Capítulo 4

Códigos corretores de erros no ensino médio: um estudo sobre o código de Hamming

Me. Everton Henrique Cardoso de Lira[1]
Dra. Márcia Pragana Dantas[2]

Resumo: Os Códigos Corretores de Erros são um tema de bastante utilidade em diversas aplicações tecnológicas nas engenharias, por exemplo, e em pesquisas na área da matemática aplicada. Embora o tema seja amplamente abordado em pesquisas matemáticas acadêmicas, o mesmo ainda é pouco explorado em nível da Educação Básica, mais precisamente, no Ensino Médio. Por esse motivo, no presente trabalho, nos propomos a contribuir com a inserção desse importante assunto na esfera do Ensino Básico. Dessa forma, realizamos uma apresentação do Código de Hamming elementar e acessível para professores do Ensino Médio, partindo de sua formulação original, proposta pelo autor em 1950, e, em seguida, apresentando esse código do ponto de vista matricial, tendo em vista o seu ensino

[1]Universidade Federal de Pernambuco/Secretaria de Educação de Pernambuco, everton.ufpe@hotmail.com
[2]Universidade Federal Rural de Pernambuco, marcia.dantas@ufrpe.br

em nível básico. Por fim, indicamos outra leitura sobre o tema, na qual apresentamos uma sequência didática para o ensino dos Códigos Corretores de Erros no Ensino Médio.

Palavras-chave: Códigos Corretores de Erros; Código de Hamming; Ensino Médio.

4.1 Introdução

A Teoria dos Códigos Corretores de Erros é, grosso modo, o ramo da matemática que estuda os problemas relacionados com o processo de transmissão e recepção de informações digitais, bem como o papel do erro nesse processo. De acordo com Hefez (2018), um Código Corretor de Erros consiste em um procedimento para a transmissão de informações, no qual a introdução sistemática de informação redundante a uma informação prévia que se deseja transmitir é realizada, de forma que a informação redundante seja utilizada posteriormente na detecção e correção dos possíveis erros ocorridos durante a transmissão.

Tais códigos são um dos grandes responsáveis pelo bom funcionamento de tecnologias que fazem parte do nosso cotidiano como, por exemplo, televisores, smartphones, computadores, música digital, internet, dentre outros. Mais ainda, suas aplicações podem ser vistas nas mais diversas áreas da ciência, tais como Engenharia Elétrica (GUIMARÃES, 2003); Biologia (ROCHA, 2010; FARIA, 2011); Computação Quântica (AGUIAR, 2010) e Criptografia (BOLLAUF, 2015). A importância dos Códigos Corretores de Erros pode ser vista também através da atenção que tem sido dada a sua divulgação para um público mais amplo do que a comunidade acadêmica. Consideremos, por exemplo, as obras de divulgação matemática de Milies (2008), Stewart (2013) e Ellenberg (2015, p. 301-325), que trazem o tema de forma mais intuitiva e informal, ou Shine (2009), Sá e Rocha (2012) e Rousseau e Aubin (2015), que apresentam abordagens um pouco mais técnicas, porém elementares.

No que diz respeito às pesquisas acadêmicas advindas do PROFMAT,

o assunto já foi abordado por: Miranda (2013), no qual se destaca o foco dado ao problema do "empacotamento de esferas"; Carvalho (2014), como campo da matemática no qual os conceitos de Matrizes, Determinantes e Polinômios são aplicados; Alves (2015), como contexto para o estudo de Aritmética e Matrizes no Ensino Médio; Nicoletti (2015), em que o autor destaca a relação entre o assunto e a Álgebra Linear, assim como a importância de se introduzir ideias básicas do mesmo no Ensino Médio; Pinz (2013) e Machado (2016), nos quais o tema é abordado através do conceito de dígitos verificadores, utilizados em códigos de barra, no CPF, em cartões de crédito, dentre outros; Dias (2017), no qual o autor apresenta uma aplicação dos Códigos Corretores de Erros realizada pela NASA em 1971 na Missão Mariner; Rodrigues (2017), em que os diagramas de Venn são inteligentemente utilizados para introduzir o assunto no Ensino Básico; e, finalmente, Schroeder (2017), no qual truques de mágica são realizados através da utilização de alguns Códigos Corretores de Erros.

Isso nos sugere o surgimento de uma tendência de introdução e adaptação deste assunto para a sua futura abordagem no Ensino Médio, uma vez que, o mesmo consiste em um rico tema para a contextualização de assuntos como Matrizes, Determinantes, Polinômios, Aritmética Binária, dentre outros assuntos relevantes para este nível de ensino. Decorre daí que, entendemos ser relevante para os professores de matemática do Ensino Básico, a devida compreensão do que são estes códigos, para que em sua atuação nas salas de aula, os mesmos sejam capazes de abordá-los de forma clara e adequada para este nível de ensino. Além disso, entendemos que propostas como esta possuem potencial para serem geradoras de outras propostas na mesma direção, o que no futuro, pode consolidar os Códigos Corretores de Erros como um tema de ensino e estudo na Educação Básica, o que dentre outras coisas, representaria uma renovação e modernização nos conteúdos abordados nos currículos de matemática da Educação Básica.

Tendo em vista que os estudos sobre os Códigos Corretores de Erros têm abordado o tema através de variados enfoques e de perspectivas diversas, neste trabalho optamos por dar um enfoque ao assunto de forma que dele possam advir contribuições relevantes para o ensino da matemática a nível

básico. Assim, nossa escolha recaiu sobre o *Código de Hamming*, desenvolvido em 1950 pelo matemático e engenheiro americano Richard Wesley Hamming (1915 - 1998). Um dos motivos para tal escolha deveu-se ao fato desse código apresentar papel de destaque no desenvolvimento inicial dos primeiros Códigos Corretores de Erros, conforme aponta Abrantes (2003), bem como pelas possibilidades de abordagem do assunto, que acreditamos possíveis de serem realizadas no Ensino Médio.

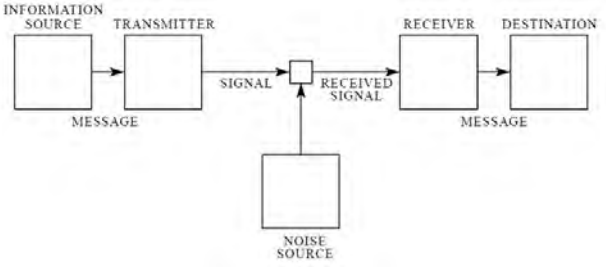
4.2 Teoria da informação e códigos corretores de erros

Os Códigos Corretores de Erros estão intimamente relacionados com a chamada Teoria da Informação, a qual foi inicialmente desenvolvida pelo matemático americano Claude Elwood Shannon (1916 - 2001), em seu clássico trabalho *A Mathematical Theory of Communication* (SHANNON, 1948). Nesse trabalho, Shannon estudou o problema fundamental da comunicação, que segundo ele "é o de reproduzir em um ponto exatamente ou aproximadamente uma mensagem selecionada em outro ponto" (SHANNON, 1948, p. 1, tradução nossa). Para isso, ele definiu inicialmente uma *unidade de medida de informação*, que chamou de *bit* (abreviação de *binary digit*) e um *sistema de comunicação*, o qual é formado basicamente por cinco componentes, a saber: *uma fonte de informação*, que produz a mensagem a ser transmitida; *um transmissor*, que atua sobre a mensagem produzindo um sinal passível de ser transmitido; *um canal*, que consiste basicamente no meio utilizado para transmitir o sinal do transmissor até o receptor; *um receptor*, que decodifica o sinal recebido na mensagem enviada pelo transmissor; e *um destino*, que é a pessoa ou equipamento que recebe a mensagem enviada[3]

Tais conceitos foram amplamente utilizados e aproveitados para o estabelecimento da Teoria dos Códigos Corretores de Erros por dois motivos.

[3]Para maiores detalhes sobre os componentes de um sistema de informação, ver Shannon (1948, p. 2) e Gleick (2013, p. 231).

Figura 4.1: Sistema de informação



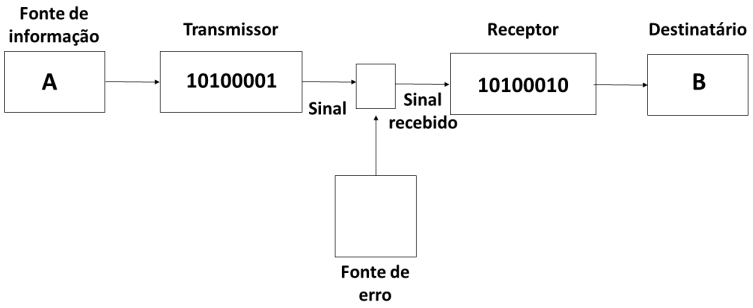
Fonte: Shannon (1948, p. 2).

O primeiro foi o fato de o bit ser adotado como a unidade de medida para a informação, possibilitando, assim, o tratamento científico da informação, o que já ocorria há séculos com outras grandezas, como, por exemplo, as físicas. O segundo foi a sistematização do processo de comunicação, o que, dentre outras coisas, possibilitou uma ampla compreensão de como o erro interfere nesse processo, como também levou Shannon a mostrar que "existe um limite fundamental de quanta informação um canal de comunicação pode transportar" (STEWART, 2013, p. 327). A partir dessa constatação, os cientistas da área buscaram desenvolver métodos e códigos eficientes para a transmissão de informações em suas mais diversas formas, sem, contudo, se preocuparem com a quantidade máxima de informação que um canal poderia transportar, visto que este problema já estava resolvido.

Sobre o papel do erro no processo de comunicação, o diagrama da Figura 4.1 nos mostra que, entre a transmissão e a recepção de uma dada mensagem, pode ocorrer um ruído, ou seja, uma interferência que eventualmente modifica o sentido da mensagem original, causando, assim, um erro de comunicação. Foi precisamente a identificação da presença do ruído interferindo na transmissão de mensagens que levou Shannon e seus companheiros à busca de uma solução para este problema. Essa busca também contribuiu no desenvolvimento dos Códigos Corretores de Erros, cuja função é, como o nome já sugere, impedir, por meio da correção de erros, que a mensagem original tenha seu sentido distorcido após o seu envio.

Por exemplo, a ação do erro na transmissão de um símbolo do código ASCII[4] pode ser representada pelo diagrama de Shannon abaixo. Neste caso, a letra A é codificada pelo símbolo 10100001 e a letra B por 10100010. O erro aqui ocorreu porque os últimos dois dígitos do símbolo enviado foram permutados, o que resultou na transmissão da letra A e da recepção da letra B:

Figura 4.2: Exemplo de erro



Fonte: Ilustração do autor.

4.2.1 O código de Hamming

Um dos pioneiros na pesquisa e no desenvolvimento dos Códigos Corretores de Erros foi o matemático americano Richard Hamming, o qual, em abril de 1950, publicou no *The Bell System Technical Journal* o artigo *Error Detecting and Error Correcting Codes* (HAMMING, 1950). Nesse artigo, conceitos fundamentais para a Teoria dos Códigos Corretores de Erros, como *métrica, redundância, equivalência de códigos e códigos sistemáticos*, por exemplo, foram primeiramente enunciados e abordados. Hamming também explica que a motivação para o estudo foi a necessidade de se resolver o problema que inevitavelmente surge no processamento de uma dada tarefa por uma máquina como um computador, a saber, os eventuais erros que ocorrem na realização da tarefa. Sobre o erro presente em um cálculo realizado por um computador, ele afirmou:

[4]Do inglês: *American Standard Code for Information Interchange*

> uma única falha geralmente significa o fracasso completo, no sentido de que se ela é detectada nenhum cálculo pode ser realizado até a falha ser localizada e corrigida, enquanto que se ela escapa da detecção então ela invalida todas as operações posteriores da máquina (HAMMING, 1950, p. 147, tradução nossa).

Dessa forma, o operador ou usuário de tal máquina se vê diante de um impasse. Se um erro ocorrer e for detectado, então a máquina não funciona até o erro detectado ser corrigido, tarefa que, na época, não era realizada em pouco tempo e sem pouco trabalho. Por outro lado, se um erro ocorrer e não for detectado, os cálculos realizados não serão úteis, pois foram afetados pelo erro, que comprometerá o resultado final de todo o trabalho realizado.

Foi justamente neste novo e pouco explorado contexto que Hamming se encontrava em 1947, enquanto trabalhava com os computadores dos laboratórios Bell. Nessa época, a utilização dos computadores da empresa era bastante restrita e disputada por seus pesquisadores, de forma que Hamming só tinha acesso aos mesmos nos finais de semana. Foi nessas pesquisas de "final de semana" quando ele percebeu que as máquinas por ele utilizadas eram capazes de detectar os erros em sua programação, entretanto isso não o ajudava em nada, pois as máquinas não possuíam a capacidade de corrigir tais erros.

Em entrevista dada em 1977, ele explica a situação em que se encontrava na época, e que, em grande parte, foi um dos motivos que o levaram a trabalhar no desenvolvimento dos Códigos Corretores de Erros:

> Em dois finais de semanas consecutivos eu fui e descobri que todas minhas coisas tinham sido descarregadas e nada tinha sido feito. Eu estava realmente aborrecido e irritado porque queria estas respostas e tinha perdido dois finais de semana. E então eu me disse "Maldição, se as máquinas podem detectar um erro, porque não podemos localizar a posição do erro e corrigi-lo"(MILLIES, 2009, p. 2).

Ao lermos o artigo de Hamming, fica claro que era com o objetivo de compreender e fazer bom uso da correção de erros que ele passou a estudar o problema da sua detecção e posterior correção, uma vez que o problema da simples detecção de erros já estava resolvido na época, como ele mesmo afirma: "parece desejável examinar o próximo passo além da detecção do erro, nomeadamente correção do erro" (HAMMING, 1950, p. 148, tradução nossa). Para desenvolver sua teoria, Hamming elabora alguns conceitos que o ajudarão nessa tarefa. A essência de tais conceitos está presente nas Definições 1, 2 e 3, a seguir:

Definição 1. *Sejam A um conjunto finito não vazio, o qual será chamado de alfabeto, e $|A|$ o seu número de elementos. Um código corretor de erros C é um subconjunto próprio qualquer de A^n, para algum n natural. Um elemento $c \in C$ é chamado um símbolo do código.*

Da definição acima decorre que, dado um conjunto finito não vazio A qualquer, podemos, a partir dele, definir quantos Códigos Corretores de Erros desejarmos, sendo tal construção limitada apenas por nossa criatividade e disposição. Por exemplo, se escolhermos como alfabeto o conjunto $A = \{0,1,2,3,4,5,6,7,8,9\}$, temos que $|A| = 10$ e o seu número de identidade é um símbolo do conjunto $C \subset A^9$, em que C é um Código Corretor de Erros. Agora, se o conjunto A escolhido for o nosso alfabeto, então o conjunto $C \subset A^{46}$, formado por todas as palavras do nosso idioma, também é um Código Corretor de Erros.[5] Por fim, os códigos de barras dos produtos que compramos, o registro de livros ISBN e o número do nosso CPF, são todos exemplos de Códigos Corretores de Erros cujo alfabeto também é $A = \{0,1,2,3,4,5,6,7,8,9\}$ e cujos símbolos estão no conjunto A^{13}, para os dois primeiros, e A^{11}, para o último.

Uma pergunta que pode surgir após esses exemplos é: Como corrigir os erros nesses códigos? Essa pergunta não possui uma única resposta. Nos casos dos números de identidade e das palavras do nosso idioma, por exemplo, a repetição de um símbolo ao transmiti-lo consiste num

[5]Nesse exemplo, consideramos que a maior palavra na língua portuguesa é Pneumoultramicroscopicossilicovulcanoconiótico, a qual, como pode ser visto, possui 46 letras.

procedimento que permite a correção de erros, porém a repetição nem sempre é o procedimento mais eficaz possível. Já para os códigos de barra, o ISBN e o CPF, existem procedimentos matemáticos um pouco mais sofisticados para a detecção e correção dos eventuais erros. Para os interessados em como esses procedimentos funcionam, ver Sá e Rocha (2012).

Sendo assim, surge aqui a necessidade de se buscar procedimentos mais eficazes para a detecção e correção de erros, tarefa essa que nem sempre é simples, principalmente quando trabalhamos com alfabetos com muitos símbolos, como os exemplos acima. Para resolver e evitar este tipo de problema, Hamming escolheu trabalhar com códigos cujos símbolos fossem compostos por sequências numéricas contendo apenas 0's e 1's em seus dígitos. Alguns desses dígitos serão utilizados para transmitir a informação desejada e outros serão utilizados para a detecção e correção dos eventuais erros. Essa escolha nos leva para a próxima Definição:

Definição 2. *Sejam C um Código Corretor de Erros e n, m e k números naturais com $n > m$. Dizemos que C é sistemático quando cada símbolo de C tem exatamente n dígitos binários, dos quais m são associados com a informação, enquanto os $k = n - m$ dígitos restantes são utilizados para a detecção e correção de erros.*

Ao se escolher trabalhar com códigos sistemáticos, nos vemos diante da seguinte pergunta: Dados dois códigos sistemáticos C e C', como decidir qual dos dois é o mais eficiente? Entendendo mais eficiente por aquele que transmite a maior quantidade de informação m, dado um valor para o comprimento dos símbolos n, ou, equivalentemente, transmite uma determinada quantidade m de informação com o menor valor possível de n. Para responder essa pergunta, Hamming propôs a seguinte definição:

Definição 3. *Sejam C um código e n e m naturais. A redundância R do código C é a razão entre o número de dígitos binários utilizados e o número mínimo necessário para transmitir a mesma informação, ou seja, $R = \frac{n}{m}$. Note que a redundância é um número maior ou igual a 1.*

A partir de agora vamos trabalhar com códigos considerando a menor redundância possível, pois essa escolha é exatamente a que Hamming faz em seu artigo. Vale destacar que sempre é possível obtermos tais códigos, que chamaremos de *códigos de redundância mínima*, uma vez que, como será visto posteriormente, m e n estão bem definidos. Além disso, salvo menção contrária, sempre usaremos $A = \{0, 1\}$.

Na primeira parte de seu artigo, Hamming apresenta a construção de códigos de redundância mínima em três casos específicos, a saber:

(1) Códigos *detectores* de um único erro;

(2) Códigos *corretores* de um único erro;

(3) Códigos *corretores* de um único erro, além de *detectores* de erros duplos.

Nas próximas subseções, detalharemos os casos (1) e (2). O caso (3) não será considerado, pois o mesmo consiste simplesmente na aplicação do algoritmo apresentado no caso (1) em um código elaborado conforme o algoritmo apresentado no caso (2). Dessa forma, no caso (3), corrigimos um erro e detectamos dois, um pelo algoritmo em (1) e um pelo algoritmo em (2). Em suma, sempre que falarmos nos códigos dos casos (1) e (2), teremos em mente as seguintes definições:

Definição 4. *Um código C é dito detector de um único erro quando, na transmissão de um dado símbolo $c \in C$, um único erro ocorrido em apenas uma de suas posições pode ser detectado.*

Definição 5. *Um código C é dito corretor de um único erro quando, na transmissão de um dado símbolo $c \in C$, um único erro ocorrido em apenas uma de suas posições pode ser detectado e corrigido pela troca de 0 por 1 ou vice versa.*

4.2.2 Códigos detectores de um único erro

Para o caso mais simples, ou seja, os códigos detectores de um único erro, Hamming propõe o seguinte algoritmo, chamado de *verificação de*

paridade, para a codificação de um símbolo composto de uma lista com n 0's e 1's:

Algoritmo 1. *Nas primeiras $n-1$ posições, nós colocamos $n-1$ dígitos de informação. Na n-ésima posição, nós colocamos outro 0 ou 1, de modo que as n posições completas tenham um número par de 1's.*

Note que o algoritmo acima é claramente um código detector de um único erro, uma vez que um único erro na transmissão deve levar a um número ímpar de 1's nos símbolos do código, o que nos permitirá concluir imediatamente que, de fato, a transmissão foi afetada pelo erro. Esse código é denotado por $C(n,n-1)$ ou $C(n,m)$, em que n é a quantidade de posições dos símbolos do código e m é a quantidade de posições que contém a informação. É possível observar abaixo um exemplo de como este algoritmo de codificação/decodificação funciona para o caso do código $C(8,7)$.

Exemplo 1. *Considerando a Tabela 4.1 a seguir, note que, com respeito aos 7 dígitos de informação, as duas primeiras linhas da tabela contêm um número ímpar de 1's. Portanto, antes de transmitir os símbolos 1000110 e 0010110 presentes nessas linhas, devemos adicionar, na 8ª posição, o dígito 1, para que a quantidade de 1's seja par, resultando nos símbolos codificados 10001101 e 00101101. Por outro lado, os símbolos nas duas últimas linhas contêm um número par de 1's nas 7 posições de informação. Assim, antes de transmitir os símbolos 0111010 e 1010011 presentes nestas linhas, devemos adicionar, na 8ª posição, o dígito 0, para que a quantidade de 1's seja par, resultando nos símbolos codificados 01110100 e 10100110. Dessa forma, se na transmissão o receptor receber um símbolo com um número ímpar de 1's, ele pode concluir que ocorreu um erro na transmissão.*

Cabe notar aqui que, como $R = \frac{n}{m} = \frac{n}{n-1} = 1 + \frac{1}{n-1}$, poderíamos supor que, para obtermos uma redundância cada vez menor, deveríamos tornar o valor de n cada vez maior. Porém o que ocorre ao se aumentar o valor de n é o indesejável aumento na probabilidade de ocorrência de erros na transmissão dos símbolos. Em suas palavras, Hamming explica: "se

Tabela 4.1: Funcionamento do código $C(8,7)$, detector de um único erro.

Dígitos de informação							**Dígito de verificação**
1	0	0	0	1	1	0	1
0	0	1	0	1	1	0	1
0	1	1	1	0	1	0	0
1	0	1	0	0	1	1	0

Fonte: Elaborada pelo autor.

$p \ll 1$ é a probabilidade de algum erro, então para n tão grande como $\frac{1}{p}$, a probabilidade de um símbolo correto é aproximadamente $\frac{1}{e} = 0,3679\ldots$, enquanto um erro duplo tem probabilidade $\frac{1}{2e} = 0,1839\ldots$." (ibid, p. 150). Como os erros duplos não são detectados por esse código, ocorre que existe uma probabilidade de aproximadamente $18,4\%$ de surgirem erros duplos passando pelo sistema, ou seja, quase um em cada cinco símbolos sendo transmitidos com erros e, pior ainda, não detectados.

Antes de passar para o próximo tipo de código, vale ressaltar que o código do exemplo acima é, de acordo com a Definição 1, um subconjunto de A^8, em que A, como já dissemos, é o conjunto $\{0,1\}$. Além disso, a redundância desse código é $R = \frac{n}{m} = \frac{8}{7} \approx 1,14$, o que, em termos práticos, significa que a transmissão dos $2^7 = 128$ símbolos desse código após sua codificação é equivalente à transmissão de $128 \times \frac{8}{7} \approx 146$ símbolos do mesmo código se eles não fossem codificados. Por esse motivo, trabalhamos com códigos com redundância mínima, pois eles possibilitam uma maior economia de dados na transmissão de uma dada informação.

4.2.3 Códigos corretores de um único erro

No caso dos códigos corretores de um único erro, Hamming desenvolve dois algoritmos. Um será utilizado para a *codificação* e outro será utilizado para a *detecção, correção* e *decodificação* de um erro em uma sequência binária de n posições, das quais m são escolhidas para conter a informação e as outras k restantes, em que k é tal que $k = n - m$, são escolhidas para a *verificação de paridade*. A relação entre n, m e k é dada pela Tabela 4.2,

que apenas utilizaremos aqui, deixando a sua construção para a subseção 2.4 (Proposição 1). No próximo exemplo, adaptado de Hefez (2008, p. 2), mostraremos como a Tabela 4.2 é utilizada na codificação de comandos para a movimentação de um robô que se move em 4 direções sobre um tabuleiro.

Tabela 4.2: Relação entre n, m e k.

n	m	k correspondente
1	0	1
2	0	2
3	1	2
4	1	3
5	2	3
6	3	3
7	4	3
8	4	4
9	5	4
10	6	4
11	7	4
12	8	4
13	9	4
14	10	4
15	11	4
16	11	5
	Etc.	

Fonte: Hamming (1950, p. 151).

Exemplo 2. *Considere um robô que se move sobre um tabuleiro quadriculado de modo que, ao darmos um dos comandos (Leste, Oeste, Norte ou Sul), o robô se desloca do centro de uma casa para o centro da casa contígua indicada pelo comando.*
Se definirmos estes comandos por: Leste $\longmapsto 00$*, Oeste* $\longmapsto 01$*, Norte* $\longmapsto 10$ *e Sul* $\longmapsto 11$*, a Tabela 4.2 mostra que* 2 *dígitos de informação* ($m = 2$)*, exigem* 3 *dígitos de verificação* ($k = 3$)*, logo, os símbolos codificados terão* 5 *posições* ($n = 5$)*.*

Segue daí que uma codificação para estes comandos é a dada por: $00 \longmapsto 00\mathbf{0}0\mathbf{0}$, $01 \longmapsto 10\mathbf{0}1\mathbf{1}$, $10 \longmapsto 11\mathbf{1}0\mathbf{0}$ e $11 \longmapsto 01\mathbf{1}1\mathbf{1}$. Em que os dígitos destacados em negrito (informação) são os comandos originais do robô pré-codificação, e os outros dígitos que aparecem sem negrito (redundância) estão todos em posições que são potências de 2. Essa não é a única forma de codificar os comandos do robô, mas, como veremos à seguir (Algoritmo 2), é uma que permite a detecção e correção de um único erro. De fato, a ocorrência de um único erro antes da codificação, como, por exemplo, o envio de 00 ao invés de 01, faria com que o robô se movimentasse na direção oposta à que queríamos que ele fosse, porém, após a codificação, a ocorrência de um único erro não gera tal situação e, mais ainda, é passível de ser corrigida, como veremos a seguir.

Num primeiro momento, o leitor pode pensar que a escolha dessa codificação particular foi feita de forma arbitrária, mas, ao contrário do que pode parecer, a mesma foi realizada seguindo um algoritmo definido em Hamming (1950), o qual exemplificaremos a seguir e generalizaremos no Algoritmo 2.

Em seu algoritmo de codificação, Hamming afirma que as posições de verificação devem estar localizadas em potências de 2, ou seja, as posições de verificação serão a 1ª, 2ª e 4ª, como vimos no Exemplo 2. Por questões de melhor entendimento, chamaremos essas posições de v_1, v_2 e v_4 e destacaremos, em negrito, os símbolos $00, 01, 10, 11$, de sorte que tais símbolos, ao serem codificados, serão escritos como: $v_1v_2\mathbf{0}v_4\mathbf{0}$, $v_1v_2\mathbf{0}v_4\mathbf{1}$, $v_1v_2\mathbf{1}v_4\mathbf{0}$ e $v_1v_2\mathbf{1}v_4\mathbf{1}$. Para determinar v_1, v_2 e v_4, seguiremos o seguinte algoritmo:

- Para a codificação do símbolo $\mathbf{00}$ em $v_1v_2\mathbf{0}v_4\mathbf{0}$, v_1 será escolhido de forma que a soma $v_1 + \mathbf{0} + \mathbf{0}$ seja par; v_2, de forma que a soma $v_2 + \mathbf{0}$ seja par; e v_4, de forma que a soma $v_4 + \mathbf{0}$ seja par. Logo, teremos $v_1 = 0, v_2 = 0$ e $v_4 = 0$ e o símbolo codificado será $00\mathbf{0}0\mathbf{0}$.

- Para a codificação do símbolo $\mathbf{01}$ em $v_1v_2\mathbf{0}v_4\mathbf{1}$, v_1 será escolhido de forma que a soma $v_1 + \mathbf{0} + \mathbf{1}$ seja par; v_2, de forma que a soma $v_2 + \mathbf{0}$ seja par; e v_4, de forma que a soma $v_4 + \mathbf{1}$ seja par. Logo, teremos $v_1 = 1, v_2 = 0$ e $v_4 = 1$ e o símbolo codificado será $10\mathbf{0}1\mathbf{1}$.

- Para a codificação do símbolo **10** em $v_1v_2\mathbf{1}v_4\mathbf{0}$, v_1 será escolhido de forma que a soma $v_1+\mathbf{1}+\mathbf{0}$ seja par; v_2, de forma que a soma $v_2+\mathbf{1}$ seja par; e v_4, de forma que a soma $v_4+\mathbf{0}$ seja par. Logo, teremos $v_1=1, v_2=1$ e $v_4=0$ e o símbolo codificado será 11**10**0.

- Para a codificação do símbolo **11** em $v_1v_2\mathbf{1}v_4\mathbf{1}$, v_1 será escolhido de forma que a soma $v_1+\mathbf{1}+\mathbf{1}$ seja par; v_2, de forma que a soma $v_2+\mathbf{1}$ seja par; e v_4, de forma que a soma $v_4+\mathbf{1}$ seja par. Logo, teremos $v_1=0, v_2=1$ e $v_4=1$ e o símbolo codificado será 01**11**1.

Assim, obtemos os símbolos codificados do Exemplo 2. Vejamos, agora, o caso geral para um símbolo $v_1v_2d_3v_4d_5d_6d_7v_8\cdots$ codificado a partir do símbolo $d_3d_5d_6d_7\cdots$.

Algoritmo 2. ***(Codificação)*** *Para determinar v_1, some os valores dos dígitos nas posições $1,3,5,7,\cdots$ de forma que a soma seja par, ou seja, "escolha" um dígito e "pule" um dígito a partir da 1ª posição. Para determinar v_2 some os valores dos dígitos nas posições $2,3,6,7,10,11,\ldots$ de forma que a soma seja par, ou seja, "escolha" dois dígitos e "pule" dois dígitos a partir da 2ª posição. Para determinar v_4 some os valores dos dígitos nas posições $4,5,6,7,12,13,14,15,\cdots$ de forma que a soma seja par, ou seja, "escolha" quatro dígitos e "pule" quatro dígitos a partir da 4ª posição. Este algoritmo continua até que sejam percorridas todas as posições nas potências de 2 do símbolo, sempre "escolhendo" e "pulando" dígitos nas potências de dois.*

Suponha agora, que, ao enviarmos o comando para o robô se movimentar para o norte, tenha ocorrido um erro e, ao invés de ser transmitido o símbolo 11100, tenha sido transmitido o símbolo 11000, com um erro na terceira posição. Como verificar e corrigir esse erro? Hamming nos responde com mais um algoritmo.

Algoritmo 3. ***(Decodificação e correção)*** *Vamos imaginar por um momento, que recebemos um símbolo de código, com ou sem um erro. Vamos aplicar as k verificações de paridade em ordem, e, para cada vez que a*

verificação de paridade especificar o valor observado em sua verificação de posição, escreveremos um 0, enquanto que, para cada vez que os valores especificado e observado diferirem, escreveremos um 1. Quando escrevermos, da direita para a esquerda, em uma linha, esta sequência de k 0's e 1's [...] ela poderá ser considerada como um número binário e será chamada de um número de verificação. Vamos exigir que esse número de verificação dê a posição de um único erro, com o valor zero significando nenhum erro no símbolo (HAMMING, 1950, p. 150, tradução nossa).

Vamos agora aplicar o Algoritmo 3 no símbolo 11000 e constatar que de fato o erro está na 3ª posição. Com efeito, para esse símbolo temos $v_1 = 1, v_2 = 1$ e $v_4 = 0$, de maneira que:

- A primeira verificação de paridade é realizada nas posições $1, 3$ e 5, logo, para que $v_1 + 0 + 0$ seja par, v_1 tem que ser igual a zero, o que não confere com o valor de v_1. Assim, essa verificação contribui com um 1 na sequência do número de verificação.

- A segunda verificação de paridade é realizada nas posições 2 e 3, de sorte que, para que $v_2 + 0$ seja par, v_2 tem que ser igual a zero, o que não confere com o valor de v_2. Assim, essa verificação também contribui com um 1 na sequência do número de verificação.

- A terceira e última verificação de paridade é realizada nas posições 4 e 5, de maneira que, para que $v_4 + 0$ seja par, v_4 tem que ser igual a zero, o que confere com o valor de v_4. Assim, essa verificação contribui com um 0 na sequência do número de verificação.

Escrevendo essa sequência como indicado no Algoritmo 3, obtemos a sequência 011, que pode ser identificada com o número 011 na base 2, que é igual a 3 na base 10. Dessa maneira, o erro se encontra na 3ª posição, como já era de se esperar. Na próxima subseção, explicaremos com detalhes por que esse algoritmo funciona e por que a sequência obtida representa, de fato, a posição onde se encontra o erro. Consideremos, agora, um exemplo em que o robô do Exemplo 2 é atualizado para se movimentar em mais quatro direções:

Exemplo 3. *Suponha que o robô do Exemplo 2 foi aprimorado, de forma que também seja possível movimentá-lo nas direções: Nordeste, Noroeste, Sudeste e Sudoeste. Se redefinirmos os comandos por: Leste* $\longmapsto 000$*, Oeste* $\longmapsto 010$*, Norte* $\longmapsto 100$*, Sul* $\longmapsto 110$*, Nordeste* $\longmapsto 001$*, Noroeste* $\longmapsto 011$*, Sudeste* $\longmapsto 101$ *e Sudoeste* $\longmapsto 111$*, o Algoritmo 2 e a Tabela 4.2 (3 dígitos de informação* $m = 3$*, requerem 3 dígitos de verificação* $k = 3$*) nos fornecerão a seguinte codificação:* $000 \longmapsto 00\boldsymbol{0}0\boldsymbol{0}$*,* $010 \longmapsto 10\boldsymbol{0}1\boldsymbol{1}\boldsymbol{0}$*,* $100 \longmapsto 11\boldsymbol{1}0\boldsymbol{0}\boldsymbol{0}$*,* $110 \longmapsto 01\boldsymbol{1}1\boldsymbol{1}\boldsymbol{0}$*,* $001 \longmapsto 01\boldsymbol{0}1\boldsymbol{0}\boldsymbol{1}$*,* $011 \longmapsto 11\boldsymbol{0}0\boldsymbol{1}\boldsymbol{1}$*,* $101 \longmapsto 10\boldsymbol{1}1\boldsymbol{0}\boldsymbol{1}$ *e* $111 \longmapsto 00\boldsymbol{1}0\boldsymbol{1}\boldsymbol{1}$*, em que os dígitos destacados em negrito correspondem aos símbolos antes da codificação.*

A forma de realizar a codificação desses símbolos é a mesma realizada no Exemplo 2, portanto não a repetiremos aqui. Entretanto, vamos apresentar uma forma mais direta para a verificação e correção do erro, ou seja, de aplicação do Algoritmo 3. Suponha que, ao darmos o comando para o robô se movimentar na direção nordeste, o símbolo 010001 tenha sido transmitido em vez do símbolo 010101, ou seja, ocorreu um erro na 4ª posição. Para detectar e corrigir esse erro, considere a Tabela 4.3 abaixo.

Tabela 4.3: Correção de um erro

v_1	v_2	d_3	v_4	d_5	d_6	Número de Verificação
0	1	0	0	0	1	
0		0		0		0
	1	0			1	0
			0	0	1	1

Fonte: Elaborada pelo autor.

Na primeira linha da tabela, rotulamos os dígitos que aparecerão nas colunas de 1 a 6 por v_1, v_2, d_3, v_4, d_5 e d_6, em que v_1, v_2 e v_4 são os dígitos de verificação e d_3, d_5 e d_6 são os dígitos de informação (os escritos em negrito no Exemplo 3). Na segunda linha da tabela, temos o símbolo que foi recebido na transmissão, a saber, 010001. Agora, note que os três zeros que aparecem na 3ª linha das seis primeiras colunas da tabela são os

valores de v_1, d_3 e d_5 e que a soma de d_3 com d_5 é par e, como $v_1 = 0$, essa verificação contribui com um 0 para o número de verificação (3ª linha e 7ª coluna). Prosseguindo a verificação, temos que a soma dos valores de d_3 e d_6, presentes na 4ª linha, é ímpar e, como $v_2 = 1$, essa verificação contribui com um 0 para o número de verificação. Finalmente, somando os valores de d_5 e d_6 na última linha, obtemos resultado ímpar e, como $v_4 = 0$, essa verificação contribui com um 1 para o número de verificação. Escrevendo o número de verificação em sua forma binária, obtemos 100, que em escrita decimal é igual a 4, ou seja, o erro se encontra na 4ª posição, como era de se esperar.

Vejamos agora um exemplo no qual consideraremos o caso do envio de um símbolo sem erro e verificaremos que o Algoritmo 3 retorna, de fato, uma sequência contendo apenas zeros como indicação da não ocorrência de erro.

Exemplo 4. *Seja* $x = 01001110$ *um símbolo pertencente a um código que transmite símbolos contendo um byte de informação, ou seja,* $m = 8$*. Para codificá-lo, consultamos a Tabela 4.2 e notamos que, para este valor de* m*, devemos escolher* $k = 4$ *e* $n = 12$*, logo, o símbolo* x*, ao ser codificado, terá 12 posições. Após utilizarmos o Procedimento 2, obtemos o símbolo* $x' = 100110011110$*, a codificação de* x*. Suponha que tal símbolo tenha sido transmitido corretamente, assim, ao utilizarmos o Procedimento 3 e calcularmos o número de verificação desse símbolo, devemos obter a sequência* 0000*, a qual indicará que não houve erro na transmissão. De fato, considerando a Tabela 4.4 abaixo, é fácil verificar que os valores dos dígitos na última coluna da mesma são realmente todos iguais a zero.*

Com esses exemplos, encerramos a apresentação do código de Hamming em sua formulação original. A seguir, mostraremos por que esses procedimentos de codificação e correção funcionam, para finalmente, na subseção 2.5, apresentar a formulação matricial desse código.

Tabela 4.4: Verificação da não ocorrência de erro

v_1	v_2	d_3	v_4	d_5	d_6	d_7	v_8	d_9	d_{10}	d_{11}	d_{12}	Número de Verificação
1	0	0	1	1	0	0	1	1	1	1	0	
1		0		1		0		1		1		0
	0	0			0	0			1	1		0
			1	1	0	0					0	0
							1	1	1	1	0	0

Fonte: Elaborada pelo autor.

4.2.4 Justificativa dos algoritmos e construção da tabela 2

A pergunta natural que surge nesse momento é: Por que estes algoritmos de codificação, decodificação e correção funcionam? A resposta para essa pergunta pode ser encontrada na relação existente entre os números escritos nas bases 2 e 10. Para entender melhor essa afirmação, consideremos o teorema a seguir e o seu corolário mais adiante, cujas demonstrações podem ser encontradas em Hefez (2014, p. 68; 73), respectivamente.

Teorema 1. *Sejam dados os números inteiros a e b, com $a > 0$ e $b > 1$. Existem números inteiros $n \geqslant 0$ e $0 \leqslant r_0, r_1, \ldots, r_n < b$, com $r_n \neq 0$, univocamente determinados, tais que $a = r_0 + r_1 b + r_2 b^2 + \cdots + r_n b^n$.*

Note que esse teorema garante que podemos escrever um número a dado, na base $b > 1$ que preferirmos. Em particular, quando $b = 10$, dizemos que o número a está escrito na base 10 ou em sua expansão decimal e escrevemos $(a)_{10}$, enquanto que, quando $b = 2$, dizemos que o número a está escrito na base 2 ou em sua expansão binária e escrevemos $(a)_2$. O corolário a seguir nos permite relacionar um número em sua representação na base 10 com a sua respectiva representação na base 2 e vice-versa. Tal relação, embora não tenha sido explicitada, está no cerne dos algoritmos de codificação, decodificação e detecção de erro desenvolvidos por Hamming.

corolário 1. *Todo número natural a escreve-se de modo único como soma de potências distintas de 2, a saber, $a = r_n \times 2^n + r_{n-1} \times 2^{n-1} + \ldots r_1 \times$*

$2^1 + r_0 \times 2^0$, *com* $r_i \in \{0,1\}$.

Exemplo 5. *Segundo o corolário anterior, o número* $(739)_{10}$ *é escrito, utilizando-se apenas potências de 2, como* $1\times 2^9 + 0\times 2^8 + 1\times 2^7 + 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$. *Ou, de forma mais sucinta,* $(739)_{10} = (1011100011)_2$.

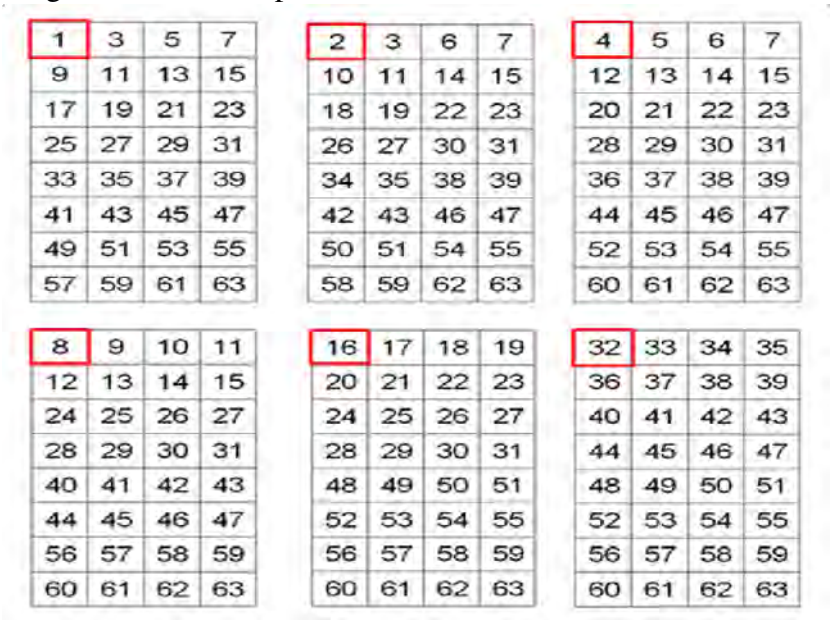
O exemplo a seguir é bastante esclarecedor para a compreensão do Algoritmo 3:

Exemplo 6. *Os cartões da Figura 4.3 abaixo podem ser utilizados para representar qualquer número natural entre 1 e 63 como a soma de potências de dois. Note ainda que a obtenção do número 39, por exemplo, é feita escolhendo os cartões que começam com* $1 = 2^0, 2 = 2^1, 4 = 2^2$ *e* $32 = 2^5$, *respectivamente, ou seja,* $39 = 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$ *ou, se preferirmos,* $(39)_{10} = (100111)_2$. *Esses cartões são também os únicos nos quais figura o número* 39.

Perceba, porém, que o que fizemos no Exemplo 6 é exatamente o que Hamming faz no Algoritmo 3. De fato, no Algoritmo 2 (para codificar um símbolo) Hamming escolhe somar os dígitos nas posições $1,3,5,7,9,11,13,15\ldots$ na obtenção de v_1; somar os dígitos nas posições $2,3,6,7,10,11,14,15\ldots$ na obtenção de v_2; somar os dígitos nas posições $4,5,6,7,12,13,14,15\ldots$ na obtenção de v_4; e assim por diante. Esses números que aparecem aqui são exatamente os que figuram nos 1º, 2º e 3º cartões da Figura 4.3, respectivamente. Desta forma, ao obter $v_1, v_2, v_4, \ldots$ por esse algoritmo, Hamming "prepara o caminho" para a utilização do Algoritmo 3.

Com efeito, de acordo com o Algoritmo 3, se o valor obtido na primeira verificação coincidir com o valor de v_1, escrevemos um 0, caso contrário, escrevemos um 1. Semelhantemente, procedemos para $v_2, v_4, \ldots$ até percorrermos todas as posições de verificação do símbolo. Desta forma, ao obtermos a sequência de k $0's$ e $1's$ ao final do cálculo envolvendo todas as posições de verificação, ela, de fato, representará um número escrito na base 2, pois escrever um 0 ou um 1 em cada etapa implica em escolher

Figura 4.3: Cartões para obter um número natural entre 1 e 63



1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31
33	35	37	39
41	43	45	47
49	51	53	55
57	59	61	63

2	3	6	7
10	11	14	15
18	19	22	23
26	27	30	31
34	35	38	39
42	43	46	47
50	51	54	55
58	59	62	63

4	5	6	7
12	13	14	15
20	21	22	23
28	29	30	31
36	37	38	39
44	45	46	47
52	53	54	55
60	61	62	63

8	9	10	11
12	13	14	15
24	25	26	27
28	29	30	31
40	41	42	43
44	45	46	47
56	57	58	59
60	61	62	63

16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63

32	33	34	35
36	37	38	39
40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63

Fonte:https://www.ticsnamatematica.com/2014/11/entenda-como-construir-cartoes-jogo-adivinhe-idade.html

ou não um dos cartões da Figura 4.3, e tal escolha significa, tão somente, escrever um número natural como a soma de potências de dois.

Para encerrar esta subseção, vamos mostrar como os valores de n, m e k presentes na Tabela 4.2 foram obtidos. Para isso, considere a seguinte proposição que relaciona o número de verificação com os valores de n, m e k:

Proposição 1. *Sejam $C \in A^n$, um código corretor de erros, e n, m e k naturais, tais que m é o número de posições de informação, k é o número de posições de verificação dos símbolos do código e $n = m + k$, vale a seguinte relação entre n e m: $\frac{2^n}{n+1} \geqslant 2^m$.*

Demonstração. De fato, note que o número de verificação deve descrever $m + k + 1$ possibilidades diferentes, a saber, $n = m + k$ posições que dizem respeito a um erro em qualquer posição no símbolo, mais uma possibilidade no caso da não existência de erro. Isso implica na necessidade de ser

$2^k \geqslant m+k+1$, uma vez que 2^k é o número de sequências com k posições contendo apenas $0's$ e $1's$. Utilizando o fato de que $n = m+k$, obtemos $2^{n-m} \geqslant n+1 \Rightarrow \frac{2^n}{2^m} \geqslant n+1 \Rightarrow \frac{2^n}{n+1} \geqslant 2^m$, o que prova o resultado. □

Com essa proposição, concluímos que atribuindo valores para n, ou seja, escolhendo a quantidade de posições que os símbolos de C possuirão, a inequação acima nos fornece o maior valor possível para m, ou seja, a maior quantidade de posições de informação que os símbolos de C possuirão. Por outro lado, feita a escolha de m, a mesma inequação nos fornece o menor valor para n, ou seja, os símbolos com menor tamanho para o código C contendo uma certa quantidade de informação. Dessa forma, a inequação acima nos permite escrever o código que carregue a maior quantidade de informação possível com a maior economia possível.

Note que, se no lugar de considerarmos a inequação $2^k \geqslant m+k+1$, como fizemos anteriormente, nós considerarmos apenas a igualdade $2^k = m+k+1$, ou seja, se o número de verificação nos der exatamente $m+k+1$ posições diferentes, e sabendo que $n = m+k$, segue que $m = 2^k-k-1$ e $n = 2^k-1$. Logo, ao representarmos um código de Hamming na forma $C(n,m)$, o mesmo será descrito por $C(2^k-1, 2^k-k-1)$ e é justamente para essa família de códigos que daremos uma abordagem matricial na próxima subseção. Códigos que satisfazem essa condição são ditos *perfeitos*. Para demonstrações de que o código de Hamming é perfeito, ver Hefez (2008, p. 100) e Shine (2009, p. 301).

4.2.5 O código $C(7,4)$ e a família de códigos $C(2^k-1, 2^k-k-1)$

Agora que estudamos o código de Hamming em sua formulação original, daremos mais um passo em nosso estudo apresentando uma formulação mais recente do mesmo, utilizando ferramentas advindas da Teoria das Matrizes. Isso nos permitiu construir uma sequência didática, na qual alguns conceitos estudados no Ensino Médio, como por exemplo, a multiplicação de matrizes e a transposta de uma matriz, foram abordados. Para os

interessados na sequência didática, ver Lira (2018a) e Lira (2018b).

Isso posto, seguiremos de perto as ideias desenvolvidas em Rousseau e Aubin (2015), fazendo as devidas modificações e alterações para tornar o texto mais acessível, pensando em sua aplicação na Educação Básica. Dessa forma, trazemos uma teoria geral para os códigos $C(2^k-1, 2^k-k-1)$, paralelamente a uma visão particular sobre o código $C(7,4)$, onde os dígitos de informação são $m=4$ e os de verificação são $k=3$. Esse código codifica todas as sequências binárias contendo 4 elementos, ou seja, $16=2^4$ símbolos, que vão de 0000 até 1111. Nosso objetivo aqui é mostrar como funcionam a codificação, a decodificação e a correção de um único erro desses símbolos de uma forma diferente, porém equivalente a apresentada por Hamming (1950). A diferença aqui é que, em vez de colocarmos os dígitos de verificação nas potências de 2, nós os colocaremos em posições diferentes dessas, a saber, nas últimas posições do símbolo, porém com a mesma verificação de paridade utilizada na subseção 2.4.

Para deixarmos nossa exposição alinhada com a encontrada nos trabalhos atuais, consideraremos com mais detalhes o papel do conjunto $\mathbb{Z}_2$ na construção desses códigos. Para isso, vamos construí-lo a partir da ideia de congruência módulo 2. Considere a seguinte definição:

Definição 6. *Sejam a, b e m inteiros com $m>1$. Dizemos que a e b são congruentes módulo m e denotamos $a \equiv b \ (mod\ m)$, quando a e b deixam o mesmo resto na divisão euclidiana por m.*

Neste trabalho, estamos interessados apenas no caso em que $m=2$. Como o resto na divisão euclidiana de um número por 2 só pode ser ou 0 ou 1, podemos classificar todos os números inteiros em dois grupos (ou conjuntos): os números que deixam resto 0, que estão reunidos no conjunto denotado por $\overline{0} = \{a \in \mathbb{Z};\ a \equiv 0\ (mod\ 2)\}$ e os números que deixam resto 1, que estão reunidos no conjunto denotado por $\overline{1} = \{a \in \mathbb{Z};\ a \equiv 1\ (mod\ 2)\}$. Note que $\overline{0} = \{\cdots, -4, -2, 0, 2, 4, \cdots\}$ e $\overline{1} = \{\cdots, -3, -1, 1, 3, \cdots\}$. Usualmente o conjunto $\mathbb{Z}_2$ é representado como $\mathbb{Z}_2 = \{\overline{0}, \overline{1}\}$, porém, por simplicidade, denotá-lo-emos por $\mathbb{Z}_2 = \{0, 1\}$, tendo sempre em mente que a soma de dois elementos de $\overline{1}$ resulta em um elemento de $\overline{0}$, pois a soma

de dois números que deixam resto 1 na divisão por 2 (ímpares) resultará em um número que deixa resto 0 na divisão por 2 (par). Assim, podemos concluir que em $\mathbb{Z}_2$ vale a relação $1+1=0$.

Isso posto, consideremos o símbolo $x = 0101$ e vejamos como o codificar e o decodificar, além de corrigir um único erro em uma de suas posições. Descrevendo os dígitos dos símbolos codificados da esquerda pra direita, os quatro primeiros serão os dígitos de informação, d_1, d_2, d_3 e d_4, e os três últimos, os de verificação, v_5, v_6 e v_7. O cálculo dos dígitos de verificação, e consequentemente sua codificação, é realizado através das igualdades: $v_5 = d_1 + d_2 + d_4, v_6 = d_1 + d_3 + d_4$ e $v_7 = d_2 + d_3 + d_4$, cuja disposição explicaremos mais adiante. Dessa forma, o símbolo codificado tem a representação $d_1d_2d_3d_4v_5v_6v_7$, na qual v_5, v_6 e v_7 são como postos acima. Assim, para o símbolo $x = 0101$, temos $d_1 = 0, d_2 = 1, d_3 = 0, d_4 = 1, v_5 = d_1 + d_2 + d_4, v_6 = d_1 + d_3 + d_4$ e $v_7 = d_2 + d_3 + d_4$. Logo, ao codificá-lo, obtemos o símbolo $x' = 0101010$.

Note que, na codificação dada pelo Algoritmo 2, o símbolo codificado tem a representação $v_1v_2d_3v_4d_5d_6d_7$, onde v_1 é escolhido de forma que a soma $v_1 + d_3 + d_5 + d_7$ seja par; v_2, de forma que a soma $v_2 + d_3 + d_6 + d_7$ seja par; e v_4, de forma que a soma $v_4 + d_5 + d_6 + d_7$ seja par, o que é o mesmo que:

$$\begin{aligned} v_1 &= d_3 + d_5 + d_7 \\ v_2 &= d_3 + d_6 + d_7 \\ v_4 &= d_5 + d_6 + d_7. \end{aligned} \tag{4.1}$$

Daí segue que a codificação, agora escrita como $d_1d_2d_3d_4v_5v_6v_7$, é equivalente à codificação da subseção 2.4, escrita como $v_1v_2d_3v_4d_5d_6d_7$, na qual a relação entre as duas codificações, quanto aos dígitos de verificação, é dada pela Tabela 4.5. Note também que, nessa forma de codificação, a relação com a codificação da subseção 2.4 é a seguinte: v_5 faz o papel de v_1; v_6 faz o papel de v_2; v_7 faz o papel de v_4; e d_1, d_2, d_3 e d_4 fazem o papel de d_3, d_5, d_6 e d_7, respectivamente. Dessa forma, para codificar um símbolo do código $C(7,4)$, utilizamos o seguinte algoritmo:

Algoritmo 4. *Para codificar um símbolo $d_1d_2d_3d_4v_5v_6v$ do código $C(7,4)$, utilize o Algoritmo 2 com a equivalência da Tabela 4.5.*

Tabela 4.5: Equivalência entre os dígitos v_5, v_6 e v_7 e v_1, v_2 e v_4

Codificação na subseção 2.4	v_1	v_2	d_3	v_4	d_5	d_6	d_7
Codificação equivalente	v_5	v_6	d_1	v_7	d_2	d_3	d_4

Fonte: Elaborada pelo autor

Vale destacar aqui que uma forma prática para codificar qualquer símbolo do código $C(7,4)$ é através da soma das colunas da Tabela 4.6 abaixo. Assim, se, por exemplo, considerarmos o símbolo $y = 0111$, temos $d_1 = 0, d_2 = 1, d_3 = 1$ e $d_4 = 1$, de modo que, ao substituirmos esses valores na Tabela 4.6 e somarmos suas respectivas colunas, obteremos o símbolo codificado $y' = 0111001$.

Tabela 4.6: Esquema para a codificação de um símbolo de $C(7,4)$

d_1	d_2	d_3	d_4	v_5	v_6	v_7
$1 \cdot d_1$	$0 \cdot d_1$	$0 \cdot d_1$	$0 \cdot d_1$	$1 \cdot d_1$	$1 \cdot d_1$	$0 \cdot d_1$
$0 \cdot d_2$	$1 \cdot d_2$	$0 \cdot d_2$	$0 \cdot d_2$	$1 \cdot d_2$	$0 \cdot d_2$	$1 \cdot d_2$
$0 \cdot d_3$	$0 \cdot d_3$	$1 \cdot d_3$	$0 \cdot d_3$	$0 \cdot d_3$	$1 \cdot d_3$	$1 \cdot d_3$
$0 \cdot d_4$	$0 \cdot d_4$	$0 \cdot d_4$	$1 \cdot d_4$	$1 \cdot d_4$	$1 \cdot d_4$	$1 \cdot d_4$

Fonte: Elaborada pelo autor

Agora que já sabemos codificar um símbolo de $C(7,4)$, a pergunta que surge é: Como decodificar e corrigir um erro de um símbolo desse código? Para responder a essa pergunta, precisamos detectar se existe um erro, sua posição e corrigi-lo, trocando 0 por 1 ou vice-versa. Isso é feito de acordo com o seguinte algoritmo, apresentado em Rousseau e Aubin (2015):

Algoritmo 5. *Para detectar um possível erro, calculamos os dígitos de verificação do símbolo recebido, os quais denotaremos por w_5, w_6 e w_7, e depois os comparamos com os respectivos valores, nas posições de verificação, do símbolo recebido. Uma das possibilidades a seguir pode ocorrer:*

1. $v_5 = w_5, v_6 = w_6$ *e* $v_7 = w_7$*, nesse caso, o símbolo foi enviado sem erro;*

2. $v_5 \neq w_5$ *e* $v_6 \neq w_6$*, nesse caso, o erro está na primeira posição;*

3. $v_5 \neq w_5$ *e* $v_7 \neq w_7$*, nesse caso, o erro está na segunda posição;*

4. $v_6 \neq w_6$ *e* $v_7 \neq w_7$*, nesse caso, o erro está na terceira posição;*

5. $v_5 \neq w_5$*,* $v_6 \neq w_6$ *e* $v_7 \neq w_7$*, nesse caso, o erro está na quarta posição;*

6. $v_5 \neq w_5$*, nesse caso, o erro está na quinta posição;*

7. $v_6 \neq w_6$*, nesse caso, o erro está na sexta posição;*

8. $v_7 \neq w_7$*, nesse caso, o erro está na sétima posição.*

Antes de mostrarmos esse algoritmo em ação, entendemos que cabe aqui uma breve explicação, caso a caso, do porquê do seu funcionamento.

1. No caso 1, não temos muito o que explicar, pois todos as verificações de paridade coincidem, logo, o simbolo enviado foi recebido sem erro.

2. No caso 2, ao notarmos que $v_5 \neq w_5$ e $v_6 \neq w_6$, concluímos que $v_7 = w_7$ e, como $v_7 = d_2 + d_3 + d_4$, o erro não pode estar em d_2, d_3 ou d_4, logo, só pode estar em d_1, pois é a discrepância de valores nesse dígito que faz com que $v_5 \neq w_5$ e $v_6 \neq w_6$.

3. No caso 3, ao notarmos que $v_5 \neq w_5$ e $v_7 \neq w_7$, concluímos que $v_6 = w_6$ e, como $v_6 = d_1 + d_3 + d_4$, o erro não pode estar em d_1, d_3 ou d_4, logo, só pode estar em d_2, pois é a discrepância de valores nesse dígito que faz com que $v_5 \neq w_5$ e $v_7 \neq w_7$.

4. No caso 4, ao notarmos que $v_6 \neq w_6$ e $v_7 \neq w_7$, concluímos que $v_5 = w_5$ e, uma vez que $v_5 = d_1 + d_2 + d_4$, o erro não pode estar em d_1, d_2 ou d_4, logo só pode estar em d_3, pois é a discrepância de valores nesse dígito que faz com que $v_6 \neq w_6$ e $v_7 \neq w_7$.

5. No caso 5, ao notarmos que $v_5 \neq w_5$, $v_6 \neq w_6$ e $v_7 \neq w_7$, concluímos que o erro só pode estar em um dígito que é comum a v_5, v_6 e v_7, o qual, como pode ser visto facilmente, é d_4, pois é a discrepância de valores nesse dígito que faz com que $v_5 \neq w_5, v_6 \neq w_6$ e $v_7 \neq w_7$.

6. Nos casos 6, 7 e 8, ao notarmos que $v_5 \neq w_5, v_6 \neq w_6$ e $v_7 \neq w_7$, respectivamente, tendo em vista as observações anteriores, só podemos concluir que o erro ocorreu na posição v_5, v_6 ou v_7, respectivamente.

Vejamos em um exemplo como esse procedimento funciona na correção de um único erro de um símbolo.

Exemplo 7. *Suponha que o símbolo* $x' = 0101010$ *tenha sido transmitido com um erro na quinta posição, ou seja, o símbolo recebido foi* $x'' = 0101110$.
Aplicando o Algoritmo 5 ao símbolo recebido, temos que $w_5 = d_1 + d_2 + d_4 = 0+1+1 = 0, w_6 = d_1 + d_3 + d_4 = 0+0+1 = 1$ *e* $w_7 = d_2 + d_3 + d_4 = 1+0+1 = 0$. *Comparando com* $v_5 = 1, v_6 = 1$ *e* $v_7 = 0$ *fica fácil ver que* $v_5 \neq w_5$, *logo, o erro está na quinta posição, como já era de se esperar. Para corrigir o erro, basta modificar o símbolo da quinta posição, trocando o 1 por 0 e, para decodificar o símbolo, basta tomar as 4 primeiras posições.*

Com os Algoritmos 4 e 5, podemos codificar, decodificar e corrigir um único erro de qualquer um dos 16 símbolos do código $C(7,4)$. Entretanto, existe uma maneira mais prática de se codificar, decodificar e corrigir um único erro nesse código, mais ainda, tal maneira é facilmente generalizada para a família de códigos $C(2^k - 1, 2^k - k - 1)$.

Considerando a Tabela 4.6 acima, observamos que os coeficientes de $d_1, \cdots, v_7$ presentes em suas entradas são os mesmos da matriz:

$$G_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

a qual é chamada de *matriz geradora* do código $C(7,4)$, visto que qualquer símbolo X desse código é codificado no símbolo X' através da sua multiplicação com a matriz geradora, ou seja, $X' = XG_3$. O índice 3 designa o número de dígitos de verificação do código que, como já vimos anteriormente, nesse caso são 3.

Exemplo 8. *Para codificar o símbolo $x = 0101$ novamente, basta realizar a multiplicação em $\mathbb{Z}_2$ das matrizes*

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

e

$$G_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

obtendo a matriz:

$$XG_3 = X' = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

a qual representa o símbolo codificado $x' = 0101010$. Note que obtemos o mesmo símbolo codificado anteriormente.

Para o caso geral da matriz geradora de um código $C(2^k - 1, 2^k - k - 1)$, nós temos a seguinte definição:

Definição 7. *A matriz geradora, denotada G_k, é uma matriz de dimensão $(2^k - k - 1) \times (2^k - 1)$ com coeficientes em $\mathbb{Z}_2$, tal que todos os elementos codificados do código C sejam obtidos através da sua multiplicação pela matriz geradora.*

Outra matriz que possui destaque no código de Hamming é a chamada *matriz de controle* ou *matriz de paridade* H_k. Para o código $C(7,4)$, temos que:

$$H_3 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

e para o caso geral, temos que a matriz de paridade de um código $C(2^k - 1, 2^k - k - 1)$ é definida como segue:[6]

Definição 8. *A matriz de paridade, denotada por H_k, é uma matriz de dimensão $k \times (2^k - 1)$ com coeficientes em $\mathbb{Z}_2$, tal que $G_k H_k^t = \mathbf{0}$, na qual H_k^t denota a matriz transposta de H_k e $\mathbf{0}$, a matriz nula.*

Note que as matrizes G_3 e H_3 do código $C(7,4)$ podem ser escritas como $G_3 = [I_4 \ \ A]$ e $H_3 = [B \ \ I_3]$, em que A e B são matrizes satisfazendo $A^t = B$, e I_3 e I_4 denotam as matrizes identidade de dimensão 3 e 4, respectivamente. Além disso, quando as matrizes G_3 e H_3 estão escritas nessa forma, dizemos que as mesmas estão em sua *forma padrão*. A generalização desse fato é o conteúdo do próximo teorema, o qual nos permitirá obter G_k em sua forma padrão, sempre que definirmos H_k também em sua forma padrão e vice versa. A demonstração desse teorema não será inserida aqui, pois se utiliza de conceitos que fogem do escopo deste trabalho, porém ela pode ser encontrada em Meneghesso (2012, p. 19-20).

Teorema 2. *Sejam $G_k = [I_{2^k-k-1} \ \ A]$ e $H_k = [B \ \ I_k]$. H_k será a matriz de verificação de paridade associada à matriz geradora G_k se, e somente se, $A^t = B$. Além disso, o código binário correspondente $C(2^k - 1, 2^k - k - 1)$ será corretor de um único erro se, e somente se, as colunas de H_k forem não nulas e distintas.*

Em vista do Teorema 2, uma pergunta que pode surgir é: As matrizes G_k e H_k são as únicas que definem um código da forma $C(2^k - 1, 2^k - k - 1)$? A

[6] O leitor com conhecimentos de Álgebra Linear deve ter percebido que as linhas da matriz geradora formam uma base para um espaço vetorial que é isomorfo a $\mathbb{Z}_2^{2^k-k-1}$. Mais ainda, nota-se também que as colunas da transposta da matriz de paridade formam uma base para o complemento ortogonal do espaço vetorial gerado pelas linhas da matriz geradora.

resposta para essa pergunta é negativa, além disso, a seguinte definição nos dá as condições para a criação de outras matrizes geradoras e de paridade, chamadas de *matrizes equivalentes*.

Definição 9. *Duas matrizes G_k e G'_k geram o mesmo código C, ou seja, são equivalentes, se uma pode ser obtida da outra através de uma sequência finita de operações do tipo:*

L1 Permutação de duas linhas;

L2 Adição de uma linha a outra;

C1 Permutação de duas colunas.

Exemplo 9. *É fácil ver que a matriz geradora do código de Hamming definido na subseção 2.4, através do Algoritmo 2 é igual a:*

$$G_3 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

A qual, por sua vez é equivalente à matriz geradora do código de Hamming que definimos nesta seção à partir da Tabela 4.6:

$$G'_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Pois podemos obter uma da outra através de aplicações sucessivas das operações acima definidas. Faça Isso!

Para o caso geral, temos a seguinte proposição, cuja demonstração segue da simples aplicação das operações $(L1), (L2)$ e $(C1)$. Para uma demonstração dessa proposição para o caso em que os coeficientes das

matrizes são elementos de um corpo K qualquer, ver Hefez (2008, p. 92-93).

Proposição 2. *Dado um código C com matriz geradora G_k, existe um código equivalente C' com matriz geradora G'_k na forma padrão.*

Isto posto, temos definida uma matriz geradora G_k. A codificação de um símbolo u qualquer do código $C(2^k-1, 2^k-k-1)$ se dá simplesmente pela multiplicação de u por G_k, obtendo-se o símbolo codificado $v = uG_k$, ou seja, a codificação é simplesmente uma multiplicação de matrizes com coeficientes em $\mathbb{Z}_2$. Para a decodificação e correção há dois casos: i) o símbolo foi transmitido sem erro; e ii) o símbolo foi transmitido com um único erro.

Para o primeiro caso, se o símbolo codificado v foi transmitido sem erro, então o mesmo é anulado pela matriz de paridade. Com efeito, $vH_k^t = (uG_k)H_k^t = u(G_kH_k^t) = u\mathbf{0} = \mathbf{0}$, assim, sempre que o produto vH_k^t for igual à matriz nula, podemos concluir que o símbolo foi transmitido sem erro.

Para o segundo caso, sejam v um símbolo do código $C(2^k-1,2^k-k-1)$ (sem erro) e $v^{(i)} \in Z_2^{2^k-1}$ o símbolo obtido pela adição, em $\mathbb{Z}_2$, de 1 ao i-ésimo digito de v. Logo, $v^{(i)}$ é um símbolo codificado transmitido com um erro no i-ésimo dígito. Assim podemos escrever $v^{(i)} = v + (0 \cdots 0 \underbrace{1}_{i-simo} dgito\ 0 \cdots 0)$, a partir do que, temos:

$$v^{(i)}H_k^t = \underbrace{vH_k^t}_{0} + \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} H_k^t$$

$$= \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} H_k^t.$$

Note que $v^{(i)}H_k^t$ é a i-ésima linha de H_k^t, logo, a i-ésima coluna de H_k. Dessa forma, um erro ocorrido na i-ésima posição da mensagem transmitida equivale a i-ésima coluna de H_k. Para corrigir o erro, portanto, temos que modificar o dígito do símbolo recebido na posição que é equivalente a i-ésima coluna da matriz de paridade. A seguir, ilustramos como esse

procedimento funciona na correção de um símbolo do código $C(7,4)$, o qual é o código para $k=3$ no código $C(2^k-1, 2^k-k-1)$:

Exemplo 10. *Consideremos novamente o símbolo $x = 0101$, que como pôde ser visto no Exemplo 8, ao ser codificado, se torna $x' = 0101010$. Assim, como fizemos no Exemplo 7, introduziremos um erro na quinta posição, obtendo $x'' = 0101110$. Para corrigir esse erro, multiplicaremos o símbolo x'' pela transposta da matriz de paridade para este código H_3^t, obtendo:*

$$X''H_3^t = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix},$$

a quinta linha de H_3^t e, consequentemente, a quinta coluna de H_3, logo, o erro se encontra na quinta posição do símbolo x'', como já esperávamos.

Antes de irmos para o próximo exemplo, note que no código $C(15,11)$, os símbolos são escritos na forma $d_1d_2d_3\cdots d_{11}v_{12}v_{13}v_{14}v_{15}$, e que, estendendo o Algoritmo 4 para esse caso, podemos definir:

$$\begin{aligned} v_{12} &= d_1+d_2+d_4+d_5+d_7+d_9+d_{11} \\ v_{13} &= d_1+d_3+d_4+d_6+d_7+d_{10}+d_{11} \\ v_{14} &= d_2+d_3+d_4+d_8+d_9+d_{10}+d_{11} \\ v_{15} &= d_5+d_6+d_7+d_8+d_9+d_{10}+d_{11} \end{aligned} \tag{4.2}$$

Essa maneira de definir v_{12}, v_{13}, v_{14} e v_{15} é equivalente à definição de v_1, v_2, v_4 e v_8 dada pelo Algoritmo 2 na subseção 2.4, e a relação entre os dígitos lá e aqui pode ser vista na Tabela 4.7 abaixo. Desta forma, para construir a matriz geradora $G_4 = [I_4 \ \ A]$, basta tomar $A = [v_{12}v_{13}v_{14}v_{15}]$, na qual cada uma das quatro colunas de A é formada pelos coeficientes dos

dígitos $d_i, i = 1 \cdots 11$, que definem v_{12}, v_{13}, v_{14} e v_{15}.

Tabela 4.7: Equivalência entre os dígitos v_{12}, v_{13}, v_{14} e v_{15} e v_1, v_2, v_4 e v_8

Subseção 2.4	v_1	v_2	d_3	v_4	d_5	d_6	d_7	v_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}
Aqui	v_{12}	v_{13}	d_1	v_{14}	d_2	d_3	d_4	v_{15}	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}

Fonte: Elaborada pelo autor

Com esse exemplo, encerramos as considerações sobre o Código de Hamming e concluímos esta seção. Para mais exemplos e outras considerações sobre esse código, como, por exemplo, sua interpretação geométrica, ver Lira (2018a).

4.3 Considerações finais

O código de Hamming definitivamente representou um marco para a Teoria dos Códigos Corretores de Erros. Os desenvolvimentos que se seguiram, em certa medida, só foram possíveis graças ao trabalho desse pioneiro. Sendo assim, a abordagem de seu código na Educação Básica é mais do que merecida, bem como representa uma excelente forma de introduzir o assunto para as novas gerações de futuros engenheiros, matemáticos e pesquisadores das mais diversas áreas da ciência.

Na dissertação que gerou este trabalho, Lira (2018a), nós desenvolvemos uma sequência didática para o ensino do código de Hamming em suas duas formulações, tal como foram apresentadas anteriormente. Devido a natureza deste trabalho, optamos por não a apresentar aqui, porém deixarmos a sugestão de leitura, e, porque não, de aplicação da mesma nas escolas, para o leitor disposto a tal.

4.4 Referências bibliográficas

ABRANTES, S. A. **Notas históricas da codificação para controle de erros**. São Paulo, 09 de jul. de 2021. Disponível em: https://docplayer.com.br/amp17613484-Notas-historicas-da-codificacao-para-controlo-de-erros.html. Acesso em 09 de jul. de 2021.

AGUIAR, J.; VIEIRA, S.; CAVALCANTE, R. Códigos quânticos corretores de erros. In: V Congresso Norte-Nordeste de Pesquisa e Inovação, 2010, Maceió. **Anais**... Maceió: IFAC, 2011.

ALVES, B. C. **Uma proposta de oficina sobre códigos para a contextualização do estudo de aritmética e matrizes no ensino médio**. 2015. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Universidade Federal de Goiás, Goiânia.

BOULLAF, F. **Códigos, reticulados e aplicações em criptografia**. 2015. Dissertação (Mestrado) – Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Campinas.

CARVALHO, S. **Matrizes, determinantes e polinômios: aplicações em códigos em corretores de erros, como estratégias motivacional para o ensino de matemática**. 2014. Dissertação (Mestrado) - Programa de Pós-Graduação e Mestrado Profissional em Matemática em Rede Nacional PROFMAT, Universidade Federal de Rondônia-UNIR, Porto Velho.

DIAS, J. **O código da mariner 9**. 2017. Dissertação (Mestrado) - Programa de Pós-Graduação e Mestrado Profissional em Matemática em Rede Nacional PROFMAT, Universidade Federal de São João del-Rei, São João del-Rei.

Ellenberg, J. **O poder do pensamento matemático**: a ciência de como não estar errado. 1 ed. Rio de Janeiro: Zahar, 2015.

FARIA, L. **Existências de códigos corretores de erros e protocolos de comunicação em sequências de DNA**. 2011. Doutorado em engenharia elétrica, Universidade Estadual de Campinas, Campinas.

GLEICK, J. **A informação**: Uma história, uma teoria, uma enxurrada. São Paulo: Companhia das Letras, 2013.

GUIMARÃES, W. Códigos corretores de erros para gravação magnética.

2003. Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Pernambuco, Recife.

HAMMING, R. Error detecting and error correcting codes. **Bell System Technical Journal**, Nova York, v. 29, n. 2, p. 147–160, 1950.

HEFEZ, A. **Aritmética**. Rio de Janeiro: SBM, 2014.

HEFEZ, A.; VILLELA, M. Códigos Corretores de Erros. Rio de Janeiro: IMPA, 2008.

LIRA, E. **Códigos corretores de erros no ensino médio**: Um estudo sobre o Código de Hamming. 2018. Dissertação (Mestrado) – Universidade Federal Rural de Pernambuco, Mestrado Profissional em Matemática, Recife.

LIRA, E.; DANTAS, M. **Uma sequência didática para o ensino de códigos corretores de erros no ensino médio**. In: III Encontro de Educação Matemática do Vale do São Francisco, 2018, Petrolina.

MACHADO, D. **Uma abordagem de dígitos verificadores e códigos corretores no ensino fundamental**. 2016. Dissertação (Mestrado) - Mestrado Profissional em Matemática em Rede Nacional, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo.

MENEGHESSO, C. **Códigos corretores de erros**. 2012. Dissertação (Graduação - Trabalho de Conclusão de Curso). Departamento de Matemática, Centro de Ciências Exatas e Tecnologia, Universidade Federal de São Carlos, São Carlos.

MILIES, C. A matemática dos códigos de barras – Detectando erros. **RPM**, Rio de Janeiro, v. 68, 2008.

MILIES, C. Breve introdução à Teoria dos Códigos Corretores de Erros. In: Sociedade Brasileira de Matemática, Colóquio de Matemática da Região Centro-Oeste, 2009, Campo Grande. **Anais**... Campo Grande: Departamento de Matemática Universidade Federal do Mato Grosso do Sul, 2009.

MIRANDA, D. **Códigos corretores de erros e empacotamentos de discos**. 2013. Trabalho de Conclusão (Mestrado Profissional) - Departamento de Matemática da Universidade Federal Rural de Pernambuco, Recife.

NICOLETTI, E. **Aplicações de álgebra linear aos códigos corretos de erros e ao ensino médio**. 2015. Dissertação (Mestrado) - Universidade Estadual Paulista, Instituto de Geociências e Ciências Exatas, Rio Claro.

PINZ, C. **Dígitos verificadores e detecção de erros**. 2013. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Instituição de Matemática, Estatística e Física, Universidade Federal do Rio Grande, Rio Grande.

ROCHA, A.. textbfModelo de sistema de comunicações digital para o mecanismo de importação de proteínas mitocondriais através de códigos corretores de erros. 2010. Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas.

RODRIGUES, N. **Códigos Corretores de Erros**. 2017. Dissertação (mestrado profissional) - Universidade Federal de Santa Catarina, Centro de Ciências Físicas e Matemáticas, Programa de Pós-Graduação em Matemática, Florianópolis.

ROUSSEAU, C.; AUBIN, Y. **Matemática e atualidade**. Volume 1. Rio de Janeiro: SBM, 2015. SCHROEDER, E. **Códigos binários e truques de mágica**. 2017. Dissertação (Mestrado) - Universidade Estadual do Mato Grosso do Sul, Curso de Mestrado Profissional de Matemática em Rede Nacional, Dourados.

SHANNON, C. A mathematical theory of communication.**Bell System Technical Journal**, Nova York, v. 27, p. 379-423. 1948.

SHINE, C. **21 Aulas de Matemática Olímpica**. Rio de Janeiro: SBM, 2009.

STEWART, I. **17 Equações que mudaram o mundo**. Rio de Janeiro: Zahar, 2013.

SÁ, C; ROCHA, J. **Treze Viagens pelo Mundo da Matemática**. 2 ed. Rio de Janeiro: SBM, 2012.