Engenharia Eletrotécnica e de Computadores

Telecomunicações

2º Trabalho Laboratorial

Análise de sistemas BPSK e QPSK:

Impacto do AWGN na taxa de erro de bit

Autores:
Daniel Dinis : 99906
Diogo Costa : 99919
João Gonçalves : 99995

Supervisora:
Profª. Stefânia de Sousa Faria

Novembro 2022

Índice

Introdução e contextualização

1. Modulação PSK

Na modulação por deslocamento de fase (*phase-shift keying*) **os bits dos dados digitais são codificados numa forma analógica, ao modificar a fase da portadora sinusoidal**. Desta forma, múltiplos bits podem ser atríbuidos à onda portadora.

A modulação PSK é definida pela seguinte equação:

$$s_m(t) = p(t)\cos\left(2\pi f_c t + \theta_m\right) \qquad m = 1, \ldots, M$$

Onde θ_m é o deslocamento de fase e $p(t)$ o *pulse shaping signal*. O tipo mais comum de PSK é o *binary phase-shift keying* (BPSK), em que um bit singular é atribuído à portadora. Em *quadrature phase-shift keying* (QPSK), são mapeados dois bits em cada fase da portadora.

As modulações BPSK e QPSK são casos específicos da modulação PSK M-ária (M-PSK), com $M = 2$ e $M = 4$ respetivamente[1]. Os valores de fase da portadora são genericamente descritos no seguinte conjunto:

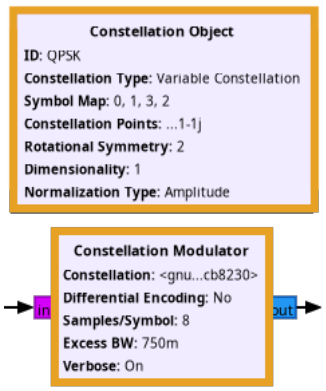
$$\left\{\frac{2\pi}{M}(m-1) + \Phi\right\}_{m=1}^{M}$$

em que Φ é uma constante de fase[2].

No âmbito laboratorial, são observadas as seguintes fases:

$$\theta_m^{\text{BPSK}} = \begin{cases} 0, & \text{bit } 1 \\ \pi, & \text{bit } 0 \end{cases} \quad \wedge \quad \theta_m^{\text{QPSK}} = \begin{cases} \pi/4, & \text{dibit } 11 \\ 3\pi/4, & \text{dibit } 01 \\ 5\pi/4, & \text{dibit } 00 \\ 7\pi/4, & \text{dibit } 10 \end{cases}$$

1.1 *Constellation Modulator*



No âmbito da modulação em ambiente *GNU Radio*:

"(...) The input is a byte stream (unsigned char) and the output is the complex modulated signal at baseband."[1]

Pelo que o *output* (que representa o sinal de banda-passante teórico em banda-base) proveniente do bloco *Constellation Modulator* toma o seguinte formato[3]:

$$z(t) = I(t) + jQ(t)$$

---

[1] Onde $M \triangleq$ número possível de fases da portadora $= 2^{\text{nº. bits}}$

[2] Atendendo aos mapeamentos no Guia de Laboratório, tem-se que $\Phi \equiv 0$ para BPSK e $\Phi \equiv \pi/4$ para QPSK.

[3] "Complex baseband equivalent representation of the real [passband] signal."[2]

Pelo que

$$\sqrt{E_b} = \sqrt{\varepsilon} \triangleq \sqrt{I(t)^2 + Q(t)^2} \quad \wedge \quad \theta(t) \triangleq tan^{-1}\left(\frac{Q(t)}{I(t)}\right)$$

Deste modo, a análise do sinal é passível de uma abordagem vetorial, como explicitado nos **diagramas de constelação** a baixo:

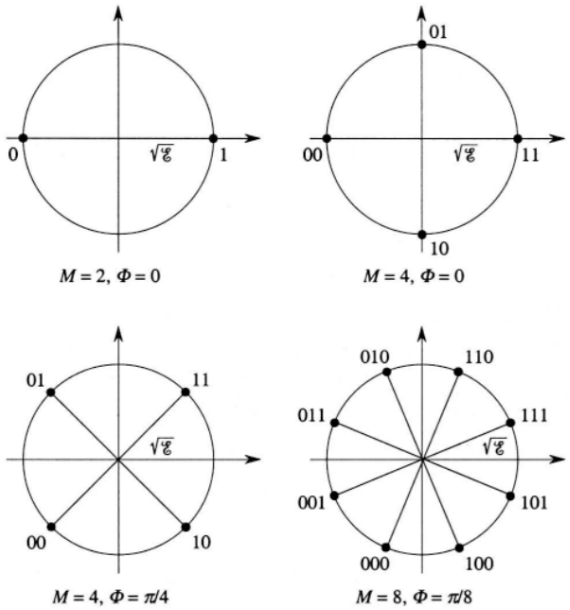


Fig. 1: Representação geométrica de conjuntos de sinais PSK *Gray-mapped*[3].

O mapeamento dos símbolos para a representação em *complex baseband* é garantido no bloco *Constellation Object*. Atendendo ao parâmetro *Variable Constellation*, para um ajuste fino, revela-se:

Forma geral: *Symbol Map* $\mapsto$ *Constellation Points*

⋆ **BPSK:** $[0, 1] \mapsto [-1, 1]$

⋆ **QPSK:** $[0, 1, 3, 2] \mapsto [\underbrace{-1 - 1j}_{00}, \underbrace{-1 + 1j}_{01}, \underbrace{1 + 1j}_{11}, \underbrace{1 - 1j}_{10}]$

2. Canal de transmissão e filtro adaptado

Numa situação ideal (i.e., na ausência de ruído branco), a desmodulação de um sinal digital requer a utilização de um filtro passa-baixo regular. Não obstante, em aplicações reais é necessário mitigar os efeitos da presença do ruído branco (térmico), cuja densidade espectral de potência é constante ($S_W(f) = N_0/2$). Esta característica torna o AWGN impossível de eliminar completamente.

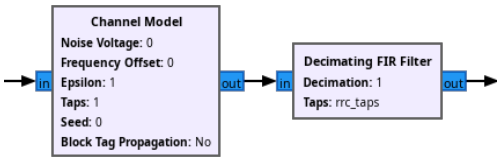


Fig. 2: Modelo do canal de transmissão e filtro adaptado (*GNU Radio*).

Nota → Em regime laboratorial, o canal de transmissão é simulado através do bloco *Channel Model*, onde é variado o *"AWGN noise level as a voltage"*[4] de 0 a 4 V.

Para atenuar o impacto deste ruído, utiliza-se um filtro adaptado que maximiza a relação sinal-ruído nos instantes de amostragem.

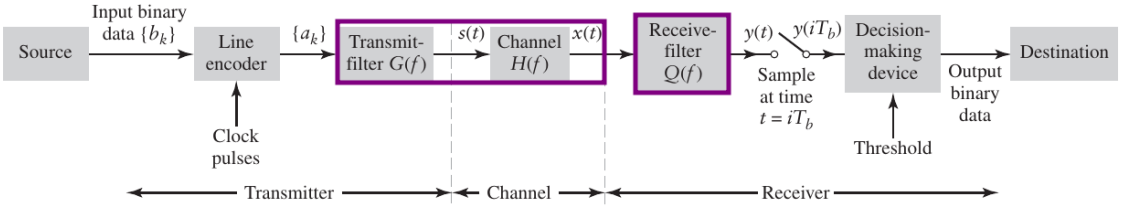


Fig. 3: Trasmissão e recepção com filtro adaptado.

Tomando $G(f)H(f) \triangleq P^{1/2}(f)$, temos que o filtro adaptado toma a seguinte forma: $Q(f) \triangleq P^{1/2}(f)$.[4]

2.1 *Pulse-shaping*

> "The most popular pulse-shaping filter seems to be the "raised-cosine" filter. It's a good low-pass filter for limiting the bandwidth our signal will occupy, and it also has the property of summing to zero at intervals of T_S [eliminar interferência intersimbólica (IIS)] (...)"[5]

Mediante o supracitado, e tendo em consideração o formato do sinal $y(t)$ (vide a Fig. 3), é trivialmente deduzido que para obter o formato *raised cosine* e executar o processo de *matched filtering* no ramo de receção (tal como no projeto corrente), recorre-se a um *root raised cosine* (como apresentado na Fig. 2).

[4]Sendo o sistema caracterizado por $p(t) = g(t) * h(t) * q(t) \xrightarrow{\mathcal{F}} P(f) = G(f)H(f)Q(f)$, obtém-se o sinal PAM: $y(t) = \sum_{k=-\infty}^{\infty} a_k p(t - kT_b)$.

3. Desmodulação

No que consta à desmodulação, o processo é efetuado em dois instantes: sincronização de símbolo (efetuada no bloco *Symbol Sync*) e reconversão para stream de dados binários (*Binary Slicer*).

3.1 *Symbol Sync*

> "Symbol timing synchronization has a unique purpose: to find the optimal instants when downsampling a sequence of samples into a series of symbols. In other words, it focuses on selecting the "best" sample out of every group of samples, such that this selected sample can better represent the transmitted symbol. The chosen sample (deemed as the symbol) is then passed on to the symbol detector."[6]

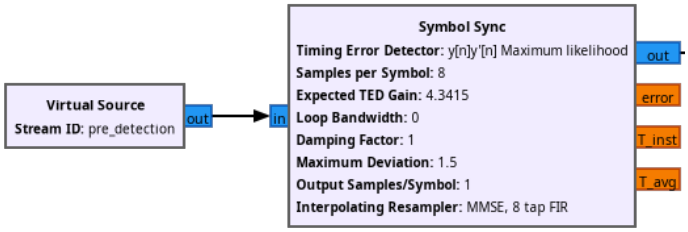


Fig. 4: Bloco que realiza a sincronização de símbolo.

Concentramos a nossa atenção sobre os seguintes parâmetros:

Loop Bandwidth:
"It should nominally be close to 0, but greater than 0. If unsure, start with a number around 2*pi*0.04, and experiment to find the value that works best for your situation."[7]

TED Gain:
"This value is normally computed by the user analytically or by simulation in a tool outside of GNURadio. This value must be correct for the loop filter gains to be computed properly from the desired input loop bandwidth (...)"[7]

3.1.1 *TED (Timing Error Detector)*

Tal como supracitado, o bloco *Symbol Sync* requer a computação do *TED Gain* fora do ambiente *GNU Radio* para um correto funcionamento face à *Loop Bandwidth*.

> "(...) The timing offset error τ results from the channel propagation delay, which can not be controlled and, therefore, introduces delays that are not simply integer multiples of the symbol period. In reality, the propagation delay is such that τ is composed of two terms: an integer and a fractional multiple of T_s. In the context of symbol timing recovery, we are only concerned with the fractional error. The integer error is handled by a frame timing recovery (or frame synchronization) scheme.
>
> (...) The timing error detector has the purpose of estimating this timing error τ, so that the receiver can adjust its timing and avoid the intersymbol interference."[6]

Este breve reparo relativamente ao *Timing Error Detector* salienta a proeminência da correta parametrização deste campo.

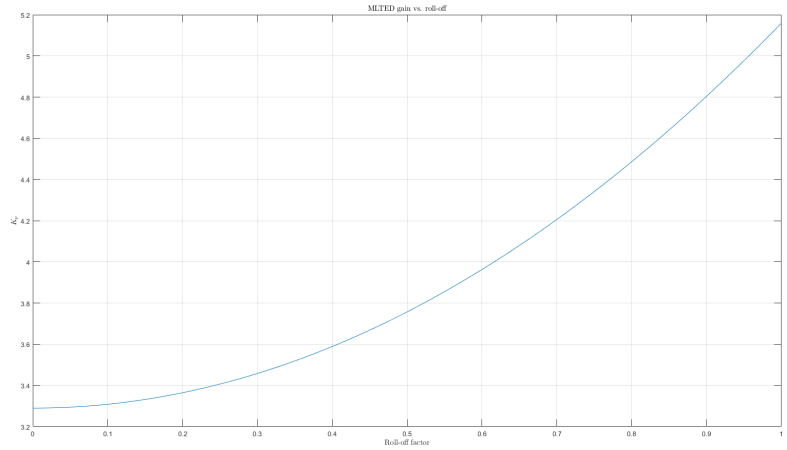


Fig. 5: Evolução do *TED Gain* face ao aumento do *roll-off factor* (MLTED)[6].

O cálculo do ganho é dependente do *roll-off factor* do *root raised cosine* e do esquema do *Timing Error Detector* utilizado (*Maximum Likelihood*). Tal evolução é aparente na Fig. 5. Para o *roll-off factor* de 0.75 especificado no Guia de Laboratório, o valor ótimo calculado[5] para o *TED Gain* verificou-se 4.3415. A aplicação deste fator é aparente na Fig. 6.

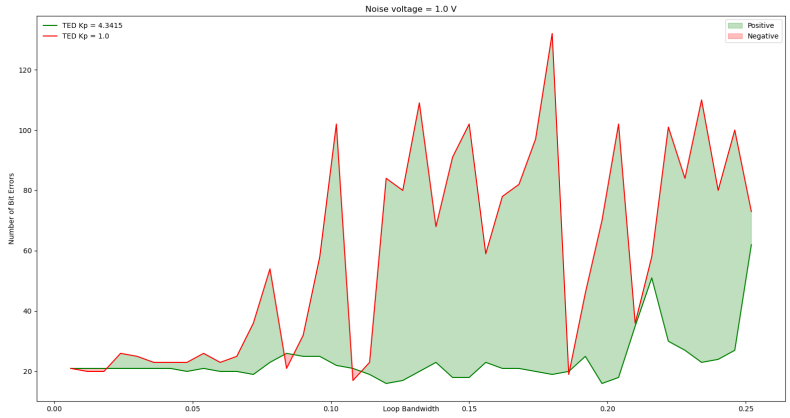


Fig. 6: Efeitos da aplicação do *TED Gain* calculado para uma *noise voltage* de 1.0 V.

A maior estabilidade observada para o *range* de valores de *Loop Bandwidth* relevantes para a análise[7], é também acompanhada por uma desejável mitigação aparente de erros de bit[6]. Em virtude das ilações expostas, as posteriores simulações são, naturalmente, **realizadas com um *TED Gain* de 4.3415**.

[5]Através do cálculo analítico da constante, com o recurso ao *script* em MATLAB disponibilizado em https://github.com/igorauad/symbol_timing_sync.

[6]Reparos especialmente prevalentes para *Loop Bandwidths* nominalmente perto de zero.

3.1.2 *Loop Bandwidth*

A função do *Loop Filter* encontrado na composição do bloco *Symbol Sync* (vide a documentação oficial) pode ser sucintamente descrita pela seguinte citação:

> "**The loop filter controls how fast the timing error can be corrected, what types of errors can be treated, and the range of correctable timing errors.** In general, it is a second-order system and often the proportional-plus-integral controller (PI controller), commonly used in feedback systems. (...)"[6]

O funcionamento deste integrante do processo de sincronização de símbolo é profundamente caracterizado pela sua *Loop Bandwidth*, que:

> "(...) sets the boundary at which your recovery loop will track timing noise (jitter/phase noise) in your signal. **Timing noise has components at all frequencies** (...), and the lower frequency components of the timing noise (...) will be tracked by the timing loop, and therefore suppressed. However the higher frequency components will not be tracked by the timing loop and therefore remain and contribute to noise as part of SNR. **The timing loop can be viewed as a high pass filter on the jitter/phase noise components**, passing the higher frequency components (including the modulation signals) and supressing the lower frequencies at a rate depending on the order of the loop. Thus you see the motivation to make the timing loop as fast as possible in the interest of tracking out as much timing jitter as possible. **However, some of the signal energy of interest will be part of the "noise" being supressed by the timing loop**; and this can be quantified by understanding the spectral characteristics of the modulation used. **As the loop BW increases, too much signal energy is also removed**, and thus it is clear how there can be an optimum setting for the Loop BW that maximizes SNR."[8]

Procuramos então um valor de *Loop Bandwidth* que não reduza (de forma comprometedora) a relação sinal-ruído, mas que suprima as componentes de baixa e alta frequência do *timing noise*.

Observação → Em regime laboratorial, verifica-se geralmente o supracitado: o número de erros de bit tende[7] a aumentar consoante o incremento após certos *thresholds* da *Loop Bandwidth*, para cada patamar de ruído.

3.2 Reconversão para stream de dados binários

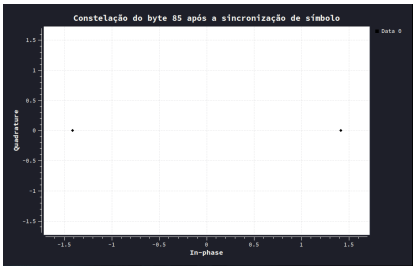
Encontrado o intervalo ótimo de amostragem, ao qual corresponde o valor máximo do pulso, segue-se a conversão para *stream* de dados binários, para a recuperação da sequência de *bits* transmitida.

[7] Relação aparente na análise posterior dos dois sistemas.

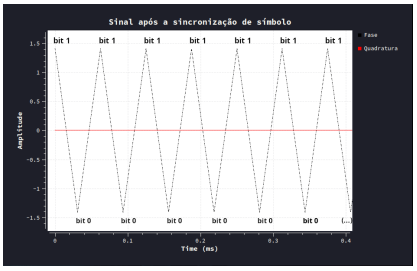
3.2.1 BPSK

Para clarificar o processo corrente, é relevante apresentar um exemplo prático.

Observando a constelação do *byte* 85 (0 1 0 1 0 1 0 1) repetido *ad eternum*, na ausência dos efeitos do AWGN:



(a) Constelação do *byte* 85 após a sincronização de símbolo.



(b) Sinal correspondente ao *byte* 85 após a sincronização de símbolo (sem presença de AWGN).

Fig. 7: Representações do *byte* 85 numa modulação BPSK (sem normalização[8]).

é aparente o mapeamento explicitado na antecedente secção 1.1:

$$bit\ 1 \mapsto [+1{+}0j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = +\sqrt{2} \\ \text{coef}_{\mathbb{I}\text{m}} = 0 \end{cases} \qquad bit\ 0 \mapsto [-1{+}0j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = -\sqrt{2} \\ \text{coef}_{\mathbb{I}\text{m}} = 0 \end{cases}$$

Após a sincronização de símbolo (vide a Fig. 7 (b)), é apenas necessário recorrer a um bloco decisor para a conversão em *bits* (*Binary Slicer* → "Positive input produces a binary 1 and negative input produces a binary zero."[9]) e efetuar um posterior agrupamento em *bytes* (*Pack K bits*). Tal como apresentado abaixo:

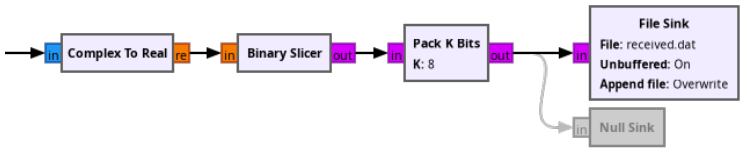


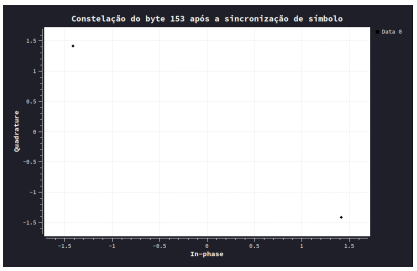
Fig. 8: Etapa final do processo de desmodulação BPSK em ambiente *GNU Radio*.

Observação → Em regime laboratorial, verifica-se um *offset* de 49 *bits* no ficheiro 'received.dat' (ficheiro de saída, vide a Fig. 8), tal como explicitado no Guia de Laboratório: "(...) De notar que no ficheiro 'bpsk_rec.dat', os bits correspondentes aos bits enviados aparecem a partir da 50ª posição sendo precedidos de 49 bits correspondentes ao buffer interno do módulo 'Symbol Sync'"[10].

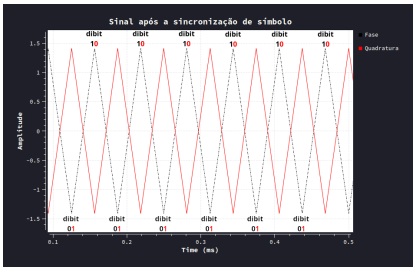
[8]Devido a incongruências observadas em diversas plataformas (Windows, GNU/Linux) da opção *Normalization Type* do bloco *Constellation Object* [11] em ambiente *GNU Radio*, **toda a análise subsequente é realizada sem qualquer tipo de normalização** (i.e., $\sqrt{\varepsilon} = \sqrt{2} \neq 1$), de modo a garantir a comparação dos sistemas BPSK e QPSK.

3.2.2 QPSK

De forma análoga, apresenta-se o exemplo do *byte* 153 (10 01 10 01), em repetição, que nos permite observar a sincronia ortogonal das componentes em fase e em quadratura (característica da modulação QPSK[9]).



(a) Constelação do *byte* 153 após a sincronização de símbolo.



(b) Sinal correspondente ao *byte* 153 após a sincronização de símbolo (sem presença de AWGN).

Fig. 9: Representações do *byte* 153 numa modulação QPSK (sem normalização).

Ainda assim, graças à particularidade referida acima (e na anterior secção 1.1), um *byte* seccionado em *dibits* pode ser mapeado (de forma ótima, sem AWGN) para as quatro regiões do diagrama de constelação, como se apresenta:

$$\underset{11}{dibit} \mapsto [+1+1j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = +\sqrt{2}/2 \\ \text{coef}_{\mathbb{I}\text{m}} = +\sqrt{2}/2 \end{cases} \qquad \underset{01}{dibit} \mapsto [-1+1j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = -\sqrt{2}/2 \\ \text{coef}_{\mathbb{I}\text{m}} = +\sqrt{2}/2 \end{cases}$$

$$\underset{00}{dibit} \mapsto [-1-1j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = -\sqrt{2}/2 \\ \text{coef}_{\mathbb{I}\text{m}} = -\sqrt{2}/2 \end{cases} \qquad \underset{10}{dibit} \mapsto [+1-1j] \rightarrow \begin{cases} \text{coef}_{\mathbb{R}\text{e}} = +\sqrt{2}/2 \\ \text{coef}_{\mathbb{I}\text{m}} = -\sqrt{2}/2 \end{cases}$$

Novamente, após este processo de deteção, segue-se a conversão para *stream* de dados binários, empacotados posteriormente em *bytes*. Este processo assemelha-se ao do BPSK, com a exceção da intercalação das componentes em fase (*bit* mais significativo do *dibit*) e em quadratura (*bit* menos significativo do *dibit*).

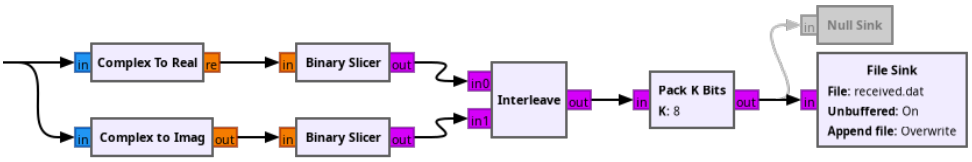


Fig. 10: Etapa final do processo de desmodulação QPSK em ambiente *GNU Radio*.

Nota → Verifica-se uma duplicação do *offset* (proveniente do *buffer* do *Symbol Sync*) incutido no ficheiro de saída, (fenómeno esperado) visto que decorrem duas sincronizações em paralelo[9] posteriormente intercaladas.

[9] "QPSK can be regarded as a pair of orthogonal BPSK systems, i.e. the real component is one BPSK system, the imaginary component is the second BPSK system (...)"[12]

4. Probabilidade de erro do bit

Antes de proceder para as etapas de tratamento de dados subsequentes é fundamental analisar a métrica "probabilidade de erro de bit", P_e, com vista a fornecer uma preâmbulo à exposição e análise de resultados. Invocando o Teorema da Probabilidade Total e supondo a equiprobabilidade dos bits '1' e '0', a probabilidade de erro será dada por:

$$P_e = P(\text{erro} \mid \text{bit } 1)P(\text{bit } 1) \ + \ P(\text{erro} \mid \text{bit } 0)P(\text{bit } 0)$$
$$P_e = \frac{P(\text{erro} \mid \text{bit } 1) \ + \ P(\text{erro} \mid \text{bit } 0)}{2}$$

4.1 Probabilidade de erro de bit para um sinal binário polar

Reconhecendo que o sinal binário polar[10] (onde se encontram os sinais das modulações em análise) pode ser expresso procedimentalmente sobre a forma $y(t) = \sum_{n=-\infty}^{\infty} a_n p(t-nT)$ de amplitudes $a_n = \pm\text{A}$, após a sincronização de símbolo e encontro do instante de amostragem ideal T_0 temos:

$$p_k = a_n \cdot p_0 = \begin{cases} +\text{A} \cdot p(T_0), & \text{bit } 1 \\ -\text{A} \cdot p(T_0), & \text{bit } 0 \end{cases}$$

em que $p(t)$ denomina a resposta do sistema transmissão-receção (vide a secção 2.).

Podemos então determinar as probabilidades de erro condicionadas:

$$\begin{cases} P(\text{erro} \mid \text{bit } 1) = P(\text{erro} \mid a_n = +A) = P(n_k < -A \cdot p(T_0)) \\ P(\text{erro} \mid \text{bit } 0) = P(\text{erro} \mid a_n = -A) = P(n_k > +A \cdot p(T_0)) \end{cases} \rightarrow Q\left(\frac{A \cdot p(T_0)}{\sigma_n}\right)$$

E subsequentemente concluir que a probabilidade erro de um sinal binário polar é:

$$P_e = Q\left(\frac{A \cdot p(T_0)}{\sigma_n}\right)$$

em que σ_n é o desvio padrão do ruído no instante de amostragem. Reescrevendo a probabilidade em função da energia de bit obtemos:

$$P_e = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

que é válida para ambas as modulações em análise[11].

Nota $\rightarrow$ Embora coerente para a probabilidade de erro de bit, o mesmo não acontece para a probabilidade de erro de símbolo: $P_{\text{es}} \approx \log_2(M)P_e$, admitindo um M = 4 para QPSK e M = 2 para BPSK (M-PSK), observamos que:

$$P_{es\_\text{QPSK}} \approx 2P_{es\_\text{BPSK}}$$

[10] Sinal à saída do filtro adaptado, vide secção 2.

[11] "QPSK can be regarded as a pair of orthogonal BPSK systems (...) Because they are orthogonal, they don't interfere (to a good approximation), hence the BER curves are largely equivalent."[12]

Exposição de dados

E.1 Método de obtenção de dados

A extração de dados foi realizada primariamente com recurso a dois *scripts* desenvolvidos em *Python*:

Source code: *Script* responsável pela comparação de ficheiros (bit_error.py)

```
#!/bin/python3
def bit_error(B):
    file_s = "sent.dat";
    file_r = "received.dat";

    if B:
        OFFSET = 49;
    else:
        OFFSET = 49*2;

    SIZE = 27*8;
    bit_errors = 0;
    sent = ""; received = "";
#-----
    read_s = open(file_s,'rb').read();
    read_r = open(file_r,'rb').read();

    for s in read_s:
        sent += '{0:08b}'.format(s);
    for r in read_r:
        received += '{0:08b}'.format(r);

    for i in range(0, SIZE):
        #print((sent[i], received[i+OFFSET]))
        if(sent[i] ≠ received[i+OFFSET]):
            bit_errors += 1;

    print("SENT: \n" + sent[:SIZE] + "\n")
    print("RECEIVED: \n" + received[OFFSET:OFFSET+SIZE])

    print("Bit errors: " + str(bit_errors));
    return bit_errors
```

Source code: *Script* responsável pela automatização da comparação de ficheiros (sim.py)

```
#!/bin/python3
import signal
from bit_error import bit_error

B=True;
BW = 0.0; N = 0.0;

if B:
    from mdBPSK_copy import mdBPSK as top_block_cls
else:
    from mdQPSK_copy import mdQPSK as top_block_cls

#-----
def sim():
    N0 = [x/2 for x in range(9)];
    loop_BW = [round(6/100 * x/10, 3) for x in range(1,43)];
    #---
    f = open("data.csv", "w"); f.write("N,BW,bit_errors\n")
    #--- .csv
    for N in N0:
        for BW in loop_BW:
            tb = top_block_cls(BW,N); #simulação
            tb.start();
            #tb.show();
            tb.wait();
            b = bit_error(B);
            f.write(str(N) + "," + str(BW) + "," + str(b) + "\n");

    tb.show();
    #---
    f.close();
    #---
    def sig_handler(sig=None, frame=None):
        tb.stop();
        tb.wait();
    #---
    signal.signal(signal.SIGINT, sig_handler);
    signal.signal(signal.SIGTERM, sig_handler);
```

As alterações necessárias foram aplicadas aos ficheiros *Python* criados de forma automática pelo *software GNU Radio* de modo a que se tornassem suscetíveis à automatização requerida para a análise de dados significantes subsequente.

E.2 Análise do sistema BPSK

Tab. E1: Taxa de erro de *bit* em função da *Loop Bandwidth*, para cada *noise voltage*.

Loop BW/Noise	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
0.006	0.0%	1.4%	9.7%	19.0%	25.5%	30.6%	34.7%	38.0%	39.4%
0.012	0.0%	1.4%	9.7%	19.0%	25.0%	29.6%	34.3%	35.6%	38.0%
0.018	0.0%	1.4%	9.7%	19.4%	25.0%	28.7%	33.3%	36.6%	33.8%
0.024	0.0%	1.4%	9.7%	19.4%	24.5%	29.6%	34.7%	39.4%	34.3%
0.03	0.0%	1.4%	9.7%	19.0%	24.5%	29.6%	34.7%	34.3%	37.5%
0.036	0.0%	1.4%	9.7%	19.0%	25.0%	28.7%	31.0%	33.3%	38.9%
0.042	0.0%	1.4%	9.7%	18.5%	25.0%	37.0%	32.4%	34.3%	39.8%
0.048	0.0%	1.4%	9.3%	18.5%	24.5%	28.2%	31.0%	36.1%	44.9%
0.054	0.0%	0.9%	9.7%	18.5%	35.2%	31.5%	31.0%	38.0%	39.4%
0.06	0.0%	0.9%	9.3%	20.8%	25.0%	28.7%	34.7%	50.0%	43.1%
0.066	0.0%	0.9%	9.3%	20.4%	25.0%	28.2%	34.7%	40.3%	41.2%
0.072	0.0%	0.9%	8.8%	19.9%	25.9%	27.3%	33.3%	56.0%	55.1%
0.078	0.0%	0.9%	10.6%	19.0%	27.3%	29.2%	40.7%	40.3%	44.4%
0.084	0.0%	0.9%	12.0%	19.4%	26.9%	29.6%	37.5%	35.6%	42.6%
0.09	0.0%	0.9%	11.6%	16.7%	27.8%	38.4%	41.2%	56.9%	51.4%
0.096	0.0%	0.9%	11.6%	17.1%	25.9%	38.4%	48.1%	32.4%	49.5%
0.102	0.0%	0.9%	10.2%	19.9%	25.5%	41.7%	44.4%	42.1%	50.5%
0.108	0.0%	0.9%	9.7%	20.4%	25.5%	38.4%	49.1%	46.3%	54.2%
0.114	0.0%	0.9%	8.8%	23.1%	30.1%	37.0%	47.7%	42.1%	50.0%
0.12	0.0%	0.9%	7.4%	18.5%	39.8%	44.4%	51.9%	48.1%	49.5%
0.126	0.0%	0.9%	7.9%	21.3%	40.7%	33.3%	48.6%	47.7%	46.3%
0.132	0.0%	0.9%	9.3%	19.9%	44.0%	33.3%	47.2%	53.7%	48.6%
0.138	0.0%	0.9%	10.6%	21.8%	37.5%	37.0%	48.1%	47.7%	50.0%
0.144	0.0%	0.9%	8.3%	16.7%	37.5%	44.9%	50.5%	44.0%	53.7%
0.15	0.0%	0.9%	8.3%	19.0%	38.0%	39.4%	43.1%	44.9%	53.2%
0.156	0.0%	0.9%	10.6%	17.1%	28.2%	36.1%	47.7%	53.2%	50.5%
0.162	0.0%	0.9%	9.7%	19.0%	44.0%	35.2%	44.4%	44.4%	53.7%
0.168	0.0%	0.9%	9.7%	21.8%	31.9%	48.1%	41.7%	52.3%	51.9%
0.174	0.0%	0.9%	9.3%	27.8%	39.8%	43.1%	53.7%	50.5%	44.4%
0.18	0.0%	0.9%	8.8%	31.0%	48.6%	50.5%	45.4%	53.2%	47.2%
0.186	0.0%	0.9%	9.3%	30.6%	42.1%	44.4%	52.3%	57.9%	46.8%
0.192	0.0%	0.9%	11.6%	21.8%	48.1%	42.6%	51.9%	52.3%	49.5%
0.198	0.0%	1.4%	7.4%	21.3%	45.8%	52.8%	45.4%	46.8%	47.2%
0.204	0.0%	1.4%	8.3%	39.8%	44.4%	57.9%	51.4%	49.1%	51.9%
0.21	0.0%	0.9%	16.2%	42.6%	34.3%	30.6%	50.5%	45.8%	50.5%
0.216	0.0%	1.9%	23.6%	40.7%	43.5%	40.3%	43.5%	45.8%	50.5%
0.222	0.0%	1.4%	13.9%	53.2%	44.0%	53.2%	51.9%	44.4%	50.9%
0.228	0.0%	1.9%	12.5%	19.9%	36.6%	48.6%	47.2%	48.1%	49.5%
0.234	0.0%	8.3%	10.6%	31.9%	25.9%	47.2%	48.6%	45.8%	54.6%
0.24	0.0%	0.9%	11.1%	38.9%	36.6%	44.9%	44.4%	50.5%	51.9%
0.246	0.0%	1.4%	12.5%	38.4%	46.8%	49.1%	52.8%	50.5%	42.1%
0.252	0.0%	1.4%	28.7%	21.8%	49.5%	44.9%	45.4%	43.5%	52.3%

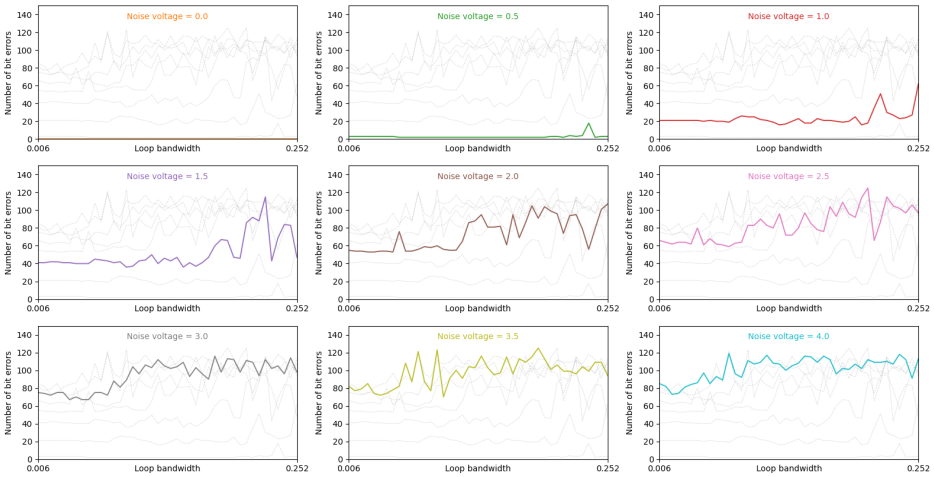


Fig. E1: Evolução do número de erros de *bit* para cada *noise voltage*, em função da *Loop Bandwidth*.

Observação 1 → Verifica-se, como esperado, um número de erros cada vez mais prevalente e mais oscilante para valores de *Loop Bandwidth* cada vez mais distantes de zero (distanciamento da gama ótima de valores[7] discutida anteriormente na secção 3.1). É também de salientar a proporcionalidade observada (e expectável) do número de erros de *bit* com o patamar de ruído (aumento da dificuldade em detetar o pulso no seio do ruído $\implies$ maior taxa de erro de *bit*).

Observação 2 → Para cada gráfico é aparente uma zona de maior estabilidade (número de erros bastante próximos) que tende a contrair e a mutar para uma extensão cada vez mais instável, com o aumento da *noise voltage*.

Nota → Encontram-se a verde na tabela os valores da *Loop Bandwidth* que minimizam a taxa de erro de *bit* para cada patamar de ruído no sistema BPSK projetado. A segunda observação é corroborada por estes valores *highlighted*, dado que o valor ótimo de *Loop Bandwidth* aparenta tender para valores nominalmente mais próximos de zero com o aumento da *noise voltage*.

E.3 Análise do sistema QPSK

Tab. E2: Taxa de erro de *bit* em função da *Loop Bandwidth*, para cada *noise voltage*.

Loop BW/Noise	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
0.006	0.0%	1.9%	11.1%	19.0%	28.2%	31.5%	34.3%	36.6%	38.0%
0.012	0.0%	1.4%	11.1%	19.0%	27.8%	31.0%	33.8%	35.6%	37.5%
0.018	0.0%	1.4%	10.6%	18.5%	26.4%	30.6%	33.3%	36.6%	32.9%
0.024	0.0%	1.4%	10.6%	18.1%	26.9%	29.6%	33.3%	32.4%	35.2%
0.03	0.0%	1.4%	9.7%	18.1%	25.5%	29.6%	31.0%	35.2%	33.8%
0.036	0.0%	0.5%	9.3%	17.6%	25.0%	28.2%	35.2%	35.6%	35.2%
0.042	0.0%	0.5%	9.3%	17.6%	23.1%	27.3%	33.8%	33.3%	41.7%
0.048	0.0%	0.5%	9.7%	17.6%	25.0%	30.1%	34.7%	32.9%	46.8%
0.054	0.0%	0.5%	9.3%	17.1%	24.5%	29.6%	36.6%	41.2%	37.0%
0.06	0.0%	0.0%	9.7%	15.7%	31.0%	30.1%	38.0%	33.8%	45.8%
0.066	0.0%	0.0%	7.9%	17.1%	24.1%	36.1%	32.9%	44.0%	37.5%
0.072	0.0%	0.0%	6.9%	19.9%	26.4%	31.5%	54.6%	44.0%	56.0%
0.078	0.0%	0.0%	8.3%	17.6%	27.8%	41.2%	34.7%	35.2%	50.5%
0.084	0.0%	0.0%	8.3%	23.1%	27.8%	45.4%	55.1%	34.3%	52.8%
0.09	0.0%	0.0%	6.9%	22.7%	25.9%	41.7%	45.4%	48.1%	46.8%
0.096	0.0%	0.0%	8.8%	25.9%	25.9%	31.9%	36.1%	38.4%	46.3%
0.102	0.0%	0.0%	6.5%	18.5%	48.6%	30.6%	51.4%	49.5%	48.6%
0.108	0.0%	0.0%	2.8%	17.6%	51.4%	40.3%	52.8%	51.9%	43.1%
0.114	0.0%	0.0%	3.7%	19.9%	48.6%	37.5%	34.3%	53.2%	38.9%
0.12	0.0%	0.0%	8.8%	33.8%	27.3%	54.6%	49.1%	50.9%	57.4%
0.126	0.0%	0.0%	10.6%	20.4%	30.6%	50.0%	54.6%	53.7%	46.3%
0.132	0.0%	0.0%	8.8%	33.3%	26.9%	34.7%	49.5%	52.8%	50.5%
0.138	0.0%	0.0%	13.9%	16.7%	27.8%	44.9%	51.4%	47.7%	53.2%
0.144	0.0%	0.0%	17.1%	16.2%	30.6%	47.7%	47.2%	53.2%	57.4%
0.15	0.0%	0.0%	15.3%	19.0%	26.4%	50.9%	50.5%	50.9%	51.9%
0.156	0.0%	0.0%	31.0%	49.1%	33.8%	33.3%	49.5%	54.2%	54.2%
0.162	0.0%	0.0%	15.7%	50.9%	25.9%	41.2%	47.2%	49.1%	52.3%
0.168	0.0%	0.0%	16.2%	47.7%	25.0%	36.1%	50.9%	52.8%	53.7%
0.174	0.0%	0.0%	8.3%	39.8%	33.3%	47.2%	47.2%	55.6%	48.6%
0.18	0.0%	0.0%	6.5%	35.2%	35.2%	32.4%	53.7%	51.4%	45.8%
0.186	0.0%	0.0%	10.2%	35.6%	31.5%	53.7%	48.6%	42.1%	50.9%
0.192	0.0%	0.0%	10.2%	21.3%	32.4%	49.1%	56.5%	52.8%	53.7%
0.198	0.0%	0.0%	29.6%	19.4%	40.3%	53.7%	52.8%	51.9%	53.2%
0.204	0.0%	0.0%	29.2%	19.4%	43.1%	52.8%	52.3%	45.8%	47.2%
0.21	0.0%	0.9%	10.6%	31.0%	50.9%	53.7%	48.6%	50.5%	42.6%
0.216	0.0%	0.9%	30.1%	23.6%	43.5%	47.2%	44.9%	50.5%	40.3%
0.222	0.0%	0.5%	10.2%	21.8%	31.9%	29.2%	46.3%	48.6%	51.4%
0.228	0.0%	0.5%	7.4%	22.2%	36.6%	48.1%	48.6%	49.1%	53.7%
0.234	0.0%	0.5%	7.9%	21.8%	32.4%	51.4%	51.9%	50.0%	51.4%
0.24	0.0%	0.5%	6.0%	47.2%	38.0%	47.7%	48.1%	44.9%	44.9%
0.246	0.0%	0.0%	6.5%	48.1%	51.9%	45.4%	47.7%	44.0%	51.9%
0.252	0.0%	0.5%	11.1%	50.0%	46.3%	55.1%	54.2%	52.3%	42.6%

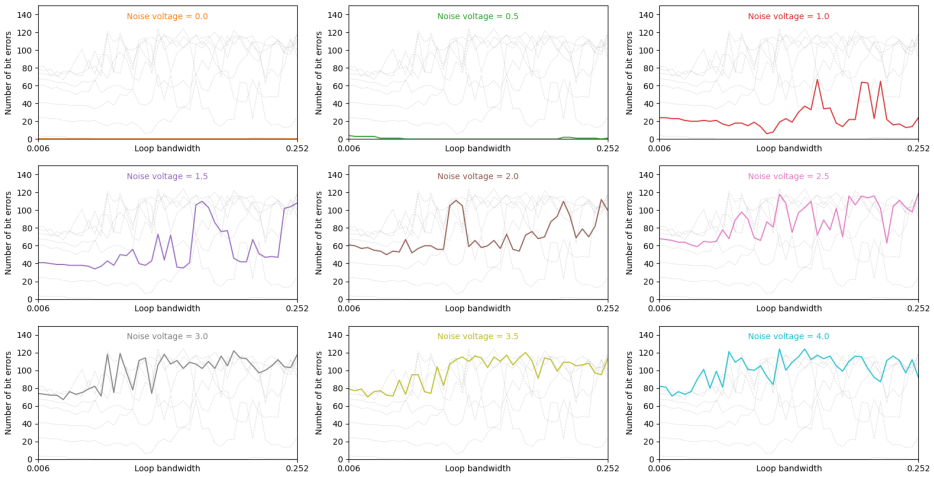
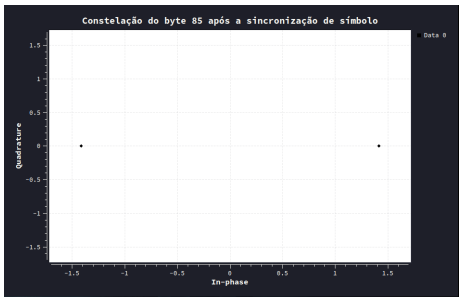


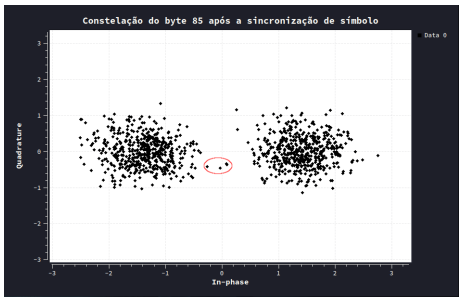
Fig. E2: Evolução do número de erros de bit para cada *noise voltage*, em função da *Loop Bandwidth*.

Nota → Os resultados e observações são bastante análogos à versão BPSK, pelo que se seguem as mesmas ilações. Salienta-se somente, para reforçar a ideia, que em média, a taxa de erro de *bit* para cada patamar de ruído impõe precisamente a mesma evolução que no caso exposto anteriormente (vide Fig. E1 e Tab. E1).

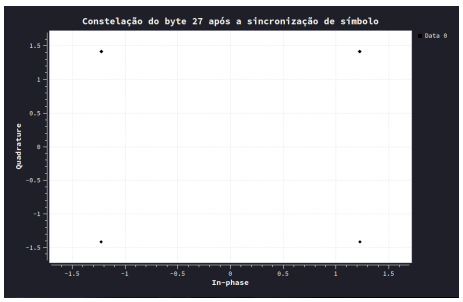
Conclusões



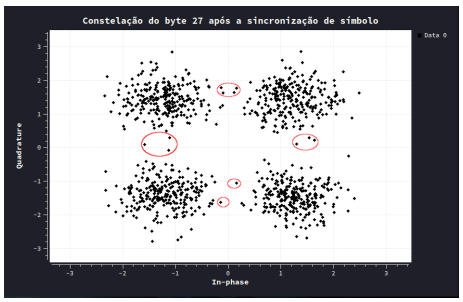
(a) BPSK sem *noise voltage*.



(b) BPSK com 0.5 V de *noise voltage*.



(c) QPSK sem *noise voltage*.



(d) QPSK com 0.5 V de *noise voltage*.

Fig. C1: Constelações com e sem presença de AWGN.

É conveniente aliar a discussão dos dados obtidos com a visualização dos diagramas de constelação perante a influência de ruído. Na Fig. C1 encontram-se explicitadas a vermelho amostras situadas numa região de indecisão face à presença de AWGN (ambiguidade na decisão entre *bit* '0' ou '1', vide a secção 3.2).

Trivialmente se correlacionam os dados adquiridos (ver Tab. E1 e Tab. E2, e respetivas análises gráfica) com este fenómeno—é aparente que com o aumento do AWGN, há uma tendência (geral) esmagadora, quase obrigatória, para o aumento do número de *bits* nestas zonas litigiosas (graças à dificuldade na deteção do pulso no seio do ruído, como apresentado anteriormente → redução da relação sinal-ruído).

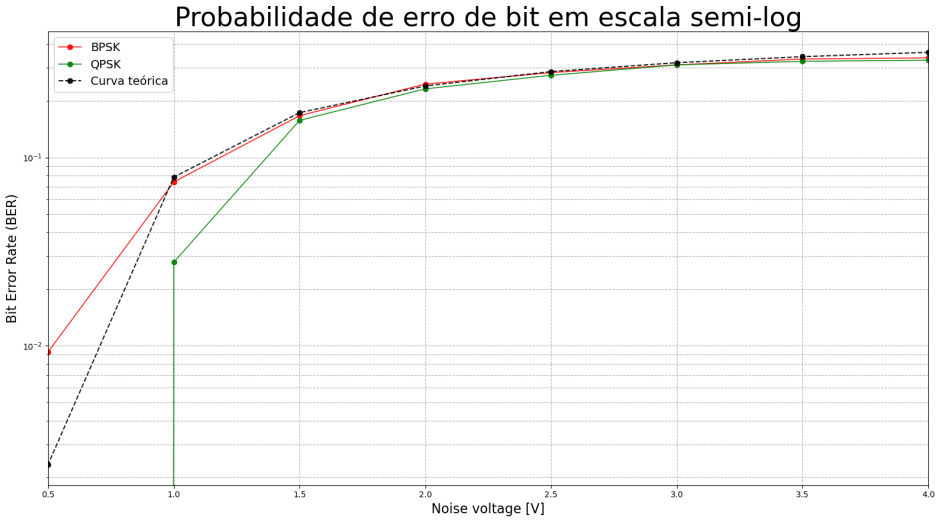


Fig. C2: Curvas de probabilidade de erro de *bit* (BER) em escala semi-logarítmica: BPSK, QPSK e teórica[12].

Não obstante, a sobreposição das curvas apresentadas na Fig. C2 verifica a similaridade (esperada e bastante proeminente por análise direta das tabelas expostas) entre as probabilidades de erro de *bit* e ratifica o conteúdo teórico—sucintamente condensado na seguinte citação:

> "The biggest and primary advantage of QPSK over BPSK is spectral efficiency: for the same BER performance and data rate we can use half the bandwidth! This is also intuitively explained that we are sending two BPSK signals independently in the same bandwidth."[13]

[12] As curvas BPSK e QPSK são computadas mediante a menor taxa de erro de *bit* observada nas tabelas apresentadas para cada patamar de ruído; e a devida curva teórica é obtida com base na expressão supramencionada na secção 4.1, aplicada sobre o *range* discreto de *noise voltages* para $\sqrt{E_b} = \sqrt{2}$ (distância à origem visualizada experimentalmente em ambiente *GNU Radio*).

Referências

[1] Constellation Modulator, 2022. URL https://wiki.gnuradio.org/index.php/Constellation_Modulator.

[2] Mathuranathan. Complex baseband equivalent models, Nov 2020. URL https://www.gaussianwaves.com/2017/10/complex-baseband-equivalent-models/.

[3] Sergio Benedetto and Ezio Biglieri. *Principles of Digital Transmission: With wireless applications*. Springer US, 2002.

[4] Channel model, 2022. URL https://wiki.gnuradio.org/index.php/Channel_Model.

[5] A guide to SDR and DSP using python. URL https://pysdr.org/content/pulse_shaping.html.

[6] Symbol timing synchronization: A tutorial - Igor Freire: EE ph.D.., Sep 2021. URL https://igorfreire.com.br/2016/10/15/symbol-timing-synchronization-tutorial/.

[7] Symbol sync, 2022. URL https://wiki.gnuradio.org/index.php/Symbol_Sync.

[8] avi1987 and Dan BoschenDan Boschen. Loop bandwidth for symbol timing recovery, Aug 1963. URL https://dsp.stackexchange.com/questions/31170/loop-bandwidth-for-symbol-timing-recovery/31186#31186.

[9] Binary slicer, 2022. URL https://wiki.gnuradio.org/index.php/Binary_Slicer.

[10] Guia de laboratório, 2022.

[11] Constellation object, 2022. URL https://wiki.gnuradio.org/index.php/Constellation_Object.

[12] "BPSK and QPSK have identical BER curve ?". URL https://www.dsprelated.com/showthread/comp.dsp/53352-1.php.

[13] Dan BoschenDan Boschen jstraughjstraugh, MBazMBaz and Ng PhNg Ph. "Do BPSK vs QPSK have the same performance?", 2022. URL https://dsp.stackexchange.com/questions/79490/do-bpsk-vs-qpsk-have-the-same-performance.

[14] Simon S. Haykin and Michael Moher. *Introduction to Analog and Digital Communications*. Wiley, 2007.

[15] Simon Haykin. *Digital Communications*. Wiley, 2013.

[16] Simon S. Haykin. *Adaptive Filter Theory*. Pearson Education, 2014.

[17] Ana A Paniagua Rodriguez. Sampling frequencies ratio estimation and symbol timing recovery for baseband binary pulse amplitude modulation. 2008.

[18] Symbol Clock Recovery and improved symbol synchronization blocks· GNU Radio, Jan 1AD. URL https://www.gnuradio.org/grcon/grcon17/presentations/symbol_clock_recovery_and_improved_symbol_synchronization_blocks/.

[19] Bakalsk projekty. URL https://www.fekt.vut.cz/conf/EEICT/archiv/sborniky/EEICT_2005_sbornik/01-Bakalarske_projekty/07-Informacni_systemy/index.html.

[20] Guided tutorial PSK demodulation. URL https://wiki.gnuradio.org/index.php/Guided_Tutorial_PSK_Demodulation.

[21] Pulse Shaping & Filtering in Communications Systems. URL https://www.ni.com/pt-pt/innovations/white-papers/06/pulse-shape-filtering-in-communications-systems.html.