UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE ENGENHARIA ELÉTRICA

TRABALHO DE CONCLUSÃO DE CURSO

SISTEMA DE TRANSMISSÃO DIGITAL USANDO MATLAB

Aluno
Johannes Dantas de Medeiros Júnior
johannesee@gmail.com

Orientador
Prof. Dr. Edmar Candeia Gurjão
ecandeia@dee.ufcg.edu.br

Campina Grande, Dezembro de 2008

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

JOHANNES DANTAS DE MEDEIROS JÚNIOR

SISTEMA DE TRANSMISSÃO DIGITAL USANDO MATLAB

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Campina Grande, como requisito parcial para a obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Edmar Candeia Gurjão

Campina Grande, Dezembro de 2008

Dedicatória

Aos meus pais, Johannes e Lúcia, à minha irmã,
Palas Atenëia e ao meu tio, Giovanni Bosco.

Lista de Figuras

Lista de Tabelas

Sumário

1 – Introdução

Como nos sistemas analógicos, nos sistemas de comunicação digital muitas vezes os sinais devem ser modulados antes de serem transmitidos, dos diversos tipos de modulação digital existentes a modulação *M-ary Phase Shift Keying* (M-PSK) foi estudada neste trabalho devido ao seu grande uso.

A seguir será apresentado como é realizada a modulação M-PSK, assim como resultados de simulações de transmissão de sinais por canais com ruído branco aditivo (*Additive White Gaussian Noise* - AWGN). Também são apresentados procedimentos para transmitir sinais usando esse tipo de modulação entre dois computadores usando suas respectivas placas de som, em todos os casos foi usado o programa Matlab® para realizar as tarefas.

Ao final do trabalho são apresentados e discutidos os resultados e algumas considerações são feitas, no apêndice são apresentadas as rotinas em Matlab® para realizar as tarefas apresentadas no trabalho.

2 – Modulação M-PSK

A modulação PSK é uma modulação digital na qual a fase do sinal modulado é determinada a partir dos valores dos dados da informação. Quando é usado um sinal binário teremos uma fase de 0° para um bit e 180° para o outro, já a amplitude do sinal modulado é constante, como mostrado na figura 1.

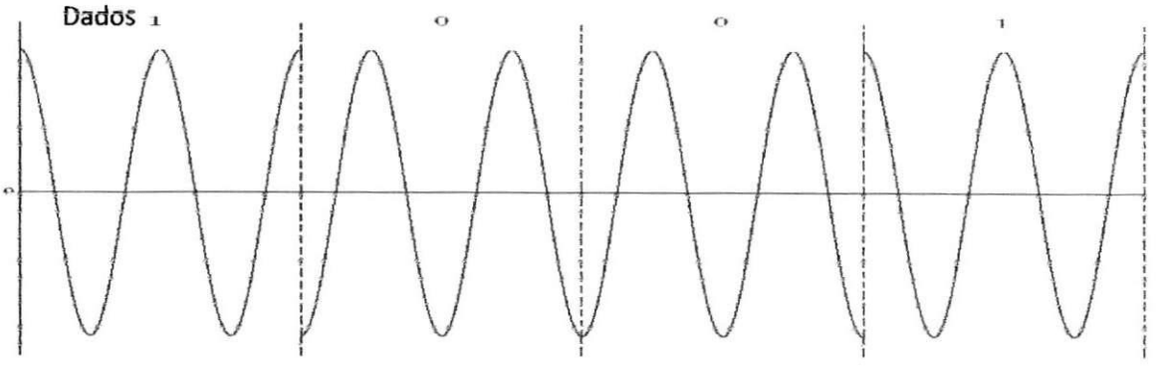
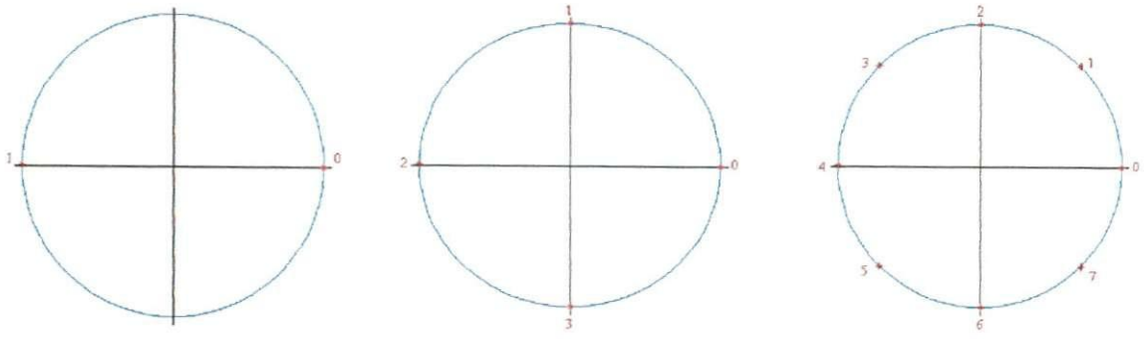


Figura 1: Modulação BPSK (HAYKIN, 2004)

A modulação mostrada na figura 1 é feita utilizando somente dois símbolos, 0 e 1, devido a este fato ela é denominada 2-PSK ou BPSK, onde "B" significa *Binary*, no entanto é possível realizar modulação de sinais com mais símbolos usando essa técnica, resultando nas modulações M-PSK, onde $M = 2^n$, com n sendo um número natural.

Sistemas com modulação M-PSK são usados em detrimento da BPSK para transmitir sinais quando a necessidade é conservar a largura de banda, mesmo necessitando de aumento da potência do sinal. O uso de modulação M-PSK proporciona redução de n vezes na largura de faixa de transmissão necessária em relação à BPSK (HAYKIN, 2004).

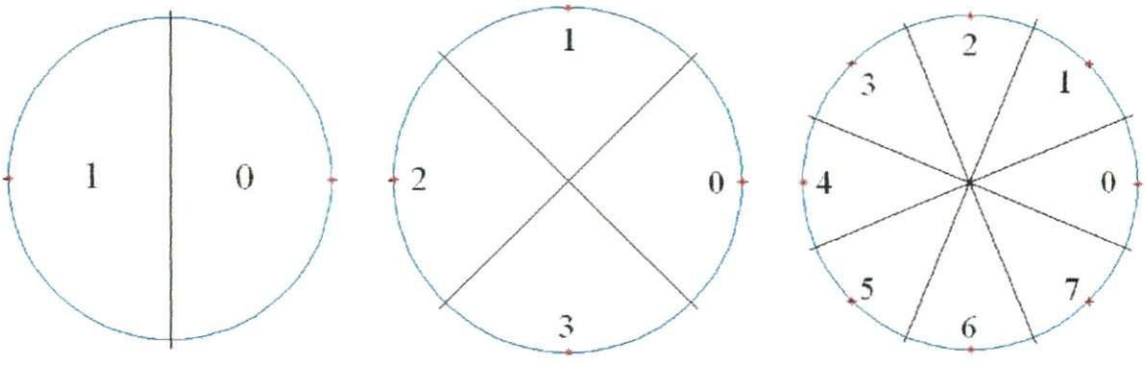
Os sinais são modulados em M-PSK atribuindo-se uma fase a cada um dos M símbolos, por exemplo, na modulação BPSK são usadas as fases de 0° e 180° para representar os bits 0 e 1, respectivamente. Os mapeamentos dos símbolos para as fases para a modulação M-PSK com M = 2, 4 e 8 são mostrados na figura 2.



(a) (b) (c)

Figura 2: Modulação M-PSK. (a) M = 2; (b) M = 4; (c) M = 8

Para realizar a demodulação M-PSK é necessário delimitar uma região de decisão para que o sinal seja recuperado corretamente, na modulação BPSK a decisão é feita baseada na parte real do sinal, caso seja positiva o bit é mapeado para 0, caso contrário para 1. Na figura 3 são mostradas as regiões de decisão para as modulações M-PSK, M = 2, 4 e 8.



(a) (b) (c)

Figura 3: Regiões de decisão. (a) M = 2; (b) M = 4; (c) M = 8

A região de decisão é necessária porque quando o sinal é transmitido através de um canal ele sofre a influência de ruido, se o modelo de canal for conhecido é possivel estimar o erro na transmissão a partir da relação sinal ruído (SNR, *Signal to Noise Ratio*). No caso do canal AWGN temos as seguintes expressões para estimar a probabilidade de erro de símbolo P_M (PROAKIS, SALEHI, 1998).

Para M = 2, temos:

$$P_2 = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{1}$$

Para M > 2:

$$P_M = 2Q\left(\sqrt{\frac{2kE_b}{N_0}}\sin\frac{\pi}{M}\right) \tag{2}$$

Onde, $k = \log_2(M)$ e E_b/N_0 é a SNR por bit do sinal.

Para obtermos a probabilidade de erro de bit, temos a seguinte aproximação:

$$P_b = \frac{1}{k}P_M \tag{3}$$

3 – Procedimentos Realizados e Resultados Obtidos

Foram desenvolvidas rotinas para realizar modulação e demodulação M-PSK, rotinas para estimar a probabilidade de erro da transmissão de sinais em canais AWGN usando esse tipo de modulação e rotinas para transmitir e receber sinais usando modulação M-PSK através da placa de som de um computador. Para testes foram utilizadas as modulações BPSK, 4-PSK e 8-PSK. Todas as rotinas foram desenvolvidas usando o Matlab® versão 6.5, essas rotinas são apresentadas no Apêndice.

3.1 - Simulação

Inicialmente foi desenvolvida uma rotina que realiza a modulação e depois a demodulação M-PSK de um sinal, ao final, o sinal resultante da demodulação era comparado com o sinal original. Para realizar a modulação foi usada a função do Matlab *dmodce* e para realizar a demodulação *ddemodce*. Utilizou-se a transmissão de 10 000 bits. Foi observado que os bits foram demodulados corretamente, resultando em erro de 0%.

Posteriormente foram realizadas simulações de transmissão de sinais modulados em M-PSK através de canais AWGN, foram feitas duas rotinas, uma para modulação BPSK e outra para as modulações 4 e 8-PSK, foram feitas cem simulações para cada uma das três modulações e também foram enviados 10 000 bits.

Para obter os valores teóricos foram usadas as equações 1, 2 e 3, para obter o valor simulado foi usada a função *gera_ruido* que gera ruído branco. Posteriormente esse ruído era somado ao sinal original e depois o sinal resultante era demodulado e comparado com o sinal original e o erro determinado para cada valor de SNR utilizado, esses valores variaram de 0 a 14 dB. As curvas obtidas são mostradas nas figuras 4, 5 e 6.

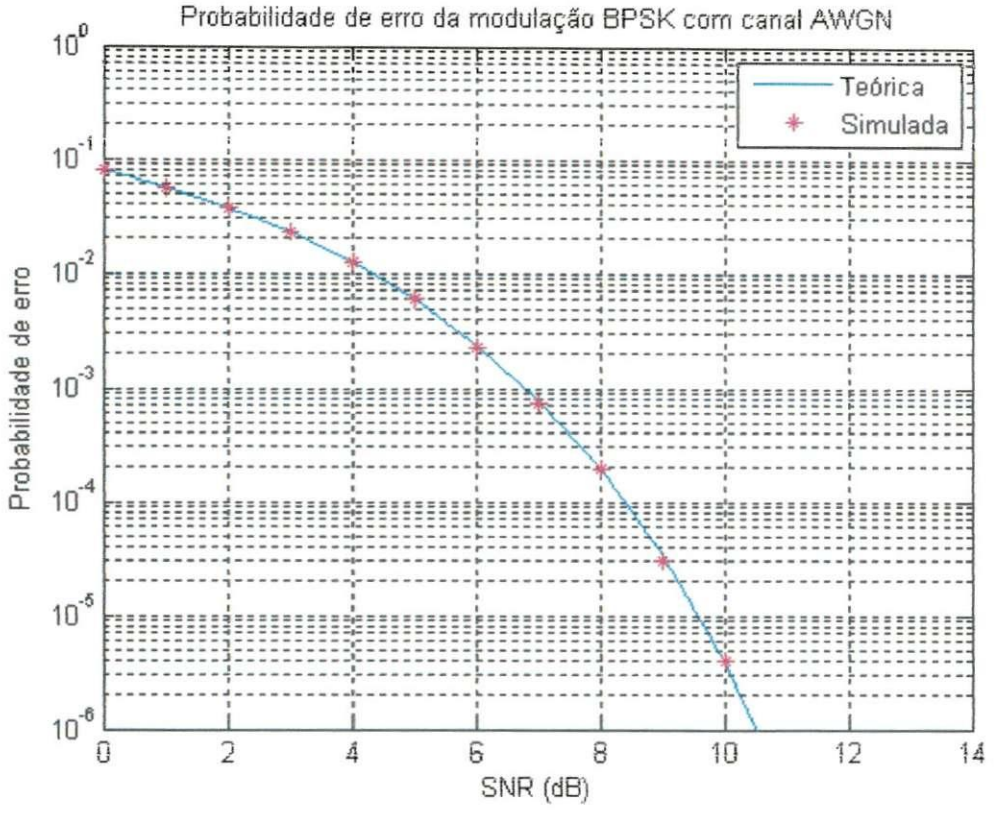


Figura 4: Resultado da simulação de modulação BPSK com canal AWGN

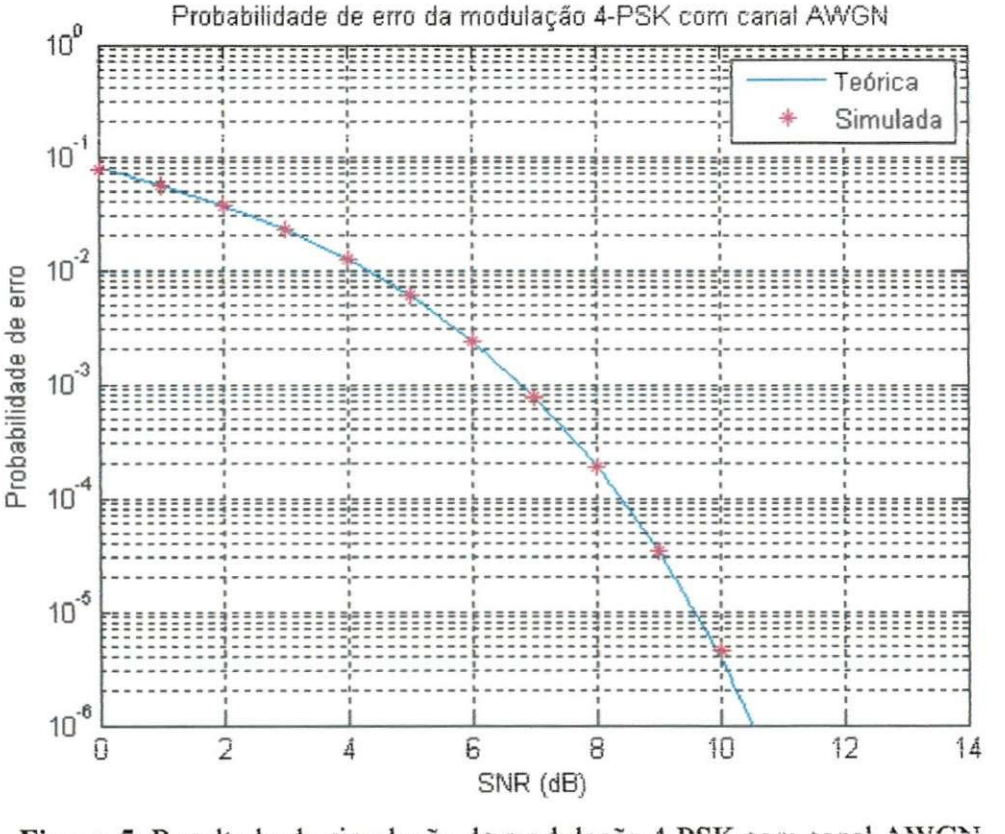


Figura 5: Resultado da simulação de modulação 4-PSK com canal AWGN

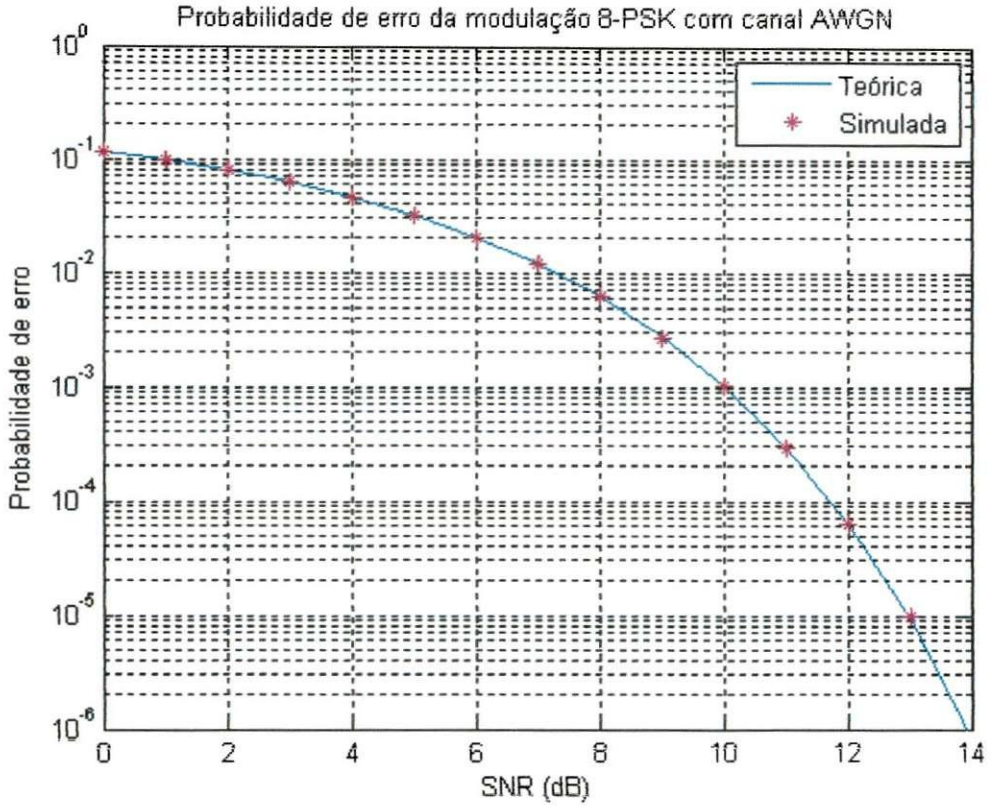


Figura 6: Resultado da simulação de modulação 8-PSK com canal AWGN

Também foram implementados algoritmos para realizar as modulações e demodulações BPSK, 4-PSK e 8-SPK, os resultados desses algoritmos foram comparados com os resultados das funções do Matlab, essas simulações foram realizadas 10 000 vezes, para cada modulação e demodulação, em todas elas o erro foi igual a 0%.

3.2 – Recepção e Transmissão

Outra parte do trabalho consistiu em transmitir e receber dados usando modulação M-PSK, novamente com M = 2, 4 e 8. Para realizar esse procedimento foram utilizados dois computadores equipados com placa de som, um computador foi usado como transmissor e outro como receptor. Na figura 7 é mostrada uma representação desse sistema e na figura 8 é mostrado o cabo utilizado para interligar os dois computadores, o cabo utilizado é um cabo P2-Macho/P2-Macho, os conectores P2 também são conhecidos como 3,5 mm.

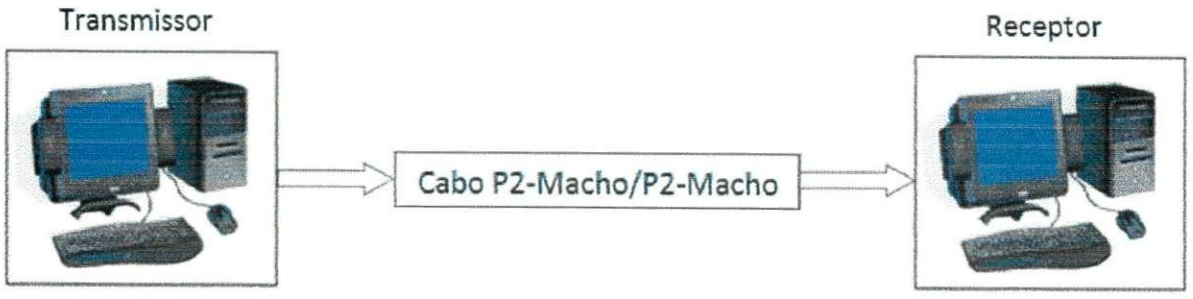


Figura 7: Esquema do sistema de transmissão/recepção

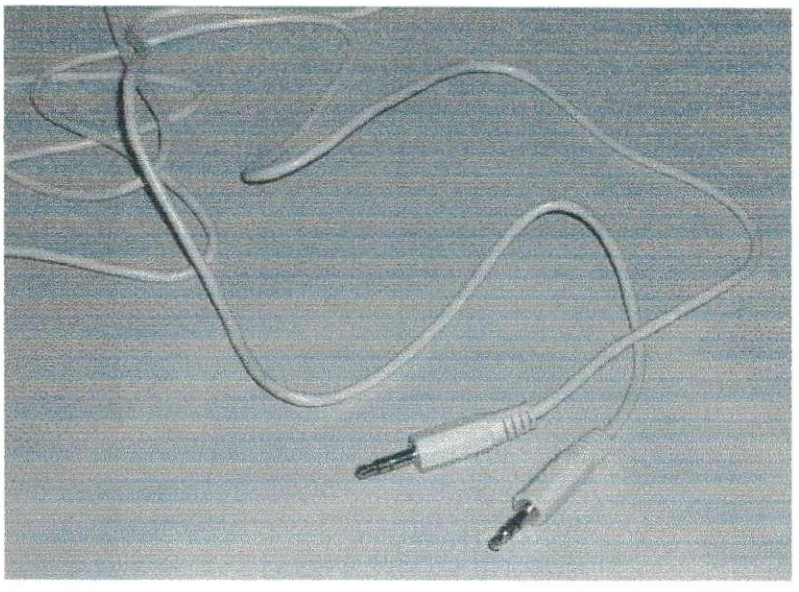


Figura 8: Cabo utilizado para interligar os computadores

Para realizar a transmissão e a recepção foram utilizadas funções do Matlab disponiveis na *Data Acquisition Toolbox*, essas funções permitem enviar e receber dados utilizando a placa de som do computador.

A placa de som trabalha com valores de tensão de -1 V a +1 V, assim o sinal modulado devia ser adequado a essa faixa de valores, quando foi utilizada a modulação BPSK isso não trouxe problemas porque usamos somente dois valores, então o bit 0 foi mapeado para 1 V e o bit 1 para -1 V.

No entanto ao utilizar as modulações 4 e 8-PSK foi necessário realizar alguns ajustes, pois os sinais resultantes dessas modulações são números complexos, mas o dispositivo só trabalha com valores reais, assim cada simbolo foi mapeado para certo valor de tensão. Esse mapeamento é mostrado nas tabelas 1 e 2.

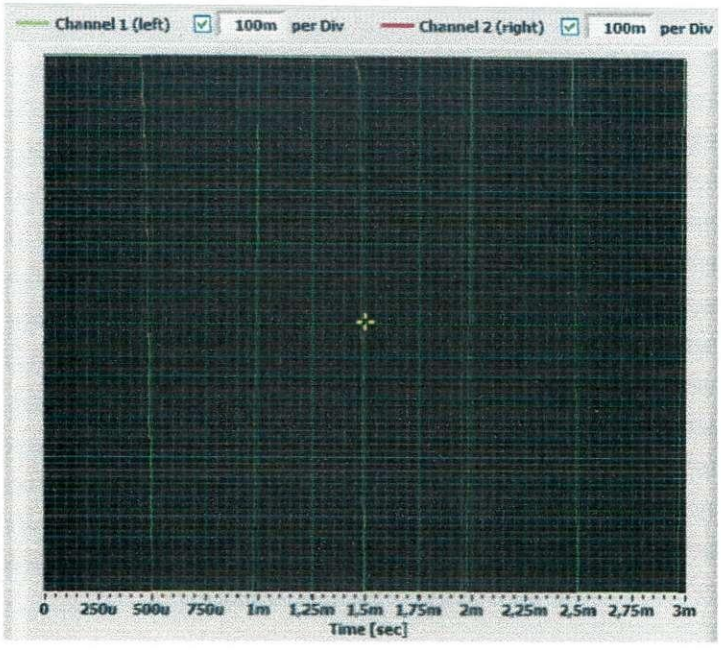
Tabela 1: Mapeamento dos símbolos em tensão na modulação 4-PSK

Símbolo	Tensão (V)
0	1
1	0,3
2	-0,3
3	-1

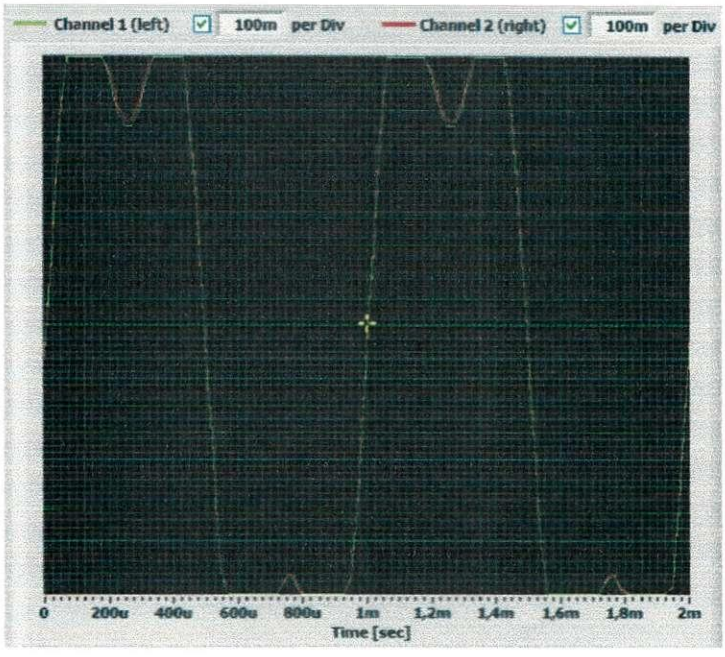
Tabela 2: Mapeamento dos símbolos em tensão na modulação 8-PSK

Símbolo	Tensão (V)
0	1
1	0,75
2	0,50
3	0,25
4	-0,20
5	-0,40
6	-0,65
7	-1

Esses valores foram obtidos de modo a facilitar a demodulação do sinal posteriormente, pois o valor que será obtido na saída depende do volume do alto-falante do transmissor, esse fato é mostrado na figura 9, onde são mostradas duas saídas obtidas para o mesmo sinal enviado, mas com volumes diferentes. Na 9.a é mostrada a saída com o volume máximo e na 9.b com o volume em torno de 15% do máximo. O sinal mostrado foi adquirido no computador receptor, para visualizá-lo foi usado o programa *Soundcard Scope*.



(a)



(b)

Figura 9: Sinal transmitido (a) Usando volume máximo; (b) Volume em 15% do máximo

Os sinais utilizados na transmissão consistiam de 8 000 amostras e eram distribuídos uniformemente de 0 a (M – 1), nas figuras 10, 11 e 12 são mostradas amostras dos sinais transmitidos antes de serem modulados, na figura 13 é mostrado o sinal recebido pela placa de som do receptor usando modulação 4-PSK e na 14 usando 8-PSK. As transmissões foram realizadas com o volume do alto-falante do computador em torno de 15%.

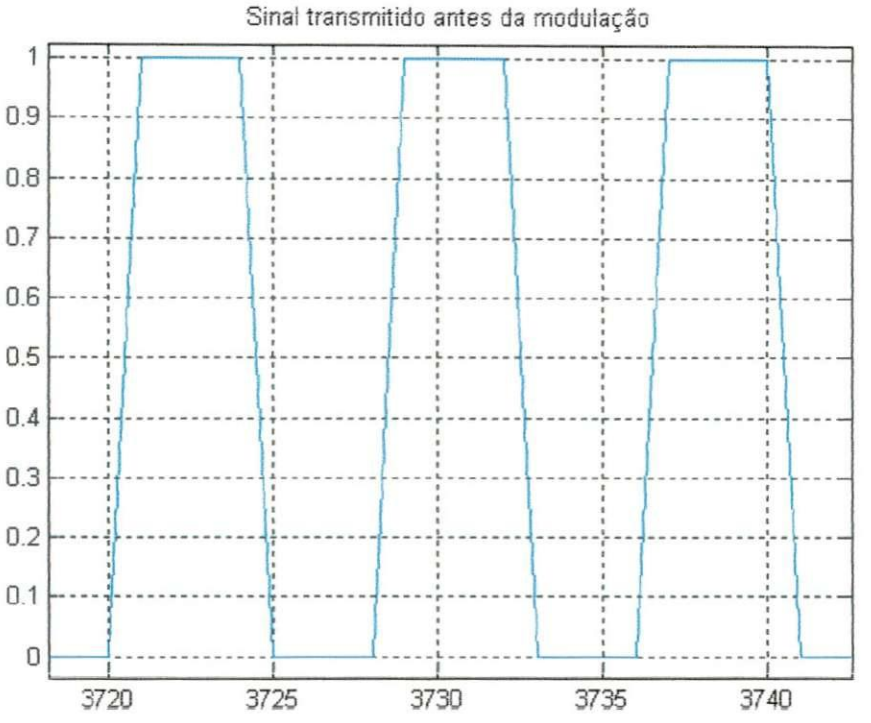


Figura 10: Amostra do sinal usado na transmissão com modulação BPSK antes de ser modulado

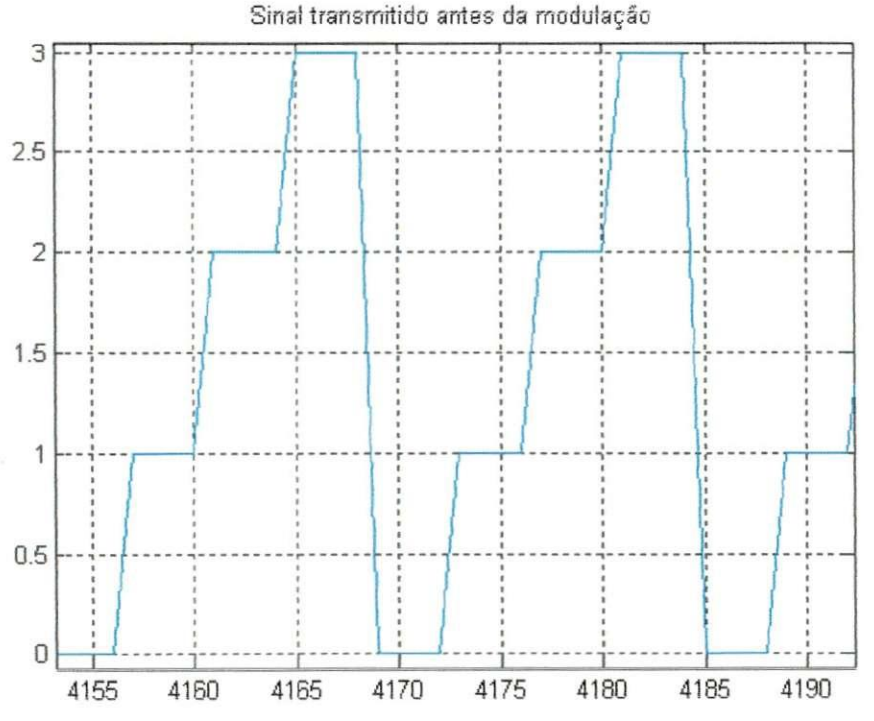


Figura 11: Amostra do sinal usado na transmissão com modulação 4-PSK antes de ser modulado

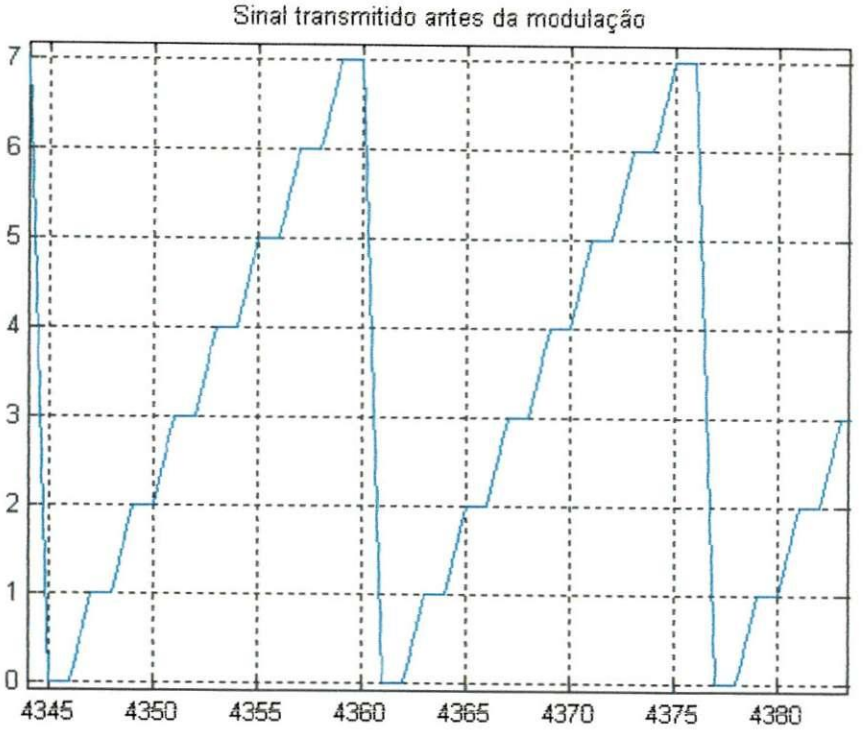


Figura 12: Amostra do sinal usado na transmissão com modulação 8-PSK antes de ser modulado

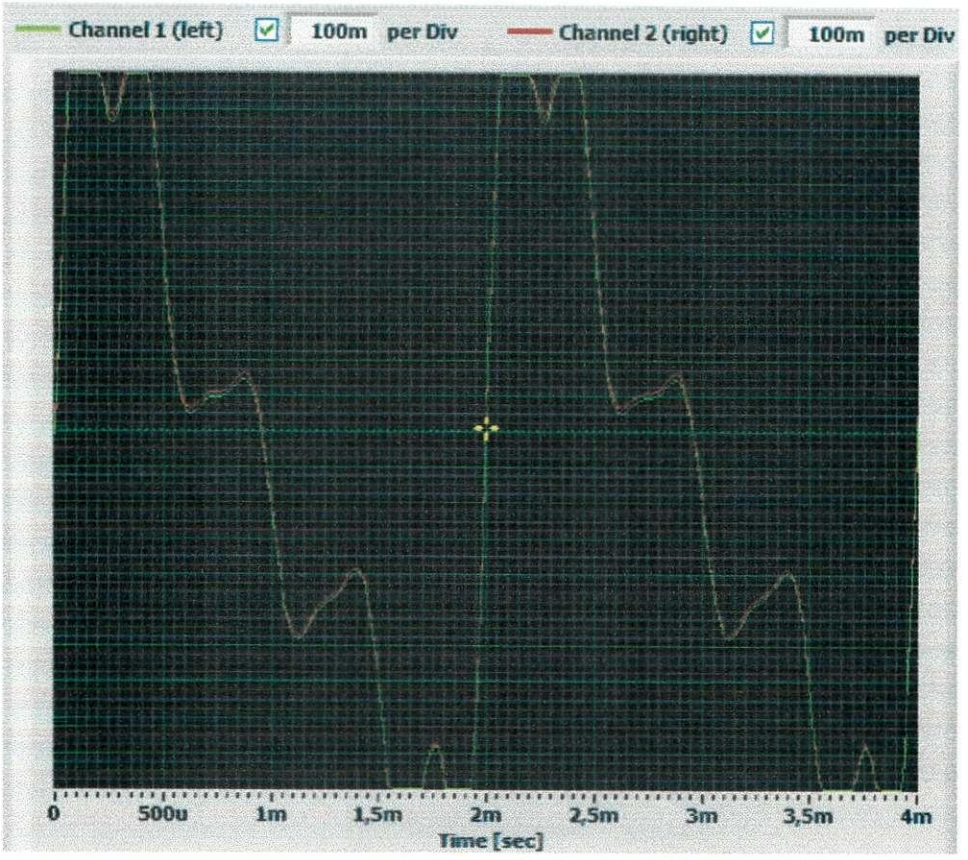


Figura 13: Sinal recebido modulado em 4-PSK

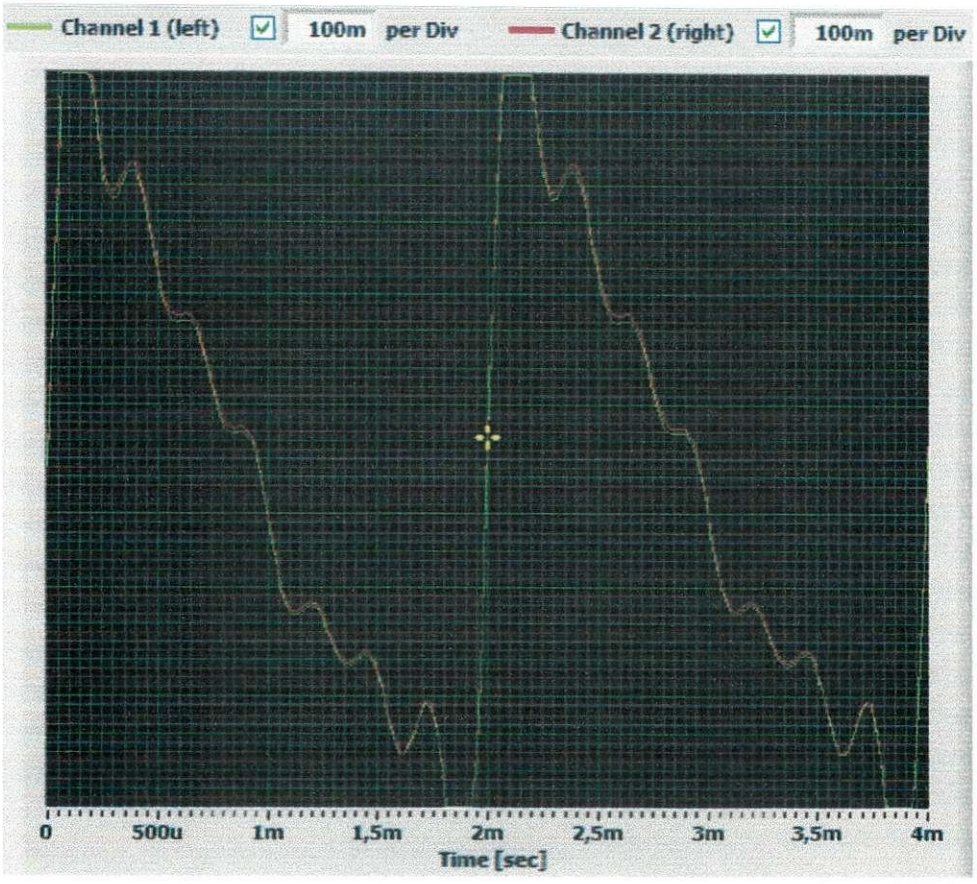


Figura 14: Sinal recebido modulado em 8-PSK

Um dos problemas encontrados na transmissão foi como detectar o início do sinal transmitido, isso foi resolvido adicionando-se oito símbolos antes do sinal, esses símbolos foram agrupados da seguinte forma: [(M - 1) (M - 1) 0 0 (M - 1) (M - 1) 0 0], assim na modulação 8-PSK temos que o sinal que será modulado é do tipo: [7 7 0 0 7 7 0 0 sinal].

Na recepção o canal do dispositivo recebe um número de amostras de até dez vezes o tamanho do sinal e demodula essas amostras, após isso tenta encontrar a seqüência que indica o início da transmissão, quando essa seqüência é encontrada as 8 000 amostras seguintes são demoduladas e o sinal obtido é comparado com o sinal transmitido. O sinal transmitido é gerado tanto no transmissor como no receptor.

Todas as amostras recebidas são demoduladas para depois serem testadas, na figura 15 é mostrado um conjunto de 80 000 amostras recebidas e demoduladas quando é usada a modulação 8-PSK. Podemos observar que o ruído é demodulado para o símbolo 4, enquanto o sinal varia de 0 a 7, essa variação não pode ser observada claramente devido à grande quantidade de amostras no gráfico. Os valores de tensão usados para realizar a decisão na demodulação foram escolhidos observando o sinal recebido.

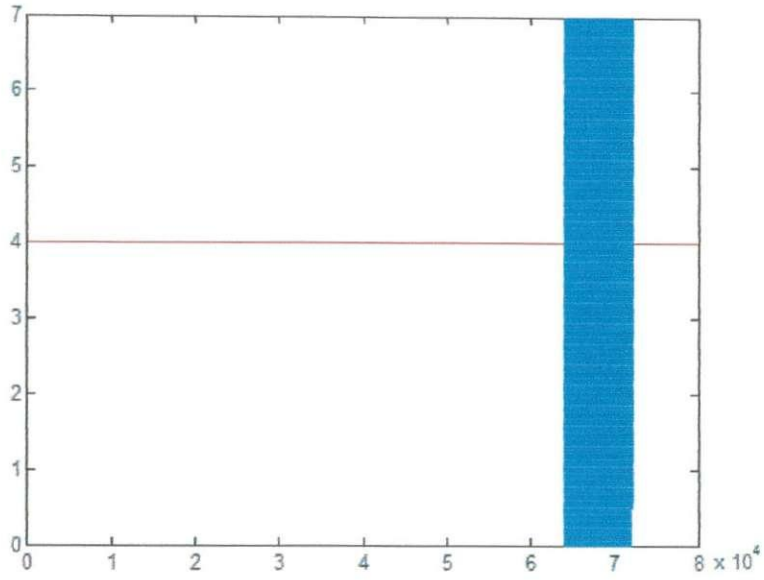


Figura 15: Exemplo de sinal recebido demodulado

Nesse sistema de transmissão podem ocorrer dois tipos de erro: a seqüência que indica o sinal não ser identificada, ou realizar a demodulação de alguma amostra errada. No 1° caso é possível identificar o início do sinal observando o gráfico do sinal recebido, como mostrado na figura 16, essa técnica foi usada na transmissão com modulação 8-PSK. Na figura 17 é mostrado o 2° tipo de erro.

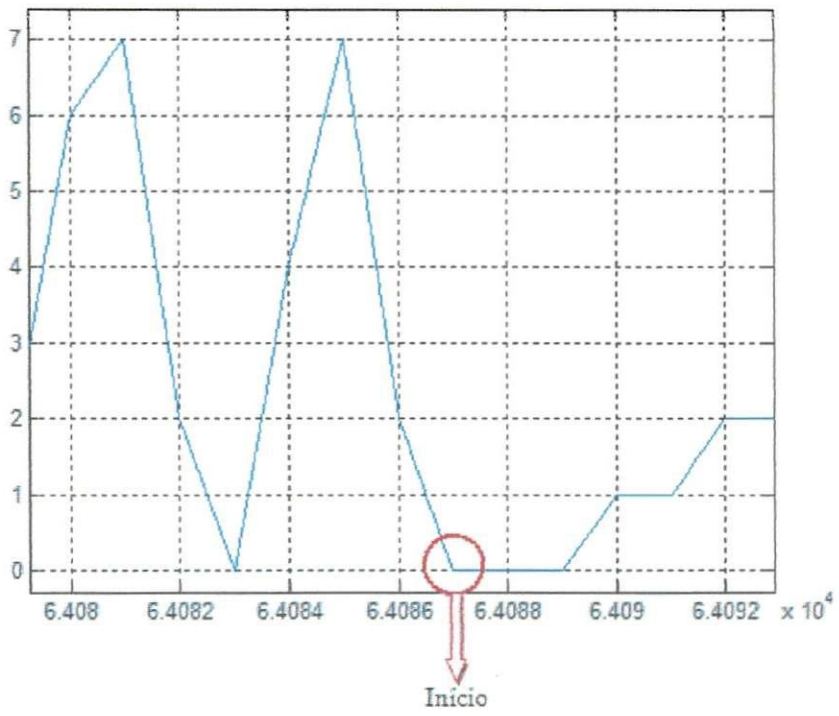


Figura 16: Exemplo de erro na identificação da seqüência de início

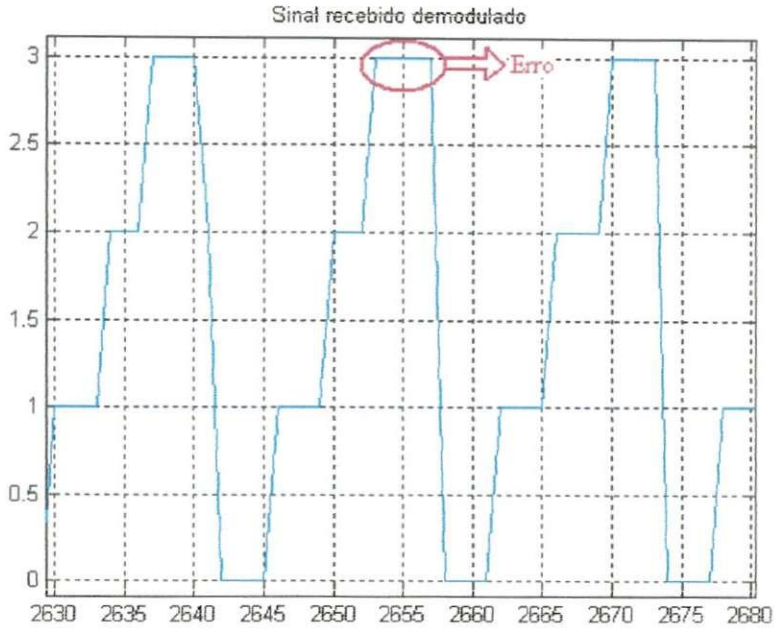


Figura 17: Exemplo de erro na demodulação do sinal

Para a modulação BPSK foram feitas dez transmissões, em todas elas o início do sinal foi identificado corretamente, para a 4-PSK foram feitas tentativas até obter dez transmissões corretas e para a 8-PSK foram feitas dez transmissões, no entanto somente duas foram concluídas corretamente. Os erros obtidos para os três tipos de modulação e o número de tentativas até obter uma transmissão correta são mostrados nas tabelas 3, 4 e 5.

Tabela 3: Resultado da transmissão usando modulação BPSK

	Número de tentativas	Erro percentual
1	1	1,0625
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0
10	1	0
Média	**1**	**0,10625**

Tabela 4: Resultado da transmissão usando modulação 4-PSK

	Número de tentativas	Erro percentual
1	3	0,1250
2	2	0,1125
3	2	0,1125
4	2	0,1250
5	1	4,4500
6	3	4,1375
7	2	0,1000
8	1	0,1500
9	1	6,3375
10	1	0,1000
Média	**1,8**	**1,5750**

Tabela 5: Resultado da transmissão usando modulação 8-PSK

	Número de tentativas	Erro percentual
1	*	18,8500
2	1	12,6125
3	*	37,1750
4	*	25,1000
5	*	56,2000
6	*	18,8250
7	*	56,1875
8	*	20,6625
9	1	12,6250
10	*	56,2000
Média	*	**31,5012**

* Seqüência de início não identificada

4 – Análise dos Resultados

As primeiras simulações realizadas que consistiam em modular e demodular um sinal não apresentaram erro, o que já era esperado por se tratar somente de operações matemáticas.

A segunda parte das simulações consistiu em simular transmissões através de canais AWGN e comparar com o resultado teórico representado pelas equações de 1 a 3, como pôde ser visto nas figuras de 4 a 6 o resultado da simulação foi condizente com o teórico.

Na última parte das simulações foram realizadas comparações entre as funções desenvolvidas no trabalho e as funções disponíveis no Matlab, o resultado obtido foi satisfatório já que não houve erro algum nas simulações, esse resultado é importante porque esses algoritmos de modulação e demodulação foram usados na transmissão e recepção dos sinais.

A transmissão na qual houve menor erro foi quando se utilizou a modulação BPSK, isso era esperado porque para haver uma demodulação errada é necessário que o sinal sofra uma grande atenuação até próximo de 0 V para que com o ruído ele tivesse a fase invertida.

Quando o sinal era modulado em 4-PSK algumas seqüências que identificam o início da transmissão foram perdidas e quando o sinal foi reconhecido o erro foi maior do que na BPSK, isso era esperado porque ao utilizarmos mais níveis de tensão os valores utilizados são menores e conseqüentemente precisam sofrer atenuação menor para mudar de fase.

Um problema encontrado ao usar essa modulação foi a forma como o sinal é recebido na placa de som. Como mostrado na figura 13 o sinal não é simétrico, no entanto deveria ser, já que foram enviados os valores de ± 1 V e ± 0,3 V, além disso o sinal apresenta uma pequena oscilação que dificulta a demodulação. Nesse caso não é possível aumentar o volume da saída da placa de som do computador porque isso faria com que os valores se aproximassem de ± 1 V, pois quando o valor enviado à placa de som está fora dessa faixa ele é grampeado para ± 1 V.

Na transmissão usando modulação 8-PSK foram encontrados problemas semelhantes aos que apareceram quando foi usada a 4-PSK. No entanto nesse caso o efeito foi pior, pois são usados oito valores de tensão, o que causou várias perdas da seqüência que indica o início do sinal e mesmo identificando o início através do gráfico do sinal recebido o erro foi muito grande quando comparado aos das outras duas modulações.

5 – Considerações Finais

O uso de modulações digitais, especialmente a PSK tem crescido muito juntamente com o avanço da transmissão digital e com esse trabalho percebemos como essa modulação pode ser aplicada na transmissão de dados.

Percebemos também que o uso de simulação é importante antes de começar a implantação de algum sistema e no caso desse trabalho vimos que a modulação que apresentou o melhor resultado foi a modulação BPSK, entretanto em outras situações onde a largura de faixa fosse um problema crítico o resultado não seria o mesmo.

6 – Referências

HAYKIN, S. **Sistemas de Comunicação**. 4ª Ed. Porto Alegre: Bookman, 2004.

PROAKIS, J. G., SALEHI, M. **Contemporary Communications Systems Using Matlab**. 1ª ed. Boston: PWS Publishing Company, 1998.

STOREY, B. D. **Using the Matlab Data Acquisition Toolbox**. Disponível em: <<http://faculty.olin.edu/bstorey/Notes/Card.pdf>>. Acesso em: 17 de Março de 2008

7 – Apêndice

mod_dmod_psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Modulação e demodulação M-PSK
%
clc;
clear all;
close all;
%
% Fd = freqüência do sinal que será transmitido (Hz)
% Fs = freqüência do sinal recebido (Hz)
% Fs / Fd deve ser um inteiro maior do que 1 (um)
Fd = 1e4;
Fs = 2e4;

duracao = 1; % s
num_bits = Fd*duracao; % Número de bits do sinal

M = 2; % M-PSK

% Sinal que vai ser modulado
x = round((M - 1)*rand(1, num_bits));

% Executa a modulação M-PSK
y = dmodce(x, Fd, Fs, 'psk', M);

% Executa a demodulação M-PSK para recuperar o sinal
z = ddemodce(y, Fd, Fs, 'psk', M);

num_erro = 0;

% Compara os dois sinais para detectar erros
for i = 1:num_bits,
    if x(i) ~= z(i)
        num_erro = num_erro + 1;
    end
end

% Erro percentual
erro = 100*num_erro/num_bits
```

erro_2psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Simulação de transmissão com canal AWGN usando modulação BPSK e
% comparação com a curva de probabilidade de erro teórica
%
clc;
clear all;
close all;
%
snr_db = (0:1:14); % SNR de 0 a 14 dB

snrv = 10.^(snr_db/10); % SNR em 'vezes'

% Probabilidade de erro modulação BPSK
Pb_2 = funcao_q(sqrt(2*snrv));

semilogy(snr_db, Pb_2, 'b'); % Traça o erro em função da SNR em dB
hold on;

% Fd = freqüência do sinal que será transmitido (Hz)
% Fs = freqüência do sinal recebido (Hz)
% Fs / Fd deve ser um inteiro maior do que 1 (um)
Fd = 1e4; %Hz
Fs = 2e4; %Hz
duracao = 1; %s
num_inter = 100; %Numero de simulações

M = 2; % BPSK

% Numero de bits do sinal
num_bits = Fd*duracao;

% Sinal a ser modulado, representa um conjunto de bits "um"
x = ones(1, num_bits);

tam_bloco = (Fs/Fd)*num_bits; % Usado na função que gera o ruído

for snr_db = 0:1:14,
    snr_v = 10^(snr_db/10); % SNR em 'vezes'
    variancia = 1/(snr_v);

    num_erro = 0;

    for inter = 1:num_inter,
        % Modulação PSK com M = 2, modulação BPSK
        y = dmodce(x, Fd, Fs, 'psk', M);

        % Gera o ruído
        ruido = gera_ruido(tam_bloco, variancia);

        % Adiciona ruído branco gaussiano ao sinal
        y_canal = y + ruido';

        % Executa a demodulação para obter o sinal
```

```
        z = ddemodce(y_canal, Fd, Fs, 'psk', M);

        % Compara os dois sinais para detectar erros
        for i = 1:num_bits,
            if x(i) ~= z(i)
                num_erro = num_erro + 1;
            end
        end

    end
    erro(snr_db + 1) = num_erro/(num_bits*num_inter);
end

snr_db = (0:1:14);
semilogy(snr_db, erro, '*r');

axis([0 14 10^-6 1]);
grid on;

title('Probabilidade de erro da modulação BPSK com canal AWGN');
legend('Teórica', 'Simulada');
xlabel('SNR (dB)')
ylabel('Probabilidade de erro')
```

erro_4_8_psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Simulação de transmissão com canal AWGN usando modulação M-PSK (M > 2) e
% comparação com a curva de probabilidade de erro teórica
%
clc;
clear all;
close all;
%
snr_db = (0:1:14); % SNR de 0 a 14 dB

snrv = 10.^(snr_db/10); % SNR em 'vezes'

M = 4;
k = log2(M);
Pb = (2/k)*funcao_q(sqrt(2*k*snrv).*sin(pi/M)); % Probabilidade de erro de
bit

% Fd = freqüência do sinal que será transmitido (Hz)
% Fs = freqüência do sinal recebido (Hz)
% Fs / Fd deve ser um inteiro maior do que 1 (um)
Fd = 1e4; %Hz
Fs = 2e4; %Hz
duracao = 10; %s
num_inter = 100; %Numero de simulações

% Número de bits do sinal
num_bits = Fd*duracao;

% Sinal que vai ser modulado
```

```
x = round((M - 1)*rand(1, num_bits));

tam_bloco = (Fs/Fd)*num_bits; % Usado na função que gera o ruído

for snr_db = 0:1:14,
    snr_db
    snr_v = 10^(snr_db/10);
    variancia = 1/(snr_v);

    num_erro = 0;

    for inter = 1:num_inter,
        % Modulacao PSK com M = 4
        y = dmodce(x, Fd, Fs, 'psk', M);

        % Gera o ruído
        ruido = gera_ruido(tam_bloco, variancia/k);

        % Adiciona ruido branco gaussiano ao sinal
        y_canal = y + ruido';

        % Executa a demodulação para obter o sinal
        z = ddemodce(y_canal, Fd, Fs, 'psk', M);

        % Compara os dois sinais para detectar erros
        for i = 1:num_bits,
            if x(i) ~= z(i)
                num_erro = num_erro + 1;
            end
        end

    end
    erro(snr_db + 1) = num_erro/(num_bits*num_inter);
end

snr_db = (0:1:14);

figure,semilogy(snr_db, Pb, 'b'); % Traça o erro em função da SNR em dB
hold on;

semilogy(snr_db, erro/k, '*r'); % Traça o erro simulado

axis([0 14 10^-6 1])
grid on;

title('Probabilidade de erro da modulação M-PSK com canal AWGN');
legend('Teórica', 'Simulada');
xlabel('SNR (dB)')
ylabel('Probabilidade de erro')
```

função_q.m

```
function y = funcao_q (x)
%
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Função Q -> Q(X)

y = 0.5*erfc(x/sqrt(2));
```

gera_ruido.m

```
%ALESSANDRA RODRIGUES
%EM 25 DE MARCO DE 2005
%GERA RUIDO

function ruido = gera_ruido(TAM_BLOCO,variancia)

r1      = sqrt(variancia)*(randn(TAM_BLOCO,1));    % Ruido Real Gerado
r2      = sqrt(variancia)*(randn(TAM_BLOCO,1));    % Ruido Imaginario Gerado

ruido   = r1 + sqrt(-1)*r2;

%var_r1=var(r1)
%var_r2=var(r2)
```

compara_mod_2psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a modulação 2-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Número de bits do sinal
num_bits = 8e3;

% Modulaçao 2-PSK
M = 2;

% Número de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Sinal que vai ser modulado
    x = round((M - 1)*rand(1, num_bits));
```

```
    % Modulação PSK com M = 2
    y_matlab = dmodce(x, 8e3, 8e3, 'psk', M);

    % Executa a modulação 2-PSK para obter o sinal
    y = (-1).^x;

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((y(a) - y_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

compara_mod_4psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a modulação 4-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Numero de bits do sinal
num_bits = 8e3;

% Modulaçao 4-PSK
M = 4;

% Numero de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Sinal que vai ser modulado
    x = round((M - 1)*rand(1, num_bits));

    %Modulacao PSK com M = 4
    y_matlab = dmodce(x, 8e3, 8e3, 'psk', M);

    %Executa a modulacao 4-PSK para obter o sinal
    for a=1:num_bits,
        if x(a) == 0,
            y(a) = 1;
        elseif x(a) == 1,
            y(a) = sqrt(-1); % = i
        elseif x(a) == 2,
            y(a) = -1;
```

```
        else
            y(a) = -sqrt(-1); % = -i
        end
    end

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((y(a) - y_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

compara_mod_8psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a modulação 8-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Numero de bits do sinal
num_bits = 8e3;

% Modulaçao 8-PSK
M = 8;

% Numero de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Sinal que vai ser modulado
    x = round((M - 1)*rand(1, num_bits));

    %Modulacao PSK com M = 8
    y_matlab = dmodce(x, 8e3, 8e3, 'psk', M);

    % Executa a modulacao 8-PSK para obter o sinal
    i = sqrt(-1);
    for a=1:num_bits,
        if x(a) == 0,
            y(a) = cos(0)+i*sin(0); % = 1
        elseif x(a) == 1,
            y(a) = cos(pi/4)+i*sin(pi/4);
        elseif x(a) == 2,
            y(a) = cos(pi/2)+i*sin(pi/2); % = i
        elseif x(a) == 3,
```

```
            y(a) = cos(3*pi/4)+i*sin(3*pi/4);
        elseif x(a) == 4,
            y(a) = cos(pi)+i*sin(pi); % = -1
        elseif x(a) == 5,
            y(a) = cos(5*pi/4)+i*sin(5*pi/4);
        elseif x(a) == 6,
            y(a) = cos(3*pi/2)+i*sin(3*pi/2); % = -i
        else
            y(a) = cos(7*pi/4)+i*sin(7*pi/4);
        end
    end

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((y(a) - y_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

compara_demod_2psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a demodulação 2-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Numero de bits do sinal
num_bits = 8e3;

% Modulaçao 2-PSK
M = 2;

% Sinal que vai ser modulado
x = round((M - 1)*rand(1, num_bits));

%Modulacao PSK com M = 2, depois esse sinal será demodulado
y = dmodce(x, 8e3, 8e3, 'psk', M);

% Numero de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Realiza a demodulacao 2-PSK para obter o sinal
    z_matlab = ddemodce(y, 8e3, 8e3, 'psk', M);
```

```
    % Realiza a demodulacao 2-PSK para obter o sinal
    for a=1:num_bits,
        if (real(y(1, a)) > 0),
            z(1, a) = 0;
        else
            z(1, a) = 1;
        end
    end

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((z(a) - z_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

compara_demod_4psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a demodulação 4-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Numero de bits do sinal
num_bits = 8e3;

% Modulaçao 4-PSK
M = 4;

% Sinal que vai ser modulado
x = round((M - 1)*rand(1, num_bits));

%Modulacao PSK com M = 4, depois esse sinal será demodulado
y = dmodce(x, 8e3, 8e3, 'psk', M);

% Numero de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Realiza a demodulacao 4-PSK para obter o sinal
    z_matlab = ddemodce(y, 8e3, 8e3, 'psk', M);

    % Realiza a demodulacao 4-PSK para obter o sinal
    for a=1:num_bits,
        fase = phase(y(a));
```

```
        if (fase > -pi/4) && (fase <= pi/4)
            z(a) = 0;
        elseif (fase > pi/4) && (fase <= 3*pi/4)
            z(a) = 1;
        elseif (fase > 3*pi/4) && (fase <= 5*pi/4)
            z(a) = 2;
        else
            z(a) = 3;
        end
    end

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((z(a) - z_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

compara_demod_8psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Compara o algoritmo desenvolvido para realizar a demodulação 8-PSK com a
% função do Matlab
%
clc;
clear all;
close all;
%
% Numero de bits do sinal
num_bits = 8e3;

% Modulaçao 8-PSK
M = 8;

% Sinal que vai ser modulado
x = round((M - 1)*rand(1, num_bits));

%Modulacao PSK com M = 8, depois esse sinal será demodulado
y = dmodce(x, 8e3, 8e3, 'psk', M);

% Numero de vezes que o sinal vai ser modulado
num_simulacoes = 10000;

% Indica o numero de erros que ocorreu
conta_erro = 0;

for p = 1: num_simulacoes,
    % Realiza a demodulacao 8-PSK para obter o sinal
    z_matlab = ddemodce(y, 8e3, 8e3, 'psk', M);

    % Realiza a demodulacao 8-PSK para obter o sinal
```

```
    for a=1:num_bits,
        fase = phase(y(a));

        if fase < 0 % Faz um ajuste para que a fase fique de 0 a 2*pi
            fase = fase + 2*pi;
        end

        if (fase > pi/8) && (fase <= (pi/4 + pi/8))
            z(a) = 1;
        elseif (fase > (pi/4 + pi/8)) && (fase <= (pi/2 + pi/8))
            z(a) = 2;
        elseif (fase > (pi/2 + pi/8)) && (fase <= (3*pi/4 + pi/8))
            z(a) = 3;
        elseif (fase > (3*pi/4 + pi/8)) && (fase <= (pi + pi/8))
            z(a) = 4;
        elseif (fase > (pi + pi/8)) && (fase <= (5*pi/4 + pi/8))
            z(a) = 5;
        elseif (fase > (5*pi/4 + pi/8)) && (fase <= (3*pi/2 + pi/8))
            z(a) = 6;
        elseif (fase > (3*pi/2 + pi/8)) && (fase <= (7*pi/4 + pi/8))
            z(a) = 7;
        else
            z(a) = 0;
        end
    end

    % Compara os sinais para determinar se houve erro
    for a = 1:num_bits,
        if abs((z(a) - z_matlab(a))) >= 1e-10,
            conta_erro = conta_erro + 1;
        end
    end
end

% Erro percentual
erro = 100*conta_erro/(num_bits*num_simulacoes)
```

receptor_bpsk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a recepção de um sinal pela placa de
% som do computador usando modulação 2-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 2; % BPSK

% Abre o dispositivo analogico e o canal
AI=analoginput('winsound',0);
chan_in = addchannel(AI,1);
```

```
% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = SampleRate*duration;

% Ajusta as propriedades do canal
set(AI,'SampleRate',SampleRate);
set(AI,'SamplesPerTrigger',10*NumSamples);
set(AI,'TriggerType','Manual');

% Entrada
start(AI);
trigger(AI);

comecou = 0;
inicio = 1;
for ii = 1 : 10*NumSamples,

    % Se chegou ao fim do sinal
    if ((ii == (inicio + NumSamples + 1)) && comecou == 1)
        break;
    end
    entrada(ii, 1) = getdata(AI, 1); % Lê uma amostra da placa de som

    % Demodulaçao BPSK
    if (real(entrada(ii,1)) > 0),
        recebe(ii,1) = 0;
    else
        recebe(ii,1) = 1;
    end

    if ((ii > 7) && (comecou == 0)),
        if (recebe((ii - 7):ii, 1) == [1 1 0 0 1 1 0 0]'),
            inicio = ii % Marca onde começou o sinal
            comecou = 1; % Indica que a recepção do sinal começou
        end
    end
end

% Sinal recebido
data_in = recebe((inicio + 1):(NumSamples + inicio), 1);

%Apaga o conteudo e fecha o canal
waittilstop(AI, 10*duration + 1)
delete(AI)
clear AI

% Sinal que foi enviado
% Onda quadrada
for a=1:(NumSamples/4),
    if (mod(a,M) == 1),
        z(((4*a-3):4*a),1) = ones(4,1);
    else
        z(((4*a-3):4*a),1) = zeros(4,1);
    end
end
```

```
% Compara os dois sinais para detectar erros
num_erro = 0;
for i = 1: NumSamples,
    if data_in(i) ~= z(i)
        num_erro = num_erro + 1;
    end
end

% Exibe o erro percentual
disp('Erro percentual');
100*num_erro/NumSamples

plot(data_in);
title('Sinal recebido demodulado');
```

receptor_4psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a recepção de um sinal pela placa de
% som do computador usando modulação 4-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 4; % 4-PSK

% Abre o dispositivo analogico e o canal
AI=analoginput('winsound',0);
chan_in = addchannel(AI,1);

% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = SampleRate*duration;

% Ajusta as propriedades do canal
set(AI,'SampleRate',SampleRate);
set(AI,'SamplesPerTrigger',10*NumSamples);
set(AI,'TriggerType','Manual');

% Entrada
start(AI);
trigger(AI);

comecou = 0;
inicio = 1;
for ii = 1 : 10*NumSamples,
        % Se chegou ao fim do sinal
    if ((ii == (inicio + NumSamples + 1)) && comecou == 1)
        break;
    end
```

```
    recebe(ii, 1) = getdata(AI, 1); % Lê uma amostra da placa de som

    % Demodulaçao 4-PSK
    % Valores ajustados de acordo com os volumes das
    % placas de som dos dois computadores
    if (real(recebe(ii,1) >= 0.5))
        sinal_recebido(ii,1) = 0;
    elseif (real(recebe(ii,1) >= 0)) && (real(recebe(ii,1) < 0.5))
        sinal_recebido(ii,1) = 1;
    elseif (real(recebe(ii,1) < 0)) && (real(recebe(ii,1) > -0.5))
        sinal_recebido(ii,1) = 2;
    else
        sinal_recebido(ii,1) = 3;
    end

    if ((ii > 7) && (comecou == 0)),
        if (sinal_recebido((ii - 7):ii, 1) == (M-1)*[1 1 0 0 1 1 0 0]'),
            inicio = ii % Marca onde começou o sinal
            comecou = 1; % Indica que a recepção do sinal começou
        end
    end
end

% Sinal recebido
data_in = sinal_recebido((inicio + 1):(NumSamples + inicio), 1);

% Apaga o conteudo e fecha o canal
waittilstop(AI, 10*duration + 1);
delete(AI);
clear AI;

% Sinal enviado
for a=1:(NumSamples/4),
    if (mod(a,M) == 1),
        z(((4*a-3):4*a),1) = zeros(4,1);
    elseif(mod(a,M) == 2),
        z(((4*a-3):4*a),1) = ones(4,1);
    elseif(mod(a,M) == 3),
        z(((4*a-3):4*a),1) = 2*ones(4,1);
    else
        z(((4*a-3):4*a),1) = 3*ones(4,1);
    end
end

% Compara os dois sinais para detectar erros
num_erro = 0;
for i = 1: NumSamples,
    if data_in(i) ~= z(i),
        num_erro = num_erro + 1;
    end
end

% Exibe o erro percentual
disp('Erro percentual');
100*num_erro/NumSamples

plot(data_in);
title('Sinal recebido demodulado');
```

receptor_8psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a recepção de um sinal pela placa de
% som do computador usando modulação 8-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 8; % 8-PSK

% Abre o dispositivo analogico e o canal
AI=analoginput('winsound',0);
chan_in = addchannel(AI,1);

% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = SampleRate*duration;

% Ajusta as propriedades do canal
set(AI,'SampleRate',SampleRate);
set(AI,'SamplesPerTrigger',10*NumSamples);
set(AI,'TriggerType','Manual');

% Entrada
start(AI);
trigger(AI);

comecou = 0;
inicio = 1;
for ii = 1 : 10*NumSamples,

    % Se chegou ao fim do sinal
    if ((ii == (inicio + NumSamples + 1)) && comecou == 1)
        break;
    end

    recebe(ii, 1) = getdata(AI, 1);

    % Demodulaçao 8-PSK
    % Valores ajustados de acordo com os volumes das
    % placas de som dos dois computadores
    if (real(recebe(ii,1) >= 0.8))
        sinal_recebido(ii,1) = 0;
    elseif (real(recebe(ii,1) >= 0.5)) && (real(recebe(ii,1) < 0.8))
        sinal_recebido(ii,1) = 1;
    elseif (real(recebe(ii,1) >= 0.)) && (real(recebe(ii,1) < 0.5))
        sinal_recebido(ii,1) = 2;
    elseif (real(recebe(ii,1) >= 0)) && (real(recebe(ii,1) < 0.2))
        sinal_recebido(ii,1) = 3;
    elseif (real(recebe(ii,1) >= -0.55)) && (real(recebe(ii,1) < 0))
```

```
        sinal_recebido(ii,1) = 4;
    elseif (real(recebe(ii,1) >= -0.65)) && (real(recebe(ii,1) < -0.55))
        sinal_recebido(ii,1) = 5;
    elseif (real(recebe(ii,1) >= -0.87)) && (real(recebe(ii,1) < -0.65))
        sinal_recebido(ii,1) = 6;
    else
        sinal_recebido(ii,1) = 7;
    end

    if ((ii > 7) && (comecou == 0)),
        if (sinal_recebido((ii - 7):ii, 1) == (M-1)*[1 1 0 0 1 1 0 0]'),
            inicio = ii % Marca onde começou o sinal
            comecou = 1; % Indica que a recepção do sinal começou
        end
    end
end

% Sinal recebido
data_in = sinal_recebido((inicio + 1):(NumSamples + inicio), 1);

%Apaga o conteudo e fecha o canal
waittilstop(AI, 10*duration + 1);
delete(AI);
clear AI;

% Sinal enviado
for a=1:(NumSamples/2),
    if (mod(a,M) == 1),
        z((2*a-1):2*a,1) = zeros(2,1);
    elseif(mod(a,M) == 2),
        z((2*a-1):2*a,1) = ones(2,1);
    elseif(mod(a,M) == 3),
        z((2*a-1):2*a,1) = 2*ones(2,1);
    elseif(mod(a,M) == 4),
        z((2*a-1):2*a,1) = 3*ones(2,1);
    elseif(mod(a,M) == 5),
        z((2*a-1):2*a,1) = 4*ones(2,1);
    elseif(mod(a,M) == 6),
        z((2*a-1):2*a,1) = 5*ones(2,1);
    elseif(mod(a,M) == 7),
        z((2*a-1):2*a,1) = 6*ones(2,1);
    else
        z((2*a-1):2*a,1) = 7*ones(2,1);
    end
end

% Compara os dois sinais para detectar erros
num_erro = 0;
for i = 1: NumSamples,
    if data_in(i) ~= z(i),
        num_erro = num_erro + 1;
    end
end

% Exibe o erro percentual
disp('Erro percentual');
100*num_erro/NumSamples

plot(data_in);
title('Sinal recebido demodulado');
```

transmissor_bpsk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a transmissão de um sinal pela placa de
% som do computador usando modulação 2-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 2; % BPSK

% Abre o dispositivo analogico e o canal
AO = analogoutput('winsound',0);
chan_out = addchannel(AO,1);

% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = 8 + SampleRate*duration;

% Ajusta as propriedades do canal
set(AO,'SampleRate',SampleRate)
set(AO,'TriggerType','Manual')

% Sinal que vai ser modulado e transmitido
% Onda quadrada
for a=1:(NumSamples/4),
    if (mod(a,M) == 1),
        x((4*a-3):4*a),1) = ones(4,1);
    else
        x((4*a-3):4*a),1) = zeros(4,1);
    end
end

% Sinal enviado
% Sequencia que indica inicio de transmissao: 1 1 0 0 1 1 0 0
sinal = [1 1 0 0 1 1 0 0 x']';

% Modulacao BPSK
data_out = (-1).^sinal;

% Envia os dados
putdata(AO,data_out)
start(AO)
trigger(AO)

% Apaga o conteudo e fecha o canal
waittilstop(AO, 2*duration)
delete(AO)
clear AO

plot(sinal);
```

```
title('Sinal transmitido antes da modulação');
axis([0 NumSamples 0 1]);

scatterplot(data_out);
title('Sinal modulado em BPSK');
```

transmissor_4psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a transmissão de um sinal pela placa de
% som do computador usando modulação 4-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 4; % 4-PSK

% Abre o dispositivo analogico e o canal
AO = analogoutput('winsound',0);
chan_out = addchannel(AO,1);

% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = 8 + SampleRate*duration;

% Ajusta as propriedades do canal
set(AO,'SampleRate',SampleRate)
set(AO,'TriggerType','Manual')

% Sinal que vai ser modulado e transmitido
for a=1:(NumSamples/4-2),
    if (mod(a,M) == 1),
        x(((4*a-3):4*a),1) = zeros(4,1);
    elseif(mod(a,M) == 2),
        x(((4*a-3):4*a),1) = ones(4,1);
    elseif(mod(a,M) == 3),
        x(((4*a-3):4*a),1) = 2*ones(4,1);
    else
        x(((4*a-3):4*a),1) = 3*ones(4,1);
    end
end

% Sinal enviado
% Sequencia que indica inicio de transmissao: 3 3 0 0 3 3 0 0
sinal = [3 3 0 0 3 3 0 0 x']';

% Modulação 4-PSK
% Valores ajustados de acordo com os volumes das
% placas de som dos dois computadores
for a = 1:NumSamples,
```

```
    if sinal(a) == 0,
        data_out(a) = 1;
    elseif sinal(a) == 1,
        data_out(a) = 0.3;
    elseif sinal(a) == 2,
        data_out(a) = -0.3;
    else
        data_out(a) = -1;
    end
end

% Envia os dados
putdata(AO, data_out')
start(AO)
trigger(AO)

% Apaga o conteudo e fecha o canal
waittilstop(AO, 2*duration)
delete(AO)
clear AO

plot(sinal);
title('Sinal transmitido antes da modulação');
axis([0 NumSamples 0 3]);

scatterplot(data_out);
title('Sinal modulado em 4-PSK');
```

transmissor_8psk.m

```
% UFCG/CEEI/DEE
% Johannes Dantas de M. Jr.
% Novembro de 2008
%
% Realiza a transmissão de um sinal pela placa de
% som do computador usando modulação 8-PSK
%
% Código adaptado do texto:
% 'Using the MATLAB Data Acquisition Toolbox', by Brian D. Storey
%
clc;
clear all;
close all;

M = 8; % 8-PSK

% Abre o dispositivo analogico e o canal
AO = analogoutput('winsound',0);
chan_out = addchannel(AO,1);

% Ajusta a taxa de amostragem e a duraçao
duration = 1;
SampleRate = 8000;
NumSamples = 8 + SampleRate*duration;

% Ajusta as propriedades do canal
set(AO,'SampleRate',SampleRate)
set(AO,'TriggerType','Manual')
```

```
% Sinal que vai ser modulado e transmitido
for a=1:(NumSamples/2),
    if (mod(a,M) == 1),
        x((2*a-1):2*a),1) = zeros(2,1);
    elseif(mod(a,M) == 2),
        x((2*a-1):2*a),1) = ones(2,1);
    elseif(mod(a,M) == 3),
        x((2*a-1):2*a),1) = 2*ones(2,1);
    elseif(mod(a,M) == 4),
        x((2*a-1):2*a),1) = 3*ones(2,1);
    elseif(mod(a,M) == 5),
        x((2*a-1):2*a),1) = 4*ones(2,1);
    elseif(mod(a,M) == 6),
        x((2*a-1):2*a),1) = 5*ones(2,1);
    elseif(mod(a,M) == 7),
        x((2*a-1):2*a),1) = 6*ones(2,1);
    else
        x((2*a-1):2*a),1) = 7*ones(2,1);
    end
end

%Sinal enviado, seqüência que indica inicio de transmissao: 7 7 0 0 7 7 0 0
sinal = [7 7 0 0 7 7 0 0 x']';

% Modulação 8-PSK
% Valores ajustados de acordo com os volumes das
% placas de som dos dois computadores
for a = 1:NumSamples,
    if sinal(a) == 0,
        data_out(a) = 1;
    elseif sinal(a) == 1,
        data_out(a) = 0.75;
    elseif sinal(a) == 2,
        data_out(a) = 0.5;
    elseif sinal(a) == 3,
        data_out(a) = 0.25;
    elseif sinal(a) == 4,
        data_out(a) = -0.2;
    elseif sinal(a) == 5,
        data_out(a) = -0.4;
    elseif sinal(a) == 6,
        data_out(a) = -0.65;
    else
        data_out(a) = -1;
    end
end
 % Envia os dados
putdata(AO, data_out')
start(AO)
trigger(AO)

% Apaga o conteudo e fecha o canal
waittilstop(AO, 2*duration)
delete(AO)
clear AO

plot(sinal);
title('Sinal transmitido antes da modulação'); axis([0 NumSamples 0 7]);

scatterplot(data_out); title('Sinal modulado em 8-PSK');
```