ELE-32 Introdução a Comunicações

Aula 2- Códigos Cíclicos

September 12, 2018

1 Representação polinomial de vetores

- Seja $\mathbf{v} = [v_0, v_1, ..., v_{n-1}]$ um vetor n-dimensional não necessariamente binário. Este vetor pode ser representado através do polinômio $v(D) = v_0 + v_1D + v_2D^2 + \vdots + v_{n-2}D^{n-2} + v_{n-1}D^{n-1}$ onde D é uma variável *dummy* que serve para indicar a posição do termo do polinômio dentro do vetor: D^i representa um deslocamento de i posições.
- Definimos $v^{(1)}(D)$ como sendo igual a $v_{n-1}, v_0, v_1, v_2, ..., v_{n-3}, v_{n-2}$, ou seja, um deslocamento circular (cíclico) do vetor. A repetição deste deslocamento circular i vezes resulta em $v^{(i)}(D)$ que é iguala $v_{n-i}, v_{n-i+1}, ..., v_{n-2}, v_{n-1}, v_0, v_1, v_2, ..., v_{n-i-2}, v_{n-i-1}$.
- É natural que $v(D) = v^{(n)}(D)$, isto é, girar ciclicamente um vetor n vezes resulta no mesmo vetor.
- O grau do polinômio é o maior valor de i tal que $v_i \neq 0$. O maior grau possível para o polinômio equivalente de um vetor com dimensão n é $n-1$.
- O polinômio não carrega diretamente a informação sobre qual é o tamanho do vetor $\mathbf{v}$ se o grau do polinômio for menor do que $n-1$.
- Por convenção omitimos os termos que são iguais a zero exceto quando representamos a palavra toda nula. Isto é, a representação de [10100] é $v(D) = 1 + D^2$ e a representação de [00000] vale $v(D) = 0$. Note que no primeiro caso o mesmo polinômio pode representar os vetores [101], [1010] ou outros. O valor de n deve ser obtido pelo contexto.
- Operações como soma, subtração, produto e divisão podem ser feitas entre polinômios, resultando em novos vetores.

2 Códigos cíclicos

- Códigos cíclicos são uma sub-classe de códigos de bloco lineares. O fato de serem cíclicos simplifica os processos de codificação e decodificação. A redução na complexidade destes processos permite que códigos de bloco com maiores valores de n(tamanho de bloco) sejam implementados.
- Por definição, se $\mathbf{v} = [v_0, v_1, ..., v_{n-1}]$ é uma palavra código, $\mathbf{v}' = [v_{n-1}, v_0, v_1, ..., v_{n-2}]$ obtida através de um deslocamento cíclico de $\mathbf{v}$, também é.
- A palavra código $\mathbf{v}$ pode ser representada através de um polinômio equivalente $\mathbf{v}(D) = v_0 + v_1D + v_2D^2 + \cdots + v_{n-1}D^{n-1}$

- Multiplicar **v** por D^i não é suficiente para gerar uma nova palavra código através do deslocamento cíclico pois o maior grau possível do polinômio equivalente (que corresponde ao comprimento da palavra código menos um) seria aumentado em i.

- Por outro lado, podemos escrever

$$v(D)D^i = q(D)(1 + D^n) + v^{(i)}(D) \tag{1}$$

Logo, o resto da divisão de $v(D)D^i$ por $(1 + D^n)$ é uma palavra código, para qualquer valor de i inteiro.

- O grau do polinômio $v(D)$ pode ser qualquer inteiro entre 0 e $n - 1$, inclusive.

- Seja $g(D)$ uma palavra código não nula de um código com grau mínimo $n - k$. As seguintes propriedades são verdadeiras:
  - $g_0 = 1$;
  - $g(D)$ é único;
  - Qualquer combinação linear de $g(D)$ e $g(D)D, g(D)D^2, ..., g(D)D^{k-1}$ é uma palavra código pela definição de código linear e cíclico
  - Todas as palavras código podem ser escritas através da combinação linear de $g(D)$ e $g(D)D, g(D)D^2, ..., g(D)D^{k-1}$

- A matriz geradora do código pode ser obtida através do polinômio $g(D)$. Uma possível matriz geradora deste código pode ser escrita da forma:

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & g_3 & \cdots & g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k-3} & g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_{n-k-4} & g_{n-k-3} & g_{n-k-2} & g_{n-k-1} & g_{n-k} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k-1} & g_{n-k} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & g_{n-k-2} & g_{n-k-1} & g_{n-k} \end{bmatrix}$$

- A multiplicação de uma palavra de informação **u** pela matriz acima resultaria numa palavra código **v** que pode ser descrita como:

$$\begin{aligned} v(D) = \; & u_0 \cdot g(D) + u_1 \cdot g(D)D + \cdots + u_{k-1}g(D)D^{k-1} \\ = \; & u_0g_0 + (u_0g_1 + u_1g_0)D + (u_0g_2 + u_1g_1 + u_2g_0)D^2 + \cdots \\ & +(u_{k-3}g_{n-k} + u_{k-2}g_{n-k-1} \\ & +(u_{k-1}g_{n-k-2})D^{n-3} + (u_{k-2}g_{n-k} + u_{k-1}g_{n-k-1})D^{n-2} + u_{k-1}g_{n-k}D^{n-1} \end{aligned}$$

- Alternativamente, as palavras código podem ser obtidas através de uma palavra de informação $u(D)$ através da multiplicação polinomial $v(D) = u(D)g(D)$. Por este motivo, o polinômio $g(D)$ é chamado de polinômio gerador. É mais fácil multiplicar dois polinômios do que um vetor e uma matriz. É mais fácil armazenar $g(D)$ do que $G(D)$.

- O grau do polinômio resultante da multiplicação polinomial de $u(D)$ com $g(D)$ é a soma dos graus de $u(D)$ e $g(D)$. Como $v(D)$ deve ter grau menor ou igual a $n - 1$, o grau de $u(D)$ é no máximo $(n - 1) - (n - k) = k - 1$.

- Pode-se mostrar que $g(D)$ é um fator de $(1 + D^n)$. Logo, qualquer $g(D)$ pode ser obtido através da combinação dos fatores irredutíveis de $(1 + D^n)$. Isto é, se $(1 + D^n) = f_1(D)f_2(D) \times \cdots \times f_L(D)$, $g(D) = \prod_l f_l(D)$, onde o produto inclui somente alguns dos fatores de $(1 + D^n)$. A fatoração deve ser feita de forma semelhante a escrever um número como o produto de números primos: se $f_l(D)$ é um fator de $1 + D^n$, ele não pode ser decomposto em dois fatores $f_l^1(D)$ e $f_l^2(D)$.
- Por exemplo, $1 + D^4 = (1 + D)(1 + D + D^2)$. Só há dois fatores. Não podemos utilizar os dois ao mesmo tempo. Logo, há somente dois códigos cíclicos de tamanho $n = 3$, um obtido fazendo $g(D) = 1 + D$ (resultando num código de taxa 2/3 de verificação de paridade) e outro fazendo $g(D) = 1 + D + D^2$ (resultando num código de repetição de taxa 1/3).
- Para $n = 7$ podemos fatorar $1 + D^7 = (1 + D)(1 + D + D^3)(1 + D^2 + D^3)$. Lembrando que o grau de $g(D)$ é $n - k$, podemos combinar estes fatores de forma a obter um polinômio de grau 1, dois de grau 3, dois de grau 4 ou um de grau 6.

3 Decodificação

- A divisão de $(1 + D^n)$ por $g(D)$ resulta no polinômio $h(D)$
- A multiplicação da palavra código $v(D)$ por $h(D)$ e resulta em:

$$r(D) = v(D)h(D) = u(D)g(D)h(D) = u(D)(1 + D^n) = u(D) + D^n u(D) \tag{2}$$

- Como $u(D)$ tem grau de no máximo $k - 1$, os primeiros k coeficientes de $r(D)$ são iguais aos coeficientes de $u(D)$, assim como os últimos k coeficientes que tem grau entre n e $n + k - 1$, inclusive.
- Sendo $v(D)$ uma palavra código, os coeficientes de $r_k, r_{k+1}, ..., r_{n-1}$ valem zero. Isto permite a construção de $n - k$ equações de verificação de paridade
- Os valores dos coeficientes de $r(D)$ podem ser obtidos através da seguinte equação:

$$r_i = \sum_{j=0, j \le k}^{i} v_j h_{i-j} \tag{3}$$

- A matriz de verificação de paridade correspondente pode ser obtida através do polinômio recíproco de $h(D)$, que é $D^k h(D^{-1})$, da mesma forma como a matriz geradora pode ser obtida através do polinômio gerador $g(D)$. Assim, $\mathbf{H}$ tem o seguinte formato:

$$\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & h_k & \cdots & h_2 & h_1 & h_0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & h_0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & h_1 & h_0 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & h_2 & h_1 & h_0 \end{bmatrix}$$

- O recíproco de $h(D)$ também é um polinômio gerador e gera o código dual daquele gerado por $g(D)$. O recíproco também é um fator de $1 + D^n$

- Como qualquer palavra código pode ser escrita como $v(D) = a(D)g(D)$, o resto da divisão de qualquer palavra-código por $g(D)$ resultará em zero. Entretanto, se $v'(D) = v(D) + e(D)$ for dividido por $g(D)$ o resto será igual ao resto de $e(D)/g(D)$, que será nulo somente se $e(D)$ for uma palavra-código. O resto da divisão de $v(D)$ por $g(D)$ é chamado de síndrome, representado pelo polinômio $s(D)$.

- Podemos associar os padrões de erro detectáveis com uma síndrome ao escolher, entre todos os erros que causam uma síndrome, o mais provável . Assim, ao identificar uma síndrome, há um possível padrão de erro que pode ser corrigido.

- Se $s(D)$ é a síndrome de um vetor recebido $v'(D)$, o polinômio $Ds(D)$ deslocado ciclicamente em $g(D)$ é a síndrome de $Dv'(D)$ deslocado ciclicamente em relação a $(1 + D^n)$.
  - No deslocamento unitário em relação a $(1 + D^n)$, multiplicamos $a(D)$ por D. Se houver um termo igual a D^n, ele é removido e o termo 1 é adicionado.
  - No deslocamento unitário em relação a $g(D) = g_0 + g_1D^1 + \cdots + g_{n-k-1}D^{n-k-1} + g_{n-k}D^{n-k}$, multiplicamos $a(D)$ por D. Se houver um termo igual a D^n, ele é removido e o termo $g_0 + g_1D + \cdots + g_{n-k-1}D^{n-k-1}$, é adicionado. Implicitamente estamos dizendo que $D^{n-k} = g_0 + g_1D + \cdots + g_{n-k-1}D^{n-k-1}$
  - Exemplo: deslocamento circular de [101] em função de $1+D+D^3$ resulta sequencialmente em:

$$\begin{array}{llll}
101 \Rightarrow & 010'1 \rightarrow & 010+110 & =100 \\
100 \Rightarrow & 010'0 \rightarrow & & =010 \\
010 \Rightarrow & 001'0 \rightarrow & & =001 \\
001 \Rightarrow & 000'1 \rightarrow & 000+110 & =110 \\
110 \Rightarrow & 011'0 \rightarrow & & =011 \\
011 \Rightarrow & 001'1 \rightarrow & 001+110 & =111 \\
111 \Rightarrow & 011'1 \rightarrow & 011+110 & =101
\end{array} \tag{4}$$

    A dupla seta indica o deslocamento circular e a seta simples indica as consequências. Implicitamente estamos dizendo que $D^3 = 1 + D$

- A detecção por síndrome pode ser simplificada armazenando somente as síndromes associadas a erros na última posição da palavra código, aproveitando-se da propriedade do item anterior, através do seguinte algoritmo:
  1. Identifique a síndrome. Se síndrome igual a zero, pule para o penúltimo passo.
  2. Síndrome dentro do conjunto de síndromes associados a erros na última posição?
     - Não - pule para o passo 3
     - Sim - Troque o valor do último bit e retorne ao passo 1
  3. Gire ciclicamente a síndrome e a palavra, apropriadamente. Retorne ao passo 1.
  4. Gire ciclicamente a palavra código até que a posição seja igual à original
  5. Obtenha a palavra de informação dividindo a palavra código por $g(D)$

4 Atividades

1. Gere códigos com taxa semelhante ($\pm 5\%$) à taxa do código de Hamming do laboratório anterior para pelo menos 5 valores distintos de n entre 8 e 16. Para isso utilize o comando `cyclpoly(n,k,'all')` do MATLAB, que gera a representação vetorial de todos os polinômios geradores para códigos

cíclicos com tamanho n e taxa k/n, se existirem. Esta é a única função específica do MATLAB que você pode usar neste laboratório.

2. Entre todas as alternativas possíveis para o mesmo valor de k e n, pode haver uma melhor do que as outras. Calcule a distância mínima dos códigos gerados no item anterior e escolha, para o mesmo n e k, aquele com a maior distância mínima
3. Implemente o codificador para os códigos escolhidos no item anterior.
4. Sabendo a distância mínima, implemente o decodificador para os códigos escolhidos. O seu decodificador só pode armazenar as síndromes associadas a erros na primeira posição da palavra código. Além disso, o seu codificador não pode armazenar o padrão de erro associado a cada síndrome.
5. Calcule a probabilidade de erro para todos os seus códigos de forma semelhante a como foi feito para o código de Hamming do laboratório anterior. Compare os resultados.
6. [*Opcional e difícil*] Implemente um codificador e decodificador eficientes para códigos BCH, que são um tipo de códigos cíclicos.

5 Perguntas a serem respondidas no relatório

O relatório deve conter, além do que os alunos consideram necessário, as resposta das seguintes perguntas:

1. Quais foram as maiores dificuldades em implementar o código, o codificador e o decodificador para os códigos cíclicos?
2. Como a dificuldade de projeto de um código cíclico se compara à dificuldade do projeto do código inventando por você no laboratório anterior?
3. Compare o desempenho dos códigos criados neste laboratório com o desempenho do código de Hamming (que também é um código cíclico)
4. Qual foi o método utilizado para encontrar o código maior? Este método é extensível para qualquer tamanho de bloco?
5. Qual é a relação medida entre o tamanho do bloco e o desempenho?
6. Qual é a complexidade de codificação e decodificação do seu sistema?

6 Bibliografia recomendada

- S. Haykin, “Communication Systems”, capítulo 10
- Shu Lin, “Error control coding : fundamentals and applications”, capítulo 5