

Relatório 3 - ELE-32

Códigos Convolucionais

Eduardo Alarcon Mady Barbosa
Departamento de
Engenharia da Computação
Instituto Tecnológico de Aeronáutica
São José dos Campos, São Paulo
Email: eduambarbosa@gmail.com

Vitor Pimenta dos Reis Arruda
Departamento de
Engenharia da Computação
Instituto Tecnológico de Aeronáutica
São José dos Campos, São Paulo
Email: vitor_pimenta97@hotmail.com

Resumo—Este relatório analisa o desempenho de códigos convolucionais por meio da implementação de um codificador de canal convolucional e seu respectivo decodificador. Foram realizados três métodos de decodificação tendo como base o algoritmo de Viterbi e variando o critério de decisão.

Para tanto, implementaram-se também um canal BSC e um canal AWGN. A taxa de erro de bit de informação foi avaliada estatisticamente para cada um dos conjuntos e para três polinômios geradores distintos, submetendo-os à passagem de um milhão de bits de informação para estimar seus desempenhos.

I. INTRODUÇÃO

A. Códigos Convolucionais

Um código convolucional é um tipo de código corretor de erro que processa uma sequência de bits. Ele é composto por um codificador convolucional e um decodificador.

B. Codificador Convolucional

Um codificador convolucional pode ser visto como uma máquina que a cada instante processa k bits de entrada e gera $n > k$ bits de saída e que possui memória, de forma que os bits a serem gerados no próximo instante dependem do estado (memória) atual do codificador. Uma forma de descrever o código do codificador convolucional é através da descrição polinomial, a qual é descrita com mais detalhes em [2] e [3]. A matriz geradora $G(D)$ é a matriz formada pelos polinômios geradores, em que cada um deles é responsável pela geração de um bit codificado. A relação entre entrada e saída é dada pelo produto do polinômio de entrada $u(D)$ pela matriz geradora:

$$V(D) = u(D)G(D) \quad (1)$$

A saída unidimensional é formada pela interpolação dos termos das saídas de $V(D)$:

$$v(D) = \sum_n \sum_{i=1} v_i(D^n) D^{i-1} \quad (2)$$

Outra descrição possível é através do diagrama de estados com um codificador recursivo, que não será abordada no relatório.

C. Decodificação

Para a decodificação, utiliza-se o algoritmo de Viterbi. O algoritmo de Viterbi usa a treliça do código para realizar a decodificação de um código convolucional. O algoritmo armazena métricas de percurso (custo) para tomar a decisão sobre qual sequência foi transmitida. A métrica do percurso é obtida através da soma das métricas dos ramos que, concatenados, formam o percurso.

Para os métodos implementados utilizou-se três métricas:

- Realizando a transmissão através de um canal BSC, utilizou-se a distância de Hamming entre o que foi recebido e os rótulos binários de saída dos ramos. A distância de Hamming entre os dois vetores binários com o mesmo comprimento é o número de posições em que estes dois vetores diferem de valor.

- Realizando a transmissão através de um canal BSC, utilizou-se a probabilidade de cada ramo ter sido transmitido dado o valor recebido naquele instante.

- Realizando a transmissão através de um canal AWGN, utilizou-se a distância Euclidiana quadrática entre o símbolo recebido e os símbolos associados aos ramos de saída.

A cada instante, o número de percursos possíveis cresce exponencialmente. Se, por exemplo, o codificador tiver k entradas, n saídas e m memórias, há 2^{kt} sequências possíveis do instante 0 até o instante t . Por outro lado, somente um pequeno conjunto de sequências são as mais prováveis. O algoritmo de Viterbi aproveita-se deste fato para reduzir a complexidade de decodificação. Em um dado instante, há vários percursos que levam ao mesmo estado σ_i^t . Dentre os vários possíveis, um deles será mais provável do que os outros. Logo, se o percurso mais provável passa pelo estado σ_i no instante t , o caminho que levou até este estado deve ser o mais provável. Assim, em cada instante, é necessário armazenar, pra cada estado, somente o caminho mais provável de chegar até ele. Isto é, precisamos armazenar $2m$ caminhos, o que é uma redução de complexidade se comparado com o número de sequências possíveis (2^{kt}).

II. IMPLEMENTAÇÃO

Todas as implementações foram feitas utilizando a linguagem de programação Julia, projetada para atender os requisitos

da computação de alto desempenho numérico e científico.

A. Codificador Convolutacional

A implementação do codificador convolutacional foi feita com base na interpretação da máquina de estados descrita anteriormente. A partir do polinômio gerador o código recebe quais memórias, e também o bit de entrada, fazem parte do bit de saída. O funcionamento da máquina de estados se dá a partir de *shifts* nas memórias, em que as memórias representam o estado atual, retornando os bits de saída e recebendo um novo bit de entrada.

B. Canais BSC e AWGN

A implementação dos canais consiste em simular a transmissão dos bits já codificados, ou seja, inserir ruídos. Para o canal BSC, o código recebe os bits já codificados e para cada bit gera um número aleatório entre 0 e 1. Para uma probabilidade de erro de bit p , muda o bit quando o número gerado for menor que p .

Para o canal AWGN, primeiro os bits codificados são remapeados para uma modulação BPSK (os bits 0 são substituídos por bits -1). Em seguida, é somado em cada bit um ruído obtido ao gerar um número aleatório tendo como distribuição uma distribuição normal com média 0 e variância $\sqrt{N_0/2}$.

C. Decodificador

A implementação do decodificador foi feita utilizando uma árvore como estrutura de dados para a realização do algoritmo de Viterbi:

1. Inicie todos os estados com custo igual a zero. Caso o estado inicial seja obrigatoriamente um estado em particular (por exemplo estado nulo), somente este terá custo zero; os outros terão custo inicial infinito.
2. Dado o símbolo recebido, calcule o custo de cada uma das transições possíveis no instante atual.
3. Para cada estado futuro, calcule o custo de todos os percursos que chegam ao estado somando o custo do estado anterior com o custo do ramo que causa a transição entre estados. O caminho sobrevivente para um estado futuro é aquele com menor custo. Este custo torna-se também o custo do estado.
4. Enquanto houver símbolos a serem processados, retorne ao passo 2.
5. Escolha o estado final que tem o menor custo e o caminho que leva a ele, obedecendo a restrições sobre o estado final, se houver.

III. RESULTADOS OBTIDOS

Cada um dos métodos de implementação foram testados com três quantidades de memória distintas, cada qual com três polinômios geradores. A quantidade de memória de cada máquina de estados e os respectivos polinômios geradores estão representados na Tabela I.

Tabela I: Representação octal dos polinômios geradores a serem utilizados para gerar os códigos com taxa 1/3

m	$g_1(D)$	$g_2(D)$	$g_3(D)$
3	13	15	17
4	25	33	37
6	117	127	155

A. Máquina de estados com 3 memórias

O resultado obtido para a máquina de estados com 3 memórias está representado na Figura 1.

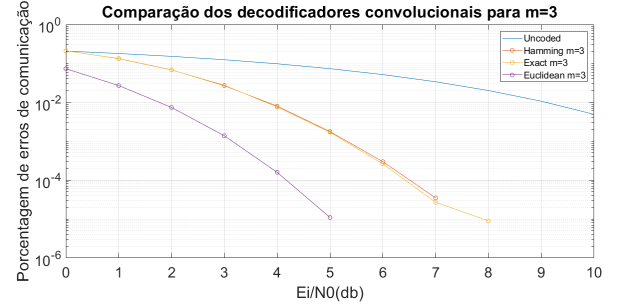


Figura 1: Gráfico da porcentagem de erro pela razão sinal ruído normalizada para $m=3$

B. Máquina de estados com 4 memórias

O resultado obtido para a máquina de estados com 4 memórias está representado na Figura 2.

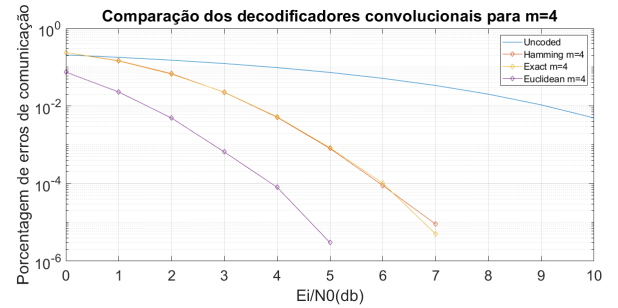


Figura 2: Gráfico da porcentagem de erro pela razão sinal ruído normalizada para $m=4$

C. Máquina de estados com 6 memórias

O resultado obtido para a máquina de estados com 6 memórias está representado na Figura 3.

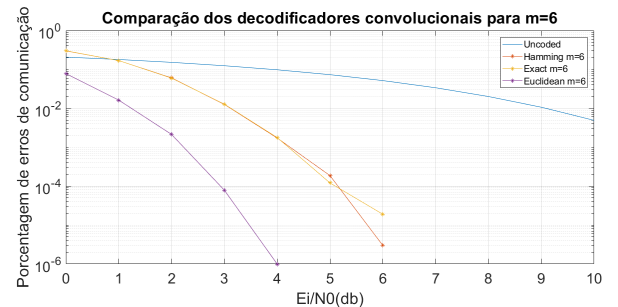


Figura 3: Gráfico da porcentagem de erro pela razão sinal ruído normalizada para $m=6$

IV. ANÁLISE

A. Decodificadores convolucionais

Analisando a Figura 1, a Figura 2 e a Figura 3, pode-se concluir que o decodificador é melhor utilizando o método da distância euclidiana como custo do algoritmo de Viterbi. Já em relação ao custo pela distância de Hamming e pela probabilidade percebe-se um comportamento já esperado: como a distância de Hamming é uma aproximação do método da probabilidade, que é exato, o último é levemente superior ao primeiro, mas ambos são piores que o método da distância euclidiana.

Para fazer uma comparação do desempenho dos decodificadores entre as diferentes quantidades de memórias das máquinas de estados, plota-se um gráfico composto pela Figura 1, Figura 2 e Figura 3, representado na Figura 4.

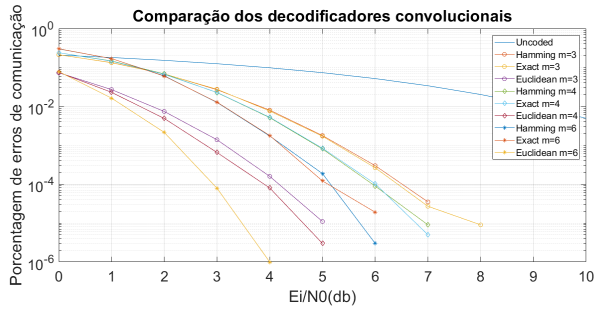


Figura 4: Gráfico da porcentagem de erro pela razão sinal ruído normalizada para todos os valores de m

Analisando a Figura 4, percebe-se que quanto maior a quantidade de memórias da máquina de estados, melhor é o desempenho do decodificador. A melhora acontece para as três métricas de cálculo de custo.

B. Comparação com vários decodificadores

Utilizando os códigos desenvolvidos em relatórios anteriores, comparou-se na Figura 5 o desempenho dos códigos de convolução com os de bloco e cíclicos. Para os códigos de bloco, foram testados o código de Hamming e o código desenvolvido pelo grupo. Já para os códigos cíclicos, foram testados um código BCH e apenas os que possuíam distância mínima de Hamming suficiente para corrigir um bit: (7, 12) e (8, 14), em que o par ordenado é composto pelos bits de informação e pelo tamanho de palavra código.

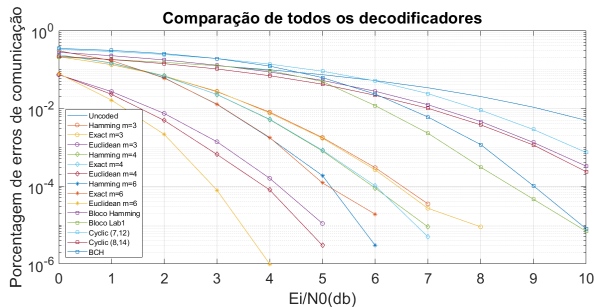


Figura 5: Gráfico da porcentagem de erro pela razão sinal ruído normalizada para todos os decodificadores

Analisando a Figura 5, percebe-se que os códigos que obtêm melhor desempenho são os códigos convolucionais. Em seguida, observa-se que os códigos cíclicos possuem um desempenho, em média, melhor que os códigos de bloco, mas todos piores que os códigos convolucionais implementados.

V. RESPOSTAS ÀS PERGUNTAS

A. Perguntas presentes em [3]

1) *Quais foram as maiores dificuldades em implementar o codificador convolucional?*: Não houve dificuldade. Modelando o codificador como uma máquina de estado, a implementação foi direta.

2) *Quanto tempo a sua solução demora para codificar cada bit de informação? Faça uma média.*: Analisando cada um dos 3 códigos convolucionais pedidos no roteiro, observou-se que o primeiro demora 330ns para codificar cada bit de informação; o segundo, 365ns; o terceiro, 443ns.

3) *Quais foram as maiores dificuldades em implementar o decodificador convolucional?*: Dado que a decodificação tem complexidade exponencial de espaço na quantidade de "memórias" do código convolucional, o maior entrave à implementação do algoritmo foi vislumbrar uma estrutura de dados não-ingênua que permitisse codificar toda a informação necessária sem uso de memória excessiva (por exemplo, liberando espaço de memória ao descartar possíveis caminhos na treliça em meio ao algoritmo).

4) *Como a probabilidade de erro de transmissão foi estimada? Qual é o seu valor? Como ela se compara com o valor de p escolhido? Como ela muda com m ?*: A probabilidade de erro de transmissão foi estimada por meio da relação sinal ruído normalizada $(\frac{E_i}{N_0})$. Seu valor foi obtido pela equação $p(\frac{E_b}{N_0}) = Q(\sqrt{\frac{2E_b}{N_0}})$, que é a mesma para os dois canais utilizados nesse roteiro (BSC e AWGN), a qual compara a relação sinal ruído normalizada com seu valor de p . Convertendo a energia de bit transmitido (E_b) para a energia de bit de informação (E_i), tal que $E_b = \frac{E_i}{3}$, obtém-se a equação final utilizada $p(\frac{E_i}{N_0}) = Q(\sqrt{\frac{2E_i}{3N_0}})$. A partir do gráfico da Figura 4, ao fixar-se um valor específico de p , obtém-se valores distintos de $(\frac{E_i}{N_0})$ para cada valor de m , de tal forma que, quanto maior for o valor de m , menor será o valor de $\frac{E_i}{N_0}$.

5) *Qual é o tamanho final do seu executável?*: O programa foi codificado em linguagem Julia, a qual usa uma técnica de compilação "just in time" para converter o código a instruções de uma LLVM ("low level virtual machine") e o executar. Apesar de existirem desenvolvedores contribuindo para tornar possível a compilação prévia (ou seja, tornar possível a geração de um executável "standalone"), o código deste laboratório não pôde ser convertido num executável puro. Ainda assim, pode-se dizer, sobre os arquivos produzidos na linguagem Julia, que nenhum supera 2Kbytes de tamanho e, juntos, não superam 20Kbytes.

6) *Quanto tempo a sua solução demora para decodificar cada bit? Faça uma média.*: Os decodificadores de cada código convolucional trabalhado demoram, em média, $30\mu s$, $60\mu s$ e $300\mu s$ para decodificar 1 bit de informação.

B. Perguntas presentes em [4]

1) Compare todos com a capacidade do canal Gaussiano estabelecendo o valor de $(\frac{E_b}{N_0})$ que teoricamente permitiria a transmissão com probabilidade de erro tão baixa quanto se queira: Para obter o valor da razão sinal ruído utiliza-se a equação de Shannon-Hartley.

$$C = B \log_2(1 + \frac{E_b}{N_0}) \quad (3)$$

Convertendo a energia de bit transmitido (E_b) para a energia de bit de informação (E_i), tal que $E_b = \frac{E_i}{3}$, obtém-se a equação final utilizada.

$$C = \frac{1}{2} \log_2(1 + \frac{E_i}{3N_0}) \quad (4)$$

A partir das taxas de cada código - $\frac{4}{7}$ para os códigos de bloco, $\frac{7}{12}$ e $\frac{8}{14}$ para os códigos cíclicos e $\frac{1}{3}$ para os códigos convolucionais - obtém-se o valor de $(\frac{E_i}{N_0})$.

Tabela II: Valores de $(\frac{E_b}{N_0})$ para probabilidade de erro tão baixa quanto se queira

Código	$E_i/N_0(dB)$
Códigos de bloco	5.59
Código cíclico (7,12)	5.72
Código cíclico (8,14)	5.59
Códigos convolucionais	2.46

2) Compare os valores de $(\frac{E_b}{N_0})$ de fato necessário para transmitir com os sistemas acima com probabilidade de erro de 10^{-4} : Observando a Figura 5 obtemos os valores de $(\frac{E_b}{N_0})$ para probabilidade de erro de 10^{-4} traçando uma reta. Os valores obtidos estão representados na Tabela III.

Tabela III: Valores de $(\frac{E_b}{N_0})$ para probabilidade de erro de 10^{-4}

Código	$E_i/N_0(dB)$
Euclidean (m=6)	2.9
Euclidean (m=4)	3.9
Euclidean (m=3)	4.2
Exact (m=6)	5.1
Hamming (m=6)	5.2
Hamming (m=4)	5.9
Exact (m=4)	6
Exact (m=3)	6.5
Hamming (m=3)	6.6
Bloco Lab1	8.6
BCH	9
Cyclic (8,14)	>10
Bloco Hamming	>10
Cyclic (7,12)	>10

VI. CONCLUSÃO

O código cíclico apresenta a vantagem de ser necessário armazenar uma quantidade menor de síndromes e de não ser necessário associar síndrome a erro na decodificação. Isso introduz maior complexidade no passo de decodificação pois não há mapeamento direto entre síndrome e erro. No entanto, no código desenvolvido pelas equipes, foi necessário criar um método para geração de códigos. Dessa forma a complexidade do desenvolvimento do código cíclico é maior no tocante ao entendimento dos conceitos envolvidos e no código do

experimento anterior no tocante à criação de um procedimento para a geração da matriz de paridade.

Pôde-se notar do gráfico presente em Figura ?? que o código de Hamming é mais eficiente que quase todos os códigos implementados, com exceção do BCH. Isso era esperado uma vez que nenhum dos outros códigos cíclicos corrige mais bits que Hamming e Hamming possui bloco menor que os demais. Nota-se que os casos de distância mínima dois são ligeiramente piores que o canal não codificado e que o caso de distância mínima três é muito próximo de Hamming.

Pode-se depreender que os códigos cíclicos, da maneira como foram desenvolvidos, são ineficazes na diminuição da distância mínima com o aumento do tamanho do bloco. Em contraste a isso, os códigos do experimento anterior apresentados na Figura ?? são, em sua maioria, mais eficazes que Hamming.

Os tamanhos de códigos cíclicos utilizados variaram de 10 ao maior tamanho requisitado (16) e o método utilizado para gerar o conjunto de síndromes é extensível para tamanhos maiores de códigos. Vale notar que a etapa crítica de geração do conjunto de síndromes só precisa ser realizada uma única vez no préprocessamento, por isso, seu tempo de execução total não é crítico.

A medida entre tamanho do bloco e desempenho é dada pela complexidade da multiplicação e divisão polinomial, nesse caso foram consideradas ambas $\mathcal{O}(n^2)$. Além disso, foram discutidos mais detalhes sobre as complexidades das implementações na explicação do algoritmo.

Por outro lado, o código BCH possui rica estrutura matemática subjacente (corpos), possibilitando completa flexibilidade para gerar tamanhos de bloco arbitrariamente grandes e distâncias mínimas desejadas. Além de ser robusto por construção, ainda se caracteriza por complexidade linear de decodificação, como foi empiricamente validado.

REFERÊNCIAS

- [1] Berlekamp, Elwyn R. *Algebraic Coding Theory*. Edição revisada. Singapura: WSPC, 2015.
- [2] Sharma, Manish *Aula 2 - Códigos Cíclicos*. ELE32 - Introdução a Comunicações. 2018.
- [3] Sharma, Manish *Aula 3 - Código Convolucional*. ELE32 - Introdução a Comunicações. 2018.
- [4] Sharma, Manish *Aula 4 - Uma comparação justa*. ELE32 - Introdução a Comunicações. 2018.