UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CÓDIGOS CORRETORES DE ERROS PARA GRAVAÇÃO MAGNÉTICA

Por

WALTER PRADO DE SOUZA GUIMARÃES

RECIFE/PE
2003

WALTER PRADO DE SOUZA GUIMARÃES

CÓDIGOS CORRETORES DE ERROS PARA GRAVAÇÃO MAGNÉTICA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco em cumprimento às exigências para obtenção do título de

Mestre em Engenharia Elétrica

Valdemar Cardoso da Rocha Júnior
Orientador

CÓDIGOS CORRETORES DE ERROS PARA GRAVAÇÃO MAGNÉTICA

A Deus, toda honra e toda glória,
aos meus pais, Antônio e Neidson
à minha esposa, Elizabeth
e ao meu filho, Felipe Augusto.

AGRADECIMENTOS

Sou profundamente agradecido ao Prof. Valdemar Cardoso da Rocha Jr. pelo apoio, pela orientação e pelo incentivo à realização deste trabalho, que é fruto de uma parceria entre duas Instituições de Ensino, UTAM e UFPE. Este trabalho só se tornou possível de realizar, graças à sua ação encorajadora e decisiva para implantação do Curso de Pós-graduação em Engenharia Elétrica da UFPE, totalmente realizado em Manaus. Os seus conselhos e orientações foram importantes em diversos momentos, em que estive em meio a dúvidas e problemas inerentes a esta tarefa.

Agradeço também à UTAM, Instituição da qual me sinto honrado de ser professor efetivo do quadro desde o ano de 1991, que também tornou possível a realização deste Curso.

Agradeço a todos os professores do Curso de Pós-graduação em Engenharia Elétrica da UFPE pelo esforço de repassar informações importantes e que, somadas as obtidas durante a realização desta dissertação, permitiram chegar ao final.

Agradeço aos outros colegas do Curso de Pós-Graduação, em especial aos colegas professores da UTAM, pelo incentivo e pelo apoio nos momentos mais difíceis, em especial, durante a realização deste trabalho.

Agradeço à minha família por estar sempre ao meu lado, dando-me carinho e energia nos momentos mais angustiantes.

Por último, agradeço a Deus por me conceder a graça de poder ter alcançado um degrau mais alto na minha carreira de docente de ensino superior.

WALTER PRADO DE SOUZA GUIMARÃES

RESUMO

À medida que a necessidade por informação, juntamente com novas ferramentas computacionais, aumenta com o passar dos anos, o armazenamento de dados se torna mais e mais importante. O processo de gravação magnética usando códigos corretores de erros vem se constituindo em uma boa solução para fazer com que sistemas se tornem bastante confiáveis e eficientes. O foco desta dissertação está em como se obter esta eficiência e confiabilidade em sistemas de gravação magnética usando códigos corretores de erros.

Esta dissertação está dividida em duas partes. A primeira trata da preservação dos dados contra problemas típicos em gravação magnética. A confiabilidade é obtida pelo uso de um eficiente código matricial com entrelaçamento simples, ou seja, uma dimensão entrelaçada (coluna) e um código corretor de erros aleatórios ou um código corretor de surtos de erros sendo usado para codificação das linhas do referido código. Este método é proposto para melhorar significativamente o desempenho na correção de manchas de erros por meio do uso apropriado de redundância de dados. Este método incorpora simplicidade e eficiência, conseqüentemente resultando em sistemas mais simples e robustos.

A segunda parte trata de criptossistemas de chave secreta usando códigos corretores de erros. É realizada uma revisão teórica do assunto juntamente com a exposição de uma nova proposta de uso de códigos matriciais com entrelaçamento simples. Este sistema simplificado é uma adaptação de criptossistemas conhecidos e tem por objetivo possibilitar a sistemas de gravação magnética a capacidade de proteção dos dados contra “ataques inimigos”.

ABSTRACT

As the need for data explodes with the passage of time as well as the demand for more computing power, data storage becomes more and more important. Magnetic recording using error correcting codes has turned out to be a good solution to make highly reliable and efficient systems. The focus of this dissertation is on how to achieve data reliability and efficiency using Error Correcting Codes for Magnetic Recording.

This dissertation consists of two parts. The first one deals with data preservation against typical magnetic recording issues. Reliability is achieved by using efficient Array Codes with simple interleaving, i.e., one dimension interleaved (the column) and random error correcting codes or burst-error correcting codes used in codeword lines. This method is proposed to improve the performance of patch error correction significantly through the proper use of data redundancy. This method gathers simplicity and efficiency, thus resulting in robust systems.

The second part deals with private key cryptosystems using error correcting codes. This subject is going to have its theory revisited along with the explanation about a new idea of using Simple Interleaving Array Codes. This comprehensive system is an adaptation of known cryptosystems and it aims to allow magnetic storage data protection against “enemy attacks”.

Introdução

Motivação

Este é o tempo da Informação. A informação é gerada, processada, transmitida e guardada em várias formas: texto, imagem, vídeo e em multimídia. Como a necessidade por dados aumenta exponencialmente com o tempo e aumenta a capacidade de computação, a guarda ou transmissão de dados se torna mais e mais importante. Da computação científica à transações de negócios, os dados são a parte mais importante. Como guardá-los ou transmiti-los de modo confiável e eficiente é uma preocupação essencial, que é o foco desta dissertação.

Armazenar dados em mídia magnética tem sido uma das preocupações de pequenas a grandes corporações, preocupadas e ansiosas por manter dados preservados contra ações do tempo e do ambiente. Já há algum tempo as empresas vem investindo em técnicas de gravação de dados na forma magnética, para tornar este processo o mais confiável possível. Uma das técnicas, que será alvo deste trabalho, consiste na inserção de códigos corretores de erros no processo de gravação dos dados, a fim de que possam ser recuperados mesmo diante de problemas como falhas de gravação, por parte do dispositivo de gravação magnética (cabeça de gravação), ou, até mesmo, relacionados com a durabilidade das fitas magnéticas utilizadas neste processo.

A preocupação desta dissertação é mostrar, de uma maneira simplificada, como uma técnica de correção de erros pode ser empregada para eliminar e/ou reduzir erros de gravação magnética.

Além disso existe uma vertente na área de comunicação que trata de assegurar que os dados gravados e/ou transmitidos estejam seguros contra ataques de "inimigos", empregando técnicas de Criptografia. Particularmente, o emprego de códigos corretores de erros para cifragem de dados, será alvo deste trabalho, sempre com o intuito de utilizar estas técnicas para gravação magnética.

Códigos Corretores de Erros para Gravação Magnética

A informação digital sobre a fita magnética é geralmente gravada em alguns canais paralelos à direção de movimento da fita. Uma simples cabeça de leitura-escrita é usualmente associada

a cada canal, e algumas cabeças lêem ou escrevem simultaneamente em intervalos discretos sobre os canais. A informação é dividida em blocos, similar a matrizes bidimensionais, graças aos intervalos (*gaps*) introduzidos em todos os canais. Estes blocos ou matrizes retangulares são formados por colunas, representadas pelos canais e por linhas que correspondem aos dígitos que passam simultaneamente sob as cabeças de leitura-escrita.

Sistemas de gravação de fita magnética sofrem de erros causados pelo mecanismo, similar àqueles encontrados em sistemas de discos ópticos e magnéticos (ou seja, defeitos de material e contaminação de superfície). Adicionalmente, a flexibilidade da fita magnética pode gerar o seu próprio conjunto de problemas (ex.: inserção de dígitos devido a estiramento da fita magnética) e também a falha do circuito de leitura-escrita, associado com algum canal, pode resultar em surtos de erros na coluna correspondente .

Em muitos sistemas digitais, como foi mencionado, os dados na fita magnética são organizados em trilhas paralelas, que ocupam o seu comprimento. Por exemplo, o sistema de gravação magnética IBM 3420 [32] usa 9 trilhas de dados sobre uma fita de ½ polegada. Dada a organização e a direção com que a fita se move, contaminantes (sujeiras) tendem a corromper uma trilha ou um pequeno número de trilhas adjacentes durante o processo de leitura. Em quase todos os sistemas de controle de erro para gravação magnética, as palavras-código são definidas verticalmente (ao longo das trilhas), fornecendo uma forma de entrelaçamento. A perda de uma ou duas trilhas então requer a correção de somente um ou dois símbolos por palavra-código.

É usual o emprego de códigos corretores de erros matriciais devido à forma como os dados são gravados sobre a fita e também devido à sua simplicidade de codificação e decodificação.

Além disso é comum o emprego, como mencionado, de entrelaçamento, que permite ao código corretor de erros a capacidade de correção de erros em um formato bidimensional, conhecido por mancha de erros (*patch* ou *cluster* de erros). No sistema IBM 3420, o processo de gravação de dados já incorpora um certo grau de entrelaçamento e evita, conseqüentemente que um surto de erros, até um dado comprimento, possa afetar os dados lidos.

Principais Contribuições da Dissertação

É introduzida uma forma de entrelaçamento unidimensional em um código matricial em vez do emprego de entrelaçamento bidimensional de blocos, que permite a correção não só de surtos de erros como também de manchas de erros bidimensionais. O sistema proposto se mostra bastante simples e eficiente no combate a manchas de erros do tipo compacto.

Além do uso de códigos matriciais entrelaçados na gravação de dados, será também dado enfoque à utilização de criptografia para salvaguardar os dados gravados contra ação de "inimigos". Neste criptossistema proposto nesta dissertação se dá ênfase à utilização de cifras de chave secreta baseadas em códigos matriciais corretores de erros.

Organização

Esta dissertação encontra-se dividida em seis capítulos e quatro apêndices, cujas breves descrições encontram-se a seguir.

Capítulo 1 Este capítulo traz uma revisão da teoria de códigos matriciais (*array codes*) com ênfase em códigos para a correção de manchas de erros.

Capítulo 2 Este capítulo traz uma revisão da teoria de entrelaçamento com ênfase em entrelaçadores de blocos e sua contribuição para a correção de surtos de erros bidimensionais.

Capítulo 3 Neste capítulo é estudada a técnica de implementação de códigos matriciais com entrelaçamento unidimensional e com linhas codificadas por códigos corretores de erros aleatórios. Esta técnica mostra-se eficiente na correção de manchas de erros compactas, que serão definidas com base na distância de Lee. Este tipo de mancha possui um papel importante na concepção deste trabalho. Serão apresentados teoremas e definições sobre os principais parâmetros, a fim de garantir o perfeito entrelaçamento.

Capítulo 4 Neste capítulo é estudada a técnica de implementação de códigos matriciais com entrelaçamento unidimensional e com linhas codificadas por códigos corretores de surtos de erros.. Esta técnica diferencia-se daquela apresentada no capítulo 3, pelo fato de usar uma matriz de transformação linear própria e ter como resultado um entrelaçamento unidimensional que garante uma capacidade maior de correção de erros, visto que os códigos corretores de surtos de erros, neste caso, são mais eficazes do que os códigos corretores de erros aleatórios. Além disso, as definições e os teoremas serão ligeiramente modificados a fim de garantir o perfeito entrelaçamento.

Capítulo 5 Neste capítulo é estudada a técnica de criptografia de chave secreta usando códigos corretores de erros matriciais (*array codes*) com entrelaçamento unidimensional. Também é realizado um breve histórico sobre a evolução da técnica e sobre os principais ataques de "inimigos", com especial atenção ao ataque de Al Jabri

[49,50]. A técnica de permutação adaptativa [48] e sua contribuição para proteger o criptosistema contra o ataque de Al Jabri é discutida e também faz parte da proposta deste trabalho.

Capítulo 6 Neste capítulo são mostrados exemplos de como esta técnica de códigos matriciais com entrelaçamento unidimensional pode ser implementada na prática.

Capítulo 7 Neste capítulo faz-se um resumo dos avanços obtidos e são apontadas sugestões para direcionamento de pesquisas posteriores.

Apêndice I Compreende a revisão da teoria de códigos corretores de erros aleatórios dos tipos BCH e Reed Solomon, que podem ser utilizados na codificação das linhas da palavra-código do código matricial, por serem códigos de máxima distância (MDS).

Apêndice II É exibida uma tabela contendo diversos valores do parâmetro β^1 da matriz de transformação linear, em função do número de linhas, n_1, do código matricial. Esta tabela é fruto de um programa de computador, desenvolvido com base nas definições e teoremas propostos no capítulo 3.

Apêndice III Neste apêndice é apresentada uma breve revisão sobre a teoria de códigos corretores de erros.

Apêndice IV Neste apêndice é apresentada uma lista dos símbolos empregados no texto.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Capítulo 1

Códigos Matriciais

1.1 Introdução

Estruturas de códigos corretores de erros em duas dimensões ocorrem freqüentemente em sistemas automáticos de computação. Algumas vezes as duas dimensões são físicas, como em gravação magnética. Na maior parte das vezes as duas dimensões são conceituais, como em matrizes numéricas, ou quando uma estrutura em duas dimensões for imposta por um programador. Par-a propósitos de correção e detecção de erros, a redundância pode ser incorporada a uma estrutura matricial (*array*) de dígitos binários, pela imposição de certas restrições sobre grupos específicos de dígitos.

Segundo Farrell [1], códigos corretores de erros na forma matricial (*array codes*) são códigos lineares que podem ser de dois tipos: de bloco ou convolucionais. Basicamente podem ser construídos a partir de algum código de paridade simples (*single parity check* ou SPC) ou de outros tipos de códigos, montados em duas ou mais direções ou dimensões geométricas, sendo dada ênfase a códigos simples, de baixa complexidade, nos métodos de codificação e decodificação. Os códigos matriciais, portanto, possuem como características principais de projeto a simplicidade e a flexibilidade. Além disso, por serem relativamente simples, possuem decodificação rápida. Por estas razões, os códigos matriciais têm sido usados em um grande número de aplicações para as quais estas propriedades são altamente desejáveis, particularmente no caso em que os símbolos de informação aparecem em padrões geométricos (e.g., fitas, cartões, discos, circuitos integrados, etc). Os dois principais campos de aplicação são em sistemas de comunicação e em sistemas de armazenamento de informação.

Como este trabalho de dissertação aborda técnicas de correção de erros em gravação magnética, é mister que se faça um resumo sobre códigos matriciais, visto que o processo físico de gravação gera uma estrutura que lembra uma matriz de dados ou arranjo

bidimensional. Sobre esta matriz será introduzida redundância para correção de erros, formando assim o que se designa de código matricial. A seguir será feito um resumo sobre as principais técnicas de construção de códigos matriciais.

1.2 Revisão da Teoria de Códigos Matriciais

Basicamente existem duas modalidades de códigos matriciais: códigos matriciais de blocos e códigos matriciais convolucionais. Em face desta dissertação ser desenvolvida sobre códigos matriciais de blocos será dada ênfase a este assunto e, em particular, a códigos matriciais para correção de surtos de erros bidimensionais ou em células, também chamados de manchas (*clusters* ou *patches* de erros).

O código matricial pode ser usado para correção de erros aleatórios, para correção de erros em surto em uma dimensão e também para a correção de erros em duas dimensões, chamados de *patches* ou *clusters*.

1.2.1 Códigos matriciais de blocos para correção de erros aleatórios

1.2.1.1 Código corretor de erros unidimensional com dígito de paridade único (SPC-S*ingle Parity Check*)

Este tipo de código corretor de erros, como ilustrado na Figura 1.1, forma um código $(n, n\text{-}1, 2) = (k\text{+}1, k, 2)$. A paridade pode ser par ou ímpar; a paridade par faz com que o código seja linear e é mais conveniente matematicamente, mas a paridade ímpar algumas vezes é usada para melhorar a sincronização e as características espectrais do código. Na Figura 1.1, o dígito de paridade é colocado ao final do bloco, para que a palavra-código tenha uma forma sistemática. Contudo o dígito de paridade pode ser colocado em qualquer posição conveniente na palavra; o único requisito é que a palavra tenha peso par. Este tipo de código pode detectar qualquer número ímpar de erros na palavra-código, ou pode corrigir um simples apagamento.

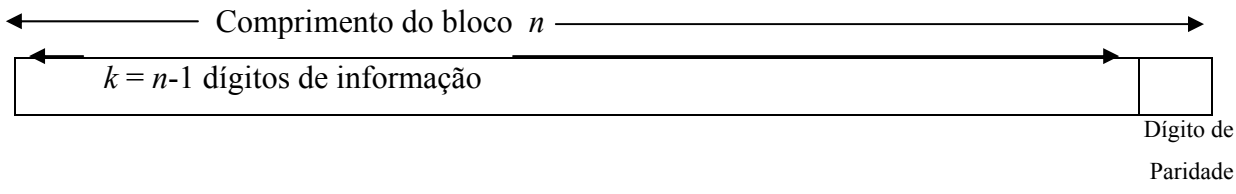


Figura 1.1: Código corretor de erros unidimensional com único dígito de paridade (SPC).

A seguir são mostradas, como exemplo, matrizes geradoras de códigos SPC com palavras dispostas nas direções horizontal e vertical, respectivamente.

$$G_H = \begin{bmatrix} 1001 \\ 0101 \\ 0011 \end{bmatrix}$$

Esta matriz tem como linhas (direção horizontal) as palavras de um código SPC com três dígitos de informação. O comprimento do bloco é igual a quatro.

Para um código SPC com palavras na direção vertical, com comprimento de bloco igual a quatro, idêntico ao do caso anterior, tem-se a seguinte matriz geradora:

$$G_V = \begin{bmatrix} 100 \\ 010 \\ 001 \\ 111 \end{bmatrix}$$

1.2.1.2 Código corretor de erros bidimensional com dígitos de paridade nas linhas e nas colunas (RAC - *Row and Column Code*)

Este tipo de código matricial é exibido na Figura 1.2, o qual consiste essencialmente de um código matricial com paridade simples nas linhas e nas colunas, também chamado de dupla-coordenada, bidirecional, ou código de palavra cruzada. Este código matricial pode ter formato quadrado ou retangular, e tem parâmetros $(n_1n_2,(n_1\text{-}1)(n_2\text{-}1),4)$, ou seja, $((k_1+1)(k_2+1), k_1k_2, 4)$. Conseqüentemente os parâmetros deste código matricial são gerados dos produtos dos correspondentes parâmetros das componentes dos códigos SPC e sua matriz geradora é o produto Kroenecker [22] das matrizes geradoras de códigos SPC. Em particular, a distância mínima de Hamming deste tipo de código matricial é d=2x2=4, o que explica a sua capacidade de corrigir um erro (as correspondentes paridades de linha e coluna falham) e detectar duplo erro, ou até três erros (pelo menos um dígito de paridade de linha e um dígito de paridade de coluna falham). Este código matricial pode detectar todos os outros padrões de

erro, exceto aqueles que são palavras-código (i.e, que têm peso par em ambas as dimensões). Alternativamente, pode corrigir até três apagamentos aleatórios.

Para um dado valor de $n=n_1n_2$, a taxa $R = k/n = k_1k_2/n$ é maximizada quando $k_1=k_2 = k_0$ e $n_1 = n_2 = n_0$ (*código matricial* com formato quadrado) [54]. A remoção do dígito de paridade sobre os dígitos de paridade (*check-on-checks*) reduz a distância do código de quatro para três. Neste caso, a correção de um erro ainda é possível, essencialmente porque os dígitos de paridade de linha e coluna formam um conjunto de duas equações de verificação de paridade sobre cada dígito de informação do código matricial. Estas duas equações de paridade são ortogonais [51] e permitem a correção de um erro por meio da técnica de decodificação por lógica de maioria (ambos os dígitos de paridade associados a um dado dígito de informação devem falhar se este dígito falhar). Usando a decodificação por lógica de maioria, J testes de equações de paridade ortogonais permitem a correção de J/2 dígitos errados.

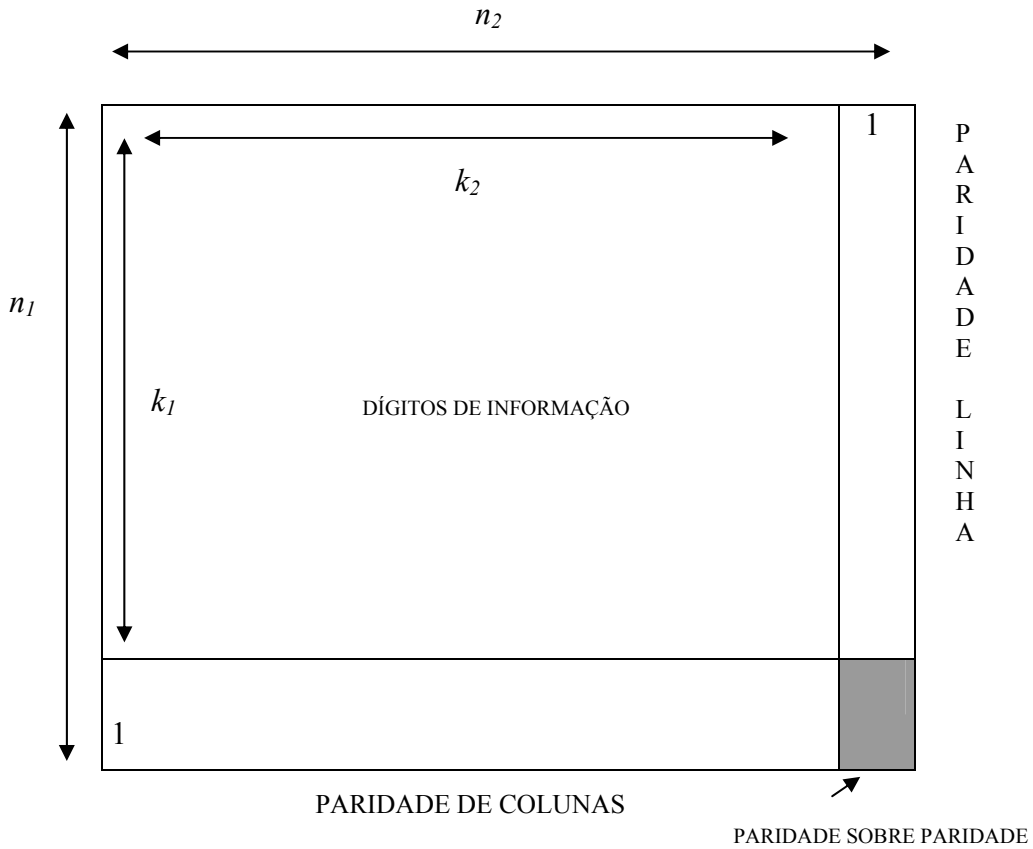


Figura 1.2: Código corretor de erros bidimensional com dígitos de paridade nas linhas e nas colunas (RAC –*Row and Column Code*).

Exemplos:

Seja a seguinte palavra-código do código matricial RAC (25,16,4):

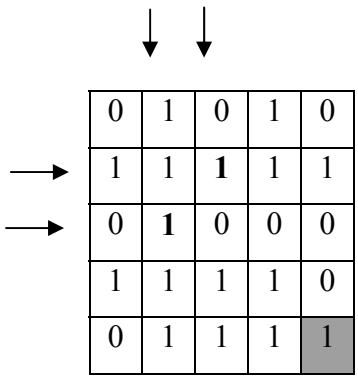
0	1	0	1	0
1	1	0	1	1
0	0	0	0	0
1	1	1	1	0
0	1	1	1	1

a) Demonstração de correção de um único erro

0	1	0	1	0
1	1	0	1	1
0	**1**	0	0	0
1	1	1	1	0
0	1	1	1	1

Neste caso as duas equações de paridade, de linha e coluna, que verificam o dígito errado, falham e por isto torna-se possível localizá-lo e corrigi-lo.

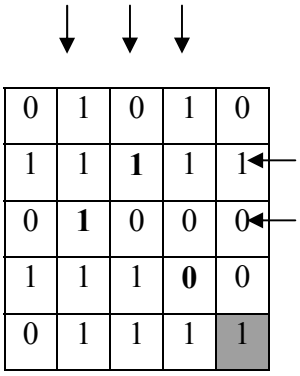
b) Demonstração de detecção de dois erros



0	1	0	1	0
1	1	**1**	1	1
0	**1**	0	0	0
1	1	1	1	0
0	1	1	1	1

Neste caso quatro equações de paridade falham e não é possível a localização dos dois erros com exatidão, devido à ambigüidade da localização das possíveis posições erradas.

c) Demonstração de detecção de três erros



0	1	0	1	0
1	1	**1**	1	1 ←
0	**1**	0	0	0 ←
1	1	1	**0**	0
0	1	1	1	1

Acima de três erros nem sempre é possível identificar a presença de todos eles, para isto basta que um número par de erros se concentre em uma mesma linha ou mesma coluna.

1.2.1.3 Códigos matriciais com dígitos de paridade ortogonais

Este método utiliza outros caminhos ao longo do arranjo para o cálculo de dígitos de paridade como linhas, colunas e diagonais. A Figura 1.3 mostra um código matricial com uma matriz de dígitos 5x5, com seis conjuntos de testes de paridade ortogonal: As paridades das linhas e das colunas, das duas diagonais principais e das duas direções que se orientam pelo " movimento de peças denominadas cavalos em jogo de xadrez". O código é capaz de corrigir até três erros aleatórios, ou seis apagamentos aleatórios e tem os seguintes parâmetros: $n = 25 +10+18 +26 = 79$, $k = 25$, $d=7$. Genericamente, para um código matricial de k_1xk_2 dígitos de informação, há dois conjuntos de $k_1 + k_2 - 1$ diagonais principais, quatro conjuntos de $k_1 + 2(k_2-1)$ no sentido da diagonal formada pelo "movimento de cavalo", quatro conjuntos de $k_1 + 3(k_2-1)$ no sentido do "movimento do cavalo um-três", etc [1]. A redundância deste tipo de código cresce muito mais rapidamente do que a sua correspondente capacidade de correção.

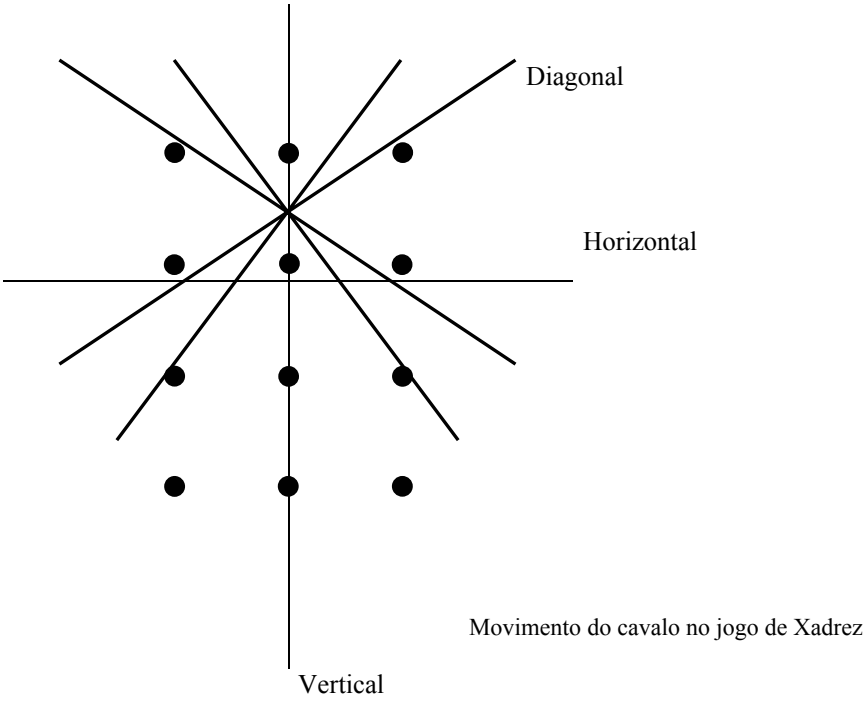
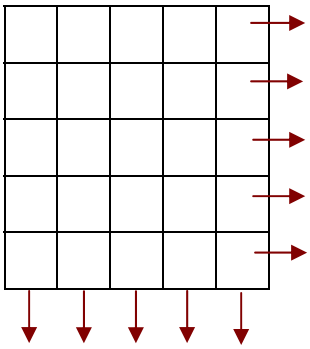


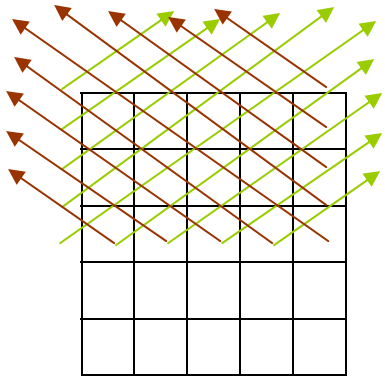
Figura 1.3: Ilustração do método de construção de código matricial 5x5 com dígitos de paridade diagonais.

Exemplo: Para uma matriz 5x5 tem-se:

a) Indicação dos 10 dígitos de paridade Horizontal e Vertical



b) Indicação dos 18 dígitos de paridade nas diagonais principais



c) Indicação do cálculo dos dígitos de paridade nas diagonais ao longo do caminho de uma peça cavalo do jogo de xadrez

13	11	1	2	3
12	6	7	8	9
11	1	2	3	4
6	7	8	9	10
1	2	3	4	5

Observação:

- Nesta ilustração aparece apenas a metade das diagonais usando o caminho do “movimento de cavalo”. O total corresponde a 26 (vinte e seis).
- As 13 (treze) diagonais são indicadas pela numeração (de 1 a 13) e pelas áreas de mesma cor.

1.2.2 Códigos matriciais de bloco para correção de erros em surto

Nos códigos matriciais considerados até agora, para a correção de erros aleatórios, a ordem em que os dígitos de informação e de paridade da matriz codificada são lidos para o canal não é importante, exceto talvez se um código sistemático for requerido. O código com dígitos de

paridade em linha e coluna (RAC), por exemplo, pode ser lido por linha ou por coluna (não-sistemático) ou pelos dígitos de informação (linhas ou colunas) seguidos pelos dígitos de paridade de linha e coluna (sistemático). No caso de controle de erros em surto (*burst*), a ordem da leitura é crucial na determinação das propriedades dos códigos matriciais.

1.2.2.1 Código matricial com único dígito de paridade e com entrelaçamento

Qualquer padrão de surto singelo de erros (*single burst*) de comprimento *b* em um bloco pode ser detectado por meio de códigos SPC com entrelaçamento de ordem *b*. Conseqüentemente um código matricial com dígito de paridade por linha, com parâmetros $(n_1n_2,\ n_1(n_2-1),\ 2)$, mostrado na Figura 1.4, é um código matricial ótimo capaz de detectar padrões com um único surto de erros, ou corrigir um único apagamento, para comprimento de surtos de erros $b \leq n_1$ [1], uma vez que os dígitos do código são lidos por colunas consecutivas, na ordem mostrada na referida Figura.

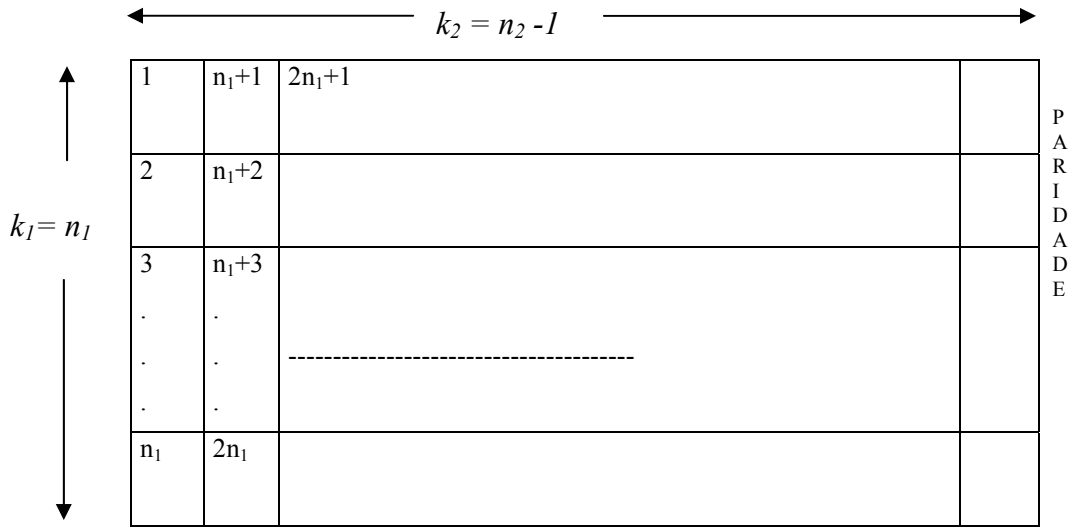


Figura 1.4: Código matricial para detecção de erros em surto.

1.2.2.2 Código matricial para correção de surtos de erros em linhas da matriz (RBC - *Row Burst Correcting Code*)

Por extensão dos conceitos anteriores, a correção de erros em surto requer dois conjuntos ortogonais de dígitos de paridade, ambos ortogonais ao processo de leitura, isto é, ortogonais à direção do surto de erros. A Figura 1.5 mostra um arranjo de um código matricial com paridades vertical e diagonal e leitura horizontal. Este código apesar de corrigir todos os padrões de erros, com um único surto de erros por palavra, de comprimento $b \leq i$ [1], torna-se inconveniente por haver uma linha de paridade com comprimento de $i + j - 1$ dígitos de paridade diagonal, enquanto que as outras linhas possuem comprimento de apenas j dígitos.

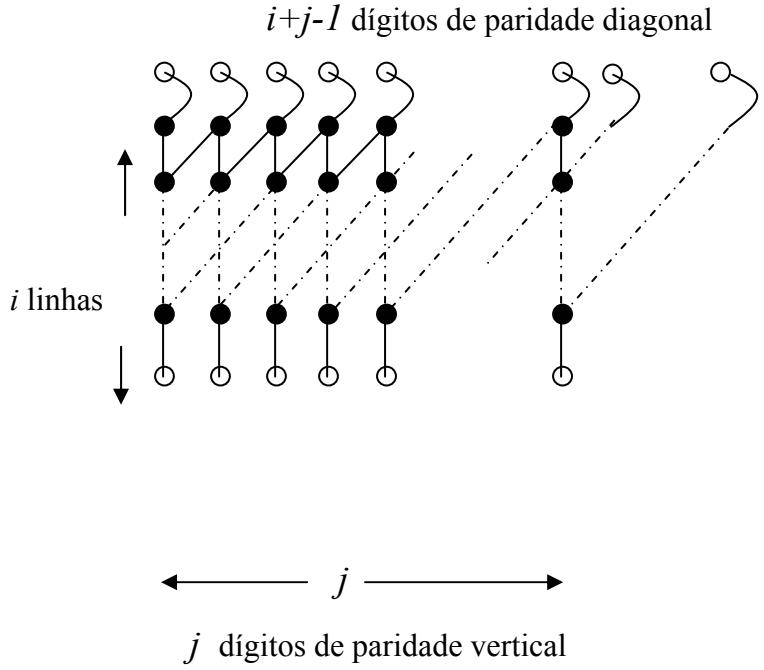


Figura 1.5: Dígitos de paridade ortogonais para correção de padrões com um único surto de erros por palavra.

Exemplo desta técnica:

Dígitos de paridade diagonal →

1	**0**	**0**	**1**	**0**
1	0	0		
0	1	1		
1	0	0		
0	**1**	**1**		

Dígitos de paridade vertical →

Ao juntar as diagonais na forma de dígitos de paridade helicoidal obtém-se o código da Figura 1.6. Este é um código matricial com parâmetros $(n_1n_2, k_1k_2, b)=((i+2)j, ij, b)$, sendo $i \leq j$, que é capaz de corrigir um surto de erros confinado a uma das $i+2$ linhas do código ou, alternativamente, de corrigir dois surtos de apagamento (*erasure burst*), em diferentes linhas [1].

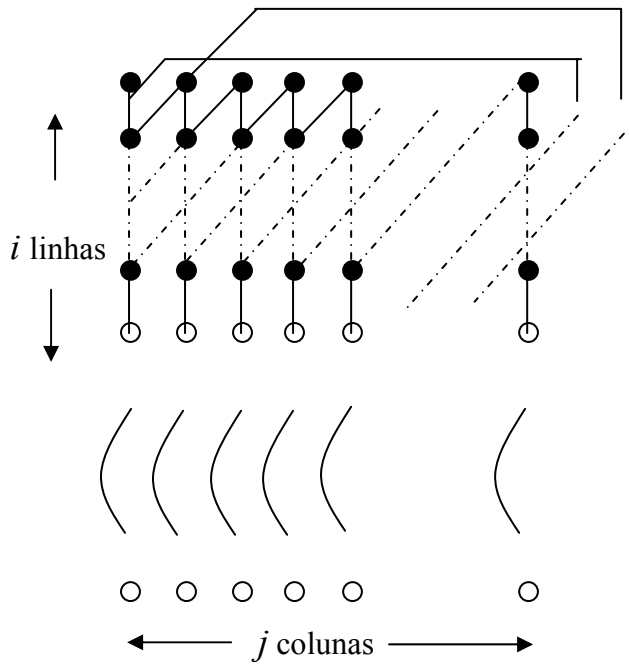
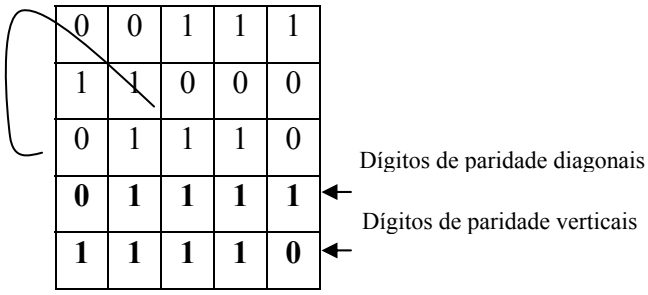


Figura 1.6: Código matricial para correção de surtos de erros em linhas (RBC) usando dígitos de paridade helicoidais.

Exemplo desta técnica:

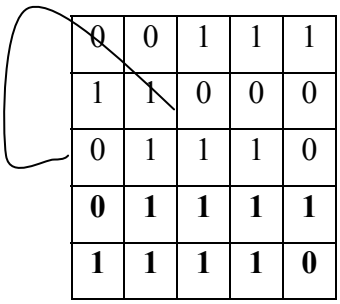


Observa-se que nesta ilustração é mostrada a técnica de determinação dos dígitos de paridade diagonais.

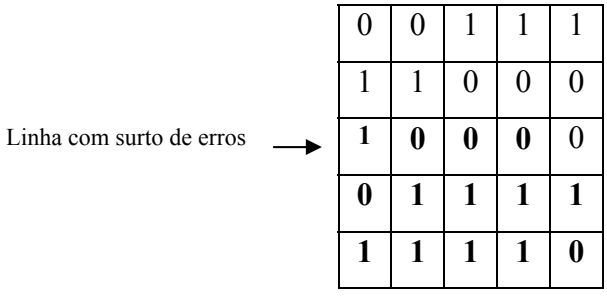
Neste tipo de montagem a capacidade de correção de erros em surto pode ser definida como: Se i, e a razão de j para i, são suficientemente grandes, então $b \leq j\text{-}1$ (uma vez que dois surtos de comprimento j podem ter a mesma síndrome). Quando $i = j$, então $2 \leq b \leq j\text{-}1$, para $j > 2$ [1]. Decodificar este tipo de código matricial de forma que possa fazer a correção do surto de erros na linha, consiste em alinhar, ciclicamente, os padrões de síndrome vertical e diagonal e então corrigir aqueles dígitos de informação para os quais ambos os dígitos de paridade falharam.

Exemplo:

Código matricial original:



Código matricial com surto de erros com comprimento quatro, na terceira linha:



Cálculo das Síndromes:

Síndrome Vertical: $S_v = 1\ 1\ 1\ 1\ 0$

Síndrome Diagonal: $S_d = 1\ 1\ 0\ 1\ 1$ (Calculado usando apenas as três primeiras linhas e a linha da paridade diagonal).

Neste caso a S_d precisa ser movimentada ciclicamente para a direita por duas vezes a fim de ficar igual a S_v. Logo, para corrigir o erro em surto, deve-se adicionar o valor de S_v à linha 2 (lembrando que neste exemplo existem as linhas 0, 1, 2, 3 e 4, observadas de cima para baixo). Desta forma o código matricial corrigido se torna:

0	0	1	1	1
1	1	0	0	0
0	1	1	1	0
0	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**

Importante:

a) Observa-se que para o exemplo anterior a capacidade de correção é de no máximo um surto de erros de comprimento quatro (equivalente a j-1), uma vez que as síndromes horizontal e diagonal, caso o comprimento do surto de erros seja cinco (igual a j), ficam todas "1", impossibilitando a identificação da linha errônea;

b) Este tipo de construção possibilita também a recuperação de dígitos apagados, como exemplificado a seguir.

0	0	1	1	1
x_1	x_2	x_3	x_4	0
y_1	y_2	y_3	1	y_4
0	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**

Nesta palavra de um código matricial, os dígitos identificados por x_1, x_2, x_3, x_4, y_1, y_2, y_3 e y_4 estão apagados. Pode-se usar a seguinte técnica de correção do apagamento:

- Recupera-se y_4 e x_4 pela síndrome vertical;
- Recupera-se x_3 pela síndrome diagonal;
- Uma vez recuperado x_3, recupera-se y_3 pela síndrome vertical;
- Uma vez recuperado y_3, recupera-se x_2 pela síndrome diagonal;
- Uma vez recuperado x_2, recupera-se y_2 pela síndrome vertical;
- Uma vez recuperado y_2, recupera-se x_1 pela síndrome diagonal;
- Recupera-se y_1 pela síndrome vertical.

Portanto é capaz de recuperar dígitos apagados em até duas linhas horizontais, desde que o comprimento do surto de apagamento, de pelo menos uma linha, seja menor ou no máximo igual a j-1.

Quando o código matricial é representado por uma matriz quadrada i x i então este tipo de código para correção de surtos de erros em linhas (RBC - *Row-Burst Correction Code*) pode ser rearranjado como mostra a Figura 1.7. Nesta ilustração, i=5 e, portanto, $b \leq 4$.

Dado que a ordem de leitura dos dígitos ainda é feita por linhas, estendendo-se até a coluna de paridade no lado direito, o valor de $b \leq j$-1 permanece o mesmo de um código não modificado [1].

1	2	3	4	**5**
6	7	8	9	**10**
11	12	13	14	**15**
16	17	18	19	**20**
21	22	23	24	**25**

Dígitos de paridade das diagonais

Dígitos de paridade das colunas

Em que os dígitos de paridade das diagonais correspondem a:

5 é dígito de paridade para 9, 13, 17, e 21
10 é dígito de paridade para 1, 14, 18 e 22
15 é dígito de paridade para 2, 6, 19 e 23
20 é dígito de paridade para 3, 7, 11 e 24
25 é dígito de paridade para 4, 8, 12 e 16

Figura 1.7: Código matricial quadrado (5 x 5) para correção de surtos de erros em linha (RBC) rearranjado.

Exemplo desta técnica:

1	0	1	0	1
0	1	0	1	0
1	0	1	1	0
1	0	1	1	0
1	1	1	1	1

Código matricial sem erros

0	**1**	**0**	**1**	1
0	1	0	1	0
1	0	1	1	0
1	0	1	1	0
1	1	1	1	1

Código matricial com surto de erros na linha 1

Cálculo das síndromes da palavra-código atingida por erros:

Síndrome vertical: S_v = 1 1 1 1 0
Síndrome diagonal: S_d = 1 1 1 1 0 (lida de baixo para cima).

Comparando-se S_v com S_d vê-se que as duas são iguais, logo basta corrigir a primeira linha por S_v = 1 1 1 1 0.

1.2.2.3 Código matricial para correção de surtos de erros nas diagonais (DBC - *Diagonal Burst Correcting Code*)

Se a leitura do código na Figura 1.7 é modificada para um padrão de leitura diagonal, conforme a Figura 1.8, o resultado é um código de correção de linhas e colunas (RAC) quadrado com leitura diagonal, capaz de corrigir um surto de erros por palavra, ou ainda

duplo surto de apagamento, os quais são confinados a um padrão diagonal. Usando a notação que foi adotada anteriormente, este código matricial é para correção de surtos de erros na diagonal (DBC - *Diagonal-burst-correcting code* ou *Phased-burst correcting code*) e tem parâmetros $(n_1n_2,\ k_1k_2,\ b)=(\ n_o^2,\ (n_o-1)^2\ ,\ b \le n_o-1)$. Colunas consistindo de dígitos de informação e dígitos de paridade vertical podem ser adicionadas a este código para produzir um código com características $(n_1n_2,\ (n_1\text{-}1)(n_2\text{-}1),\ b \le n_1-1)$ [1].

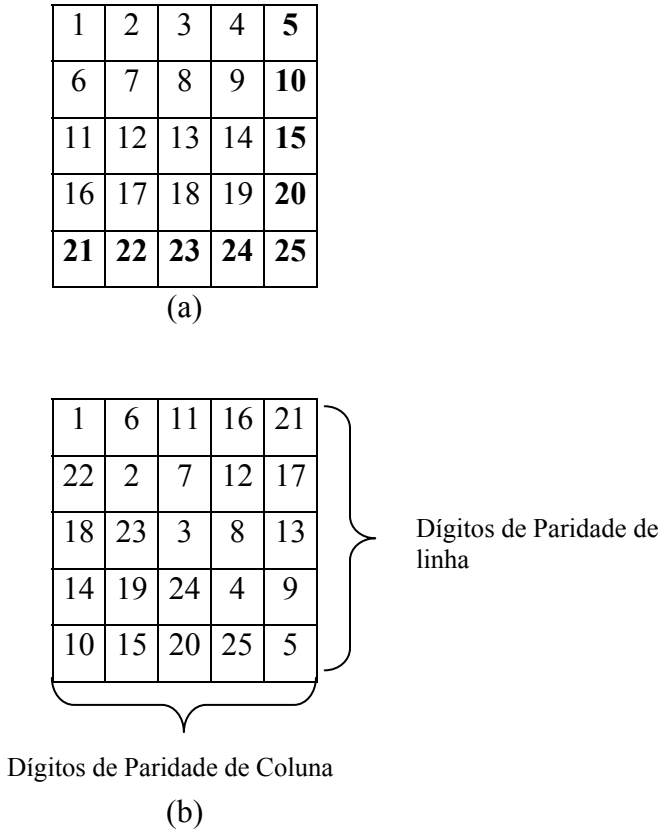
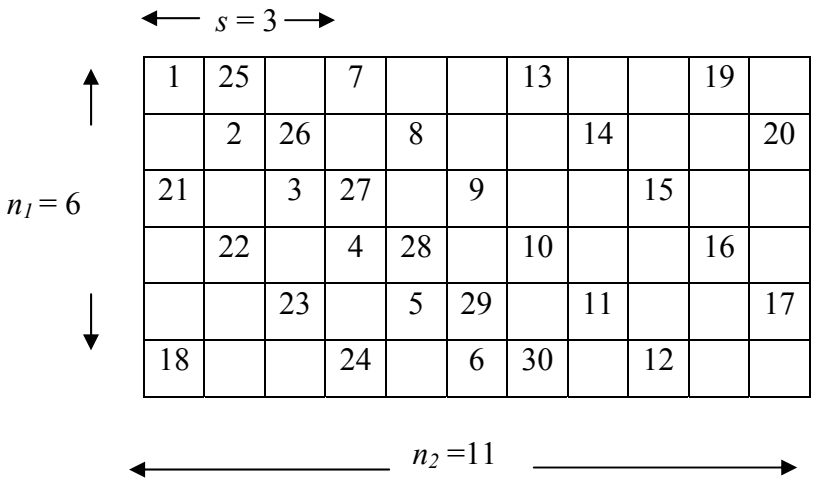


Figura 1.8: (a) Código matricial para correção de surtos de erros nas linhas (RBC);
(b) Código matricial para correção de surtos de erros nas diagonais (DBC), com leitura diagonal.

Dada a importância dos códigos matriciais de correção de surtos de erros nas diagonais (DBC - *Diagonal Burst Correcting Code*) será feita a seguir uma descrição com mais detalhes de algumas classes destes códigos para a correção de um surto de erros por palavra e, ao final, serão descritas outras classes usadas para correção de múltiplos surtos de erros.

Seja um código matricial binário n_1 x n_2 ($n_2 \geq n_1$), no qual cada linha e coluna têm peso par (código matricial para correção de linhas e colunas - RAC). A leitura diagonal começa do topo no lado esquerdo, caminha pela diagonal principal, "pula" para diagonal s ($1 \leq s \leq n_2 - 1$) e continua até a última diagonal ser lida. Um exemplo, para $s = 3$ é dado na Figura 1.9.



1	25		7			13			19	
	2	26		8			14			20
21		3	27		9			15		
	22		4	28		10			16	
		23		5	29		11			17
18			24		6	30		12		

Figura 1.9 – Código matricial 6x11, com parâmetro $s = 3$, para correção de surtos de erros nas diagonais (DBC – *Diagonal Burst Correcting Code*).

Os parâmetros n_2 e s têm que ser primos entre si, a fim de que todos os dígitos da palavra-código sejam lidos uma única vez. Códigos com esta construção podem corrigir um surto de erros por palavra, com comprimento $b \leq n_1$. O valor exato de b é função de n_1, n_2 e s. Primeiramente, estes códigos foram estudados por Gilbert [15] que usou $s = n_1$, e conseqüentemente n_1 e n_2 tinham que ser primos entre si. Neste modelo, os códigos eram cíclicos, com polinômio gerador dado por:

$$g(x) = (x^{n_1} - 1)(x^{n_2} - 1)/(x - 1).$$

O máximo comprimento para o surto de erros, b, que o código de Gilbert podia corrigir foi somente descoberto para alguns casos, até que Zhang e Wolf [2] determinaram b de maneira exata. Eles descobriram que $b \leq n_1$-s, se $1 \leq s \leq n_1$-1, ou $b \leq n_1$ se $n_1 \leq s \leq n_2$-1. Eles também foram capazes de determinar o valor de s que dá o máximo valor de b para qualquer combinação particular de valores n_1 e n_2.

Exemplo: $n_1 = 6$, $n_2 = 11$ e $s = 3$ ($1 \leq s \leq n_1\text{-}1$), logo $b \leq n_1\text{-}s = 3$

1	25	49	7	31	55	13	37	61	19	43
44	**2**	26	50	8	32	56	14	38	62	20
21	45	**3**	27	51	9	33	57	15	39	63
64	22	46	4	28	52	10	34	58	16	40
41	65	23	47	5	29	53	11	35	59	17
18	42	66	24	48	6	30	54	12	36	60

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 1\ 1\ 0\ 0\ 0$

Síndrome vertical: $S_v = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$.

Neste caso o erro pode ser corrigido uma vez que não existe nenhuma perturbação nas síndromes vertical e horizontal. Observa-se que tanto a síndrome vertical quanto a horizontal indicam a presença de três erros.Os casos seguintes têm comportamento similar:

1	25	49	7	31	55	13	37	61	19	43
44	2	26	50	8	32	56	14	38	62	20
21	45	3	27	51	9	33	57	15	39	63
64	22	46	**4**	28	52	10	34	58	16	40
41	65	23	47	**5**	29	53	11	35	59	17
18	42	66	24	48	**5**	30	54	12	36	60

Cálculo das síndromes:

Síndrome horizontal: $S_h = 0\ 0\ 0\ \ 1\ 1\ 1$

Síndrome vertical: $S_v = 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0$

1	25	49	**7**	31	55	13	37	61	19	43
44	2	26	50	8	32	56	14	38	62	20
21	45	3	27	51	9	33	57	15	39	63
64	22	46	4	28	52	10	34	58	16	40
41	65	23	47	**5**	29	53	11	35	59	17
18	42	66	24	48	**6**	30	54	12	36	60

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 0\ 0\ \ 0\ 1\ 1$

Síndrome vertical: $S_v = 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0$

Já no exemplo a seguir, com surto de erros de comprimento, *b*, igual a quatro, torna-se impossível a correção:

1	25	49	**7**	31	55	13	37	61	19	43
44	2	26	50	8	32	56	14	38	62	20
21	45	3	27	51	9	33	57	15	39	63
64	22	46	**4**	28	52	10	34	58	16	40
41	65	23	47	**5**	29	53	11	35	59	17
18	42	66	24	48	**6**	30	54	12	36	60

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 0\ 0\ \ 1\ 1\ 1$

Síndrome vertical: $S_v = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0$

Deve-se observar que, neste caso, enquanto a síndrome horizontal acusa a presença de quatro erros, a vertical indica duas posições erradas, uma vez que perde a informação de erro na coluna em que estão os dígitos 4 e 7. Neste caso, não seria possível a correção.

Exemplo:

$n_1 = 6$, $n_2 = 11$ e $s = 1$ ($1 \leq s \leq n_1\text{-}1$), logo $b \leq n_1\text{-}s = 5$

1	**7**	13	19	25	31	37	43	49	55	61
62	2	**8**	14	20	26	32	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	**4**	10	16	22	28	34	40	46
47	53	59	65	**5**	11	17	23	29	35	41
42	48	54	60	66	**6**	12	18	24	30	36

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 1\ 0\ \ 1\ 1\ 1$

Síndrome vertical: $S_v = 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0$

Observa-se que tanto a síndrome vertical quanto a horizontal indicam a presença de cinco erros (comprimento do surto de erros igual a cinco). Neste caso, o erro pode ser corrigido, uma vez que não existe nenhuma perturbação nas síndromes vertical e horizontal.

Supõe-se um surto de erros de comprimento igual a seis, mostrado a seguir:

1	**7**	13	19	25	31	37	43	49	55	61
62	2	**8**	14	20	26	32	38	44	50	56
57	63	3	**9**	15	21	27	33	39	45	51
52	58	64	**4**	10	16	22	28	34	40	46
47	53	59	65	**5**	11	17	23	29	35	41
42	48	54	60	66	**6**	12	18	24	30	36

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 1\ 1\ \ 1\ 1\ 1$

Síndrome vertical: $S_v = 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0$

Vide que neste caso a síndrome vertical é prejudicada na coluna onde estão os dígitos 4 e 9. Neste caso não seria possível a correção.

Exemplo:

$n_1 = 6$, $n_2 = 11$ e $s = 7$ ($n_1 \leq s \leq n_2 - 1$), logo $b \leq n_1 = 6$

1	49	31	13	61	43	25	**7**	55	37	19
20	2	50	32	14	62	44	26	**8**	56	38
39	21	3	51	33	15	63	45	27	**9**	57
58	40	22	**4**	52	34	16	64	46	28	10
11	59	41	23	**5**	53	35	17	65	47	29
30	12	60	42	24	**6**	54	36	18	66	48

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 1\ 1\ \ 1\ 1\ 1$

Síndrome vertical: $S_v = 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0$

Neste caso o erro pode ser corrigido uma vez que não existe nenhuma perturbação nas síndromes vertical e horizontal.Estas são perfeitamente definidas. Diferentemente do caso a seguir.

1	49	31	13	61	43	25	**7**	55	37	19
20	2	50	32	14	62	44	26	**8**	56	38
39	21	3	51	33	15	63	45	27	**9**	57
58	40	22	**4**	52	34	16	64	46	28	**10**
11	59	41	23	**5**	53	35	17	65	47	29
30	12	60	42	24	**6**	54	36	18	66	48

Cálculo das síndromes:

Síndrome horizontal: $S_h = 1\ 1\ 1\ \ 0\ 1\ 1$

Síndrome vertical: $S_v = 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1$

Vide que neste caso a síndrome horizontal é prejudicada na linha onde estão os dígitos 4 e 10. Neste caso não seria possível a correção.

O caso s=1, que foi ilustrado anteriormente, foi bastante investigado por Farrel e Hopkins [16,17]. Independentemente, Campello de Souza [18] e Blaum [3] provaram que $b = n_1 - 1$ para $n_2 \geq 2n_1$-3.

Supõe-se o seguinte exemplo, para s=1 e surto de erros de comprimento n_1-1 = 5, com erros nas posições 1 e 5:

1	7	13	19	25	31	37
38	2	8	14	20	26	32
33	39	3	9	15	21	27
28	34	40	4	10	16	22
23	29	35	41	**5**	11	17
18	24	30	36	42	6	12

As síndromes vertical e horizontal são:

$$S_h = 1\ 0\ 0\ 0\ 1\ 0$$
$$S_v = 1\ 0\ 0\ 0\ 1\ 0\ 0.$$

Estas mesmas síndromes poderiam ser geradas caso houvesse um surto de erros de comprimento três, como mostra a ilustração a seguir.

1	7	13	19	**25**	31	37
38	2	8	14	20	26	32
33	39	3	9	15	21	27
28	34	40	4	10	16	22
23	29	35	41	5	11	17
18	24	30	36	42	6	12

Cálculo das síndromes:

$$S_h = 1\ 0\ 0\ 0\ 1\ 0$$
$$S_v = 1\ 0\ 0\ 0\ 1\ 0\ 0.$$

Logo, comprova-se que não é possível corrigir um surto de erros de comprimento cinco, uma vez que não é atendida a condição $n_2 \geq 2n_1$-3, demonstrada pelos autores anteriormente mencionados.

Há muitas maneiras de decodificar estes códigos. O método mais simples é descrito por Mário Blaum [3] (artigo "*A Class of Burst Error Correcting Array Code*", nas equações (9), (10) e (11)) e consiste em se definir cada posição das síndromes como sendo uma determinada posição no código matricial. Vide o exemplo a seguir.

1	**7**	13	19	25	31	37	43	49	55	61
62	2	**8**	14	20	26	32	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	**4**	10	16	22	28	34	40	46
47	53	59	65	**5**	11	17	23	29	35	41
42	48	54	60	66	**6**	12	18	24	30	36

Cálculo das síndromes:

Síndrome horizontal: S_h = 1 1 0 1 1 1

Síndrome vertical: S_v = 0 1 1 1 1 1 0 0 0 0 0.

Neste exemplo a síndrome horizontal estaría representando as posições das linhas (*i*), enquanto a síndrome vertical representa as posições das colunas (*j*), logo os valores das posições seriam:

S_h	1	1	0	1	1	1
I	0	1	2	3	4	5

e

S_v	0	1	1	1	1	1	0	0	0	0	0
J	0	1	2	3	4	5	6	7	8	9	10

Logo as posições (*i*,*j*) que representam a solução do problema (para os quais os dígitos na síndrome valem um) são:

i	0	1	3	4	5
j	1	2	3	4	5

Solução: (0,1), (1,2), (3,3), (4,4) e (5,5)

Outro exemplo:

1	7	13	19	25	31	37	43	49	55	61
62	2	**8**	14	20	26	32	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	**4**	10	16	22	28	34	40	46
47	53	59	65	5	11	17	23	29	35	41
42	48	54	60	66	6	12	18	24	30	36

Cálculo das síndromes:

Síndrome horizontal: $S_h = 0\ 1\ 0\ 1\ 0\ 0$

Síndrome vertical: $S_v = 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

S_h	0	1	0	1	0	0
I	0	1	2	3	4	5

e

S_v	0	0	1	1	0	0	0	0	0	0	0
J	0	1	2	3	4	5	6	7	8	9	10

Logo as posições (*i*,*j*) que representam a solução do problema são:

i	1	3
j	2	3

Solução: (1,2) e (3,3).

A seguir é descrito um outro algoritmo que pode ser usado para a correção de erros em surto neste tipo de código matricial.

Seja

$f(i,j) = i + (j-i)(n_1) + 1 \bmod (n_1 n_2)$.

O valor de $f(i,j)$ representa cada posição no código matricial, e

$d_m = | f(i_1,j_1) - f(i_2,j_2) | + 1 \bmod (n_1 n_2)$.

O parâmetro d_m representa a distância entre as posições da matriz (i_1,j_1) e (i_2,j_2).

Exemplo:

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(0,8)	(0,9)	(0,10)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)	(1,10)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)	(2,10)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)	(3,10)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)	(4,9)	(4,10)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)	(5,9)	(5,10)

Isto gera a seguinte matriz com os valores de $f(i,j)$

1	7	13	19	25	31	37	43	49	55	61
62	2	8	14	20	26	32	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	4	10	16	22	28	34	40	46
47	53	59	65	5	11	17	23	29	35	41
42	48	54	60	66	6	12	18	24	30	36

Detalhamento do algoritmo:

Tomando, por exemplo, o último caso analisado, tem-se

1	7	13	19	25	31	37	43	49	55	61
62	2	8	14	20	26	**32**	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	4	10	16	22	**28**	34	40	46
47	53	59	65	5	11	17	23	29	35	41
42	48	54	60	66	6	12	18	24	30	36

Síndrome horizontal: S_h = 0 1 0 1 0 0

Síndrome vertical: S_v = 0 0 0 0 0 0 1 1 0 0 0.

Então, pode-se identificar as possíveis posições em que ocorreram os erros em função dos "1" nas síndromes vertical e horizontal, como exibido a seguir.

i	S_h												
0	0												
1	1												
2	0												
3	1												
4	0												
5	0												
		0	0	0	0	0	0	1	1	0	0	0	S_v
		0	1	2	3	4	5	6	7	8	9	10	j

Por esta representação existem dois erros (peso das síndromes) que podem estar localizados nas posições (1,6) com (3,7) ou (1,7) com (3,6). O processo de decodificação se resume ao cálculo da menor distância entre os pontos que tem que ser menor do que n_1-1 = 5 (neste caso). Então

Distância entre os pontos (1,6) e (3,7):

d_m = | f(1,6) – f(3,7)| +1 = |32-28| + 1 = 5.

Distância entre os pontos (1,7) e (3,6):

d_m = | f(1,7) – f(3,6)| + 1 = | 38 –22 | +1 = 17 > 5. Logo pela regra da menor distância a solução seria corrigir as posições (1,6) e (3,7).

Seja a seguinte situação:

1	**7**	13	19	25	31	37	43	49	55	61
62	2	**8**	14	20	26	32	38	44	50	56
57	63	3	9	15	21	27	33	39	45	51
52	58	64	**4**	10	16	22	28	34	40	46
47	53	59	65	**5**	11	17	23	29	35	41
42	48	54	60	66	**6**	12	18	24	30	36

Síndrome horizontal: S_h = 1 1 0 1 1 1

Síndrome vertical: S_v = 0 1 1 1 1 1 0 0 0 0 0

Existem várias combinações possíveis (todas as que estão hachuradas na matriz acima), porém apenas uma atende à condição de menor distância. Caso contrário, não é possível a correção do surto de erros. O processo de correção teria a seguinte seqüência:

a) Inicialmente, após o cálculo das síndromes do código matricial, verifica-se que ambas possuem o mesmo peso e este é inferior ou no máximo igual a n_1-1. Se estas condições iniciais são atendidas é possível a correção do surto de erros e passa-se para o procedimento seguinte.

b) Em seguida, com base nas síndromes, isola-se as posições válidas, como mostrado a seguir:

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

Portanto as posições válidas são:

1) Para as linhas (*i*): 0, 1, 3, 4, 5
2) Para as colunas (*j*): 1, 2, 3, 4, 5

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

c) Elege-se uma das posições da parte superior do arranjo como sendo parte do surto de erros, por exemplo: $i = 0$ e $j=1$. Neste caso o campo de possíveis erros passa a ser o seguinte.

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

A partir daí procura-se na linha subseqüente (neste caso i=1) a posição que apresenta a menor distância (menor ou igual que cinco, para este exemplo). Lembrando que neste caso uma vez fixado este ponto de origem, a coluna j=1 não faz parte do grupo de posições a ser testado.

Caso não se encontre nenhum ponto, dentre estes mostrados na representação anterior, que tenha distância inferior à máxima permitida, este possível erro é descartado e parte-se para a próxima coluna na linha i=0 e repete-se o procedimento descrito. Caso ao final, não se encontre a solução, pode-se afirmar que não é possível a correção do surto de erros (pelo menos por este método).

Demonstra-se que, para este exemplo, a posição (1,2) é a solução, pois apresenta a menor distância em relação à posição (0,1) de valor igual a um, como se pode ver na equação a seguir.

$$d_m = |\, f(0,1) - f(1,2)| + 1 = |7-8| + 1 = 2 \leq 5\ (=n_1-1)$$

Uma vez encontrado este segundo ponto, a área de possíveis erros passa a ser:

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

d) Neste caso, fixa-se i=3 e testa-se as posições (3,3), (3,4) e (3,5) em relação aos últimas posições consideradas com erros. Neste exemplo, a posição (3,3) é a que apresenta menor distância:

$$d_m = |\, f(1,2) - f(3,3)| + 1 = |8-4| + 1 = 5 \leq 5\ (=n_1-1).$$
$$d_m = |\, f(0,1) - f(3,3)| + 1 = |7-4| + 1 = 4 \leq 5\ (=n_1-1).$$

Uma vez encontrado este terceiro ponto, a área de possíveis erros passa a ser:

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

e) Neste caso, fixa-se i=4 e testa-se as posições (4,4) e (4,5) em relação aos últimos erros considerados válidos. Neste exemplo a resposta seria a posição (4,4) que apresenta a menor distância:

$$d_m = |\, f(3,3) - f(4,4)| + 1 = |4-5| + 1 = 2 \leq 5\ (=n_1-1).$$
$$d_m = |\, f(1,2) - f(4,4)| + 1 = |8-5| + 1 = 4 \leq 5\ (=n_1-1).$$
$$d_m = |\, f(0,1) - f(4,4)| + 1 = |7-5| + 1 = 3 \leq 5\ (=n_1-1).$$

Uma vez encontrado este quarto ponto, a área de possíveis erros passa a ser:

0	1											
1	1											
2	X											
3	1											
4	1											
5	1											
		X	1	1	1	1	1	X	X	X	X	X
		0	1	2	3	4	5	6	7	8	9	10

Portanto, fica restrita exclusivamente à posição (5,5). Neste caso para se confirmar que atende ao requisito de distância máxima, executa-se o seguinte cálculo em relação a todos os erros encontrados:

$$d_m = |\, f(0,1) - f(5,5)| + 1 = |7-6| + 1 = 2 \leq 5 \; (= n_1 - 1).$$
$$d_m = |\, f(1,2) - f(5,5)| + 1 = |8-6| + 1 = 3 \leq 5 \; (= n_1 - 1).$$
$$d_m = |\, f(3,3) - f(5,5)| + 1 = |4-6| + 1 = 3 \leq 5 \; (= n_1 - 1).$$
$$d_m = |\, f(4,4) - f(5,5)| + 1 = |5-6| + 1 = 2 \leq 5 \; (= n_1 - 1).$$

Portanto, a solução para o surto de erros, com base neste algoritmo, seria formada pelas posições (0,1), (1,2), (3,3), (4,4) e (5,5).

O caso $s = n_2 - 1$ (abreviado $s = -1$) é interessante porque é um dos dois valores de s que permitem $b = n_1$ [5]. É demonstrado que para $s = -1$, $b = n_1$ se $n_2 \geq 2n_1 + 1$. Pode-se utilizar um algoritmo de decodificação rápido e simples. O número de dígitos de paridade é de $n\text{-}k = 3n_1$ quando $n_2 = 2n_1 + 1$.

Exemplos da construção $s = -1$:

0	72	66	60	**54**	48	42	36	30	24	18	12	6
7	1	73	67	61	55	49	43	37	31	25	19	13
14	8	2	74	68	62	56	50	44	38	32	26	20
21	15	9	3	75	69	63	57	51	45	39	33	27
28	22	16	10	4	76	70	64	58	52	46	40	34
35	29	23	17	11	5	77	71	65	**59**	53	47	41

Síndrome horizontal: S_h = 1 0 0 0 0 1

Síndrome vertical: S_v = 0 0 0 0 1 0 0 0 0 1 0 0 0

Pode-se, para corrigir o surto de erros, utilizar o mesmo algoritmo descrito anteriormente. Primeiramente observa-se que o peso das síndromes é o mesmo e corresponde a dois, indicando a existência de dois erros, que podem ser:

0	72	66	60	**54**	48	42	36	30	24	18	12	6
7	1	73	67	61	55	49	43	37	31	25	19	13
14	8	2	74	68	62	56	50	44	38	32	26	20
21	15	9	3	75	69	63	57	51	45	39	33	27
28	22	16	10	4	76	70	64	58	52	46	40	34
35	29	23	17	11	5	77	71	65	**59**	53	47	41

ou

0	72	66	60	54	48	42	36	30	**24**	18	12	6
7	1	73	67	61	55	49	43	37	31	25	19	13
14	8	2	74	68	62	56	50	44	38	32	26	20
21	15	9	3	75	69	63	57	51	45	39	33	27
28	22	16	10	4	76	70	64	58	52	46	40	34
35	29	23	17	**11**	5	77	71	65	59	53	47	41

A solução recai naquele par que oferece a menor distância. Neste caso no par representado pelas posições 54 e 59. Observa-se que neste caso a distância vale seis (menor ou igual a n_1) e, portanto está dentro do limite aceitável de correção deste código.

Pode-se calcular as posições da seguinte forma:

$$f(i,j) = i + (i-j)(n_1) \bmod (n_1 n_2)$$

Também pode-se chegar ao mesmo resultado e de forma muito mais simples usando o método descrito por Blaum [3], ou seja:

S_h	1	0	0	0	0	1
I	0	1	2	3	4	5

e

S_v	0	0	0	0	1	0	0	0	0	1	0	0	0
J	0	1	2	3	4	5	6	7	8	9	10	11	12

Logo as posições (i,j) que representam a solução do problema são:

i	0	5
j	4	9

Solução: (0,4) e (5,9).

O caso $s = -2$, a exemplo do caso $s = -1$, também permite $b = n_1$ [4], desde que $n_1 = 4u + v + 2$ e $n_2 = 6u + 2v + 5$, $u \geq 1$ e $v \geq 0$, mas sendo v diferente de 1.

Exemplo desta técnica:

$u = 1$ e $v = 0$, então $n_1 = 6$ e $n_2 = 11$,

0	30	60	24	54	18	48	12	42	6	36
37	1	31	61	25	55	19	49	13	43	7
8	38	2	32	62	26	56	20	50	14	44
45	9	39	3	33	63	27	57	21	51	15
16	46	10	40	4	34	64	28	58	22	52
53	17	47	11	41	5	35	65	29	59	23

Exemplo de decodificação:

Supõe-se erros nas posições 27 e 30, indicando surto de comprimento quatro

0	**30**	60	24	54	18	48	12	42	6	36
37	1	31	61	25	55	19	49	13	43	7
8	38	2	32	62	26	56	20	50	14	44
45	9	39	3	33	63	**27**	57	21	51	15
16	46	10	40	4	34	64	28	58	22	52
53	17	47	11	41	5	35	65	29	59	23

Cálculo das Síndromes:

Síndrome horizontal: S_h = 1 0 0 1 0 0

Síndrome vertical: S_v = 0 1 0 0 0 0 1 0 0 0 0

Baseando-se no método de Blaum [3], chega-se seguinte solução:

S_h	1	0	0	1	0	0
I	0	1	2	3	4	5

e

S_v	0	1	0	0	0	0	1	0	0	0	0
J	0	1	2	3	4	5	6	7	8	9	10

Logo as posições (*i, j*) que representam a solução do problema são:

i	0	3
j	1	6

Solução: (0,1) e (3,6).

Para a correção de t surtos de erros pode-se usar um código matricial binário n_1 x n_2 ($n_2 \geq n_1$), cujas linhas consistem de palavras-código de um código (n_2, k_2, d=2t) e colunas com paridade par (i.e., as colunas tem como parâmetros de código (n_1, n_1-1, 2)), perfazendo um código matricial com parâmetros (n_1n_2, k_1k_2, d) = (n_1n_2, (n_1-1)k_2, 4t). A leitura diagonal e o parâmetro s são usados também neste caso. Daniel [19,20] desenvolveu códigos matriciais deste tipo com parâmetros t=2 e s=1. Estes códigos foram construídos heuristicamente, por manipulação da matriz de verificação de paridade do código estendido de Hamming, para assegurar correção de duplo surto de erros. Em muitos casos $b = n_1$-1 foi conseguido, mas em alguns casos somente $b= n_1 -2$ foi possível, porque n_2 era muito pequeno. Estes códigos foram todos capazes de detecção de único surto de erros de comprimento $b= 2n_1$-1.

Estes códigos matriciais foram efetivamente generalizados por Blaum [5] que provou que, com o parâmetro s = 1, o código matricial definido no parágrafo anterior pode corrigir até t surtos de erros de comprimento menor ou igual a n_1-1 se $n_2 \geq 2t(n_1-2) +1$. Também foi demonstrado em [5] que para s = -1 o código pode corrigir até t surtos de erros de comprimento menor ou igual a n_1 se $n_2 \geq 2tn_1 + 1$. Em ambos os casos, as linhas são constituídas de palavras-código de distância 2t.

O processo de decodificação foi descrito em [5], usando um algoritmo, que consistia em calcular as síndromes de cada linha. Caso o peso da síndrome de determinada linha fosse menor do que t então o código de linha poderia corrigir, caso contrário, ou seja, se o peso da síndrome da linha fosse igual a t, teria que ser utilizado um algoritmo mais complexo, que consistia em identificar uma seqüência de zeros (*gap*) livre de erros na síndrome das colunas, localizar a coluna seguinte à seqüência e corrigi-la, conforme descrito nos exemplos a seguir.

Exemplos: para t=2

1	**5**	9	13	17	21	25	29	33
34	**2**	**6**	10	14	18	22	26	30
31	35	**3**	**7**	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Neste caso, observa-se que a primeira linha apresentará síndrome de peso dois (igual a t) e, portanto, não é possível corrigir pelo código de linha. A síndrome das colunas, que é designada de síndrome vertical, S_v, é dada por:

Posição	1	2	3	4	5	6	7	8	9
S_v	1	0	0	1	**0**	**0**	**0**	**0**	**0**

Observa-se que o maior vazio (*gap*) ocorre entre as posições 5 a 9. Pelo algoritmo, deve-se escolher a posição subseqüente à última coordenada deste *gap* para correção, ou seja, posição 1. Logo, deve-se corrigir a posição indicada pela linha 1 e coluna 1. Corrigindo esta posição, na linha 1 fica apenas um erro, no símbolo 5, e portanto é possível a correção pelo código de linha. O código matricial fica assim representado:

1	5	9	13	17	21	25	29	33
34	**2**	**6**	10	14	18	22	26	30
31	35	**3**	**7**	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Passa-se para a próxima linha e calcula-se novamente a sua síndrome. Neste caso, verifica-se que possui erro de peso dois e que, portanto, o código de linha não consegue corrigir. Parte-se novamente para o cálculo da síndrome vertical, mostrada a seguir:

Posição	1	2	3	4	5	6	7	8	9
S_v	**0**	1	0	1	**0**	**0**	**0**	**0**	**0**

Vê-se que a maior seqüência de zeros consecutivos, vistos ciclicamente, finda na posição 1. Portanto a posição (2,2), que corresponde ao símbolo 2 no código matricial, deverá ser corrigida. Uma vez corrigido este símbolo, o código de linha corrige o símbolo 6 e o código matricial fica da seguinte forma:

1	5	9	13	17	21	25	29	33
34	2	6	10	14	18	22	26	30
31	35	**3**	**7**	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Neste caso,

Posição	1	2	3	4	5	6	7	8	9
S_v	**0**	**0**	1	1	**0**	**0**	**0**	**0**	**0**

A seqüência de zeros consecutivos finda na posição 2. Portanto, deve-se corrigir a posição indicada pela linha 3 e pela coluna 3, que corresponde ao símbolo 3. Em seguida, fechando a seqüência de decodificação, o código de linha corrige a posição representada pelo símbolo 7.

Outro exemplo:

1	5	9	13	17	21	25	**29**	33
34	**2**	6	10	14	18	22	26	**30**
31	35	**3**	7	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Neste caso, observa-se que a primeira linha também apresenta síndrome de peso dois (igual a t) e, portanto, não é possível corrigir pelo código de linha. A síndrome vertical, S_v, é dada por:

Posição	1	2	3	4	5	6	7	8	9
S_v	0	1	1	**0**	**0**	**0**	**0**	1	1

Observa-se que a mais longa seqüência de zeros consecutivos ocorre entre as posições 4 e 7. Pelo algoritmo, deve-se escolher a posição subseqüente à última coordenada deste *gap*, ou seja, posição 8. Logo, deve-se corrigir a posição indicada pela linha 1 e coluna 8 (representada pelo símbolo 29). Corrigindo esta posição, na linha 1 fica apenas um erro, no símbolo 1, e portanto é possível a correção pelo código de linha. A palavra do código matricial fica assim representada:

1	5	9	13	17	21	25	29	33
34	**2**	6	10	14	18	22	26	**30**
31	35	**3**	7	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Passa-se para a linha subsequente e calcula-se novamente a sua síndrome. Neste caso, verifica-se que possui erro de peso dois e que, portanto, o código de linha não consegue corrigir. Parte-se então para o cálculo da síndrome vertical, mostrada a seguir:

Posição	1	2	3	4	5	6	7	8	9
S_v	1	1	1	**0**	**0**	**0**	**0**	**0**	1

Vê-se que o maior *gap* (visto ciclicamente) finda na posição 8. Logo a posição (2,9), representada pelo símbolo 30 deverá ser corrigida. Uma vez corrigido este símbolo o código de linha corrige o símbolo 2 e o código matricial fica da seguinte forma:

1	5	9	13	17	21	25	29	33
34	2	6	10	14	18	22	26	30
31	35	**3**	7	11	15	19	23	27
28	32	36	4	8	12	16	20	24

Para este caso,

Posição	1	2	3	4	5	6	7	8	9
S_v	1	0	1	**0**	**0**	**0**	**0**	**0**	**0**

O *gap* finda na posição 9. Portanto, deve-se corrigir a posição indicada pela linha 3 e coluna 1, que corresponde ao símbolo 31. Em seguida, fechando a sequência de decodificação, o código de linha corrige a posição representada pelo símbolo 3.

Problema:

1	5	9	**13**	17	21	25	29	33
34	2	6	10	14	18	22	26	30
31	35	3	7	11	15	19	23	27
28	32	36	**4**	8	12	**16**	20	24

Pelo método demonstrado por Blaum [5] é necessária a identificação na síndrome vertical de um *gap* com comprimento maior ou igual a n_1 (nesta ilustração n_1=4) posições, livres de erro. Neste caso, observa-se que a síndrome vertical fica igual a:

Posição	1	2	3	4	5	6	7	8	9
S_v	1	**0**	**0**	**0**	**0**	**0**	1	0	0

Portanto o *gap* inicia na posição 2 e finda na posição 6, tendo comprimento igual a cinco. No entanto, este *gap* não é livre de erro, uma vez que os erros nas posições 13 e 4 geraram um zero na posição 4. Se for usado o algoritmo anteriormente mostrado, será introduzido mais um erro na posição 25 (correspondente à linha 1 e coluna 7) e quando for recalculada a síndrome horizontal da linha se descobrirá que o peso é três e portanto maior que t=2. Logo este tipo de surto de erros de comprimento n_1 não poderá ser corrigido por este tipo de código matricial. A seguir é apresentado mais um exemplo:

1	5	**9**	13	17	21	25	29	33
34	2	6	10	14	18	22	26	30
31	35	**3**	7	**11**	15	19	23	27
28	32	36	4	8	12	16	20	24

Posição	1	2	3	4	5	6	7	8	9
S_v	1	0	0	0	1	**0**	**0**	**0**	**0**

Neste caso, o maior *gap* inicia-se na posição 6 e finda na posição 9. A correção dos erros, pelo uso do método exposto, é plenamente exeqüível e funcional, como já foi ilustrado nos exemplos anteriores.

Uma outra técnica geral para a construção de códigos matriciais para correção de múltiplos surtos de erros (*multiple-burst-correcting array codes*) é baseada no uso de códigos de linha para correção de t-erros e códigos para as colunas com um único dígito de paridade (SPC *code*). Este tipo de código matricial irá corrigir até $2t$ surtos de erros. Isto deve-se às propriedades do produto dos códigos matriciais: Se o código da linha tem uma distância para correção de surtos de erros de $d = 2t+1$, então o código matricial gerado terá uma distância $d = 2.(2t+1) = 4t+2$, ou seja, o código matricial pode corrigir até $(4t + 1)/2 = 2t$ surtos de erros de comprimento b. O código da linha pode ser, por si próprio, um código matricial, sendo que neste caso resulta num código de três dimensões.

Bridwell e Wolf [21] estudaram as propriedades de correção de múltiplos surtos de erros para códigos de duas dimensões, construídos a partir de códigos usados nas linhas e nas colunas (não necessariamente cíclicos) com parâmetros $(n, k, d) = (n_1, k_1, d_{bb1})$ e $(n, k, d) = (n_2, k_2, d_{bb2})$. Neste caso, o processo de leitura diagonal de Gilbert [15] foi usado.

1.2.3 Códigos matriciais de bloco para correção de surtos de erros em duas dimensões (*Clusters* ou *patches* de erros)

Surtos de erros em duas dimensões (*clusters* ou *patches*) ocorrem em aplicações como transmissões digitais, processamento digital de sinais ou sistemas de armazenamento, nas quais se deseja que os dados sejam formatados em duas dimensões (por exemplo, fita magnética, etc.). Códigos matriciais são adequados para corrigir erros em estruturas de dados de duas dimensões.

A forma mais simples de código para correção de surto bidimensional de erros (*single cluster*) é o código matricial de dígitos de paridade para as linhas e colunas (RAC-*row and column code*) entrelaçado. Um surto de erros bidimensional (*cluster*) $b_i x b_j = l_p$ x l_p de ε erros, $1 \leq \varepsilon \leq l_p^2$, em um código matricial de parâmetros $(k_1 + l_p)$ $(k_2 + l_p)$, com uma estrutura matricial de informação k_1 x k_2, pode ser corrigido por entrelaçamento de linhas e colunas, por uma profundidade l_p. Os surtos de erros bidimensionais na maioria das vezes são retangulares e neste caso a profundidade de entrelaçamento será distinta para linhas e colunas.

Códigos matriciais entrelaçados por linhas em uma profundidade l_p, podem ser usados para corrigir um surto de erros bidimensional de l_p x l_p erros, com ε erros, $1 \leq \varepsilon \leq l_p^2$. O entrelaçamento de colunas também é possível.

Códigos matriciais do tipo corretor de surtos de erros nas suas linhas (RBC – *row burst correcting code*) ou corretor de surtos de erros nas suas diagonais (DBC – *diagonal burst correcting code*) também podem ser utilizados. Estes códigos são mais eficientes do que os códigos RAC entrelaçados. Um exemplo de código matricial para correção de surtos de erros bidimensionais usando esta técnica foi desenvolvido por Blaum e Farrell [9] cuja descrição é dada a seguir.

Estes códigos matriciais com n_1 linhas e n_2 colunas são capazes de corrigir um surto de erros bidimensional na forma retangular de dimensões $b_i x b_j = b_1 x b_2$ de ε erros, em que $1 \leq \varepsilon \leq b_1 b_2$. As linhas e colunas deste código matricial têm paridade par (tipo SPC - *single parity check*).

A codificação e a decodificação deste tipo de código matricial envolve operações de lógica boolena X-OR e deslocamentos cíclicos de vetores binários.

Supõe-se que se deseja corrigir um surto de erros bidimensional de tamanho máximo b_1 x b_2 em um código matricial n_1 x n_2. Este surto de erros bidimensional é visto ciclicamente em ambas as direções, ou seja, o código matricial tem a topologia de um toróide. Por exemplo, se $n_1 = n_2 = 6$, os seguintes surtos de erros bidimensionais tem tamanho de no máximo 2 x 3:

		1	0	1	
		1	1	1	

1	1				1
1	1				1

1	1				1
1	0				1

Seja $n_1 \geq 2b_1b_2 - b_1$, $n_2 \geq 2b_1b_2$ e supõe-se que b_1 divida n_1 e b_2 divida n_2. Se for denotada por ρ uma rotação do vetor para a direita, por exemplo, seja o vetor $\mathbf{u}$ = 0 1 0 1 0, então $\rho(\mathbf{u})$ = 0 0 1 0 1. Conseqüentemente ρ^q representa q rotações do vetor para a direita ao passo que ρ^{-q} representa q rotações para a esquerda, sendo $q \geq 0$. Se $\mathbf{u}$ é um vetor vertical, ρ^q representa q rotações para baixo. Seja $<u>_v$ representando um único inteiro κ, $0 \leq \kappa \leq v$-1, tal que $\kappa \equiv u (\text{mod } v)$.

A informação é armazenada em uma estrutura matricial A_{-1} de dimensões (n_1-1)x(n_2-1). O primeiro passo é codificá-lo em um arranjo A de dimensões $n_1 x n_2$, usando paridade par para as linhas e colunas.

No passo a seguir para cada linha r_i de A, $0 \leq i \leq n_1 - 1$, aplica-se uma rotação $\rho^{b_2 <i>_{b_1}}$. Representa-se por A_r o arranjo obtido desta forma. Em seguida, deve-se aplicar a cada coluna c_j de A_r , $0 \leq j \leq n_2 - 1$, a rotação $\rho^{b_1 <j>_{b_2}}$. Representa-se por A_b o arranjo obtido desta operação, completando a codificação. Farrell e Blaum [9] provaram que atendidas as condições $n_1 \geq 2b_1b_2 - b_1$, $n_2 \geq 2b_1b_2$ e supondo que b_1 divida n_1 e b_2 divida n_2, este tipo de código é capaz de corrigir um surto de erros bidimensionais do tamanho b_1x b_2.

Exemplo: Seja $b_1 = 2$ e $b_2 = 3$, então $n_1 \geq 10$ e $n_2 \geq 12$. Adotando-se para este exemplo $n_1 = 10$ e $n_2 = 12$, então (observa-se que b_1 divide n_1 e b_2 divide n_2).

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}
e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}
f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}
j_0	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}

Geração de A_r:

Para cada linha r_i de A, $0 \leq i \leq n_1 - 1$, aplica-se uma rotação $\rho^{b_2 <i>_{b_1}} = \rho^{3<i>_2}$. Então

Linha (i)	$\langle i \rangle_2$	Número de rotações
0	0	0
1	1	3
2	0	0
3	1	3
4	0	0
5	1	3
6	0	0
7	1	3
8	0	0
9	1	3

Logo a matriz A_r será:

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_9	b_{10}	b_{11}	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
d_9	d_{10}	d_{11}	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}
f_9	f_{10}	f_{11}	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
h_9	h_{10}	h_{11}	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}
j_9	j_{10}	j_{11}	j_0	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8

Geração de A_b:

Aplicar a cada coluna c_j de A_r, $0 \leq j \leq n_2 - 1$, a rotação $\rho^{b_1 \langle j \rangle_{b_2}} = \rho^{2 \langle j \rangle_3}$

Coluna (j)	$\langle j \rangle_3$	Número de rotações
0	0	0
1	1	2
2	2	4
3	0	0
4	1	2
5	2	4
6	0	0
7	1	2
8	2	4
9	0	0
10	1	2
11	2	4

Logo a matriz A_b fica:

a_0	i_1	g_2	a_3	i_4	g_5	a_6	i_7	g_8	a_9	i_{10}	g_{11}
b_9	j_{10}	h_{11}	b_0	j_1	h_2	b_3	j_4	h_5	b_6	j_7	h_8
c_0	a_1	i_2	c_3	a_4	i_5	c_6	a_7	i_8	c_9	a_{10}	i_{11}
d_9	b_{10}	j_{11}	d_0	b_1	j_2	d_3	b_4	j_5	d_6	b_7	j_8
e_0	c_1	a_2	e_3	c_4	a_5	e_6	c_7	a_8	e_9	c_{10}	a_{11}
f_9	d_{10}	b_{11}	f_0	d_1	b_2	f_3	d_4	b_5	f_6	d_7	b_8
g_0	e_1	c_2	g_3	e_4	c_5	g_6	e_7	c_8	g_9	e_{10}	c_{11}
h_9	f_{10}	d_{11}	h_0	f_1	d_2	h_3	f_4	d_5	h_6	f_7	d_8
i_0	g_1	e_2	i_3	g_4	e_5	i_6	g_7	e_8	i_9	g_{10}	e_{11}
j_9	h_{10}	f_{11}	j_0	h_1	f_2	j_3	h_4	f_5	j_6	h_7	f_8

Exemplo de erros que podem ser corrigidos:

a_0	i_1	g_2	a_3	i_4	g_5	a_6	i_7	g_8	a_9	i_{10}	g_{11}
b_9	j_{10}	h_{11}	b_0	j_1	h_2	b_3	j_4	h_5	b_6	j_7	h_8
c_0	a_1	i_2	c_3	a_4	i_5	c_6	a_7	i_8	c_9	a_{10}	i_{11}
d_9	b_{10}	j_{11}	d_0	b_1	j_2	d_3	b_4	j_5	d_6	b_7	j_8
e_0	c_1	a_2	e_3	$\mathbf{c_4}$	$\mathbf{a_5}$	$\mathbf{e_6}$	c_7	a_8	e_9	c_{10}	a_{11}
f_9	d_{10}	b_{11}	f_0	$\mathbf{d_1}$	$\mathbf{b_2}$	$\mathbf{f_3}$	d_4	b_5	f_6	d_7	b_8
g_0	e_1	c_2	g_3	e_4	c_5	g_6	e_7	c_8	g_9	e_{10}	c_{11}
h_9	f_{10}	d_{11}	h_0	f_1	d_2	h_3	f_4	d_5	h_6	f_7	d_8
i_0	g_1	e_2	i_3	g_4	e_5	i_6	g_7	e_8	i_9	g_{10}	e_{11}
j_9	h_{10}	f_{11}	j_0	h_1	f_2	j_3	h_4	f_5	j_6	h_7	f_8

Após o desentrelaçamento:

a_0	a_1	a_2	a_3	a_4	$\mathbf{a_5}$	a_6	a_7	a_8	a_9	a_{10}	a_{11}
b_0	b_1	$\mathbf{b_2}$	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
c_0	c_1	c_2	c_3	$\mathbf{c_4}$	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
d_0	$\mathbf{d_1}$	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}
e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}
f_0	f_1	f_2	$\mathbf{f_3}$	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}
h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}
i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}
j_0	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	j_{11}

Outros exemplos de erro:

a_0	i_1	g_2	a_3	i_4	g_5	a_6	i_7	g_8	a_9	i_{10}	g_{11}
b_9	j_{10}	h_{11}	b_0	j_1	h_2	b_3	j_4	h_5	b_6	j_7	h_8
c_0	a_1	i_2	c_3	a_4	i_5	c_6	a_7	i_8	c_9	a_{10}	i_{11}
d_9	**b_{10}**	j_{11}	d_0	b_1	j_2	d_3	b_4	j_5	d_6	b_7	**j_8**
e_0	**c_1**	a_2	e_3	c_4	a_5	e_6	c_7	a_8	e_9	c_{10}	**a_{11}**
f_9	d_{10}	b_{11}	f_0	d_1	b_2	f_3	d_4	b_5	f_6	d_7	b_8
g_0	e_1	c_2	g_3	e_4	c_5	g_6	e_7	c_8	g_9	e_{10}	c_{11}
h_9	f_{10}	d_{11}	h_0	f_1	d_2	h_3	f_4	d_5	h_6	f_7	d_8
i_0	g_1	e_2	i_3	g_4	e_5	i_6	g_7	e_8	i_9	g_{10}	e_{11}
j_9	h_{10}	f_{11}	j_0	h_1	f_2	j_3	h_4	f_5	j_6	h_7	f_8

ou

a_0	**i_1**	g_2	a_3	i_4	g_5	a_6	i_7	g_8	a_9	i_{10}	**g_{11}**
b_9	j_{10}	h_{11}	b_0	j_1	h_2	b_3	j_4	h_5	b_6	j_7	h_8
c_0	a_1	i_2	c_3	a_4	i_5	c_6	a_7	i_8	c_9	a_{10}	i_{11}
d_9	b_{10}	j_{11}	d_0	b_1	j_2	d_3	b_4	j_5	d_6	b_7	j_8
e_0	c_1	a_2	e_3	c_4	a_5	e_6	c_7	a_8	e_9	c_{10}	a_{11}
f_9	d_{10}	b_{11}	f_0	d_1	b_2	f_3	d_4	b_5	f_6	d_7	b_8
g_0	e_1	c_2	g_3	e_4	c_5	g_6	e_7	c_8	g_9	e_{10}	c_{11}
h_9	f_{10}	d_{11}	h_0	f_1	d_2	h_3	f_4	d_5	h_6	f_7	d_8
i_0	g_1	e_2	i_3	g_4	e_5	i_6	g_7	e_8	i_9	g_{10}	e_{11}
j_9	**h_{10}**	f_{11}	j_0	h_1	f_2	j_3	h_4	f_5	j_6	h_7	**f_8**

Observa-se que em todos os casos acima o erro é espalhado, após o desentrelaçamento, um para cada linha do código matricial. Portanto, se houver um código de linha que seja capaz de corrigir um erro, o surto de erros bidimensional mostrado no exemplo anterior deverá ser totalmente corrigido. Outra possibilidade é usar o método de decodificação apresentado pelos autores citados [9].

Capítulo 2

Entrelaçamento

2.1 Introdução

Neste capítulo serão abordadas duas formas de utilização de entrelaçamento. A primeira delas se refere à transmissão de dados em canais susceptíveis a surtos de erros (*bursty channel*) e a segunda aplicação refere-se ao armazenamento de dados em estruturas, nas quais eles estão organizados em duas dimensões, como é o caso de sistemas de gravação magnética, que é objeto desta dissertação.

Um entrelaçador é um dispositivo que reordena uma seqüência de símbolos de uma forma determinística. Associado ao entrelaçador está um desentrelaçador, o qual é um dispositivo que converte a seqüência reordenada para a sua forma original. Entrelaçadores e desentrelaçadores têm uma variedade de aplicações em criptografia e em tecnologia de comunicação.

Em muitas das aplicações em tecnologia de comunicação o entrelaçamento é usado conjuntamente com um código corretor de erros aleatórios. Uma técnica, usual para alguns tipos de canais sujeitos a surtos de erros, consiste em inserir um entrelaçador entre o codificador do canal e o próprio canal. O entrelaçador redistribui os símbolos para que aqueles pertencentes a uma mesma palavra-código sejam mutuamente separados por uma distância maior do que o comprimento de um surto de erros típico, a que o canal está sujeito. Conseqüentemente, o entrelaçamento efetivamente faz com que o canal se comporte como um canal sujeito a erros aleatórios.

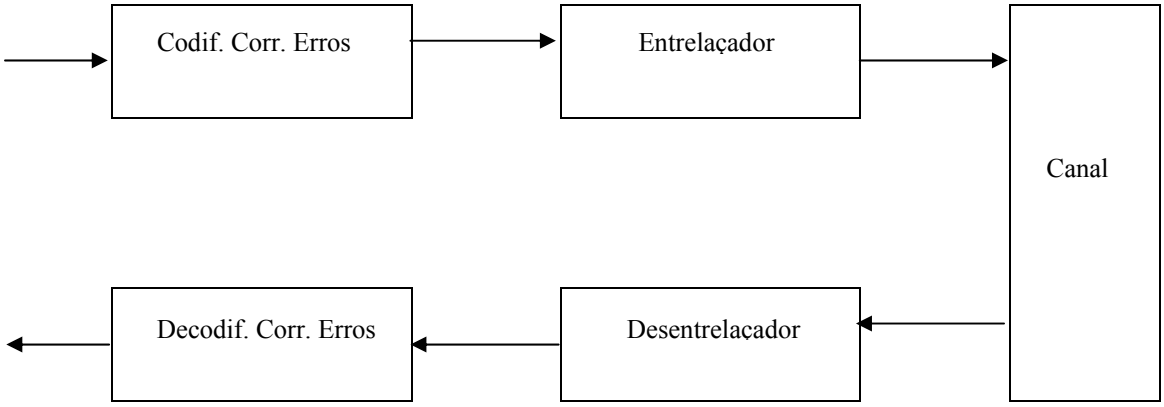


Figura 2.1: Utilização do entrelaçador em sistemas de comunicação.

Para exemplificar a importância do entrelaçador, supõe-se que, por um sistema de transmissão digital similar ao da Figura 2.1, utilizando código corretor de erros de bloco (n, k, d), capaz de corrigir até t erros aleatórios por palavra-código, e que não emprega um entrelaçador, seja enviada uma seqüência de $b.n$ dígitos: $x_1, x_2, x_3, x_4, ,x_n, x_{n+1},.....,x_{bn}$, em que n é o comprimento da palavra-código e b é o grau de entrelaçamento. Esta seqüência se for submetida a um surto de b erros consecutivos poderá produzir padrões com mais de t erros ao longo de uma ou mais palavras-código, que estarão impossibilitadas de serem corrigidas.A possível solução deste tipo de problema é entrelaçar (reordenar) os dígitos, pertencentes às palavras-código, antes de enviá-los pelo canal. Esta ação fará com que o surto de erros seja espalhado entre as palavras-código, permitindo, ao final, que códigos corretores de erros aleatórios sejam capazes de corrigi-lo.

Uma forma simples de entrelaçamento pode ser conseguida, por exemplo, organizando-se as palavras-código nas linhas de um arranjo matricial, para que, em seguida, possam ser enviados, ao longo do canal, os dígitos lidos por colunas, como está ilustrado na Figura 2.2, isto é, os dígitos consecutivos enviados pelo canal pertencem a palavras-código distintas. Portanto é enviada a seqüência $x_1, x_{n+1},...., x_{(b-1)n+1}, x_2, x_{n+2},...,x_{bn}$. Na outra extremidade do canal (receptor ou destino) executa-se o desentrelaçamento, para recuperar a informação original, segundo a mesma regra.

Neste caso um surto de erros de comprimento b será espalhado igualmente pelas palavras-código. Consequentemente, caso o comprimento total do surto de erros seja menor ou igual a $b.t$, cada palavra-código é afetada por no máximo t erros, possibilitando a sua correção por um código corretor de erros aleatórios.

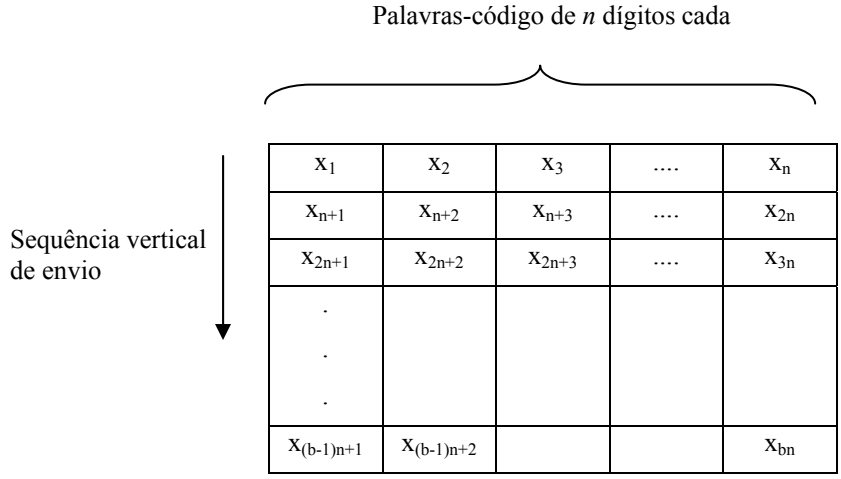


Figura 2.2 – Estratégia de entrelaçamento para canais susceptíveis a surto de erros.

O entrelaçamento também pode ser aplicado em sistemas de transmissão de informação com canais susceptíveis a desvanecimento e/ou sujeitos a interferência intencional (*jamming*).

A técnica de entrelaçamento vista até este ponto sob a ótica temporal, poderá também ser empregada em sistemas com característica espacial ou bidimensional, como é o caso de sistemas de gravação de dados. O objetivo principal do método é mantido, ou seja, tornar mais potentes os códigos corretores de erros aleatórios.

Outros aspectos relevantes a serem considerados, no estudo do par entrelaçador/desentrelaçador, são o retardo de codificação, o qual é definido como o máximo retardo encontrado por qualquer símbolo antes de ser retirado na seqüência de saída, e as capacidades de estocagem E e E_u, os quais representam o número de símbolos estocados nos circuitos do entrelaçador e no desentrelaçador, respectivamente. Estes parâmetros devem ser levados em conta na tomada de decisão sobre qual tipo de entrelaçador/desentrelaçador é mais eficiente para um dado sistema.

2.2 Entrelaçamento Aplicado em Canais de Comunicação Susceptíveis a Erros em Surto

Um canal susceptível a erros em surto é aquele sobre o qual ocorrem feixes ou surtos (*bursts*) de padrões aleatórios de erros. Estes canais usualmente contêm alguns agentes causadores de

erros no meio físico, cuja constante de tempo efetiva excede a taxa de transmissão de símbolo no canal. Por exemplo, um arranhão sobre um disco compacto (*compact disc*) pode obscurecer algumas seqüências de dígitos sobre cada uma das trilhas adjacentes, conseqüentemente causando múltiplos surtos de erros, quando o disco é executado.

Vários códigos corretores de erros de bloco são para correção de erros aleatórios. Um código para correção de erros aleatórios pode corrigir até *t* símbolos errados por palavra-código, independentemente da colocação destes erros. Um problema para estes códigos surge sempre que eles são utilizados em um canal sujeito a surtos de erros. Um surto de erros pode ficar confinado a alguns símbolos pertencentes a um pequeno número de palavras-código recebidas, enquanto que outras palavras-código podem não ser atingidas por qualquer tipo de erro. A capacidade de correção de erros, necessitada para as palavras atingidas, é conseqüentemente desperdiçada nas palavras não afetadas pelo surto de erros.

Um grande esforço tem sido feito para o desenvolvimento de códigos especialmente projetados para canais susceptíveis a surtos de erros. Tais códigos incluem os códigos de Fire [32] e certos códigos cíclicos encurtados [32].Uma forma de tornar efetivo o uso de códigos de correção de erros aleatórios sobre estes tipos de canais, como foi mencionado, é usando as técnicas de entrelaçamento.

Um entrelaçador é um dispositivo que mistura os símbolos das palavras-código para que estes estejam bem separados durante a transmissão. Quando as palavras-código são reconstruídas no destino por um desentrelaçador, os surtos de erros introduzidos pelo canal são separados e espalhados ao longo de algumas palavras-código. O par entrelaçador/desentrelaçador modela efetivamente um canal aleatório.

Os tipos mais frequentes de entrelaçadores, usados para esta aplicação, são de Bloco, Convolucionais ou Helicoidais.

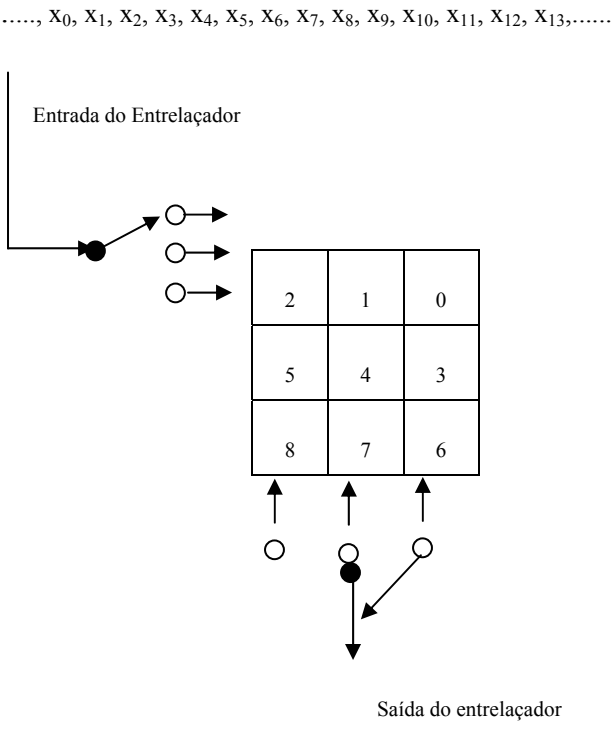
2.2.1 Entrelaçadores de bloco

Muitas das aplicações em correção de erros em sistemas de comunicação, estão relacionadas com os códigos de bloco, logo é compreensível que entrelaçadores de bloco tenham sido

primeiramente adotados. Um exemplo típico de entrelaçamento de bloco é dividir a seqüência de símbolos em blocos e colocá-los em uma correspondente matriz bidimensional, inserindo estes símbolos por linhas e retirando-os por colunas, como foi ilustrado na Figura 2.2.

Um entrelaçador de bloco n_1 x n_2 e o correspondente desentrelaçador são mostrados na Figura 2.3. Os dois circuitos são idênticos, cada um consistindo de n_1 linhas com n_2 elementos de memória (colunas).

A seqüência de dados é colocada nas linhas do entrelaçador na ordem mostrada na Figura 2.3. O conteúdo do entrelaçador então é removido por colunas. Quaisquer dois símbolos adjacentes da entrada são conseqüentemente separados por (n_1-1) outros símbolos na saída. O comprimento da linha n_2 é freqüentemente selecionado para que cada linha retenha uma palavra-código inteira. Um surto de erros de comprimento b causa o máximo de b/n_1 erros em uma ou mais palavras-código, após o desentrelaçamento. Se este código tem capacidade de corrigir padrões de erros aleatórios de peso t ou menos, estas palavras-código entrelaçadas podem ser atingidas por um surto de erros com comprimento não excedendo (n_1t + 1).



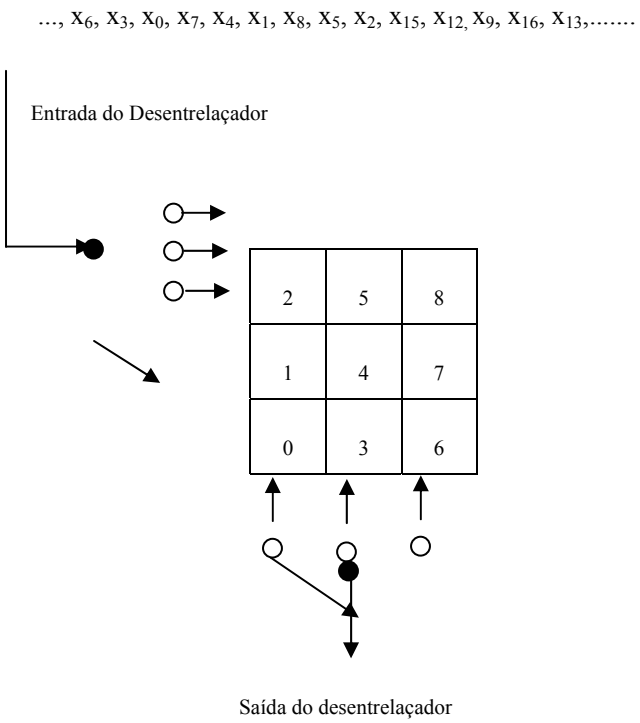


Figura 2.3: Par entrelaçador/desentrelaçador de bloco n_1 x n_2 ($n_1 = n_2 = 3$).

Para detalhar esta idéia, supõe-se um entrelaçador de blocos usual (b,n) consistindo de uma matriz de n linhas e b colunas, em que n representa o comprimento da palavra-código de um código de blocos (n, k, d) e b representa a profundidade de entrelaçamento, l_p. Os símbolos ingressam na matriz pelas colunas e saem desta pelas linhas e é admitido que o símbolo no canto superior esquerdo é o primeiro a entrar e o primeiro a sair. Qualquer surto de erros com comprimento menor do que b erros sobre o canal resultam em erros separados por pelo menos n símbolos na saída do desentrelaçador. Numerando as linhas da matriz de 0 a n-1 e as colunas de 0 a b-1, então um símbolo na posição qualquer (i,j), em que i representa a linha e j a coluna, é submetido a um retardo no entrelaçador de nb+(b-1)i-(n-1)j símbolos, enquanto no desentrelaçador este retardo é de nb + (n-1)j – (b-1)i símbolos.

Os símbolos no topo esquerdo e na parte de baixo à direita são submetidos a um retardo de nb símbolos, tanto no entrelaçador quanto no desentrelaçador (j =0 e i =0), presumindo-se que a matriz seja preenchida antes de qualquer símbolo ser removido.

O retardo mínimo deste tipo de entrelaçador ocorre para o símbolo na posição superior à direita e corresponde a $b + n - 1$. Já no desentrelaçador este mesmo símbolo é submetido a um retardo de $2nb - b - n + 1$, o que perfaz um retardo total de $2nb$.

Com base nas considerações anteriores é possível modelar soluções para que o retardo em ambos, transmissor e receptor, seja reduzido para $b + n - 1$ dando um total de $2(b-1)(n-1)$ símbolos, o que corresponde a duas vezes o valor mínimo estabelecido, a princípio.

Deve-se observar que não é necessário esperar até que toda a matriz esteja preenchida antes de começar o processo de entrelaçamento, e esta simples estratégia permite alcançar menores valores de retardo.

Exemplo: Seja o seguinte entrelaçador de blocos (4,8), isto é, b=4 e n=8

A_0	B_0	C_0	D_0
A_1	B_1	C_1	D_1
A_2	B_2	C_2	D_2
A_3	B_3	C_3	D_3
A_4	B_4	C_4	D_4
A_5	B_5	C_5	D_5
A_6	B_6	C_6	D_6
A_7	B_7	C_7	D_7

Em que A, B, C e D representam quatro palavras-código de comprimento oito. Estas palavras-código são introduzidas por colunas e removidas por linhas, dando como resultado a seguinte seqüência entrelaçada:

Sequência entrelaçada: $A_0,B_0,C_0,D_0,A_1.....A_7,B_7,C_7,D_7$.

Deve-se observar que dois dígitos de uma mesma palavra-código estão separados por três outros dígitos pertencentes a outras palavras-código. Portanto, se houver um surto de erros de comprimento até quatro, por exemplo, este afeta apenas um dígito em cada palavra-código.

A seguir são mostrados alguns valores relativos a retardos para este exemplo. É importante ressaltar que para este exemplo a remoção dos símbolos (dígitos) somente ocorre após a inclusão de todos eles na matriz.

Símbolo	Retardo
A_0	$(b\text{-}1)n + (n\text{-}1) + 1 = nb$
B_0	$(b\text{-}2)n + (n\text{-}1) + 2 = nb - 2n + n + 1 = nb\text{-}n + 1$
C_0	$(b\text{-}3)n + (n\text{-}1) + 3 = nb - 3n + n + 2 = nb - 2n + 2$
D_0	$(n\text{-}1) + b = b + n - 1$
A_1	$(b\text{-}1)n + (n\text{-}2) + b + 1 = nb - n + n + b - 1 = nb + b - 1$
B_1	$(b\text{-}2)n + (n\text{-}2) + b + 2 = nb - 2n + n + b = (n+1)b - n$
C_1	$(b\text{-}3)n + (n\text{-}2) + b + 3 = nb - 3n + n - 2 + b + 3 = (n+1)b - 2n + 1$
D_1	$(n\text{-}2) + 2b = 2b + n - 2$
A_2	$(b\text{-}1)n + (n\text{-}3) + 2b + 1 = nb + 2b - 2$

Algumas formas de implementação deste tipo de entrelaçador/desentrelaçador fazem uso de memórias RAM's.

2.2.2 Entrelaçadores convolucionais

Contrariamente aos entrelaçadores de bloco, os entrelaçadores convolucionais utilizam registradores de deslocamento e linhas de retardo. Os dados trafegam ao longo de uma estrutura contínua com vários estágios. Permanece, no entanto, o cuidado de organizar os pontos de retiradas de dados e a seqüência do comutador para assegurar que todos os símbolos sejam transmitidos com retardo apropriado.

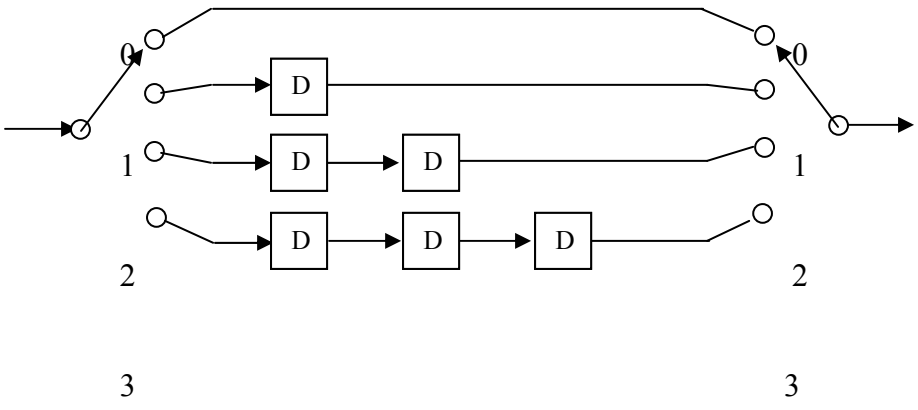
Em algumas aplicações tais como aquelas que lidam com códigos corretores de erros convolucionais, é mais natural considerar entrelaçadores convolucionais, no qual o símbolo é retirado a cada vez que um símbolo é inserido nos estágios de registradores de deslocamento. Estes entrelaçadores constituem uma classe mais geral do que entrelaçadores de bloco, uma vez que qualquer função de um entrelaçador de bloco pode ser realizada por entrelaçadores convolucionais.

Um exemplo básico é mostrado na Figura 2.4. O circuito é caracterizado pelo índice *m*, que corresponde ao número de linhas de retardo. Cada bloco *D* corresponde a uma linha de retardo de *D* símbolos. Os símbolos de entrada são postos nas linhas de retardo na ordem mostrada na Figura. A saída é lida na mesma ordem.

Considerando, por exemplo, um par de símbolos consecutivos na entrada, x_0 e x_1, pode-se observar que estes dois símbolos são postos sobre linhas de retardo adjacentes, uma com retardo *tD* e a outra com retardo (*t*+1)*D*. Quando x_0 atinge a posição de saída da sua linha de retardo, x_1 ainda estará *D* elementos de retardo próximo da saída na sua própria linha. Depois que x_0 é lido, todas as saídas das *m* linhas subseqüentes são lidas *D* vezes antes de x_1 chegar à saída do entrelaçador. Portanto há *mD* símbolos separando símbolos de palavras-código adjacentes na saída do entrelaçador.

Supõe-se que *m* é escolhido para que seja igual ou exceda o comprimento da palavra-código. Cada símbolo em uma palavra-código é consequentemente posto sobre uma diferente linha de retardo. Na saída do entrelaçador, os símbolos da palavra-código aparecerão em ordem, com cada símbolo separado do seu vizinho por *mD* símbolos de outras *mD* palavras-código. Um surto de erros de comprimento *b* pode conseqüentemente causar [*b*/(*mD* +1)] erros em uma ou mais palavras-código. Caso seja empregado um código corretor de erros aleatórios capaz de corrigir *t* erros, a decodificação de erros se torna possível quando o comprimento do surto de erros for igual ou inferior a (*mD* +1)(*t*-1) +

….., x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13},……



….., x_0, x_{-3}, x_{-6}, x_{-9}, x_4, x_1, x_{-2}, x_{-5}, x_8, x_5, x_2, x_{-1}, x_{12}, x_9,……

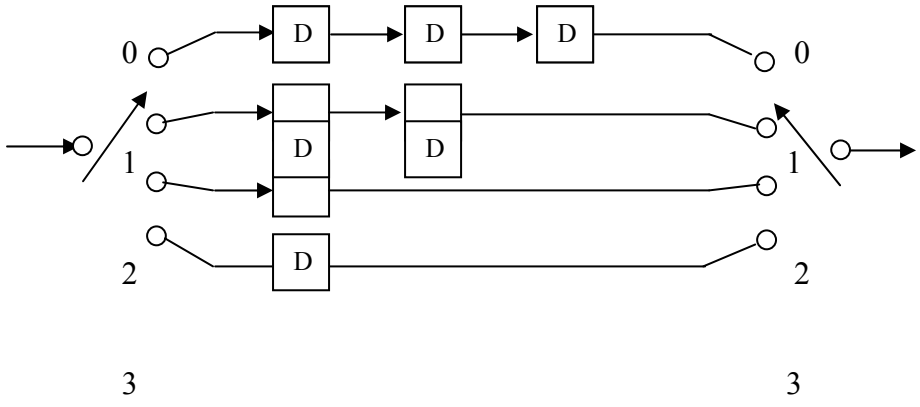


Figura 2.4: Entrelaçador convolucional e seu correspondente desentrelaçador (m = 4 e D =1 símbolo).

De uma forma mais detalhada pode-se dizer que entrelaçadores convolucionais (b,n) são aqueles que reordenam a seqüência de símbolos de entrada a fim de que na sequência de saída a cada b símbolos não existam quaisquer símbolos que eram separados por menos do que n símbolos na seqüência original de entrada, ou seja, não pertencem às mesmas palavras-código.

Nesta terminologia, n reflete o comprimento de uma palavra-código e b o máximo comprimento de surto de erros sobre o canal. Então, para um surto de erros isolado, nenhuma palavra-código na entrada do decodificador é afetada mais do que uma vez pelo surto de erros.

As limitações deste tipo de entrelaçador são originadas pelo retardo de codificação e pela capacidade combinada de estocagem, $E + E_u$, atingível por qualquer entrelaçador do tipo (b,n).

Um entrelaçador é considerado ótimo se atinge mínimo retardo de codificação e mínima capacidade de estocagem. Ramsey [29] mostra quatro tipos simples de implementação de entrelaçadores convolucionais (b,n) e quais os parâmetros a serem considerados para a otimização. Estes aspectos serão tratados em breve.

Inicialmente é importante definir algumas terminologias para entrelaçadores convolucionais. São utilizadas as terminologias adotadas trabalhos de Ramsey [29] e Rocha [27].

Seja .....,x_{z1}, x_{z2},..... uma seqüência de símbolos na saída do entrelaçador, em que z_1, z_2... correspondem as posições destes símbolos na seqüência original de entrada. Para um entrelaçador, portanto

$$| z_i - z_j | \geq b, \tag{2.1a}$$

sempre que

$$| i - j | \leq n - 1. \tag{2.1b}$$

É importante salientar que um desentrelaçador "casado" a um entrelaçador do tipo (b,n) corresponde a um entrelaçador (n,b). Esta afirmação pode ser constatada a partir dos argumentos a seguir.

Seja ....., x_{z1}, x_{z2},.... uma seqüência de símbolos na saída do entrelaçador, a qual corresponde também à entrada de um desentrelaçador, em que.....z_1, z_2,..... correspondem às posições destes símbolos na seqüência original. Seja....., $x_{z'1}$, $x_{z'2}$,..... a seqüência destes símbolos na saída do desentrelaçador. Uma vez que o desentrelaçador restaura a seqüência original então,

$$z'_i = z_i + D' \quad \text{para todo } i, \tag{2.2}$$

em que D' é um retardo fixo introduzido pelo processo entrelaçamento-desentrelaçamento.

Consequentemente

$$| z'_i - z'_j | \geq b, \tag{2.3a}$$

sempre que

$$| i - j | \leq n - 1. \tag{2.3b}$$

Mas as equações 2.3a e 2.3b implicam que se

$$| z'_i - z'_j | \leq b - 1, \tag{2.4a}$$

então

$$| i - j | \geq n. \tag{2.4b}$$

Isto completa a verificação uma vez que (2.4a) e (2.4b) representam as equações de um entrelaçador (n,b).

O retardo introduzido pelo entrelaçador e o introduzido pelo desentrelaçador são ambos iguais ao introduzido pela operação geral de entrelaçamento-desentrelaçamento.

Para considerar o retardo de um entrelaçador, primeiramente será assumido sem perda de generalidade que $min(i - z_i) = 0$ sendo que, como notado, uma vez que o entrelaçador é dito realizável, $i \geq z_i$. O máximo retardo experimentado por um símbolo dentro do entrelaçador é $a = max(j - z_j)$. Uma vez que a seqüência de saída do desentrelaçador é uma versão atrasada da seqüência de entrada do entrelaçador, supondo-se por D símbolos, a soma do retardo de um símbolo no entrelaçador e desentrelaçador é D.

Uma vez que é assumido que existe pelo menos um símbolo com retardo de zero ao longo do entrelaçador, e similarmente do desentrelaçador, o retardo geral D é no máximo a. Se o retardo geral é a então quando o símbolo x_i aparece na saída do desentrelaçador, símbolos x_{i+1}, x_{i+2},...., x_{i+a} devem ter ingressado no entrelaçador. Daí vem o fato de que o armazenamento combinado do entrelaçador/desentrelaçador é pelo menos a.

Pode-se mostrar que o máximo retardo de um entrelaçador (b,n) é pelo menos (b-1)(n-1) e para que isto seja comprovado basta considerar a locação de n símbolos da entrada: $x_i x_{i+1}.....x_{i+n-1}$. Por definição devem haver pelo menos b-1 símbolos, dentre estes símbolos, na seqüência de saída. A dimensão total destes símbolos reunidos na seqüência de saída é consequentemente pelo menos (n-1)b+1, ou seja, (n-1) (b-1)+n= (n-1)b+1. No pior caso, o último símbolo na seqüência de entrada é também o último símbolo na seqüência de saída e o máximo retardo deve ser consequentemente pelo menos (n-1)b – (n-1) = (n-1)(b-1). É também observado que o retardo médio de um entrelaçador (b,n) é pelo menos (n-1)(b-1)/2.

O período de um entrelaçador convolucional é o comprimento mínimo da seqüência para a qual o padrão inteiro do retardo se repete. É notado que o período de um entrelaçador (b,n) é pelo menos $min(b,n)$.

A profundidade de um entrelaçador, l_p, é definida como o maior comprimento de surto de erros que não pode atingir qualquer palavra-código duas vezes, sendo assumido que os símbolos de entrada do entrelaçador são divididos em palavras-código.

Exemplo: Seja um entrelaçador (b,n) do tipo (5,3), ou seja, para uma seqüência de três símbolos na entrada existe uma separação entre os pares por pelo menos quatro símbolos (equivalente a b-1) na seqüência de saída, ou seja:

Seqüência de entrada: 0, 1, 2,....
Seqüência de saída: 0 x x x x 1 x x x x 2,.... em que x representa símbolos que intercalam a seqüência de entrada.

Deve-se observar que a dimensão da seqüência da saída passa a ser (n-1)b + 1=11, o máximo retardo sofrido pelo símbolo 2 é igual a oito (o qual corresponde a (n-1)(b-1)) e que o retardo médio é igual a quatro (o qual corresponde a (n-1)(b-1)/2).

Um entrelaçador convolucional é definido como uniforme se não há um conjunto de (b+1) símbolos na seqüência de saída para o qual todo o par de símbolos é separado por pelo menos n símbolos na seqüência de entrada. Caso contrário, o entrelaçador é definido como não-uniforme. Pode ser mostrado que o retardo de codificação de um entrelaçador uniforme (b,n) é pelo menos (b-1)(n+1) e para um não-uniforme é pelo menos b(n+1).

Em seguida serão mostrados alguns tipos de entrelaçadores convolucionais mostrados nos trabalhos de Ramsey [29] e Rocha[27]. Os tipos mostrados em [29] foram designados como sendo do tipo I, tipo II, tipo III e tipo IV, enquanto que em [27] é abordado o tipo de Forney [33] [34], como podem ser vistos a seguir.

2.2.2.1 Tipo I: Entrelaçador (b,n)

Sempre que n e b + 1 são primos entre si e $n > b$+1, o dispositivo mostrado na Figura 2.5 é um entrelaçador (b,n) não-uniforme. Este dispositivo consiste de um conjunto de [b(n-1) + 1]

estágios de registradores de deslocamento, com pontos de retirada de dados (*taps*) em todo (*n*-1)-ésimo estágio intermediário, e de um comutador de (*b*+1) posições que ciclicamente amostra os *b*+1 pontos em ordem reversa de suas distâncias da entrada do registrador de deslocamento. O retardo de codificação é de *b*(*n*-1).

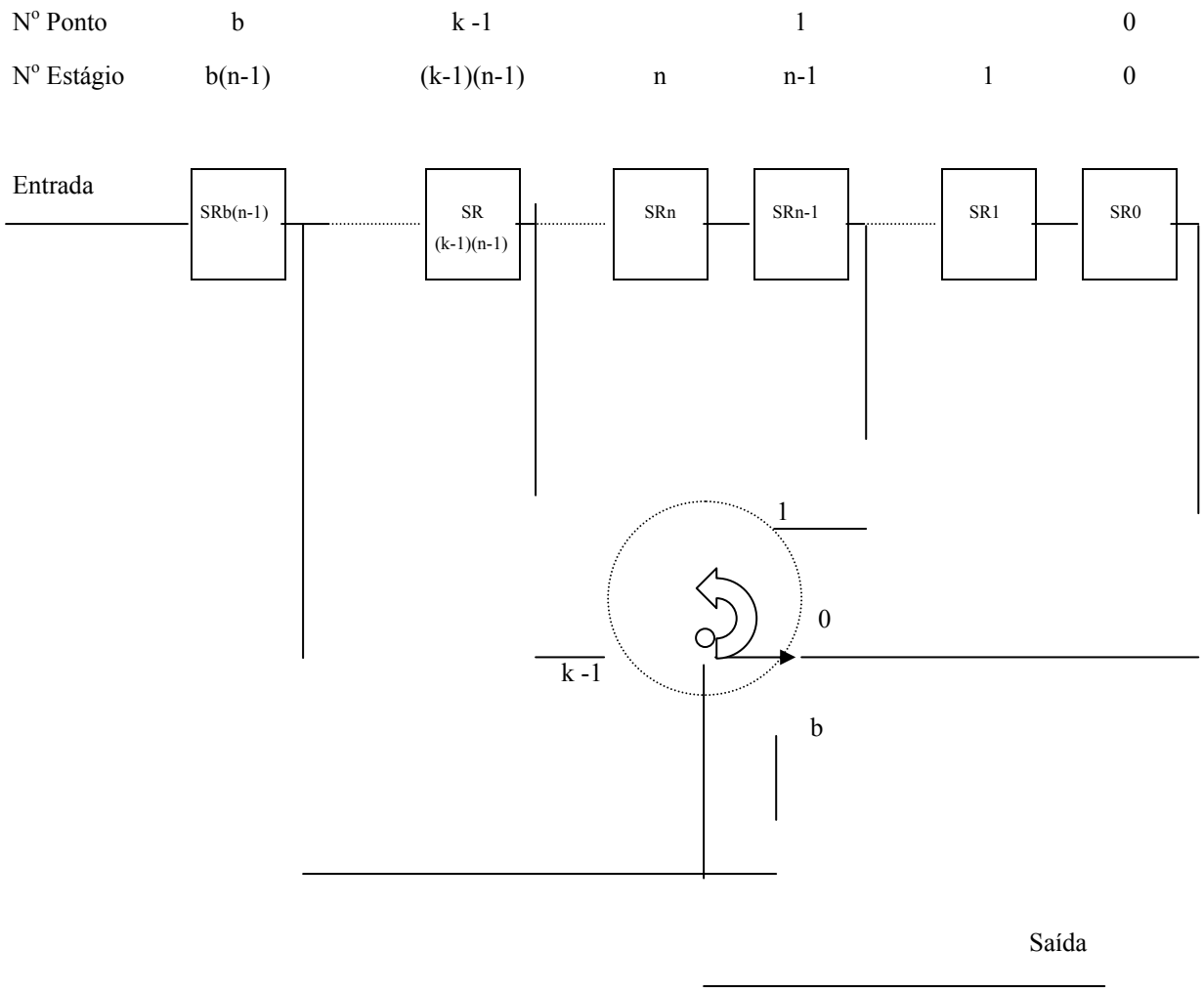


Figura 2.5: Entrelaçador Tipo I.

Neste trabalho de Ramsey[29] está demonstrado que este tipo e os demais reunem as duas características principais que garantem a sua funcionalidade: Primeiro, que nenhuma seqüência contínua de *b* símbolos na saída do entrelaçador contenha quaisquer símbolos que estejam separados por menos do que *n* símbolos na seqüência de entrada; e segundo, que cada símbolo na seqüência de entrada irá aparecer na seqüência de saída.

A primeira afirmação assegura a separação entre símbolos e a segunda afirmação é importante para mostrar que o dispositivo faz mapeamento unívoco entre as seqüências de entrada e de saída.

Exemplo deste entrelaçador convolucional tipo I:

Seja n = 5 e b =2 (Implica que numa seqüência reordenada na saída, dois símbolos quaisquer estão separados por pelo menos cinco símbolos na seqüência original).

Total de estágios de registradores de deslocamento (*shift register*) = [b(n-1) + 1] = 9.
Total de posições do comutador = b +1 = 3.
Total de estágios entre pontos = (n-1) = 4 (à exceção do final que tem um único estágio).

Logo,

SR8	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}	x_{-6}	x_{-7}	x_{-8}	x_{-9}
x_0	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}	x_{-6}	x_{-7}	x_{-8}
x_1	X_0	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}	x_{-6}	x_{-7}
x_2	X_1	x_0	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}	x_{-6}
x_3	X_2	x_1	x_0	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}
x_4	X_3	x_2	x_1	x_0	x_{-1}	x_{-2}	x_{-3}	x_{-4}
x_5	X_4	x_3	x_2	x_1	x_0	x_{-1}	x_{-2}	x_{-3}
x_6	X_5	x_4	x_3	x_2	x_1	x_0	x_{-1}	x_{-2}
x_7	X_6	x_5	x_4	x_3	x_2	x_1	x_0	x_{-1}
x_8	X_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
x_9	X_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1
x_{10}	X_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2
x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3
x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4
x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5
x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6
x_{15}	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7
x_{16}	x_{15}	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8
x_{17}	x_{16}	x_{15}	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9
x_{18}	x_{17}	x_{16}	x_{15}	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}
x_{19}	x_{18}	x_{17}	x_{16}	x_{15}	x_{14}	x_{13}	x_{12}	x_{11}

SR0, SR1,...e SR8 indicam os símbolos existentes nos registradores de deslocamento.

Observa-se que os símbolos são introduzidos a partir de SR8. As áreas hachuradas (em tom azulado) representam as posições instantâneas do comutador. Neste momento os símbolos são retirados do entrelaçador.

Seqüência de Entrada:

$.....,x_0,x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19},.....$

Seqüência de Saída:....., $x_{-4}, x_1, x_{-6}, x_{-1}, x_4, x_{-3}, x_2, x_7, x_0, x_5, x_{10}, x_3, x_8, x_{13}, x_6, x_{11}, x_{16}, x_9, x_{14}, x_{19}, x_{12}, x_{17}, x_{22}, x_{15}, x_{20}, x_{25}, x_{18}, x_{23}, x_{28}, x_{21}, x_{26}, x_{31}, x_{24}, x_{29}, x_{34}, x_{27},.....$

Observa-se que, em grupo de dois elementos adjacentes na saída, existem dois símbolos distanciados entre si por no mínimo cinco símbolos na seqüência de entrada, daí a designação de entrelaçador (2,5).

Exemplo do desentrelaçador para o entrelaçador anterior:

Seqüência de Entrada: 0, 0, x_1, 0, 0, x_4, 0, x_2, x_7, x_0, x_5, x_{10}, x_3, x_8, x_{13}, x_6, x_{11}, x_{16}, x_9, x_{14}, x_{19}, x_{12}, x_{17}, x_{22}, x_{15}, x_{20}, x_{25}, x_{18}, x_{23}, x_{28}, x_{21}, x_{26}, x_{31}, x_{24}, x_{29}, x_{34}, x_{27},.....
(Obs.:Considera-se todos os símbolos anteriores a x_0 como sendo nulos).

SR8	SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
x_1	0	0	0	0	0	0	0	0
0	x_1	0	0	0	0	0	0	0
0	0	x_1	0	0	0	0	0	0
x_4	0	0	x_1	0	0	0	0	0
0	x_4	0	0	x_1	0	0	0	0
0	0	x_4	0	x_2	x_1	0	0	0
x_7	0	0	x_4	0	x_2	x_1	0	0
0	x_7	0	0	x_4	0	x_2	x_1	x_0
0	0	x_7	0	x_5	x_4	0	x_2	x_1
x_{10}	0	0	x_7	0	x_5	x_4	0	x_2
0	x_{10}	0	0	x_7	0	x_5	x_4	x_3
0	0	x_{10}	0	x_8	x_7	0	x_5	x_4

x_{13}	0	0	x_{10}	0	x_8	x_7	0	x_5
0	x_{13}	0	0	x_{10}	0	x_8	x_7	x_6
0	0	x_{13}	0	x_{11}	x_{10}	0	x_8	x_7
x_{16}	0	0	x_{13}	0	x_{11}	x_{10}	0	x_8
0	x_{16}	0	0	x_{13}	0	x_{11}	x_{10}	x_9
0	0	x_{16}	0	x_{14}	x_{13}	0	x_{11}	x_{10}
x_{19}	0	0	x_{16}	0	x_{14}	x_{13}	0	x_{11}
0	x_{19}	0	0	x_{16}	0	x_{14}	x_{13}	x_{12}
0	0	x_{19}	0	x_{17}	x_{16}	0	x_{14}	x_{13}
x_{22}	0	0	x_{19}	0	x_{17}	x_{16}	0	x_{14}
0	x_{22}	0	0	x_{19}	0	x_{17}	x_{16}	x_{15}
0	0	x_{22}	0	x_{20}	x_{19}	0	x_{17}	x_{16}
x_{25}	0	0	x_{22}	0	x_{20}	x_{19}	0	x_{17}
0	x_{25}	0	0	x_{22}	0	x_{20}	x_{19}	x_{18}
0	0	x_{25}	0	x_{23}	x_{22}	0	x_{20}	x_{19}
x_{28}	0	0	x_{25}	0	x_{23}	x_{22}	0	x_{20}
0	x_{28}	0	0	x_{25}	0	x_{23}	x_{22}	x_{21}
0	0	x_{28}	0	x_{26}	x_{25}	0	x_{23}	x_{22}
x_{31}	0	0	x_{28}	0	x_{26}	x_{25}	0	x_{23}

As áreas hachuradas representam as posições de entrada de dados no desentrelaçador. Observa-se que a última coluna (que corresponde ao dado no registrador de deslocamento SR0) representa a saída do desentrelaçador.

Deve-se observar que para este exemplo o retardo total sofrido por qualquer símbolo, somando os retardos de entrelaçamento e de desentrelaçamento, é igual a oito (o qual corresponde a $b(n\text{-}1)$).

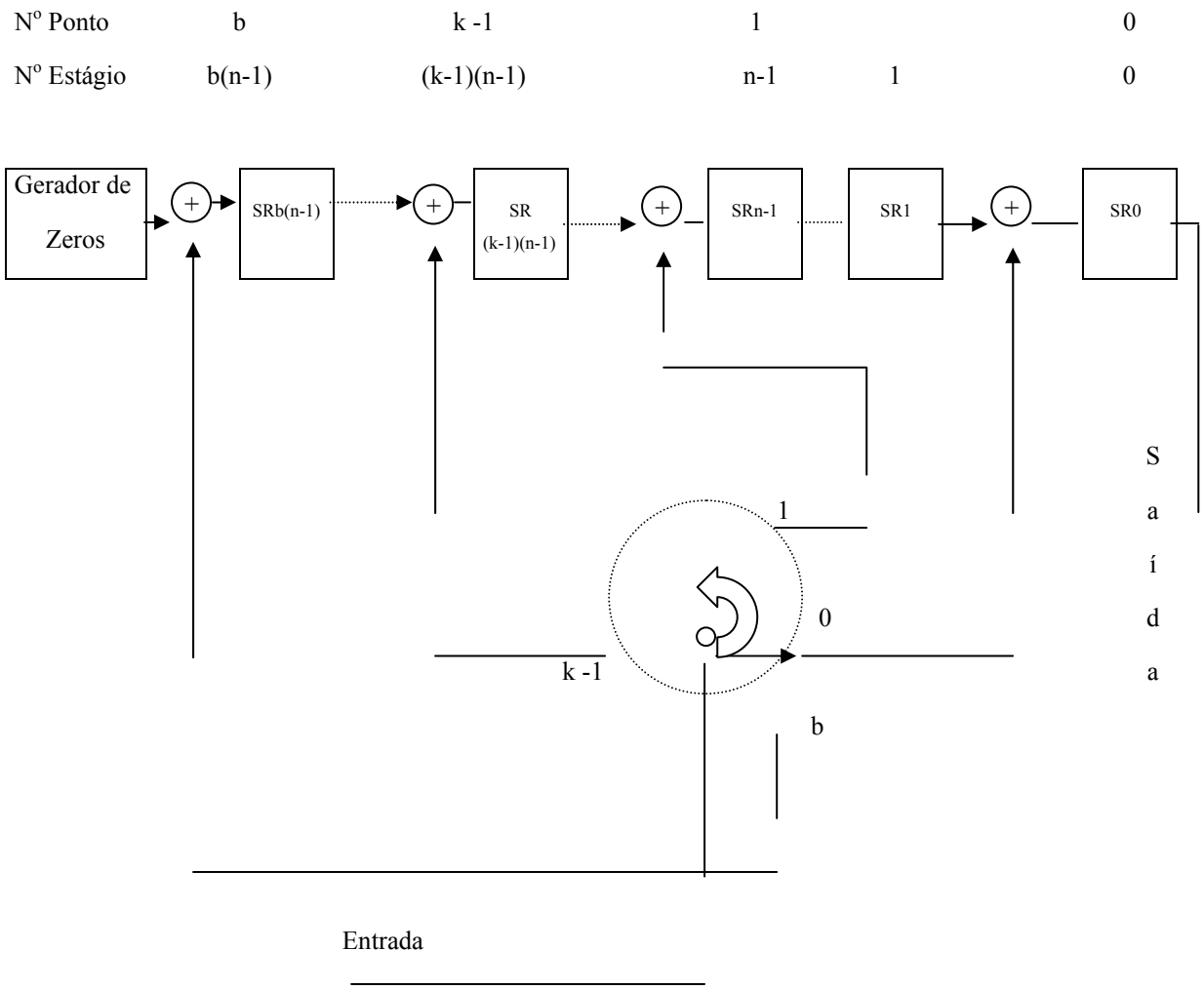


Figura 2.6: Desentrelaçador do Tipo I.

2.2.2.2 Tipo II: Entrelaçador (*b*,*n*)

Primeiramente deve-se lembrar que o dispositivo desentrelaçador para um entrelaçador (*b*,*n*) é na verdade um entrelaçador (*n*,*b*). Usando este fato, vê-se que sempre que *b* e *n* +1 são relativamente primos entre si e $b > n+1$, um entrelaçador (*b*,*n*) pode ser realizado por um dispositivo usando registrador de deslocamento de [*n*(*b*-1) +1] estágios com pontos a cada (*b*-1) estágios intermediários e um comutador com *n* +1 posições, como mostrado na Figura 2.6.

2.2.2.3 Tipo III: Entrelaçador (*n*, *b*)

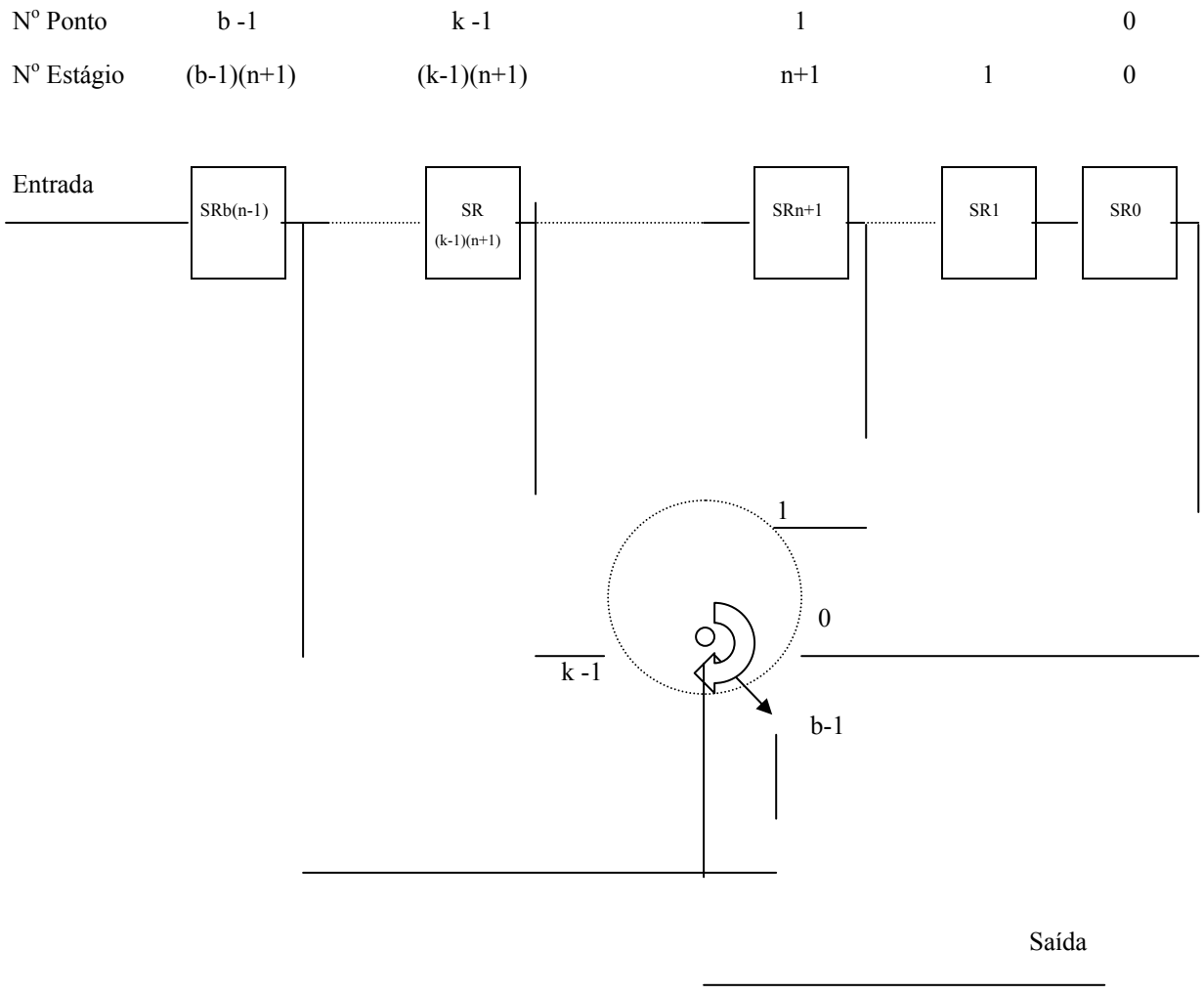


Figura 2.7: Entrelaçador Tipo III.

2.2.2.4 Tipo IV: Entrelaçador (*n*,*b*)

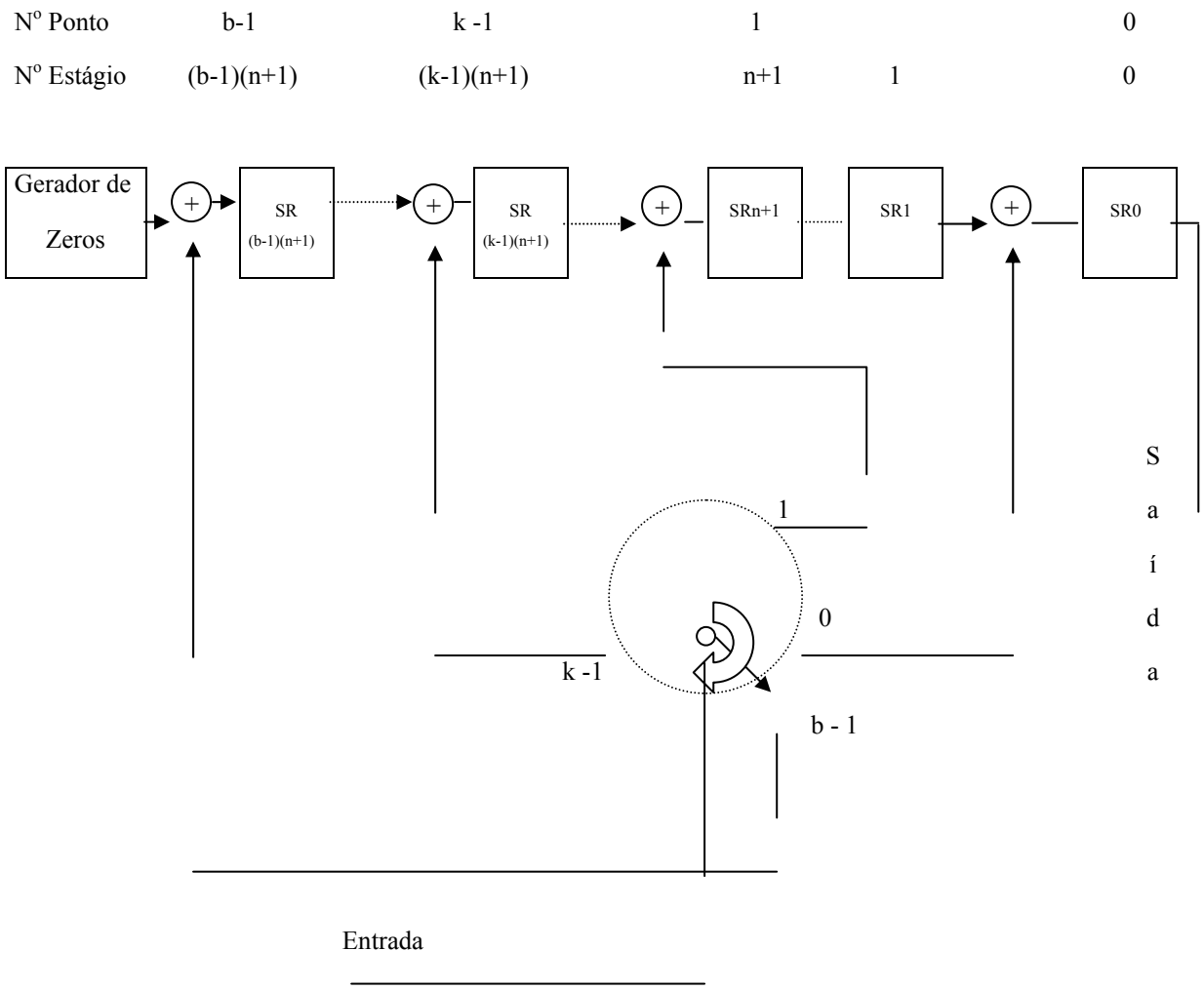


Figura 2.8: Entrelaçador Tipo IV.

Ramsey [29] também mostra em seu trabalho uma Tabela contendo algumas condições que otimizam estes entrelaçadores propostos. Esta Tabela é reproduzida a seguir.

Tipo	Retardo de Codificação (capacidade de armazenamento combinado)	Faixa de Optimização	Restrições
I	$b(n\text{-}1)$	$b < n < 2b$	n e b+1 rel. primos
II	$n(b\text{-}1)$	$n < b < 2n$	b e n+1 rel. primos
III	$(b\text{-}1)(n\text{+}1)$	$n \geq 2b$	n e b rel. primos
IV	$(n\text{-}1)(b\text{+}1)$	$b \geq 2n$	n e b rel. primos

Tabela 2.1: Sumário de parâmetros otimizados para os tipos de entrelaçadores propostos em [29].

Um par entrelaçador/desentrelaçador é dito otimizado se atinge simultaneamente o mínimo tempo do retardo de codificação e a mínima capacidade de armazenamento. Entrelaçadores ótimos são atingidos se as condições mostradas na Tabela 2.1 são conseguidas para todos os pares n, b que satisfazem certas condições de primos relativos. A implementação deste entrelaçadores é feita usando registradores de deslocamento do tipo longo. O trabalho de Forney [33] [34] realiza o entrelaçamento a partir de registradores mais curtos e de comutadores.

Numa versão mais simples do esquema de Forney os símbolos são primeiro divididos em blocos com comprimento L, como se fosse uma conversão serial para paralelo. O i-ésimo símbolo de cada bloco é inserido em um registrador de deslocamento de comprimento iD, i= 0,1,..., L-1. As saídas dos L registradores de deslocamento são amostradas sequencialmente (conversão paralela-serial) e transmitidas pelo o canal. A estrutura do desentrelaçador é similar, exceto que o i-ésimo elemento de cada bloco entra em um registrador de deslocamento de comprimento $(L\text{-}1\text{-}i)D$', i = 0,1,..., L-1. Cada símbolo recebe um retardo total de $(L\text{-}1)D$ unidades de tempo (cada unidade de tempo corresponde a L símbolos) ou $(L\text{-}1)DL$ símbolos e o total, somando-se entrelaçamento e desentrelaçamento, é igual a $(L\text{-}1)DL$. Baseado na terminologia anterior identifica-se n (comprimento da palavra-código) como sendo DL e b (comprimento do surto de erros) como L. Este modelo portanto refere-se a um entrelaçador (b,n).

Exemplo:

D =5 e L = 5.

E_0	D_0	C_0	B_0	A_0	=>				=>	O_0
E_1	D_1	C_1	B_1	A_1	=>			*D*	=>	O_1
E_2	D_2	C_2	B_2	A_2	=>		*D*	*D*	=>	O_2
E_3	D_3	C_3	B_3	A_3	=>	*D*	*D*	*D*	=>	O_3
E_4	D_4	C_4	B_4	A_4	=>	*D*	*D*	*D*	=>	O_4

Seqüência de Entrada: $A_0A_1A_2A_3A_4B_0B_1$.....$E_0\,E_1\,E_2\,E_3\,E_4$.....

Seqüência de Saída: $A_0\,0\,0\,0\,0\,B_0\,A_1\,0\,0\,0\,C_0\,B_1\,A_2\,0\,0\,D_0\,C_1\,B_2\,A_3\,0\,E_0\,D_1\,C_2\,B_3\,A_4$.....

Observa-se que até aparecerem na saída todos os símbolos pertencentes à palavra *A* (A_0 a A_4) a seqüência de saída do entrelaçador exibirá um total de 25 símbolos e também que entre cada símbolo pertencente à palavra *A* existem cinco símbolos de outras palavras, consequentemente n = 25 e b=5. Para o desentrelaçamento basta fazer o inverso, como mostrado a seguir.

E_0	D_0	C_0	B_0	A_0	=>	*D*	*D*	*D*	*D*	=>	O_0
D_1	C_1	B_1	A_1	0	=>		*D*	*D*	*D*	=>	O_1
C_2	B_2	A_2	0	0	=>			*D*	*D*	=>	O_2
B_3	A_3	0	0	0	=>				*D*	=>	O_3
A_4	0	0	0	0	=>					=>	O_4

2.2.3 Entrelaçadores helicoidais

São muito similares aos entrelaçadores de bloco com sucessivas colunas deslocadas entre si por um único símbolo. A Figura 2.9 a seguir ilustra esta técnica.

*		*
A	c	d
B	C	e
f	D	E
a	b	F
A	c	d
B	C	e
f	D	E
a	b	F
*	*	*

Figura 2.9: Entrelaçador helicoidal (n = 4).

Nesta ilustração, os símbolos são inseridos por colunas de comprimento quatro (neste caso o tamanho da palavra-código) e retirados por linhas. Em geral as colunas serão de comprimento n e o número de colunas será n-1. Consequentemente não há a flexibilidade de escolha do comprimento do entrelaçador helicoidal. As letras maiúsculas e minúsculas representam a mesma localização física de memória. A Figura 2.10 demonstra a ordem dos símbolos entrando e saindo do entrelaçador /desentrelaçador

Entrada no Entrelaçador	A	B	F	a	C	D	b	c	E	F	d	e
Canal	a	b	F	A	c	d	B	C	e	f	D	E
Saída do Desentrelaçador	A	B	F	a	C	D	b	c	E	F	d	e

Figura 2.10: Entrada/saída do entrelaçador helicoidal, n=4, período=12 e retardo geral=12.

A simetria inerente ao entrelaçador helicoidal está clara e o tamanho da RAM requerida para o entrelaçador helicoidal de comprimento n é de $n(n\text{-}1)/2$, o mínimo retardo é de n-1, o máximo retardo é de $(n\text{-}1)^2$ e o retardo total de entrelaçamento é de $n(n\text{-}1)$. Este tipo é considerado como sendo um entrelaçador (n-1, n).

2.3. Entrelaçamento Aplicado a Sistemas de Armazenamento de Dados

2.3.1 Introdução

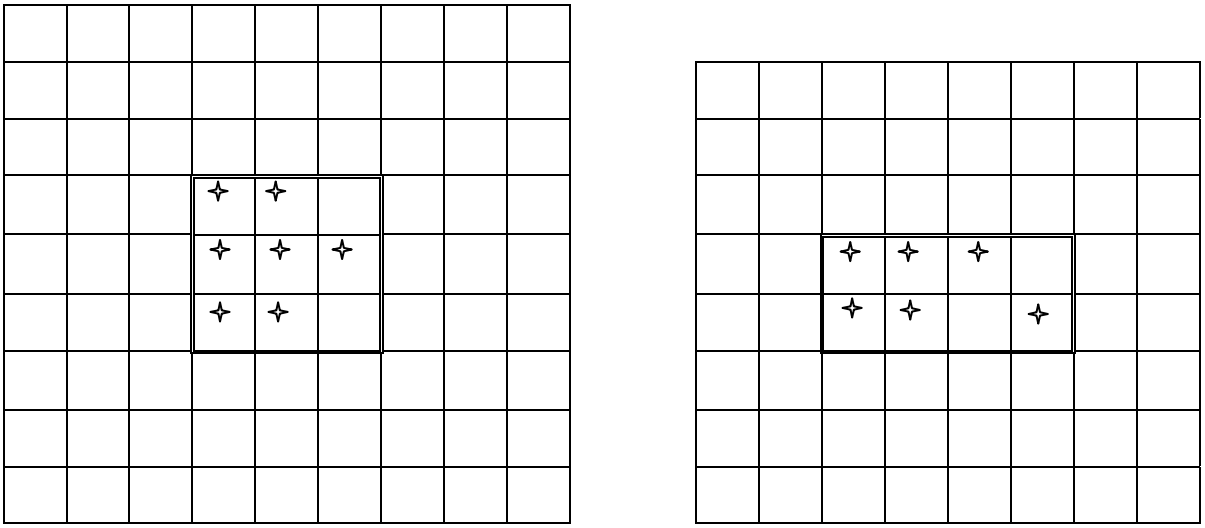
O entrelaçamento, como anteriormente mencionado, é um procedimento bastante disseminado para o combate aos efeitos de surtos de erros em sistemas digitais para transmissão e/ou armazenamento de informação. Um caso de interesse prático, por exemplo, ocorre em gravação magnética, para a qual as trilhas podem ser afetadas por surtos de erros de duas dimensões (*clusters* ou *patches*). Os problemas inerentes aos esquemas de entrelaçamento para tais aplicações foram considerados no trabalho de Farrell e Blaum [9], que foi descrito na Seção 1.2.3. Além deste, podem ser citados também os trabalhos de Kauffman[25] e de Almeida[26], que são complementares, abordando uma matriz de Q x Q dígitos, em que cada linha corresponde a uma palavra-código de um código linear binário com capacidade de correção de erro aleatório t=1. Também o caso t=2 foi considerado. Como o trabalho de Farrel e Blaum [9] já mereceu toda a atenção na Seção 1.2.3 desta dissertação, passa-se a descrever a seguir alguns aspectos relevantes dos trabalhos de Kauffman[25] e de Almeida[26], que dão uma noção sobre a técnica empregada.

2.3.2 Descrição do formato

Considera-se C (n,k,d) um código linear de bloco com comprimento n, dimensão k, distância mínima de Hamming d e capacidade de correção de t erros aleatórios, em que $d = 2t+1$. Para o propósito de correção de surtos de erros bidimensionais, um subconjunto de Q palavras-código de C, sendo $Q = n$, é escrito como uma matriz Z de Q x Q dígitos. O caso $t = 1$ é considerado, para que o esquema seja projetado para corrigir surtos de erros bidimensionais (*cluster*) com peso não maior que Q. Uma mancha de erros $b_i x b_j$ de peso Q, a qual será escrita como B_w, é dita ser um surto de erros bidimensional que corrompe os elementos de Z, afetando ε dos seus dígitos, em que $\varepsilon = w \leq Q$. Para tal surto de erros, pode-se assumir várias formas geométricas que podem ser descritas como:

i – Surto de erros com forma quadrada – Neste caso a forma dos dígitos corrompidos forma uma submatriz quadrada de Z (denotada como Z_q), de dimensões r x s, sendo $r = s$, com $Q = rs = r^2$ elementos, ou que podem ser minimamente cobertos por Z_q (Figura 2.11 (a)).

ii- Surto de erros com forma retangular – Neste caso, os dígitos corrompidos formam tanto uma submatriz não-quadrada de Z (denotada por Z_r), de dimensões r x s, em que $Q = rs$, ou que podem ser minimamente cobertos por Z_r (fig 2.11 (b))



(a) (b)

Figura 2.11: (a) Uma forma quadrada (Q=9, r=3, ε=7); (b) Uma forma retangular (Q=8, r=2, s=4, ε=6).

É importante observar que embora o caso (i) possa ser obtido diretamente do caso (ii), é preferível considerá-los separadamente. Em ambos os casos, contudo, as posições afetadas envolvem linhas e colunas cujos valores são elementos dos conjuntos de inteiros (0,1,2,.....,Q-1), para que as operações possam ser efetuadas módulo Q. A posição do surto de erros bidimensional dentro da matriz é irrelevante.

2.3.3 Matrizes de transformação ótimas

Para construir famílias de transformação cuja ação sobre um surto de erros B_w é espalhá-lo de tal modo que se torna uma série de Q padrões de erros simples, será usada aritmética modular.

Este entrelaçamento bidimensional (*2-D*) é obtido por mapeamento de um elemento em uma linha *i* e coluna *j*, (*i*,*j*), para um elemento (*i'*,*j'*), da forma

$(i,j) \rightarrow (i',j') = T((i,j))$, em que T é uma matriz de transformação linear, aqui definida pelos parâmetros e, f, g e h,

$$T = \begin{bmatrix} e & f \\ g & h \end{bmatrix}. \tag{2.5}$$

Lemma 2.1 – Uma condição necessária e suficiente para T ser uma transformação inversível é que MDC[$(eh\text{-}fg)$, Q] =1.

Prova – Em forma de matriz

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix}. \tag{2.6}$$

Para que a matriz de transformação inversa, T^{-1}, possa ser definida, deve-se garantir a existência de um inverso multiplicativo de Δ (mod Q), em que $\Delta = eh - fg$, isto é, a congruência $(eh - fg)\ \Delta^{-1} \equiv 1$ (mod Q) deve ter uma única solução. A condição necessária e suficiente para isto é que

$$\text{MDC}\ [(eh\text{-}fg), Q] = 1. \tag{2.7}$$

Teorema 2.1: Os parâmetros de transformação $e = h = 1$, $f = \pm\, r$ e $g = s$ definem um mapeamento de elementos de qualquer submatriz Z_r de Z.

Prova: Considerando que MDC $[(eh\text{-}fg),Q]$ = MDC $[(1 \pm rs),rs] = 1$, T é inversível. Por simplicidade considera-se Z_r localizado em Z como mostrado na Figura 2.12.

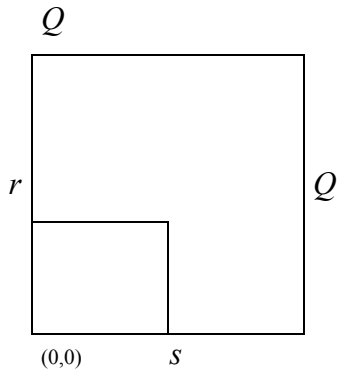


Figura 2.12: Uma submatriz Z_r (r x s) em Z.

Sem perda de generalidade, pode-se dizer que se $(i+x) \equiv (i+y) \pmod{r}$, com $0 \leq x, y \leq r-1$, então $x \equiv y \pmod{r}$, o que implica $x = y$. Sabe-se também que um elemento (i,j) em Z, $0 \leq i \leq r-1$, $0 \leq j \leq s-1$, pode ser mapeado em uma linha c, em que $c = ie + jf \pmod{Q}$, ou $c = i \pm jr \pmod{Q}$. De uma maneira similar, para um elemento (i_1,j_1), tem-se $c' = i_1 \pm j_1 r \pmod{Q}$. Se $c = c' \pmod{Q}$, então,

$$i \pm jr \equiv i_1 \pm j_1 r \pmod{rs}. \tag{2.8}$$

Uma vez que $i \pm jr \equiv i_1 \pm j_1 r \pmod{r}$, então $i \equiv i_1 \pmod{r}$ ou $i = i_1$. Consequentemente (2.8) pode ser reescrita como:

$$jr \equiv j_1 r \pmod{rs}. \tag{2.9}$$

Logo $j \equiv j_1 \pmod{s}$ e $j = j_1$. Consequentemente, se $c \equiv c' \pmod{Q}$, os elementos (i, j) e (i_1, j_1) são os mesmos.

O caso de uma matriz B_w com formato quadrado pode ser abordado da mesma forma. Contudo uma importante diferença é o fato que agora é possível obter o que se chama "entrelaçamento perfeito", em que o mapeamento existe não somente dentro das linhas de A, mas também nas colunas.

Teorema 2.2 – A transformação de parâmetros $e=h=1$, g, $f = \pm o$, define um mapeamento dos elementos de qualquer submatriz quadrada Z_q de Z para linhas e colunas de Z.

Prova: Como pode ser verificado, T é inversível. A localização de Z_q em Z e a prova do mapeamento das linhas seguem os mesmos passos do Teorema 2.1. Para o mapeamento das colunas tem-se

$$u = \pm\, i\, o + j \pmod{Q} \tag{2.10}$$

e

$$u' = \pm\, i_1\, o + j_1 \pmod{Q}, \tag{2.11}$$

com $Q = o^2$ e $0 \le i, j, i_1, j_1 \le o\text{-}1$.

Se $u \equiv u'$ (mod Q), então

$$\pm\, io + j \equiv \pm\, i_1\, o + j_1 \pmod{o^2}, \tag{2.12}$$

o que implica em $\pm\, io + j \equiv \pm\, i_1\, o + j_1$ (mod o), para que $j \equiv j_1$ (mod o) e $j = j_1$. De (2.12), $\pm\, io \equiv \pm\, i_1 o$ (mod o^2), implicando em $i \equiv i_1$ (mod o) e $i = i_1$. Consequentemente $c \equiv c'$ (mod Q) e $u \equiv u'$ (mod Q) se e somente se $i \equiv i_1$ (mod Q) e $j \equiv j_1$(mod Q), o que significa entrelaçamento perfeito dos Q dígitos de Z_q.

Entrelaçamento perfeito, neste caso, significa que se as colunas de Z são palavras-código de um código corretor de único erro, então elas podem ser usadas para corrigir B_w.

2.3.4 Utilização de códigos corretores de erros com capacidade de correção de t erros ($t > 1$)

A generalização do uso do esquema com códigos lineares de bloco com capacidade de correção de t erros é obtida considerando transformações sucessivas atuando sobre surtos de erros bidimensionais tendo peso menor ou igual a tQ. A idéia básica é considerar a transformação descrita com aritmética modular tQ, como se estivesse operando com uma matriz estendida com dimensões (tQ)x(tQ).

Conseqüentemente com $tQ = o^2$, os mapeamentos descritos pelo Teorema 2.2 podem ser aplicados para produzir um entrelaçamento perfeito de B_w, para $\varepsilon = w \leq tQ$. Após isto, é necessário reduzir módulo Q todos os valores de linhas e colunas que são maiores do que Q-1. O resultado será de no máximo Q palavras-código corrompidas com padrões de erros de peso não maior do que t, que podem então ser corrigidas pelo código corretor de t erros.

Exemplos desta técnica:

Seja $Q = 6$, $r = 3$ e $s = 2$, logo T é do tipo

$$T = \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix}, \text{ então}$$

Matriz sem entrelaçamento:

A_0	A_1	A_2	A_3	A_4	A_5
B_0	B_1	B_2	B_3	B_4	B_5
C_0	C_1	C_2	C_3	C_4	C_5
D_0	D_1	D_2	D_3	D_4	D_5
E_0	E_1	E_2	E_3	E_4	E_5
F_0	F_1	F_2	F_3	F_4	F_5

Matriz com entrelaçamento:

A_0	D_1	A_2	D_3	A_4	D_5
B_4	E_5	B_0	E_1	B_2	E_3
C_2	F_3	C_4	F_5	C_0	F_1
D_0	A_1	D_2	A_3	D_4	A_5
E_4	B_5	E_0	B_1	E_2	B_3
F_2	C_3	F_4	C_5	F_0	C_1

Vide a área hachurada da matriz que sofreu o entrelaçamento bidimensional. Observa-se que após o desentrelaçamento, os dígitos (símbolos) exibidos, ocupam palavras-código diferentes, daí a necessidade de um código corretor de linha de capacidade de correção t=1 erro.

Supõe-se o seguinte caso:

$Q = 6$, $r = 3$ e $s = 2$, logo T é

$$T = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}, \text{ então}$$

Matriz sem entrelaçamento:

A_0	A_1	A_2	A_3	A_4	A_5
B_0	B_1	B_2	B_3	B_4	B_5
C_0	C_1	C_2	C_3	C_4	C_5
D_0	D_1	D_2	D_3	D_4	D_5
E_0	E_1	E_2	E_3	E_4	E_5
F_0	F_1	F_2	F_3	F_4	F_5

Matriz com entrelaçamento:

A_0	E_1	C_2	A_3	E_4	C_5
B_3	F_4	D_5	B_0	F_1	D_2
C_0	A_1	E_2	C_3	A_4	E_5
D_3	B_4	F_5	D_0	B_1	F_2
E_0	C_1	A_2	E_3	C_4	A_5
F_3	D_4	B_5	F_0	D_1	B_2

Vide a área hachurada da matriz que sofreu o entrelaçamento bidimensional. Observa-se que após o desentrelaçamento, os dígitos (símbolos) exibidos ocupam palavras-código diferentes, daí a necessidade de um código corretor de linha de capacidade de correção t=1 erro.

Também, deve-se observar que a troca de valores na matriz de transformação apenas alterou a profundidade de entrelaçamento nas linhas e nas colunas.

2.3.5 Entrelaçamento bidimensional

Como pôde ser visto nos exemplos anteriores a técnica de entrelaçamento bidimensional permite deslocar a posição do símbolo tanto em linhas quanto em colunas.

A principal aplicação desta técnica é no armazenamento (gravação) de informação, de modo a se conseguir proteção contra surtos de erros bidimensionais ou espaciais. Caso esta técnica não fosse aplicada, os dígitos armazenados ou gravados estariam mais susceptíveis a uma série de problemas como imperfeições da mídia ou dispositivo de gravação, mau funcionamento do dispositivo de leitura/gravação, presença de corpos estranhos como sujeira, pó, umidade, etc.

A técnica de entrelaçamento bidimensional proposta no trabalho de Almeida [26] separa e entrelaça os dígitos, situados em uma determinada região, denominada região de cobertura, antes de armazená-los. Desta forma, na ocorrência de um surto espacial de erros, é possível espalhá-lo após o desentrelaçamento.

Exemplo 1: Dada uma região em que foram armazenados 5 x 5 = 25 dígitos como ilustra a Figura 2.13. Um material estranho que obstruísse, quando da leitura, os cinco dígitos da região circular em torno de b_{13}, isto é, b_8, b_{12}, b_{13}, b_{14}, b_{18}, faria com que existíssem, no pior caso, três erros em uma mesma linha. Portanto o código corretor de erros deveria ser capaz decorrigir pelo menos três erros, para que todos os erros deste surto pudessem ser corrigidos. No entanto, com o entrelaçamento, como será visto a seguir, é suficiente que as linhas sejam codificadas por um código de correção de um único erro.

Os dígitos deste arranjo serão espalhados segundo a matriz de transformação $T = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$. Sobre o resultado desta operação, aplica-se a operação módulo 5, para que o dígito permaneça nesta região. Por exemplo o dígito b_1 de coordenadas $B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ficará no mesmo lugar, pois $B' = TB$. O dígito b_4 será transladado de $B = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ para $B' = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$. A Figura 2.13 mostra o arranjo de 25 dígitos já entrelaçados.

	0	1	2	3	4	x/Δ
0	b_1	b_2	b_3	b_4	b_5	
1	b_6	b_7	b_8	b_9	b_{10}	
2	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	
3	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	
4	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	
y/Δ						

	0	1	2	3	4	x/Δ
0	b_1	b_{20}	b_9	b_{23}	b_{12}	
1	b_{24}	b_{13}	b_2	b_{16}	b_{10}	
2	b_{17}	b_6	b_{25}	b_{14}	b_3	
3	b_{15}	b_4	b_{18}	b_7	b_{21}	
4	b_8	b_{22}	b_{11}	b_5	b_{19}	
y/Δ						

Figura 2.13: Região de 25 dígitos antes e depois do entrelaçamento.

Supõe-se um surto de erros circular em torno de b_{25}, ou seja, supondo que os dígitos b_2, b_6, b_{25}, b_{14} e b_{18} estejam errados. Comprova-se que estes cinco dígitos se distribuirão, quando for feito o desentrelaçamento, ocupando cada dígito uma linha diferente. Desse modo, com um código com capacidade de correção de apenas um erro, consegue-se corrigir um surto de cinco erros. O entrelaçador protege também regiões aparentemente sem continuidade como é o caso da região centrada em b_8, que consiste dos dígitos b_{15}, b_{19}, b_8, b_{22} e b_1. Quando for feito o desentrelaçamento, verifica-se, mediante a observação da mesma Figura, que estes cinco dígitos se distribuirão, um em cada linha.

Almeida[26] definiu ganho de entrelaçamento como a capacidade de correção do código corretor de erro usando entrelaçamento dividido pela capacidade de correção do código sem entrelaçamento.

$$G_e = t_{com}/t_{sem.} \tag{2.13}$$

A Tabela 2.2 ilustra, segundo Almeida, os únicos esquemas de entrelaçamento obtidos denominados perfeitos, ou seja, quando em um arranjo de QxQ dígitos, Q dígitos são espalhados um em cada linha ou coluna, sendo Q ímpar.

Q	T	N	G_e
5	|2 1| |1 2|	5	3
9	|3 1| |1 3|	9	3
13	|5 1| |1 5|	13	5
25	|5 1| |1 5|	25	5
29	|5 1| |1 5|	27	7
49	|7 1| |1 7|	47	9
81	|9 1| |1 9|	69	11
113	|31 1| |1 31|	109	13
149	|13 1| |1 13|	143	15
197	|17 1| |1 17|	155	17
253	|16 1| |1 16|	199	19
317	|16 1| |1 16|	227	21

Tabela 2.2: Matriz de transformação, número de erros espalhados e ganho de entrelaçamento dos arranjos denominados perfeitos.

Exemplo 2: Seja uma região de 4x4 = 16 dígitos, como mostrada na Figura 2.14. A mesma Figura apresenta ainda este arranjo após o entrelaçamento. Observa-se que os dígitos b_{16}, b_2, b_5, b_{11} quando desentrelaçados pertencem cada um a uma única linha.

	0	1	2	3	x/Δ
0	b_1	b_2	b_3	b_4	
1	b_5	b_6	b_7	b_8	
2	b_9	b_{10}	b_{11}	b_{12}	
3	b_{13}	b_{14}	b_{15}	b_{16}	
y/Δ					

	0	1	2	3	x/Δ
0	b_1	b_7	b_9	b_{15}	
1	b_{10}	b_{16}	b_2	b_8	
2	b_3	b_5	b_{11}	b_{13}	
3	b_{12}	b_{14}	b_4	b_6	
y/Δ					

Figura 2.14: Região de 16 dígitos antes e depois do entrelaçamento.

A Tabela 2.3 ilustra alguns esquemas de entrelaçamento denominados perfeitos, para valores de Q tipo par.

Q	T	N	G_e
4	|2 1| |1 2|	4	2
12	|3 2| |2 3|	12	4
16	|4 1| |1 4|	16	4
24	|4 3| |3 4|	24	6
44	|8 1| |1 8|	42	8
68	|8 1| |1 8|	64	10
96	|20 3| |3 20|	90	12
140	|12 1| |1 12|	134	14
180	|24 5| |5 24|	166	16
232	|16 1| |1 16|	224	18
284	|30 1| |1 30|	270	20

Tabela 2.3: Matriz de transformação, número de erros espalhados e ganho de entrelaçamento dos arranjos denominados perfeitos.

É importante ressaltar que as Tabelas apresentadas tratam de surtos de erros, de tamanho *N*, e que os erros quando espalhados ocorrem no máximo um por linha, de tal modo que, com o uso de um código corretor que corrija apenas um erro, consegue-se correção total do surto.

O esquema de entrelaçamento apresentado por Almeida[26] também é eficaz no espalhamento de erros em surtos circulares que apresentam mais de *Q* erros. Por exemplo, em um arranjo de 25 pontos, consegue-se corrigir um surto circular de até nove erros com um código de dupla correção, conforme ilustra a Tabela 2.4. Observa-se que neste caso, os ganhos de entrelaçamento são menores que os ganhos obtidos com códigos que corrigiam apenas um erro. Pode-se notar ainda que neste caso as melhores matrizes de transformação, não são necessariamente iguais aquelas dos casos anteriores.

Q	***T***	***N***	G_e
5	|2 1| |1 2|	9	1,5
13	|5 1| |1 5|	25	2,5
29	|4 1| |1 4|	49	4

Tabela 2.4: Matriz de transformação, número de erros espalhados e ganho de entrelaçamento dos principais arranjos, para espalhamento de erros em códigos corretores com $t > 1$.

Capítulo 3

Código Matricial para Gravação Magnética Usando Entrelaçamento Unidimensional e Códigos Corretores de Erros Aleatórios

3.1 Introdução

A correção de surtos de erros bidimensionais, para formatos retangulares ou quadrados, tem sido abordada por diversos autores, dentre estes pode-se destacar Almeida[26] e Kauffman *et al* [25]. Em [26] houve um tratamento heurístico (por pesquisa computacional) da correção de surtos bidimensionais simétricos em códigos matriciais quadrados, ou seja, para matrizes nas quais o número de linhas, n_1, é igual ao número de colunas, n_2, sendo n_1 e n_2 inteiros positivos. Para tal, empregou-se uma transformação linear das posições dos dígitos da palavra-código para conseguir o entrelaçamento desejado. Em [25] provou-se matematicamente parte dos resultados obtidos em [26] para códigos matriciais quadrados, com lado $Q = rs$, sendo r e s inteiros positivos, e para os quais o surto bidimensional de erros apresentava tanto formato retangular r x s, para $r \neq s$, quanto formato quadrado, para $r = s$.

Neste trabalho de dissertação, baseado no artigo de Rocha [35], serão estudados códigos matriciais de dimensão n_1 x n_2, para n_1 e n_2 inteiros positivos, empregando em uma das dimensões (linha ou coluna) códigos corretores de erros aleatórios do tipo (n,k,d), com capacidade de correção de t erros, combinado com uma transformação linear de entrelaçamento, T, atuando somente na outra dimensão do código matricial. Esta proposição difere do tratamento mais usual, que se destaca pelo emprego de códigos corretores de erros ao longo das duas dimensões do código matricial em combinação com a transformação linear de entrelaçamento bidimensional.

Esta abordagem tem o intuito de não só simplificar como também de consolidar a forma de solução para o problema de correção de múltiplos surtos bidimensionais de erros.

3.2 Definições e Preliminares

Rocha [35] apresenta algumas definições que serão utilizadas neste trabalho. Considera-se que será empregado um código linear de blocos, ao longo de uma das dimensões da palavra-código pertencente a um código matricial. Esta palavra-código terá dimensão n_1 x n_2, com $n_2 = n$, o que corresponde a uma matriz cujas n_1 linhas são palavras-código de um código corretor linear de blocos (n,k,d). As posições dos dígitos na palavra-código são designadas pelos pares (x_i, y_i), sendo $0 \le x_i \le n_2\text{-}1$ e $0 \le y_i \le n_1 - 1$, ou seja, as posições em cada linha variam de 0 a n_2-1 e as posições em cada coluna variam de 0 a n_1-1. Doravante neste texto, designar-se-á *palavra-código pertencente ao código matricial* como sendo *matriz de dígitos*.

Para propósitos tanto de armazenamento quanto de transmissão de dados, esta matriz de dígitos é então entrelaçada por permutação dos seus dígitos em cada coluna, como detalhado a seguir.

Definição 3.1: Define-se uma matriz T de transformação linear como

$$T = \begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix}, \tag{3.1}$$

em que β é um inteiro positivo relativamente primo a n_1. Esta matriz de transformação executa o entrelaçamento das posições do dígito (x_i, y_i), $0 \le x_i \le n_2\text{-}1$ e $0 \le y_i \le n_1\text{-}1$, em uma matriz de dígitos retangular, A, de dimensões n_1 x n_2, por meio de uma transformação linear, como segue

$$(x_i, y_i).\begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix} = (x_i, -\beta.x_i + \beta.y_i) = (x'_i, y'_i), \tag{3.2}$$

em que (x'_i, y'_i), $0 \le x'_i \le n_2\text{-}1$ e $0 \le y'_i \le n_1\text{-}1$, corresponde à posição do dígito na matriz de dígitos entrelaçada, A_u. Salienta-se que esta operação é em módulo n_1.

A transformação anterior preserva a primeira coordenada, ou seja, $x_i = x'_i$, e atua somente na segunda coordenada. Isto é equivalente, para um dado valor fixo da primeira coordenada (coluna), a executar permutações dos dígitos na segunda coordenada da matriz de dígitos A com dimensão n_1 x n_2, a fim de obter a matriz entrelaçada A_u. Este tipo de entrelaçamento será considerado como "entrelaçamento simples ou unidimensional".

Exemplo:

Considerando uma matriz de dígitos A de dimensão 5 x 7, $n_1 = 5$ e $n_2 = 7$, mostrada na Figura 3.1a, cujas linhas são consideradas palavras-código de um código de blocos corretor de um erro aleatório, e a seguinte matriz de transformação linear T

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix}.$$

Como se observa $\beta = 2$, o qual é relativamente primo com $n_1 = 5$.

A matriz de dígitos resultante do entrelaçamento, A_u, é mostrada na Figura 3.1b.

y_i \ x_i	0	1	2	3	4	5	6
0	□$_1$	□$_2$	□$_3$	□$_4$	□$_5$	□$_6$	□$_7$
1	∇$_1$	∇$_2$	∇$_3$	∇$_4$	∇$_5$	∇$_6$	∇$_7$
2	♣$_1$	♣$_2$	♣$_3$	♣$_4$	♣$_5$	♣$_6$	♣$_7$
3	♦$_1$	♦$_2$	♦$_3$	♦$_4$	♦$_5$	♦$_6$	♦$_7$
4	♥$_1$	♥$_2$	♥$_3$	♥$_4$	♥$_5$	♥$_6$	♥$_7$

(a)

y_i \ x_i	0	1	2	3	4	5	6
0	□$_1$	∇$_2$	♣$_3$	♦$_4$	♥$_5$	□$_6$	∇$_7$
1	♦$_1$	♥$_2$	□$_3$	∇$_4$	♣$_5$	♦$_6$	♥$_7$
2	∇$_1$	♣$_2$	♦$_3$	♥$_4$	□$_5$	∇$_6$	♣$_7$
3	♥$_1$	□$_2$	∇$_3$	♣$_4$	♦$_5$	♥$_6$	□$_7$
4	♣$_1$	♦$_2$	♥$_3$	□$_4$	∇$_5$	♣$_6$	♦$_7$

(b)

Figura 3.1: (a) Matriz de dígitos A com dimensão 5 x 7;
(b) Matriz de dígitos entrelaçada A_u.

Observa-se pelo exemplo que há a manutenção da primeira coordenada, x_i, e que os dígitos se deslocam apenas no sentido vertical correspondente à segunda coordenada, y_i.

Definição 3.2: Define-se uma *mancha-d_L* de erros em uma matriz de dígitos n_1 x n_2 como sendo um conjunto de dígitos de erros cuja distância de Lee entre seus pares é de no máximo d_L.

Tão importante quanto definir a matriz de transformação linear T, que perfaz o entrelaçamento, é a definição precisa de surto bidimensional de erros. Embora haja inúmeras definições de manchas de erros, como há para surtos de erros em transmissão serial de dados, é razoável considerar uma definição que leva em conta quão próximo entre si estão os dígitos na mancha de erros, mais do que levar em conta a sua forma. Consequentemente esta definição de mancha de erros, aqui considerada, leva em conta a distância de Lee, d_L, entre pares de posições de erros. A distância de Lee [36, p. 41] entre duas posições da matriz de dígitos com coordenadas (x_1,y$_1$) e (x_2,y_2), é definida como sendo

$$d_L = min\ \{|x_2 - x_1|, |n_2 - (x_2 - x_1)|\} + min\ \{|\, y_2 - y_1|, |\, n_1 - (y_2 - y_1)|\}. \qquad (3.3)$$

Exemplo: Considerando-se a matriz entrelaçada de dígitos A_u com dimensão 5 x 7 mostrada a seguir, pode-se dizer que o conjunto dos dígitos ∇_2, $\blacklozenge_1$, $\heartsuit_2$, $\square_3$ e $\clubsuit_2$ constitui uma *mancha-d_L* com distância de Lee máxima entre seus pares igual a dois.

y_i \ x_i	0	1	2	3	4	5	6
0	$\square_1$	∇_2	$\clubsuit_3$	$\blacklozenge_4$	$\heartsuit_5$	$\square_6$	∇_7
1	$\blacklozenge_1$	$\heartsuit_2$	$\square_3$	∇_4	$\clubsuit_5$	$\blacklozenge_6$	$\heartsuit_7$
2	∇_1	$\clubsuit_2$	$\blacklozenge_3$	$\heartsuit_4$	$\square_5$	∇_6	$\clubsuit_7$
3	$\heartsuit_1$	$\square_2$	∇_3	$\clubsuit_4$	$\blacklozenge_5$	$\heartsuit_6$	$\square_7$
4	$\clubsuit_1$	$\blacklozenge_2$	$\heartsuit_3$	$\square_4$	∇_5	$\clubsuit_6$	$\blacklozenge_7$

Figura 3.2: Ilustração de uma *mancha-d_L*, para $d_L = 2$.

Definição 3.3: Define-se peso do surto bidimensional de erros como sendo a quantidade encontrada de dígitos de erros.

Definição 3.4: Define-se uma mancha de erros casada em uma matriz de dígitos n_1 x n_2 como sendo uma *mancha-d_L* de erros de peso n_1, ou seja, o número de dígitos na mancha de erros coincide com o número de linhas da matriz de dígitos.

Definição 3.5: Define-se uma mancha de erros compacta em uma matriz de dígitos n_1 x n_2 como sendo qualquer padrão de erros bidimensional de peso no máximo n_1, o qual é coberto pelas manchas de erros casadas.

Diz-se que uma mancha de erros P2 é coberta por outra mancha de erros P1 se qualquer posição de erro em P2 está contida em P1.

Exemplo: Na Figura 3.3 está ilustrada uma *mancha-d_L* que se constitui ao mesmo tempo em uma mancha de erros casada e compacta, uma vez que seu peso é igual a cinco (igual a n_1).

y_i \ x_i	0	1	2	3	4	5	6
0	□1	∇2	♣3	♦4	♥5	□6	∇7
1	♦1	♥2	□3	∇4	♣5	♦6	♥7
2	∇1	♣2	♦3	♥4	□5	∇6	♣7
3	♥1	□2	∇3	♣4	♦5	♥6	□7
4	♣1	♦2	♥3	□4	∇5	♣6	♦7

Figura 3.3: Ilustração de uma mancha de erros compacta e casada, tipo *mancha-d_L*, com peso igual a cinco.

Manchas de erros compactas representam um papel chave neste trabalho de dissertação porque constituem os surtos bidimensionais de erros que garantidamente são corrigidos por este método de entrelaçamento unidimensional. Outros tipos de manchas de erros, além das manchas de erros compactas, podem também ser corrigidos por este esquema, o que na realidade se torna um ganho extra e não um resultado direto da proposição deste trabalho.

Lemma 3.1: A matriz de transformação linear para desentrelaçamento T^{-1} obtida de T, é dada por

$$T^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & \beta^{-1} \end{bmatrix}, \tag{3.4}$$

em que β^{1} é o inverso multiplicativo de β módulo n_1.

Prova: Uma vez que foi assumido que β e n_1 são relativamente primos, logo β^{1} existe. Imediatamente é verificado que $T.T^{-1} = I_2$, em que I_2 é uma matriz unitária de ordem 2.

3.3 Espalhamento e Correção

O entrelaçamento perfeito de uma mancha de erros compacta de peso n, em uma matriz de dígitos quadrada n x n, foi definido em [26] como sendo o entrelaçamento para o qual existe no máximo um erro ocupando a mesma linha ou a mesma coluna na matriz de dígitos

quadrada, após ser desentrelaçada. Aplicando esta definição para uma matriz de dígitos n_1 x n_2 observa-se que o entrelaçamento perfeito de uma dada mancha de erros pode somente ser conseguido se o seu máximo peso não exceder $min\ \{ n_1, n_2\}$. Contudo, caso se use um código corretor de t erros aleatórios para codificar as linhas da matriz de dígitos (antes do entrelaçamento), não deve existir uma preocupação em se ter o entrelaçamento perfeito. Tudo o que precisa ser garantido é que, depois do desentrelaçamento, cada linha da matriz de dígitos contenha no máximo t posições com erro.

Para o propósito deste trabalho a definição de entrelaçamento perfeito será modificada ligeiramente, como segue.

Definição 3.6: Define-se espalhamento perfeito de uma mancha de erros de peso máximo n_1, associado com uma matriz de dígitos A_u de dimensão n_1 x n_2, obtido por entrelaçamento unidimensional, como sendo aquele em que, após o desentrelaçamento, dois ou mais erros na mancha não ocupam a mesma linha na matriz de dígitos resultante.

Exemplo: Na matriz de dígitos entrelaçada, A_u, exibida na Figura 3.4, observa-se que há entrelaçamento perfeito, uma vez que a mancha de erros em destaque com peso igual a cinco e formada pelo conjunto de dígitos ∇_2, $\blacklozenge_1$, $\heartsuit_2$, $\square_3$ e $\clubsuit_2$, ao passar pelo desentrelaçamento será espalhada igualmente pelas linhas da matriz de dígitos, resultando em, no máximo, um erro por linha. Uma vez que a mancha é espalhada, um código corretor de erros aleatório com t=1 pode ser usado em cada linha da matriz para corrigi-la.

y_i \ x_i	0	1	2	3	4	5	6
0	$\square_1$	∇_2	$\clubsuit_3$	$\blacklozenge_4$	$\heartsuit_5$	$\square_6$	∇_7
1	$\blacklozenge_1$	$\heartsuit_2$	$\square_3$	∇_4	$\clubsuit_5$	$\blacklozenge_6$	$\heartsuit_7$
2	∇_1	$\clubsuit_2$	$\blacklozenge_3$	$\heartsuit_4$	$\square_5$	∇_6	$\clubsuit_7$
3	$\heartsuit_1$	$\square_2$	∇_3	$\clubsuit_4$	$\blacklozenge_5$	$\heartsuit_6$	$\square_7$
4	$\clubsuit_1$	$\blacklozenge_2$	$\heartsuit_3$	$\square_4$	∇_5	$\clubsuit_6$	$\blacklozenge_7$

Figura 3.4: Espalhamento perfeito de uma mancha de erros de peso igual a n_1.

A seguir será estabelecida uma relação entre o inteiro β, usado na matriz de transformação linear T, e as posições dos dígitos na mancha de erros com peso máximo igual a n_1, associado com uma dada matriz de dígitos n_1 x n_2.

Teorema 3.1: Seja A_u uma matriz de dígitos n_1 x n_2 entrelaçada por meio de T, possivelmente corrompida por manchas de erros. Seja T^{-1} a transformação linear de desentrelaçamento. Um espalhamento perfeito existe se e somente se há um inteiro β tal que

$$(x_2 - x_1) + \beta^{-1}(y_2 - y_1) \not\equiv 0 (mod\, n_1), \tag{3.5}$$

para qualquer par de posições de dígitos (x_1, y_1) e (x_2, y_2) pertencente a uma mancha de erros de peso de no máximo n_1, associado com uma matriz de dígitos n_1 x n_2.

Prova: Supõe-se duas posições quaisquer (x_1, y_1) e (x_2, y_2) na mancha de erros com peso de no máximo n_1 e aplicando a transformação T^{-1} para cada uma das posições tem-se

$$(x_1, y_1)\begin{bmatrix} 1 & 1 \\ 0 & \beta^{-1} \end{bmatrix} = (x_1, x_1 + \beta^{-1} y_1). \tag{3.6}$$

$$(x_2, y_2)\begin{bmatrix} 1 & 1 \\ 0 & \beta^{-1} \end{bmatrix} = (x_2, x_2 + \beta^{-1} y_2). \tag{3.7}$$

Se as posições transformadas $(x_1, x_1 + \beta^1 y_1)$ e $(x_2, x_2 + \beta^1 y_2)$ ocupam a mesma linha da matriz de dígitos, então de (3.6) e (3.7) tem-se

$$x_1 + \beta^1 y_1 \equiv x_2 + \beta^1 y_2 \mod n_1, \tag{3.8}$$

ou

$$(x_2 - x_1) + \beta^1 (y_2 - y_1) \equiv 0 \mod n_1, \tag{3.9}$$

o que é uma contradição com a expressão em (3.5).

Consequentemente (3.5) é a condição suficiente para assegurar espalhamento perfeito dos dígitos da mancha de erros de peso no máximo n_1. Em outras palavras se é assumido que (3.6) e (3.7) representam posições em duas linhas distintas da matriz de dígitos então segue que $x_1 + \beta^1 y_1 \equiv x_2 + \beta^1 y_2 \ (mod\ n_1)$, e desta forma verifica-se que (3.5) também é uma condição necessária para o espalhamento perfeito dos dígitos em uma mancha de erros de peso no máximo n_1, e assim o Teorema é provado.

Exemplo: Para a matriz de dígitos A com dimensão 5 x 7 identificou-se a seguinte matriz de transformação linear T

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix},$$

em que $\beta = 2$ e $\beta^1 = 3$. Desta forma o Teorema 3.1 pode ser ilustrado para a mancha exibida na Figura 3.4, como segue.

Dígito	x_i	y_i	$x_i + \beta^1 y_i \ (mod\ n_1)$
♦1	0	1	3
∇2	1	0	1
♥2	1	1	4
♣2	1	2	2
□3	2	1	0

Portanto satisfazendo a condição $x_1 + \beta^1 y_1 \equiv x_2 + \beta^1 y_2 \ (mod\ n_1)$ para quaisquer posições (x_1,y_1) e (x_2,y_2) pertencentes à mesma mancha de erros.

Pode-se usar diretamente a relação (3.5), ou seja, $(x_2 - x_1) + \beta^1 (y_2 - y_1) \equiv 0 \bmod n_1$. Neste caso $0 \le (x_2 - x_1) \le 2$ e $0 \le (y_2 - y_1) \le 2$, em que $(x_2 - x_1)$ e $(y_2 - y_1)$ não podem valer zero simultaneamente, uma vez que estaria significando a mesma posição da matriz. Além disto $(x_2 - x_1) + (y_2 - y_1) \le d_L = 2$, conforme definição 3.2, por se tratar de uma mancha compacta, do tipo d_L-*mancha*, com distância máxima de Lee igual a dois. Então neste caso as possíveis equações de incongruência são:

a) Para $(x_2 - x_1) = 0$

$\beta^1(y_2 - y_1) \not\equiv 0 \bmod n_1$, ou seja, para $\beta^1 = 3$, $1 \leq (y_2 - y_1) \leq 2$ e $n_1 = 5$, então

$3 \not\equiv 0 \bmod 5$ e

$1 \not\equiv 0 \bmod 5$

b) Para $(x_2 - x_1) = 1$

$1 + \beta^1(y_2 - y_1) \not\equiv 0 \bmod n_1$, ou seja, para $\beta^1 = 3$, $0 \leq (y_2 - y_1) \leq 1$ e $n_1 = 5$, então

$1 \not\equiv 0 \bmod 5$ e

$4 \not\equiv 0 \bmod 5$. E por último

c) Para $(x_2 - x_1) = 2$

$2 + \beta^1(y_2 - y_1) \not\equiv 0 \bmod n_1$, ou seja, para $\beta^1 = 3$, $(y_2 - y_1) = 0$ e $n_1 = 5$, então

$2 \not\equiv 0 \bmod 5$

Portanto o valor escolhido para β satisfaz a condição de espalhamento perfeito, conforme a expressão (3.5).

Dado que (3.5) é provado e desde que a transformação linear T é reversível, conclui-se que, se t ou menos manchas de erros de peso no máximo n_1 afetam a matriz de dígitos A_u, depois do desentrelaçamento cada linha da matriz de dígitos resultante terá no máximo t erros. Desta forma um código corretor de t-erros aleatórios seria capaz de corrigir os erros nas linhas.

Lemma 3.2 O inverso aditivo de β^1 mod n_1 também satisfaz a relação $(x_2 - x_1) + \beta^{-1}(y_2 - y_1) \equiv 0 (mod\ n_1)$ e portanto satisfaz ao critério de espalhamento perfeito.

Prova:

Dado que $\beta^1 = n_1 - K$, em que K corresponde ao inverso aditivo de β^1 mod n_1, observa-se que a relação 3.5 se torna

$$(x_2\text{-}x_1) + (n_1\text{-}K)(y_2\text{-}y_1) \not\equiv 0 \bmod n_1, \tag{3.10}$$

então

$$(x_2\text{-}x_1) - K(y_2\text{-}y_1) \not\equiv 0 \bmod n_1, \tag{3.11}$$

ou

$$(x_2\text{-}x_1) + K(y_1\text{-}y_2) \not\equiv 0 \bmod n_1. \tag{3.12}$$

Demonstrando-se desta forma que o inverso aditivo de β^1 também satisfaz a relação 3.5. QED.

Teorema 3.2: Seja A_u uma matriz de dígitos n_1 x n_2 obtida de uma matriz de dígitos A pela transformação linear T, possivelmente corrompida por manchas de erros. Considerando que as linhas de A são palavras-código de um código corretor de blocos (n,k,d) capaz de corrigir t erros aleatórios e supondo que β satisfaz o Teorema 3.1, assegura-se que por meio da operação de desentrelaçamento T^1 qualquer padrão de no máximo t manchas de erros de peso máximo n_1 em A_u é corrigível.

É importante restringir os tipos de manchas de erros para garantir que a desigualdade (3.5) se mantenha válida. Por esta razão, a partir de agora, serão consideradas somente manchas de erros compactas ou mais especificamente *manchas-d_L* compactas.

Lemma 3.3 A distância de Lee, d_L, entre pares de posições de dígitos em uma mancha compacta de erros satisfaz a seguinte desigualdade

$$d_L \leq -1 + \sqrt{2n_1 - 1}\,. \tag{3.13}$$

Sendo n_1 o número de linhas da palavra-código.

Prova: O número N_{d_L} de posições com distância de Lee de no máximo d_L em relação a uma posição arbitrária, porém fixa, da matriz de dígitos, é dada por [35]

$$N_{d_L} = 1 + 2d_L(d_L + 1)\,. \tag{3.14}$$

Desta forma, uma *mancha-d_L* de erros tem peso no máximo N_{d_L}. Consequentemente por definição, uma mancha de erros compacta tem peso máximo n_1, o que significa que $N_{d_L} \leq n_1$, ou

$$2d_L^2 + 2d_L + 1 \leq n_1\,. \tag{3.15}$$

Resolvendo (3.15) para d_L obtém-se o resultado desejado depois da multiplicação por dois, para cobrir o caso de erros eqüidistantes do centro da mancha.

Exemplo: No caso considerado até então, ilustrado na Figura 3.4, se trata de uma *mancha-d_L* de erros compacta com peso igual a cinco e distância de Lee máxima igual a dois, o que pode ser demonstrado pelas equações (3.13) e (3.14), considerando-se $n_1 = 5$.

Teorema 3.3: Sejam (x_1,y_1) e (x_2,y_2) duas posições que representam dois dígitos pertencentes a uma mancha de erros compacta associada com uma matriz de dígitos n_1 x n_2. Estas manchas de erros são corrigíveis por entrelaçamento unidimensional se e somente se há um inteiro positivo β que satisfaz a seguinte condição

$$|z|_L + |\beta^{-1}z|_L \rangle - 1 + \sqrt{2n_1 - 1}\,, \tag{3.16}$$

em que z é um inteiro positivo, $1 \leq z \leq -2 + \sqrt{2n_1 - 1}$, e $|.|_L = d_L(.,0) \bmod n_1$.

Prova: Sem perda de generalidade pode-se supor que (x_1,y_1) é a solução da congruência $x + \beta^1 y \equiv c \bmod n_1$. Desta forma, para uma solução qualquer (x,y) desta congruência, pode-se escrever

$$(x\text{-}x_1) + \beta^1(y\text{-}y_1) \equiv 0 \bmod n_1. \tag{3.17}$$

Considerando $y\text{-}y_1 \equiv z$, então

$$x - x_1 \equiv -\beta^1 z, \tag{3.18}$$

em que z é um inteiro. Consequentemente a distância de Lee d_L entre (x,y) e (x_1,y_1) pode ser escrita como

$$d_L = |z|_L + |\beta^1 z|_L. \tag{3.19}$$

Pelo Teorema 3.1 os dígitos na mancha compacta de erros devem satisfazer a inequação (3.5). Consequentemente conclui-se que (x,y) e (x_1,y_1) não podem pertencer a uma mesma mancha compacta de erros. Consequentemente pelo Lemma 3.3 a distância de Lee do par deve satisfazer $d_L \rangle - 1 + \sqrt{2n_1 - 1}$, ou seja,

$$|z|_L + |\beta^{-1} z|_L \rangle - 1 + \sqrt{2n_1 - 1}\,.$$

Qualquer valor de β satisfazendo esta inequação, para $1 \leq z \leq -2 + \sqrt{2n_1 - 1}$, garante simultaneamente que pares de posições satisfazendo a congruência não pertencem ambos a mesma mancha compacta de erros e sua distância de Lee é maior do que aquela obtida entre quaisquer pares de erros na mancha de erros compacta.

Exemplo:

Considerando uma matriz de dígitos entrelaçada A_u, com dimensão 5 x 7, mostrada na Figura 3.5, observa-se que os dígitos $\heartsuit_2$ e $\heartsuit_4$ ou $\diamondsuit_1$ e $\diamondsuit_3$, por exemplo, apesar de pertencerem, após o desentrelaçamento, à mesma linha, pelo critério da *mancha-d_L* compacta ambos pertencem a manchas de erros distintas. Desta forma o código corretor de erros aleatórios terá que ser capaz de corrigir pelo menos dois erros. A obtenção destes dígitos em manchas de erros ou *manchas-d_L* compactas distintas só é possível porque β^1 atende ao Teorema 3.3.

	0	1	2	3	4	5	6
0	□1	∇2	♣3	♦4	♥5	□6	∇7
1	♦1	♥2	□3	∇4	♣5	♦6	♥7
2	∇1	♣2	♦3	♥4	□5	∇6	♣7
3	♥1	□2	∇3	♣4	♦5	♥6	□7
4	♣1	♦2	♥3	□4	∇5	♣6	♦7

Figura 3.5: Matriz de dígitos entrelaçada, A_u, exibindo duas manchas de erros distintas.

Lemma 3.4: A repetição do padrão de entrelaçamento, que se passa a considerar período, ocorre a cada n_1 colunas da matriz de dígitos entrelaçada A_u.

Prova:

Sejam dois pontos quaisquer da matriz de dígitos, sendo o primeiro (x_1,y_1) e o segundo (x_n,y_n), pertencentes à mesma linha, ou seja, $y_1=y_n=y$. Após o entrelaçamento passam a ocupar respectivamente as seguintes coordenadas:

$$(x_1,-x_1\beta+y\beta) \text{ e } (x_n,-x_n\beta+y\beta). \tag{3.20}$$

Supondo que os mesmos ocupem a mesma linha (repetição do padrão), então

$$-x_1\beta+y\beta \equiv -x_n\beta+y\beta \bmod n_1. \tag{3.21}$$

Desta forma

$-x_1\beta \equiv -x_n\beta \bmod n_1$. Como $x_n = x_1+\theta$, sendo θ o período de entrelaçamento, tem-se

$$-x_1\beta \equiv -x_1\beta-\theta\beta \bmod n_1. \tag{3.22}$$

Logo, $\theta\beta \equiv 0 \bmod n_1$. Como β e n_1 são relativamente primos, a solução da congruência é θ múltiplo de n_1, ou seja,

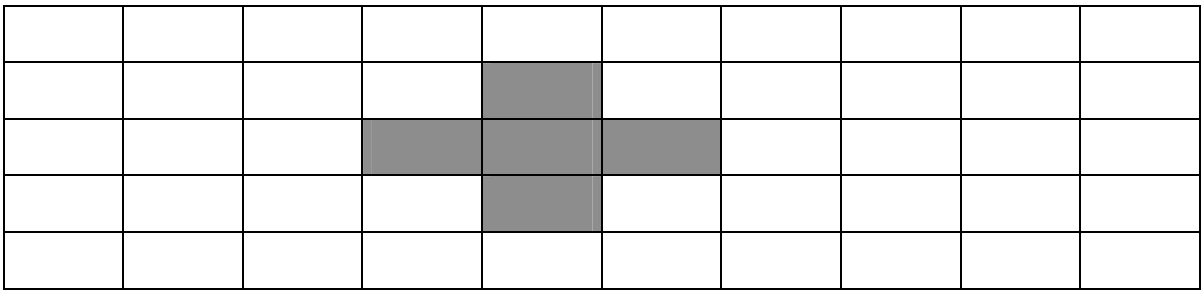
$$\theta = \tau.\, n_1, \tag{3.23}$$

para $\tau = 1, 2, 3,.....$

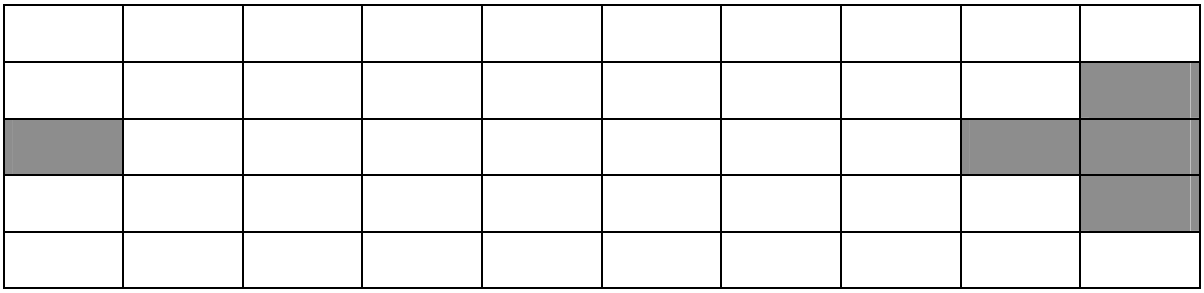
Logo o padrão de entrelaçamento se repete de $\boldsymbol{n_1}$ em $\boldsymbol{n_1}$ colunas. QED.

Exemplo: Vide que na Figura 3.5 padrão de entrelaçamento da coluna 0 se repete na coluna 5 e o padrão da coluna 1 se repete na coluna 6, demonstrando o Lemma 3.4.

Lemma 3.5: Para assegurar que as manchas de erros compactas (*manchas-d_L* compactas), possam ser vistas ciclicamente sobre a matriz de dígitos, o número de colunas da matriz, n, deve ser múltiplo de n_1.



(a)



(b)

Figura 3.6: (a) Mancha de erros compacta centralizada;
(b) Mancha de erros compacta vista ciclicamente.

Para que isto aconteça, com base no Lemma 3.4, basta que n_2 seja múltiplo de n_1, ou seja:

$$n_2 = \gamma . n_1, \tag{3.24}$$

para $\gamma = 1,2,3,.....$

3.4 Tipos de Manchas de Erros Corrigíveis

Como foi ressaltado, o objetivo principal deste método é procurar corrigir manchas de erros "compactas". A definição 3.5 e o Lemma 3.3 tratam deste tipo de mancha como sendo constituído por no máximo n_1 erros e com um valor máximo estabelecido para distância de Lee entre os pares de erros.

Apesar deste tipo de mancha exercer um papel fundamental neste trabalho é possível ainda a correção de outros tipos de manchas, que acabam se configurando em um ganho extra.

Na Figura 3.7 estão reproduzidos alguns formatos de manchas de erros passíveis de correção por este método. Neste exemplo considera-se uma matriz de dígitos entrelaçada A_u com dimensão 5 x 10, n_1 = 5 e n_2 =10, obtida a partir do entrelaçamento de uma matriz de dígitos A pela matriz de transformação linear, T, mostrada a seguir:

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix}.$$

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

(a)

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

(b)

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

(c)

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

(d)

Figura 3.7: Exemplos de manchas de erros passíveis de correção.

Na Figura 3.7, apesar de todas as manchas de erros exemplificadas atenderem ao critério de peso máximo igual a n_1, apenas a mancha de erros representada em 3.7.a atende ao critério da distância máxima de Lee, conforme estabelecido no Lemma 3.3. Desta forma é o único tipo de mancha de erros considerado compacto e que, portanto, está contemplado integralmente na proposta deste trabalho.

Quanto às demais, representadas de 3.7.b a 3.7.d, atendem somente ao critério do peso máximo e portanto não podem ser consideradas como manchas compactas ou *manchas -d_L* de erros. Apesar disto, elas podem ser corrigidas pelo método proposto, tornando-se portanto um bônus adicional a este trabalho.

3.5 Cálculo do Parâmetro β da Matriz de Transformação Linear

Um dos cuidados que se deve ter para que ocorra o entrelaçamento perfeito é a determinação exata do parâmetro β da matriz de transformação linear. Este parâmetro deve atender as condições estabelecidas pelos Teoremas 3.1, 3.2 e 3.3. É com base nestes Teoremas e nas definições de distância de Lee e de manchas compactas de erros que será estabelecida uma seqüência lógica de cálculo deste parâmetro. Como resultado desta metodologia de cálculo foi gerado um programa de computador capaz de montar uma Tabela relacionando os valores de β com o parâmetro n_1 (número de linhas) da matriz de dígitos. Esta Tabela está ilustrada no Apêndice II deste trabalho.

A seguir serão dados alguns exemplos de como funciona esta metodologia de cálculo, que foi empregada no programa de computador, e, por tabela, alguns resultados relacionando os parâmetros β e n_1.

Observação: Por simplicidade, considerar-se-á, como solução, apenas valores de β menores que n_1.

Exemplo 1: Considerando-se uma matriz de dígitos A com número de linhas igual a cinco, ou seja, $n_1 = 5$, a matriz de transformação linear, T, deverá ter o seguinte formato:

$$T = \begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix}.$$

Existem $\phi(n_1) = \phi(5) = 4$ [32] possíveis valores de β (para $\beta < n_1$), a saber: 1, 2, 3 e 4. Como todos estes valores são relativamente primos com n_1, a faixa de soluções para β^1 também se torna a mesma.

A partir deste ponto o que se fará é uma filtragem destas possíveis soluções a fim de encontrar aquelas que satisfazem aos Teoremas 3.1 e 3.3.

De acordo com o Teorema 3.3, o valor de β^1 deve satisfazer a expressão (3.16) reproduzida a seguir:

$|z|_L + |\beta^{-1} z| \rangle - 1 + \sqrt{2n_1 - 1}$, em que $1 \le z \le -2 + \sqrt{2n_1 - 1}$.

Desta forma, para $n_1 = 5$, tem-se

$1 \le z \le 1$. Logo a expressão (3.16) se torna

$1 + \beta^1 > 2$.

Da faixa inicial considerada para possíveis valores β^1, apenas os números 2, 3 e 4 satisfazem à relação.

Finalmente para determinar os valores definitivos de β^1 emprega-se o Teorema 3.1, cuja expressão (3.5) é reproduzida a seguir:

$(x_2 - x_1) + \beta^1 (y_2 - y_1) \not\equiv 0 \bmod n_1$.

Antes de empregar esta relação, deve-se determinar qual a faixa de valores para $(x_2 - x_1)$ e $(y_2 - y_1)$, para tal será usada a expressão (3.14) do Lemma 3.3 que calcula o peso da mancha compacta de erros e a partir deste ponto é possível definir a mancha de erros e as posições do dígitos. Esta expressão é reproduzida a seguir:

$$N_{d_L} = 1 + 2d_L(d_L + 1),$$

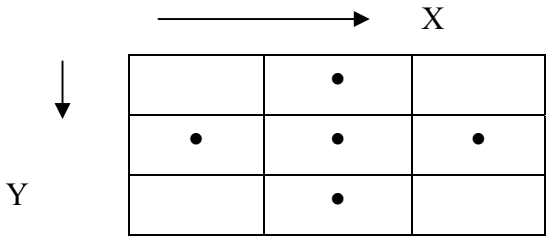
em que

$$d_L \le (-1 + \sqrt{2n_1 - 1})/2 = 1,$$

que corresponde à distância de Lee em relação ao centro da mancha de erros, então

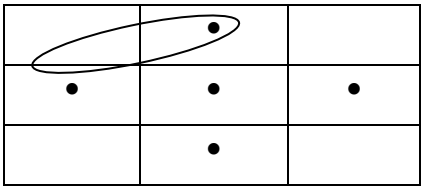
$N_{d_L} = 5$.

Logo o formato da mancha de erros compacta passa a ser

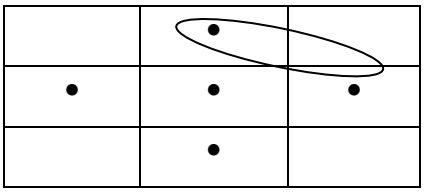


Portanto pode-se ter as seguintes combinações de pares de dígitos da mancha:

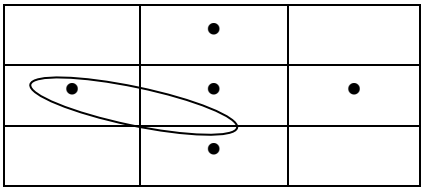
a) Para $x_2-x_1=y_2-y_1 = 1$



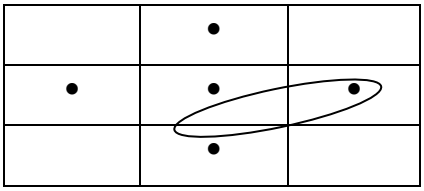
ou



ou



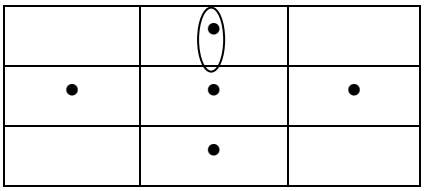
ou



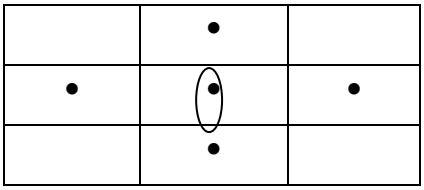
Para este caso a expressão (3.5) fica sendo

$$1 + \beta^1 \not\equiv 0 \bmod 5.$$

b) Para x_2-x_1=0 e y_2-y_1 = 1



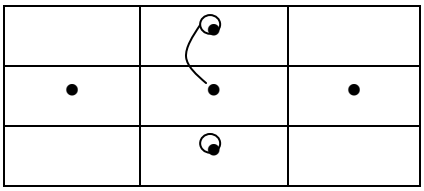
ou



Para este caso a expressão (3.5) fica sendo

$$\beta^1 \not\equiv 0 \bmod 5.$$

c) Para x_2-x_1=0 e y_2-y_1 = 2



Para este caso a expressão (3.5) fica sendo

$$2\,\beta^1 \not\equiv 0 \bmod 5.$$

Tendo por base estas três relações, pode-se demonstrar que o número 4 não satisfaz a todas, logo os valores de β^1, e por tabela de β (porque ambos são relativamente primos a n_1), que permitem o entrelaçamento perfeito para n_1=5 são:

n_1	β **ou** β^1
5	2 ou 3

Neste caso as matrizes de transformação linear, T, podem ser:

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}.$$

Este resultado confirma o Lemma 3.2, demonstrando que as soluções são inverso aditivos módulo n_1, ou seja, para módulo cinco, dois é inverso aditivo de três e vice-versa. Por conta deste Lemma o número 4 já poderia ter sido descartado uma vez que o número 1, que é o seu inverso aditivo, foi eliminado no primeiro teste. Este tipo de análise, tendo por objetivo simplificar e minimizar o tempo de processamento, foi implementado no aplicativo computacional.

Exemplo 2: Considerando uma matriz de dígitos A com número de linhas igual a 13, ou seja, $n_1 = 13$, a matriz de transformação linear, T, deverá ter o seguinte formato:

$$T = \begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix}.$$

Existem $\phi(n_1) = \phi(13) = 12$ [32] possíveis valores de β (para $\beta < n_1$), a saber: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 e 12.

Da mesma forma que no exemplo 1, será realizada uma filtragem destas possíveis soluções de β, a fim de encontrar aquelas que satisfazem aos Teoremas 3.1 e 3.3.

De acordo com o Teorema 3.3, o valor de β^1 deve satisfazer a expressão (3.16) mais uma vez reproduzida a seguir:

$$|z|_L + \left|\beta^{-1} z\right\rangle - 1 + \sqrt{2n_1 - 1}\,, \text{ em que } 1 \le z \le -2 + \sqrt{2n_1 - 1}\,.$$

Desta forma, para $n_1 = 13$, tem-se

$1 \leq z \leq 3$. Logo substituindo os valores de z na expressão (3.16) obtém-se as seguintes inequações:

$$1+ \beta^1 > 4;$$
$$2+ 2\beta^1 > 4;$$
$$3+ 3\beta^1 > 4.$$

Da faixa inicial considerada para possíveis valores β^1, apenas os números 4, 5, 6, 7, 8, 9, 10, 11 e 12 satisfazem as três inequações.

Finalmente para determinar os valores definitivos de β^1 emprega-se o Teorema 3.1, cuja expressão (3.5) é reproduzida a seguir:

$$(x_2\text{-}x_1) + \beta^1(y_2\text{-}y_1) \not\equiv 0 \bmod n_1.$$

Antes de empregar esta relação, deve-se determinar qual a faixa de valores para $(x_2\text{-}x_1)$ e $(y_2\text{-}y_1)$, para tal será usada a expressão (3.14) do Lemma 3.3 que calcula o peso da mancha compacta de erros e, a partir deste ponto, é possível definir a mancha de erros e as posições do dígitos. Esta expressão é reproduzida a seguir:

$$N_{d_L} = 1 + 2d_L(d_L + 1),$$

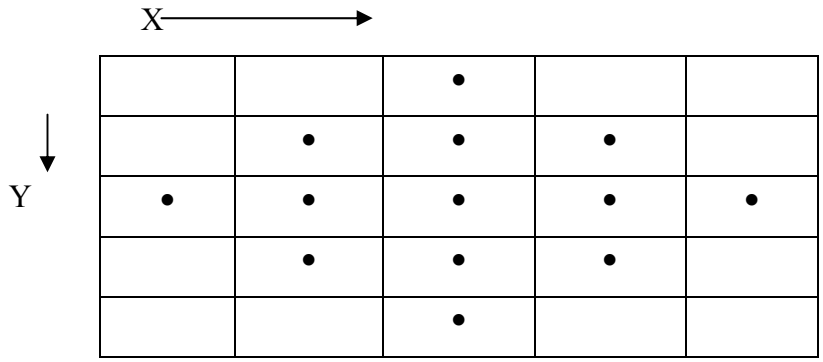
em que

$$d_L \leq (-1 + \sqrt{2n_1 - 1}) / 2 = 2,$$

que corresponde à distância de Lee em relação ao centro da mancha de erros, então

$$N_{d_L} = 13.$$

Logo o formato da mancha de erros compacta passa a ser



Portanto neste caso irão ser consideradas as seguintes faixas para (x_2-x_1) e (y_2-y_1):

a) Faixa de (x_2-x_1):

$$0 \leq x_2\text{-}x_1 \leq 3.$$

De acordo com a ilustração anterior, esta faixa deveria se estender até 4, que ocorrem entre os dois dígitos localizados nos extremos da mancha de erros. No entanto quando x_2-x_1 = 4, o par de dígitos se encontra sobre a mesma linha da matriz e portanto y_2-y_1 = 0, anulando o termo β^1 na expressão (3.5).

b) Faixa de (y_2-y_1):

$$1 \leq y_2 - y_1 \leq 4.$$

Para a montagem das relações, baseadas na expressão (3.5), deve-se levar em conta que a máxima distância de Lee entre dois dígitos quaisquer da mancha de erros compacta, d_L-*mancha*, tem que ser no máximo igual a $-1+\sqrt{2n_1-1} = 4$, conforme o que está descrito na expressão (3.13) do Lemma 3.3. Desta forma deve ser observado que

$$(x_2\text{-}x_1) + (y_2\text{-}y_1) \leq 4.$$

Com base no exposto serão montadas as seguintes relações baseadas na expressão (3.5):

a) Para x_2-x_1= 0

$$\beta^1 \not\equiv 0 \bmod 13;$$
$$2\,\beta^1 \not\equiv 0 \bmod 13;$$
$$3\,\beta^1 \not\equiv 0 \bmod 13;$$
$$4\,\beta^1 \not\equiv 0 \bmod 13.$$

b) Para x_2-x_1= 1

$$1 + \beta^1 \not\equiv 0 \bmod 13;$$
$$1 + 2\,\beta^1 \not\equiv 0 \bmod 13;$$
$$1 + 3\,\beta^1 \not\equiv 0 \bmod 13.$$

c) Para x_2-x_1= 2

$$2 + \beta^1 \not\equiv 0 \bmod 13;$$
$$2 + 2\,\beta^1 \not\equiv 0 \bmod 13.$$

d) Para x_2-x_1= 3

$$3 + \beta^1 \not\equiv 0 \bmod 13.$$

Tendo por base estas relações, pode-se demonstrar que os números 4, 10, 11 e 12 não satisfazem a todas, logo os possíveis valores de β^1, e por tabela de β (porque ambos são relativamente primos a n_1), que permitem o entrelaçamento perfeito para n_1=13 são:

n_1	β **ou** β^1
13	5, 7, 8 ou 9

Finalmente, para se chegar aos resultados definitivos, aplica-se o Lemma 3.2 que assegura que no conjunto de soluções sempre estão presentes pares de inversos aditivos módulo n_1. Desta forma os números 7 e 9 são descartados uma vez que seus respectivos

inversos aditivos módulo 13 não atenderam previamente aos Teoremas 3.1 e 3.3 e não estão presentes. Logo os resultados de β, para a matriz de transformação linear que perfaz o entrelaçamento perfeito em uma matriz de dígitos com $n_1 = 13$, são:

n_1	β **ou** β^1
13	5 ou 8

Neste caso as matrizes de transformação linear, T, podem ser:

$$T = \begin{bmatrix} 1 & 5 \\ 0 & 8 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 8 \\ 0 & 5 \end{bmatrix}.$$

Exemplo 3: Considerando uma matriz de dígitos A com número de linhas igual a 25, ou seja, $n_1 = 25$, a matriz de transformação linear, T, deverá ter o seguinte formato:

$$T = \begin{bmatrix} 1 & -b \\ 0 & b \end{bmatrix}.$$

Existem $\phi(n_1) = \phi(25) = 20$ [32] possíveis valores de β (para $\beta < n_1$), a saber: 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23 e 24.

Da mesma forma que nos exemplos anteriores, será realizada uma filtragem destas possíveis soluções de β, a fim de encontrar aquelas que satisfazem aos Teoremas 3.1 e 3.3.

De acordo com o Teorema 3.3, o valor de β^1 deve satisfazer a expressão (3.16), mais uma vez reproduzida a seguir:

$|z|_L + |\beta^{-1} z\rangle - 1 + \sqrt{2n_1 - 1}$, em que $1 \le z \le -2 + \sqrt{2n_1 - 1}$.

Desta forma, para $n_1 = 25$, tem-se

$1 \leq z \leq 5$. Logo substituindo os valores de z na expressão (3.16) obtém-se as seguintes inequações:

$1+ \beta^1 > 6;$
$2+ 2\beta^1 > 6;$
$3+ 3\beta^1 > 6;$
$4+ 4\beta^1 > 6;$
$5+ 5\beta^1 > 6.$

Da faixa inicial considerada para possíveis valores β^1, apenas os números 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23 e 24 satisfazem as cinco inequações.

Finalmente para determinar os valores definitivos de β^1 emprega-se o Teorema 3.1, cuja expressão (3.5) é reproduzida a seguir:

$(x_2\text{-}x_1) + \beta^1(y_2\text{-}y_1) \not\equiv 0 \bmod n_1.$

Antes de empregar esta relação, deve-se determinar qual a faixa de valores para $(x_2\text{-}x_1)$ e $(y_2\text{-}y_1)$, para tal será usada a expressão (3.14) do Lemma 3.3 que calcula o peso da mancha compacta de erros e a partir deste ponto é possível definir a mancha de erros e as posições do dígitos. Esta expressão é reproduzida a seguir:

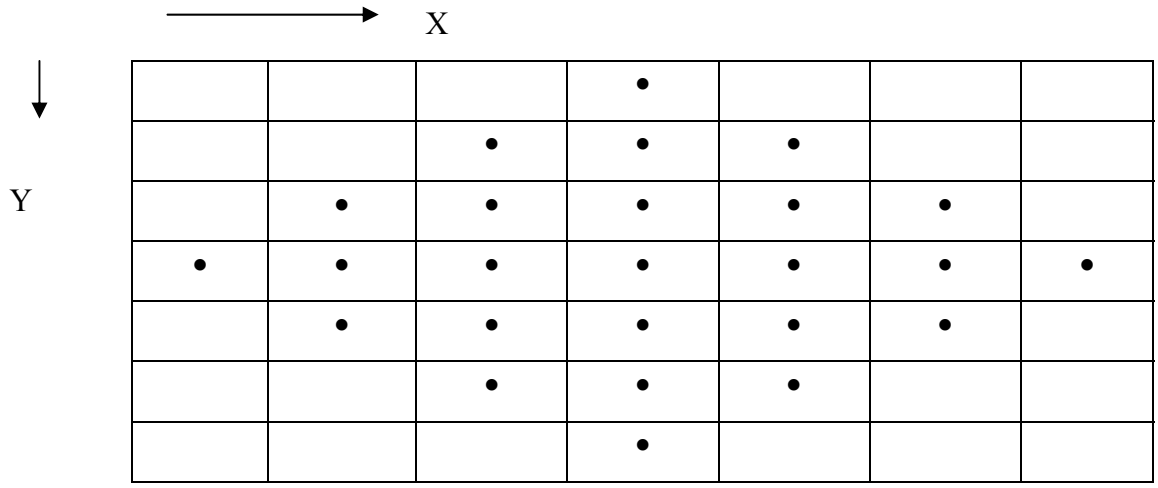
$$N_{d_L} = 1 + 2d_L(d_L + 1),$$

em que

$d_L \leq (-1 + \sqrt{2n_1 - 1})/2 = 3$, que corresponde à distância de Lee em relação ao centro da mancha de erros, então

$$N_{d_L} = 25.$$

Logo o formato da mancha de erros compacta passa a ser



Portanto neste caso irão ser consideradas as seguintes faixas para (x_2-x_1) e (y_2-y_1):

a) Faixa de (x_2-x_1):

$$0 \leq x_2\text{-}x_1 \leq 5.$$

De acordo com a ilustração anterior, esta faixa deveria se estender até 6, que ocorrem entre os dois dígitos localizados nos extremos da mancha de erros. No entanto quando x_2-x_1 = 6, o par de dígitos se encontra sobre a mesma linha da matriz e portanto y_2-y_1 = 0, anulando o termo β^1 na expressão (3.5).

b) Faixa de (y_2-y_1):

$$1 \leq y_2 - y_1 \leq 6.$$

Para a montagem das relações, baseadas na expressão (3.5), deve-se levar em conta que a máxima distância de Lee entre dois dígitos quaisquer da mancha de erros compacta, d_L-*mancha*, tem que ser no máximo igual a $-1+\sqrt{2n_1-1} = 6$, conforme o que está descrito na expressão (3.13) do Lemma 3.3. Desta forma deve ser observado que

$$(x_2\text{-}x_1) + (y_2\text{-}y_1) \leq 6.$$

Com base no exposto serão montadas as seguintes relações baseadas na expressão (3.5):

a) Para $x_2\text{-}x_1 = 0$

$$\beta^1 \not\equiv 0 \bmod 25;$$
$$2\,\beta^1 \not\equiv 0 \bmod 25;$$
$$3\,\beta^1 \not\equiv 0 \bmod 25;$$
$$4\,\beta^1 \not\equiv 0 \bmod 25;$$
$$5\,\beta^1 \not\equiv 0 \bmod 25;$$
$$6\,\beta^1 \not\equiv 0 \bmod 25.$$

b) Para $x_2\text{-}x_1 = 1$

$$1 + \beta^1 \not\equiv 0 \bmod 25;$$
$$1 + 2\,\beta^1 \not\equiv 0 \bmod 25;$$
$$1 + 3\,\beta^1 \not\equiv 0 \bmod 25;$$
$$1 + 4\,\beta^1 \not\equiv 0 \bmod 25;$$
$$1 + 5\,\beta^1 \not\equiv 0 \bmod 25.$$

c) Para $x_2\text{-}x_1 = 2$

$$2 + \beta^1 \not\equiv 0 \bmod 25;$$
$$2 + 2\,\beta^1 \not\equiv 0 \bmod 25;$$
$$2 + 3\,\beta^1 \not\equiv 0 \bmod 25;$$
$$2 + 4\,\beta^1 \not\equiv 0 \bmod 25.$$

d) Para $x_2\text{-}x_1 = 3$

$$3 + \beta^1 \not\equiv 0 \bmod 25;$$
$$3 + 2\,\beta^1 \not\equiv 0 \bmod 25;$$
$$3 + 3\,\beta^1 \not\equiv 0 \bmod 25.$$

e) Para x_2-x_1= 4

$$4 + \beta^1 \not\equiv 0 \bmod 25;$$
$$4 + 2\,\beta^1 \not\equiv 0 \bmod 25.$$

f) Para x_2-x_1= 5

$$5 + \beta^1 \not\equiv 0 \bmod 25.$$

Tendo por base estas relações, pode-se demonstrar que os números 6, 8, 11, 12, 16, 21, 22 e 23 não satisfazem a todas, logo os possíveis valores de β^1, e por tabela de β (porque ambos são relativamente primos a n_1), que permitem o entrelaçamento perfeito para n_1 =25 são:

n_1	β **ou** β^1
25	7, 9, 13, 14, 17, 18 ou 19

Finalmente, para se chegar aos resultados definitivos, aplica-se o Lemma 3.2 que assegura que no conjunto de soluções sempre estão presentes pares de inversos aditivos módulo n_1. Desta forma os números 9, 13, 14, 17 e 19 são descartados uma vez que seus respectivos inversos aditivos módulo 25 não atenderam previamente aos Teoremas 3.1 e 3.3 e não estão presentes. Logo os resultados de β, para a matriz de transformação linear que perfaz o entrelaçamento perfeito em uma matriz de dígitos com n_1 = 25, são:

n_1	β **ou** β^1
25	7 ou 18

Neste caso as matrizes de transformação linear, T, podem ser:

$$T = \begin{bmatrix} 1 & 7 \\ 0 & 18 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 18 \\ 0 & 7 \end{bmatrix}.$$

Em todos os casos exemplificados anteriormente observa-se que a mancha de erros compacta é ao mesmo tempo mancha de erros casada, uma vez que a expressão (3.13), ou seja, $d_L \leq -1 + \sqrt{2n_1 - 1}$, sempre resulta em um número inteiro. Nos exemplos representados a seguir a única preocupação do método é com a correção de manchas de erros compactas, uma vez que a expressão (3.13) não resulta em um número inteiro.

Exemplo 4: Considerando uma matriz de dígitos A com número de linhas igual a sete, ou seja, $n_1 = 7$, a matriz de transformação linear, T, deverá ter o seguinte formato:

$$T = \begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix}.$$

Existem $\phi(n_1) = \phi(7) = 6$ [32] possíveis valores de β (para $\beta < n_1$), a saber: 1, 2, 3, 4, 5 e 6.

Da mesma forma que nos exemplos anteriores, será realizada uma filtragem destas possíveis soluções de β, a fim de encontrar aquelas que satisfazem aos Teoremas 3.1 e 3.3.

De acordo com o Teorema 3.3, o valor de β^1 deve satisfazer a expressão (3.16) mais uma vez reproduzida a seguir:

$|z|_L + |\beta^{-1} z| \rangle - 1 + \sqrt{2n_1 - 1}$, em que $1 \leq z \leq -2 + \sqrt{2n_1 - 1}$.

Desta forma, para $n_1 = 7$, tem-se

$1 \leq z \leq 1{,}60$. Como z é um número inteiro, a inequação se converte em

$$1 \leq z \leq 1.$$

Logo substituindo os valores de z na expressão (3.16) obtém-se a seguinte inequação:

$$1 + \beta^1 > 2{,}60.$$

Da faixa inicial considerada para possíveis valores β^1, apenas os números 2, 3, 4, 5 e 6 satisfazem a referida inequação.

Finalmente para determinar os valores definitivos de β^1 emprega-se o Teorema 3.1, cuja expressão (3.5) é reproduzida a seguir:

$(x_2\text{-}x_1) + \beta^1(y_2\text{-}y_1) \not\equiv 0 \bmod n_1$.

Antes de empregar esta relação, deve-se determinar qual a faixa de valores para $(x_2\text{-}x_1)$ e $(y_2\text{-}y_1)$, para tal será usada a expressão (3.14) do Lemma 3.3 que calcula o peso da mancha compacta de erros e, a partir deste ponto, é possível definir a mancha de erros e as posições do dígitos. Esta expressão é reproduzida a seguir:

$$N_{d_L} = 1 + 2d_L(d_L + 1),$$

em que

$$d_L \leq (-1 + \sqrt{2n_1 - 1}) / 2 = 1{,}30,$$

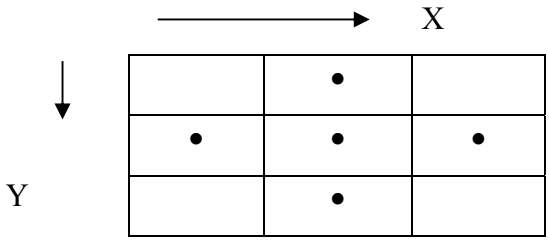
que corresponde à distância de Lee em relação ao centro da mancha de erros. Como a distância de Lee corresponde a um número inteiro, a inequação é reescrita como

$$d_L \leq 1.$$

Então

$$N_{d_L} = 5.$$

Logo o formato da mancha de erros compacta passa a ser



Portanto neste caso irão ser consideradas as seguintes faixas para (x_2-x_1) e (y_2-y_1):

a) Faixa de (x_2-x_1):

$$0 \leq x_2\text{-}x_1 \leq 1.$$

De acordo com a ilustração anterior, esta faixa deveria se estender até 2, que ocorrem entre os dois dígitos localizados nos extremos da mancha de erros. No entanto quando x_2-x_1 = 2, o par de dígitos se encontra sobre a mesma linha da matriz e portanto y_2-y_1 = 0, anulando o termo β^1 na expressão (3.5).

b) Faixa de (y_2-y_1):

$$1 \leq y_2 – y_1 \leq 2.$$

Para a montagem das relações, baseadas na expressão (3.5), deve-se levar em conta que a máxima distância de Lee entre dois dígitos quaisquer da mancha de erros compacta, d_L-*mancha*, tem que ser no máximo igual a $-1+\sqrt{2n_1-1} = 2{,}60,$ conforme a expressão (3.13) do Lemma 3.3. Considerando que este valor só pode ser um número inteiro, então $d_L=2$. Desta forma deve ser observado que

$$(x_2\text{-}x_1) + (y_2\text{-}y_1) \leq 2.$$

Com base no exposto serão montadas as seguintes relações baseadas na expressão (3.5):

a) Para x_2-x_1= 0

$$\beta^1 \not\equiv 0 \bmod 7;$$
$$2\,\beta^1 \not\equiv 0 \bmod 7.$$

b) Para x_2-x_1= 1

$$1 + \beta^1 \not\equiv 0 \bmod 7.$$

Tendo por base estas relações, pode-se demonstrar que o número 6 não satisfaz a todas, logo os possíveis valores de β^1, e por tabela de β (porque ambos são relativamente primos a n_1), que permitem o entrelaçamento perfeito para n_1 =7 são:

n_1	β **ou** β^1
7	2, 3, 4 ou 5

Observa-se que estes valores atendem ao Lemma 3.2 que assegura que no conjunto de soluções sempre estão presentes pares de inversos aditivos módulo n_1, ou seja, 2 é inverso aditivo de 5 e 3 é inverso aditivo de 4.

Neste caso as matrizes de transformação linear, T, para obtenção do entrelaçamento perfeito, são:

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 5 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 5 \\ 0 & 2 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 4 \end{bmatrix},$$

ou

$$T = \begin{bmatrix} 1 & 4 \\ 0 & 3 \end{bmatrix}.$$

A seguir é feita uma ilustração do processo de entrelaçamento unidimensional de uma matriz de dígitos com dimensão 7 x 9, n_1 =7 e n_2 =9, usando a seguinte matriz de transformação linear, T, que assegura o entrelaçamento perfeito:

$$T = \begin{vmatrix} 1 & 2 \\ 0 & 5 \end{vmatrix}.$$

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9

(a)

A_1	B_2	C_3	D_4	E_5	F_6	G_7	A_8	B_9
D_1	E_2	F_3	G_4	A_5	B_6	C_7	D_8	E_9
G_1	A_2	B_3	C_4	D_5	E_6	F_7	G_8	A_9
C_1	D_2	E_3	F_4	G_5	A_6	B_7	C_8	D_9
F_1	G_2	A_3	B_4	C_5	D_6	E_7	F_8	G_9
B_1	C_2	D_3	E_4	F_5	G_6	A_7	B_8	C_9
E_1	F_2	G_3	A_4	B_5	C_6	D_7	E_8	F_9

(b)

Figura 3.8: (a) Matriz de dígitos A com dimensão 7 x 9;

(b) Matriz de dígitos entrelaçada A_u exibindo a mancha de erros compacta.

Neste exemplo fica evidenciada a necessidade de se restringir a capacidade de correção de erros, deste método proposto, ao tipo de mancha denominada compacta, d_L-*mancha,* como a que está exposta na Figura 3.8.b, uma vez que nem sempre é possível a correção de manchas de erros consideradas casadas.

Para constatar este fato, vê-se na Figura 3.8.b que a mancha de erros casada formada pela mancha compacta hachurada acrescida dos símbolos B_4 e E_6 pode ser corrigida, diferentemente da mancha casada formada pela mesma mancha compacta acrescida dos dígitos C_4 e D_6.

Por último, será mostrado a seguir um exemplo em que não existem valores de β^1 que garantam o entrelaçamento perfeito. Em geral isto ocorre porque as possíveis soluções não são relativamente primas com o valor de n_1.

Exemplo 5: Considerando uma matriz de dígitos A com número de linhas igual a seis, ou seja, $n_1 = 6$, a matriz de transformação linear, T, deverá ter o seguinte formato:

$$T = \begin{bmatrix} 1 & -\beta \\ 0 & \beta \end{bmatrix}.$$

Existem $\phi(n_1) = \phi(6) = 2$ [32] possíveis valores de β (para $\beta < n_1$), a saber: 1 e 5.

Da mesma forma que nos exemplos anteriores, será realizada uma filtragem destas possíveis soluções de β, a fim de encontrar aquelas que satisfazem aos Teoremas 3.1 e 3.3.

De acordo com o Teorema 3.3, o valor de β^1 deve satisfazer a expressão (3.16) mais uma vez reproduzida a seguir:

$|z|_L + |\beta^{-1} z| \rangle - 1 + \sqrt{2n_1 - 1}$, em que $1 \le z \le -2 + \sqrt{2n_1 - 1}$.

Desta forma, para $n_1 = 6$, tem-se

$1 \le z \le 1{,}31$. Como z é um número inteiro, a inequação se converte em

$$1 \le z \le 1.$$

Logo substituindo os valores de z na expressão (3.16) obtém-se a seguinte inequação:

$$1+ \beta^1 > 2{,}31.$$

Da faixa inicial considerada para possíveis valores β^1, apenas o número 5 satisfaz a referida inequação.

Finalmente para determinar os valores definitivos de β^1 emprega-se o Teorema 3.1, cuja expressão (3.5) é reproduzida a seguir:

$(x_2\text{-}x_1) + \beta^1(y_2\text{-}y_1) \not\equiv 0 \bmod n_1$.

Antes de empregar esta relação, deve-se determinar qual a faixa de valores para $(x_2\text{-}x_1)$ e $(y_2\text{-}y_1)$, para tal será usada a expressão (3.14) do Lemma 3.3 que calcula o peso da mancha compacta de erros e, a partir deste ponto, é possível definir a mancha de erros e as posições do dígitos. Esta expressão é reproduzida a seguir:

$$N_{d_L} = 1 + 2d_L(d_L + 1),$$

em que

$$d_L \le (-1 + \sqrt{2n_1 - 1})/2 = 1{,}15,$$

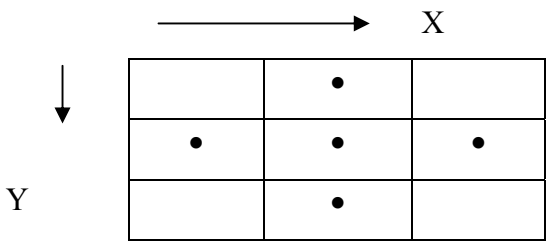
que corresponde à distância de Lee em relação ao centro da mancha de erros. Como a distância de Lee corresponde a um número inteiro, a inequação é rescrita como

$$d_L \le 1,$$

então

$$N_{d_L} = 5.$$

Logo o formato da mancha de erros compacta passa a ser



Portanto neste caso irão ser consideradas as seguintes faixas para (x_2-x_1) e (y_2-y_1):

a) Faixa de (x_2-x_1):

$$0 \leq x_2\text{-}x_1 \leq 1.$$

De acordo com a ilustração anterior, esta faixa deveria se estender até 2, que ocorrem entre os dois dígitos localizados nos extremos da mancha de erros. No entanto quando x_2-x_1 = 2, o par de dígitos se encontra sobre a mesma linha da matriz e portanto y_2-y_1 = 0, anulando o termo β^1 na expressão (3.5).

b) Faixa de (y_2-y_1):

$$1 \leq y_2 - y_1 \leq 2.$$

Para a montagem das relações, baseadas na expressão (3.5), deve-se levar em conta que a máxima distância de Lee entre dois dígitos quaisquer da mancha de erros compacta, d_L-*mancha*, tem que ser no máximo igual a $-1+\sqrt{2n_1-1} = 2{,}31$, conforme a expressão (3.13) do Lemma 3.3. Considerando que este valor só pode ser um número inteiro, então $d_L=2$. Desta forma deve ser observado que

$$(x_2\text{-}x_1) + (y_2\text{-}y_1) \leq 2.$$

Com base no exposto serão montadas as seguintes relações baseadas na expressão (3.5):

a) Para x_2-x_1= 0

$$\beta^1 \not\equiv 0 \bmod 6;$$
$$2\,\beta^1 \not\equiv 0 \bmod 6.$$

b) Para x_2-x_1= 1

$$1 + \beta^1 \not\equiv 0 \bmod 6.$$

Tendo por base estas relações, pode-se demonstrar que o número 5 não satisfaz a todas, logo não existem soluções possíveis para β^1, e por tabela de β, que permitem o entrelaçamento perfeito para n_1=6, logo

n_1	β ou β^1
6	Não existe solução

Com base neste roteiro, descrito nos exemplos anteriores, foi desenvolvido um programa de computador capaz de calcular os valores de $\beta^{1.}$ A Tabela gerada que foi gerada está mostrada no Apêndice II.

3.6 Cálculo da Eficiência

Baseando-se no artigo de Almeida [26], pode-se expressar o cálculo da eficiência deste método pela relação:

$$G_e = (t_{com}/t_{sem}). \tag{3.25}$$

Em que

G_e – Ganho de entrelaçamento.
t_{com} – Capacidade de correção de erros com o uso de entrelaçamento.
t_{sem} – Capacidade de correção de erros sem o uso de entrelaçamento.

Este ganho pode ser expresso da seguinte forma:

$$G_e = 2.\, d_L + 1. \qquad (3.26)$$

Sendo d_L a distância de Lee tomada em relação ao centro da mancha compacta de erros.

É importante ressaltar que esta expressão é válida tanto para códigos matriciais para a correção de uma única mancha de erros compacta, quanto para códigos matriciais para a correção de múltiplas manchas de erros. Esta afirmação é ilustrada nos exemplos seguintes.

Exemplo 1: Código matricial com entrelaçamento unidimensional usando códigos corretores de erros aleatórios em cada linha da matriz de dígitos, com capacidade de correção de um único erro aleatório.

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

Neste exemplo observa-se que se não fosse pelo entrelaçamento haveria a necessidade de códigos corretores de erros aleatórios com capacidade de correção de três erros em cada linha da matriz de dígitos. Como é usado o entrelaçamento basta o emprego de um código corretor de erros aleatórios, por linha, com capacidade de correção de um único erro. Desta maneira consegue-se uma eficiência ou ganho de entrelaçamento igual a três.

Exemplo 2: Código matricial com entrelaçamento unidimensional usando códigos corretores de erros aleatórios em cada linha da matriz de dígitos, com capacidade de correção de dois erros aleatórios.

□1	∇2	♣3	♦4	♥5	□6	∇7	♣8	♦9	♥10
♦1	♥2	□3	∇4	♣5	♦6	♥7	□8	∇9	♣10
∇1	♣2	♦3	♥4	□5	∇6	♣7	♦8	♥9	□10
♥1	□2	∇3	♣4	♦5	♥6	□7	∇8	♣9	♦10
♣1	♦2	♥3	□4	∇5	♣6	♦7	♥8	□9	∇10

Neste segundo exemplo observa-se que se não fosse pelo entrelaçamento haveria a necessidade de códigos corretores de erros aleatórios com capacidade de correção de seis erros em cada linha da matriz de dígitos. Como é usado o entrelaçamento basta o emprego de um código corretor de erros aleatórios, por linha, com capacidade de correção de dois erros. Desta maneira consegue-se uma eficiência ou ganho de entrelaçamento igual a três.

Com base na expressão (3.26) é possível montar a Tabela a seguir ilustrando alguns valores de ganho para alguns casos mencionados anteriormente.

n_1	G_e
5	3
7	3
13	5
25	7

Tabela 3.1 – Valores de ganho de entrelaçamento, G_e, para alguns valores de n_1.

Observa-se que este ganho se torna mais expressivo à medida que o número de linhas da matriz de dígitos cresce.

3.7 Conclusão

Foi introduzido em [35] o entrelaçamento unidimensional como uma técnica prática para corrigir múltiplas manchas de erros em uma matriz de dígitos n_1 x n_2. Arbitrariamente foi decidido focalizar sobre manchas de erros compactas. Esta decisão se mostrou recompensadora, uma vez que tornou possível a especificação do valor de β para classes de manchas de erros. No entrelaçamento convencional usado para combater surtos de erros em transmissão de dados seriais, uma matriz de dígitos n_1 x n_2 contém palavras-código nas linhas e os dígitos da matriz de dígitos são extraídos pela leitura das colunas. O entrelaçamento convencional permite a correção de no máximo t surtos de erros de comprimento n_1. A eficiência do entrelaçamento unidimensional, medida em termos de redundância de código empregado e o número de erros que podem ser corrigidos é, pelo menos, a mesma que do entrelaçamento convencional, uma vez que quaisquer t manchas de erros casadas em cada

matriz de dígitos são corrigíveis, como podem ser também algumas outras manchas de erros. O resultado no Teorema 3.1 pode ser estendido para cobrir as condições associadas com o entrelaçamento perfeito como definido em [26].

Capítulo 4

Código Matricial para Gravação Magnética Usando Entrelaçamento Unidimensional e Códigos Corretores de Surtos de Erros

4.1 Introdução

É proposto no capítulo anterior um código matricial com dimensão n_1 x n_2, com as n_1 linhas constituídas por palavras-código de um código de correção de erros aleatórios, do tipo MDS (com máxima distância de Hamming), como por exemplo os códigos BCH ou Reed-Solomon. Em [23] são propostos códigos matriciais usando entrelaçamento unidimensional, com uma capacidade maior de correção de surtos bidimensionais ou manchas de erros. Com este objetivo são empregados códigos corretores de surtos de erros nas linhas da matriz de dígitos e, conjuntamente, é feita uma pequena adaptação na matriz de transformação linear, T.

São considerados códigos matriciais n_1 x n_2 quadrados, ou seja, $n_1 = n_2 = n$, em que n é um inteiro positivo, e, para efeito de correção, são também consideradas manchas de erros (*patches*) no formato retangular ou formato quadrado, observadas ciclicamente nesta referida matriz de dígitos.

4.2 Definições e Teoria Básica

Considera-se como um código de bloco linear (n,k,b) aquele de comprimento n, dimensão k e capacidade de correção de surtos de erros de comprimento b [36]. Será considerada uma matriz de dígitos n_1 x n_2 de dimensão n x n, cujas n linhas são palavras-código de um código

corretor linear de blocos (n,k,b). As posições dos dígitos da matriz são designadas pelos pares (x_i, y_i), em que $0 \le x_i \le n$-1 e $0 \le y_i \le n$-1, ou seja, em ambas, as coordenadas variam de 0 a n-1.

Para propósitos tanto de armazenamento quanto de transmissão esta matriz de dígitos é entrelaçada. Ocorrendo então a troca das posições dos dígitos em cada coluna, sendo este deslocamento considerado cíclico, com o giro realizado para baixo.

Definição 4.1: Define-se uma matriz de transformação linear, T, como

$$T = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}. \tag{4.1}$$

Esta matriz de transformação executa o entrelaçamento das posições do dígito (x_i, y_i), $0 \le x_i \le n$-1 e $0 \le y_i \le n$-1, em uma matriz de dígitos quadrada, A, de dimensões n x n, por meio de uma transformação linear, como segue.

$$(x_i, y_i).\begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = (x_i, x_i - y_i) = (x_i', y_i'), \tag{4.2}$$

em que (x_i', y_i'), $0 \le x_i' \le n$-1 e $0 \le y_i' \le n$-1, corresponde às coordenadas do dígito na matriz entrelaçada, A_u.

A transformação anterior preserva a primeira coordenada, ou seja, $x_i = x_i'$, e atua somente na segunda coordenada.

Definição 4.2: Define-se uma mancha de erros compacta em uma matriz de dígitos n x n como sendo qualquer padrão de erros bidimensional de peso máximo $b.n$ cujas posições são cobertas por uma mancha retangular b x n na matriz n x n.

Para os propósitos deste tipo de código matricial com entrelaçamento unidimensional e que emprega, nas suas linhas, códigos corretores de surtos de erros, decidiu-se pela modificação da definição de manchas de erros compactas, uma vez que a definição expressa no capítulo anterior não é adequada a esta nova forma de construção do código. Mas, mesmo

assim, é importante restringir a ação deste código a um determinado tipo de mancha, designada como mancha de erros compacta, uma vez que se estabelece o escopo de ação do código proposto. Todavia é possível que outros tipos de manchas de erros, além das manchas de erros compactas, possam também ser corrigidas por este esquema, o que se torna, mais uma vez, um bônus extra e não um resultado direto desta proposição.

Lemma 4.1: A matriz de transformação linear para desentrelaçamento T^{-1} obtida de T, é dada por

$$T^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}. \tag{4.3}$$

Prova: Imediatamente é verificado que $T.T^{-1} = I_2$, em que I_2 é a matriz unitária de ordem 2.

4.3 Espalhamento e Correção

O entrelaçamento perfeito de uma dada mancha de erros, em uma matriz de dígito quadrada n x n, pode somente ser conseguido se o seu máximo peso não exceder n. Contudo, caso se use um código corretor de surtos de erros, do tipo (n,k,b), para codificar as linhas da matriz de dígitos (antes do entrelaçamento), não deve existir uma preocupação em se ter o entrelaçamento perfeito. Tudo o que precisa ser garantido é que, depois do desentrelaçamento, cada linha da matriz de dígitos contenha no máximo um surto de erros de comprimento b, conforme a definição a seguir.

Definição 4.5: Define-se perfeito espalhamento de um surto de erros pertencente a uma mancha de erros de peso máximo $b.n$, associado com uma matriz de dígitos A_u de dimensão n x n, obtida por entrelaçamento unidimensional, como sendo aquele em que, após o desentrelaçamento, dois ou mais surtos de erros de comprimento b pertencentes à mancha não ocupam a mesma linha na matriz de dígitos resultante.

A seguir será estabelecida uma relação entre o inteiro b, que corresponde ao comprimento de surto de erros, e o padrão da mancha de erros que pode ser corrigida em uma dada matriz de dígitos n x n, satisfazendo a condição de perfeito espalhamento de surto de erros.

Teorema 4.1: Seja A_u uma matriz de dígitos n x n entrelaçada por meio de T, possivelmente corrompida por manchas de erros. Seja T^{-1} a transformação linear de desentrelaçamento. Um perfeito espalhamento do surto de erros existe para todas as manchas de erros cujas posições são cobertas por uma região retangular b x n em A_u, considerada ciclicamente nas duas dimensões.

Prova: Deve-se notar que a matriz de transformação T, quando aplicada à matriz A, causa na diagonal principal de A_u o aparecimento de uma palavra-código correspondente à linha 0 da matriz original A, e cada diagonal paralela à principal também terá palavras-código distintas, visto ciclicamente. Desta forma qualquer retângulo de dimensão b x n em A_u, considerado ciclicamente nas duas direções, conterá no máximo b dígitos consecutivos ocupando a mesma linha após o desentrelaçamento. Esta é a condição requerida para o código de linha (n,k,b) para corrigir qualquer surto de erros de comprimento b.

Teorema 4.2: Seja A_u uma matriz de dígitos n x n obtida de uma matriz de dígitos A pela transformação linear T, possivelmente corrompida por manchas de erros. Considerando que as linhas de A são palavras-código de um código corretor de blocos (n,k,b) capaz de corrigir t surtos de erros de comprimento b. Por meio da operação de desentrelaçamento T^{-1} qualquer padrão de no máximo t manchas de erros ciclicamente cobertos por uma região b x n em A_u é corrigível.

Prova: Dado que a transformação linear T é reversível, conclui-se que haverá após o desentrelaçamento no máximo t surtos de erros de comprimento máximo b em cada linha, desde que t ou menos manchas de erros cobertas por regiões b x n, que podem se sobrepor, afetem a matriz A_u de dimensão n x n. Desta forma um código corretor de t-surtos de erros, que tenham comprimento máximo b, seria capaz de corrigir os erros nas linhas.

Exemplo:

Considerando uma matriz de dígitos 7 x 7 mostrada na Figura 4.1a, cujas linhas são consideradas palavras-código de um código corretor de surto de erros (7,3,2) com comprimento de surto, *b*, igual a dois. A matriz de dígitos resultante do entrelaçamento, após a aplicação da transformação, é mostrada na Figura 4.1b. Observa-se que qualquer mancha de erros coberta pelo retângulo *r* x *s,* em que $r = 2$ e $1 \leq s \leq 7$, na Figura 4.1b é espalhada depois do desentrelaçamento de uma maneira que não se permita que dois ou mais surtos de erros de comprimento máximo igual a dois ocupem a mesma linha na matriz resultante do desentrelaçamento. Por exemplo a mancha formada pelos dígitos □$_3$, Δ_4, ♣$_5$, ♦$_6$ e ♥$_7$, ♠$_3$, □$_4$, Δ_5, ♣$_6$ e ♦$_7$ na Figura 4.1b está coberta por um retângulo 2 x 7 e após o desentrelaçamento é espalhada em seis distintas linhas, sendo que cada linha com no máximo dois erros consecutivos. Este é um exemplo de entrelaçamento perfeito.

	0	1	2	3	4	5	6
0	□$_1$	□$_2$	□$_3$	□$_4$	□$_5$	□$_6$	□$_7$
1	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
2	♣$_1$	♣$_2$	♣$_3$	♣$_4$	♣$_5$	♣$_6$	♣$_7$
3	♦$_1$	♦$_2$	♦$_3$	♦$_4$	♦$_5$	♦$_6$	♦$_7$
4	♥$_1$	♥$_2$	♥$_3$	♥$_4$	♥$_5$	♥$_6$	♥$_7$
5	∇_1	∇_2	∇_3	∇_4	∇_5	∇_6	∇_7
6	♠$_1$	♠$_2$	♠$_3$	♠$_4$	♠$_5$	♠$_6$	♠$_7$

(a)

	0	1	2	3	4	5	6
0	□1	Δ2	♣3	♦4	♥5	∇6	♠7
1	♠1	□2	Δ3	♣4	♦5	♥6	∇7
2	∇1	♠2	□3	Δ4	♣5	♦6	♥7
3	♥1	∇2	♠3	□4	Δ5	♣6	♦7
4	♦1	♥2	∇3	♠4	□5	Δ6	♣7
5	♣1	♦2	♥3	∇4	♠5	□6	Δ7
6	Δ1	♣2	♦3	♥4	∇5	♠6	□7

(b)

Figura 4.1: (a) Matriz de dígitos 7 x 7; (b) Versão entrelaçada desta matriz.

4.4 Tipos de Manchas de Erros Corrigíveis

Este método proposto neste trabalho visa essencialmente a correção de manchas de erros compactas em uma matriz de dígitos n x n, que segundo a definição 4.4 correspondem a qualquer padrão de erro bidimensional de peso máximo $b.n$, cujas posições são cobertas por uma mancha retangular b x n na matriz n x n.

É possível também corrigir outros padrões de erro que não necessariamente estejam contidos no retângulo b x n, bastando que qualquer diagonal contida na mancha, vista ciclicamente, não possua mais que b posições com erros pertencentes à mesma linha da matriz original (sem entrelaçamento).

Exemplo: Supõe-se um código matricial 7 x 7, cujas linhas são palavras-código de um código corretor de surtos de erros do tipo (7,3,2) capaz de corrigir surtos de erros com comprimento b de no máximo dois. Neste caso é possível corrigir os tipos de manchas de erros mostrados a seguir.

	0	1	2	3	4	5	6
0	□1	Δ2	♣3	♦4	♥5	∇6	♠7
1	♠1	□2	Δ3	♣4	♦5	♥6	∇7
2	∇1	♠2	□3	Δ4	♣5	♦6	♥7
3	♥1	∇2	♠3	□4	Δ5	♣6	♦7
4	♦1	♥2	∇3	♠4	□5	Δ6	♣7
5	♣1	♦2	♥3	∇4	♠5	□6	Δ7
6	Δ1	♣2	♦3	♥4	∇5	♠6	□7

(a)

	0	1	2	3	4	5	6
0	□1	Δ2	♣3	♦4	♥5	∇6	♠7
1	♠1	□2	Δ3	♣4	♦5	♥6	∇7
2	∇1	♠2	□3	Δ4	♣5	♦6	♥7
3	♥1	∇2	♠3	□4	Δ5	♣6	♦7
4	♦1	♥2	∇3	♠4	□5	Δ6	♣7
5	♣1	♦2	♥3	∇4	♠5	□6	Δ7
6	Δ1	♣2	♦3	♥4	∇5	♠6	□7

(b)

Figura 4.2: (a) Mancha de erros compacta contida no retângulo *b* x *n*;

(b) Mancha de erros não contida no retângulo *b* x *n* mas que é passível de correção.

4.5 Cálculo da Eficiência

Diferentemente do caso anterior, não é possível usar o método de cálculo de eficiência estabelecido no artigo de Almeida [26], uma vez que o tipo de mancha de erros compacta, que é o alvo principal de correção por este método, pode ocupar toda a linha correspondente da

matriz e, portanto, não é possível usar códigos corretores de erros aleatórios que sejam capazes de corrigir todos os dígitos contidos nas linhas.

Desta forma, o procedimento a ser empregado para mensurar a eficiência deste método é baseado na comparação da capacidade total de correção de erros dos métodos propostos nesta dissertação. Para ilustrar este procedimento pode-se usar o exemplo a seguir.

Exemplo: Supõe-se que esteja sendo usado um código matricial 7 x 7, tendo as linhas codificadas por um código corretor de surto de erros (7,3,2). Para este caso o código matricial seria capaz de corrigir uma mancha de erros de tamanho 2 x 7, o que totalizaria um máximo de 14 erros nesta matriz.

Caso fosse utilizado para codificar as linhas um código corretor de erros aleatórios, como por exemplo um código BCH, este teria dimensão 3, sendo capaz de corrigir no máximo um único erro por linha e, desta forma, totalizaria até sete erros passíveis de correção dentro da matriz.

Com base neste exemplo, percebe-se que houve um ganho significativo com relação ao método anterior no que tange à capacidade de correção de erros, uma vez que neste caso os códigos corretores de surtos de erros são mais eficazes do que os códigos corretores de erros aleatórios.

Para consolidar estas informações, vide outro exemplo.

Exemplo: Seja um código matricial com dimensão 15 x 15. Caso seja utilizado nas linhas um código corretor de surtos com capacidade de correção de surtos de erros de comprimento igual a cinco, o código deverá ter a característica (15,5,5). Para este caso é possível a correção de manchas de erros contendo até 15 x 5 = 75 erros. Caso fosse usado código corretor de erros aleatórios, como por exemplo o BCH, com dimensão igual a cinco, este código só teria capacidade de corrigir no máximo três erros, o que indicaria uma capacidade total de 15 x 3 = 45 erros na matriz.

Desta forma observa-se mais uma vez que a opção pelo uso de códigos corretores de surtos de erros, nas linhas das palavras-código de um dado código matricial, garante uma eficiência maior.

No entanto é importante observar que os métodos propostos exercem o papel de correção de manchas de erros compactas que são definidas sob critérios diferenciados. Isto faz com que se tenha que optar por um deles, dependendo do tipo de mancha de erros que se queira combater num dado sistema de gravação ou armazenamento de dados.

4.6 Conclusão

Foi introduzido em [23] o uso de códigos corretores de surtos de erros com entrelaçamento unidimensional como uma técnica prática para corrigir múltiplas manchas de erros em uma matriz de dígitos n_1 x n_2, em que $n_1 = n_2 = n$. Arbitrariamente foi decidido focalizar sobre manchas de erros compactas. No entrelaçamento comum usado para combater surtos de erros em transmissão de dados seriais, uma matriz de dígitos n_1 x n_2 contém palavras-código nas linhas com capacidade de correção de t erros e os dígitos da matriz de dígitos são extraídos pela leitura das colunas. O entrelaçamento ordinário permite a correção de no máximo t surtos de erros de comprimento n_1.

A eficiência deste esquema comparada com o entrelaçamento comum, vem do fato que, para valores fixos de n e k, usualmente o número b (comprimento do surto de erros capaz de ser corrigido) é maior do que t (capacidade de correção de erros aleatórios). Consequentemente, para a correção de manchas compactas de erros de um dado peso, requer-se menores matrizes quando se utiliza códigos corretores de surtos de erros. Por outro lado, fixando a dimensão da matriz, observa-se que códigos corretores de surtos de erros são capazes de "atacar" manchas compactas de erros com muito mais erros do que é possível com códigos corretores de erros aleatórios.

Capítulo 5

Códigos Matriciais com Entrelaçamento Unidimensional Usados em Criptografia de Chave Secreta

5.1 Introdução

A primeira cifra de blocos baseada em códigos corretores de erro foi a cifra de chave pública proposta por McEliece [37] em 1978 e permanece segura até hoje. McEliece propôs um criptosistema de chave pública que usava códigos Goppa [39] para correção de t erros, baseado no fato que existem algoritmos eficientes de decodificação para tais códigos, o que não é verdadeiro para um código linear em geral. Entretanto esta cifra é de difícil implementação.

Rao e Nam [38] propuseram a primeira cifra de chave secreta baseada em códigos de correção de erros mais simples. Contrastando em termos de segurança com a cifra de McEliece, alguns autores mostraram como quebrar esta cifra por meio de ataque de texto claro e propuseram duas versões modificadas à cifra original que combatiam o tipo de ataque considerado. Estas cifras foram posteriormente quebradas por Hin [42] e Struik- Tilburg [43].

Em [40] o uso de códigos corretores de surtos de erros foi introduzido para criptosistemas de chave secreta. O esquema é baseado no fato de que a capacidade de correção de um código corretor de surtos de erros é em geral maior que a capacidade de correção de códigos corretores de erros aleatórios. Campello de Souza [41] propôs esquema usando esta técnica obtendo novos resultados referentes à segurança e sugeriu uma classe de códigos para implementação da técnica. O artigo é baseado no uso de códigos matriciais muito simples e eficientes que têm uma capacidade de correção de erros aleatórios fixa (t =1) e capacidade de correção de surtos de erros arbitrariamente maior.

5.2 Criptosistema de Chave Secreta Baseado em Códigos Corretores de Erros

O processo de cifragem e decifragem de criptosistemas de chave secreta é resumidamente descrito a seguir.

Denota-se por G a matriz geradora do código q-*ário* C (n,k,d) na forma padrão, sendo $G=[I_k: g]$, em que I_k e g são respectivamente, a matriz identidade k x k e uma matriz k x (n-k). Quando G está na forma padrão é comum aplicar um embaralhamento nos blocos do texto claro μ executando o produto μS, em que S é uma matriz densa e não-singular. O processo de cifragem do texto claro consiste dos seguintes passos:

1. Segmentar o texto claro em μ blocos, de k dígitos cada, e, em seguida, embaralhar cada bloco executando o produto μS.
2. Codificar cada conjunto de k dígitos dos blocos embaralhados para produzir palavras-código η do código C, cuja matriz geradora é denotada por G, executando a operação μSG, ou seja, produzir a n-*úpla* $\eta = \mu SG$.
3. Adicionar à palavra-código η o vetor de erro $\boldsymbol{e_r}$, o qual é produzido por um gerador de erros, para obter $r_c = \eta + \boldsymbol{e_r}$.
4. Finalmente o criptograma y_c é obtido pela permutação das coordenadas da n-*úpla* r_c por meio do produto $r_c P$, ou seja, $y_c = r_c P$, em que P é uma matriz de permutação n x n.

O processo de decifragem do criptograma y_c consiste dos seguintes passos:

1. Aplicar a permutação inversa P^{-1} ao criptograma y_c com o objetivo de recuperar r_c.
2. Decodificar r_c, obtido no passo 1, pelo emprego do decodificador para o código C, o qual reproduz em sua saída o texto claro embaralhado μS.
3. Multiplicar o texto claro embaralhado μS por S^{-1} para extrair o texto claro μ.

A fim de ilustrar este esquema, passa-se a descrever o método utilizado por Campello de Souza [41] e Rocha [44]. Ao final, será proposto um modelo usando o código estudado nesta dissertação.

Campello de Souza [41] usou um código de blocos linear C (n, k, b) de comprimento n, dimensão k e capaz de corrigir surtos de erros de comprimento b. É presumido que $b > t$, em que t é a capacidade de correção de erros aleatórios do código, expressa por $t = (d-1)/2$. Com G denotando a matriz geradora de C (n, k, b), as operações de cifragem e decifragem são descritas abaixo. O texto cifrado y_c é uma *n-úpla* binária, o texto claro μ é uma k-upla binária e P é uma matriz de permutação n x n.

(i) Processo de cifragem: O texto cifrado y_c é calculado do texto claro μ usando a relação $y_c = (\mu G + E_{l,w}).P$, em que $E_{l,w}$ representa um surto aleatório de erros de comprimento l e peso w. Neste contexto um surto de erros de comprimento l e peso w é uma *n-úpla* binária cujos w componentes não nulos são confinados a l posições consecutivas, o primeiro e o último dos quais são não nulos. Será assumido que $w_{min} \leq w \leq l \leq b$, em que w_{min} é um número fixo e maior do que t.

(ii) Processo de decifragem: μ é recuperado de y_c. Primeiramente calcula-se $y_c^{'} = yP^{-1} = \mu G + E_{l,w}$. Para se obter μ de $y_c^{'}$ usa-se o algoritmo de decodificação do código C (n, k, b).

(iii) Processo de Criptoanálise: Campello de Souza [41] sugeriu a criptoanálise que é típica para criptosistemas baseados em códigos corretores de erros. Primeiramente o algoritmo de cifragem é reescrito como

$$C = (\mu G + E_{l,w}).P = \mu G^{'} + E^{'}_{l,w} \tag{5.1}$$

em que

$G^{'} = GP$ e $E^{'}_{l,w} = E_{l,w}.P$.

A matriz $G^{'}$ pode ser descoberta por ataque de texto claro conhecido, similar a aquele usado em [38]. O criptoanalista escolhe um texto claro da forma μ_i = (00....010..00) com somente um "1" na i-ésima posição para i = 1,....., k. Cifra-se μ_i um número de vezes e obtem-se um valor estimado de $g^{'}_i$, a i-ésima coluna da matriz $G^{'}$ com um desejado grau de certeza. Repetindo este passo para i = 1,....., k obtem-se a matriz $G^{'}$.

Neste esquema, a princípio, é impossível decodificar usando $G^{'}$, porque esta corresponde a uma matriz geradora de um código de blocos linear de formato geral, capaz de corrigir até t erros aleatórios, e $E^{'}_{l,w}$ é um erro aleatório de peso $w > t$.

Portanto conhecendo G' o criptoanalista tem duas escolhas:

(a) Encontrando G e P de $G^{'}$: Neste caso a segurança depende do número, N_B, de códigos corretores de surto de erros que são combinatoriamente equivalentes para um dado conjunto de parâmetros, isto é, b, n, d e k. Em cada julgamento, o criptoanalista testa uma das N_B permutações que conduz G' para G'', a matriz geradora de um dos N_B códigos da classe de códigos corretores de surto de erros. Em cada teste o criptoanalista tenta decodificar um texto cifrado y_c de um par texto claro/texto cifrado conhecido, (μ, y_c). Conseqüentemente o fator de trabalho é $T_{r1} = (1/2)\text{x}\ N_B$ x complexidade de cifragem.

(b) Recuperando μ de y_c sem as chaves: Esta criptoanalise envolve a solução para k incógnitas de um conjunto de n equações. Sejam $\mu=(\mu_1, \mu_2, \mu_3,...., \mu_k)$ e $y_c = (y_{c1}, y_{c2},...., y_{cn})$ constituintes de um par texto claro/texto cifrado, $E'_{l,w} = (e_1, e_2,..., e_k,...,e_n)$ o vetor de erro permutado e $G'= [g'_{i,j}]$, $i = 1,.....,k$; $j = 1,....., n$ a matriz geradora permutada. Então de $y_c = \mu G' + E'_{l,w}$ o criptoanalista tem um conjunto de n equações lineares. Ele agora escolhe k equações linearmente independentes, e se os e_j dessas equações forem zero, ele estará apto para solucionar corretamente para μ. Notar que sempre é possível encontrar k equações para as quais os e_j são nulos porque $w \leq b \leq ((n\text{-}k)/2) < n\text{-}k$. Para um texto cifrado y_c, ele repete este procedimento até que consiga sucesso em obter um texto claro μ. A máxima probabilidade de sucesso, isto é, de obter um número de k equações linearmente independentes no qual não ocorrem erros, é

$$P_r(sucesso \mid W = w) = \binom{n-w}{k} \Big/ \binom{n}{k}. \tag{5.2}$$

Traduzindo a expressão anterior, tem-se a relação entre número de posições não contaminadas de k em k pela número de combinações totais de k em k.

A probabilidade de ocorrência de um vetor de peso w é

$$P_r(W = w) = \frac{N_e(w)}{Nt_e(w_{min})} = \frac{\sum_{i=w}^{b} n\binom{i-2}{w-2}}{\sum_{w=w_{min}}^{b}\sum_{i=w}^{b} n\binom{i-2}{w-2}} \quad , \tag{5.3}$$

em que $N_e(w)$ é o número de vetores de erro de peso w e $Nt_e(w_{min})$ é o total de vetores de erro para um dado w_{min}. Conseqüentemente a máxima probabilidade de se obter um conjunto de k equações linearmente independentes no qual não ocorrem erros é:

$$P_{r1} = \sum_{w=w_{min}}^{b} \left[\binom{n-w}{k} \Big/ \binom{n}{k} \right] x \frac{N_e(w)}{Nt_e(w_{min})} . \tag{5.4}$$

O número médio de repetições para um bloco de mensagens é P_{r1}^{-1}. Em cada repetição o criptoanalista tem que resolver um sistema de k equações para as k incógnitas e isto requer $O(k^3)$ operações. Conseqüentemente o fator de trabalho deste ataque será $T'_{r1} = k^3$ x P_{r1}^{-1}.

O criptoanalista, em vez de tentar encontrar k equações linearmente independentes e livres de erro, poderia simplesmente fixar k equações e então tentar adivinhar o padrão de erro e resolver o sistema até que obtenha um texto claro significativo (um candidato a padrão de erros pode ser obtido encriptando a mensagem toda nula). A probabilidade de encontrar o correto padrão de erros é $P_{r2} = Nt_e(w_{min})^{-1}$. Isto resulta em um fator de trabalho $T'_{r2} = k^3$ x P_{r2}^{-1}. Conseqüentemente o fator de trabalho geral do ataque, que consiste em recuperar μ de y_c sem as chaves, é $T_{r2} = \min(T'_{r1}, T'_{r2})$.

Uma vez estabelecido o processo de cifragem e decifragem, Campello de Souza [41] sugeriu uma classe de códigos de correção de surto de erros que satisfazem os requisitos acima e que foram introduzidos por Farrell [16]. São códigos matriciais extremamente simples, cujos códigos constituintes são de paridade única C_1 (n_1, k_1, d_1=2) e C_2 (n_2, k_2, d_2 = 2). O código matricial é do tipo C (n,k, d=4), para $n = n_1n_2 = (k_1 + 1)(k_2 + 1)$, $k = k_1k_2$ e tem taxa $R = k_1k_2/(k_1+1)(k_2+1)$, a qual está próxima de um. Para corrigir surtos de erros, é feita a leitura diagonal, o que é equivalente ao embaralhamento das colunas da matriz H. Esta operação deixa inalterada a capacidade de correção de erro aleatório (t =1) mas resulta em uma melhora no valor do comprimento de surto de erros, b, passível de ser corrigido. É mostrado que a

condição $k_2 \geq 2(k_1\text{-}1)$ é necessária e suficiente para obter $b = k_1$. Desta forma, o valor de b pode ser feito alto o bastante enquanto que t permanece inalterado. Também o processo de codificação e decodificação deste código é extremamente simples.

Para esta classe de códigos, o número de códigos corretores de surto de erros que são combinatoriamente equivalentes para um dado código é $N_B = n(k_1!k_2!)$. A complexidade computacional do algoritmo de decodificação é $O(k)$ e portanto, $T_{r1} = ½ \times k_1! \times k_2! \times nk$. Para k_1=8 e k_2=15 então $T_{r1} = 4.6 \times 10^{20}$.

Na Tabela 5.1 alguns valores da taxa R foram gerados para valores ótimos w_{min} e o fator de trabalho T_{r2} resultante, para alguns valores de $k_1 = b$ e k_2, são mostrados.

k_1	k_2	n	k	R	w_{min}	T_{r2}
8	15	144	120	0.83	5	$1.6\text{x}10^{10}$
12	22	299	264	0.88	6	$8.2\text{x}10^{12}$
16	31	544	496	0.91	6	$2.1\text{x}10^{15}$
20	38	819	760	0.93	7	$1.8\text{x}10^{17}$
24	46	1175	1104	0.94	7	$1.3\text{x}10^{19}$
28	54	1595	1512	0.95	7	$7.4\text{x}10^{20}$
32	63	2112	2016	0.96	7	$3.7\text{x}10^{22}$

Tabela 5.1: Alguns exemplos de parâmetros do criptosistema.

A exemplo deste trabalho de Campello de Souza [41], Rocha e Blaum [44] publicaram artigo empregando esta mesma técnica de criptografia usando códigos corretores de erro. Este sistema é baseado no uso de códigos matriciais para a correção de surto de erros em um sistema criptográfico. A codificação e a decodificação são principalmente baseadas em deslocamentos cíclicos e operações lógicas do tipo OU Exclusivo. Ao mesmo tempo, os novos códigos matriciais são otimizados ou de máxima distância (MDS), porque são códigos baseados em aritmética de campo finito, como os códigos Reed Solomon.

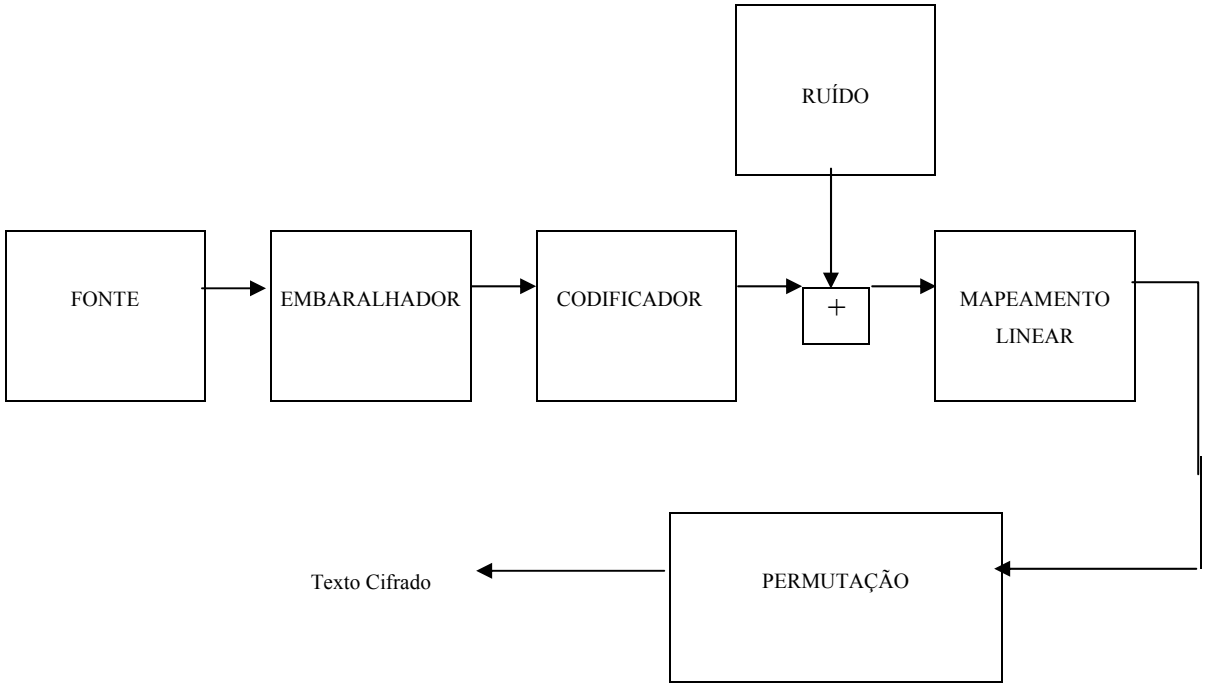


Figura 5.1: O processo de cifragem.

O esquema proposto por Rocha e Blaum [44] funciona da seguinte forma:

Assume-se que se quer codificar k dígitos de informação, denotado por μ, o texto claro. Deve-se multiplicar μ por uma matriz k x k denominada S (embaralhador). O resultado é codificado usando o método descrito por Blaum [45]. O código resultante, em duas dimensões, é uma matriz $(p\text{-}1)$x n, em que $n \leq p$, sendo p um número primo. Escrevendo-o em uma dimensão, ele tem comprimento $(p\text{-}1)n$ (lê-se uma coluna após outra). Seja η este vetor de comprimento $(p\text{-}1)n$. Agora, a este vetor será adicionado um ruído a um dado número de colunas até a capacidade de correção de colunas do código. Seja r_c a versão ruidosa de η. Finalmente, seja P uma permutação sobre as coordenadas $(p\text{-}1)n$, então se aplica P a r_c, completando a cifragem. Este processo está ilustrado na Figura 5.1.

A decifragem é um processo inverso ao descrito na codificação.

Exemplo:

Considerar um código matricial 4 x 5 que pode corrigir qualquer coluna em erro, isto é, as últimas duas colunas são redundantes. Tem-se paridade horizontal e diagonal, ou paridades sobre linhas de inclinação 0 e inclinação 1, como desenhado nas Figuras 5.2 e 5.3.

□	□	□	□	□
∇	∇	∇	∇	∇
♣	♣	♣	♣	♣
♦	♦	♦	♦	♦

Figura 5.2: Linhas com inclinação 0.

□	∇	♣	♦	♥
∇	♣	♦	♥	□
♣	♦	♥	□	∇
♦	♥	□	∇	♣

Figura 5.3: Linhas com inclinação 1.

Por exemplo, seja o vetor μ que se quer codificar

μ = 1 0 1 0 0 1 1 1 0 1 0 0.

Se este vetor for escrito na forma de coluna, obtem-se:

1	0	0		
0	1	1		
1	1	0		
0	1	0		

As últimas duas colunas conterão as informações redundantes. Usando o método descrito em [45] ou [47] tem-se

1	0	0	**0**	**1**
0	1	1	**0**	**0**
1	1	0	**1**	**1**
0	1	0	**1**	**0**

Observa-se que a paridade é para as linhas de inclinação 0 e 1.

Em seguida será introduzido um ruído aleatório a uma das colunas. Supõe-se neste caso, na terceira coluna:

1	0	**1**	0	1
0	1	**0**	0	0
1	1	**0**	1	1
0	1	**1**	1	0

Escrevendo este vetor em apenas uma dimensão tem-se

$r_c = 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0$

Para completar a cifragem, necessita-se aplicar uma permutação a r_c. Por exemplo, considerando a seguinte permutação sobre as 20 coordenadas:

$$P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 7 & 1 & 3 & 12 & 20 & 19 & 13 & 18 & 17 & 14 & 8 & 15 & 9 & 10 & 11 & 16 & 5 & 4 & 6 & 2 \end{pmatrix}$$

O texto cifrado é

$y_c = P.\ r_c = 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0$

Para decifrar a mensagem, tem-se que reverter as operações acima. O primeiro passo é aplicar a permutação inversa P^{-1} ao texto cifrado recebido y_c. Fazendo isto obtem-se

$P^{-1}.\ y_c = r_c = 1\ 0\ 1\ 0\ 0\ \ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0$

Escrevendo em forma de coluna se tem

1	0	1	0	1
0	1	0	0	0
1	1	0	1	1
0	1	1	1	0

Para decodificar, usa-se os métodos descritos em [45]. O primeiro passo é avaliar os valores das síndromes horizontal e diagonal, assumindo uma linha fictícia toda zero na parte inferior do código matricial. A síndrome horizontal é

$$S_h = 1\ 1\ 0\ 1\ 0$$

Enquanto que a síndrome diagonal é

$$S_d = 1\ 0\ 1\ 1\ 0$$

Seja $\rho^q(\boldsymbol{x})$ os q deslocamentos para a direita do vetor $\boldsymbol{x}$. Se somente uma coluna está em erro, supondo a coluna j, $0 \le j \le p$-1, então $\rho^j(S_h) = S_d$ e $\rho^c(S_h) \neq S_d$ para todo $c \neq j$, $0 \le c \le p$-1. Observa-se que neste caso, $\rho^2(S_h) = S_d$, conseqüentemente a coluna 2 está em erro. O algoritmo de Shiloach [46] é um algoritmo com complexidade linear para determinar se um vetor é um deslocamento cíclico de um outro e, em caso afirmativo, a quantidade de deslocamentos necessários.

O erro é dado por S_h. Adicionando este erro à coluna 2 em módulo 2, obtém-se

1	0	0	0	1
0	1	1	0	0
1	1	0	1	1
0	1	0	1	0

Escrevendo estes dígitos horizontalmente, o receptor conclui que o texto claro original transmitido foi

$$\mu = 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0$$

5.3 O Ataque de Al Jabri

Em outro artigo, Rocha [48], discute o ataque de Al Jabri [49] [50] sobre a reconstrução da matriz de permutação P em esquemas de chave secreta. Al Jabri [50] mostrou como reconstruir a permutação empregada em criptografia de chave secreta baseada em códigos matriciais com correção de um único surto de erros, conseqüentemente reduzindo significativamente o fator de trabalho para quebrar o esquema. Al Jabri explorou algumas invariantes no criptosistema proposto por Rocha e Blaum [44] e reduziu a ambigüidade na permutação de um nível de bloco, isto é, de $(n(p\text{-}1))!$ para um nível de coluna, $n!(p\text{-}1)!$, em que n é um inteiro positivo denotando o número de colunas de comprimento $(p\text{-}1)$ em um código matricial retangular binário n x $(p\text{-}1)$, sendo p um número primo. Em um artigo posterior Rocha [48] introduziu um esquema de criptografia de chave secreta, baseado em códigos de correção de surto de erros e em permutações adaptativas, o que é resistente a ataques conhecidos e especialmente talhado para contrapor-se ao ataque de Al Jabri sobre permutações.

O ataque de Al Jabri baseia-se em atacar o criptosistema em um número, M', suficientemente grande de blocos de texto claro todo nulo, gerando uma quantidade M' de texto cifrado. Dentre estes M' textos cifrados seleciona-se uma quantidade igual a M, sendo $M \leq M$', tal que cada um deles tem b dígitos não nulo. Chama-se de b-texto cifrado qualquer texto cifrado com b dígitos não nulos produzido por um texto claro todo nulo. Estes b-texto cifrados contêm informações, em uma forma conveniente, para extrair a permutação que afeta os blocos de b posições consecutivas. Seja i_b, $1 \leq i_b \leq n$, denotando as posições dos dígitos "1" no b-texto cifrado y_{lm}, para $1 \leq lm \leq M$. Estas posições dos dígitos não nulos em cada b-texto cifrado y_{lm} são armazenados como elementos de um conjunto associado S_{lm}. Um grupo de n distintos Q_{ib}, $1 \leq i_b \leq n$, é agora construído dos conjuntos S_{lm} de maneira que cada conjunto Q_{ib} é a união daqueles conjuntos S_{lm} contendo o elemento i_b. Desta forma a cardinalidade de cada Q_{ib} é $2b$-1. Os conjuntos Q_{ib} são ordenados e renomeados como uma lista de n conjuntos P_{ib}, $1 \leq i_b \leq n$, com a propriedade que qualquer dois conjuntos P_{ib} consecutivos tem $2b$-2 elementos idênticos, isto é, diferindo somente em um elemento.

Exemplo:

Seja n =7, a permutação do tipo (6,4,1,3,5,2,7) e seja b=2. Obtém-se como uma seqüência de S_{lm}: $S_1 = \{4,6\}$, $S_2 = \{1,4\}$, $S_3 = \{1,3\}$, $S_4 = \{3,5\}$, $S_5 = \{2,5\}$, $S_6 = \{2,7\}$ e $S_7 = \{6,7\}$.

Dos valores de S_{lm} surgem os seguintes

Q_{ib}: $Q_1 = S_2 \cup S_3 = \{1,3,4\}$, $Q_2 = S_5 \cup S_6 = \{2,5,7\}$, $Q_3 = S_3 \cup S_4 = \{1,3,5\}$, $Q_4 = S_1 \cup S_2 = \{1,4,6\}$, $Q_5 = S_4 \cup S_5 = \{2,3,5\}$, $Q_6 = S_1 \cup S_7 = \{4,6,7\}$ e $Q_7 = S_6 \cup S_7 = \{2,6,7\}$.

A partir deste ponto geram-se os valores de P_{ib} dos conjuntos que difiram em apenas uma posição:

$P_1 = \{1,3,4\}$, $P_2 = \{1,3,5\}$, $P_3 = \{2,3,5\}$, $P_4 = \{2,5,7\}$, $P_5 = \{2,6,7\}$, $P_6 = \{4,6,7\}$ e $P_7 = \{1,4,6\}$.

Logo

$$P_1 \cap P_2 \cap P_3 = \{3\};$$
$$P_2 \cap P_3 \cap P_4 = \{5\};$$
$$P_3 \cap P_4 \cap P_5 = \{2\};$$
$$P_4 \cap P_5 \cap P_6 = \{7\};$$
$$P_5 \cap P_6 \cap P_7 = \{6\};$$
$$P_6 \cap P_7 \cap P_1 = \{4\};$$
$$P_7 \cap P_1 \cap P_2 = \{1\}.$$

Então tem-se a seqüência: {3,5,2,7,6,4,1}que rotacionada para a direita em três posições gera a seqüência de permutação.

Rocha [48] demonstra que também esse ataque é eficiente a criptosistemas baseados em códigos corretores de múltiplos surtos de erros, sem modificar a seqüência de solução que foi estabelecida por Al Jabri [50].

Em sendo desta forma, estes esquemas tornam-se vulneráveis a ataques de "inimigos". Como solução, Rocha [48] sugere o uso de permutação adaptativa para evitar o ataque de Al Jabri, a qual funciona da seguinte forma:

O que se propõe são n distintas permutações P_ψ, $1 \leq \psi \leq n$, com a propriedade que as posições ψ até $\psi + b - 1$ são fixadas por P_ψ, ou seja, não ocorre permutação. Durante o processo de cifragem o surto produzido pelo gerador de erros é usado para selecionar uma das permutações P_ψ's. A P_ψ selecionada é uma que fixa as posições de surtos de erros, isto é, o surto de erros iniciando na posição 1 de uma *n-úpla* se estenderá até à posição b, estas posições de 1 a b da dada *n-úpla* serão fixadas, graças à seleção adequada da permutação P_1. Desta forma o ataque proposto por Al Jabri não ganha informação relevante sobre a permutação empregada, uma vez que o surto não é espalhado pelas permutações. Por outro lado, a capacidade de correção de surto de erros dos códigos é destruída por tais permutações e somente alguém com conhecimento do conjunto das n permutações empregadas é capaz de recuperar eficientemente a mensagem cifrada.

5.4 Sistema de Criptografia Baseado em Códigos Matriciais Usando Entrelaçamento Unidimensional

A exemplo de Campello de Souza [41] e Rocha [44], o esquema proposto nesta dissertação de código matricial com entrelaçamento unidimensional empregando códigos corretores de erros aleatórios ou códigos corretores de surtos de erros, também poderá ser usado em sistemas criptográficos de chave secreta. Neste caso, ao usar o primeiro tipo empregando códigos corretores de erros aleatórios, além da matriz geradora do código, G, e do elemento de permutação P, o fator de entrelaçamento, β, torna-se uma chave secreta. O esquema proposto tem a seguinte esquematização:

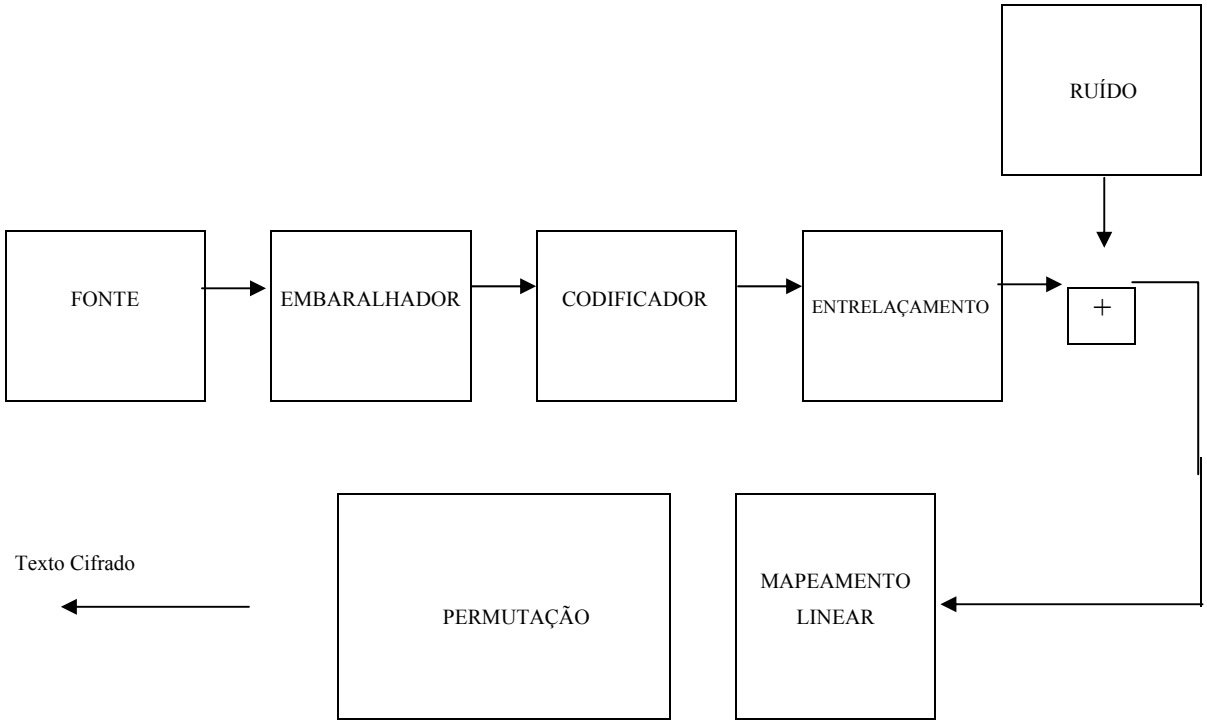


Figura 5.4: Esquema do processo de cifragem de chave secreta proposto.

O esquema pode ser descrito da seguinte forma:

Denota-se por μ o texto claro contendo k dígitos de informação. Deve-se multiplicar μ por uma matriz k x k denominada por S (matriz de embaralhamento), que executa a modificação no texto claro. O resultado é codificado usando o método descrito no capítulo 3. O código resultante, em duas dimensões, constitui uma matriz n_1 x n_2. Sobre esta matriz executa-se o entrelaçamento unidimensional para, em seguida, ser adicionada uma mancha compacta de erros aleatórios (ruído) com peso menor ou igual a n_1. Escrevendo esta matriz com apenas uma dimensão, ele passa a ter comprimento $n_1.n_2$, e é designado por r_c. Finalmente, aplicando uma permutação P sobre as coordenadas de vetor r_c, gera-se o texto cifrado y_c, completando a cifragem. Este processo está ilustrado na Figura 5.4.

A decifragem corresponde ao processo inverso descrito acima. Primeiramente, aplica-se a permutação inversa, P^{-1}, ao texto cifrado y_c, de comprimento $n_1.n_2$, recuperando o código r_c. Em seguida organiza-se r_c em uma matriz de dimensão n_1 x n_2. Aplica-se a transformação linear inversa, T^{-1}, para desentrelaçá-la. Uma vez desentrelaçada haverá apenas um único erro sobre cada linha, que constitui uma palavra-código de um código de correção de erros do tipo

BCH ou Reed-Solomon. Usando os métodos de decodificação destes códigos, corrige-se o erro em cada linha. Os *k* primeiros símbolos desta matriz resultante, lidos coluna por coluna, constituem o bloco de mensagem de comprimento *k*, que fôra embaralhado. Por último, após o perfilamento destes *k* dígitos, aplica-se a matriz de desembaralhamento, S^{-1}, recuperando o texto claro.

Exemplo: Por simplicidade, supõe-se um código Reed-Solomon *4-ário* de comprimento q^m-1 = 3 com capacidade de correção de um dígito aleatório. Para que seja executado o entrelaçamento unidimensional, considera-se o número de linhas da matriz igual a cinco. Desta forma a matriz resultante deverá ter dimensões 5 x 3, uma vez que cada linha desta matriz representa uma palavra-código do código Reed-Solomon *4–ário*.
Seja α a raiz primitiva de $x^2 + x + 1$ e conseqüentemente de ordem três. O campo de Galois *GF(4)* pode ser representado pelas potências de α como sendo:

0 corresponde ao símbolo "0"
1 corresponde ao símbolo "1"
α corresponde ao símbolo "2" e
$\alpha + 1$ corresponde ao símbolo "3".

Portanto este campo possui quatro elementos. Em representação binária isto poderia ser traduzido como:

"0" eqüivale a 00;
"1" eqüivale a 01;
"2" eqüivale a 10;
"3" eqüivale a 11.

Para que este código corrija um erro, o polinômio gerador, *g(x)*, será da forma:

$$g(x) = (x-\alpha)(x-\alpha^2) = x^2 + x + 1.$$

Como tem grau igual a dois, resulta em um código Reed-Solomon (3,1) de comprimento três e de dimensão um. Logo para a seguinte seqüência de texto claro

$m(x) = 0111000101:1010011101:1101010100:0.....$

Separa-se a seqüência em blocos de dois dígitos, como indicado anteriormente, formando por exemplo:

	01	11	00	01	01
GF(4)	1	3	0	1	1

Escrevendo este vetor na forma de coluna, obtém-se:

1		
3		
0		
1		
1		

As últimas duas colunas conterão as informações redundantes, ou seja, os dígitos de paridade.

Cada linha desta matriz representa uma palavra-código do *RS(3,1)*, cuja Tabela de geração é:

m(x)	***m(x).g(x)***	***Código***
0	0	0 0 0
1	$x^2 + x + 1$	1 1 1
2	$2x^2 + 2x + 2$	2 2 2
3	$3x^2 + 3x + 3$	3 3 3

Logo a matriz de dígitos se torna:

1	**1**	**1**
3	**3**	**3**
0	**0**	**0**
1	**1**	**1**
1	**1**	**1**

Nesta matriz aplica-se a transformação linear *T*, que possui o seguinte valor:

$$T = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}.$$

Resultando em

1	3	0
0	1	1
1	1	3
3	0	1
1	1	1

Em seguida acrescenta-se a mancha de erros aleatória de peso máximo igual a cinco, como por exemplo:

1	3	0
0	**2**	1
2	**3**	**0**
3	**1**	1
1	1	1

Resultando no vetor de uma dimensão r_c= 1 0 2 3 1 3 2 3 1 1 0 1 0 1 1.

Para completar a cifragem, necessita-se aplicar uma permutação a r_c. Por exemplo, seja a seguinte permutação sobre as 15 coordenadas:

$$P = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 7 & 1 & 3 & 12 & 15 & 14 & 13 & 5 & 4 & 6 & 8 & 2 & 9 & 10 & 11 \end{pmatrix}$$

O texto cifrado é

$$y_c = P.r_c = 2\ 1\ 2\ 1\ 1\ 1\ 0\ 1\ 3\ 3\ 3\ 0\ 1\ 1\ 0.$$

Para decifrar a mensagem, necessita-se trilhar o caminho inverso. O primeiro passo é aplicar a permutação inversa P^{-1} ao texto cifrado recebido y_c. Fazendo isto obtém-se

$$P^{-1}.\ y_c = r_c = 1\ 0\ 2\ 3\ 1\ 3\ 2\ 3\ 1\ 1\ 0\ 1\ 0\ 1\ 1.$$

Escrevendo em forma de coluna, tem-se

1	3	0
0	**2**	1
2	**3**	**0**
3	**1**	1
1	1	1

Em seguida aplica-se a transformação inversa ao entrelaçamento, T^{-1}, em que

$$T^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}.$$

Obtém-se

1	**3**	1
3	3	**0**
0	**1**	0
1	**2**	1
2	1	1

Neste exemplo descobre-se que apenas uma posição está em erro, visto que os códigos válidos são aqueles mostrados anteriormente. Logo a matriz corrigida passa a ser:

1	1	1
3	3	3
0	0	0
1	1	1
1	1	1

Colocando em uma única dimensão, apenas a primeira coluna, obtém-se o texto claro original:

Texto claro: 1 3 0 1 1 em *GF(4)*.

Na forma binária corresponde a:

Texto claro: 0111000101.

Para tornar este método mais seguro contra ataques conhecidos e que foram descritos anteriormente pode-se empregar também a permutação adaptativa sugerida por Rocha [48].

5.5 Conclusão

É proposta uma ligeira modificação no esquema de criptografia de chave secreta desenvolvido por Campello de Souza [41] e Rocha [44] mediante o emprego de códigos matriciais com entrelaçamento unidimensional, com suas linhas geradas por um código corretor de erros aleatórios ou por um código de correção de surto de erros. Este esquema, ao usar códigos corretores de erros aleatórios, mostra-se interessante uma vez que introduz mais uma componente de chave secreta correspondente ao valor β da matriz T de transformação linear. Para maior segurança deste tipo de chave secreta contra ataques de "inimigos", pode-se empregar o esquema de permutação adaptativa proposto por Rocha [48].

Capítulo 6

Emprego dos Códigos Matriciais com Entrelaçamento Unidimensional em Gravação Magnética

6.1 Introdução

A informação digital sobre a fita magnética é geralmente gravada em alguns canais paralelos à direção de movimento da fita. Uma única cabeça de leitura-escrita é usualmente associada com cada canal, e algumas cabeças lêem ou escrevem simultaneamente em intervalos discretos sobre os canais. A informação pode ser dividida em blocos, mediante intervalos que ocorrem simultaneamente em todos os canais. Estes blocos tomam o aspecto de matrizes retangulares, nas quais os canais são as colunas e os conjuntos de dígitos, que passam simultaneamente sob as cabeças de leitura-escrita, são as linhas.

Sistemas de gravação de fita magnética estão sujeitos a erros causados pelo mecanismo, similar àqueles encontrados em sistemas de discos ópticos e magnéticos, ou seja, defeitos de material e contaminação de superfície. Adicionalmente, a flexibilidade da fita magnética pode gerar o seu próprio conjunto de problemas, como por exemplo a inserção de dígitos devido a estiramento da fita magnética, assim como a falha do circuito de leitura-escrita, associado com algum canal, pode resultar em surtos de erros na coluna correspondente. Desta forma quando são usados códigos corretores de surtos de erros, estes devem ser aplicados pelo menos às colunas.

Em muitos sistemas digitais, como foi mencionado anteriormente, os dados são organizados em trilhas paralelas, que ocupam o comprimento da fita magnética. Cita-se, por exemplo, o sistema de gravação magnética IBM 3420 que usa nove trilhas de dados sobre

uma fita de ½ polegada, como mostra a Figura 6.1. Em função da organização e do sentido em que a fita se move, contaminantes (sujeiras) tendem a corromper uma trilha ou um pequeno número de trilhas adjacentes durante o processo de leitura. Em quase todos os sistemas de controle de erro para gravação magnética as palavras-código são definidas verticalmente (ao longo das trilhas), provendo uma forma de entrelaçamento. A perda de uma ou duas trilhas então requer a correção de somente um ou dois símbolos por palavra-código.

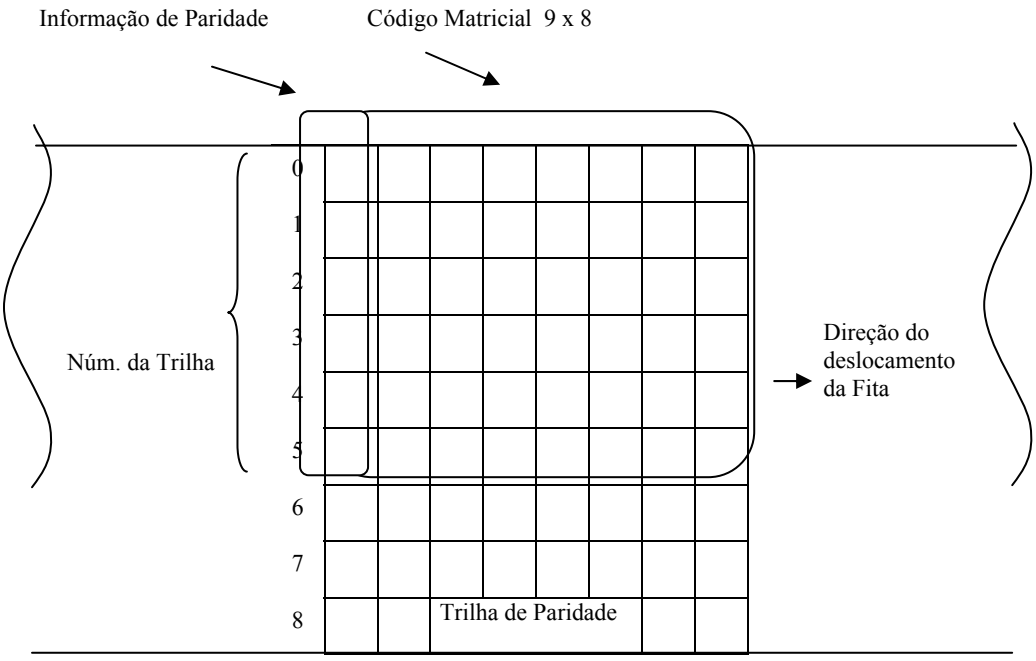


Figura 6.1: Formato dos dados para o sistema de gravação magnética IBM 3420.

Este código pode corrigir erros causados pela perda de uma simples trilha e, caso haja um mecanismo externo de apontamento de erro que declara o apagamento de trilha, ele pode corrigir também erros em duas trilhas. A confiabilidade do sistema permite densidades de gravação de 250 dígitos por milímetro.

Em 1975 a IBM introduziu o 3850 *Mass Storage System* (MSS), um enorme sistema de gravação magnética. Este sistema podia conter até 4720 cartuchos de gravação guardados em compartimentos hexagonais, sendo que cada cartucho tinha 3,5 polegadas de comprimento

e 1,9 polegadas de diâmetro. Além disso, cada cartucho continha 64 pés de fita magnética de 2,7 polegadas de largura. O MSS tinha um mecanismo que selecionava cartuchos para leitura, que os substituíam quando não eram mais necessários. A capacidade total do sistema era de apenas 50,4 MB (*Megabytes*).

O IBM 3850 usava uma cabeça giratória de leitura-escrita. A fita se movia helicoidalmente em relação à cabeça, como mostra a Figura 6.2. Graças às posições relativas da fita e da cabeça de gravação magnética, os dados eram gravados em trilhas inclinadas (*stripe*), como mostra a Figura 6.3. Cada trilha era dividida em 20 segmentos, que por sua vez eram divididos em 15 seções de 16 palavras de oito dígitos cada. As seções também continham um dígito de paridade (paridade ímpar) e alguns dígitos de sincronização.

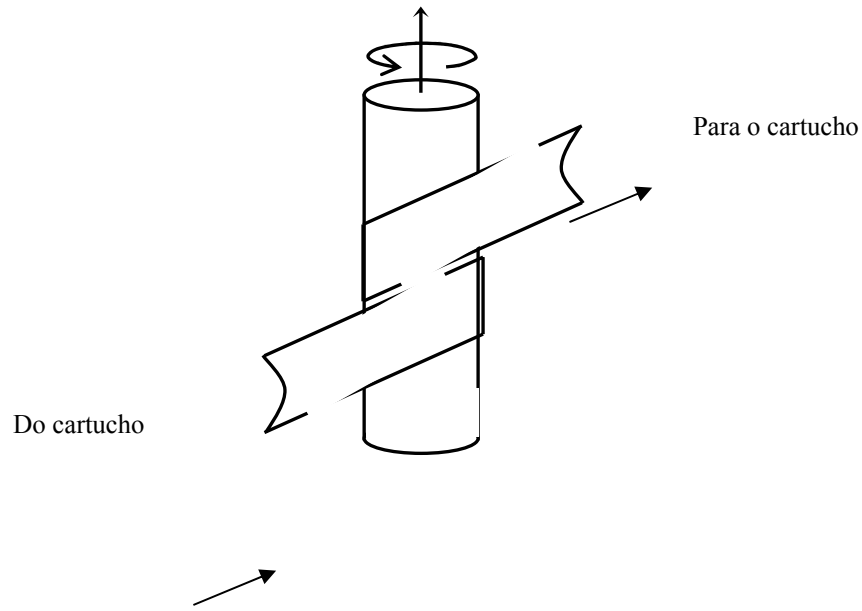


Figura 6.2: Mecanismo de gravação/leitura no IBM 3850 MSS.

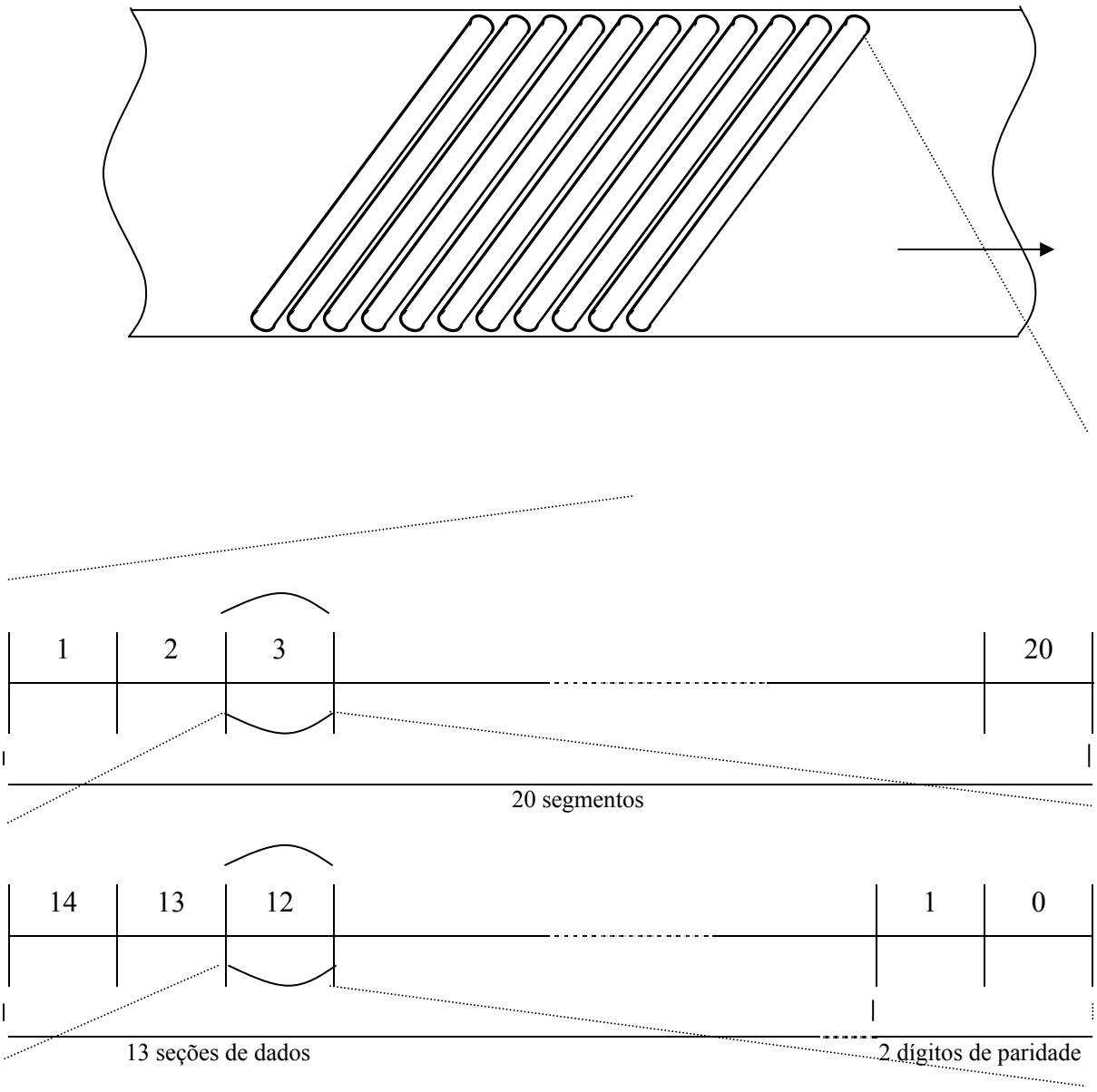


Figura 6.3: Formato dos dados no IBM3850 MSS.

Cada segmento consistia de 16 palavras-código entrelaçadas, geradas por um código *BCH (15,13) 8-ário*. Os 15 símbolos em cada palavra-código eram distribuídos ao longo de 15 seções do segmento. A primeira seção continha os primeiros símbolos para as 16 palavras-código, a segunda seção continha o segundo símbolo, e assim por diante.

O campo $GF(2^8)$ pode ser formado pelo polinômio primitivo $p(x) = x^8 + x^5 + x^3 + x + 1$.

Considerando α uma raiz primitiva de $p(x)$, de ordem 255, sabe-se que todos os elementos não nulos de $GF(2^8)$ podem ser expressos como uma potência de α. Considerando $\varphi = \alpha^{17}$ e sabendo que 17 x 15 =255, então segue que a ordem de φ é 15. Desta forma $GF(2^8)$ contém o subcampo $GF(2^4) = \{0, \varphi, \varphi^2, \varphi^3, ..., \varphi^{15}\}$.

O código IBM3858 MSS era definido por um polinômio gerador $g(x) = (x+1)(x+\varphi)$. Este código tinha distância mínima de três e podia corrigir portanto erros aleatórios simples nos endereços de oito dígitos. Se fosse usado um ponteiro externo indicador de erros, o código podia corrigir apagamentos em duas posições de endereços distintos, simultaneamente.

Uma vez que o código era cíclico, circuitos baseados em registradores de deslocamento podiam ser usados para codificação e decodificação.

Retornando à análise da série de gravadores magnéticos modelo IBM3420, vale ressaltar que o seu esquema de correção de erros foi vislumbrado por Patel e Hong [51] [52]. Este esquema foi um sucesso e permitia gravações magnéticas com densidade de gravação de 6250 dígitos/polegada. Este esquema de correção de erros era capaz, como foi anteriormente mencionado, de corrigir qualquer padrão de erros sobre uma única trilha ou qualquer padrão de erros sobre duas trilhas, desde que fossem identificadas as coordenadas das trilhas, por meio de algum ponteiro externo. Em um outro artigo, McEliece e Blaum [53], apresentam um subcódigo do código de Patel-Hong capaz de corrigir uma trilha e, conjuntamente, recuperar uma trilha apagada ou simplesmente recuperar três trilhas apagadas. É importante explorar este modelo adaptado por McEliece e Blaum para melhor compreender o esquema original de Patel e Hong.

Como foi mencionado, o sistema IBM3420 escreve caracteres em paralelo, ao longo de nove trilhas em uma fita de meia polegada, como mostrado na Figura 6.1. Cada caractere consistindo de oito dígitos de informação e de um dígito de paridade geral.

As linhas e colunas desta matriz são consideradas elementos de um campo de Galois de ordem 2^8, $GF(2^8)$. A exemplo do código desenvolvido por Patel-Hong, o polinômio irredutível usado, definido em $GF(2^8)$, era $g(x) = 1 + x^3 + x^4 + x^5 + x^8$. Denota-se os primeiros

oito dígitos de cada coluna por B_i, $0 \leq i \leq 7$, e cada linha por Z_j, $0 \leq j \leq 8$. A linha Z_8 corresponde àquela que contém os dígitos de paridade. No código proposto por McEliece e Blaum, B_0, B_1 e Z_8 passaram a ser consideradas trilhas redundantes com dígitos de paridade; consequentemente, a taxa deste código proposto é 2/3. As equações do código são definidas como:

$$\sum_{j=0}^{8} Z_j = 0, \tag{6.1}$$

$$\sum_{i=0}^{7} x^i B_i = 0, \tag{6.2}$$

$$\sum_{i=0}^{7} x^{2i} B_i, \tag{6.3}$$

em que

$$Z_j = \sum_{k=0}^{7} b_{jk} x^k \in GF(2^8), \tag{6.4}$$

$$B_i = \sum_{k=0}^{7} b_{ik} x^k \in GF(2^8). \tag{6.5}$$

As operações em (6.1), (6.2) e (6.3) são tomadas módulo *g(x),* ou seja, elas são operações em $GF(2^8)$. As equações (6.1) e (6.2) definem o código Patel-Hong, logo este código proposto por McEliece e Blaum é na realidade um subcódigo do original.

No processo de codificação, Z_8 é obtido usando o procedimento descrito em (6.1). B_2, B_3, B_4, B_5, B_6 e B_7 são dados, uma vez que eles contem os símbolos de informação. De (6.2) e (6.3)

$$B_0 + xB_1 = \sum_{i=2}^{7} x^i B_i \,, \tag{6.6}$$

$$B_0 + x^2 B_1 = \sum_{i=2}^{7} x^{2i} B_i \,. \tag{6.7}$$

Solucionando (6.7) obtém-se

$$B_0 = \sum_{i=2}^{7} x^{i+1}(x^{i-2} + ... + 1) B_i \,, \tag{6.8}$$

$$B_1 = \sum_{i=2}^{7} x^{i-1}(x^{i-1} + ... + 1) B_i \,. \tag{6.9}$$

Circuitos para executar as operações (6.8) e (6.9) são perfeitamente possíveis de implementação.

Já no processo de decodificação, considera-se que as linhas $\hat{Z}_0$, $\hat{Z}_1$,....., $\hat{Z}_8$ são recebidas/lidas (colunas $\hat{B}_0$, $\hat{B}_1$,....., $\hat{B}_8$, respectivamente). O primeiro passo do decodificador é calcular as três síndromes

$$S_0 = \sum_{j=0}^{8} \hat{Z}_j \quad , \tag{6.10}$$

$$S_1 = \sum_{i=0}^{7} x^i \hat{B}_i \quad , \tag{6.11}$$

$$S_2 = \sum_{i=0}^{7} x^{2i} \hat{B}_i \quad . \tag{6.12}$$

Se não ocorrem erros, estas síndromes valem zero. Existe outra forma de calcular estas síndromes usando os valores de $\hat{Z}_0$, $\hat{Z}_1$,..., $\hat{Z}_8$, que seria por meio do Lemma descrito a seguir.

Lemma 6.1:

$$S_1 = \sum_{i=0}^{7} x^i \hat{Z}_i \quad , \tag{6.13}$$

$$S_2 = \sum_{i=0}^{7} x^i \hat{Z}_i^2 \quad . \tag{6.14}$$

No seu trabalho, McEliece e Blaum [53] demonstram como é possível realizar a correção de erros e ao final propõem um modelo geral.

6.2 Sistema de Correção de Erros para Gravação Magnética Baseado em Códigos Matriciais Usando Entrelaçamento Unidimensional

No sistema de gravação proposto nesta dissertação além de se poder executar o entrelaçamento previsto no trabalho de Patel-Hong [51] [52], o qual permite a correção de erros e apagamentos de trilhas, existe a possibilidade de correção de manchas de erros sobre mais de uma trilha.

A fim de ilustrar como este esquema poderia ser implementado, é mostrado um simples exemplo que passa a ser descrito a seguir.

Seja um sistema composto de sete cabeças de gravação magnética, tendo cada uma a função de gravar uma trilha paralela ao movimento da fita, como mostra a Figura 6.4.

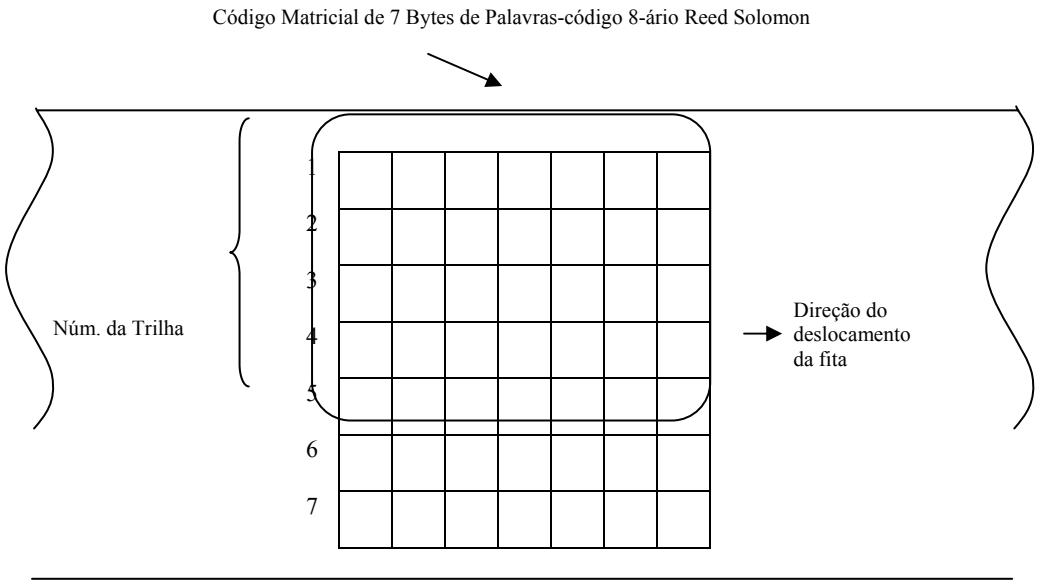


Figura 6.4: Ilustração do processo de gravação magnética proposto.

Neste sistema cada coluna representa uma palavra-código formada por sete dígitos, sendo cada dígito gravado ou lido por uma correspondente cabeça de gravação.

A palavra-código gravada em cada coluna é gerada por um código Reed-Solomon *8-ário* para a correção de um único erro aleatório. Portanto este código é gerado em *GF(8)* cujo polinômio primitivo é $x^3 + x + 1$ que possui ordem sete. Sendo α a raiz primitiva de $x^3 + x + 1$, os elementos de *GF(8)* podem ser obtidos como mostrado a seguir:

Potência de α	**Símbolo em *GF(8)***	**Representação binária**
α	α	(0,1,0)
α^2	α^2	(1,0,0)
α^3	$\alpha + 1$	(0,1,1)
α^4	$\alpha^2 + \alpha$	(1,1,0)
α^5	$\alpha^2 + \alpha + 1$	(1,1,1)
α^6	$\alpha^2 + 1$	(1,0,1)
α^7	1	(0,0,1)
*	0	(0,0,0)

Como nesta proposição se almeja um código que seja capaz de corrigir um único erro, o seu polinômio gerador deve ter como raiz $2t = 2$ potências consecutivas de α. Com base nisso, um polinômio gerador de senso estrito é construído da seguinte forma:

$$g(x) = (x-\alpha)(x-\alpha^2) = x^2 + \alpha^4 x + \alpha^3.$$

Uma vez que o polinômio gerador tem grau dois, o código Reed-Solomon (7,5) que ele define tem dimensão cinco sobre *GF(8)* e consequentemente $8^5 = 32768$ palavras-código. A matriz de verificação de paridade é mostrada a seguir.

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha & \alpha^3 & \alpha^5 \end{bmatrix}.$$

Como se pôde observar, cada três dígitos binários é equivalente a um símbolo em *GF(8)*. Desta forma o código Reed-Solomon (7,5) têm dimensão 15 e comprimento total 21 em *GF(2)*. Portanto, para a geração da palavra-código do código *RS* (7,5) são separados 15 dígitos binários da seqüência de entrada de dados a serem gravados, que correspondem a cinco símbolos em *GF(8),* para, em seguida, ser implementada a operação $c(x) = m(x).g(x)$, em que $c(x)$ é a palavra-código, $m(x)$ são os dados a serem codificados e $g(x)$ é o polinômio gerador.

Neste esquema, para a geração do código matricial sobre a fita magnética, serão utilizadas sete palavras-código de *RS* (7,5), o que implica em um código matricial com dimensão 7 x 7, ou seja, as sete colunas equivalendo às sete palavras-código e as sete linhas aos sete símbolos de cada palavra-código. Como cada símbolo em *GF(8)* corresponde a três dígitos binários, este código matricial gravado sobre a fita magnética terá fisicamente a dimensão 7x21. Após a gravação desta estrutura matricial, será gerada uma área de não-gravação (banda de segurança) para que, em seguida, possa ser gravada a próxima seqüência de dados na forma de código matricial. Este processo será repetido até que todos os dados estejam gravados na fita.

Para exemplificar, considere-se a seguinte seqüência de entrada de dados a serem gravados na fita magnética:

Seqüência de entrada: 000111000111001

Esta sequência corresponde a $0\ \alpha^5\ 0\ \alpha^5\ 1$ em *GF(8),* ou seja, o dado a ser gravado é escrito na forma polinomial como

$$m(x) = \alpha^5 x^3 + \alpha^5 x + 1.$$

Desta forma será gerada a palavra código

$$c(x) = m(x).g(x) = (\alpha^5 x^3 + \alpha^5 x + 1).(x^2 + \alpha^4 x + \alpha^3) = \alpha^5 x^5 + \alpha^2 x^4 + \alpha^6 x^3 + \alpha^6 x^2 + \alpha^2 x + \alpha^3.$$

Portanto a palavra-código gerada seria formada pela seqüência $(0, \alpha^5, \alpha^2, \alpha^6, \alpha^6, \alpha^2, \alpha^3)$.

Este processo se repete por mais seis seqüências de 15 dígitos binários cada, gerando ao final sete palavras-código, sendo cada uma constituída por sete símbolos em *GF(8).*

Estas sete palavras-código irão ocupar as linhas de um código matricial de dimensão 7x7, e, a partir deste ponto, executa-se o entrelaçamento unidimensional proposto neste trabalho. Isto pode ser implementado usando dispositivos de memória, como mostrado a seguir.

a_1	a_2	a_3	a_4	a_5	a_6	a_7
b_1	b_2	b_3	b_4	b_5	b_6	b_7
c_1	c_2	c_3	c_4	c_5	c_6	c_7
d_1	d_2	d_3	d_4	d_5	d_6	d_7
e_1	e_2	e_3	e_4	e_5	e_6	e_7
f_1	f_2	f_3	f_4	f_5	f_6	f_7
g_1	g_2	g_3	g_4	g_5	g_6	g_7

Figura 6.5: Código Matricial 7x7 gerado pelas sete palavras-código do código *RS*(7,5).

No código matricial ilustrado na Figura 6.5, as linhas *a*, *b*, *c*, *d*, *e*, *f* e *g* representam as sete palavras-código *RS*(7,5).

A matriz de transformação linear, *T*, para executar o entrelaçamento unidimensional pode ser a seguinte.

$$T = \begin{bmatrix} 1 & 3 \\ 0 & 4 \end{bmatrix}.$$

Aplicando esta transformação ao código matricial, resulta em

a_1	b_2	c_3	d_4	e_5	f_6	g_7
f_1	g_2	a_3	b_4	c_5	d_6	e_7
d_1	e_2	f_3	g_4	a_5	b_6	c_7
b_1	c_2	d_3	e_4	f_5	g_6	a_7
g_1	a_2	b_3	c_4	d_5	e_6	f_7
e_1	f_2	g_3	a_4	b_5	c_6	d_7
c_1	d_2	e_3	f_4	g_5	a_6	b_7

Figura 6.6: Código Matricial 7x7 entrelaçado.

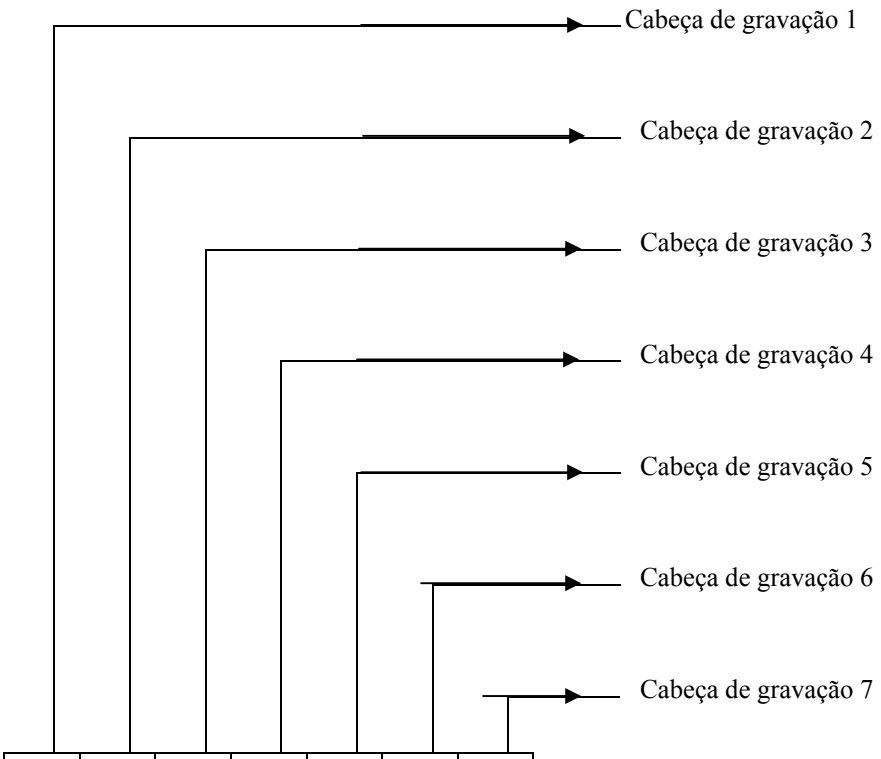
Cada coluna representa os dados que serão gravados por cada cabeça de gravação magnética. Portanto observa-se que há a necessidade de se aguardar pela última palavra-código (correspondente à palavra-código *g*) antes de iniciar o processo de gravação. Isto gera um retardo que é comum quando se opera com entrelaçamento, como é detalhado no capítulo 2. Para reduzir este retardo pode-se alocar os símbolos que formam as palavras-código diretamente na posição que estes ocupariam após o processo de entrelaçamento, ou seja, a seqüência de dados é recebida, em seguida, é gerada a palavra-código e por último aloca-se os dígitos desta palavra-código nas suas posições reservadas de memória para formar o código matricial entrelaçado.

Como cada símbolo é gerado a partir de três dígitos binários, a alocação física destes dados na memória apresenta o formato mostrado na Figura 6.7.

$a_{1(1)}$	$b_{2(1)}$	$c_{3(1)}$	$d_{4(1)}$	$e_{5(1)}$	$f_{6(1)}$	$g_{7(1)}$
$a_{1(2)}$	$b_{2(2)}$	$c_{3(2)}$	$d_{4(2)}$	$e_{5(2)}$	$f_{6(2)}$	$g_{7(2)}$
$a_{1(3)}$	$b_{2(3)}$	$c_{3(3)}$	$d_{4(3)}$	$e_{5(3)}$	$f_{6(3)}$	$g_{7(3)}$
$f_{1(1)}$	$g_{2(1)}$	$a_{3(1)}$	$b_{4(1)}$	$c_{5(1)}$	$d_{6(1)}$	$e_{7(1)}$
$f_{1(2)}$	$g_{2(2)}$	$a_{3(2)}$	$b_{4(2)}$	$c_{5(2)}$	$d_{6(2)}$	$e_{7(2)}$
$f_{1(3)}$	$g_{2(3)}$	$a_{3(3)}$	$b_{4(3)}$	$c_{5(3)}$	$d_{6(3)}$	$e_{7(3)}$
$d_{1(1)}$	$e_{2(1)}$	$f_{3(1)}$	$g_{4(1)}$	$a_{5(1)}$	$b_{6(1)}$	$c_{7(1)}$
$d_{1(2)}$	$e_{2(2)}$	$f_{3(2)}$	$g_{4(2)}$	$a_{5(2)}$	$b_{6(2)}$	$c_{7(2)}$
$d_{1(3)}$	$e_{2(3)}$	$f_{3(3)}$	$g_{4(3)}$	$a_{5(3)}$	$b_{6(3)}$	$c_{7(3)}$
$b_{1(1)}$	$c_{2(1)}$	$d_{3(1)}$	$e_{4(1)}$	$f_{5(1)}$	$g_{6(1)}$	$a_{7(1)}$
$b_{1(2)}$	$c_{2(2)}$	$d_{3(2)}$	$e_{4(2)}$	$f_{5(2)}$	$g_{6(2)}$	$a_{7(2)}$
$b_{1(3)}$	$c_{2(3)}$	$d_{3(3)}$	$e_{4(3)}$	$f_{5(3)}$	$g_{6(3)}$	$a_{7(3)}$
$g_{1(1)}$	$a_{2(1)}$	$b_{3(1)}$	$c_{4(1)}$	$d_{5(1)}$	$e_{6(1)}$	$f_{7(1)}$
$g_{1(2)}$	$a_{2(2)}$	$b_{3(2)}$	$c_{4(2)}$	$d_{5(2)}$	$e_{6(2)}$	$f_{7(2)}$
$g_{1(3)}$	$a_{2(3)}$	$b_{3(3)}$	$c_{4(3)}$	$d_{5(3)}$	$e_{6(3)}$	$f_{7(3)}$
$e_{1(1)}$	$f_{2(1)}$	$g_{3(1)}$	$a_{4(1)}$	$b_{5(1)}$	$c_{6(1)}$	$d_{7(1)}$
$e_{1(2)}$	$f_{2(2)}$	$g_{3(2)}$	$a_{4(2)}$	$b_{5(2)}$	$c_{6(2)}$	$d_{7(2)}$
$e_{1(3)}$	$f_{2(3)}$	$g_{3(3)}$	$a_{4(3)}$	$b_{5(3)}$	$c_{6(3)}$	$d_{7(3)}$
$c_{1(1)}$	$d_{2(1)}$	$e_{3(1)}$	$f_{4(1)}$	$g_{5(1)}$	$a_{6(1)}$	$b_{7(1)}$
$c_{1(2)}$	$d_{2(2)}$	$e_{3(2)}$	$f_{4(2)}$	$g_{5(2)}$	$a_{6(2)}$	$b_{7(2)}$
$c_{1(3)}$	$d_{2(3)}$	$e_{3(3)}$	$f_{4(3)}$	$g_{5(3)}$	$a_{6(3)}$	$b_{7(3)}$

Figura 6.7 : Distribuição dos dados do código matricial no dispositivo de memória.

Sendo, por exemplo, $a_{1(1)}$,$a_{1(2)}$ e $a_{1(3)}$ os três dígitos binários que formam o símbolo a_1 em *GF(8)* da palavra-código *a*. Portanto a memória física deve ter 7 x 21 = 147 endereços ou posições de memória, para que possa comportar todos os dígitos binários do código matricial entrelaçado. Os dados serão gravados na fita magnética (considerando o seu sentido de deslocamento) da forma indicada na Figura 6.8.



$a_{1(1)}$	$b_{2(1)}$	$c_{3(1)}$	$d_{4(1)}$	$e_{5(1)}$	$f_{6(1)}$	$g_{7(1)}$
$a_{1(2)}$	$b_{2(2)}$	$c_{3(2)}$	$d_{4(2)}$	$e_{5(2)}$	$f_{6(2)}$	$g_{7(2)}$
$a_{1(3)}$	$b_{2(3)}$	$c_{3(3)}$	$d_{4(3)}$	$e_{5(3)}$	$f_{6(3)}$	$g_{7(3)}$
$f_{1(1)}$	$g_{2(1)}$	$a_{3(1)}$	$b_{4(1)}$	$c_{5(1)}$	$d_{6(1)}$	$e_{7(1)}$
$f_{1(2)}$	$g_{2(2)}$	$a_{3(2)}$	$b_{4(2)}$	$c_{5(2)}$	$d_{6(2)}$	$e_{7(2)}$
$f_{1(3)}$	$g_{2(3)}$	$a_{3(3)}$	$b_{4(3)}$	$c_{5(3)}$	$d_{6(3)}$	$e_{7(3)}$
$d_{1(1)}$	$e_{2(1)}$	$f_{3(1)}$	$g_{4(1)}$	$a_{5(1)}$	$b_{6(1)}$	$c_{7(1)}$
$d_{1(2)}$	$e_{2(2)}$	$f_{3(2)}$	$g_{4(2)}$	$a_{5(2)}$	$b_{6(2)}$	$c_{7(2)}$
$d_{1(3)}$	$e_{2(3)}$	$f_{3(3)}$	$g_{4(3)}$	$a_{5(3)}$	$b_{6(3)}$	$c_{7(3)}$
$b_{1(1)}$	$c_{2(1)}$	$d_{3(1)}$	$e_{4(1)}$	$f_{5(1)}$	$g_{6(1)}$	$a_{7(1)}$
$b_{1(2)}$	$c_{2(2)}$	$d_{3(2)}$	$e_{4(2)}$	$f_{5(2)}$	$g_{6(2)}$	$a_{7(2)}$
$b_{1(3)}$	$c_{2(3)}$	$d_{3(3)}$	$e_{4(3)}$	$f_{5(3)}$	$g_{6(3)}$	$a_{7(3)}$
$g_{1(1)}$	$a_{2(1)}$	$b_{3(1)}$	$c_{4(1)}$	$d_{5(1)}$	$e_{6(1)}$	$f_{7(1)}$
$g_{1(2)}$	$a_{2(2)}$	$b_{3(2)}$	$c_{4(2)}$	$d_{5(2)}$	$e_{6(2)}$	$f_{7(2)}$
$g_{1(3)}$	$a_{2(3)}$	$b_{3(3)}$	$c_{4(3)}$	$d_{5(3)}$	$e_{6(3)}$	$f_{7(3)}$
$e_{1(1)}$	$f_{2(1)}$	$g_{3(1)}$	$a_{4(1)}$	$b_{5(1)}$	$c_{6(1)}$	$d_{7(1)}$
$e_{1(2)}$	$f_{2(2)}$	$g_{3(2)}$	$a_{4(2)}$	$b_{5(2)}$	$c_{6(2)}$	$d_{7(2)}$
$e_{1(3)}$	$f_{2(3)}$	$g_{3(3)}$	$a_{4(3)}$	$b_{5(3)}$	$c_{6(3)}$	$d_{7(3)}$
$c_{1(1)}$	$d_{2(1)}$	$e_{3(1)}$	$f_{4(1)}$	$g_{5(1)}$	$a_{6(1)}$	$b_{7(1)}$
$c_{1(2)}$	$d_{2(2)}$	$e_{3(2)}$	$f_{4(2)}$	$g_{5(2)}$	$a_{6(2)}$	$b_{7(2)}$
$c_{1(3)}$	$d_{2(3)}$	$e_{3(3)}$	$f_{4(3)}$	$g_{5(3)}$	$a_{6(3)}$	$b_{7(3)}$

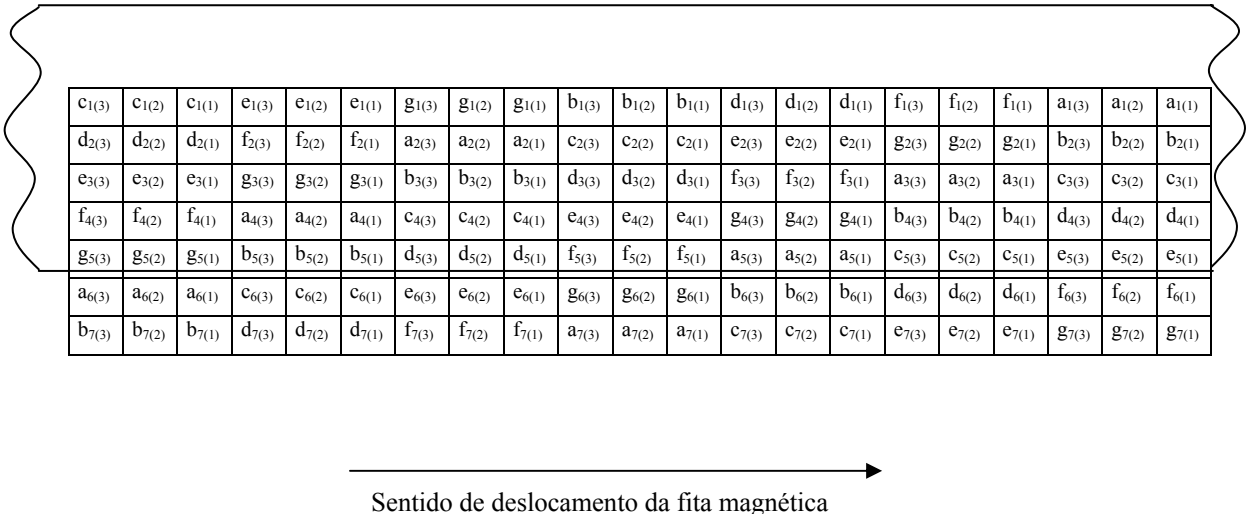


Figura 6.8: Formato de gravação do código matricial sobre a fita magnética.

Como existe um retardo inerente a este processo, como mencionado anteriormente, em função de se ter que aguardar pelo preenchimento de todas as posições da memória física antes de iniciar a transferência destes dados para as cabeças de gravação, formar-se-á um espaço vazio sem gravação (*gap*), separando na fita duas palavras-código sucessivas do código matricial gerado. Para minimizar este *gap,* tornando o aproveitamento da fita magnética mais eficiente pode-se distribuir estas palavras-código em mais do que um dispositivo de memória, ou seja, trabalha-se com dois ou mais grupos de memória e a descarga para a fita magnética se dá de forma sequencial e contínua. A Figura 6.9 a seguir ilustra este procedimento.

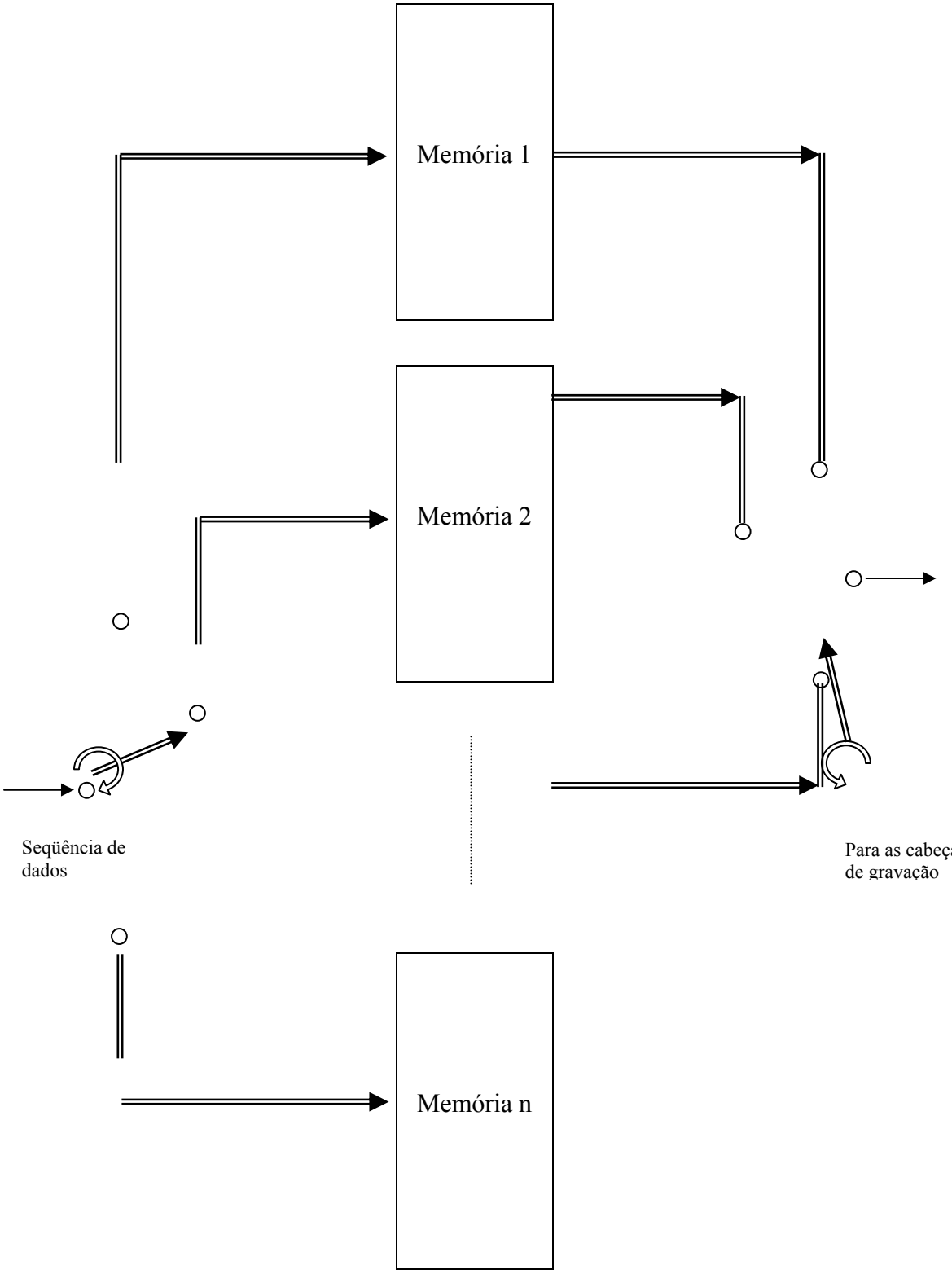


Figura 6.9: Processo de gravação dos dados de forma contínua devido ao uso de um banco de memórias.

Neste caso, como o processo de gravação é contínuo, não haverá separação física entre as estruturas matriciais sucessivas. Portanto, no processo de leitura pode haver perda de dados por falha de sincronização. Para evitar este efeito, adiciona-se ao esquema mostrado na Figura 6.9 um gerador de sincronização que deve sempre introduzir um dado (*safe guard*) ao final da gravação de uma palavra válida do código matricial proposto. Esta separação pode ser constituída apenas pela não gravação de dados na fita. Neste caso, o dígito binário "1" equivaleria a um sentido de polarização dos domínios magnéticos da fita, enquanto o nível "0" corresponderia ao outro sentido de polarização, ou seja, a ausência de polarização dos domínios magnéticos da fita representaria o *gap* de sincronização ou simplesmente ausência de informação. Este tipo de codificação binária é conhecida como bipolar.

O processo de leitura dos dados gravados na fita magnética ocorre de forma inversa ao processo de gravação descrito anteriormente. Deve-se ressaltar que nesta etapa os dados lidos irão formar o código matricial entrelaçado, sobre o qual deve ser aplicada a transformação inversa de entrelaçamento para, em seguida, ser executada a correção de erros e/ou apagamentos das palavras-código que compõem as linhas da estrutura matricial. Ao final, os dados voltarão ao seu estado original, anterior ao processo de gravação.

O fato de ter escolhido o código Reed-Solomon neste exemplo, possibilita que no código matricial 7 x 21 resultante existam 105 dígitos binários de informação e apenas 42 dígitos binários de paridade. Este código corresponde a um código *BCH* (21,15). No entanto no primeiro caso pode-se corrigir uma mancha de erros de peso 7 x 3 = 21 dígitos binários ou sete símbolos em *GF(8),* enquanto que se fosse usado o código *BCH* correspondente, só poderia ser corrigida uma mancha de peso igual a sete dígitos binários.

Os principais formatos de manchas de erros que poderão ser corrigidas neste exemplo estão ilustrados na Figura 6.10. Nas estruturas matriciais representadas, as colunas representam as cabeças de gravação magnética.

a_1	b_2	c_3	d_4	e_5	f_6	g_7
f_1	g_2	a_3	b_4	c_5	d_6	e_7
d_1	e_2	f_3	g_4	a_5	b_6	c_7
b_1	c_2	d_3	e_4	f_5	g_6	a_7
g_1	a_2	b_3	c_4	d_5	e_6	f_7
e_1	f_2	g_3	a_4	b_5	c_6	d_7
c_1	d_2	e_3	f_4	g_5	a_6	b_7

(a)

a_1	b_2	c_3	d_4	e_5	f_6	g_7
f_1	g_2	a_3	b_4	c_5	d_6	e_7
d_1	e_2	f_3	g_4	a_5	b_6	c_7
b_1	c_2	d_3	e_4	f_5	g_6	a_7
g_1	a_2	b_3	c_4	d_5	e_6	f_7
e_1	f_2	g_3	a_4	b_5	c_6	d_7
c_1	d_2	e_3	f_4	g_5	a_6	b_7

(b)

a_1	b_2	c_3	d_4	e_5	f_6	g_7
f_1	g_2	a_3	b_4	c_5	d_6	e_7
d_1	e_2	f_3	g_4	a_5	b_6	c_7
b_1	c_2	d_3	e_4	f_5	g_6	a_7
g_1	a_2	b_3	c_4	d_5	e_6	f_7
e_1	f_2	g_3	a_4	b_5	c_6	d_7
c_1	d_2	e_3	f_4	g_5	a_6	b_7

(c)

a_1	b_2	c_3	d_4	e_5	f_6	g_7
f_1	g_2	a_3	b_4	c_5	d_6	e_7
d_1	e_2	f_3	g_4	a_5	b_6	c_7
b_1	c_2	d_3	e_4	f_5	g_6	a_7
g_1	a_2	b_3	c_4	d_5	e_6	f_7
e_1	f_2	g_3	a_4	b_5	c_6	d_7
c_1	d_2	e_3	f_4	g_5	a_6	b_7

(d)

Figura 6.10: (a) Surto de erros em uma cabeça magnética; (b) Surto de erros bidimensional (Mancha de erros) gerado por duas cabeças de gravação; (c) Mancha de erros gerada em cinco cabeças de gravação; (d) Mancha de erros gerada em três cabeças de gravação.

Para o exemplo detalhado anteriormente, as linhas do código matricial em vez de serem codificadas por um código corretor de erros aleatórios, podem ser codificadas por um código corretor de surtos de erros, conforme é proposto no capítulo 4. As vantagens desta modificação são descritas no referido capítulo. Com esta mudança, o esquema de gravação/reprodução de dados proposto passa por pequenas adaptações. O detalhamento deste esquema é visto a seguir.

Conforme estudado no capítulo 4, o código matricial a ser gravado deverá ter dimensão 7 x 7, uma vez que serão utilizadas sete cabeças de gravação, sendo cada uma responsável pela gravação de uma linha do código matricial.

Suponha que o código corretor de surtos de erros que codifica as linhas do código matricial seja capaz de corrigir um único erro. Desta maneira este código de linha será do tipo (7,5), ou seja, comprimento sete (corresponde ao número de colunas do código matricial) e dimensão cinco. Sendo assim o código matricial terá duas colunas redundantes.

Para se formar este código matricial, separam-se 7 x 5 = 35 dígitos binários na seqüencia de dados de entrada para a gravação e se distribuem estes dígitos pelas linhas do código matricial a ser gerado, como está ilustrado na Figura 6.11.

Seqüência de dados: $a_1a_2a_3a_4a_5b_1b_2b_3b_4b_5c_1c_2c_3c_4c_5d_1d_2d_3d_4d_5e_1e_2e_3e_4e_5f_1f_2f_3f_4f_5g_1g_2g_3g_4g_5$.....

a_1	a_2	a_3	a_4	a_5		
b_1	b_2	b_3	b_4	b_5		
c_1	c_2	c_3	c_4	c_5		
d_1	d_2	d_3	d_4	d_5		
e_1	e_2	e_3	e_4	e_5		
f_1	f_2	f_3	f_4	f_5		
g_1	g_2	g_3	g_4	g_5		

Figura 6.11: Arranjo dos dados na estrutura matricial.

Em seguida codifica-se as linhas, usando um código corretor de surtos [55], gerando a matriz mostrada na Figura 6.12.

a_1	a_2	a_3	a_4	a_5	$\mathbf{a_6}$	$\mathbf{a_7}$
b_1	b_2	b_3	b_4	b_5	$\mathbf{b_6}$	$\mathbf{b_7}$
c_1	c_2	c_3	c_4	c_5	$\mathbf{c_6}$	$\mathbf{c_7}$
d_1	d_2	d_3	d_4	d_5	$\mathbf{d_6}$	$\mathbf{d_7}$
e_1	e_2	e_3	e_4	e_5	$\mathbf{e_6}$	$\mathbf{e_7}$
f_1	f_2	f_3	f_4	f_5	$\mathbf{f_6}$	$\mathbf{f_7}$
g_1	g_2	g_3	g_4	g_5	$\mathbf{g_6}$	$\mathbf{g_7}$

Figura 6.12: Geração do código matricial 7x7. As duas últimas colunas representam os dígitos de paridade.

No código matricial ilustrado na Figura 6.12, as linhas *a*, *b*, *c*, *d*, *e*, *f* e *g* representam as sete palavras-código do código corretor de surto de erros (7,5).

Aplica-se ao código matricial, para executar o entrelaçamento unidimensional, a seguinte matriz de transformação linear, *T*:

$$T = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}.$$

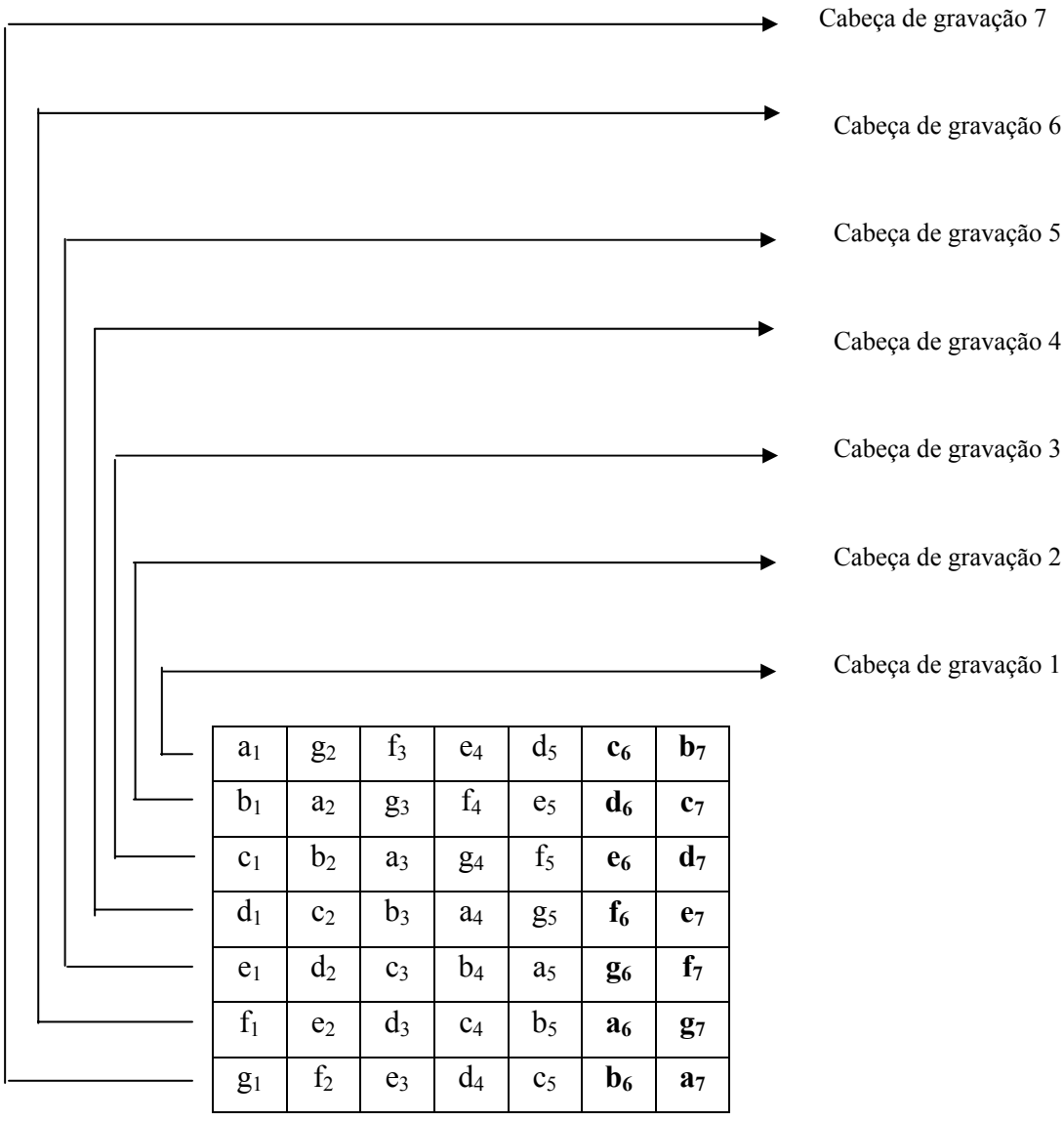
Resultando no código matricial entrelaçado mostrado na Figura 6.13.

a_1	g_2	f_3	e_4	d_5	$\mathbf{c_6}$	$\mathbf{b_7}$
b_1	a_2	g_3	f_4	e_5	$\mathbf{d_6}$	$\mathbf{c_7}$
c_1	b_2	a_3	g_4	f_5	$\mathbf{e_6}$	$\mathbf{d_7}$
d_1	c_2	b_3	a_4	g_5	$\mathbf{f_6}$	$\mathbf{e_7}$
e_1	d_2	c_3	b_4	a_5	$\mathbf{g_6}$	$\mathbf{f_7}$
f_1	e_2	d_3	c_4	b_5	$\mathbf{a_6}$	$\mathbf{g_7}$
g_1	f_2	e_3	d_4	c_5	$\mathbf{b_6}$	$\mathbf{a_7}$

Figura 6.13: Código Matricial 7x7 entrelaçado.

Cada linha do código matricial entrelaçado representa os dados que serão gravados por cada cabeça de gravação magnética.

Os dados serão gravados na fita magnética (considerando o seu sentido de deslocamento) da forma indicada na Figura 6.14.



a_1	g_2	f_3	e_4	d_5	$\mathbf{c_6}$	$\mathbf{b_7}$
b_1	a_2	g_3	f_4	e_5	$\mathbf{d_6}$	$\mathbf{c_7}$
c_1	b_2	a_3	g_4	f_5	$\mathbf{e_6}$	$\mathbf{d_7}$
d_1	c_2	b_3	a_4	g_5	$\mathbf{f_6}$	$\mathbf{e_7}$
e_1	d_2	c_3	b_4	a_5	$\mathbf{g_6}$	$\mathbf{f_7}$
f_1	e_2	d_3	c_4	b_5	$\mathbf{a_6}$	$\mathbf{g_7}$
g_1	f_2	e_3	d_4	c_5	$\mathbf{b_6}$	$\mathbf{a_7}$

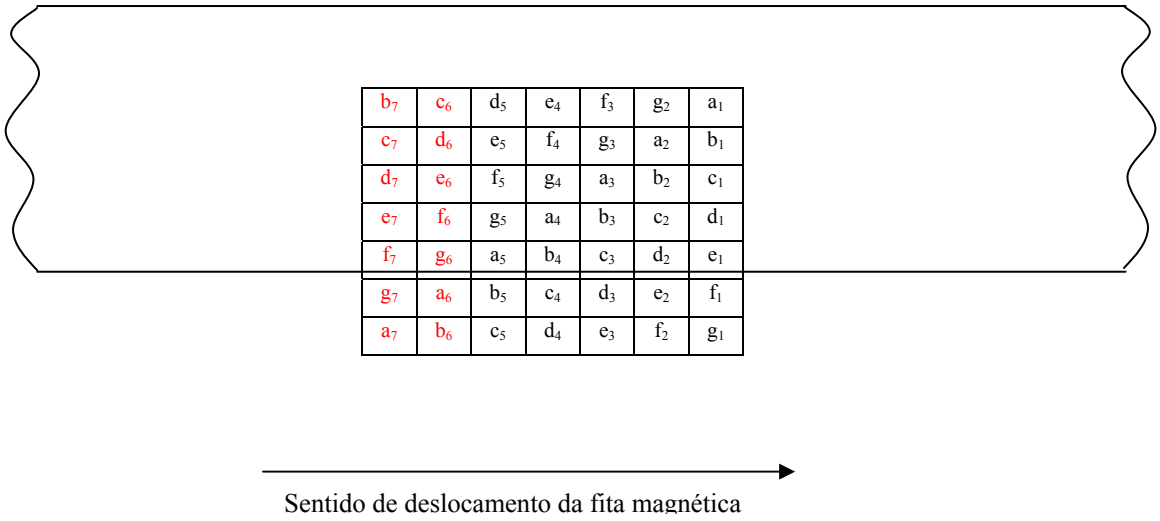


Figura 6.14: Formato de gravação do código matricial sobre a fita magnética.

As considerações feitas anteriormente com relação a retardos, utilização de bancos de memórias e *gap* também podem ser aplicados neste caso.

O principal formato de mancha de erros que poderá ser corrigida por este método de gravação está ilustrado na Figura 6.15. Na estrutura matricial representada, as linhas representam as cabeças de gravação magnética.

a_1	g_2	f_3	e_4	d_5	**c_6**	**b_7**
b_1	a_2	g_3	f_4	e_5	**d_6**	**c_7**
c_1	b_2	a_3	g_4	f_5	**e_6**	**d_7**
d_1	c_2	b_3	a_4	g_5	**f_6**	**e_7**
e_1	d_2	c_3	b_4	a_5	**g_6**	**f_7**
f_1	e_2	d_3	c_4	b_5	**a_6**	**g_7**
g_1	f_2	e_3	d_4	c_5	**b_6**	**a_7**

Figura 6.15: Surto de erros em uma cabeça magnética.

6.3 Conclusão

Foi mostrada a forma de implementação prática dos códigos matriciais com entrelaçamento unidimensional proposto neste trabalho. Esta proposta se mostra vantajosa porquanto garante a correção tanto de surtos de erros unidimensionais quanto de manchas bidimensionais de erros. As desvantagens inerentes ao entrelaçamento podem ser minimizadas pela utilização de lógica adequada usando banco de memórias.

Capítulo 7

Conclusões

Esta dissertação trata de códigos corretores de erros empregados em Sistemas de Gravação Magnética, com o intuito de salvaguardar os dados contra possíveis perdas acarretadas por problemas na mídia e/ou nos dispositivos de gravação magnética (cabeças de gravação magnética).

Numa primeira etapa, é proposto o entrelaçamento unidimensional como uma técnica prática para corrigir múltiplas manchas de erros em uma matriz de dígitos n_1 x n_2, em que as n_1 linhas da palavra-código são codificadas por um código corretor de erros aleatórios, do tipo MDS (máxima distância), como é o caso dos códigos *BCH* e Reed-Solomon. O entrelaçamento, assim proposto, modifica apenas a posição vertical dos símbolos constituíntes da palavra-código deste código matricial. Consequentemente, adquire-se a capacidade de correção de múltiplas manchas de erros compactas, que são definidas com base na métrica de Lee [35]. No entrelaçamento convencional usado para combater surtos de erros em transmissão de dados seriais, uma matriz de dígitos n_1 x n_2 contém palavras-código nas linhas e os dígitos da matriz são extraídos pela leitura das colunas. O entrelaçamento convencional permite a correção de no máximo t surtos de erros de comprimento n_1. A eficiência do entrelaçamento unidimensional proposto, medida em termos de redundância de código empregado e o número de erros que podem ser corrigidos é, pelo menos, a mesma que do entrelaçamento convencional, uma vez que quaisquer t manchas de erros casadas em cada matriz de dígitos são corrigíveis, como podem ser também algumas outras manchas de erros.

Numa segunda etapa, é feita uma modificação no entrelaçamento unidimensional proposto, graças ao uso de códigos corretores de surtos de erros na codificação das linhas de

uma matriz de dígitos n_1 x n_2 quadrada, em que $n_1 = n_2 = n$, ou seja, n x n. Neste caso, ainda é empregado o mesmo tipo de entrelaçamento nas colunas da palavra-código do código matricial. Esta modificação se mostra interessante, porquanto observa-se uma eficiência maior deste esquema comparada com o entrelaçamento comum, pelo fato de que, para valores fixos de n (comprimento do código) e k (dimensão do código), usualmente o número b (comprimento do surto de erros capaz de ser corrigido) é maior do que t (capacidade de correção de erros aleatórios). Consequentemente, para a correção de manchas compactas de erros de um dado peso, precisa-se de matrizes menores quando se utiliza estes tipos de códigos corretores de surtos de erros. Por outro lado, fixando a dimensão da matriz, observa-se que, com o uso de códigos corretores de surtos de erros nas suas linhas, consegue-se corrigir manchas compactas de erros com muito mais erros do que se fossem empregados códigos corretores de erros aleatórios. No entanto, é importante salientar que uma solução não invalida a outra, dado que os formatos das manchas compactas de erros, definidos em cada caso, são distintos e, portanto, dependendo do tipo de mancha de erros que afeta o sistema de gravação, pode-se empregar eficientemente uma das duas soluções propostas.

Numa terceira etapa, é proposta uma ligeira modificação no esquema de criptografia de chave secreta desenvolvido por Campello de Souza [41] e Rocha [44], mediante o emprego de códigos matriciais com entrelaçamento unidimensional, nos quais as linhas podem ser geradas por um código corretor de erros aleatórios ou por um código de correção de surto de erros. Este esquema, ao usar códigos corretores de erros aleatórios, mostra-se interessante uma vez que introduz mais uma componente de chave secreta correspondente ao valor β da matriz T de transformação linear, que é a responsável pelo entrelaçamento unidimensional. Esta solução pode ser empregada não só em sistemas de gravação magnética como em diversas outras aplicações na área de comunicação.

Finalmente, é ilustrada uma forma prática de implementação das duas versões propostas de entrelaçamento unidimensional, que se mostram vantajosas porquanto garantem a correção tanto de surtos de erros unidimensionais quanto de manchas bidimensionais de erros. As desvantagens inerentes ao entrelaçamento, como por exemplo o retardo de gravação e reprodução, podem ser minimizadas pela utilização de lógica adequada, em especial de banco de memórias.

Por mais que a pesquisa tenha apresentado resultados satisfatórios naquilo que foi proposto, ou seja, na utilização de códigos matriciais com entrelaçamento unidimensional para sistemas de gravação magnética, capazes de corrigir eficientemente manchas compactas de erros, muito ainda pode ser realizado para a sua melhoria. Dentre estes tópicos de pesquisas futuras pode-se citar:

a) A pesquisa por novos códigos corretores de erros aleatórios, do tipo MDS, que tenham características de codificação e decodificação simplificada. Porquanto, a codificação das linhas de códigos matriciais por códigos como BCH ou Reed-Solomon, por exemplo, faz com que se perca uma das grandes vantagens de códigos matriciais, que é a não utilização de lógica de campo finito para a sua codificação e a decodificação.
b) A pesquisa por novos códigos corretores de surtos de erros que atendam a dois critérios: Limite de Reiger [56] e simplicidade de codificação/decodificação. Este desenvolvimento possibilitará o uso de códigos matriciais com entrelaçamento undimensional de baixa redundância, portanto mais eficientes, e com baixa complexidade de codificação/decodificação.
c) Estudo da capacidade de canais com memória ao se usar os esquemas propostos.
d) Estudo e otimização do retardo de entrelaçamento e desentrelaçamento dos esquemas propostos.
e) Desenvolvimento de um esquema de criptografia de chave secreta que utiliza entrelaçamento unidimensional e permutação adaptativa [48] e análise matemática do esforço de quebra desta cifra.

Apêndice I

Códigos BCH e Reed Solomon

I.1 Introdução

Códigos Reed Solomon e *BCH* formam o cerne dos códigos algébricos conhecidos mais eficazes e têm sido disseminados em várias aplicações nos últimos trinta anos.

O código *BCH* deve seu nome às iniciais dos seus autores Bose, Chaudhuri e Hocquenghem e teve sua publicação em 1959, em francês por Hocquenghem, e em 1960, em inglês por Bose e Chaudhuri.

O código Reed Solomon que recebeu o nome dos seus autores foi primeiro descrito em junho de 1960.

O código Reed Solomon é um caso especial do código *BCH*, do tipo MDS (distância máxima separável), ou seja, as palavras-código pertencentes ao código Reed- Solomon apresentam máxima distância de Hamming entre si, permitida pelo número de dígitos de verificação de paridade do código.

I.2 Polinômio Gerador para Códigos BCH

Quando se constrói um código cíclico qualquer, não se sabe *a priori* qual será a distância mínima resultante. Dado um polinômio gerador arbitrário *g(x)*, deve-se fazer uma pesquisa computacional para se descobrir a palavra código *c(x)* não nula que possua o menor peso e desta forma se descobrir a distância do código. Por outro lado, códigos *BCH* têm a vantagem de garantir uma distância mínima já na fase de projeto dada uma restrição particular sobre o polinômio gerador. Este resultado é conhecido por cota *BCH*.

Teorema I.1: Seja *C* um código cíclico *q-ário* (*n,k,d*) com polinômio gerador *g(x)*. Seja m_i a ordem multiplicativa de *q* módulo *n* ($GF(q^{m_i})$ é, conseqüentemente, a menor extensão de *GF(q)* que contem uma *n-ésima* raiz primitiva). Seja α a *n-ésima* raiz primitiva.

Selecionando *g(x)* para ser o polinômio de grau mínimo em *GF(q)[x]* tal que $g(\alpha^{\iota})=g(\alpha^{\iota+1})=....=g(\alpha^{\iota+\delta-2}) = 0$ para alguns inteiros $\iota \geq 0$ e $\delta \geq 1$. O polinômio *g(x)* tem consequentemente (δ - 1) potências consecutivas de α como zeros. Este código *C* definido por g*(x)* tem distância mínima $d \geq \delta$.

I.3 Procedimento de Construção de Códigos *BCH*

Para construir um código *BCH q-ário* que corrija *t* erros de comprimento *n*, deve-se:

1. Encontrar uma *n-ésima* raiz primitiva α em um campo $GF(q^{m_i})$, no qual m_i é mínimo.
2. Selecionar (δ - 1) = 2*t* potências consecutivas de α, começando com α^{ι} para um inteiro ι positivo.
3. O polinômio *g(x)* é o mínimo múltiplo comum dos polinômios mínimos para as potências selecionadas de α com respeito a *GF(q)*.

I.3.1 Definições

a) O parâmetro δ no Teorema I.1 representa a distância projetada para o código *BCH* gerado pelo polinômio gerador *g(x)*.

b) Se ι =1 então o código BCH é dito de senso restrito. Se $n = q^{m_i} - 1$ para algum inteiro positivo m_i, então o código *BCH* é primitivo, pois a *n-ésima* raiz de unidade α é um elemento primitivo em $GF(q^{m_i})$.

c) Fixando ι, nota-se que um código *BCH* de comprimento *n* e distância projetada δ_1 contém como subespaços lineares todos os códigos *BCH* de comprimento *n* com distância projetada $\delta_2 \geq \delta_1$. Isto pode ser provado da seguinte maneira:

$d = \delta_1 \quad \rightarrow g_1(x) = (x - \alpha^t)(x-\alpha^{t+1})\ldots\ldots\ldots\ldots(x-\alpha^{t+\delta 1-2})$ (I.1)

$d = \delta_2 \geq \delta_1 \rightarrow g_2(x) = (x - \alpha^t)(x-\alpha^{t+1})\ldots\ldots\ldots\ldots(x-\alpha^{t+\delta 1-2})\ldots\ldots\ldots(x - \alpha^{t+\delta 2-2})$, (I.2)

logo,

- $g_2(x) = m(x)\, g_1(x)$
- Todos os múltiplos de $g_2(x)$ são múltiplos de $g_1(x)$
- Todas as palavras-código em C_2 são palavras-código em C_1

I.3.2 Exemplos de construção do código BCH

a) Códigos Binários de comprimento 31

Seja α raiz primitiva do polinômio primitivo $x^5 + x^2 + 1$. Ele é consequentemente um elemento primitivo do campo *GF(32)*. Uma vez que 31 é da forma $2^{m_i} - 1$, este código *BCH* é dito primitivo.

Classes ciclotômicas (C_r)	**Polinômios mínimos ($M_r(x)$)**
$C_0 = \{0\}$	$M_{(0)}(x) = x + 1$
$C_1 = \{ 1, 2, 4, 8, 16\}$	$M_{(1)}(x) = x^5 + x^2 + 1$
$C_3 = \{ 3, 6, 12, 24, 17\}$	$M_{(3)}(x) = x^5 + x^4 + x^3 + x^2 + 1$
$C_5 = \{ 5, 10, 20, 9, 18\}$	$M_{(5)}(x) = x^5 + x^4 + x^2 + x + 1$
$C_7 = \{ 7, 14, 28, 25, 19\}$	$M_{(7)}(x) = x^5 + x^3 + x^2 + x + 1$
$C_{11} = \{ 11, 22, 13, 26, 21\}$	$M_{(11)}(x) = x^5 + x^4 + x^3 + x + 1$
$C_{15} = \{ 15, 30, 29, 27, 23\}$	$M_{(11)}(x) = x^5 + x^3 + 1$

Se *C* é para ser um código cíclico binário, então ele deve ter um polinômio gerador *g(x)* que é fatorado por um ou mais polinômios anteriores. Se *C* é para ser um código *BCH* de correção de *t* erros, então *g(x)* deve ter como zeros 2*t* potências consecutivas de α.

a.1) Código *BCH* primitivo de senso estrito para correção de um erro

Dada a restrição anterior, então $\iota = 1$ e $\delta = 3$. Portanto o polinômio gerador deve ter como raízes α e α^2. Desta forma $M_{(1)}(x)$ é o polinômio mínimo de ambos α e α^2. Daí o polinômio gerador ser gerado por:

$$g(x) = MMC\,(M_{(1)}(x)M_{(2)}(x)) = M_{(1)}(x) = x^5 + x^2 + 1.$$

Uma vez que o grau do polinômio gerador é 5, a dimensão do código resultante é 31 –5 = 26. O polinômio gerador $g(x)$ define portanto um código *BCH* (31,26) binário de correção de um único erro.

A matriz de verificação de paridade para este código é mostrada seguir. [32]

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & ..... & \alpha^{29} & \alpha^{30} \\ 1 & \alpha^2 & \alpha^4 & ..... & \alpha^{27} & \alpha^{29} \end{bmatrix}.$$

a.2) Código BCH primitivo de senso estrito para correção de dois erros

Novamente $\iota = 1$ e δ é incrementado para cinco. Portanto o polinômio gerador deve ter como raízes α, α^2, α^3 e α^4. Desta forma, $M_{(1)}(x)$ é o polinômio mínimo de α e α^2 e $M_{(3)}(x)$ é o polinômio mínimo de α^3 e α^4. Daí o polinômio gerador ser formado por:

$$g(x) = MMC\,(M_{(1)}(x)M_{(2)}(x)M_{(3)}(x)M_{(4)}(x)) = M_{(1)}(x)M_{(3)}(x)$$
$$= (x^5 + x^2 + 1).(x^5 + x^4 + x^3 + x^2 + 1) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1.$$

Uma vez que o grau do polinômio gerador é 10, a dimensão do código resultante é 31 –10 = 21. O polinômio gerador $g(x)$ define portanto um código *BCH* (31,21) binário de correção de duplo erro.

A matriz de verificação de paridade para este código é mostrada a seguir. [32]

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & ..... & \alpha^{29} & \alpha^{30} \\ 1 & \alpha^2 & \alpha^4 & ..... & \alpha^{27} & \alpha^{29} \\ 1 & \alpha^3 & \alpha^6 & ..... & \alpha^{25} & \alpha^{28} \\ 1 & \alpha^4 & \alpha^8 & ..... & \alpha^{23} & \alpha^{27} \end{bmatrix}.$$

b) Código *4-ário* de comprimento 21

É preciso primeiro identificar γ, uma raiz primitiva de ordem 21. O menor valor de m_i para o qual a relação $21 | 4^{m_i} - 1$ é verdadeira é três, e portanto γ pode ser encontrado em $GF(4^3) = GF(64)$, mas não em campo menor. Seja o polinômio primitivo $x^6 + x + 1$ para $GF(64)$ e α a raiz primitiva, ou seja, $\alpha^{63} = 1$. Como $\gamma^{21} = 1$ então $\gamma = \alpha^3$. Os elementos de $GF(4)$ são:

$$\{ 0,1, \gamma^7 = \varphi, \gamma^{14} = \varphi^2 \}.$$

As classes ciclotômicas módulo 21 com respeito a $GF(4)$ e seus associados polinômios mínimos são:

Classes ciclotômicas (C_r)	**Polinômios mínimos ($M_r(x)$)**
$C_0 = \{0\}$	$M_{(0)}(x) = x + 1$
$C_1 = \{ 1, 4, 16\}$	$M_{(1)}(x) = x^3 + \varphi^2 x + 1$
$C_2 = \{ 2, 8, 11\}$	$M_{(2)}(x) = x^3 + \varphi x + 1$
$C_3 = \{ 3, 12, 6\}$	$M_{(3)}(x) = x^3 + x^2 + 1$
$C_5 = \{ 5, 20, 17\}$	$M_{(5)}(x) = x^3 + \varphi^2 x^2 + 1$
$C_7 = \{ 7\}$	$M_{(7)}(x) = x + \varphi$
$C_9 = \{ 9, 15, 18\}$	$M_{(9)}(x) = x^3 + x + 1$
$C_{10} = \{ 10, 19, 13\}$	$M_{(10)}(x) = x^3 + \varphi x^2 + 1$
$C_{14} = \{14\}$	$M_{(14)}(x) = x + \varphi^2$

b.1) Código *BCH 4-ário* de senso estrito para correção de um erro

O polinômio gerador deve ter como raízes γ e γ^2. Daí o polinômio gerador ser formado por:

$$g(x) = MMC\,(M_{(1)}(x)M_{(2)}(x)) = M_{(1)}(x)M_{(2)}(x) = (x^3 + \varphi^2 x + 1).(x^3 + \varphi x + 1),$$

então

$$g(x) = x^6 + x^4 + x^2 + x + 1.$$

Como o grau de $g(x)$ é 6, o código *BCH 4-ário* de correção de único erro é do tipo (21,15) e a matriz de verificação de paridade é mostrada a seguir. [32]

$$H = \begin{bmatrix} 1 & \gamma & \gamma^2 & ..... & \gamma^{19} & \gamma^{20} \\ 1 & \gamma^2 & \gamma^4 & ..... & \gamma^{17} & \gamma^{19} \end{bmatrix} = \begin{bmatrix} 1 & \alpha^3 & \alpha^6 & ..... & \alpha^{57} & \alpha^{60} \\ 1 & \alpha^6 & \alpha^{12} & ..... & \alpha^{51} & \alpha^{57} \end{bmatrix}$$

b.2) Código BCH 4-ário de senso estrito para correção de dois erros

O polinômio gerador deve ter como raízes γ, γ^2, γ^3 e γ^4. Daí o polinômio gerador ser:

$$\begin{aligned} g(x) &= MMC\,(M_{(1)}(x)M_{(2)}(x)M_{(3)}(x)M_{(4)}(x)) = M_{(1)}(x)M_{(2)}(x)M_{(3)}(x) \\ &= (x^3 + \varphi^2 x + 1).(x^3 + \varphi x + 1)\,(x^3 + x^2 + 1) \\ &= x^9 + x^8 + x^7 + x^5 + x^4 + x + 1. \end{aligned}$$

Uma vez que o grau do polinômio gerador é nove, o código resultante é do tipo (21,12).

A matriz de verificação de paridade para este código é mostrada a seguir.[32]

$$H = \begin{bmatrix} 1 & \gamma & \gamma^2 & ..... & \gamma^{19} & \gamma^{20} \\ 1 & \gamma^2 & \gamma^4 & ..... & \gamma^{17} & \gamma^{19} \\ 1 & \gamma^3 & \gamma^6 & ..... & \gamma^{15} & \gamma^{18} \\ 1 & \gamma^4 & \gamma^8 & ..... & \gamma^{13} & \gamma^{17} \end{bmatrix} = \begin{bmatrix} 1 & \alpha^3 & \alpha^6 & ..... & \alpha^{57} & \alpha^{60} \\ 1 & \alpha^6 & \alpha^{12} & ..... & \alpha^{51} & \alpha^{57} \\ 1 & \alpha^9 & \alpha^{18} & ..... & \alpha^{45} & \alpha^{54} \\ 1 & \alpha^{12} & \alpha^{24} & ..... & \alpha^{39} & \alpha^{51} \end{bmatrix}.$$

I.4 **Códigos Reed-Solomon**

Um código Reed-Solomon é um código BCH q^{m_i} *-ário* de comprimento $q^{m_i} - 1$. Um código Reed-Solomon (n,k,d) tem distância mínima ($n - k + 1$). O código Reed- Solomon é chamado de MDS ou código de máxima distância. [32]

I.4.1 Exemplos de construção de códigos Reed-Solomon

a) Código Reed-Solomon *8-ário* para correção de dois erros

Seja α a raiz primitiva de $x^3 + x + 1$ e consequentemente de ordem sete. O campo de Galois *GF(8)* pode ser representado por potências consecutivas de α:

$$\alpha = \alpha \qquad \alpha^5 = \alpha^2 + \alpha + 1$$
$$\alpha^2 = \alpha^2 \qquad \alpha^6 = \alpha^2 + 1$$
$$\alpha^3 = \alpha + 1 \qquad \alpha^7 = 1$$
$$\alpha^4 = \alpha^2 + \alpha \qquad 0 = 0$$

Se o código resultante é para ser de correção de duplo erro, deve ter $2t = 4$ potências consecutivas de α como zeros. Um polinômio gerador de senso estrito é construído da seguinte forma:

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) = x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3.$$

Uma vez que o polinômio gerador tem grau 4, o código Reed-Solomon (7,3) que ele define tem dimensão três sobre *GF(8)* e consequentemente $8^3 = 512$ palavras-código. A matriz de verificação de paridade é mostrada a seguir.[32]

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha & \alpha^4 \\ 1 & \alpha^4 & \alpha & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{bmatrix}.$$

b) Código Reed-Solomon *64-ário* para correção de três erros, comprimento 63

Seja α a raiz do polinômio primitivo $x^6 + x + 1$. Para o código ser de correção de três erros, deve ter seis potências consecutivas de α como zeros, ou seja:

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)$$

$$= x^6 + \alpha^{59}x^5 + \alpha^{48}x^4 + \alpha^{43}x^3 + \alpha^{55}x^2 + \alpha^{10}x + \alpha^{21}.$$

O polinômio $g(x)$ define um código Reed-Solomon (63,57) para correção de triplo erro. Este código tem $64^{57} = 8.96 \times 10^{102}$ palavras-código.

c) Código Reed-Solomon *7-ário* para correção de dois erros com comprimento seis.

As operações em *GF(7)* são simples adições e multiplicações módulo sete. Os símbolos inteiros { 0, 1, 2, 3, 4, 5, 6} são usados para representar os elementos de campo. Para *GF(7)* o número cinco é um elemento primitivo. Uma vez que se precisa de quatro potências consecutivas de cinco como zeros, então $g(x)$ é

$$g(x) = (x-5)(x-5^2)(x-5^3)(x-5^4) = x^4 + 4x^3 + 6x^2 + 5x + 2.$$

O polinômio $g(x)$ define um código Reed-solomon de correção de dois erros do tipo (6,2). Este código possui $7^2 = 49$ palavras-código.

I.5 Emprego dos Códigos BCH e Reed Solomon no Código Matricial de Correção de Erros para Gravação Magnética.

Seja um código matricial com n_1 = 5, indicando que são utilizados cinco palavras-código. Para tornar este código matricial cíclico e capaz de corrigir manchas de erros compactas de peso igual a n_1, os valores de n_2 (número de colunas) poderão ser 5, 10, 15, 20,... etc. Supõe-se n_2 =15 (que corresponde ao comprimento do código corretor de erros na linha) e que o código seja capaz de corrigir um erro aleatório. Neste caso é possível desenvolver este código *BCH* em $GF(2^4)$. Seja o polinômio primitivo gerador do campo igual a $x^4 + x + 1$ e α a raiz primitiva de ordem 15, logo

Classes ciclotômicas (C_r)	**Polinômios mínimos ($M_r(x)$)**
$C_0 = \{0\}$	$M_{(0)}(x) = x + 1$
$C_1 = \{ 1, 2, 4, 8\}$	$M_{(1)}(x) = x^4 + x + 1$
$C_3 = \{ 3, 6, 12, 9\}$	$M_{(3)}(x) = x^4 + x^3 + x^2 + x + 1$
$C_5 = \{ 5, 10\}$	$M_{(5)}(x) = x^2 + x + 1$
$C_7 = \{ 7, 14, 13, 11\}$	$M_{(7)}(x) = x^4 + x^3 + 1$

Para correção de apenas um erro precisa-se de duas potências consecutivas de α no polinômio gerador, ou seja, α e α^2, logo

$$g(x) = M_{(1)}(x) = x^4 + x + 1.$$

Como o grau do polinômio é quatro, conclui-se que o código *BCH* é do tipo (15,11), ou seja, das quinze colunas do código matricial, 11 correspondem à dimensão do código e as outras quatro correspondem à redundância.

Para a correção de dois erros este número cresce para oito colunas de redundância.

Caso seja utilizado um código Reed-Solomon, trabalha-se para um campo de Galois *GF(16)* que possui os seguintes elementos:

$\alpha = \alpha$

$\alpha^2 = \alpha^2$

$\alpha^3 = \alpha^3$

$\alpha^4 = \alpha + 1$

$\alpha^5 = \alpha^2 + \alpha$

$\alpha^6 = \alpha^3 + \alpha^2$

$\alpha^7 = \alpha^3 + \alpha + 1$

$\alpha^8 = \alpha^2 + 1$

$\alpha^9 = \alpha^3 + \alpha$

$\alpha^{10} = \alpha^2 + \alpha + 1$

$\alpha^{11} = \alpha^3 + \alpha^2 + \alpha$

$\alpha^{12} = \alpha^3 + \alpha^2 + \alpha + 1$

$\alpha^{13} = \alpha^3 + \alpha^2 + 1$

$\alpha^{14} = \alpha^3 + 1$

$\alpha^{15} = 1$

$0 = 0$

Estes correspondem a todos os elementos de *GF(16)*. Para se obter a correção de apenas um erro, basta

$g(x) = (x+\alpha).(x+ \alpha^2) = x^2 + \alpha^5 x + \alpha^3$, que corresponde ao polinômio primitivo sobre *GF(4)*.

Neste caso, o código Reed-Solomon é do tipo (15,13), ou seja, a dimensão do código é igual a 13 e o número de colunas redundantes é igual a dois.

Como já foi mencionado, uma vez que as linhas do código matricial estejam codificadas, seja usando código *BCH* ou Reed-Solomon, o passo seguinte é aplicar a matriz de transformação linear, *T*, antes de gravar estes dados na fita magnética.

Apêndice II

Tabela de β^1 para Alguns Valores Válidos de n_1

n_1	d_L	N_{dL}	Valores de β^1
5	2	5	2; 3 (**)
7	2	5	2; 3; 4; 5
8	2	5	3; 5
10	3	8	3; 7
11	3	8	3; 4; 7; 8
12	3	8	5; 7
13	4	13	5; 8 (**)
15	4	13	4; 11
17	4	13	4; 5; 7; 10; 12; 13
18	4	13	5; 7; 11; 13
21	5	18	8; 13
22	5	18	5; 9; 13; 17
23	5	18	5; 9; 14; 18
24	5	18	5; 7; 17; 19
25	6	25	7; 18 (**)
29	6	25	8; 11; 12; 17; 18; 21
31	6	25	7; 9; 12; 13; 18; 19; 22; 24
32	6	25	7; 9; 23; 25
34	7	32	13; 21
37	7	32	8; 14; 23; 29
38	7	32	7; 11; 27; 31
39	7	32	7; 11; 28; 32
40	7	32	7; 9; 11; 17; 23; 29; 31; 33
41	8	41	9; 32 (**)
43	8	41	12; 18; 25; 31
45	8	41	8; 17; 19; 26; 28; 37
47	8	41	13; 18; 29; 34
49	8	41	9; 11; 38; 40
50	8	41	9; 11; 19; 21; 29; 31; 39; 41
55	9	50	12; 21; 23; 32; 34; 43
58	9	50	9; 13; 17; 41; 45; 49

59	9	50	9; 13; 46; 50
60	9	50	11; 13; 23; 37; 47; 49
61	10	61	11; 50 (**)
65	10	61	18; 47
67	10	61	12; 28; 39; 55
69	10	61	19; 29; 40; 50
71	10	61	11; 13; 16; 21; 27; 31; 40; 44; 50; 55; 58; 60
72	10	61	11; 13; 59; 61
74	11	72	31; 43
76	11	72	21; 29; 47; 55
78	11	72	17; 23; 55; 61
79	11	72	12; 33; 46; 67
81	11	72	31; 34; 47; 50
82	11	72	11; 15; 67 ;71
83	11	72	11; 15; 18; 23; 60; 65; 68; 72
84	11	72	11; 13; 19; 23; 25; 31; 37; 47; 53; 59; 61; 65; 71; 73
85	12	85	13; 72 (**)
89	12	85	16; 34; 39; 50; 55; 73
91	12	85	12; 25; 27; 38; 40; 51; 53; 64; 66; 79
95	12	85	17; 28; 67; 78
n_1	d_L	N_{dL}	**Valores de β^1**
97	12	85	13; 15; 21; 22; 37; 60; 75; 76; 82; 84
98	12	85	13; 15; 27; 29; 41; 43; 55; 57; 69; 71; 83 ;85
105	13	98	16; 29; 31; 44; 46; 59; 61; 74; 76; 89
106	13	98	19; 23; 39; 67; 83; 87
107	13	98	41; 47; 60; 66
109	13	98	30; 40; 69; 79
110	13	98	13; 17; 93; 97
111	13	98	13; 17; 94; 98
112	13	98	13; 15; 17; 31; 33; 41; 43; 47; 65; 69; 71; 79; 81; 95; 97; 99
113	14	113	15; 98 (**)
115	14	113	34; 44; 71; 81
117	14	113	43; 49; 68; 74
121	14	113	16; 53; 68; 105
123	14	113	22; 28; 34; 47; 76; 89; 95; 101
125	14	113	19; 27; 37; 46; 79; 88; 98; 106
127	14	113	15; 17; 29; 35; 92; 98; 110; 112
128	14	113	15; 17; 47; 49; 79; 81; 111; 113
130	15	128	57; 73

134	15	128	29; 37; 97; 105
136	15	128	31; 57; 79; 105
137	15	128	16; 60; 77; 121
139	15	128	30; 41; 51; 61; 78; 88; 98; 109
142	15	128	15; 19; 123; 127
143	15	128	15; 19; 31; 60; 83; 112; 124; 128
144	15	128	17; 19; 31; 53; 55; 65; 79; 89; 91; 113; 125; 127
145	16	145	17; 128 (**)
149	16	145	34; 44; 57; 92; 105; 115
151	16	145	20; 68; 83; 131
153	16	145	16; 35; 67; 86; 118; 137
155	16	145	57; 68; 87; 98
157	16	145	28; 34; 60; 97; 123; 129
159	16	145	44; 47; 112; 115
161	16	145	17; 19; 142; 144
162	16	145	17; 19; 35; 37; 71; 73; 89; 91; 125; 127; 143; 145
166	17	162	49; 61; 105; 117
170	17	162	47; 123
171	17	162	20; 37; 77; 94; 134; 151
173	17	162	51; 78; 95; 122
174	17	162	23; 31; 53; 73; 101; 121; 143; 151
177	17	162	49; 65; 112; 128
178	17	162	17; 21; 27; 33; 145; 151; 157; 161
179	17	162	17; 21; 158; 162
180	17	162	17; 19; 41; 53; 79; 101; 127; 139; 161; 163
181	18	181	19; 162 (**)
185	18	181	68; 117
187	18	181	57; 82; 105; 130
191	18	181	20; 34; 73; 86; 105; 118; 157; 171
193	18	181	44; 57; 71; 81; 87; 106; 112; 122; 136; 149
197	18	181	23; 26; 35; 45; 53; 60; 137; 144; 152; 162; 171; 174
199	18	181	19; 21; 37; 43; 55; 76; 123; 144; 156; 162; 178; 180
200	18	181	19; 21; 59; 61; 139; 141; 179; 181
202	19	200	91; 111
206	19	200	47; 57; 149; 159
n_1	d_L	N_{dL}	**Valores de β^1**
208	19	200	37; 45; 163; 171
211	19	200	20; 95; 116; 191
212	19	200	57; 81; 89; 93; 119; 123; 131; 155

213	19	200	59; 65; 148; 154
214	19	200	25; 77; 137; 189
215	19	200	49; 79; 136; 166
217	19	200	47; 60; 64; 78; 139; 153; 157; 170
218	19	200	19; 23; 33; 185; 195; 199
219	19	200	19; 23; 50; 92; 127; 169; 196; 200
220	19	200	19; 21; 23; 29; 39; 41; 59; 61; 67; 79; 81; 91; 101; 119; 129; 139; 141; 153; 159; 161; 179; 181; 191; 197;199; 201
221	20	221	21; 200 (**)
223	20	221	68; 82; 141; 155
227	20	221	61; 67; 160; 166
229	20	221	24; 105; 124; 205
231	20	221	20; 41; 43; 50; 62; 64; 83; 85; 97; 104; 106; 125; 127; 134; 146; 148; 167; 169; 181; 188; 190; 211
233	20	221	71; 89; 105; 128; 144; 162
237	20	221	44; 70; 98; 104; 133; 139; 167; 193
239	20	221	25; 86; 153; 214
241	20	221	21; 23; 55; 92; 149; 186; 218; 220
242	20	221	21; 23; 43; 45; 65; 67; 87; 89; 109; 111; 131; 133; 153; 155; 175; 177; 197; 199; 219; 221
250	21	242	57; 193
253	21	242	24; 45; 47; 68; 70; 91; 93; 114; 116; 137; 139; 160; 162; 183; 185; 206; 208; 229
254	21	242	55; 75; 97; 105; 149; 157; 179; 199
257	21	242	69; 108; 149; 188
259	21	242	46; 59; 79; 107; 152; 180; 200; 213
262	21	242	21; 25; 41; 115; 147; 221; 237; 241
263	21	242	21; 25; 57; 60; 203; 206; 238; 242
264	21	242	23; 25; 47; 49; 71; 73; 95; 97; 109; 119; 145; 155; 167; 169; 191; 193; 215; 217; 239; 241
265	22	265	23; 242 (**)
269	22	265	50; 82; 113; 156; 187; 219
271	22	265	75; 112; 159; 196
277	22	265	24; 60; 127; 150; 217; 253
279	22	265	85; 128; 151; 194
281	22	265	44; 50; 64; 83; 101; 118; 163; 180; 198; 217; 231; 237
283	22	265	76; 104; 108; 117; 166; 175; 179; 207
285	22	265	43; 53; 232; 242
287	22	265	23; 25; 30; 45; 51; 67; 220; 236; 242; 257; 262; 264
288	22	265	23; 25;119; 121; 167; 169; 263; 265
290	23	288	133; 157
292	23	288	89; 105; 187; 203

294	23	288	67; 79; 215; 227
296	23	288	55; 113; 183; 241
298	23	288	45; 53; 245; 253
300	23	288	47; 83; 217; 253
301	23	288	24; 138; 163; 277
302	23	288	111; 117; 185; 191
303	23	288	109; 139; 164; 194
304	23	288	71; 137; 167; 233
305	23	288	46; 82; 93; 126; 179; 212; 223; 259
n_1	d_L	N_{dL}	**Valores de β^1**
306	23	288	29; 95; 211; 277
307	23	288	57; 70; 119; 129; 178; 188; 237; 250
309	23	288	55; 67; 83; 118; 191; 226; 242; 254
310	23	288	23; 27; 283; 287
311	23	288	23; 27; 47; 86; 225; 264; 284; 288
312	23	288	23; 25; 41; 47; 49; 71; 73; 95; 97; 119; 121; 131; 137; 145; 167; 175; 181; 191; 193; 215; 217; 239; 241; 263; 265; 271; 287; 289
313	24	313	25; 288 (**)
317	24	313	114; 121; 131; 186; 196; 203
319	24	313	50; 134; 185; 269
323	24	313	28; 60; 70; 150; 173; 253; 263; 295
325	24	313	24; 49; 51; 74; 76; 99; 101; 124; 126; 149; 151; 174; 176; 199; 201; 224; 226; 249; 251; 274; 276; 301
329	24	313	75; 136; 193; 254
331	24	313	50; 89; 119; 139; 192; 212; 242; 281
335	24	313	29; 44; 71; 99; 104; 151; 184; 231; 236; 264; 291; 306
337	24	313	25; 27; 60; 73; 148; 189; 264; 277; 310; 312
338	24	313	25; 27; 51; 53; 77; 79; 103; 105; 129; 131; 155; 157; 181; 183; 207; 209; 233; 235; 259; 261; 285; 287; 311; 313
346	25	338	75; 93; 143; 203; 253; 271
351	25	338	28; 53; 55; 80; 82; 107; 109; 134; 136; 161; 163; 188; 190; 215; 217; 242; 244; 269; 271; 296; 298; 323
353	25	338	127; 164; 189; 226
355	25	338	77; 83; 272; 278
358	25	338	31; 47; 99; 127; 231; 259; 311; 327
359	25	338	76; 137; 222; 283
361	25	338	67; 97; 110; 128; 233; 251; 264; 294
362	25	338	25; 29; 49; 107; 133; 159; 203; 229; 255; 313; 333; 337
363	25	338	25; 29; 334; 338

364	25	338	25; 27; 29; 55; 57; 79; 83; 85; 111; 113; 129; 131; 139; 141; 167; 197; 223; 225; 233; 235; 251; 253; 279; 281; 285; 307; 309; 335; 337; 339
365	26	365	27; 338 (**)
367	26	365	114; 132; 235; 253
369	26	365	80; 143; 226; 289
373	26	365	79; 85; 288; 294
375	26	365	133; 172; 203; 242
377	26	365	70; 307
379	26	365	28; 176; 203; 351
381	26	365	50; 89; 137; 160; 221; 244; 292; 331
383	26	365	60; 83; 103; 119; 264; 280; 300; 323
387	26	365	82; 104; 107; 118; 160; 170; 217; 227; 269; 280; 283; 305
389	26	365	31; 51; 61; 115; 138; 251; 274; 328; 338; 358
391	26	365	27; 29; 53; 59; 332; 338; 362; 364
392	26	365	27; 29; 83; 85; 139; 141; 251; 253; 307; 309; 363; 365
394	27	392	183; 211
398	27	392	93; 107; 291; 305
404	27	392	53; 61; 75; 167; 237; 329; 343; 351
406	27	392	55; 155; 251; 351
407	27	392	28; 189; 218; 379
409	27	392	110; 127; 145; 190; 219; 264; 282; 299
410	27	392	73; 337
412	27	392	109; 173; 181; 189; 223; 231; 239; 303
413	27	392	111; 160; 253; 302
415	27	392	77; 97; 318; 338
n_1	d_L	N_{dL}	**Valores de β^1**
416	27	392	115; 123; 293; 301
418	27	392	27; 31; 387; 391
419	27	392	27; 31; 55; 160; 259; 364; 388; 392
420	27	392	29; 31; 89; 113; 149; 151; 197; 223; 269; 271; 307; 331; 389; 391
421	28	421	29; 392 (**)
425	28	421	132; 293
431	28	421	80; 114; 155; 167; 264; 276; 317; 351
433	28	421	32; 179; 203; 230; 254; 401
435	28	421	28; 59; 202; 233; 376; 407
437	28	421	66; 155; 192; 203; 234; 245; 282; 371
439	28	421	93; 118; 321; 346
441	28	421	103; 137; 304; 338
443	28	421	60; 96; 101; 131; 186; 193; 250; 257; 312; 342; 347; 383

445	28	421	104; 123; 184; 261; 322; 341
447	28	421	70; 83; 364; 377
449	28	421	29; 31; 80; 85; 174; 206; 243; 275; 364; 369; 418; 420
450	28	421	29; 31; 59; 61; 119; 121; 209; 211; 239; 241; 329; 331; 389; 391; 419; 421
454	29	450	141; 161; 293; 313
458	29	450	85; 97; 107; 351; 361; 373
462	29	450	179; 191; 271; 283
465	29	450	32; 61; 218; 247; 404; 433
466	29	450	73; 83; 101; 203; 263; 365; 383; 393
467	29	450	145; 219; 248; 322
469	29	450	87; 124; 345; 382
470	29	450	71; 89; 139; 169; 301; 331; 381; 399
471	29	450	110; 167; 304; 361
473	29	450	62; 206; 267; 411
474	29	450	35; 55; 131; 149; 181; 199; 275; 293; 325; 343; 419; 439
475	29	450	111; 184; 291; 364
477	29	450	85; 101; 376; 392
478	29	450	29; 33; 57; 109; 369; 421; 445; 449
479	29	450	29; 33; 75; 198; 281; 404; 446; 450
480	29	450	29; 31; 89; 91; 127; 149; 151; 209; 211; 223; 257; 269; 271; 329; 331; 353; 389; 391; 449; 451
481	30	481	31; 450 (**)
485	30	481	172; 313
487	30	481	66; 214; 273; 421
491	30	481	93; 104; 132; 203; 288; 359; 387; 398
493	30	481	155; 191; 229; 264; 302; 338
497	30	481	32; 233; 264; 465
499	30	481	132; 155; 177; 234; 265; 322; 344; 367
503	30	481	60; 66; 76; 109; 133; 139; 208; 221; 282; 295; 364; 370; 394; 427; 437; 443
505	30	481	107; 118; 212; 293; 387; 398
509	30	481	35; 59; 69; 89; 160; 183; 326; 349; 420; 440; 450; 474
511	30	481	31; 33; 61; 67; 80; 198; 313; 431; 444; 450; 478; 480
512	30	481	31; 33; 95; 97; 159; 161; 223; 225; 287; 289; 351; 353; 415; 417; 479; 481
514	31	512	241; 273
518	31	512	121; 137; 381; 397
524	31	512	71; 111; 155; 203; 321; 369; 413; 453
526	31	512	61; 69; 457; 465
529	31	512	32; 248; 281; 497
530	31	512	83; 447

531	31	512	124; 167; 364; 407
n_1	d_L	N_{dL}	**Valores de β^1**
532	31	512	93; 143; 389; 439
533	31	512	95; 101; 141; 189; 344; 392; 432; 438
535	31	512	62; 233; 302; 473
536	31	512	81; 127; 225; 249; 287; 311; 409; 455
537	31	512	142; 157; 208; 236; 301; 329; 380; 395
538	31	512	37; 189; 349; 501
539	31	512	73; 96; 100; 221; 318; 439; 443; 466
541	31	512	71; 160; 168; 190; 351; 373; 381; 470
542	31	512	31; 35; 199; 207; 335; 343; 507; 511
543	31	512	31; 35; 85; 115; 428; 458; 508; 512
544	31	512	31; 33; 35; 63; 65; 95; 97; 101; 103; 127; 129; 159; 161; 169; 171; 191; 193; 223; 225; 237; 239; 257; 287; 305; 307; 319; 321; 351; 353; 373; 375; 383; 385; 415; 417; 441; 443; 447; 449; 479; 481; 509; 511; 513
545	32	545	33; 512 (**)
547	32	545	172; 194; 353; 375
549	32	545	104; 227; 322; 445
551	32	545	163; 240; 311; 388
553	32	545	66; 243; 310; 487
555	32	545	97; 103; 452; 458
557	32	545	118; 132; 173; 384; 425; 439
559	32	545	36; 264; 295; 523
561	32	545	32; 65; 67; 76; 98; 100; 131; 133; 155; 164; 166; 197; 199; 230; 232; 263; 265; 296; 298; 329; 331; 362; 364; 395; 397; 406; 428; 430; 461; 463; 485; 494; 496; 529
563	32	545	177; 264; 299; 386
565	32	545	107; 132; 433; 458
567	32	545	101; 247; 320; 466
569	32	545	66; 86; 250; 319; 483; 503
571	32	545	106; 121; 151; 167; 404; 420; 450; 465
573	32	545	154; 160; 178; 235; 338; 395; 413; 419
575	32	545	37; 202; 373; 538
577	32	545	33; 35; 60; 125; 452; 517; 542; 544
578	32	545	33; 35; 67; 69; 101; 103; 135; 137; 169; 171; 203; 205; 237; 239; 271; 273; 305; 307; 339; 341; 373; 375; 407; 409; 441; 443; 475; 477; 509; 511; 543; 545
586	33	578	111; 155; 227; 359; 431; 475
590	33	578	163; 257; 333; 427
595	33	578	36; 69; 71; 104; 106; 139; 141; 174; 176; 209; 211; 244; 246; 279; 281; 314; 316; 349; 351; 384; 386; 419; 421; 454; 456; 489; 491; 524; 526; 559

598	33	578	81; 111; 167; 251; 347; 431; 487; 517
599	33	578	142; 232; 367; 457
601	33	578	105; 159; 166; 189; 412; 435; 442; 496
602	33	578	79; 187; 221; 235; 367; 381; 415; 523
605	33	578	169; 179; 426; 436
607	33	578	95; 115; 237; 251; 356; 370; 492; 512
610	33	578	33; 37; 233; 377; 573; 577
611	33	578	33; 37; 574; 578
612	33	578	35; 37; 71; 73; 107; 109; 143; 145; 169; 179; 181; 215; 217; 239; 251; 253; 287; 325; 359; 361; 373; 395; 397; 431; 433; 443; 467; 469; 503; 505; 539; 541; 575; 577
613	34	613	35;578 (**)
617	34	613	194; 239; 253; 364; 378; 423
619	34	613	171; 181; 438; 448
623	34	613	111; 118; 132; 174; 449; 491; 505; 512
n_1	d_L	N_{dL}	**Valores de β^1**
625	34	613	146; 244; 381; 479
629	34	613	264; 274; 355; 365
631	34	613	36; 298; 333; 595
633	34	613	199; 299; 334; 434
635	34	613	66; 86; 96; 111; 168; 223; 246; 279; 356; 389; 412; 467; 524; 539; 549; 569
637	34	613	76; 135; 151; 176; 461; 486; 502; 561
639	34	613	100; 121; 169; 262; 377; 470; 518; 539
641	34	613	112; 119; 179; 237; 265; 269; 372; 376; 404; 462; 522; 529
643	34	613	188; 228; 236; 251; 392; 407; 415; 455
645	34	613	67; 77; 101; 281; 364; 544; 568; 578
647	34	613	35; 37; 69; 75; 103; 201; 446; 544; 572; 578; 610; 612
648	34	613	35; 37; 179; 181; 251; 253; 395; 397; 467; 469; 611; 613
650	35	648	307; 343
652	35	648	205; 229; 423; 447
654	35	648	155; 173; 481; 499
656	35	648	139; 269; 387; 517
658	35	648	103; 115; 543; 555
660	35	648	193; 277; 383; 467
662	35	648	79; 243; 419; 583
664	35	648	69; 77; 587; 595
666	35	648	71; 197; 469; 595
667	35	648	36; 315; 352; 631
668	35	648	101; 119; 247; 291; 377; 421; 549; 567
669	35	648	235; 316; 353; 434

670	35	648	277; 283; 387; 393
671	35	648	105; 159; 211; 262; 409; 460; 512; 566
672	35	648	107; 157; 515; 565
673	35	648	178; 188; 247; 276; 397; 426; 485; 495
674	35	648	141; 239; 435; 533
675	35	648	118;143;532;557
676	35	648	177; 297; 305; 317; 359; 371; 379; 499
677	35	648	106; 187; 198; 286; 391; 479; 490; 571
678	35	648	41; 89; 215; 259; 419; 463; 589; 637
679	35	648	81; 142; 263; 285; 394; 416; 537; 598
681	35	648	79; 103; 119; 250; 431; 562; 578; 602
682	35	648	35; 39; 643; 647
683	35	648	35; 39; 71; 202; 481; 612; 644; 648
684	35	648	35; 37; 71; 73; 107; 109; 127; 143; 145; 179; 181; 191; 215; 217; 251; 253; 265; 287; 289; 307; 325;359; 377; 395; 397; 419; 431; 433; 467; 469; 493; 503; 505; 539; 541; 557; 575; 577; 611; 613; 647;649
685	36	685	37; 648 (**)
689	36	685	146; 242; 269; 420; 447; 543
691	36	685	193; 290; 401; 498
695	36	685	192; 257; 438; 503
697	36	685	111; 132; 270; 427; 565; 586
701	36	685	40; 106; 205; 333; 368; 496; 595; 661
703	36	685	36; 73; 75; 110; 112; 147; 149; 184; 186; 221; 223; 258; 260; 295; 297; 332; 334; 369; 371; 406; 408; 443; 445; 480; 482; 517; 519; 554; 556; 591; 593; 628; 630; 667
707	36	685	82; 187; 276; 319; 388; 431; 520; 625
709	36	685	96; 111; 168; 198; 249; 460; 511; 541; 598; 613
713	36	685	76; 127; 135; 169; 197; 320; 393; 516; 544; 578; 586; 637
715	36	685	277; 302; 413; 438
n_1	d_L	N_{dL}	**Valores de β^1**
719	36	685	41; 107; 168; 228; 491; 551; 612; 678
721	36	685	37; 39; 86; 109; 612; 635; 682; 684
722	36	685	37; 39; 75; 77; 113; 115; 151; 153; 189; 191; 227; 229; 265; 267; 303; 305; 341; 343; 379; 381; 417; 419; 455; 457; 493; 495; 531; 533; 569; 571; 607; 609; 645; 647; 683; 685
730	37	722	173; 557
734	37	722	111; 139; 205; 301; 433; 529; 595; 623
741	37	722	40; 77; 79; 116; 118; 155; 157; 194; 196; 233; 235; 272; 274; 311; 313; 350; 352; 389; 391; 428; 430; 467; 469; 506; 508; 545; 547; 584; 586; 623; 625; 662; 664; 701
742	37	722	205; 333; 409; 537
743	37	722	261; 353; 390; 482

746	37	722	89; 101; 111; 285; 289; 325; 421; 457; 461; 635; 645; 657
747	37	722	113; 119; 628; 634
749	37	722	277; 338;411;472
751	37	722	78; 178; 308; 337; 414; 443; 573; 673
753	37	722	112; 316; 437; 641
754	37	722	43; 71; 99; 223; 263; 491; 531; 655; 683; 711
755	37	722	132; 143; 612; 623
757	37	722	238; 264; 493; 519
758	37	722	37; 41; 73; 135;173; 333; 425; 585; 623; 685; 717; 721
759	37	722	37; 41; 212; 290; 469; 547; 718; 722
760	37	722	37; 39; 41; 79; 81; 113; 119; 121; 159; 161; 199; 201; 239; 241; 267; 279; 281; 319; 321; 341; 343; 359; 401; 417; 419; 439; 441; 479; 481; 493; 519; 521; 559; 561; 599; 601; 639; 641; 647; 679; 681; 719; 721; 723
761	38	761	39; 722 (**)
763	38	761	242; 268; 495; 521
767	38	761	116; 324; 443; 651
769	38	761	82; 347; 422; 687
771	38	761	146; 283; 301; 346; 425; 470; 488; 625
773	38	761	115; 121; 317; 456; 652; 658
777	38	761	271; 367;410; 506
781	38	761	40; 371; 410; 741
783	38	761	106; 205; 229; 275; 508; 554; 578; 677
785	38	761	186; 249; 342; 443; 536; 599
787	38	761	119; 149; 206; 291; 496; 581; 638; 668
789	38	761	76; 82; 187; 218; 308; 356; 433; 481; 571; 602; 707; 713
791	38	761	332; 355; 436; 459
793	38	761	92; 118; 168; 181; 303; 335; 458; 490; 612; 625; 675; 701
795	38	761	139; 302; 308; 326; 469; 487; 493; 656
797	38	761	43; 75; 85; 278; 519; 712; 722; 754
799	38	761	39; 41; 77; 83; 173; 351; 448; 626; 716; 722; 758; 760
800	38	761	39; 41; 119; 121; 279; 281; 359; 361; 439; 441; 519; 521; 679; 681; 759; 761
802	39	800	381; 421
806	39	800	191; 211; 595; 615
808	39	800	153; 169; 639; 655
812	39	800	317; 333; 479; 495
814	39	800	301; 311; 503; 513
818	39	800	77; 85; 143; 675; 733; 741
820	39	800	79; 111; 229; 301; 519; 591; 709; 741
821	39	800	40; 390; 431; 781

823	39	800	195; 261; 287; 391; 432; 536; 562; 628
824	39	800	129; 241; 313; 359; 465; 511; 583; 695
n_1	d_L	N_{dL}	**Valores de β^1**
827	39	800	123; 173; 196; 316; 511; 631; 654; 704
829	39	800	78; 132; 157; 217; 340; 372; 457; 489; 612; 672; 697; 751
830	39	800	99; 109; 159; 261; 569; 671; 721; 731
832	39	800	199; 393; 439; 633
835	39	800	89; 113; 133; 146;319; 326; 509; 516; 689; 702; 722; 746
836	39	800	97; 181; 655; 739
838	39	800	39; 43; 795; 799
839	39	800	39; 43; 79; 308; 531; 760; 796; 800
840	39	800	41; 43; 79; 127; 199; 293; 319; 377; 379; 401; 439; 461; 463; 521; 547; 641; 713; 761; 797; 799
841	40	841	41; 800 (**)
845	40	841	268; 577
851	40	841	82; 178; 349; 384; 467; 502; 673; 769
853	40	841	129; 324; 333; 520; 529; 724
857	40	841	116; 205; 301; 362; 495; 556; 652; 741
859	40	841	44; 410; 449; 815
861	40	841	40; 83; 163; 206; 409; 452; 655; 698; 778; 821
863	40	841	92; 197; 241; 301; 376; 410; 453; 487; 562; 622; 666; 771
865	40	841	253; 612
867	40	841	227; 275; 592; 640
869	40	841	332; 390; 479; 537
871	40	841	82; 228; 340; 393; 478; 531; 643; 789
873	40	841	139; 358; 515; 734
875	40	841	137; 153; 183; 198; 677; 692; 722;738
877	40	841	168;308;569;709
879	40	841	119; 140; 325; 383; 496; 554; 739; 760
881	40	841	41; 43; 121; 154; 233; 246; 387; 494; 635; 648; 727; 760; 838; 840
882	40	841	41; 43; 83; 85; 167; 169; 209; 211; 335; 337; 419; 421; 461; 463; 545; 547; 671; 673; 713; 715; 797; 799; 839; 841
886	41	882	281; 309; 577; 605
890	41	882	233; 657
894	41	882	121; 133; 187; 349; 545; 707; 761; 773
898	41	882	143; 157; 205; 403; 495; 693; 741; 755
902	41	882	381; 393; 509; 521
903	41	882	44;85; 173; 214; 431; 472; 689; 730; 818; 859
905	41	882	287; 432; 473; 618

906	41	882	119; 137; 205; 253; 265; 335; 571; 641; 653; 701; 769; 787
907	41	882	217; 372; 535; 690
909	41	882	238; 317; 592; 671
910	41	882	97; 197; 713; 813
911	41	882	208; 346; 565; 703
913	41	882	191; 239; 674; 722
914	41	882	95; 109; 279; 321; 373; 541; 593; 635; 805; 819
916	41	882	47; 869
917	41	882	146; 358; 559; 771
918	41	882	47; 79; 293; 337; 581; 625; 839; 871
919	41	882	174; 375; 544; 745
921	41	882	121; 137; 161; 389; 532; 760; 784; 800
922	41	882	41; 45; 81; 107; 387; 405; 517; 535; 815; 841; 877; 881
923	41	882	41; 45; 147; 270; 653; 776; 878; 882
924	41	882	41; 43; 89; 125; 127; 211; 221; 293; 295; 353; 377; 379; 439; 485; 545; 547; 571; 629; 631; 703; 713; 797; 799; 835; 881; 883
925	42	925	43; 882 (**)
n_1	d_L	N_{dL}	**Valores de β^1**
929	42	925	178; 324; 381; 548; 605; 751
937	42	925	196; 212; 358; 579; 725; 741
941	42	925	140; 223; 384; 410; 531; 557; 718; 801
943	42	925	51; 301; 448; 495; 642; 892
947	42	925	44; 205; 277; 400; 425; 452; 495; 522; 547; 670; 742; 903
949	42	925	227; 265; 301; 331; 453; 496; 618; 648; 684; 722
953	42	925	82; 129; 228; 362; 372; 430; 523; 581; 591; 725; 824; 871
955	42	925	49; 92; 167; 183; 218; 737; 772; 788; 863; 906
959	42	925	145; 291; 668; 814
961	42	925	52; 132; 143; 153; 168; 182; 201; 208; 365; 596; 753; 760; 779; 793; 808; 818; 829; 909
963	42	925	47; 916
965	42	925	47; 83; 93; 112; 308; 657; 853; 872; 882; 918
967	42	925	43; 45; 85; 91; 127; 154; 270; 355; 434; 444; 523; 533; 612;697; 813; 840; 876; 882; 922; 924
968	42	925	43; 45; 131; 133; 219; 221; 307; 309; 395; 397; 571; 573; 659; 661; 747; 749; 835; 837; 923; 925
970	43	968	463; 507
974	43	968	233; 255; 719; 741
976	43	968	187; 381; 595; 789
980	43	968	53; 927
982	43	968	129; 373; 609; 853

986	43	968	95; 157; 301; 685; 829; 891
988	43	968	85; 147; 289; 699; 841; 903
991	43	968	44; 473; 518; 947
992	43	968	277; 301; 379; 419; 573; 613; 691; 715
993	43	968	260; 317; 676; 733
994	43	968	51; 943
995	43	968	174; 238; 347; 406; 589; 648; 757; 821
997	43	968	57; 131; 137; 191; 261; 736; 806; 860; 866; 940
998	43	968	119; 189; 369; 433; 565; 629; 809; 879
999	43	968	209; 226; 239; 305; 694; 760; 773; 790
1000	43	968	173; 237; 763; 827

(**) – Mancha de erros(*Patch*) compacta coincide com a mancha de erros casada.

Para

n_1 – Número de linhas do código matricial

d_L – Distância de Lee

N_{dL} – Peso da Mancha de Erros

Apêndice III

Códigos Lineares de Bloco

III.1 Introdução

Os códigos de bloco podem ser caracterizados pelo seu processo de codificação, que consiste em segmentar a mensagem a ser transmitida em blocos de k dígitos e acrescentar, a cada bloco, um total de n-k dígitos de redundância. Estes por sua vez, são obtidos a partir dos k dígitos de mensagem e destinam-se à detecção e correção de erros que porventura venham a afetar a transmissão. Estes erros são acarretados pela presença de ruídos durante uma transmissão. Tais erros podem ser esporádicos e independentes, sendo então designados de **erros aleatórios** ou podem ocorrer em **surtos** de vários erros de cada vez e são chamados de **surto de erros**, neste caso diz-se que o canal de comunicação tem memória.

Nos chamados códigos lineares de bloco, os dígitos redundantes são calculados como combinações lineares dos dígitos de informação. Para isto, são empregadas ferramentas matemáticas como a álgebra linear e a teoria de corpos finitos, ou *campos de Galois*.

Um código linear de blocos binário é do tipo (n,k,d) sendo formado por um conjunto de 2^k *n-uplas* binárias, chamadas palavras-código, as quais diferem entre si em pelo menos d posições e formam um subespaço do espaço vetorial de todas as *n-uplas*.

O processo de codificação é mostrado a seguir.

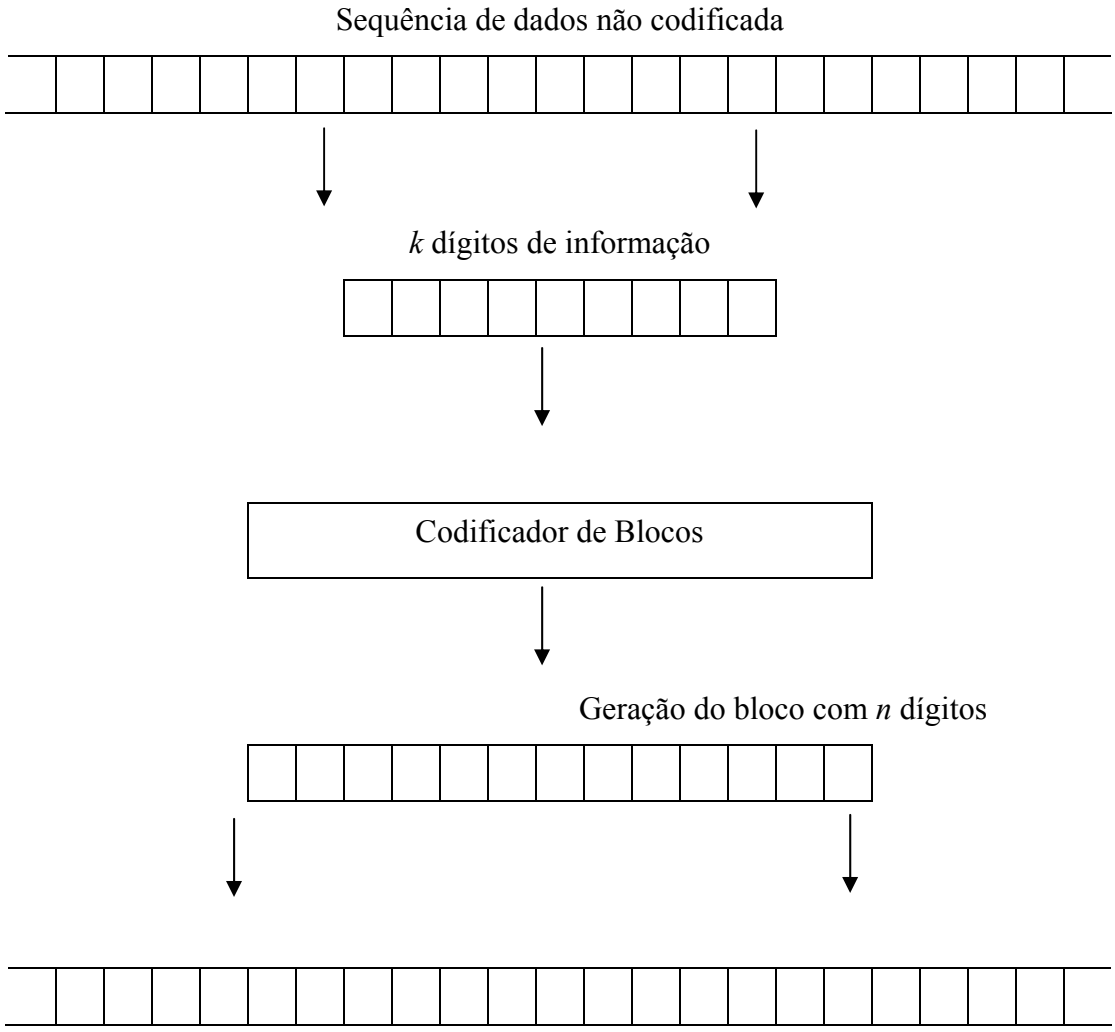


Figura III.1 – Esquema do codificador linear de blocos.

A título de exemplo, com o objetivo de esclarecer os conceitos mostrados, considere-se o código (7,3,4), cujas equações para cálculo dos dígitos redundantes, ou dígitos de verificação de paridade, são as seguintes:

$$c_1 = k_1 \oplus k_2,\ c_2 = k_2 \oplus k_3,$$

$$c_3 = k_1 \oplus k_3 \text{ e } c_4 = k_1 \oplus k_2 \oplus k_3.$$

Definição III.3.3 – Distância mínima de um código é a menor distância de Hamming encontrada entre suas palavras-código.

Em um código que tenha distância *d*, o menor número de mudanças necessárias para converter uma palavra-código em outra é pelo menos *d*. Portanto, a ocorrência, durante a transmissão, de até $d-1$ erros por palavra-código poderá ser detectada, pois o resultado será uma *n-upla* que não pertence ao código. Com respeito à correção de erros, uma vez detectada a presença destes, deve-se decidir por qual palavra-código é a mais próxima da *n-upla* recebida, em termos da distância de Hamming. Esta decisão será sempre correta desde que o padrão de erros a atingir uma palavra-código seja constituído de no máximo *t* erros e que satisfaça a seguinte relação

$$2t + 1 \leq d. \tag{III.3}$$

Definição III.3.4 – A taxa de código ou eficiência do código é dada pela relação entre a dimensão do código, *k,* e o comprimento do código, *n*, ou seja,

$$R = k / n \ , \tag{III.4}$$

em que

R – Corresponde à taxa de código.

III.4 Síndrome de Erros e Decodificação

Supondo-se que uma palavra $\boldsymbol{v_x}$ de um código linear, com matriz geradora *G* e matriz de verificação de paridade *H*, é transmitida por um canal ruidoso. No receptor, devido ao ruído introduzido na transmissão, é recebida uma *n-upla* $\boldsymbol{r_x}$ que difere da palavra-código $\boldsymbol{v_x}$. A tarefa do decoficador é recuperar $\boldsymbol{v_x}$ a partir de $\boldsymbol{r_x}$. O primeiro passo na decodificação consiste em se verificar se $\boldsymbol{r_x}$ é uma palavra-código válida. Isto é feito da seguinte forma

$$\boldsymbol{r_x}.H^T = \boldsymbol{s_x}, \tag{III.5}$$

em que $\boldsymbol{s_x}$ representa um vetor com $n - k$ componentes, chamado de síndrome. Se $\boldsymbol{s_x} = 0$ então não ocorreram erros, ou seja, $\boldsymbol{r_x} = \boldsymbol{v_x}$. No entanto, caso $\boldsymbol{s_x} \neq 0$, $\boldsymbol{r_x}$ não é uma palavra-código e o decodificador irá utilizar esta síndrome para fins de detecção e/ou correção. A *n-upla* recebida $\boldsymbol{r_x}$ pode ser escrita como

$$\boldsymbol{r_x} = \boldsymbol{v_x} + \boldsymbol{e_x} \tag{III.6}$$

em que $\boldsymbol{e_x}$ é uma *n-upla* representando a configuração de erros.

O processo de decodificação envolve uma decisão sobre qual palavra-código foi transmitida. Uma maneira sistemática de implementar a decodificação é distribuir as 2^n *n-uplas* em 2^{n-k} conjuntos distintos, sendo que cada um deles possui apenas uma palavra-código válida. Desta forma a decodificação será feita corretamente se a *n-upla* recebida estiver dentro do subconjunto. A título de exemplo, pode-se colocar as k *n-uplas* válidas encabeçando cada subgrupo, como mostrado a seguir:

0	$\boldsymbol{v_{x1}}$	$\boldsymbol{v_{x2}}$	..... $\boldsymbol{v_{x2^k-1}}$
$\boldsymbol{e_{x1}}$	$\boldsymbol{e_{x1} \oplus v_{x1}}$	$\boldsymbol{e_{x1} \oplus v_{x2}}$	..... $\boldsymbol{e_{x1} \oplus v_{x2^k-1}}$

As linhas subsequentes são formadas de maneira análoga, sendo que cada nova linha começa com um elemento não usado previamente.

As linhas obtidas são chamadas de *classes laterais* e o elemento mais à esquerda em cada classe lateral é chamado de *líder*, e este arrando é chamado de *arranjo padrão*.

III.4.1 Decodificação por máxima verossimilhança

Esta forma de decodificação é baseada na expectativa de que todas as palavras-código (n,k,d) tenham a mesma probabilidade de serem enviadas através de um canal.

Desta forma, ao receber uma *n-upla* $\boldsymbol{r_x}$, o decodificador compara-a com todas as possíveis palavras-código. Exemplificando para o caso binário, isto significa que se compara $\boldsymbol{r_x}$ com as 2^k *n-uplas* distintas que formam o código. A palavra-código que estiver mais próxima de $\boldsymbol{r_x}$, em termos da distância de Hamming, é a selecionada, ou seja, escolhe-se aquela que diferir o menor número de posições em relação à $\boldsymbol{r_x}$.

III.4.2 Decodificação por busca sistemática

Consiste em associar cada síndrome não nula uma configuração corrigível de erros. Uma das propriedades do arranjo padrão é que todas as *n-uplas*, pertencentes a uma mesma classe lateral, têm a mesma síndrome. Baseando nesta propriedade pode-se aplicar o seguinte procedimento de decodificação:

1. Calcular a síndrome para a *n-upla* recebida.
2. Por busca sistemática, achar o padrão de erros corrigíveis, ou seja, o líder da classe lateral associado à síndrome da *n-upla* recebida.
3. Subtrair da *n-upla* recebida a configuração de erros encontrada no passo anterior a fim de corrigí-la.

III.5 Códigos Simples

A seguir são apresentados alguns códigos simples que dão uma noção da construção destes tipos de código.

III.5.1 Códigos de Repetição

É caracterizado pelos seguintes parâmetros: $k = 1$, $c_n \geq 1$ e $n = k + n_c = 1 + n_c$, em que k corresponde à dimensão do código, n corresponde ao comprimento do código e n_c corresponde ao número de dígitos redundantes. Como $k = 1$, este código tem apenas duas palavras, uma delas é uma sequência de n zeros e a outra é uma sequência de n 1's.

III.5.2 Códigos de um único dígito de paridade

Estes códigos têm um único dígito redundante por palavra. Este dígito redundante é calculado de modo a tornar par o número de dígitos 1's na palavra-código. Os parâmetros destes códigos são: $k \geq 1$, $n_c = 1$ e $n = k + n_c = k + 1$.

A distância de Hamming e a eficiência destes códigos são respectivamente $d = 2$ e $R = k / n = k / (k + 1)$.

III.5.3 Códigos de Hamming

Estes códigos são lineares e corrigem um erro por palavra, ou seja, têm distância mínima igual a três. São códigos que têm comprimento $n \leq 2^{n_c} - 1$. Esta condição sobre n assegura a disponibilidade de redundância suficiente para verificar a ocorrência de um erros numa palavra, porque o número de síndromes não nulas, $2^{n_c} - 1$, é sempre igual ou maior ao número de posições onde um erro pode estar.

Um típico exemplo pode ser dado pela construção do código de Hamming (7,4,3), em que $n = 7$, $k = 4$ e $d = 3$, desta forma o número de dígitos redundantes é $n_c = n - k = 7 - 4 = 3$.

A palavra-código é formada da seguinte forma:

Palavra-código

k_1	k_2	k_3	k_4	c_1	c_2	c_3

em que

$$c_1 = k_1 \oplus k_2 \oplus k_3,$$

$$c_2 = k_1 \oplus k_3 \oplus k_4 \text{ e}$$

$$c_3 = k_2 \oplus k_3 \oplus k_4.$$

Cada bloco que contem três dígitos de informação é codificado em uma palavra-código de sete dígitos. Para ilustrar, observe-se os seguintes casos.

Mensagens				Palavras-código						
k_1	k_2	k_3		k_1	k_2	k_3	c_1	c_2	c_3	c_4
0	0	1		0	0	1	0	1	1	1
1	0	1	⟹	1	0	1	1	1	0	0
1	1	1		1	1	1	0	0	0	1

III.2 Representação Matricial

As palavras-código podem ser representadas por vetores com n componentes. Estes componentes são em geral elementos de um corpo finito com q elementos, representado por *GF(q)*, também conhecido por *campo de Galois*. Comumente é empregado o campo binário, cujos elementos são representados por 0 e 1, i.e., fazem parte de *GF(2)*. Um código linear de blocos, como foi citado, constitui um subespaço e qualquer palavra-código pode ser representada por uma combinação linear de vetores da base do subespaço. Os vetores da base contituem as linhas de uma matriz, chamada matriz geradora do código [32]. Esta matriz é formada de k linhas e n colunas e para o exemplo mostrado anteriormente ela pode ser representada como

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

A partir desta matriz geradora G pode-se formar uma outra matriz H, com n-k linhas e n colunas, tal que o espaço das linhas de G seja ortogonas a H, isto é, se $\boldsymbol{v}_x$ é um vetor no espaço das linhas de G então

$$\boldsymbol{v}_x . H^T = 0. \qquad \text{(III.1)}$$

Esta matriz H é chamada de matriz de verificação de paridade do código, e pode ser representada como

$$H = [h_H : I_{n-k}], \tag{III.2}$$

em que h_H é uma matriz $(n\text{-}k)$ x k e I_{n-k} é uma matriz unitária $(n-k)$x$(n\text{-}k)$.

A matriz geradora G forma um código (n,k,d), em que n representa o comprimento da palavra-código, k é a dimensão e d representa a distância mínima de Hamming [32].

A título de ilustração a matriz de verificação de paridade, H, para o exemplo anterior é a seguinte

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

III.3 Distância Mínima e Taxa do Código

A capacidade de detecção e/ou correção de erros por um código linear de blocos está ligada a um parâmetro designado por distância mínima. Antes porém de definir este parâmetro é imperioso que se façam algumas definições preliminares.

Definição III.3.1 – Peso de Hamming corresponde ao número de posições não nulas de um vetor.

Definição III.3.2 – Distância de Hamming entre dois vetores, de mesmo número de componentes, corresponde ao número de posições em que estes vetores diferem.

Exemplo III.3.1- Considerando os vetores $\boldsymbol{v}_{x1}$ = (0, 1, 0, 0, 2) e $\boldsymbol{v}_{x2}$ = (1, 1, 0, 3, 2), os pesos de Hamming destes dois vetores são 2 e 4 , respectivamente. A distância de Hamming entre estes vetores é igual a 2.

Palavra-código

k_1	k_2	k_3	k_4	c_1	c_2	c_3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	0	1	0
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	1

Apêndice IV

Lista de Símbolos

d	Distância de Hamming
(n,k,d)	Indica o formato de um código linear de blocos com comprimento n, dimensão k e distância de Hamming d
b	Comprimento de um surto de erros
(n,k,b)	Indica o formato de um código linear de blocos com comprimento n, dimensão k e capacidade de correção de surto de erros de comprimento b
t	Indica a capacidade de correção de erros aleatórios ou em surto em Códigos corretores de erros aleatórios ou Códigos corretores de erros em surto, respectivamente
G_H	Matriz geradora de códigos de único dígito de paridade com palavras dispostas na direção horizontal
G_V	Matriz geradora de códigos de único dígito de paridade com palavras dispostas na direção vertical
(n_1n_2,k_1k_2,d)	Formato de código matricial de blocos para correção de erros aleatórios, com n_1 linhas e n_2 colunas, com dimensão vertical k_1 e horizontal k_2 e distância de Hamming d
R	Taxa do código corretor de erros que é expressa pela relação k / n.
(n_1n_2, k_1k_2, b)	Formato de código matricial de blocos para correção de surtos de erros, com n_1 linhas e n_2 colunas, com dimensão vertical k_1 e horizontal k_2 e comprimento do surto de erros b
(i,j)	Indica a posição de um dígito dentro de uma palavra-código de um código matricial, em que i é o número da linha e j é o número da coluna.
$f(i,j)$	Função que calcula, em algarismo decimal, a posição correspondente ao dígito pertencente a um código matricial
d_m	Representa a distância entre as posições (i_1,j_1) e (i_2,j_2) de um código matricial
S_h	Síndrome Horizontal
S_v	Síndrome Vertical
S_d	Síndrome Diagonal
(n_1n_2, k_1k_2, b, s)	Formato de código matricial de blocos para correção de surtos de erros, com n_1 linhas e n_2 colunas, com dimensão vertical k_1 e horizontal k_2, comprimento do surto de erros b e s indicando o intervalo de leitura entre diagonais sucessivas
b_ixb_j	Corresponde a uma mancha de erros de dimensão b_i x b_j, em que b_i corresponde à profundidade vertical e b_j à profundidade horizontal
ε	Número de erros presentes em uma mancha de erros de dimensão b_i x b_j

ρ^{q}	Indica a rotação de um vetor qualquer em q posições para a direita
ρ^{-q}	Indica a rotação de um vetor qualquer em q posições para a esquerda
$<u>_{v}$	Representa um único inteiro κ, $0 \leq \kappa \leq v$-1, tal que $\kappa \equiv u(\text{mod } v)$
$n_1 x n_2$	Representa uma estrutura matricial ou de blocos com n_1 linhas e n_2 colunas
$(n_1\text{-}1)x(n_2\text{-}1)$	Representa uma estrutura matricial com $(n_1\text{-}1)$ linhas e $(n_2\text{-}1)$ colunas
A	Arranjo matricial de dimensão $n_1 x n_2$
A_u	Arranjo matricial de dimensão $n_1 x n_2$ com entrelaçamento unidimensional
A_{-1}	Arranjo matricial de dimensão $(n_1\text{-}1)x(n_2\text{-}1)$
A_r	Arranjo matricial de dimensão $n_1 x n_2$ com entrelaçamento feito pela rotação das linhas
A_b	Arranjo matricial de dimensão $n_1 x n_2$ com entrelaçamento nas linhas e colunas
r_i	Especifica o número da linha em um código matricial
c_j	Especifica o número da coluna em um código matricial
$x_1, x_2, x_3, ...$	Representa uma seqüência de dígitos
E	Indica a capacidade de estocagem de entrelaçamento
E_u	Indica a capacidade de estocagem de desentrelaçamento
D	Representa uma linha de retardo de D símbolos
m	Corresponde ao número de linhas de retardo em um ramo do entrelaçador convolucional
(b,n)	Formato de um entrelaçador com palavras-código de comprimento n e capacidade de correção de surtos de comprimento b
(n,b)	Formato de um desentrelaçador para o entrelaçador do tipo (b,n)
z_i	Corresponde à posição de um dígito na seqüência original de dados
z'_i	Corresponde à posição de um dígito na seqüência original de dados restaurada
D'	Retardo fixo introduzido pelo par entrelaçador-desentrelaçador
$min()$	Função que gera o valor mínimo entre duas ou mais variáveis
$max()$	Função que gera o valor máximo entre duas ou mais variáveis
a	Máximo retardo a que é submetido um dígito dentro do entrelaçador
SRx	Registrador deslocamento de número x.
l_p	Profundidade de entrelaçamento
L	Comprimento do bloco de dígitos no esquema de Forney
C	Código linear de blocos
Q	Quantidade de palavras-código separada de um código C para a formação de um subconjunto matricial Z
Z	Matriz de dimensão Q x Q formada por Q palavras-código de um código linear de blocos C
B_w	Mancha de erros b_i x b_j com $\varepsilon = w \leq Q$ erros
Z_q	Submatriz de Z com dimensão r x s, com $r = s$
Z_r	Submatriz de Z com dimensão r x s, com $r \neq s$
T	Matriz de Transformação Linear
T^{-1}	Matriz de Transformação Linear Inversa
(i',j')	Posição dos dígitos em uma matriz entrelaçada

Δ	Determinante da matriz T
B	Corresponde às coordenadas dos dígitos, (i,j), em um formato matricial
B'	Corresponde às coordenadas transladas dos dígitos, (i',j'), em um formato matricial
G_e	Ganho de entrelaçamento
t_{com}	Capacidade de correção de erros de um código matricial empregando entrelaçamento
t_{sem}	Capacidade de correção de erros de um código matricial sem entrelaçamento
N	Número de erros, em uma matriz QxQ, espalhados pelo processo de entrelaçamento bidimensional
(x_i,y_i)	Coordenadas dos dígitos em um código matricial, em que x_i indica posição do dígito no sentido horizontal (na linha) e y_i no sentido vertical (na coluna).
(x'_i,y'_i)	O equivalente a (x_i,y_i) para uma palavra-código de um código matricial que foi submetida ao processo de entrelaçamento unidimensional
β	Valor inteiro positivo relativamente primo a n_1 que faz parte da matriz de transformação linear T que executa entrelaçamento unidimensional
β^{-1}	Parãmetro que faz parte da matriz de transformação linear inversa T^{-1} que perfaz o desentrelaçamento
d_L	Distância de Lee
I_2	Matriz unitária de ordem 2
K	Inverso aditivo de β^{-1} módulo n_1
N_{d_L}	Peso máximo de uma *mancha-*d_L

θ	Período de repetição do padrão do entrelaçamento unidimensional
ϕ ()	Função de Euler que determina a quantidade de números relativamente primos e menores ao argumento da função
G	Matriz geradora de um código *q-ário C*
I_k	Matriz identidade de dimensão k x k que compõe a matriz G
g	Matriz de dimensão k x $(n-k)$ que compõe a matriz G
S	Matriz de embaralhamento, densa e não-singular
S^{-1}	Matriz de embaralhamento inversa
μ	Bloco de texto claro com dimensão k
μ_i	Bloco de texto claro com dimensão k com dígito "1" apenas na posição i
G	Matriz geradora de um código *q-ário C*
I_k	Matriz identidade de dimensão k x k que compõe a matriz G
g	Matriz de dimensão k x $(n-k)$ que compõe a matriz G
S	Matriz de embaralhamento, densa e não-singular
S^{-1}	Matriz de embaralhamento inversa
μ	Bloco de texto claro com dimensão k
μ_i	Bloco de texto claro com dimensão k com dígito "1" apenas na posição i
η	Palavra-código do código $C\ (n,k,d)$ gerada pelo produto μSG
e_r	Código de erro de comprimento n
r_c	Palavra-código gerada pela adição de η com e_r
y_c	Bloco de texto cifrado com dimensão n obtido da permutação das coordenadas de r_c
P	Matriz de permutação
P^{-1}	Matriz de permutação inversa
$E_{l,w}$	Representa um surto aleatório de erros de comprimento l e peso w
G'	Matriz geradora de um código *q-ário C* permutada por P
g'_i	Corresponde a i-ésima coluna da matriz G'
$E'_{l,w}$	Representa um surto aleatório de erros de comprimento l e peso w permutado por P
N_B	Número de códigos corretores de erros que são combinatoriamente equivalentes para um dado conjunto de parâmetros b, n, d e k.
(μ, y_c)	Par texto claro/ texto cifrado
T_r	Fator de trabalho do criptoanalista
$P_r(\ )$	Cálculo de probabilidade
$N_e\ (w)$	Número de vetores de erro de peso w
$Nt_e(w_{min})$	Total de vetores de erro para um dado w_{min}
$O\ (\ )$	Número de operações necessárias para resolução de uma criptoanálise
H	Matriz de verificação de paridade
p	Número primo qualquer
M'	Indica a quantidade de mensagens de texto claro toda nula, usada no esquema de Al Jabri
M	Indica a quantidade de mensagens de texto cifrado separada do total de M' com b posições não nulas, usada no esquema de Al Jabri

i_b	Indica as posições de dígitos "1" no *b*-texto cifrado
y_{lm}	Corresponde a um *lm-esimo b*-texto cifrado obtido no esquema de Al Jabri, em que $1 \leq lm \leq M$
S_{lm}	Conjunto contendo informações das posições não nulas de um *b*-texto cifrado y_{lm}
Q_{ib}	Conjunto obtido a partir dos conjuntos S_{lm} contendo o elemento i_b
P_{ib}	Conjunto obtido a partir do ordenamento dos conjuntos Q_{ib}
P_{ψ}	Indica uma ψ*-ésima* permutação, em que $1 \leq \psi \leq n$, usada no esquema de permutação adaptativa
GF(q)	Campo de Galois *q-ario*
p(x)	Polinômio primitivo
α	Raiz primitiva de um polinômio primitivo *p(x)*
c(x)	Corresponde a uma palavra-código
g(x)	Polinômio gerador
m(x)	Sequência de texto claro ou mensagem/dados a ser codificada
RS( )	Código Reed-Solomon
BCH ( )	Código BCH
B_i	Indica a posição dos dígitos em cada coluna no esquema de Patel-Hong, em que $0 \leq i \leq 7$
Z_j	Denota o número da linha no esquema de Patel-Hong, em que $0 \leq j \leq 8$
S_x	Síndrome de número *x*
m_i	Ordem multiplicativa de *q* módulo *n* de um campo de Galois *q-ário* $GF(q^{m_i})$
δ	Distância projetada para o código *BCH*
C_r	Classe ciclotômica de ordem *r*
$M_r(x)$	Polinômio mínimo
MMC()	Função que gera o mínimo múltiplo comum entre os parâmetros relacionados no argumento da função
h_H	Corresponde a uma matriz (*n-k*) x *k* que compõe a matriz de paridade *H*
I_{n-k}	Corresponde a uma matriz unitária (*n-k*)x(*n-k*) que compõe a matriz de paridade *H*
MDC[]	Função que gera o máximo divisor comum entre os parâmetros relacionados no argumento da função
c_i	Corresponde ao *i-esimo* dígito de paridade de uma palavra-código pertencente a um código linear
k_i	Corresponde ao *i-esimo* dígito de informação de uma palavra-código pertencente a um código linear
$\boldsymbol{v_x}$	Vetor no espaço formado pelas linhas de *G*
H^T	Matriz de verificação de paridade transposta
$\boldsymbol{r_x}$	Corresponde a uma *n-upla* recebida que difere da palavra-código $\boldsymbol{v_x}$
$\boldsymbol{s_x}$	Síndrome vetorial de $n - k$ elementos
$\boldsymbol{e_x}$	Corresponde a uma *n-upla* vetorial representando o erro introduzido
n_c	Número de dígitos redundantes em um código linear

Bibliografia

1. Farrell, P.G. - *An Introduction to Array Error Control Codes; Geometries, Codes and Cryptography* Ed. G. Longo, M. Marchi, A. Sgasso; CISM Courses and Lectures, Springer-Verlag, pp. 101-128, 1990.

2. Zhang, W. and Wolf, J. K. – *A Class of Binary Burst Error-Correcting Quasi-Cyclic Codes*; IEEE Trans. on Info. Theory, vol. 34, no. 3, pp. 463-479, May 1988.

3. Blaum, M., Farrell, P.G. and Tilborg, H.C.A.V – *A Class of Burst Error-Correcting Array Codes*; IEEE Trans. on Info. Theory, vol. 32, no 6, pp. 836-839, Nov. 1986.

4. Blaum, M. – *A Family of Efficient Burst-Correcting Array Codes*; IBM Res. Report, March 1989.

5. Blaum, M., Farrell, P.G and Tilborg H.C.A.V. – *Multiple Burst-Correcting Array Codes*; IEEE Trans. on Info. Theory, vol. IT-34, no. 5, pp. 1061-1068, Sept. 1988.

6. Blaum, M. and Roth, R.M. – *New Array Codes for Multiple Phased Burst Correction*; IEEE Trans. on Info. Theory, vol. 39, no. 1, pp. 66-77, Jan. 1993.

7. Peng, X. H. and Farrell, P. G. – *Bounds on Parity Checks Based Array Codes*; Communication Res. Group, University of Manchester, UK, pp. 193.

8. Sayano, M., Goodman, R.M. and McEliece, R.J. – *Single Phased Burst Error Correcting Array Codes*; Department of Elect. Engr., California Institute of Technology, pp. 195.

9. Blaum, M. and Farrell, P.G – *Array Codes for Cluster-Error Correction*; Electronics Letters, Vol. 30, No. 21, pp. 1752-1753, Oct. 1994.

10. Breitbach, M., Bossert, M., Zyablov, V. and Sidorenko, V. – *Array Codes Correcting a Two Dimensional Cluster of Errors*; IEEE Trans. on Info. Theory, vol. 44, no. 5, pp. 2025-2031, Sept. 1998.

11. Haslach, C. and Vinck, A.J.H. – *A Decoding Algorithm with Restrictions for Array Codes*; IEEE Trans. on Info. Theory, vol. 45, no. 7, pp. 2339-2344, Nov. 1999.

12. Farrell, P.G. and Mabogunje, A.O. – *Burst Error Correction Capability of Square Array Codes*; IEE Electronics Letters, Vol. 27, No. 13, pp. 1215-1216, June 1991.

13. Bruck, J. and Blaum, M. – *Array Codes for Correction of Criss-Cross Errors*; ISIT, Germany, July 1997.

14. Sayano, Masahiro, Goodman, R.M. and McEliece, R.J.- *Phased Burst Error Correcting Array Codes*; IEEE Trans. on Information Theory, Vol. 39, No. 2, pp. 684-693, March 1993.

15. Gilbert, E.N. – *A Problem in Binary Encoding*; Proc. Symp. Applied Maths., Vol. 10, Ann. Math. Soc., pp. 291-297, 1960.

16. Farrell, P.G. and Hopkins, S.J. – *Burst-Error-Correcting Array Codes*; Radio and Electronic Engr., Vol. 52, No 4, pp. 182-192, April 1982

17. Farrell, P.G. and Hopkins, S.J. – *Decoding Algorithms for a Class of Burst-Error-Correcting Array Codes*; Proc. IEEE Int. Symp. on Info. Theory, Les Arcs, France, June 21-25, 1982.

18. Campello de Souza, R. – *Single-Burst Correcting Array Codes*; Comms. Res. Group Report, University of Manchester, 1984.

19. Daniel, J.S. – *Double-Burst-Correcting Array Codes: Generation and Decoding*; Proc. Proc. IEEE Int. Symp. on Theory, Brighton, UK, June 1985.

20. Daniel, J.S. – *Synthesis and Decoding of Array Error Control Codes*; Ph.D. Thesis, University of Manchester, UK, August 1985.

21. Bridwell, J.D. and Wolf, J.K. – *Burst Distance and Multiple-Burst Correction*; BSTJ, Vol. 49, pp. 889-909, May-June 1970.

22. Slepian, D. – *Some Further Theory of Group Codes*; BSTJ, Vol. 39, pp. 1219-1252, Sept. 1960.

23. Rocha, V.C., Jr., Farrell, P. and Guimarães,W.P.S. – *Two-dimensional Interleaving with Burst Error-Correcting Codes*; IEE Electronics Letters, Vol. 38, No. 18, pp. 1042-1043, 29th August 2002.

24. Haslach, C. Vinck, A.J.H. – *Efficient Decoding of Interleaved Linear Block Codes*; Proc. IEEE Int. Symp. on Info.Theory, June 25-30, 2000, p.149.

25. Kauffman, A.N., Moraes, M.A.C, Lima, R.C.C and Campello de Souza, R.M. – *A Technique for Correcting Clusters of Errors*; International Telecomunications Symposium, Acapulco, Mexico, 1998, pp. 64-66.

26. Almeida, C., Palazzo, R. – *Efficient Two-dimensional Interleaving Technique by Use of the Set Partioning Concept*; IEE Electronics Letters, vol. 32, no 6, pp. 538 –540, 14th March 1996.

27. *Interleaving; Prepared for the Department of Communication under Contract* No. 36001-8-3529/01-SS, Jan. 1990 (comunicação pessoal do Prof. V. C. da Rocha Jr.).

28. Blaum, M. and Bruck, J. – *Correcting of Two-Dimensional Clusters by Interleaving of Symbols*; Proc. IEEE Int. Symp. on Info. Theory, June 1994.

29. Ramsey, J. L. – *Realization of Optimum Interleavers*; IEEE Trans. on Info. Theory, vol. 16, 1970, pp. 338-345.

30. Bruck, J. and Blaum, M. – *Two-dimensional Interleaving Schemes with Repetitions*; Sloan Research, 1995.

31. Bruck, J. and Blaum, M. – *Interleaving Schemes for Multidimensional Cluster Errors*; IBM Research Report, RJ 9984, Oct. 1995.

32. Wicker, S.B. – *Error Control Systems for Digital Communication and Storage*; Prentice Hall Inc., 1995.

33. Forney, G.D. – *Burst Correcting Codes for the Classical Bursty Channel*; IEEE Trans. on Comm. Technology, vol.19, 1971, pp. 772-781.

34. Forney, G.D. – Interleavers; U.S. Patent No. 3652998, March 28, 1972..

35. Rocha, V. C., Jr. – *Two-Dimensional Interleaving*, Communications and Coding; Editors M. Darnell and B. Honary, Research Press Studies, John Wiley & Sons, pp. 82-88, 1998, England.

36. Peterson, W.W and Weldon, E.J. – *Error Correcting Codes*; MIT Press, Second Edition, 1972.

37. McEliece, R.J. – *A Public Cryptosystem Based on Algebraic Coding Theory*; DSN Progress Report 42-44, pp. 114-116, Jet Propulsion Laboratory, CA, January and February 1978.

38. Rao, T.R.N. and Nam, K. – *Private-key Algebraic Code Encryptions*; IEEE Trans. on Info. Theory, vol.35, no.4, 1989, pp. 829-833.

39. Berklamp, E.R. – *Goppa Codes*; IEEE Trans. on Info. Theory, vol.19, no.5, 1973 pp. 590-593.

40. Alencar, F.M.R., Léo, A.M.P. and Campello de Souza, R.M. - *Private-key Burst Correcting Code Encryption*; Proc. IEEE Int. Symp. on Info. Theory, January 1993, p.227.

41. Campello de Souza, R.M. and Campello de Souza, J. - *Array Codes for Private-key Encryption*, Electronics Letters, vol.30, no.17, 1994, pp. 1394-1396

42. Hin, P.J.M. - *Channel-Error-Correcting Privacy Cryptosystems*; Ph.D. Thesis, Delft University of Technology, 1986.

43. Struik R. and Tilburg, J.V. - *The Rao-Nam Scheme is Insecure Against a Chosen-plaintext Attack*; Lecture Notes in Computer Science 293; Advances in Cryptology: Proceedings of Crypto'87, C. Pomerance, Ed. Santa Barbara, CA, Aug. 16-20, Berlin: Springer-Verlag, 1988, pp. 445-457.

44. Rocha, V.C., Jr. and Blaum, M. - *A Secret-key Cryptosystem Based on Phased Burst Correcting Codes*; 2nd International Symposium on Communication Theory and Applications, Ambleside, Lake District, UK, 11-16 July 1993, pp. 136-142.

45. Blaum, M. and Roth, R. - *New Array Codes for Multiple Phased Burst Correction*; IEEE Transactions Information Theory, IT-39 (1993), pp. 66-77.

46. Shiloach, Y. - *A Fast Equivalence-checking Algorithm for Circular Lists*, Inform. Proc. Lett., 8 (1979), 236 –238.

47. Blaum, M. - *A Coding Technique for Recovery Against Double Disk Failures in Disk Arrays*; IEEE Int. Conf. on Communications, 1992, pp.1366-1368.

48. Rocha, V.C., Jr. - *Coding for Privacy with Burst Adaptive Permutations*; IEEE Globecom, Rio de Janeiro, 05-09 Dec. 1999, Proceedings pp. 2432-2435.

49. Al Jabri, A. - *Security of Private-key Encryption Based on Array Codes*; IEE Electronics Letters, Vol. 32, No. 24, pp. 2226-2227, 21st November 1996.

50. Al Jabri, A. - *On Security of Cryptosystems Based on Phased Burst Array Codes*, 4th International Symposium on Communication Theory and Applications, Ambleside, Lake District, UK, 13-18 July 1997, pp. 197-200.

51. Lin, S. and Costello, D.J. - *Error Control Coding: Fundamentals and Applications*; Englewoods Cliffs, NJ: Prentice-Hall, 1983, sec. 16.2

52. Patel, A.M. and Hong, S.J. - *Optimal Rectangular Code for High Density Magnetic Tapes* - IBM J. Res. Dev., Vol 18, pp. 579-588, Nov. 1974.

53. Blaum, M. and McEliece, R.J. - *Coding Protection for Magnetic Tapes: A Generalization of the Patel-Hong Code*; IEEE Transactions on Information Theory, IT-31, No. 5, pp. 690-693, Sept. 1985.

54. Calingaert, P. - *Two Dimensional Parity Checking*; Jour. Assoc. Comp. Machinery, Vol. 8, No. 2, pp. 186-200, 1961.

55. Rocha, V.C., Jr. – *Efficient Burst-Correcting Cyclic Codes*; Electronics Letters, vol.19, No.2, pp. 44-45, 1983.

56. Reiger, S.H. – *Codes for the Correction of Cluster Errors*; IRE Trans. Inform. Theory, IT-16, pp. 16-21, 1960.