinar a relação entre o dígito e as informações do código usadas para fazer a identificação das informações do EAN 13, vamos representar os seus algarismos na forma de vetor α:

$$\alpha = (a_1, a_2, a_3, a_4, ..., a_{11}, a_{12}, a_{13}) \tag{2.3}$$

Relacionamos o vetor α com um vetor de pesos fixo $\lambda = (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1)$ de acordo com a equação 2.2, o EAN 13 utiliza um sistema de verificação módulo 10.

$$\alpha.\lambda = (a_1, a_2, ..., a_{12}, a_{13}).(1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) \equiv \ 0 \ mod \ 10 \tag{2.4}$$

Exemplo 2.1. *Usando o código de barras da figura* (2.1)*, temos que os doze primeiros algarismos são 560103610017 e o dígito verificador é 4.*

1 Disponível em http://origemdascoisas.com/a-origem-do-codigo-de-barras/ acesso em 09/01/2017

Podemos verificar que com este valor numérico os algarismos do código EAN 13 estão corretos, para isso consideremos $\alpha = (5, 6, 0, 1, 0, 3, 6, 1, 0, 0, 1, 7, a_{13})$ *e usando o vetor de pesos a equação tem solução para* $a_{13} = 4$*, pois* $0 \leq a_{13} \leq 9$*.*

$$\begin{aligned}
(5, 6, 0, 1, 0, 3, 6, 1, 0, 0, 1, 7, a_{13}).(1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) &\equiv 0 \ mod\ 10 \\
5.1 + 6.3 + 0.1 + 1.3 + 0.1 + 3.3 + 6.1 + 1.3 + 0.1 + 0.3 + 1.1 + 7.3 + a_{13}.1 &\equiv 0 \ mod\ 10 \\
5 + 18 + 0 + 3 + 0 + 9 + 6 + 3 + 0 + 0 + 1 + 21 + a_{13} &\equiv 0 \ mod\ 10 \\
66 + a_{13} &\equiv 0 \ mod\ 10
\end{aligned}$$

2.1.2 *Sistema ISBN*

O sistema ISBN - Internacional Standard Book Number - foi criado em 1967 com a finalidade de identificar, de forma numérica, livros, CD-ROMs e publicações em Braile segundo o título, o autor, o país e a editora, individualizando-os por edição.[2]

Desde a sua criação até 01 de janeiro de 2007, os códigos ISBN tinham um formato com dez algarismos. A partir desta data os números ISBN passaram a ter 13 algarismos. Um ISBN é um código constituído por cinco partes que permitem identificar o país, editor e título conforme figura 2.2.[3]

2 Disponível em http://www.isbn.bn.br/website/ acesso em 13/11/2016

3 Disponível em http://isbn.world/7476217384691273498125298343876 49/sobre˙isbn.html Acesso em 13/11/2016.

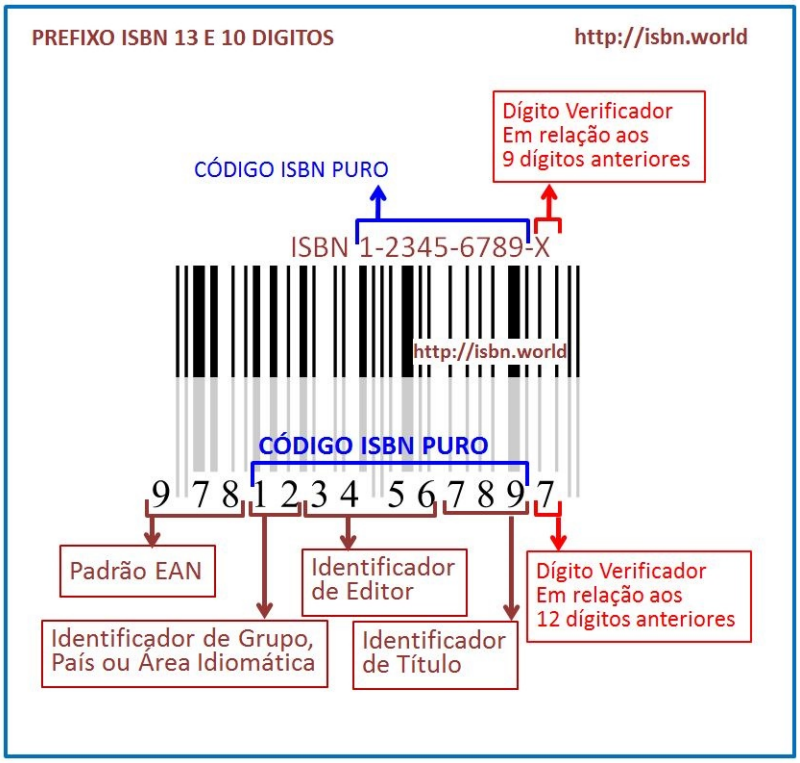

Figura 2.2: Exemplo de um código ISBN

O sistema ISBN de treze dígitos $\alpha = (a_1, a_2, ..., a_{12}, a_{13})$ é semelhante ao código de barras EAN 13 , da seção 2.1.1, para se determinar o valor numérico do dígito verificador, utiliza-se um vetor peso $\lambda = (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1)$, e o produto escalar entre α e λ deve ser congruente a zero módulo 10.

$$\alpha.\lambda = (a_1, a_2, ..., a_{12}, a_{13}).(1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) \equiv 0 \ mod \ 10 \tag{2.5}$$

Exemplo 2.2. *O livro A Década de 50 - Populismo e metas desenvolvimentistas no Brasil escrito por Marly Rodrigues tem como ISBN 9788508042173, sendo o ultimo algarismo o dígito verificador.*

Fazendo a verificação, o dígito 3 é o único algarismo que torna verdadeira os algarismos de identificação do código ISBN pois $0 \leq a_{13} \leq 9$ e:

$$(9,7,8,8,5,0,8,0,4,2,1,7,a_{13}).(1,3,1,3,1,3,1,3,1,3,1,3,1) \equiv 0 \ mod\ 10$$
$$9.1 + 7.3 + 8.1 + 8.3 + 5.1 + 0.3 + 8.1 + 0.3 + 4.1 + 2.3 + 1.1 + 7.3 + a_{13}.1 \equiv 0 \ mod\ 10$$
$$9 + 21 + 8 + 24 + 5 + 0 + 8 + 0 + 4 + 6 + 1 + 21 + a_{13} \equiv 0 \ mod\ 10$$
$$107 + a_{13} \equiv 0 \ mod\ 10$$

2.1.3 *Dígitos do CPF - Cadastro de pessoas físicas na Receita Federal*

O número do CPF de uma pessoa no Brasil é composto por onze algarismos divididos em duas seções. A primeira, com nove números, e a segunda, por dois, que são os dígitos de verificação que são calculados separadamente.

Na primeira seção, os oito primeiros números do CPF são gerados aleatoriamente e nono algarismo indica o Estado da região fiscal responsável pela inscrição. Considere, por exemplo, o CPF $XXX.XXX.XX8 - XX$, como o nono algarismo é 8, temos que esta pessoa pertence à região de São Paulo. Outras regiões usam outros números de 0 até 9 para a sua identificação, por exemplo, o número 9 indica as regiões de Paraná e Santa Catarina.

Na segunda parte dos algarismos do CPF, o décimo e o décimo primeiro algarismo representam, respectivamente, os dois dígitos verificadores. O primeiro dígito, verifica a validade dos nove primeiros algarismos e o segundo tem a finalidade de verificar os dez primeiros e identificar um erro, que talvez, não seja detectado no primeiro dígito.

Para calcular os dois dígitos verificadores do CPF, usamos uma relação de congruência módulo 11.

Considere o seguinte número de CPF $\alpha = (a_1, ..., a_9, a_{10}, a_{11})$, sendo a_{10} e a_{11}, respectivamente, o primeiro e o segundo dígito verificador.

Para determinar o valor numérico de a_{10}, utilizamos o vetor de pesos

$$\lambda_1 = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$$

e o vetor $\alpha_1 = (a_1, ..., a_9, a_{10})$, que são os dez primeiros números de α. Realizamos a seguinte operação:

$$\lambda_1.\alpha_1 \equiv 0 \ mod\ 11 \tag{2.6}$$

Para determinar o valor de a_{11}, utilizamos o vetor de pesos $\lambda_2 = (11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$ e o vetor $\alpha_2 = a_1, ..., a_9, a_{10}, a_{11})$. Realizamos a seguinte operação:

$$\lambda_2.\alpha_2 \equiv 0 \ mod \ 11 \tag{2.7}$$

Observação 2.1. *O número do CPF utiliza dígitos de verificação de erros módulo* 11 *restrito, nesse caso utiliza-se o algarismo* 0 *para representar o número* 10.

Exemplo 2.3. *Considere a seguinte numeração de um CPF:* $395725638a_{10}a_{11}$ *, neste caso, para encontrar o valor numérico dos dois dígitos verificadores, devemos realizar as seguintes operações:*

Primeiramente, vamos calcular o valor de a_{10}*:*

$$\begin{aligned}(3, 9, 5, 7, 2, 5, 6, 3, 8, a_{10}).(10, 9, 8, 7, 6, 5, 4, 3, 2, 1) &\equiv 0 \ mod 11\\ 3.10 + 9.9 + 5.8 + 7.7 + 2.6 + 5.5 + 6.4 + 3.3 + 8.2 + a_{10}.1 &\equiv 0 \ mod 11\\ 30 + 81 + 40 + 49 + 12 + 25 + 24 + 9 + 16 + a_{10} &\equiv 0 \ mod 11\\ 286 + a_{10} &\equiv 0 \ mod 11\end{aligned}$$

Como $0 \leq a_{10} \leq 10$*, temos que* $a_{10} = 0$*, pois 286 é múltiplo de 11.*

Para determinar o valor de a_{11}*, usaremos os nove primeiros dígitos e o décimo encontrado na equação anterior.*

$$\begin{aligned}(3, 9, 5, 7, 2, 5, 6, 3, 8, 0, a_{11}).(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) &\equiv 0 \ mod 11\\ 3.11 + 9.10 + 5.9 + 7.8 + 2.7 + 5.6 + 6.5 + 3.4 + 8.3 + 0.2 + a_{11}.1 &\equiv 0 \ mod 11\\ 33 + 90 + 45 + 56 + 14 + 30 + 30 + 12 + 24 + 0 + a_{11} &\equiv 0 \ mod 11\\ 334 + a_{11} &\equiv 0 \ mod 11\end{aligned}$$

Como $0 \leq a_{11} \leq 10$*, temos que* $a_{11} = 7$*.*

Podemos concluir que, neste exemplo, o número do CPF com os dígitos verificadores é 395 725 638 - 07.

Quando digitamos os algarismos do CPF em algum aparelho eletrônico, é feita a verificação dos dígitos informados usando os dois dígitos verificadores. Caso algum número seja digitado errado, aparece uma mensagem informando a invalidade dos números.

2.2 ERROS DETECTÁVEIS E NÃO DETECTÁVEIS

Nesta seção, vamos fazer uma análise dos erros que podem ser ou não detectados pelo dígito verificador. Quando fazemos o registro manual de um código numérico, podemos cometer vários equívocos na digitação e nem sempre o dígito consegue identificar a presença destes erros.

Exemplo 2.4 (Erro detectável)**.** *Consideremos o código de barras EAN 13 da figura 2.1 cuja sequência é 5601036100174 seja digitado com um erro único 5602036100174.*

$$(5,6,0,2,0,3,6,1,0,0,1,7,4).(1,3,1,3,1,3,1,3,1,3,1,3,1) \equiv 0 \ mod\ 10$$
$$5+18+0+6+0+9+6+3+0+0+1+21+4 = 73 \not\equiv 0 \ mod\ 10$$

Como o resultado obtido não é múltiplo de 10, será emitido um aviso informando um erro nos números digitados.

Exemplo 2.5 (Erro não detectável)**.** *Usando as informações do exemplo anterior, suponha a mesma sequência com dois erros de digitação 5602036170174.*

$$(5,6,0,2,0,3,6,1,7,0,1,7,4).(1,3,1,3,1,3,1,3,1,3,1,3,1) \equiv 0 \ mod\ 10$$
$$5+18+0+6+0+9+6+3+7+0+1+21+4 = 80 \equiv 0 \ mod\ 10$$

Os dois algarismos que foram digitados errados se compensam e o dígito verificador não detecta a presença de erros.

O matemático holandês Jacobus Koos Verhoef, nascido em 1927, citado na dissertação de mestrado da autora Carla Rejane Fick Pinz [6], 2013 , pesquisou os principais erros cometidos por falha humana, e os desacertos cometidos ao digitar uma sequência numérica podem ser por digitação errada de um único termo, transposição adjacente, transposição alternada, erro gêmeo, erro gêmeo alternado e outros que representa a soma de erros com frequência menor de 1%.

Tipo de erro		%
Erro único	$... \ a \mapsto b \ ...$	79
Transposição adjacente	$ab \mapsto ba$	10,2
Transposição alternada	$abc \mapsto cba$	0,8
Erro gêmeo	$aa \mapsto bb$	0,6
Erro gêmeo alternada	$aba \mapsto cbc$	0,3
Outros		9,1

Tabela 1: Tipos de erros e suas frequências

Será feito uma análise da eficiência dos dígitos verificadores módulo k na presença de erro único, transposição adjacente e transposição alternada que são os principais erros que possivelmente podem ser cometidos por falha humana. Estes desacertos representam aproximadamente 90% dos erros mais comuns.

2.2.1 *Análise de dígitos de verificação módulo k*

Nesta subseção, será feita uma análise geral dos sistemas modulares de identificação módulo k.

Para que o dígito verificador faça a identificação de erros de transposição adjacente, que pesquisas indicam ser aproximadamente 10% dos erros mais comuns, o sistema de verificação módulo k utiliza o vetor de pesos com entradas diferentes nas posições adjacentes, pois, caso contrário, uma simples troca da posição dos algarismos não altera o valor da soma e o resultado continua sendo congruente a 0 módulo k.

$$a_1 + a_2 + a_3 + ... + a_{10} + a_{11} + a_{12} + a_{13} \equiv \ 0 \ mod \ 10$$

A seguir, apresentaremos dois teoremas que estabelecem uma condição entre o vetor de pesos e k para que os dígitos detectem erro único e erros de transposição.

Teorema 2.2. *Um dígito verificador de erros módulo k, com vetor de pesos* $(p_1, ..., p_n)$*, detecta todo erro único* $\alpha_i \to \alpha_i^{'}$*, na i-ésima posição se, e somente se, mdc* $(p_i, k) = 1$.

Demonstração. Por hipótese, temos um erro único na i-ésima posição $\alpha_i \to \alpha_i^{'}$, com $\alpha_i, \alpha_i^{'} \in \{0, ..., k-1\}$. O dígito identificará a presença do erro se $p_i(\alpha_i - \alpha_i^{'}) \not\equiv 0 \ mod \ k$, portanto, pelo Teorema 1.16, *mdc* $(p_i, k) = 1$.

Reciprocamente, se $mdc\ (p_i, k) = 1$ e se existissem diferentes α_i, $\alpha_i^{'} \in \{1, ..., k-1\}$ tais que $k \mid p_i(\alpha_i - \alpha_i^{'})$ teríamos $k \mid (\alpha_i - \alpha_i^{'})$ o que seria um absurdo, pois $\alpha_i \neq \alpha_i^{'}$. □

Teorema 2.3. *Um dígito verificador de erros módulo k, com vetor de pesos* $(p_1, ..., p_i, ..., p_j, ..., p_n)$, *detecta todos os erros de transposição dos algarismos* α_i *e* α_j, *nas posições i e j se, e somente se,* $mdc\ (p_i - p_j, k) = 1$.

Demonstração. Sem o erro de transposição nas posições i e j, teríamos a seguinte equação:

$$\alpha_1 p_1 + ... + \alpha_i p_i + ... + \alpha_j p_j + ... + \alpha_n p_n \equiv 0\ mod\ k \tag{2.8}$$

Por hipótese, teve um erro de transposição nas posições i e j, logo, o dígito não identificará a presença deste erro se:

$$\alpha_1 p_1 + ... + \alpha_j p_i + ... + \alpha_i p_j + ... + \alpha_n p_n \equiv 0\ mod\ k \tag{2.9}$$

Subtraindo as equações 2.8 e 2.9, encontramos:

$$(p_i - pj)(\alpha_i - \alpha_j) \equiv 0\ mod\ k \tag{2.10}$$

Neste caso o dígito detecta a presença de erros de transposição dos algarismos nas posições i e j se, e somente se, para quaisquer $\alpha_i, \alpha_j \in \{1, ..., k-1\}$ com $\alpha_i \neq \alpha_j$, se $(p_i - p_j)(\alpha_i - \alpha_j) \not\equiv\ 0\ mod\ k$, portanto, pelo Teorema 1.16, $mdc\ (p_i - p_j, k) = 1$.

Reciprocamente, se $mdc\ (p_i - p_j, k) = 1$ e $(p_i - p_j)(\alpha_i - \alpha_j) \equiv\ 0\ mod\ k$, temos que $k \mid (\alpha_i - \alpha_j)$ o que é um absurdo pois $\alpha_i, \alpha_j \in \{1, ..., k-1\}$.

□

Podemos concluir que os dígitos verificadores de erros módulo k, apresenta um melhor resultado para $k = 11$ devido a facilidade de encontrar pesos primos com 11. Para $k = 10$ o dígito apresenta uma desvantagem, pois o dígito não consegue satisfazer simultaneamente os dois teoremas, pois se os pesos são ímpares para atender o primeiro o segundo não é verificado, uma vez que a diferença de dois números ímpares resulta em número par. Não é muito utilizado valores de $k < 10$ devido ao tamanho reduzido de algarismos que podem ser usados sendo impossível verificar todos erros únicos e de transposição.

2.2.2 *Análise de dígitos de verificação módulo 10*

Mostraremos nesta subseção, uma análise da eficiência dos dígitos verificadores que usam o módulo 10. Analisaremos os principais erros que podem ser cometidos por falha humana, que são os erros de transposição e erro único.

Quando digitamos um código de barras EAN 13 ou outra sequência numérica que usa o módulo 10, podemos cometer alguns erros que o dígito verificador consegue identificá-los, e, em outros casos, isto não é possível como mostrados nos Exemplos 2.4 e 2.5.

No Exemplo 2.4, a sequência numérica apresenta um erro único, fato que Verhoef indica ser o erro mais comum, e o dígito é eficiente para identificar o erro . Tal fato ocorre, pois pelo Teorema 2.2, os pesos utilizados no EAN 13 são primos com 10.

Caso seja cometido mais de um erro, o dígito pode ser eficiente para identificá-lo, mas não com toda certeza, pois os números poderiam se compensar mutuamente e a soma continuar sendo múltiplo de 10. Fato que acontece no exemplo 2.5.

Existem erros de transposição nos quais não é possível identificar que a sequência está incorreta.

Exemplo 2.6. *Considere um código de barras 7896283800245 que, ao ser digito, foi cometido um erro de transposição adjacente 7896238800245, temos:*

$$(7,8,9,6,2,3,8,8,0,0,2,4,5).(1,3,1,3,1,3,1,3,1,3,1,3,1) \equiv 0 \ mod\ 10$$
$$7+24+9+18+2+9+8+24+0+0+2+12+5 = 120 \equiv 0 \ mod\ 10$$

Nesse exemplo 2.6 o dígito é ineficiente para identificar a presença de erro. Isto ocorre, pelo fato do Teorema 2.3 não ser satisfeito, pois a diferença de pesos ímpares é um número par e com isso não é primo com 10. A seguir, apresentaremos um Teorema específico para o sistema de identificação módulo 10, para que seja possível identificar um erro de transposição:

Teorema 2.4. *Uma única transposição adjacente é detectada pelo dígito de sistema de verificação módulo* 10 *se* $|a_i - a_{i+1}| \neq 5$

Demonstração. Seja $a_1, a_2, ..., a_i, a_{i+1}, ..., a_{12}, a_{13}$ uma sequência de algarismos de um código de barras EAN 13. Temos que:

$$a_1 + 3.a_2 + a_3 + ... + 3.a_i + a_{i+1} + ... + 3.a_{12} + a_{13} \equiv 0 \ mod\ 10 \tag{2.11}$$

Considere um erro de transposição adjacente, logo: $a_1, a_2, ..., a_{i+1}, a_i, ..., a_{12}, a_{13}$
O erro não será identificado pelo dígito verificador se:

$$a_1 + 3.a_2 + a_3 + ... + 3.a_{i+1} + a_i + ... + 3.a_{12} + a_{13} \equiv 0 \ mod\ 10 \tag{2.12}$$

Subtraindo as equações 2.11 e 2.12, temos:

$$2.a_i - 2a_{i+1} \equiv 0 \ mod\ 10$$
$$2.(a_i - a_{i+1}) \equiv 0 \ mod 10$$

Como $0 \leq a_i \leq 9$, que permite concluir que $\mid a_i - a_{i+1} \mid = 5$. □

Um erro de transposição alternada no código de barras EAN 13 não será identificado pelo dígito verificador.

Observação 2.5. *Uma transposição alternada do tipo:*

$$..., a_i, a_{i+1}, a_{i+2}, ... \mapsto ...a_{i+2}, a_{i+1}, a_i, ...$$

Não é detectada pelo dígito verificador do código de barras EAN 13.

De fato, seja $a_1, a_2, ..., a_i, a_{i+1}, a_{i+2}..., a_{12}, a_{13}$ uma sequência de algarismos de um código de barras EAN 13. Temos:

$$a_1 + 3.a_2 + a_3 + ... + 3.a_i + a_{i+1} + 3.a_{i+2} + ... + 3.a_{12} + a_{13} \equiv 0 \ mod\ 10 \tag{2.13}$$

Suponha um erro de transposição alternada e que tenha sido digitado a seguinte sequência: $a_1, a_2, ..., a_{i+2}, a_{i+1}, a_i..., a_{12}, a_{13}$. Temos:

$$a_1 + 3.a_2 + a_3 + ... + 3.a_{i+2} + a_{i+1} + 3.a_i + ... + 3.a_{12} + a_{13} \equiv 0 \ mod\ 10 \tag{2.14}$$

Logo, temos que as equações 2.13 e 2.14 são iguais, ou seja, o dígito verificador não indicará nenhum erro ao operador.

2.2.3 *Análise de dígitos de verificação módulo 11*

O sistema de identificação módulo 11 consegue satisfazer simultaneamente os dois Teoremas 2.2 e 2.3, ou seja, o dígito consegue identificar todos os erros únicos e de transposição que representam 90% dos erros mais comuns. Porém apresentam uma pequena desvantagem para representar o número 10, ele pode ser representado em algarismo romano X e nesse caso recebe o nome de sistema completo módulo 11 ou ser representado por 0 e recebe o nome de sistema restrito módulo 11.

Um sistema completo módulo 11 por introduzir uma letra, apresenta uma pequena inconveniência, como por exemplo quando é necessário informar uma sequencia numérica em um auto atendimento telefônico. Já um sistema restrito módulo 11 não é possível identificar erros que alteram o dígito de 0 para 10.

Podemos concluir que o dígito verificador, ou identificador de erros, não é eficaz em todos os casos, porém, consegue identificar a presença de erros em sequências numéricas nos casos de equívocos mais comuns.

3

CÓDIGOS CORRETORES DE ERROS

Neste capítulo, iremos mostrar o funcionamento de códigos que fazem a detecção e correção de erros. Foram usadas as seguintes referências [4], [3], [7], [8], [9] e [10].

Os códigos corretores de erros estão presentes em nosso cotidiano toda vez que assistimos a um programa de TV ou a um filme em um aparelho de DVD, gravamos dados em um Pendrive, enviamos um e-mail, trocamos mensagens pelo telefone, e em várias outras situações em que é necessário transmitir uma informação ou fazer um arquivo de dados.

Um código corretor de erros tem como princípio identificar e corrigir erros que ocorrem em dados que, ao serem enviados ou armazenados pela fonte, chegam incorretos no receptor por alguma interferência. Para que seja possível fazer a identificação e correção de um eventual erro no comando, são adicionados termos redundantes junto com a mensagem original para que seja possível fazer uma comparação entre as possibilidades de envio e o comando recebido.

Para entender melhor o funcionamento desta teoria dos códigos que fazem a correção de erros, considere a seguinte situação:

Exemplo 3.1. *Um carrinho de brinquedo que funciona com um controle remoto. Neste controle, as opções de movimento são: frente, ré, direita e esquerda.*

Estes comandos do controle remoto podem ser codificados em um produto cartesiano $\{0,1\} \times \{0,1\}$ da seguinte forma:

$$\begin{aligned} \textit{Frente} &\longmapsto 00 \\ \textit{Ré} &\longmapsto 01 \\ \textit{Esquerda} &\longmapsto 10 \\ \textit{Direita} &\longmapsto 11 \end{aligned}$$

Considere que, ao serem transmitidos esses sinais no controle remoto, o sinal enviado tenha sido 00 e, por alguma interferência externa, o carrinho tenha recebido 01, logo, o movimento será contrário ao enviado. Neste caso, não é possível fazer a identificação e, consequentemente, a correção do erro. Para que seja possível fazer essa identificação, são adicionadas, ao código original, redundâncias.

Usando esse Exemplo 3.1, suponha que os comandos de movimento fossem o seguinte:

$$\begin{aligned} \textit{Frente} &\longmapsto 0000 \\ \textit{Ré} &\longmapsto 0101 \\ \textit{Esquerda} &\longmapsto 1010 \\ \textit{Direita} &\longmapsto 1111 \end{aligned}$$

Considere que tenha sido enviado do controle remoto o sinal 0100, como o código recebido pelo carrinho não é reconhecido, é identificado que o sinal sofreu algum erro ao ser transmitido. Comparando o código recebido com os que podem ser enviados, o carrinho não conseguirá fazer a correção, pois, mesmo supondo que houve um único erro, o código correto pode ser 0101 ou 0000.

Admita agora que o controle remoto desse mesmo exemplo use os seguintes comandos de movimentos:

$$\begin{aligned} \textit{Frente} &\longmapsto 000000 \\ \textit{Ré} &\longmapsto 010001 \\ \textit{Esquerda} &\longmapsto 100011 \\ \textit{Direita} &\longmapsto 110011 \end{aligned}$$

Se o sinal para *frente*, 000000, for emitido pelo controle, e o carrinho receber 000010, é possível identificar e corrigir o erro, pois o código mais próximo[1] do que pode ser enviado pela fonte é 000000.

Como ilustrado neste exemplo, um código corretor de erros tem como princípio adicionar termos redundantes ao comando inicial para que seja permitido identificar e corrigir possíveis erros. Abaixo segue uma ilustração das etapas do funcionamento de um código corretor de erros.

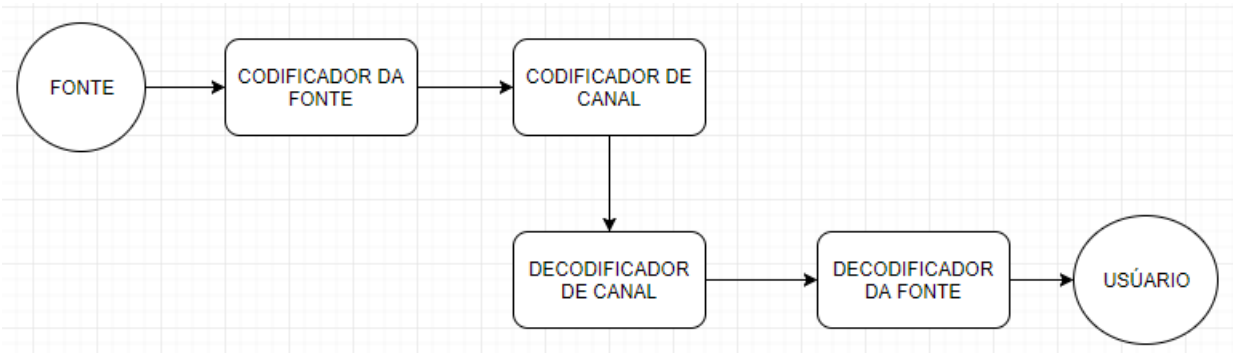

Figura 3.1: Esquema de um código corretor de erros

As informações da Figura 3.1 são:

FONTE : Informação original que vai ser transmitida ou armazenada.

CODIFICADOR FONTE : Compressão de dados, sua finalidade é usar a menor quantidade de símbolos possível para transmitir ou armazenar a informação da fonte.

CODIFICADOR DE CANAL : Adiciona informações redundantes para que o receptor use estas informações para detecção e correção de erros ocorridos por ruídos ou interferências que afetam a informação original quando ela é transmitida ou armazenada.

DECODIFICADOR DE CANAL : Tenta reconstruir a informação do código da fonte com base no codificador da fonte e na redundância adicionada.

DECODIFICADOR DA FONTE : Recupera a informação da fonte.

1 A expressão *mais próximo*, intuitivamente clara, será melhor explicada ao tratarmos de Distância de Hamming.

USUÁRIO : Recebe a informação da fonte com os possíveis erros já corrigidos pelo decodificador de canal.

O comando frente do carrinho de controle remoto do Exemplo 3.1, pode ser representado no seguinte diagrama:

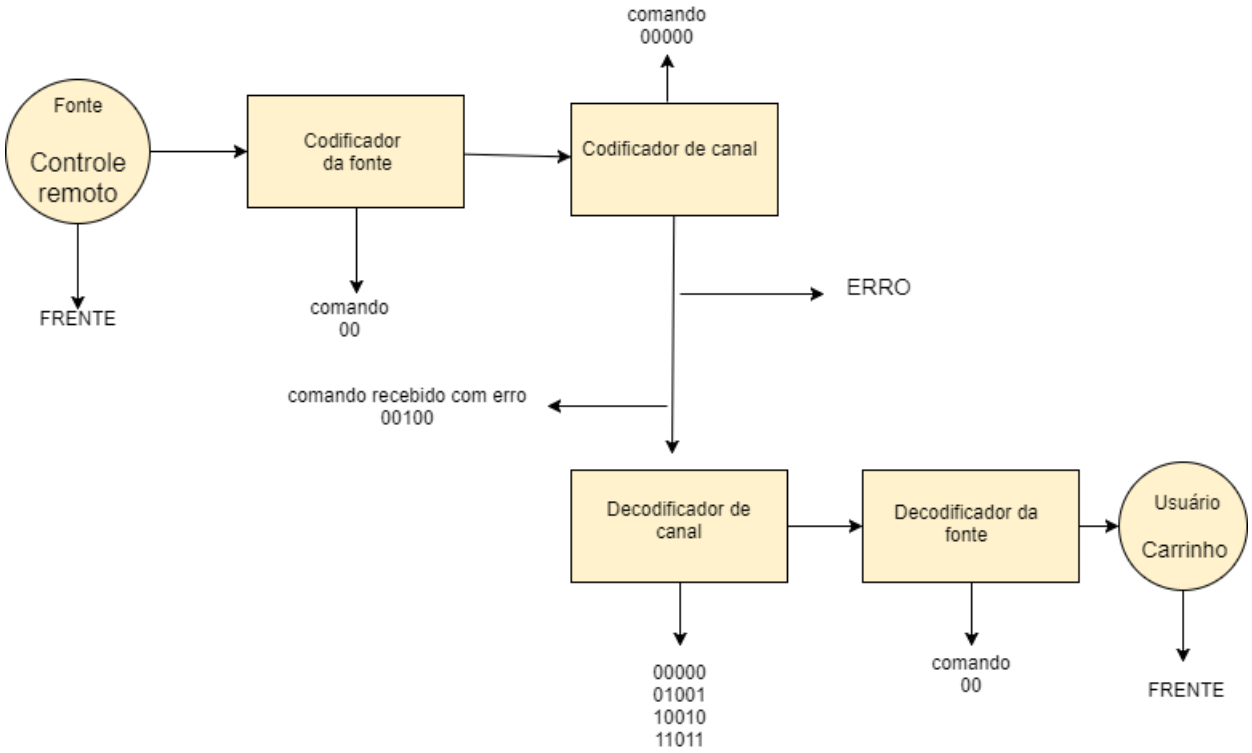

Figura 3.2: Exemplo: carrinho de controle remoto

Neste capítulo, será feito um estudo de como se transforma o código da fonte em código do canal, sendo possível fazer a identificação e correção de erros ao decodificar o comando ou mensagem da fonte.

Será feito um estudo somente de **canais simétricos**. Segundo Abramo Hefez e Maria Lúcia T. Villela [3], canais simétricos são aqueles que apresentam as seguintes propriedades:

- Todos os símbolos transmitidos têm a mesma probabilidade (pequena) de serem recebidos errados.
- Se um símbolo é recebido errado, a probabilidade de ser qualquer um dos outros elementos do alfabeto é a mesma.

3.1 MÉTRICA DE HAMMING

Mas afinal, o que é um código? Formalmente, para se definir um código, precisamos inicialmente de um conjunto finito A de símbolos denominado **alfabeto** (em geral, trabalharemos com alfabetos numéricos, de modo que podemos nos referir a seus símbolos como *dígitos*). O número de elementos de A, denotado por $|A|$, será simbolizado neste trabalho pela letra q. Uma **palavra** desse alfabeto é um elemento de A^n, onde n é o **comprimento** da palavra. Um **código** construído sobre o alfabeto A é um subconjunto C de A^n (podem-se considerar códigos em que as palavras tenham comprimentos diferentes, mas esse caso não nos interessará neste trabalho).

Além disso, para que seja possível construir um código *corretor de erros*, será necessário também termos um conceito de *distância* entre palavras de um código, de modo a medir a proximidade entre elas.

Definição 3.1. *A* **distância de Hamming** *entre dois pares de palavras* $u = (u_1, ..., u_n)$ *e* $v = (v_1, ..., v_n)$*, denotado por* $d_H(u,\ v)$*, é o número de posições onde os dígitos correspondentes são diferentes.*

$$d_H(u, v) = |\{i;\ u_i \neq v_i,\ 1 \leq i \leq n\}|$$

Considere o seguinte exemplo:

Exemplo 3.2. *Considere um alfabeto* $A = \{0, 1\}$*, com as palavras* $u = (1,\ 0,\ 0,\ 0,\ 1)$ *e* $v = (0,\ 1,\ 0,\ 0,\ 0)$ *tomadas em* A^5*. A distância Hamming entre u e v é* $d_H(u\ ,\ v) = 3$.

Proposição 3.2. *A distância de Hamming é uma métrica em* A^n*, ou seja:*

i - *Positividade:* $d_H(u\ ,\ v) \geq 0, \forall u, v \in A$*, sendo igual se, e somente se,* $u = v$*.*

ii - *Simetria:* $d_H(u\ ,\ v) = d_H(v\ ,\ u), \forall u, v \in A$*.*

iii - *Desigualdade Triangular:* $d_H(u\ ,\ v) \leq d_H(u\ ,\ w) + d_H(w\ ,\ v), \forall u, v, w \in A$*.*

Demonstração. Os itens i e ii são triviais.

iii - Para demonstrar esta propriedade, será analisado a contribuição das i-ésimas coordenadas de u, v para $d_H(u, v)$. Temos duas situações para ser verificada, se a contribuição das i-ésimas coordenadas é igual a zero se $u_i = v_i$ e igual a um se $u_i \neq v_i$.

- Para $u_i = v_i$, temos que a contribuição das i-ésimas coordenadas a $d_H(u, v)$ é menor ou igual a das i-ésimas coordenadas de $d_H(u, w) + d_H(w, v)$ que pode ser 0, 1 ou 2.
- Agora, se $u_i \neq v_i$, não podemos ter $v_i = w_i$ e $u_i = w_i$. Portanto a contribuição das i-ésimas coordenadas a $d_H(v, w) + d_H(u, w)$ é maior ou igual a 1, que é a contribuição das i-ésimas coordenadas de $d_H(u, v)$.

□

As propriedades da Proposição 3.2 caracterizam uma métrica, por isso, a distância de Hamming entre os elementos de A^n é chamado de métrica de Hamming.

Definição 3.3. *Considere u uma palavra de A^n e um número natural $r > 0$. Será definido como* ***disco*** *e* ***bola*** *de raio r e centro u como sendo, respectivamente, os seguintes conjuntos:*

$$D(u\ , r\ ) = \{\ v\ \in\ A^n;\ d_H(v\ , u\ )\ \leqslant\ r\}$$
$$S(u\ , r\ ) = \{\ v\ \in\ A^n;\ d_H(v\ , u\ )\ =\ r\}$$

Definição 3.4. *A* ***distância mínima*** *de um código C é dado por:*

$$d = \ min\ \{d_H(u\ ,\ v);\ u,\ v \in\ C\ e\ u\ \neq\ v\}.$$

Exemplo 3.3. *Dado um alfabeto $A = \{0, 1\}$ e um código $C \subset A^4$,*

$$C = \{(0,\ 0,\ 0,\ 0),\ (1,\ 1,\ 1,\ 1),\ (1,\ 1,\ 0,\ 1),\ (0,\ 0,\ 1,\ 1)\},$$

temos que as distâncias entre as palavras são:

$$d_H(0000, 1111) = 4$$
$$d_H(0000, 1101) = 3$$
$$d_H(0000, 0011) = 2$$
$$d_H(1111, 1101) = 1$$
$$d_H(1111, 0011) = 2$$
$$d_H(1101, 0011) = 3$$

Logo a distância mínima do código C é $d = 1$.

Sendo M a quantidade de elementos de um código C, para determinar a distância mínima devemos calcular $\begin{pmatrix} M \\ 2 \end{pmatrix}$ distâncias, ou seja, calcular a distância dos elementos

de dois em dois, para então determinar a mínima. Esta operação se torna inviável para códigos grandes (muitas palavras-código), pois apresenta um custo computacional elevado. Será apresentado até o final deste capítulo, uma maneira econômica para calcular a distância mínima d.

Seja C um código com distância mínima $d \geq 1$, ponha-se:

$$k = \left[\!\left[\frac{d-1}{2} \right]\!\right] \tag{3.1}$$

onde $[\![x]\!]$ representa a parte inteira de um número real x.

Observação 3.5. *Note que* $0 \leq 2k < d$ *e que* $k = 0 \Leftrightarrow d = 1$.

Lema 3.6. *Dado um código* C *com distância mínima* $d > 1$. *Se* c *e* c' *são palavras distintas de* C, *então:*

$$\big(D(c\,,k\,) \cap D(c'\,,k\,)\big) \cap C = \emptyset$$

Em outras palavras, discos de raio k centrados em uma palavra do código não contêm outras palavras do código.

Demonstração. Suponha por absurdo que exista uma palavra-código $x \in D(c\,,k\,) \cap D(c'\,,k\,)$, logo:

De acordo com a Definição 3.3, teríamos:

$$d_H(x\,,c\,) \leqslant k$$
$$d_H(x\,,c'\,) \leqslant k$$

logo pela desigualdade triangular iii da proposição 3.2, temos:

$$d_H(c\,,c'\,) \leqslant d_H(x\,,c\,) + d_H(x\,,c'\,) \leqslant 2k < d$$

Isso é um absurdo, pois $d_H(c\,,c'\,) \geqslant d$.

□

O resultado acima sugere a seguinte ideia para a correção de erros na transmissão (veja figura 3.3): se uma palavra v recebida estiver numa bola de raio k centrada numa palavra u do código ($v \neq u$), assume-se que a palavra transmitida possui $d_H(u, v)$ erros (identificação do erro) e que a correta (a palavra transmitida) é a palavra u (correção do erro).

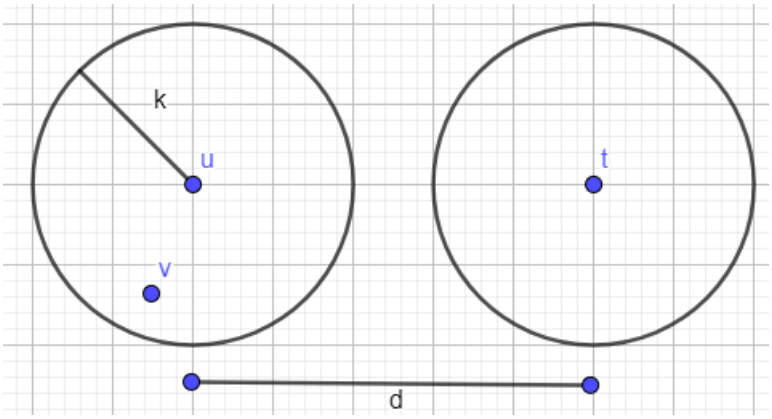

Figura 3.3: Representação geométrica dos códigos corretores de erros

Teorema 3.7. *Um código C com distância mínima $d > 1$ pode detectar até $d-1$ erros em uma palavra transmitida e corrigir até $k = \left[\!\left[\frac{d-1}{2}\right]\!\right]$ erros.*

Demonstração. Suponha que, ao transmitir uma palavra a do código C, sejam cometidos t erros na transmissão, com $t \leqslant k$, recebendo a palavra b com $d(a\,,b\,) = t \leqslant k$. Neste caso, temos que $b \in D(a\,,k\,)$, pelo lema 3.6, temos que $b \notin D(a'\,,k\,)$ para $a \neq a'$, e, assim, podemos concluir que $d_H(a\,,b\,)$ é menor que a distância de b a qualquer outra palavra do código C e, então, teremos a a partir b.

Por outro lado, se na transmissão de uma palavra-código podemos introduzir nela até $d-1$ erros sem encontrar outra palavra código, e assim , a detecção do erro será possível.

□

Com base no Teorema 3.7, podemos concluir que quanto maior a distância mínima entre as palavras do código, melhor será a sua capacidade de corrigir erros. Na Teoria dos Códigos, é fundamental poder calcular d ou, pelo menos, determinar uma cota inferior para ele.

Definição 3.8. *Seja um código $C \subset A^n$ com distância mínima d e seja $k = \left[\!\left[\frac{d-1}{2}\right]\!\right]$. Dizemos que este código é* **perfeito** *se:*

$$\bigcup_{c\,\in\,C} D(c\,,k\,) = A^n$$

Um código é perfeito quando todos os elementos de A^n pertence a alguma bola de raio k centradas nas palavras-código de C. Um código perfeito consegue corrigir de

maneira única, uma palavra recebida, ou seja, uma palavra recebida sempre pertence a alguma bola.

De acordo com o Teorema 3.7, podemos obter uma estratégia para detecção e correção de erros de uma palavra recebida. Considere C um código com distância mínima d. Temos que a quantidade de erros que este código corrige é $k = \left[\!\left[\frac{d-1}{2}\right]\!\right]$. Se uma palavra a é recebida temos duas situações para serem verificadas segundo Hefez e Villea [3]:

i - A palavra a encontra-se em um disco de raio k em torno de uma palavra c do código. (essa palavra é única, pela prova do Teorema (3.7)). Nesse caso, substitui-se a por c.

ii - A palavra a não se encontra em nenhum disco de raio k em torno de uma palavra c do código. Nesse caso, não é possível decodificar a com boa margem de segurança.

Na primeira situação (i), não se sabe a quantidade de erros que foram cometidos na transmissão, pois poderíamos ter cometido mais do que k erros, afastando, assim, a da palavra original, e a aproximando de uma outra palavra do código, e, o item (ii), não ocorre em códigos perfeitos. Existe uma maneira de otimizar a correção, pois quanto maior for d, maior será a chance de que seja um caso do item (i).

A Teoria dos Códigos Corretores de Erros apresenta três parâmetros fundamentais:

- M: Quantidade de palavras (número de elementos) do código.
- d: Distância miníma entre duas palavras quaisquer do código.
- n: Comprimento de uma palavra-código (o número n, corresponde ao espaço de A^n de um código).

O objetivo da teoria é estabelecer uma relação entre estes parâmetros produzindo códigos mais eficientes e eficazes. Os códigos que interessam são aqueles em que M e d sejam grandes relativamente a n, ou seja, um código capaz de transmitir ou armazenar bastante informação M com uma boa capacidade de correção com d relativamente grande. Isso não é uma tarefa fácil, pois quando se aumenta o número de palavras M, diminui-se a distância mínima d entre elas.

3.2 EQUIVALÊNCIA DE CÓDIGOS

Um conceito importante nos códigos é a noção de equivalência, para que seja possível definir equivalência entre códigos, é necessário definir o conceito de isometria.

Definição 3.9. *Dado um alfabeto A e n um número natural, dizemos que uma função $F : A^n \to A^n$ é uma **isometria** de A^n se ela preserva distância de Hamming. Ou seja:*

$$d_H(F(x), F(y)) = d_H(x, y); \quad \forall x, y \in A^n$$

As isometrias para a métrica de Hamming apresentam as seguintes propriedades:

Proposição 3.10. *Toda isometria de A^n é uma bijeção de A^n.*

Demonstração. Suponha que $F : A^n \to A^n$ seja uma isometria. Seja $x, y \in A^n$, tal que $F(x) = F(y)$, logo, temos que $d_H(x, y) = d_H(F(x), F(y)) = 0$, implicando em $x = y$. Podemos concluir que F é injetora e, como toda aplicação injetora de um conjunto finito nele próprio é sobrejetora, temos que a função F é uma bijeção. □

Definição 3.11. *Sejam dois códigos C e C' em A^n, dizemos que C' é equivalente a C se existir uma isometria F de A^n tal que $F(C) = C'$.*

A equivalência de códigos é uma relação de equivalência, ou seja, apresenta as seguintes propriedades:

i Reflexiva: Todo código é equivalente a si próprio.

ii Simétrica: Se C' é equivalente a C, então C é equivalente a C'.

iii Transitiva: Se C'' é equivalente a C' e C' é equivalente a C, então C'' é equivalente a C.

Proposição 3.12. *Dois códigos equivalentes possuem os mesmos parâmetros.*

Demonstração. Como os dois códigos estão contido em A^n eles apresentam o mesmo comprimento n. Considerando que são códigos equivalentes, existe uma bijeção de A^n que leva um ao outro, ou seja, eles apresentam a mesma cardinalidade M. E como existe uma isometria por hipótese, os códigos C e C' apresentam a mesma distância miníma. □

3.3 CÓDIGOS LINEARES

A classe de códigos mais utilizada é a linear, caracterizada por utilizar-se de uma estrutura vetorial. Para que seja possível construir um **código linear**, serão tomados:

- $\mathbb{F}$ um corpo finito com q elementos tomado como alfabeto.
- Um espaço vetorial $\mathbb{F}^n$ sobre $\mathbb{F}$.
- C um subconjunto de $\mathbb{F}^n$.

Definição 3.13. *Um código $C \subset \mathbb{F}^n$ será chamado de código linear se for um subespaço vetorial de $\mathbb{F}^n$.*

Exemplo 3.4. *Considere o carrinho de controle remoto usado no começo do capítulo com os seguintes comandos de movimento:*

$$\begin{aligned} \textit{Frente} &\longmapsto 00 \\ \textit{Re} &\longmapsto 01 \\ \textit{Esquerda} &\longmapsto 10 \\ \textit{Direita} &\longmapsto 11 \end{aligned}$$

Para que seja possível fazer a identificação e correção de possíveis erros quando o sinal é transmitido, são adicionadas redundâncias ao código inicial, como se segue:

$$\begin{aligned} \textit{Frente} &\longmapsto 00000 \\ \textit{Re} &\longmapsto 01101 \\ \textit{Esquerda} &\longmapsto 10110 \\ \textit{Direita} &\longmapsto 11011 \end{aligned}$$

Neste exemplo o alfabeto é $A = \mathbb{F} = \{0, 1\}$ e o código é imagem de $\mathbb{F}^2$ através de uma transformação linear, sendo subespaço vetorial de $\mathbb{F}^5$. De fato:

$$\begin{array}{rccc} \mathbb{T}: & \mathbb{F}^2 & \to & \mathbb{F}^5 \\ & (x_1, x_2) & \mapsto & (x_1, x_2, x_1 + x_2, x_1, x_2) \end{array}$$

Por definição, todo código linear é um espaço vetorial de dimensão finita. Seja b a dimensão de um código C e $v_1, v_2, ..., v_b$ uma de suas bases. Todo elemento do código pode ser escrito de forma única:

$$c = u_1 v_1 + u_2 v_2 + ... + u_b v_b$$

sendo $u_i, i = 1, ..., b$ elementos de $\mathbb{F}$. Temos então que número de palavras de um código é:

$$M = q^b$$

Como os códigos lineares são subespaços vetoriais, obtemos uma vantagem para calcular a distância mínima d, pois o vetor 0, elemento neutro da soma, sempre estará em um código linear e nos permite introduzir a seguinte:

Definição 3.14. *Dado $x \in \mathbb{F}^n$, define-se **peso** de x como sendo o número inteiro:*

$$w(x) = |\{i / x_i \neq 0\}| = d_H(x, 0)$$

*e **peso mínimo** de um código linear:*

$$w(C) = min\{w(x); x \in C, x \neq 0\}.$$

Proposição 3.15. *Seja $C \subset \mathbb{F}^n$ um código linear com uma distância mínima d. Então:*

i) $\forall x, y \in \mathbb{F}^n$, *temos que* $d_H(x, y) = w(x - y)$.

ii) $d = w(C)$.

Demonstração. Para demonstrar item i, suponha $x = (a_1, a_2, ..., a_n)$ e $y = (b_1, b_2, ..., b_n)$ palavras de um código C. Pela definição da métrica de Hamming e da definição do peso , temos:

$$d_H(x, y) = |\{i, a_i \neq b_i, 1 \leq i \leq n\}| = |\{i, a_i - b_i \neq 0, 1 \leq i \leq n\}| = d_H(x - y, 0) = w(x - y)$$

O item ii decorre do fato que, para todo par de elementos $x, y \in C$ com $x \neq y$, tem-se um elemento $z = x - y \in C$ e portanto $d(x, y) = w(z)$. □

Exemplo 3.5. *Dado o corpo* $\mathbb{F} = \{0, 1\}$ *e o código* $C = \{00000, 01011, 10100, 11111\}$ *subespaço vetorial de* $\mathbb{F}^5$ *e uma imagem de* $\mathbb{F}^2$ *através de uma transformação linear. Para obter a distância mínima desse código, calculamos* $M - 1 = 4 - 1 = 3$ *pesos:*

$$w(01011) = 3$$
$$w(10100) = 2$$
$$w(11111) = 5$$

Portanto a distância mínima do código é 2.

Na seção 3.1, vimos que, para conhecer a distância mínima de um código, era necessário calcular $\begin{pmatrix} M \\ 2 \end{pmatrix}$ distâncias, para então determinar a mínima, isso não será mais necessário, pois a distância mínima é igual ao peso do código linear, ou seja, será necessário calcular a distância entre cada um dos $M - 1$ elementos não nulos e o elemento neutro.

Ainda assim, não se resolve o problema de códigos grandes, mesmo calculando-se $M - 1$ distâncias, apresenta-se um custo computacional alto, portanto, será necessário desenvolver outro método para calcular a distância mínima.

3.3.1 *Código Linear obtido como imagem ou núcleo de uma transformação linear*

Podemos descrever códigos lineares como subespaços vetoriais de C de um espaço vetorial $\mathbb{F}^n$ de duas maneiras: uma como **imagem**, e outra como **núcleo** de uma transformação linear. Nesta subseção e na próxima, será estabelecido uma estrutura matemática no processo de codificação e decodificação. Uma palavra-código será construída a partir de um vetor informação e será acrescentada redundância através de uma transformação linear.

Será obtida uma representação do código C através de uma transformação linear. Para isso escolha uma base $v_1, v_2, ..., v_b$ de C e considere a seguinte aplicação:

$$\begin{array}{rccc} \mathbb{T}: & \mathbb{F}^b & \to & \mathbb{F}^n \\ & (u_1, ..., u_b) & \mapsto & u_1.v_1 + u_2.v_2 + ... + u_b.v_b \end{array}$$

Temos que $\mathbb{T}$ é uma transformação linear injetora e $Im(\mathbb{T}) = C$.

Exemplo 3.6. *Suponha um código* $C \subset \mathbb{F}^6$ *sobre um corpo* $\mathbb{F} = \{0, 1\}$ *com a seguinte base* $(001000, 100100, 010010, 000001)$. *Seja* $x = (1001)$ *a palavra a ser enviada de espaço vetorial* $\mathbb{F}^4$. *Para codificar a palavra* x, *fazemos:*

$$\mathbb{T}(x) = 1.(001000) + 0.(100100) + 0.(010010) + 1.(000001)$$
$$\mathbb{T}(x) = 001001$$

Assim, o vetor informação $x = (1001)$ *é codificada e será enviada como vetor código* 001001.

Seja C um código linear representado por uma transformação linear $\mathbb{T}$. Para determinar se $z \in \mathbb{F}^n$ é ou não um vetor código é necessário obter uma base $v_1, v_2, ..., v_b$ de $Im(\mathbb{T})$ e resolver o sistema de equações:

$$u_1.v_1 + u_2.v_2 + ... + u_b.v_b = z$$

A resolução desse sistema de equações, em geral, pode apresentar um custo computacional elevado. Para reduzir este custo, podemos obter um código linear C de uma outra maneira, usando o núcleo de uma transformação linear. Para isso, seja C' um subespaço de $\mathbb{F}^n$ complementar de C, isto é:

$$C \oplus C' = \mathbb{F}^n$$

e considere a aplicação linear:

$$\begin{array}{rccc} \mathbb{H}: & C \oplus C' & \to & \mathbb{F}^{n-b} \\ & u \oplus z & \mapsto & z \end{array}$$

Sendo o núcleo o código C. Para saber se uma palavra-código pertence a C, basta verificar se $H(z) = 0$, o que apresenta um baixo custo computacional.

3.3.2 *Matriz Geradora de um Código*

Mostraremos nesta subseção, um processo para codificar o vetor informação. Para isso, seja $\mathbb{F}$ um corpo finito e $C \subset \mathbb{F}^n$ um código linear. Iremos chamar a terna de inteiros (n, b, d) de *parâmetros do código linear* sendo:

- n - comprimento do vetor código;

- b - dimensão do código C;
- d - distância mínima.

Para determinar a **matriz geradora** de um código linear, iremos considerar $\{j_1, ..., j_b\}$ como base canônica de $\mathbb{F}^b$, a base canônica de $\mathbb{F}^n$ representada por $\{f_1, ..., f_n\}$ e $\{v_1, ..., v_b\}$ sendo uma base do código C. A matriz geradora transmite elementos de um espaço $\mathbb{F}^b$, ou seja, uma transformação linear $\mathbb{T}$:

$$\mathbb{T}: \quad \mathbb{F}^b \quad \rightarrow \quad \mathbb{F}^n$$

É possível escrever todos os elementos da base do código linear C na base canônica de $\mathbb{F}^n$:

$$\begin{cases} v_1 &= u_{11}f_1 &+ \cdots &+ u_{1n}f_n \\ \vdots &= \vdots &+ \ddots &+ \vdots \\ v_b &= u_{b1}f_1 &+ \cdots &+ u_{bn}f_n \end{cases}$$

sendo os coeficientes u_{ij} elementos de $\mathbb{F}$. Então, a matriz geradora de um código é:

$$G = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{b1} & u_{b2} & \cdots & u_{bn} \end{pmatrix}$$

Cada linha da matriz geradora G corresponde a um vetor que pertence ao código C, isto é, C é um subespaço de $\mathbb{F}^n$ gerado pelas linhas de G, que é uma base do código. Todos os elementos do código C são os vetores $z \in \mathbb{F}^n$ da forma $x.G = z$ para todo $x \in \mathbb{F}^b$.

Exemplo 3.7. *Consideremos $\mathbb{F} = \{0, 1\}$ e $C \subset \mathbb{F}^6$ um código. Se $\beta = \{(0, 1, 0, 1, 0, 0), (1, 0, 1, 1, 0, 1)\}$ é uma base de C, temos a seguinte matriz geradora:*

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

Neste caso, a palavra código da fonte $x = 10$ é codificada como:

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} = 010100$$

Exemplo 3.8. *Usando os mesmos dados do exemplo anterior, suponhamos, agora, que seja dada a palavra código* 101101 *de um código e que queremos descobrir a palavra do código da fonte, ou seja,* $x \in \mathbb{F}^2$*. Temos:*

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Resolvendo o produto matricial, encontramos:

$$\begin{pmatrix} x_2 & x_1 & x_2 & x_1 + x_2 & 0 & x_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Assim, conseguimos determinar a palavra do código da fonte, que é $x = 01$*.*

O sistema de equações usado no exemplo anterior apresentou uma resolução simples, pois a matriz geradora não era complexa. Esse fato não acontece em todos os casos, em alguns, a resolução pode ser muito trabalhosa. Para facilitar a resolução, na matriz geradora, podemos efetuar operações algébricas que correspondem a mudança de bases, tais como:

L1. Permutar duas linhas.

L2. Multiplicar uma linha por um escalar não nulo.

L3. Adicionar um múltiplo escalar de uma linha a outra.

Definição 3.16. *Uma* ***matriz geradora*** *G de um código C está na forma padrão se tivermos:*

$$G = (Id_b | A)$$

onde Id_b *é a matriz identidade de ordem b, e A, uma matriz* $b \times (n - b)$*.*

Exemplo 3.9. *Considere o código linear* $C \subset \mathbb{F}^5$*, sendo* $\mathbb{F} = \{0, 1\}$ *com a seguinte matriz geradora:*

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

Efetuando permutações das linhas, podemos obter:

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

que é uma matriz geradora na forma padrão de um código C.

Não são todas as matrizes geradoras que são possíveis de serem transformadas na forma padrão usando somente as operações com as linhas, como por exemplo, se a matriz geradora do exemplo anterior fosse:

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

Se permutarmos as colunas, será possível obter a matriz geradora na forma padrão. De uma forma geral, podemos efetuar as seguintes operações:

C1. Permutação de duas colunas.

C2. Multiplicação de uma coluna por um escalar não nulo.

Dessa forma, podemos encontrar a matriz geradora na forma padrão de um código C' equivalente a C.

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

Observação 3.17. *Ao efetuar as operações com as colunas em uma base de C, implica efetuá-las em todas as palavras-código de C, o que caracteriza uma isometria.*

Usando essas operações com as colunas, chegamos ao seguinte resultado:

Teorema 3.18. *Dado um código C, existe um código equivalente C' com matriz geradora na forma padrão.*

Demonstração. Suponhamos G uma matriz geradora de código C. Usando as operações $L1$, $L2$, $L3$ e $C1$, podemos determinar G na forma padrão:

Seja:

$$\begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{b1} & g_{b2} & \cdots & g_{bn} \end{pmatrix}$$

Como a primeira linha não é nula, pois os vetores são linearmente independentes, existe um elemento não nulo e por meio de $C1$, podemos assumir $g_{11} \neq 0$ e, usando $L2$, podemos multiplicar os elementos dessa linha por g_{11}^{-1}, encontrando $g_{11} = 1$. Usando $L3$, podemos somar as linhas seguintes de G com a primeira linha multiplicada respectivamente por $(-)g_{21}$, $(-1)g_{31}$, etc. Depois dessas operações, encontramos a seguinte matriz:

$$\begin{pmatrix} 1 & b_{12} & \cdots & b_{1n} \\ 0 & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & b_{b2} & \cdots & b_{bn} \end{pmatrix}$$

Na segunda linha, temos, certamente, um elemento não nulo que, quando multiplicado pelo seu inverso, $L2$, e ao usar $C1$, podemos encontrar a seguinte matriz:

$$\begin{pmatrix} 1 & c_{12} & c_{13} & \cdots & c_{1n} \\ 0 & 1 & c_{23} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & c_{b2} & c_{b3} & \cdots & c_{bn} \end{pmatrix}$$

Usando $L3$, obtemos:

$$\begin{pmatrix} 1 & 0 & d_{13} & \cdots & d_{1n} \\ 0 & 1 & d_{23} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & d_{b3} & \cdots & d_{bn} \end{pmatrix}$$

e assim, sucessivamente, podemos encontrar a matriz geradora na forma padrão.

$$G' = (Id_b | A)$$

□

3.3.3 *Códigos Duais*

Definição 3.19. *Seja $C \subset \mathbb{F}^n$ um código linear com uma matriz geradora G. Definimos como código dual $C^\perp$:*

$$C^\perp = \{v \in \mathbb{F}^n; \langle v, u \rangle = 0, \forall u \in C\}$$

Um código $C^\perp$ apresenta as seguintes propriedades:

i. $C^\perp$ é um subespaço vetorial de $\mathbb{F}^n$;

ii. $x \in C^\perp \iff Gx^t = 0$.

Demonstração. No item i, consideremos $u, v \in C^\perp$ e $\alpha \in \mathbb{F}^n$, e, usando a definição 3.19, temos que para todo $x \in C$:

$$\langle u + \alpha v, x \rangle = \langle u, x \rangle + \alpha . \langle v, x \rangle = 0$$

portanto, podemos concluir que $u + \alpha v \in C^\perp$, logo, $C^\perp$ é um subespaço vetorial de $\mathbb{F}^n$.

Temos, no item ii, que $x \in C^\perp$ se, somente se, x é ortogonal a todos os elementos de uma base C, o que é equivalente a dizer $G.x^t = 0$, pois as linhas de G são uma base de C. □

Como $C^\perp$ é um subespaço vetorial de $\mathbb{F}^n$, ortogonal a C, é, portanto, um código linear que será chamado de **código dual**.

Proposição 3.20. *Seja $C \subset \mathbb{F}^n$ um código linear de dimensão b com matriz geradora, na forma padrão, $G = (Id_b|A)$. Então:*

i. $dim C^\perp = n - b$

ii. $H = (-A^t|Id_{n-b})$ *é uma matriz geradora de* $C^\perp$.

Demonstração. i - Sendo $x = (x_1, ..., x_n) \in C^\perp$ se, e somente se, $G.x^t = 0$, como G está na forma padrão, podemos escrever a seguinte relação:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & a_{1(b+1)} & a_{1(b+2)} & \cdots & a_{1n} \\ 0 & 1 & 0 & \cdots & 0 & a_{2(b+1)} & a_{2(b+2)} & \cdots & a_{2n} \\ 0 & 0 & 1 & \cdots & 0 & a_{3(b+1)} & a_{3(b+2)} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & a_{b(b+1)} & a_{b(b+2)} & \cdots & a_{bn} \end{pmatrix} . \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Efetuando o produto das matrizes, encontramos:

$$\begin{pmatrix} x_1 + a_{1(b+1)}.x_{b+1} + a_{1(b+2)}.x_{b+2} + \cdots + a_{1n}.x_n \\ x_2 + a_{2(b+1)}.x_{b+1} + a_{2(b+2)}.x_{b+2} + \cdots + a_{2n}.x_n \\ x_3 + a_{3(b+1)}.x_{b+1} + a_{3(b+2)}.x_{b+2} + \cdots + a_{3n}.x_n \\ \vdots \\ x_b + a_{b(b+1)}.x_{b+1} + a_{b(k+2)}.x_{b+2} + \cdots + a_{bn}.x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Podemos escrever a matriz produto da seguinte maneira:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_b \end{pmatrix} = - \begin{pmatrix} a_{1(b+1)}.x_{b+1} + a_{1(b+2)}.x_{b+2} + \cdots + a_{1n}.x_n \\ a_{2(b+1)}.x_{b+1} + a_{2(b+2)}.x_{b+2} + \cdots + a_{2n}.x_n \\ a_3(b+1).x_{b+1} + a_{3(b+2)}.x_{b+2} + \cdots + a_{3n}.x_n \\ \vdots \\ a_{b(b+1)}.x_{b+1} + a_{b(b+2)}.x_{b+2} + \cdots + a_{bn}.x_n \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_b \end{pmatrix} = - \begin{pmatrix} a_{1(b+1)} & a_{1(b+2)} & \cdots & a_{1n} \\ a_{2(b+1)} & a_{2(b+2)} & \cdots & a_{2n} \\ a_{3(b+1)} & a_{3(b+2)} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{b(b+1)} & a_{b(b+2)} & \cdots & a_{bn} \end{pmatrix} . \begin{pmatrix} x_{b+1} \\ x_{b+2} \\ x_{b+3} \\ \vdots \\ x_n \end{pmatrix}$$

Logo, $C^{\perp}$ possui q^{n-b} elementos onde $x_{b+1}, ..., xn$ podem ser escolhidos aleatoriamente, portanto, podemos concluir que $dim C^{\perp} = n - b$

ii - As linhas de H são linearmente independentes, portanto, geram um subespaço vetorial de dimensão $n - b$. Como as linhas de H são ortogonais às linhas de G, temos que o espaço gerado pelas linhas de H está contido em $C^{\perp}$. E, como estes subespaços apresentam a mesma dimensão, eles coincidem. Isso prova que $H = (-A^t | Id_{n-b})$ é a matriz geradora de $C^{\perp}$. □

Proposição 3.21. *Seja C um código de dimensão b em $\mathbb{F}^n$, e uma matriz geradora G. Uma matriz H de ordem $(n - b) \times n$, com coeficientes em $\mathbb{F}$ e com linhas linearmente independentes, é uma matriz geradora de $C^{\perp}$ se, e somente se, $G.H^t = 0$.*

Demonstração. As linhas de H geram um subespaço vetorial de $\mathbb{F}^n$ com dimensão $n - b$, logo, a mesma dimensão de $C^\perp$. Se $h_1, h_2, ..., h_{n-b}$ e $g_1, g_2, ..., g_b$ são as linhas respectivamente de H e G, temos:

$$(G.H^t)_{ij} = \langle g_i, h_j \rangle$$

Desse modo, $G.H^t = 0$ é equivalente dizer que todos os vetores do subespaço gerado pelas linhas de H estão em $C^\perp$. E, como esse subespaço tem a mesma dimensão de $C^\perp$, então:

$G.H^t = 0 \iff C^\perp$ é gerado pelas linhas de H.

□

Corolário 3.22. $(C^\perp)^\perp = C$

Demonstração. Sejam G e H matrizes geradoras, respectivamente, de C e $C^\perp$. Como $G.H^t = 0$, temos:

$$(G.H^t)^t = 0^t$$

$$H.G^t = 0$$

logo, G é a matriz geradora de $(C^\perp)^\perp$. □

A seguir, será apresentado um resultado que nos permitirá definir com mais facilidade se uma palavra pertence ou não ao código C. Esta proposição apresenta um custo computacional mais barato.

Proposição 3.23. *Seja C um código linear, e suponhamos que H seja uma matriz geradora de $C^\perp$. Então, temos que $v \in C$ se, somente se, $H.v^t = 0$.*

Demonstração. Pelo Corolário 3.22 e Definição 3.19 ii, temos que $v \in C$ se, somente se, $v \in (C^\perp)^\perp$ se, e somente se, $H.v^t = 0$. □

Podemos concluir que é possível descobrir se uma palavra-código pertence a um código apenas com o produto da matriz geradora de $C^\perp$ por um vetor. A matriz H é chamada de **matriz teste de paridade** de C.

Dado um código C com teste de paridade H e um vetor $v \in \mathbb{F}^n$, será chamado o vetor $H.v^t$ de *síndrome* de v.

Exemplo 3.10. *Seja C um código sobre o corpo $\mathbb{F} = \{0, 1\}$ com a seguinte matriz geradora na forma padrão, isto é $G = (Id_b|A)$:*

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

Podemos determinar a matriz teste de paridade $H = (-A^t|Id_{n-b})$:

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

Assim, dados dois vetores $v_1 = (100111)$ e $v_2 = (010101)$, podemos verificar se as palavras códigos v_1 e v_2 pertencem ao código C.

$$H.v_1^t = 0$$

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

e

$$H.v_2^t = 0$$

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

Portanto, podemos concluir, pela proposição 3.23, que $v_1 \in C$ e $v_2 \notin C$.

A matriz H teste de paridade apresenta informações sobre o valor do peso d do código.

Lema 3.24. *Dada uma matriz teste de paridade H de um código C, temos que o peso de C é maior ou igual a* **s** *se, e somente se, quaisquer* **s-1** *colunas de H são linearmente independentes.*

Demonstração. Consideremos uma matriz de paridade H e que cada conjunto de $s-1$ colunas é linearmente independente. Sejam $c = (c_1, ..., c_n)$ uma palavra não nula de um código C e $h_1, ..., h_n$ as colunas de H, como $H.c^t = 0$, temos que:

$$0 = H.c^t = \sum c_i h_i$$

Se $w(c) \leq s-1$ é o número de componentes não nulas da palavra c e a equação $H.c^t = 0 = \sum c_i h_i$, há uma combinação nula de um número t de colunas, isto é, $1 \leq t \leq s-1$, o que é uma contradição. Portanto, podemos concluir que $w(c) \geq s$, então, $w(C) \geq s$.

Reciprocamente, consideremos $W(C) \geq s$, e suponhamos, por absurdo, que H tenha $s-1$ colunas $h_1, ..., h_{s-1}$ linearmente dependentes. Dessa forma:

$$c_i.h_i + ... + c_{s-i}.h_{s-i} = 0$$

Logo, $c = (0, ..., c_i, 0, ..., 0, c_{s-i}, 0, ..., 0) \in C$, e isso implicaria em $w(c) \leq s-1 < s$, o que seria um absurdo. □

Teorema 3.25. *Dada uma matriz teste de paridade H de um código C, temos que o peso C é igual a s se, e somente se, quaisquer $s-1$ colunas de H forem linearmente independentes e existirem s colunas dependentes.*

Demonstração. Suponhamos que $w(C) = s$. Temos que todo o conjunto de $s-1$ colunas de H é linearmente independente. Logo, existem s colunas de H linearmente dependentes, pois, caso contrário, $w(C) \geq s+1$.

Reciprocamente, vamos supor agora que todo conjunto de colunas de H é linearmente independente e existem s colunas linearmente dependentes. Logo, temos $w(C) \geq s$, porém $w(C)$ não pode ser maior que s, pois, novamente, todo conjunto com s colunas é linearmente independente, o que é uma contradição. □

3.4 DECODIFICAÇÃO DE CANAL

A decodificação de canal é o procedimento que permite detectar e corrigir erros de um código. O método original foi apresentado na década de 90 no laboratório

Bell por D. Slepian, e apresentava um custo computacional elevado. Esse método foi aperfeiçoado e será apresentado nesta seção.

Para isso, vamos definir o **vetor erro** e como a diferença entre o vetor recebido r e o enviado c,ou seja:

$$e = r - c$$

Exemplo 3.11. *Consideremos um código* $C \in \mathbb{F} = \{0, 1\}$ *e que tenha sido transmitido a palavra-código* 101101*, e recebido a palavra-código* 010101*, então:*

$$e = 010101 - 101101 = 111000$$

O vetor erro serve para identificar, através de seu peso, a quantidade de erros cometidos na transmissão. No exemplo anterior, $e = 111000$, apresenta peso igual a 3.

Sabemos que H a matriz teste de paridade de um código C, e para $c \in C$, temos que $H.c^t = 0$. Portanto podemos escrever a seguinte relação:

$$H.e^t = H.(r^t - c^t) = H.r^t - H.c^t = H.r^t$$

Lembre que chamamos de **síndrome** de v o vetor $H.v^t$. Como $H.e^t = H.r^t$, temos por consequência que a palavra recebida r e o vetor erro e tem a mesma síndrome. Considere h_i a i-ésima coluna da matriz teste de paridade H e $e = (\alpha_1 ... \alpha_n)$, temos que:

$$\sum_{i=1}^{n} \alpha_i h_i = H.e^t = H.r^t$$

A seguir, apresentaremos um importante resultado que permite efetuar a correção de uma palavra transmitida com erro.

Lema 3.26. *Dado um código linear C sobre um corpo* $\mathbb{F}^n$ *com capacidade k de correção de erros. Se* $r \in \mathbb{F}^n$ *e* $c \in C$ *são tais que* $d(c, r) \leq k$*, existe um único vetor* $\mathbf{e}$ *com* $w(e) \leq k$*, cuja síndrome é igual à síndrome de r e tal que* $c = r - e$*.*

Demonstração. Seja $e = r - c$, temos que $H.e^t = H.r^t$ e $w(e) = d(c, r) \leq k$, logo, e satisfaz a condição desejada. Para provar a unicidade, suponhamos que $e = (\alpha_1 ... \alpha_n)$ e $e' = (\alpha'_1 ... \alpha'_n)$ sejam dois vetores em que $w(e) \leq k$ e $w(e') \leq k$ apresentam a mesma síndrome que r. Se H é a matriz teste de paridade de C, temos que:

$$H.e^t = H.e'^t$$

$$\sum_{i=2}^{n} \alpha_i.h_i = \sum_{i=1}^{n} \alpha'_i.h_i$$

$$\alpha_1 h_1 + ... + \alpha_n h_n - \alpha'_1.h_1 - ... - \alpha'_n.h_n = 0$$

$$(\alpha_1 - \alpha'_1)h_1 + ... + (\alpha_n - \alpha'_n)h_n = 0$$

o que nos fornece uma relação de dependência linear entre $2k$ colunas de H, pois sabemos que $k = \left[\!\left[\frac{d-1}{2}\right]\!\right]$, ou seja, $2k \leq d-1$, e como quaisquer $d-1$ colunas de H são linearmente independentes, pelo Teorema 3.25, temos que $\alpha_i = \alpha'_i$ para todo i, logo, temos que $e = e'$. □

Agora, vamos determinar e quando $w(e) \leq 1$, ou seja, o vetor erro tem todas as suas entradas nulas ou existe apenas uma de suas entradas não nula. Para que isso seja possível, suponhamos um código C que tenha distância mínima $d \geq 3$ e que o vetor erro e, introduzido entre a palavra transmitida c e a recebida r, seja tal que $w(e) \leq 1$.

Se $H.e^t = 0$, a palavra recebida não apresenta erro, logo $c = r$.

Agora se $H.e^t \neq 0$, então $w(e) = 1$ e $e = (0, ..., 0, \alpha, 0, ..., 0)$, com $\alpha \neq 0$, apenas a i-ésima entrada não é nula. Como:

$$H.r^t = H.e^t = \alpha h_i$$

Para que seja possível saber em qual entrada do vetor erro e está o elemento não nulo, basta analisar as colunas de H e localizar a coluna h_i, sendo, assim, possível identificar o erro. A matriz teste de paridade H deve apresentar as seguintes propriedades:

i) Nenhuma coluna de H deve ser nula, caso contrário, um erro na posição correspondente à linha nula não seria detectado.

ii) Todas as colunas de H devem ser únicas. Caso duas colunas sejam iguais, erros nas posições correspondentes a essas linhas podem ser inidentificáveis.

Usando todas essas informações, podemos descrever um algoritmo que permite a decodificação em um código corretor de um erro.

Algoritmo 1. *Seja H uma matriz teste de paridade de um código C e seja r um vetor recebido. Considere $d \geq 3$.*

(i) Determine $H.r^t$;

(ii) *Se $H.r^t = 0$, aceite a palavra recebida r;*

(iii) *Se $H.r^t = s^t \neq 0$, compare s^t com as colunas de H;*

(iv) *Se existirem i e α, tais que $s^t = \alpha.h_i$, para $\alpha \in \mathbb{F}$, então* **e** *é a n-upla com α na posição i e zeros nas outras posições. Corrija r pondo $c = r - e$;*

(v) *Se não ocorrer o item iv, foi cometido mais de um erro.*

Exemplo 3.12. *Consideremos C um código com a seguinte matriz geradora:*

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

Sendo $G = (Id_k|A)$, ou seja, uma matriz geradora na forma padrão, e sendo $Id_{2\times 2}$ a matriz identidade e $A_{2\times 3}$ a matriz A.

Podemos determinar a matriz de teste de paridade $H_{(n-k)\times n}$, pois $H = (-A^t|Id_{n-k})$:

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

Vamos supor que seja recebida uma palavra r = (10100), temos:

$$H.e^t = H.r^t$$

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

Como o produto matricial não é o vetor nulo, temos que a palavra foi transmitida com erro. Analisando as colunas de H encontramos que o erro aconteceu na quarta coluna, portanto, podemos escrever da seguinte maneira o vetor erro e = 00010, e o vetor enviado como:

$$c = r - e$$

$$c = 10100 - 00010 = 10110$$

ou seja, a palavra código transmitida foi c = 10110.

A seguir mostraremos um segundo algoritmo para a decodificação de mensagens. Para isso, seja $C \subset \mathbb{F}^n$ um código corretor de erros, sendo H a matriz de teste de paridade, d a distância mínima, $M = q^b$ o número de palavras do código C e que o código tenha a capacidade de corrigir $k = \left[\!\left[\frac{d-1}{2}\right]\!\right]$ erros. Quando $H.e^t = H.r^t$, dizemos que r e e apresentam a mesma síndrome, e se $w(e) = d(r,c) < k$, então e é determinado de maneira única por r.

Definição 3.27. *Seja $v \in \mathbb{F}^n$, chamamos cada conjunto da forma $v + C$ de* **classe lateral** *de v segundo C:*

$$v + C = \{v + c; c \in C\}$$

Lema 3.28. *Sejam $u, v \in \mathbb{F}^n$, u e v têm a mesma síndrome se, e somente se, $u \in v + C$*

Demonstração.

$$H.u^t = H.v^t \Leftrightarrow H.(u-v)^t = 0 \Leftrightarrow u - v \in C \Leftrightarrow u \in v + C$$

□

A classe lateral de v segundo C apresenta as seguintes propriedades:

i) $v + C = v' + C \Leftrightarrow v - v' \in C$;

ii) $(v + C) \cap (v' + C) \neq 0 \Longrightarrow v + C = v' + C$;

iii) $\bigcup_{v \in F^n} (v + C) = \mathbb{F}^n$;

iv) $|(v + C)| = |C| = q^b$

As classes laterais são classes de equivalência da relação $v - v' \in C$. O número de classes laterias de v segundo C é q^{n-b}, sendo n o comprimento do código e b a sua dimensão.

Exemplo 3.13. *Seja $\mathbb{F} = \{0, 1\}$ um corpo e $C \subset \mathbb{F}^n$ seja um código gerado pela seguinte matriz:*

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

Temos que:

- $q = 2$, *pois o alfabeto $\mathbb{F}$ tem dois elementos;*

- $n = 4$, *pois a matriz geradora G tem quatro colunas;*
- $b = 2$, *pois a matriz geradora, que é uma base de C, tem duas linhas;*
- $|(v + C)| = 2^2 = 4$, *ou seja, o número de elementos da classe lateral é 4;*
- *O número de classes laterais é* $2^{4-2} = 4$.

Usando a matriz geradora, temos que o código $C = \{0000, 0101, 1011, 1110\}$ *e as classes laterais segundo C são:*

$$0000 + C = \{0000, 0101, 1011, 1110\}$$

$$1000 + C = \{1000, 1101, 0011, 0110\}$$

$$0100 + C = \{0100, 0001, 1111, 1010\}$$

$$0010 + C = \{0010, 0111, 1001, 1100\}$$

Podemos encontrar os elementos da classe lateral construindo uma tabela da seguinte maneira: na primeira linha e coluna, colocamos, a palavra-código zero e, nos demais elementos da primeira linha, as outras palavras-código de C. Para encontrar o vetor v da segunda linha e primeira coluna, devemos encontrar o vetor que apresenta a menor distância Hamming do vetor que está na primeira linha e primeira coluna, os demais itens da segunda linha são determinados através da soma do vetor v com as demais colunas da primeira linha, sendo a soma em $\mathbb{Z}_2$, no caso do exemplo anterior, e assim, sucessivamente, até a linha q^{n-b}. Abaixo, segue a tabela do exemplo anterior:

0000	1011	0101	1110
1000	0011	1101	0110
0100	1111	0001	1010
0010	1001	0111	1100

Chamamos de **líder de classe** o elemento que apresenta o menor peso entre os elementos da classe lateral. No exemplo anterior, temos:

- 0000 é líder de $0000 + C$;
- 1000 é líder de $1000 + C$;
- 0100 e 0001 são líderes de $0100 + C$;

- 0010 é líder de $0010 + C$.

Podemos observar que, em alguns casos, temos mais de um líder de classe lateral, e nos interessam as que apresentam somente um líder. A seguir, apresentaremos uma importante proposição:

Proposição 3.29. *Seja C um código linear em $\mathbb{F}^n$ com distância mínima d. Se $u \in \mathbb{F}^n$ é tal que:*

$$w(u) \leq \left[\!\left[\frac{d-1}{2}\right]\!\right] = k$$

então u é o único elemento líder de sua classe.

Demonstração. Vamos supor $u, v \in \mathbb{F}^n$, com $w(u) \leq \left[\!\left[\frac{d-1}{2}\right]\!\right]$ e $w(v) \leq \left[\!\left[\frac{d-1}{2}\right]\!\right]$. Como já sabemos que $u - v \in C$, temos:

$$w(u-v) \leq w(u) + w(v) \leq \left[\!\left[\frac{d-1}{2}\right]\!\right] + \left[\!\left[\frac{d-1}{2}\right]\!\right] \leq d-1$$

Portanto, $u - v$ tem peso menor que d, ou seja, $u = v$. □

Novamente usando essas informações, podemos descrever um algoritmo que permite a decodificação em código corretor de um erro pela síndrome que apresenta um número de erros menor ou igual a k. Iremos usar a última proposição para encontrar o elemento líder de classe, ou seja, vamos determinar $u \in \mathbb{F}^n$, tal que $w(u) \leq k$, e, em seguida, calcular as síndromes desses elementos e organizá-los em uma tabela.

Algoritmo 2. *i) Determine a síndrome $s^t = H.r^t$;*

ii) Se **s** *estiver na tabela, seja* **l** *o elemento líder da classe determinada por* **s**, *troque* **r** *por $r - l$;*

iii) Se **s** *não estiver na tabela, então, a mensagem recebida contém mais do* **k** *erros.*

Para entender o funcionamento do algoritmo, vamos mostrar dois exemplos. O primeiro exemplo foi construído com distância mínima 3, ou seja, corrige apenas um erro, já o segundo, foi construído com distância mínima 5, então, este código consegue fazer a correção de no máximo dois erros.

Exemplo 3.14. *Considere um código binário criado para transmitir a palavra PROFMAT. Para isso, considere C um código com a seguinte matriz geradora e que na transmissão seja cometido no máximo um erro. Este exemplo é adaptação do exercício* 5.5 *da página* 111 *do livro [3] :*

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

Usando as operações com as linhas e colunas, podemos encontrar G na sua forma padrão:

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

e a matriz teste de paridade H:

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

A distância mínima deste código é 3*, sendo assim, esse código faz a correção de um e detecta dois erros. Tomando como exemplo o alfabeto da nossa língua, podemos definir um código fonte para transmitir a mensagem da seguinte forma:*

$$P = 00011$$
$$R = 10110$$
$$O = 00101$$
$$F = 11000$$
$$M = 01001$$
$$A = 10000$$
$$T = 11010$$

Iremos codificar o código fonte em palavra-código fazendo um produto matricial do código fonte com a matriz geradora na forma padrão G', de acordo com o seguinte diagrama:

Vamos enviar a palavra PROFMAT, usando a matriz geradora, encontraremos as seguintes palavras código:

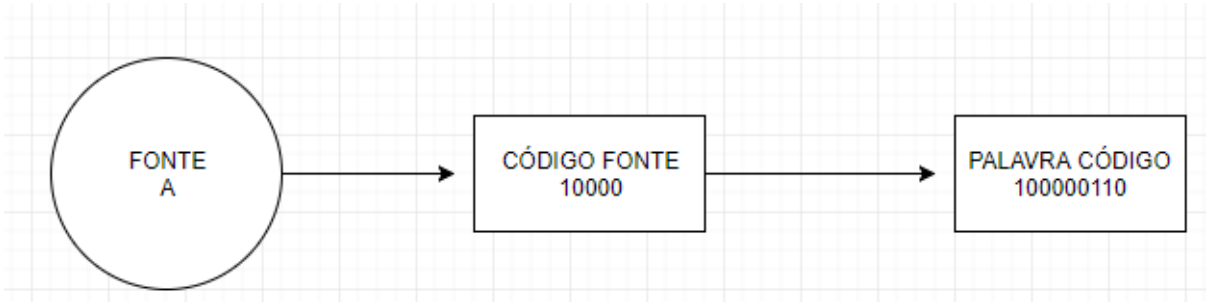

Figura 3.4: Transformar código fonte em código canal

$$P = 000110011$$
$$R = 101101011$$
$$O = 001011110$$
$$F = 110001010$$
$$M = 010010101$$
$$A = 100000110$$
$$T = 110100000$$

Para que seja possível fazer a decodificação de um código recebido, vamos determinar o elemento líder de classe, ou seja, vamos determinar todos os elementos do corpo do código com peso menor ou igual a 1:

Líder	*síndrome*
000000000	0000
100000000	0110
010000000	1100
001000000	0111
000100000	1010
000010000	1001
000001000	1000
000000100	0100
000000010	0010
000000001	0001

Para entender o funcionamento do algoritmo, considere que seja recebido o código 000110011, *como* $H.r^t = (0000)^t$, *temos que não houve erro na transmissão e, fazendo a decodificação, temos que a palavra fonte é* P.

Consideremos, agora, que seja recebido o seguinte código 111001010, *como* $H.r^t = (0111)^t$, *temos que a mensagem foi recebida com erro. Consultando a tabela, temos que o líder de classe é* 001000000 *e, usando o algoritmo, ou seja, trocando o vetor recebido r por* $r - l =$ 111001010 − 001000000 = 110001010, *fazendo a decodificação, encontramos a letra F.*

Exemplo 3.15. *Vamos usar as informações do exemplo 3.1 que foi utilizado no início deste capítulo. Seja C um código com distância mínima 5, logo, este código é eficiente para corrigir até dois erros.*

Será usado a seguinte matriz geradora G:

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

e matriz teste de paridade H

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

Temos as seguintes informações:

Comando	*Código Fonte*	*Código Canal*
Frente	*00*	*000000000*
Ré	*01*	*010111001*
Esquerda	*10*	*101100110*
Direita	*11*	*111011111*

Para fazer a correção de erro é necessário construir a tabela de líder com peso menor ou igual a dois com sua respectiva síndrome:

Líder	*Síndrome*	*Líder*	*Síndrome*
000000000	0000000	010000010	0111011
100000000	1100110	010000001	0111000
010000000	0111001	001100000	1100000
001000000	1000000	001010000	1010000
000100000	0100000	001001000	1001000
000010000	0010000	001000100	1000100
000001000	0001000	001000010	1000010
000000100	0000100	001000001	1000001
000000010	0000010	000110000	0110000
000000001	0000001	000101000	0101000
110000000	1011111	000100100	0100100
101000000	0100110	000100010	0100010
100100000	1000110	000100001	0100001
100010000	1110110	000011000	0011000
100001000	1101110	000010100	0010100
100000100	1100010	000010010	0010010
100000010	1100100	000010001	0010001
100000001	1100111	000001100	0001100
011000000	1111001	000001010	0001010
010100000	0011001	000001001	0001001
010010000	0101001	000000110	0000110
010001000	0110001	000000101	0000101
010000100	0111101	000000011	0000011

Suponha que o carrinho deste exemplo receba os seguintes comandos:

Com um erro:

$r = 001100110$, *calculando* $H.r^t = (1100110)^t$, *a mensagem chegou ao receptor com erro e consultando a tabela, trocamos r por* $r - l = 001100110 - 100000000 = 101100110$ *e o comando é* ESQUERDA.

Com dois erros:

$r = 010100001$, *calculando* $H.r^t = (0011000)^t$, *a mensagem chegou ao receptor com erro e consultando a tabela, trocamos r por* $r - l = 01010001 - 000011000 = 010111001$ *e o comando é* RÉ.

4

APLICAÇÕES

Neste capítulo, apresentamos duas sequências de atividades com o objetivo de mostrar aplicações de alguns conteúdos do Ensino Fundamental e Médio usando os dígitos verificadores e códigos corretores de erros.

Mostrar uma aplicação de um conteúdo de forma prática facilita o processo de ensino-aprendizagem: a Matemática torna-se interessante para o aluno quando ele consegue perceber e aplicar a teoria em situações práticas. Um processo significativo para o educando, deve permitir que ele consiga fazer relações com o seu cotidiano, segundo os Parâmetros Curriculares Nacionais [11]:

> [...] O significado da Matemática para o aluno resulta das conexões que ele estabelece entre ela e as demais disciplinas, entre ela e seu cotidiano e das conexões que ele estabelece entre os diferentes temas matemáticos. (p.19)

O objetivo das sequências de atividades é instigar no aluno uma maneira diferente de entender o conteúdo do currículo, perceber a Matemática aplicada no seu dia a dia, mostrar que a Matemática não é um ciência encerrada em si mesma. Com estas atividades é possível oportunizar uma participação melhor dos alunos no seu processo de aprendizagem.

4.1 SEQUÊNCIA PARA O ENSINO FUNDAMENTAL

Para o Ensino Fundamental, apresentamos uma sequência de atividades que utilizam os dígitos verificadores de erros para aplicar alguns conteúdos deste segmento educacional e desenvolver a curiosidade, investigação, criar estratégias e compreender padrões. Segundo a Base Nacional [12], sobre a Matemática:

> No Ensino Fundamental, essa área, por meio da articulação de seus diversos campos – Aritmética, Álgebra, Geometria, Estatística e Probabilidade, precisa garantir que os alunos relacionem observações empíricas do mundo real a representações (tabelas, figuras e esquemas) e associem essas representações a uma atividade matemática (conceitos e propriedades), fazendo induções e conjecturas. Assim, espera-se que eles desenvolvam a capacidade de identificar oportunidades de utilização da matemática para resolver problemas, aplicando conceitos, procedimentos e resultados para obter soluções e interpretá-las segundo os contextos das situações. (p.265)

Esta atividade é um sugestão para o sexto ano Ensino Fundamental. Com estas atividades, será possível mostrar uma aplicação para as operações dos números naturais, divisibilidade e números primos. Segundo o PCN [11], um aluno do ensino fundamental do ciclo 3 deve:

> Reconhecer os significados dos números naturais em diferentes contextos e estabelecimento de relações entre números naturais, tais como "ser múltiplo", "ser divisor de".

Mostraremos várias etapas as quais foram pensadas para serem desenvolvidas em 10 aulas; esta sugestão pode ser adaptada para as demais séries.

4.1.1 *1ª etapa - o que é um código?*

Começar esta etapa investigando o conceito dos alunos sobre códigos, verificar o sentido dessa palavra para eles. Depois, solicitar que façam uma consulta a um dicionário da Língua Portuguesa – uma sugestão é dicionário on-line Priberam [1]:

> 1. Coleção de leis. 2. Coleção de regras, preceitos, fórmulas, etc. 3. Sistema de símbolos que permite interpretar, transmitir uma mensagem, representar uma informação de dados. 4. Sistema convencional, rigorosamente estruturado, de símbolos ou de sinais e de regras combinatórias integrado no processo da comunicação.

1 "código", in Dicionário Priberam da Língua Portuguesa [em linha], 2008-2013, https://www.priberam.pt/dlpo/c%C3%B3digo [consultado em 17-01-2018].

Em seguida, questionar se eles reconhecem algum código no seu dia a dia. Usando as respostas dos alunos, explicar e explorar o conceito de códigos numéricos, sua utilidade e importância no nosso cotidiano para fazer identificações. Dar exemplo práticos, como códigos de barras, números de CPF e RG, CEP – código de endereçamento postal etc.

O objetivo desta etapa é mostrar que utilizamos esses códigos numéricos em diversas situações corriqueiras nos nossos hábitos cotidianos. Com essas investigações podemos motivar os alunos para as etapas seguintes e ajudá-los a perceber que a Matemática está presente em diversas situações.

4.1.2 *2ª etapa - vamos entender um código de barras EAN 13?*

Nesta etapa vamos mostrar um código de barras, apresentar as partes que identificam o país, produto, empresa e o dígito verificador. Salientar a sua importância e utilidade nos supermercados, por exemplo, e que seus algarismos somente estão corretos se seguirem um conjunto de regras.

Escrever os algarismos de um código de barras na lousa e mostrar tal conjunto de regras. Não será usado um rigor matemático para descrever essas regras; uma sugestão é escrever os números do código de barras multiplicando de forma alternada e respectiva por 1 e 3 e depois somar os produtos. Veja um exemplo:

Vamos usar os seguintes algarismos de um código de barras: 7896051020158

$$7*1+8*3+9*1+6*3+0*1+5*3+1*1+0*3+2*1+0*3+1*1+5*3+8*1$$

$$7+24+9+18+0+15+1+0+2+0+1+15+8=100$$

Para que estejam corretos os algarismos do código de barras, a soma dever ser múltiplo de 10; caso não seja, algum número foi escrito de forma incorreta.

Propor que façam o cálculo de outros números de códigos de barras e em alguns com um erro nos algarismos (um erro que seja detectável e outro não detectável)

Após os alunos calcularem alguns exemplos para sistematizar o processo, fazer alguns questionamentos:

- - Será que todos os códigos numéricos usam o mesmo processo do código de barras?
- - Por que, em alguns casos, mesmo com o erro na escrita dos algarismos do código de barras, o resultado foi um número múltiplo de 10?

- - Qual o motivo de usar um sistema de regras que não identifica todos os erros?

O objetivo desses questionamentos é despertar a curiosidade para outros códigos numéricos e mostrar que, por mais que pareçam simples, para os algarismos do código de barras ou números de um CPF existe uma matemática aplicada.

4.1.3 *3ª etapa - o que é um dígito verificador?*

Nesta etapa, salientar a importância de usar um dígito verificar em um código numérico: em destaque, o do código de barras. Em seguida, questionar os alunos se depois da segunda etapa eles conseguem determinar o valor numérico de um dígito verificador.

Para calcular dígito verificador do código de barras, deve-se repetir o processo da etapa anterior sem escrever o dígito e determinar qual algarismo dever ser adicionado para que seja um número múltiplo de 10.

Segue um exemplo:

$$7*1+8*3+9*1+6*3+0*1+5*3+1*1+0*3+2*1+0*3+1*1+5*3=92$$

Logo, o valor numérico do algarismo do dígito dever ser 8. Propor para calcular o dígito de outros números de código de barras.

4.1.4 *4ª etapa - construindo um código numérico para identificar sólidos geométricos*

Em grupo, utilizar quatro cores de cartolinas e montar 4 tamanhos diferentes de paralelepípedos, cubos e tetraedros. Cada grupo deverá montar um código numérico para identificar seus sólidos geométricos e escolher um método para calcular o dígito que irá identificar a presença de erros. Neste momento, deixar os alunos fazerem escolhas de acordo com os seus critérios e depois fazer uma socialização de cada grupo e fazer os seguintes questionamentos:

- - Se vocês trocarem o código numérico de um sólido geométrico com outros grupos, será possível identificar as características do sólido?
- - Não será necessário um padrão para o código numérico?

- - O que é relevante para identificar os sólidos?
- - O método escolhido para determinar o valor numérico do dígito é eficiente? Como saber se foi feita a melhor escolha para calcular o dígito?

Depois desses questionamentos, montar um código numérico padrão para a sala de aula, como na sugestão a seguir:

Um código numérico para identificar nesta ordem: COR - COMPRIMENTO DA ARESTA DA BASE – NOME DO SÓLIDO GEOMÉTRICO – DÍGITO. Escolher algarismos para representar cada característica do sólido geométrico e calcular o dígito conforme as orientações a seguir:

Para determinar o dígito, pode-se orientar os alunos o seguinte método:

- 1^o - Determinar números para multiplicar o algarismo que representa a cor, comprimento da aresta da base, nome do sólido geométrico e dígito, como por exemplo, 2, 3, 5 e 1 respectivamente.
- 2^o - Escolher um número para dividir a soma, como por exemplo, 6 ou 7. Nesse primeiro momento, seria interessante fazer a escolha do número 6 e mostrar que não é eficiente para identificar erro único e erro de transposição e depois destacar que o algarismo 7 ou qualquer outro número primo é mais eficiente para identificar tais erros.

Para sistematizar o processo, pedir para os grupos trocarem os códigos numéricos para verificar a existência de erro.

4.1.5 *6^a etapa - avaliação*

Para a avaliação, propor algumas sequências numéricas de códigos de barras e pedir aos alunos para determinarem o dígito verificador e outras sequências de códigos de barras para verificarem se existe erro nos números informados.

4.2 SEQUÊNCIA PARA O ENSINO MÉDIO

A sequência de atividade destinada para o Ensino Médio foi escolhida para mostrar aplicações do conteúdo de sistemas lineares, números binários, congruências módulo

M e matrizes e incentivar a investigação, criatividade, criação de estratégias e de algoritmos.

Cada etapa, tem o objetivo de levar o aluno a um aprendizado real e significativo, segundo os Parâmetros Curriculares do Ensino Médio [11] um deve:

> Aplicar seus conhecimentos matemáticos a situações diversas, utilizando-os na interpretação da ciência, na atividade tecnológica e nas atividades cotidianas.(p.42)

Esta sequência tem um tempo estimado de 10 aulas.

4.2.1 *1ª etapa - entendendo um código corretor de erros*

Vamos utilizar um programa editor de texto para escrever a palavra "PARALELEPÍPRDO" e mostrar para os alunos que o programa identifica o erro e sugere como correção a palavra correta "PARALELEPÍPEDO". Em seguida escrever a palavra "HACA" e, neste caso, o corretor identifica o erro e sugere como correção "VACA, FACA, JACA"e com estas opções não é possível fazer a correção, pois não é informado nenhuma informação adicional.

Com estes dois exemplos, é possível explicar a distância de Hamming e mostrar que é possível fazer a correção do erro quando uma única palavra apresenta uma distância mínima de outra. Em seguida, explorar outros exemplos com os alunos e enfatizar que usamos códigos corretores em toda informação transmitida ou armazenada e que para identificar e corrigir erros é necessário adicionar redundâncias na informação original. Citar exemplos para despertar a curiosidade dos alunos.

Aproveitando esta etapa, explicar os dígitos dos códigos numéricos que verificam a existência de erros e não faz a indicação do algarismo errado ou a correção do mesmo. Usamos diariamente os códigos numéricos e os códigos corretores e não percebemos a Matemática envolvida nessas situações.

O objetivo desta etapa é explicar os conceitos básicos de um código corretor de erros, mostrar a sua importância e aplicações no nosso cotidiano.

4.2.2 *2º etapa - congruência Módulo M e Classes Residuais*

Um conteúdo de extrema importância para as próximas etapas é a Matemática dos Restos ou Congruência Módulo M e Classes Residuais. Explorar os conceitos básicos, estes conteúdos serão usados nas operações com matrizes. A matriz informação utiliza somente os algarismos 0 ou 1, e o produto entre a matriz informação e a matriz geradora é em $\mathbb{Z}_2$ pois assim é possível fazer a decodificação. O aluno deve entender que só pode usar como algarismos 1 ou 0 e compreender o motivo de trocar, por exemplo, o número ímpar por um ou par por zero.

4.2.3 *3ª etapa - codificar uma informação*

Nesta etapa e nas demais será usado uma adaptação de um exercício do livro [3] página 111. Este mesmo exercício foi usado nesta dissertação como exemplo na seção 3.4. Será usado um código que é possível fazer a detecção e correção de um erro.

Vamos usar um código binário para mandar e receber mensagens, para isso cada letra do nosso alfabeto será relacionado com uma matriz binária e será chamada de matriz informação. Para codificar, usaremos uma matriz geradora de código e produto matricial entre a matriz informação e a matriz geradora será chamado de matriz código.

Com a multiplicação da matriz informação e a geradora de código é possível codificar uma mensagem e adicionar nela redundância, facilitando assim detectar e corrigir erros.

Usaremos a notação G para indicar a matriz geradora de códigos:

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

Matriz informação das letras do nosso alfabeto:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$
$$D = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad E = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad F = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$
$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad I = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$
$$J = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad K = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$
$$M = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad N = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \end{pmatrix} \quad O = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$
$$P = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$
$$S = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$V = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad W = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix} \quad X = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$
$$Y = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Para usar espaço entre as letras: ESPAÇO = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
Segue um exemplo para codificar a letra A:

$$A : \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Observação 4.1. *As operações com as matrizes são em $\mathbb{Z}_2$, ou seja, quando o resultado da soma for um número par, substituir o resultado por zero e quando o resultado da soma for ímpar, substituir por 1. Ver seção 1.5.*

Usando estas instruções, dividir os alunos em grupos e cada grupo codificar uma mensagem para passá-la para outro grupo fazer a decodificação. Para fazer a decodificação usaremos sistemas lineares, e como a matriz codificadora está na forma padrão a solução não será complicada. Veja um exemplo, para decodificar a matriz código $\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$.

Como a informação é uma matriz linha com cinco colunas, e neste caso como ela é desconhecida usaremos uma matriz genérica: $\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix}$. Sabemos que o produto desta matriz genérica pela matriz geradora deve resultar na matriz código:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Resolvendo o sistema encontramos a seguinte matriz informação: $\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}$. Logo, foi transmitido a letra *C*.

4.2.4 *4ª etapa - método para corrigir um erro*

Vamos mostrar um método para identificar e corrigir um erro em um código numérico. Será usado uma matriz para verificar se existe erro e, caso exista, será possível fazer a sua identificação e como estamos usando matrizes binárias é possível fazer a correção pois cada termo a_{ij} é 0 ou 1.

A matriz teste está relacionada com a matriz geradora do código e suas linhas são ortogonais, logo, é fácil descobrir se uma matriz código apresenta erro, pois o produto entre a matriz teste e transposta da matriz código deve ser a matriz nula e caso não seja existe um erro em algum termo a_{ij}.

Usaremos a seguinte matriz teste:

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

Para exemplificar a correção do erro, suponhamos que o receptor tenha recebido a matriz código

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

Fazendo o teste para verificar erro:

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

Como o produto não é a matriz nula, a informação foi recebida com erro. Para identificar a posição do erro basta identificar na matriz de teste a coluna que corresponde ao resultado. Fazendo está análise identificamos a 6ª coluna com erro,logo para corrigir basta trocar o termo a_{16} da matriz código: $\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & \underline{0} & 1 & 0 & 1 \end{pmatrix}$.

Fazendo um novo teste, encontramos:

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

Após os alunos entenderem o processo para verificar a existência de erro, agora em grupo novamente, solicitar que codifiquem uma nova mensagem e em algumas letras acrescentar um erro e depois trocar entre os grupos para fazer a decodificação, identificando e corrigindo erros, caso exista.

Caso seja cometido mais do que um erro, não será possível fazer a identificação e correção. Explicar para os alunos que existem códigos que fazem correção de mais de um erro e usam um método apropriado.

4.2.5 *5^a etapa - avaliação*

Uma sugestão de avaliação é usar a quarta etapa para verificar se os alunos compreenderam o processo de codificação e decodificação de informações corrigindo erros.

BIBLIOGRAFIA

[1] HEFEZ, Abramo; Curso de Álgebra, Rio de Janeiro, Instituto de Matemática Pura e Aplicada, 2002.

[2] NETO, Antonio Caminha Muniz; Tópicos de Matemática Elementar, Teoria dos Números. SBM, Coleção do Professor de Matemática, Vol.5, Rio de Janeiro, 2012.

[3] HEFEZ, Abramo; VILLELA, Maria Lúcia T.; Códigos Corretores de Erros, Rio de Janeiro, Instituto de Matemática Pura e Aplicada, 2008.

[4] ALENCAR, Marcelo; Informação, codificação e segurança de redes, Rio de Janeiro, Editora Elsevier LTDA, 2015.

[5] MILIES, César Polcino, COELHO, Sônia Pitta; Números - Uma introdução Matemática, São Paulo, Editora Universidade de São Paulo,3ª edição, 2001.

[6] PINZ, Carla Rejane Fick. Dígitos verificadores e detecção de erros. Dissertação de Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, Universidade Federal do Rio Grande do Norte, 2013.

[7] SILVEIRA, Raphael Bruno Rodrigues da; Códigos Corretores de Erros: Exemplos da Matemática aplicada em situações Cotidiana; Dissertação de Mestrado em Rede Nacional - PROFMAT, Universidade Federal Rural do Rio de Janeiro, 2015.

[8] NICOLETTI, Everton Rodrigo, Aplicações de Álgebra Linear aos Códigos Corretores de Erros e ao Ensino Médio; Dissertação de Mestrado em rede Nacional - PROFMAT, Universidade Estadual Paulista - Campus de Rio Claro, 2015.

[9] MILIES, César Polcino; Breve introdução á Teoria dos Códigos Corretores de Erros; In: Colóquio de Matemática da Região Centro-Oeste, 2009, Universidade Federal de Mato Grosso do Sul, Campo Grande, 2009.

[10] BAHIA, Flaviano; Um primeiro curso sobre códigos corretores de erros, In: I Encontro Regional de Matemática Aplicada e Computacional, Universidade São João del-Rei, 2010.

[11] BRASIL, Ministério da Educação(MEC);Secretária da Educação Média e Tecnológica(Semtec); Parâmetros Curriculares Nacionais para o Ensino Médio, Brasília, 2000.

[12] BRASIL, Ministério da Educação (MEC); Base Nacional Comum Curricular, Brasília, 2017.