Segundo Projeto de Comunicações Móveis

Vítor Gabriel Lemos Lopes

[1]Departamento de Engenharia de Comunicações-UFRN

Resumo. *Este Trabalho foi feito com intuito de aprendizado sobre o uso de OFDM (Orthogonal Frequency Division Multiplexing) em canais AWGN(Additive White Gaussian Noise) para modulações BPSK,64QAM e 256QAM. Para isso foi utilizado a linguagem de programação em Python para fazer a FFT(Fast Fourier Transform) e a iFFT(inverse Fast Fourier Transform), a modelagem do canal e a probabilidade de erro de cada modulação, com isso foram feitos gráficos para comparação do simulado com a probabilidade de erro teórica.*

1. Introdução

Como sabemos a transmissão sem fio dois principais tipos de atenuação, Desvanecimentos em larga escala e o desvanecimento em pequena escala. Este projeto tem como objetivo estudar o desvanecimento em pequena escala em um sistema de comunicação digital sem fio.

Para melhorar a comunicação sem fio e para diminuir o efeito do desvanecimento em pequena escala, multi percurso é um deles,utilizamos o OFDM que é uma técnica de transmissão de dados que utiliza sua banda dividida em múltiplas portadoras ortogonais, chamadas subportadoras, para modulação[Weiss and Jondral 2004]. Neste trabalho vamos utilizar um sinal BPSK, 64QAM e 256QAM e utilizaremos das técnicas do OFDM em um canal AWGN e vamos ver como se comporta e comparar a sua probabilidade de erro teórica e simulada com OFDM.

2. Formulação matemática

Foi pedido que fossem feitas duas curvas para cada modulação, uma para BER vs Eb/No para com OFDM e, no mesmo gráfico, o gráfico da Pe vs Eb/No (fórmula teórica) da modulação BPSK, 64-QAM e 256-QAM sem OFDM com apenas ruído AWGN. Com isto foi criado bits aleatórios,depois foi modulado, depois o sinal passa pela iFFT, acrescentando o ruido AWGN, depois o sinal passa pela FFT e é feita a demodulação e contada a quantidade de bits errados, com isso vamos ter a probabilidade de erro de bits. Mas para calcular o AWGN foi feito uma variável aleatória gaussiana com média 0 e variância $N0/2$ [Haykin 2001], o qual $N0$ é:

$$N0 = Eb * 10^{-SNR_{dB}/10} \tag{1}$$

Os quais:

- **N0**:Densidade espectral do ruído.
- **Eb**: Energia de bit dado no problema.
- **SNR**: Relação sinal ruído em dB dado no problema.

Mas nós não temos o valor da Energia de bit, para isso vamos usar a PSD(Power Spectral Density) que é a energia de bit média do sinal [Lathi 2009]:

$$PSD = \frac{\sum sinal * conjugado(sinal)}{tamanhodosinal} \tag{2}$$

2.1. Probabilidade de erro de bits

Para um canal AWGN a probabilidade de erro teórica de um BPSK [Lathi 2009] é:

$$Pe = Q\left(\sqrt{\frac{2Eb}{N0}}\right)$$

(3)

E para modulações M-QAM para quando $log_2(M)$ é par [Lathi 2009]:

$$Pe = \frac{4}{log_2(M)} * (1 - \frac{1}{\sqrt{M}}) * Q\left(\sqrt{\frac{3 * log_2(M) * Eb/No}{M-1}}\right)$$

(4)

2.2. Códigos e Gráficos

Agora com ajuda da biblioteca Numpy podemos criar os bits aleatórios para a simulação fazendo:

```
bit=np.random.randint(0,2,size=b)
```

Em seguida modulamos com a ajuda da biblioteca Commpy,para M-QAM usamos QAMModem e para BPSK o PSKModem:

```
qam=commpy.QAMModem(M)                  #Fazendo mapeamento para a
    modula   o MQAM
sinal=qam.modulate(bit)                 #Fazendo a modula   o
psk=commpy.PSKModem(M)                  #Fazendo mapeamento para a
    modula   o MPSK
sinal=psk.modulate(bit)                 #Fazendo a modula   o
```

Para continuar temos que fazer todos os símbolos ficarem normalizados para que a distância entre os pontos mais distantes seja a mesma fazendo a PSD e dividindo pelo sinal de símbolos, após isso escolhemos uma janela de transmissão do OFDM, neste trabalho escolhemos 40 aleatoriamente e por ser divisor da quantidade da quantidade de bits que colocamos, passamos pela IFFT,e para adicionamos o AWGN, precisamos calcular a Energia bit novamente pela PSD dessa vez com o sinal no tempo e usarmos a formula 2 para calcular a variância do ruído,depois de adicionarmos o ruído e passamos pela FFT demodulamos usando a função:

```
resultbit=qam.demodulate(sinalfreq,'hard')
resultbit=psk.demodulate(sinalfreq,'hard')
```

Que já define o limiar de decisão e já dá a resposta em bits.Para definirmos os bits errados, é pego os bits recebidos e é comparado com os bits originais:

```
bit_error=np.where(bit!=resultbit,1,0)
ber.append((sum(bit_error))/b)
```

Com isso já temos a BER para cada valor de SNR, e podemos plotar o gráfico.

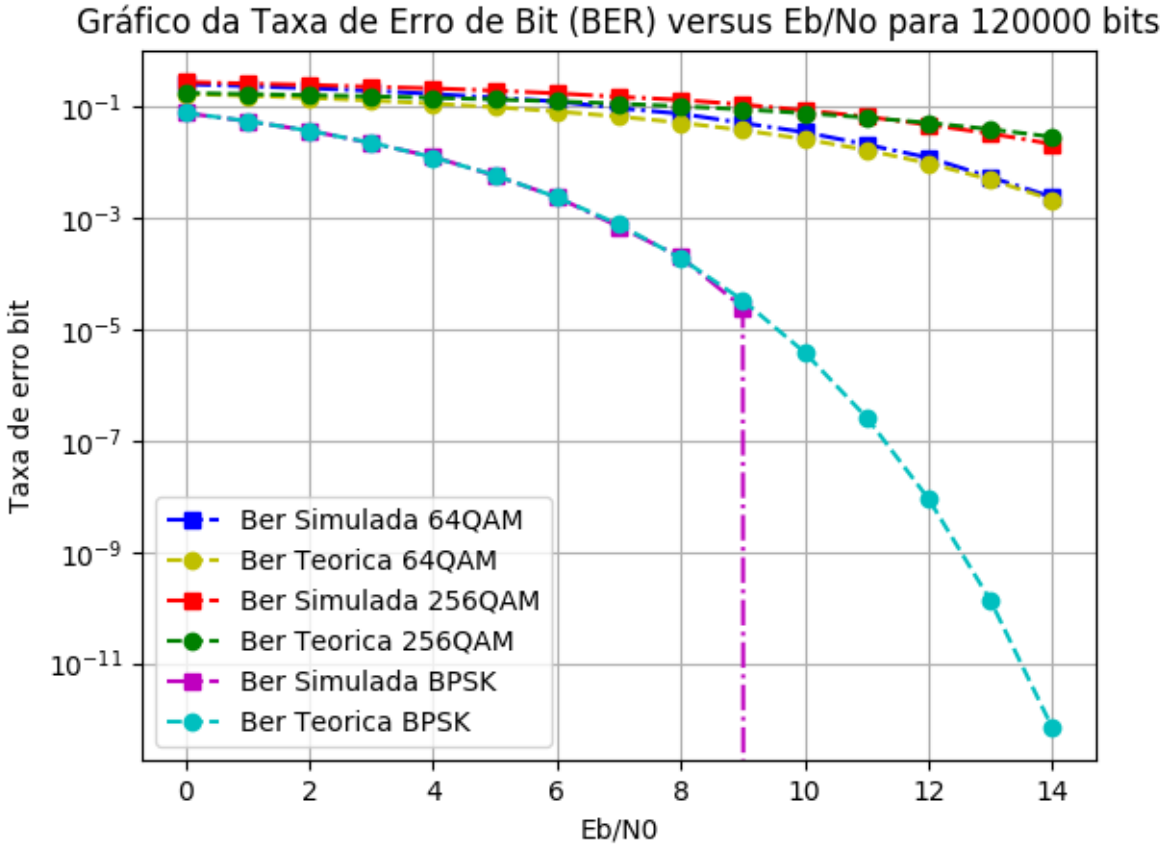

Figura 1. Constelação de bits

Com este gráfico podemos observar que não há diferença entre o sinal com ou sem OFDM(teórico), e quanto maior o numero de bits por simbolo (bits por sim = $\log_2(M)$), maior será a probabilidade de erro de bits. Foi escolhido essa quantidade de bits para a transmissão por causa do processamento computacional que causa colocar uma maior quantidade de bits. Podemos perceber também que com aumento do Eb/N0 a probabilidade de erro de bits diminui, isso é devido a que os bits ficam menos espalhados e mais próximos do sinal original assim acontecendo uma menor quantidade de erros de bits. Também é possível perceber no gráfico que a quantidade de erro de cai pra zero, isso significa que não há bits suficientes para poder estimar a BER até os 15 dB de Eb/N0.

3. References

[Proakis 2011]

Referências

Haykin, S. (2001). *Sistemas de comunicação- analógicos e digitais*. 4th edition.

Lathi, B. P. (2009). *Modern Digital and Analog Communication Systems*. Oxford University Press, Inc., New York, NY, USA, 4nd edition.

Proakis, J. G. (2011). *Contemporary Communication Systems Using MATLAB®*.

Weiss, T. and Jondral, F. (2004). Spectrum pooling: an innovative strategy for the enhancement of spectrum efficiency. *IEEE Communications Magazine*, 42:S8–14.