ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO
Mini-Project

ĐỀ TÀI

INTRODUCTION TO COMMUNICATION ENGINEERING

Giảng viên hướng dẫn: PhD. Trịnh Văn Chiến

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*	*Mã lớp*
Phan Trung Dũng (1)	20205073	135279
Hoàng Minh Ngọc (2)	20200440	135279
Hoàng Vân Trường (3)	20205134	135279

Hà Nội, 2023

Mục Lục

MINI-PROJECT

INTRODUCTION TO COMMUNICATION ENGINEERING

General regulations:

- It is either a single-based or group-based (recommended, maximum 3 students) project. For the latter, each member performs one task and understands another task.
- Write the report using the latex template (link: https://soict.hust.edu.vn/bieu-mau-va-quy-dinh-danh-cho-sinh-vien.html)
- Present the source code and the report to the instructor.
- Students are greatly appreciated to propose their own interesting tasks in this course.

Task 1: Modulate and demodulate amplitude shift keying (ASK) signal from a random binary sequence

1. Modulation: To perform the ASK modulation, the following instruction may be useful
   - Generate and plot the carrier signal
   - Generate and plot the binary data sequence
   - Perform ASK modulation and plot the ASK modulated signal
2. Demodulation: To perform the ASK demodulation, the following instruction may be useful
   - Correlate the ASK modulated signal with the carrier signal to generate decision variables
   - Obtain the demodulated binary data based on the decision variables
3. Investigate the ASK modulation/demodulation under the effects of Gaussian noise, the following instruction may be useful
   - Gaussian noise with zero mean and variance $N_0/2$ is added to the transmitted waveform as $r(t) = s(t) + n(t)$
   - Numerically compute the error probability
4. Theory: Derive the bit error probability of a Gaussian channel using the ASK modulation/demodulation used in your task

Task 2: Modulate and demodulate phase shift keying (PSK) signal from a random binary sequence

1. Modulation: To perform the PSK modulation, the following instruction may be useful
   - Generate and plot the carrier signals
   - Generate and plot the binary data sequence
   - Perform PSK modulation and plot the PSK modulated signal
2. Demodulation: To perform the PSK demodulation, the following instruction may be useful
   - Correlate the PSK modulated signal with the carrier signal to generate decision variables
   - Obtain the demodulated binary data based on the decision variables
3. Investigate the PSK modulation/demodulation under the effects of Gaussian noise, the following instruction may be useful
   - Gaussian noise with zero mean and variance $N_0/2$ is added to the transmitted waveform as $r(t) = s(t) + n(t)$

- Numerically compute the error probability

4. Theory: Derive the bit error probability of a Gaussian channel using the PSK modulation/demodulation used in your task

Task 3: Modulate and demodulate frequency shift keying (FSK) signal from a random binary sequence

1. Modulation: To perform the FSK modulation, the following instruction may be useful
   - Generate and plot the carrier signal(s)
   - Generate and plot the binary data sequence
   - Perform PSK modulation and plot the FSK modulated signal
2. Demodulation: To perform the FSK demodulation, the following instruction may be useful
   - Correlate the FSK modulated signal with the carrier signal to generate decision variables
   - Obtain the demodulated binary data based on the decision variables
3. Investigate the FSK modulation/demodulation under the effects of Gaussian noise, the following instruction may be useful
   - Gaussian noise with zero mean and variance $N_0/2$ is added to the transmitted waveform as $r(t) = s(t) + n(t)$
   - Numerically compute the error probability
4. Theory: Derive the bit error probability of a Gaussian channel using the FSK modulation/demodulation used in your task

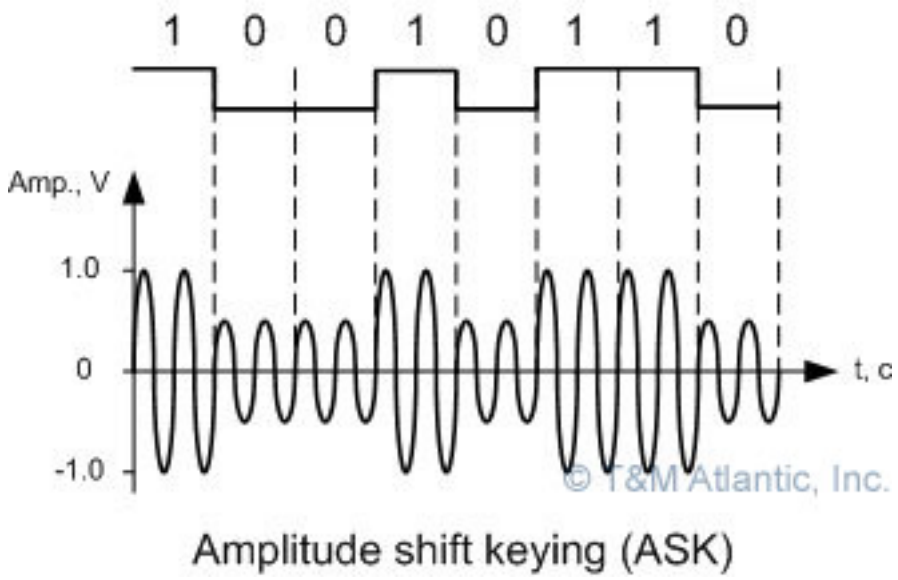
Lời cảm ơn

Sau khi đào sâu tìm hiểu, cũng như tự hệ thống lại kiến thức để có thể thực hiện Mini Project này, em tự thấy bản thân đã học học được thêm rất nhiều kiến thức mới, hiểu kỹ và sâu sắc hơn những công thức trên lý thuyết được học. Em cũng rất mong thời gian tới sẽ có cơ hội được tiếp xúc, thực hành nhiều hơn với các hệ thống truyền thông để chúng em có thể hiểu được áp dụng triệt để những lý thuyết đã học ở trường.
em xin trân thành cảm ơn thầy Trịnh Văn Chiến đã cho chúng em được làm bài, hệ thống lại kiến thức vô cùng bổ ích của môn học Kỹ Thuyết Truyền Thông, bọn em thấy bài học khá là hay và thú vị, rất là ý nghĩa trong chương trình học của bọn em, em mong muốn có thể gặp thầy trong những bài học, học phần hấp dẫn tới.
chúng em xin chân thành cảm ơn.

1 Điều chế và giải điều chế ASK (Amplitude Shift Keying)

1.1 Không gian tín hiệu 2-ASK:

ASK (Binary Amplitude Shift Keying) là một loại kỹ thuật điều chế biên độ số. Tín hiệu ASK có dạng sóng giao động tần số f, sóng tín hiệu truyền đi được tạo ra bằng cách thay đổi biên độ của sóng mang.



Amplitude shift keying (ASK)

Trong điều chế 2-ASK, một bit sẽ được gán với một sóng mang với biên độ khác nhau.
Không gian tín hiệu 2-ASK:

$$M = \{s_1 = A_1 sin(2\pi ft); s_2 = A_2 sin(2\pi ft)\}$$

Cơ sở trực chuẩn: $b(t) = \sqrt{\frac{2}{T}} sin(2\pi ft)$ với $T = \frac{1}{R_b}$.

1.2 Kênh tạp âm trắng có tính cộng AWGN

Ở mọi hệ thống truyền thông, tín hiệu nhận được đều bị méo đi do có nhiễu. Ở đây, ta giả thiết nhiễu là nhiễu Gauss trắng có tính cộng.
Kênh AWGN có các tính chất sau:

- Tuyến tính và bất biến theo thời gian
- Đáp ứng tần số lý tưởng $H(f) = 1$
- Tạp âm Gaus có tính cộng

– Tiến trình ngẫu nhiên ergodic: các đặc trưng thống kê của tiến trình có thể suy ra được từ một chuỗi các mẫu đủ dài của nó

– Mỗi biến ngẫu nhiên tuân theo phân phối chuẩn Gauss với giá trị trung bình bằng 0

- Mật độ công suất phổ tín hiệu là hằng số $G_n(f) = \frac{N_0}{2}$

1.3 Bộ lọc phối hợp (Matched Filter)

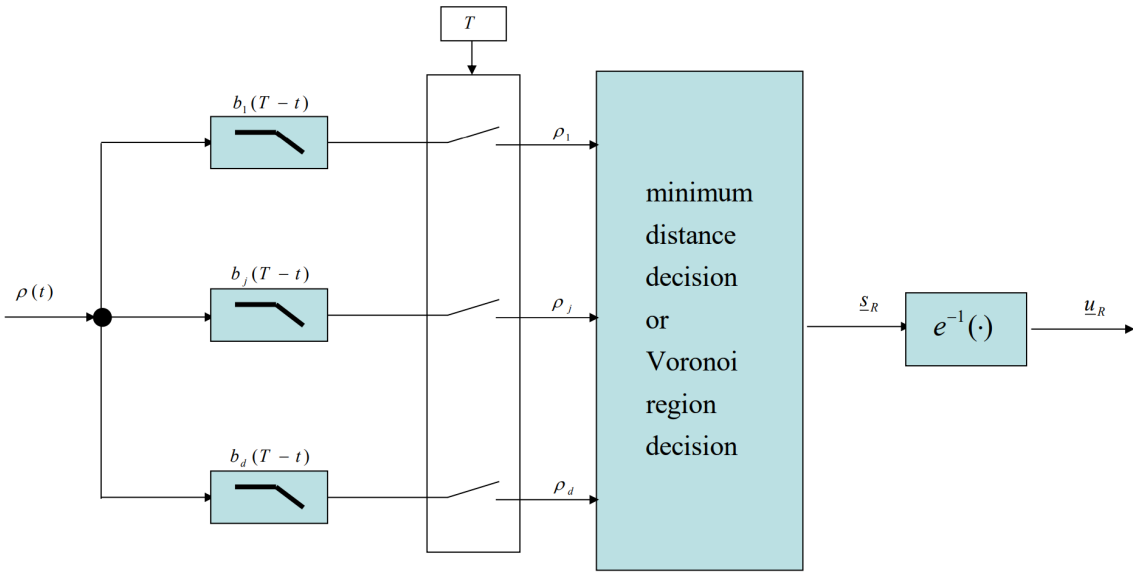
Chọn bộ lọc có đáp ứng xung $h(t) = b(T - t)$ ta được bộ lọc phối hợp.
Tín hiệu đầu vào bộ lọc là tín hiệu nhận được bên bộ thu: $\rho(t)$.
Đáp ứng xung là: $h(t) = b_j(t)$
Đầu ra của bộ lọc phối hợp là:

$$y(t) = \int_{-\infty}^{+\infty} \rho(\tau)h(t-\tau)\ dx = \int_{-\infty}^{+\infty} \rho(\tau)b_j(T-t+\tau)\ dt$$

Giả thiết lấy mẫu tín hiệu đầu ra tại thời điểm $T = 1$:

$$y(t=T) = \int_{-\infty}^{+\infty} \rho(\tau)b_j(\tau)\ dx = \int_0^T \rho(\tau)b_j(\tau)\ d\tau$$

Như vậy, sử dụng MF cho ta một phương án thay thế có thể được dùng để tính toán các phép chiếu $b_j(t)$ thay vì phải sử dụng các bộ tích phân các bộ tích phân



1.4 Giải thích Code

Đầu tiên ta khai báo các thư viện để tính toán và mô phỏng các tín hiệu:

```
import random
import numpy as np
import matplotlib.pyplot as plt
import math
```

Số lượng bit truyền đi: 8 Thời gian truyền 1 bit: 1 Thời gian truyền tín hiệu: 8 Tạo mạng bit ngẫu nhiên:

```
N = 8
Tb = 1
T = 1*N
t = np.arange(0, Tb, Tb/800)
def random_bits_array(n):
    return [random.randint(0, 1) for _ in range(n)]
m = random_bits_array(N)
```

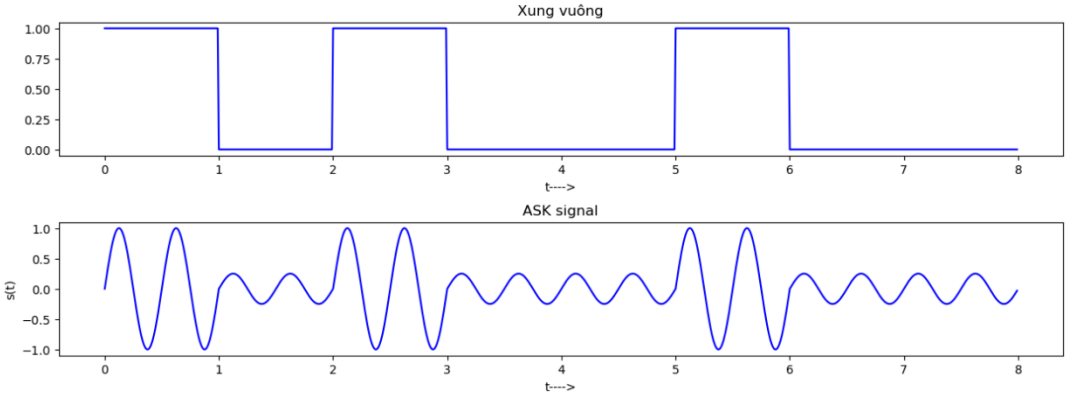
Điều chế 2-ASK
Chuyển các bit thành các tín hiệu tương ứng:

- 1: $A_1 sin(2\pi ft)$
- 0: $A_2 sin(2\pi ft)$

Ta chọn $A_1 = 1$, $A_2 = 0.25$, $f = 2R_b$

```
f = 2
A1 = 1
A2 = 0.25
def modulation():
    out = []
    f = 2
    A1 = 1
    A2 = 0.25
    t = np.arange(0, Tb, Tb/100)
    s1 = A1 * np.sin(2 * np.pi * f * t)
    s2 = A2 * np.sin(2 * np.pi * f * t)
    for i in range(len(m)):
        if m[i] == 0:
            out.extend(s2)
        if m[i] == 1:
            out.extend(s1)
    return out
ask = modulation()
```

Kết quả:

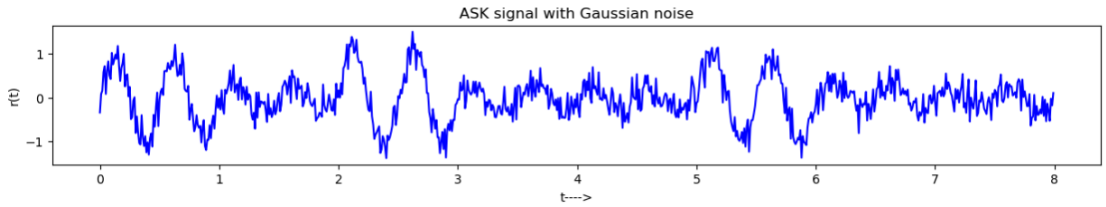


Thêm nhiễu Gauss

Nhiễu Gauss được thêm vào có giá trị trung bình là 0, phương sai $\frac{N_0}{2}$

```
N0 = 0.1
noise = np.random.normal(0, np.sqrt(N0/2), len(ask))
ask_noisy = ask + noise
```

Kết quả:



Giải điều chế tín hiệu

Cách làm: Ta tính hình chiếu của tín hiệu $r(t)$ trên cơ sơ trực chuẩn B , sau đó ra quyết định bằng tiêu chuẩn khoảng cách ngắn nhất để khôi phục tín hiệu $s(t)$ ở nơi nhận.

Cơ sở trực chuẩn $B = \{b(t) = \sqrt{\frac{2}{T_b}} sin(2\pi ft)\}$

Sử dụng bộ lọc phối hợp tính các hình chiếu của $r(t)$ lên không gian trực chuẩn M:

$\rho[n] = \int_{nT_b}^{(n+1)T_b} \rho(t) b_j(t)\ dt$

Áp dụng tiêu chuẩn khoảng cách nhỏ nhất ta suy ra:

Nếu $\rho[n] \geq \sqrt{\frac{T_b}{2}} \frac{(A_1+A_2)}{2}$, ký tự nhị phân khôi phục $s'[n] = 1$.

Nếu $\rho[n] < \sqrt{\frac{T_b}{2}} \frac{(A_1+A_2)}{2}$, ký tự nhị phân khôi phục $s'[n] = 0$.

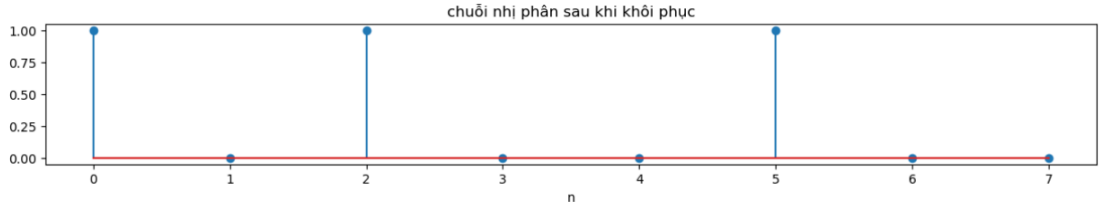
Source code:

```
def demodulation():
    t = np.arange(0, Tb, Tb/100)
    hx = math.sqrt(2/Tb)*np.sin(2 * np.pi * f * t)
    ask_noisy_list = np.array_split(ask_noisy, 8)
    output = []
    for i in range(len(ask_noisy_list)):
        if np.trapz(ask_noisy_list[i] * hx,t) >= np.sqrt(Tb
/2)*(A1+A2)/2:
```

```
            output.append(1)
        else:
            output.append(0)
    return output
```

```
demod = demodulation()
```

Kết quả:



1.5 Xác xuất lỗi:

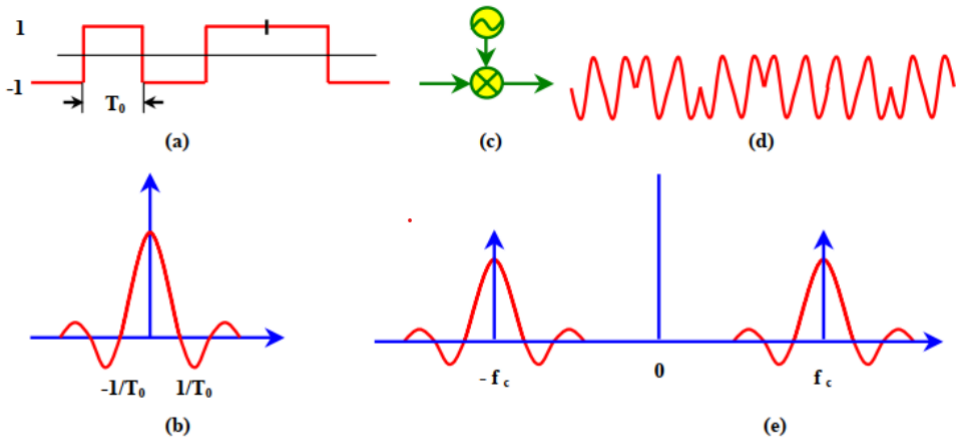
Ta đã có kết luận: Các không gian tín hiệu khác nhau (khác dạng sóng truyền) nhưng có cùng không gian vector thì có BER như nhau.
Do đó $BER = SER = \frac{1}{2}erfc(\sqrt{\frac{E_0}{N_0}})$

2 Điều chế và Giải điều chế PSK (Phase Shift Keying)

2.1 Giới thiệu về PSK

PSK là sơ đồ điều chế kỹ thuật số truyền dữ liệu bằng cách thay đổi hoặc điều chế pha của tín hiệu tham chiếu (sóng mang). PSK sử dụng một số giai đoạn hữu hạn, mỗi giai đoạn được gán một mẫu chữ số nhị phân duy nhất. Thông thường, mỗi pha mã hóa một số bit bằng nhau. Mỗi mẫu bit định dạng biểu tượng được biểu thị bằng pha cụ thể.
Trong hệ thống điều chế BPSK, tín hiệu băng tần gốc được gắn vào sóng mang bằng cách thay đổi pha của sóng mang tùy thuộc vào tín hiệu băng gốc.
Giả sử có sóng mang được biểu diễn:
$x_0(t) = A.\sin(\omega_0 + \varphi)$
Biểu thức tín hiệu gốc: S(t) là tín hiệu nhị phân (0,1) hay là chuỗi NRZ.
Do đó ta có:
- Khi $s(t) = 0 : P(t) = A.sin(\omega_0)$
- Khi $s(t) = 0 : P(t) = A.sin(\omega_0 + \pi)$

Đối với khóa dịch pha PSK, thông tin chứa trong pha tức thời của sóng mang điều chế. Thường thì pha này được ấn định và so sánh tương thích với sóng mang của pha đã biết - PSK kết hợp. Kiểu điều chế BPSK này còn được gọi là điều chế khóa đảo pha PRK (Phase Reversal Keying). Hình 7.7, trình bày bộ phát PRK, dạng sóng tín hiệu và phổ.



Hình 7.7 *Điều chế PRK: dạng sóng, bộ điều chế, phổ*

(b) Tín hiệu băng gốc (b) Phổ điện áp của tín hiệu băng gốc (c) Bộ điều chế PRK

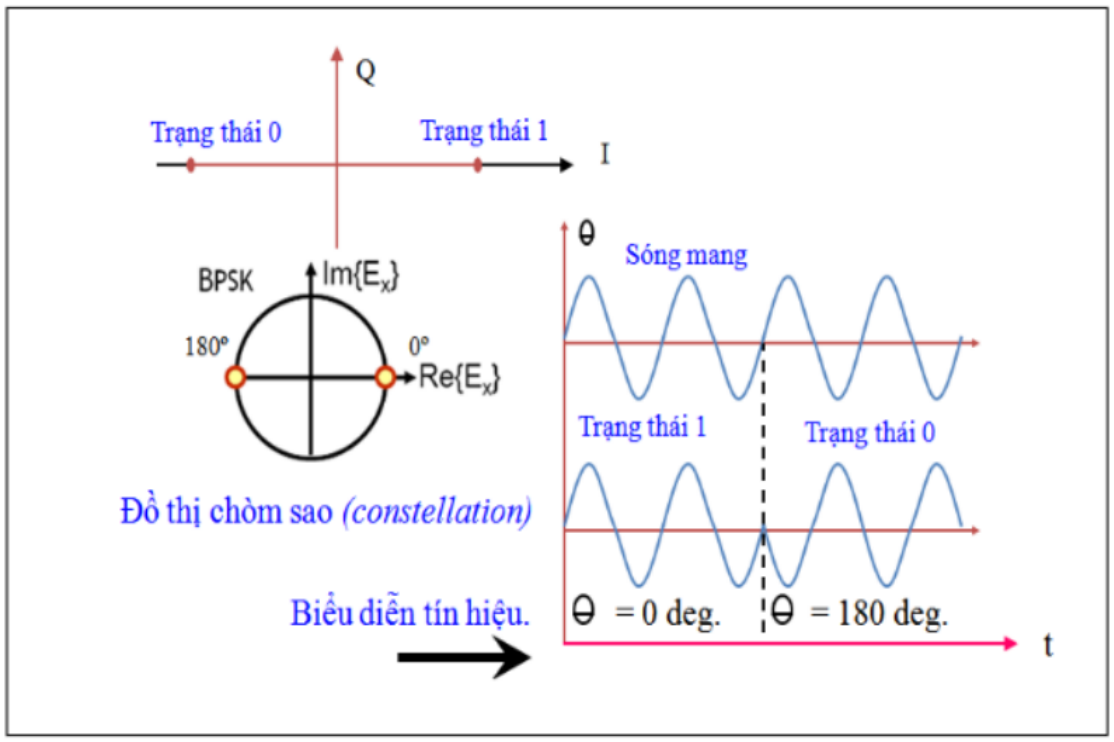
(d) Tín hiệu PRK thông dải (e) Phổ điện áp của tín hiệu PRK thông dải

2.2 Nguyên tắc khóa dịch pha nhị phân

Nguyên tắc : Các tín hiệu nhị phân tác dụng lên sóng mang làm thay đổi pha của sóng mang. Cụ thể là:
• Bit 1: pha của sóng mang là 0.
• Bit 0: pha của sóng mang là 180^o.
Bảng chân lý của tín hiệu điều chế BPSK:

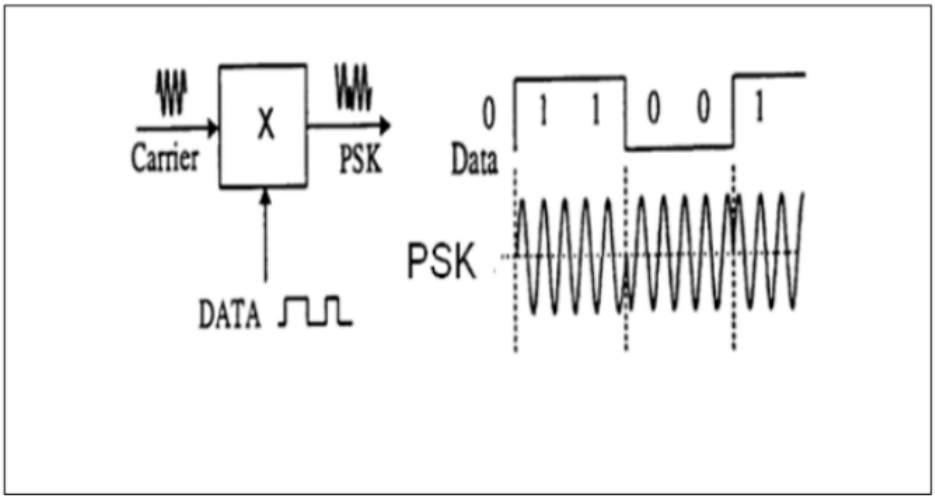
Nhị phân đầu vào	Pha đầu ra
Logic 0	180^o
Logic 1	0^o

Có thể thấy rõ hơn trong cách biểu diễn trên đồ thị thời gian và trạng thái của tín hiệu BPSK.



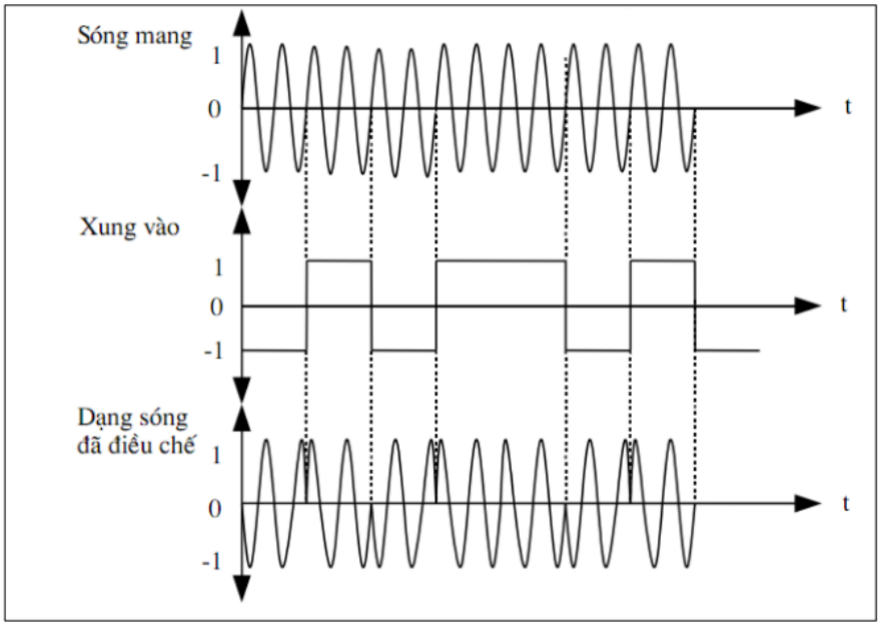
Hình 1: Đồ thị thời gian và trạng thái.

2.3 Điều chế PSK



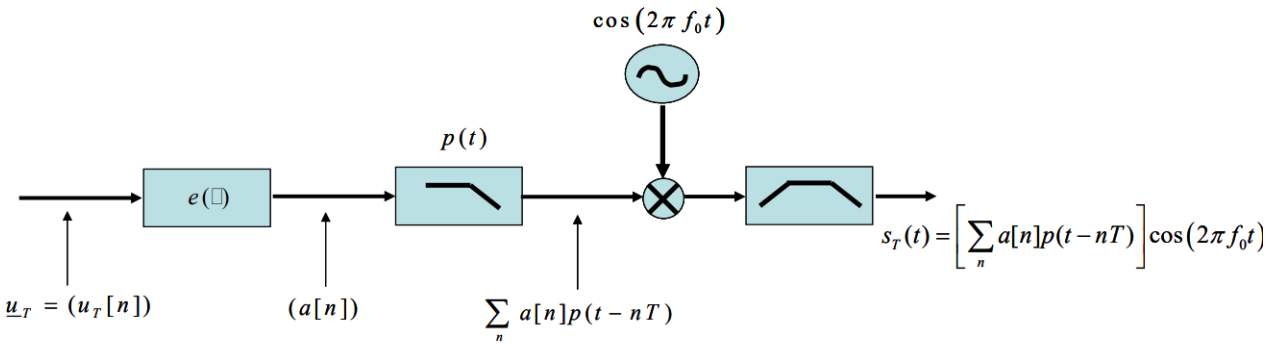
Hình 2: Sơ đồ khối thực hiện điều chế PSK

Phương pháp điều chế 2-PSK hay BPSK (Binary PSK) hay điều chế ngược pha (Phase Reversal Keying) được giới thiệu ở hình trên.
Sơ đồ tạo tín hiệu BPSK dạng sin với hai giá trị pha tùy thuộc vào giá trị Data:
- Khi data bit = 1, tín hiệu BPSK cùng pha với sóng mang
- Khi data bit = 0, tín hiệu BPSK ngược pha (180^o) với sóng mang.



Hình 3: Quan hệ pha/thời gian
ở đầu ra bộ điều chế PSK theo tín hiệu vào

Mô hình điều chế PSk



Mô hình điều chế PSK

2.4 Mã nguồn điều chế tín hiệu PSK sử dụng ngôn ngữ Python, trình bày Jupyter Notebook

- Thêm thư viện

```
import random
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt
```

hàm tạo mảng bit ngẫu nhiên

```
def random_bits_array(n):
    return [random.randint(0, 1) for _ in range(n)]
```

Hàm tạo tín hiệu xung vuông và xung vuông đảo ngược

```
def message_signal(m,T):
    t = np.arange(0, T, T/100)
    message = []
    not_message = []
    for i in range(len(m)):
        if m[i] == 1:
            m_s = np.ones(len(t))
            invm_s = np.zeros(len(t))
        else:
            m_s = np.zeros(len(t))
            invm_s = np.ones(len(t))
        message.extend(m_s)
        not_message.extend(invm_s)
    return message , not_message
```

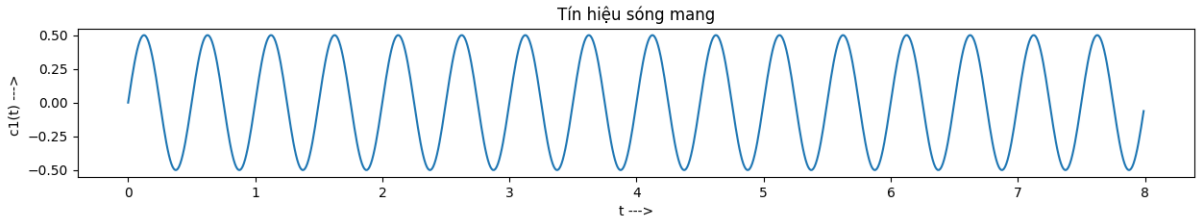
khai báo

```
N = 8
T = 1
Tb = 1*N
t = np.arange(0, Tb, Tb/800)
m = random_bits_array(N)
fc = 2
songmang = np.sqrt(2/Tb) * np.sin(2 * np.pi * fc * t)
```

Vẽ sóng mang

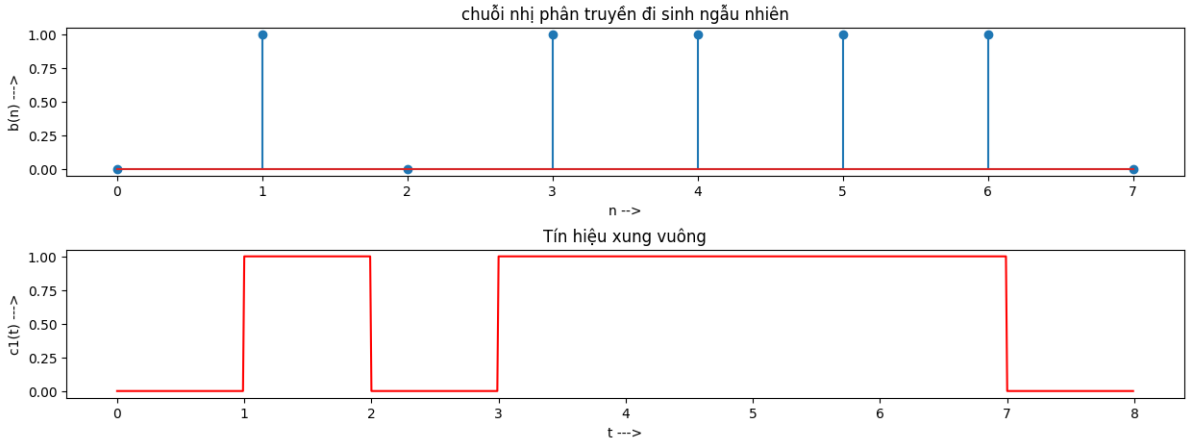
```
fig, ax = plt.subplots(1, 1, figsize=(15, 2))
fig.subplots_adjust(hspace=0.5)
ax.plot(t, c)
ax.set_title('Tin hieu song mang')
ax.set_xlabel('t --->')
```

```
ax.set_ylabel('c1(t) ---> ')
```



Vẽ chuỗi nhị phân và xung vuông tương ứng

```
fig, ax = plt.subplots(2, 1, figsize=(15, 5))
fig.subplots_adjust(hspace=0.5)
ax[0].stem(m)
ax[0].set_title('Chuoi nhi phan sinh ngau nhien')
ax[0].set_xlabel('n -->')
ax[0].set_ylabel('b(n) --->')
message , not_message = message_signal(m,T)
print(len(message), len(not_message))
ax[1].plot(t,message, 'r')
ax[1].set_title('Tin hieu xung vuong')
ax[1].set_xlabel('t --->')
ax[1].set_ylabel('c1(t) --->')
```

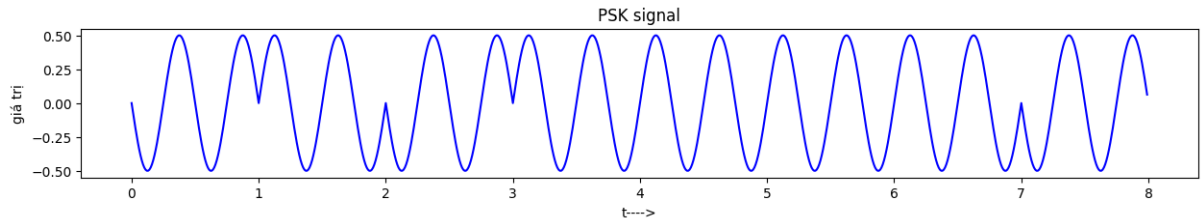


Điều chế PSK

```
def modulation(message_signal,not_message_signal):
    return message_signal*songmang
           - not_message_signal*songmang
psk = modulation(message,not_message)
```

Tín hiệu PSK sau khi điều chế

```
fig , ax = plt.subplots(1, 1, figsize=(15, 2))
ax.figsize=(15, 5)
ax.plot(t,psk, 'b')
ax.set_title('PSK signal')
ax.set_xlabel('t------>')
ax.set_ylabel('Gia tri')
```



2.5 Giải điều chế PSK

Bộ giải điều chế BPSK bao gồm:
- Sơ đồ lấy bình phương để chuyển các tín hiệu khác pha về cùng 1 pha
- Vòng giữ pha PLL phát lại nhịp với tần số gấp đôi tần số sóng mang
- Bộ chia pha $\Delta\phi$ để hiệu chỉnh pha
- Bộ chia hai để đưa tần số tín hiệu tái lập về bằng tần số sóng mang
- Bộ nhân tín hiệu thực hiện nhân sóng điều chế PSK với sóng mang tái lập

Giả sử tần số sóng mang là fc, ω c = 2πfc, ta có hai trường hợp:
Khi tín hiệu PSK là $+\sin(\omega_c t)$ ứng với Data bit = 1, sóng mang tái lập là $\sin(\omega_c t)$, sơ đồ nhân sẽ cho tín hiệu:
$\sin(\omega_c t).\sin(\omega_c t)) = \sin(\omega_c t)^2 = \frac{1}{2}.(1-\cos(2\omega_c t)) = \frac{1}{2} - \frac{1}{2}\cos(2.\omega_c t)$
Trong biểu thức trên, thành phần thứ hai là xoay chiều, có tần số gấp đôi tần số sóng mang. Khi sử dụng bộ lọc thông thấp với tần số cắt bằng tần số sóng mang, ta có thể khử bỏ thành phần xoay chiều và thế dương của thành phần một chiều thứ nhất được giữ lại sẽ biểu diễn trạng thái "1" của bit Data.
Khi tín hiệu PSK là $-\sin(\omega_c t)$ ứng với Data bit = 0,sơ đồ nhân sẽ cho tín hiệu
$-\sin(\omega_c t).\sin(\omega_c t)) = -\sin(\omega_c t)^2 = -\frac{1}{2}.(1-\cos(2\omega_c t)) = -\frac{1}{2}+\frac{1}{2}\cos(2.\omega_c t)$
Trong biểu thức trên, thành phần thứ hai là xoay chiều, có tần số gấp đôi tần số sóng mang. **Khi sử dụng bộ lọc thông thấp với tần số cắt bằng tần số sóng mang, ta có thể khử bỏ thành phần xoay chiều** và thế âm của thành phần một chiều thứ nhất được giữ lại sẽ biểu diễn trạng thái "0" của bit Data.

2.6 Mã nguồn giải điều chế tín hiệu PSK sử dụng ngôn ngữ Python, trình bày Jupyter Notebook

Hàm giải mã tín hiệu Nếu tín hiệu mẫu trùng với sóng sin được sử dụng, thì tổng của tích sóng sẽ lớn, còn nếu tín hiệu mẫu không trùng với sóng sin được sử dụng, tổng của tích sóng sẽ nhỏ. Để tính tổng này, chúng ta dùng vòng lặp để chạy qua mỗi giá trị của N, tính tổng với khoảng tín hiệu tương ứng, rồi cập nhật giá trị start và end cho lần chạy tiếp theo của vòng lặp.

```
    def demodulation(filtered_signal,N):
#  g i i  m  t n  h i u  n h n  c
    start = 0
    end = 100
    demod = np.zeros(N)
    for i in range(N):
        x = np.sum(songmang[start:end]
                   * filtered_signal[start:end])
        if x > 0:
            demod[i] = 1
        else:
            demod[i] = 0
        start += 100
        end += 100
    return demod
```

Vẽ tín hiệu sau khi điều chế

3 Điều tra điều chế/giải điều chế PSK dưới tác động của nhiễu Gaussian

Nhiễu Gaussian là một tín hiệu ngẫu nhiên có mật độ phân bố công suất phẳng nghĩa làtín hiệu nhiễu có công suất bằng nhau trong toàn khoảng băng thông. Tín hiệu này có tên là nhiễu trắng vì nó có tính chất tương tự với ánh sáng trắng. Chúng ta không thể tạo ra nhiễu trắng theo đúng lý thuyết vì theo định nghĩa của nó, nhiễu trắng có mật độ phổ công suất phân bố trong khoảng tần vô hạn và do vậy nó cũng phải có công suất vô hạn. Tuy nhiên, trong thực tế, chúng ta chỉ cần tạo ra nhiễu trắng trong khoảng băng tần của hệ thống chúng ta đang xem xét.
Với khóa dịch pha nhị phân (PSK), các bit nhị phân 1 và 0 có thể được biểu thị bằng các mức năng lượng tương ứng : $+\sqrt{E_b}$ và $-\sqrt{E_b}$.

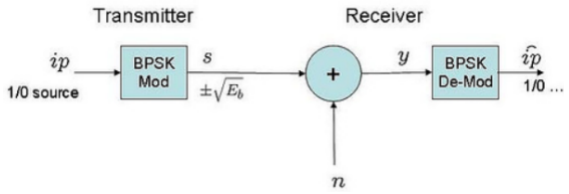


Figure: Simplified block diagram with BPSK transmitter-receiver

3.1 Thêm nhiễu Gaussian vào dạng sóng truyền đi

a) Khi sóng truyền có dạng r(t) = s(t) + n(t), 0 t Tb.

Tín hiệu ra tại thời điểm lấy mẫu $t = T_b$ là:

$$r = E + n$$

Trong đó, năng lượng tín hiệu E và thành phần tạp âm cộng n được tính theo:

$$n = \int_0^{T_b} n(t)s(t)\,\mathrm{d}t.$$

Vì quá trình tạp âm cộng có trung bình 0 nên E[n] = 0 và phương sai của thành phần tạp âm n là:

$$\begin{aligned}
\sigma^2 &= E[n^2]b \\
&= \int_0^{T_b}\int_0^{T_b} E[n(t)n(\tau)]\,\mathrm{d}t\mathrm{d}\tau \\
&= \frac{N_0}{2}\int_0^{T_b}\int_0^{T_b} \sigma(t-\tau)s(t)s(\tau)\,\mathrm{d}t\mathrm{d}\tau \\
&= \frac{N_0}{2}\int_0^{T_b} s^2(t)\,\mathrm{d}t \\
&= \frac{N_0 E}{2}
\end{aligned}$$

Theo đó, hàm mật độ xác suất của r khi chuyền s(t) qua kênh gauss trắng (AWGN) là:

$$p(r|s(t) \text{ đã được truyền đi}) \equiv p(r|0) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{\frac{-(r+E)^2}{2\sigma^2}} \quad \text{với } \mu = 0 \text{ và } \sigma^2 = \frac{N_0}{2}$$

4.2 Tính toán xác suất lỗi

Xét xác suất lỗi với điều kiện kênh AWGN, dưới tác động của tạp âm làm cho tín hiệu đầu ra bị lỗi được tính như sau.
Nếu s(t) đã được truyền qua kênh, xác xuất lỗi nhận được là:

$$
\begin{aligned}
P_e &= P(r < 0) \\
&= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{0} e^{-(r-E)^2/2\sigma^2} \, \mathrm{d}r \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-E/\sigma} e^{-r^2/2} \, \mathrm{d}r \\
&= Q\left(\sqrt{\frac{2E}{N_0}}\right)
\end{aligned}
$$

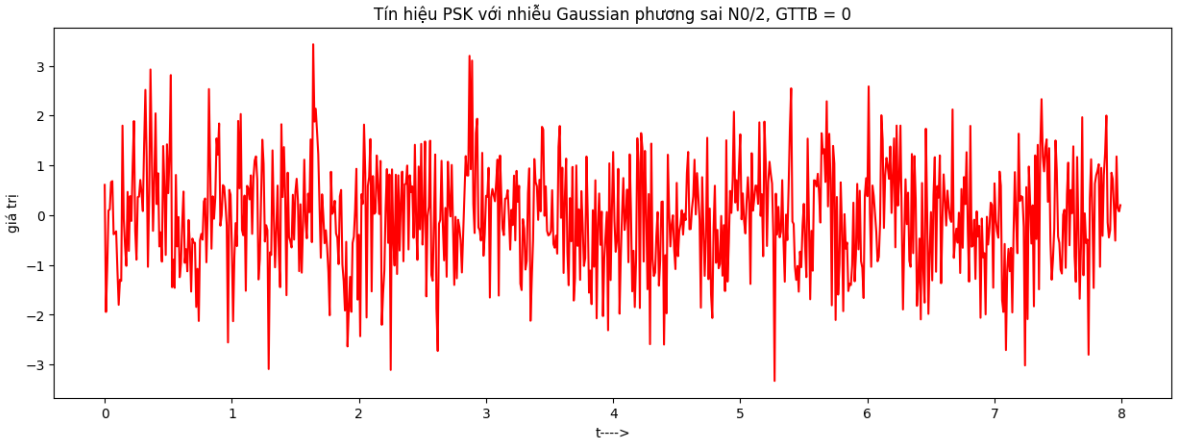
3.2 Mã nguồn điều chế tín hiệu/ giải điều chế có sự can thiệp bởi nhiễu Gaussian PSK sử dụng ngôn ngữ Python, trình bày Jupyter Notebook

Thêm nhiễu Gaussian với phương sai N0/2, giá trị trung bình = 0

```
N0 = 2
noise = np.random.normal(0, np.sqrt(N0/2), psk.shape)
psk_noisy = psk + noise #r(t) = s(t) + n(t)
```

Vẽ tín hiệu khi đã thêm nhiễu nhiệt

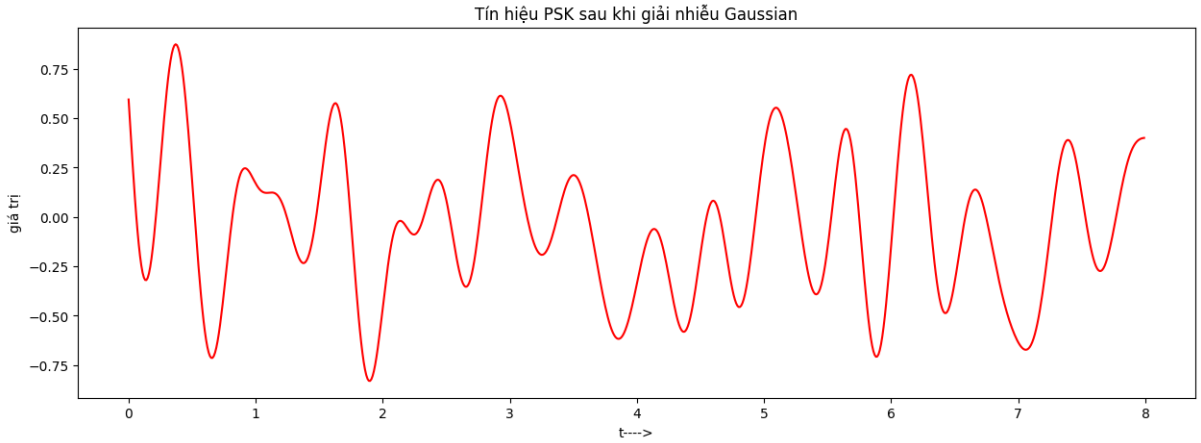
```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(t, psk_noisy, 'r')
ax.set_title('Tin hieu PSK, nhieu Gaussian')
ax.set_xlabel('t------>')
ax.set_ylabel('Gia tri')
plt.show()
```



Giải nhiễm Gaussian

```
def gaussian_noise_cancellation(signal,
                                cutoff=0.05, order=5):
    b, a = butter(order, cutoff, btype='low', analog=False)
    filtered_signal = filtfilt(b, a, signal)
    return filtered_signal
```

Vẽ tín hiệu sau khi giải nhiễu



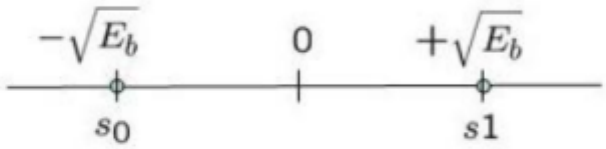
4 Tính xác suất lỗi bit của kênh Gaussian PSK

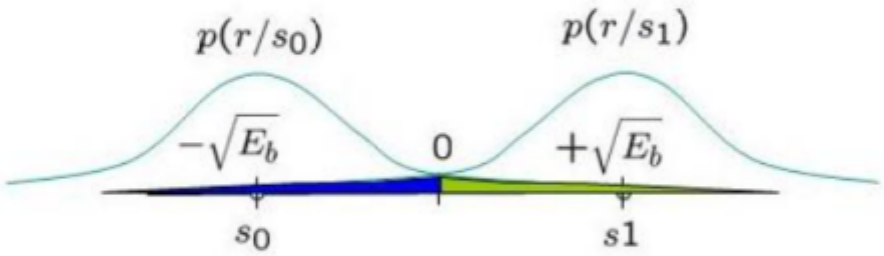
4.1 Xác suất lỗi bit theo lí thuyết

Xét xác suất lỗi với điều kiện kênh AWGN, dưới tác động của tạp âm làm cho tín hiệu đầu ra bị lỗi được tính như sau.
Nếu s(t) đã được truyền qua kênh, tín hiệu nhận được là:
$r = s_1 + n$ khi bit 1 được truyền đi và
$r = s_0 + n$ khi bit 0 được truyền đi.
Hàm phân phối xác suất có điều kiện của r trong hai trường hợp là:

$$p(r|s_0) = \frac{1}{\sqrt{\pi N_0}} e^{\frac{-\left(r+\sqrt{E_b}\right)^2}{N_0}}$$

$$p(r|s_1) = \frac{1}{\sqrt{\pi N_0}} e^{\frac{-\left(r-\sqrt{E_b}\right)^2}{N_0}}$$





Hình: Hàm mật độ xác suất có điều kiện trong điều chế BPSK

Giả sử xác suất xảy ra s_1 và s_0 là như nhau, khi đó: $p(s_1) = p(s_0) = \frac{1}{2}$.

- Nếu tín hiệu nhận được $r > 0$, thì máy thu cho rằng s_1 đã được truyền đi
- Nếu tín hiệu nhận được $r \leq 0$, thì máy thu cho rằng s_0 đã được truyền đi.

Xác suất lỗi khi s_1 được truyền đi:
Với ngưỡng này, xác suất lỗi khi s_1 truyền đi là (vùng màu xanh dương):

$$p(e|s_1) = \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} e^{\frac{-(y-\sqrt{E_b})^2}{N_0}} dy = \frac{1}{\sqrt{\pi}} \int_{\sqrt{\frac{E_b}{N_0}}}^{\infty} e^{-z^2} dz = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

Trong đó: $erfc(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-x^2} \mathrm{d}x$ là hàm trả về hàm bù của sai số Gauss tại một giá trị: $erfc(x) = 1 - erf(x)$

Xác suất lỗi khi s0 được truyền đi:
Tương tự, xác suất lỗi khi s_0 được truyền đi là (khu vực trong vùng màu xanh lá cây)

$$p(e|s_0) = \frac{1}{\sqrt{\pi N_0}} \int_{0}^{\infty} e^{\frac{-(y+\sqrt{E_b})^2}{N_0}} dy = \frac{1}{\sqrt{\pi}} \int_{\sqrt{\frac{E_b}{N_0}}}^{\infty} e^{-z^2} dz = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

Ta có tổng xác suất lỗi bit:

$$P_b = p(s_1)p(e|s_1) + p(s_0)p(e|s_0).$$

Vì s_1 và s_0 được giả sử xác suất xảy ra như nhau, tức là $p(s_1) = p(s_0) = \frac{1}{2}$.
Xác suất lỗi bit là:

$$P_b = \frac{1}{2}\, erfc\left(\sqrt{\frac{E_b}{N_0}}\right).$$

4.2 Mã nguồn tính xác suất lỗi bit

```
def calculate_bit_error_rate(bitBanDau, bitNhanDuoc):
num_errors = 0
for i in range(len(bitBanDau)):
    if bitBanDau[i] != bitNhanDuoc[i]:
        num_errors += 1
ber = num_errors / len(bitBanDau)
return ber
bitBanDau = m
bitNhanDuoc = demod
ber = calculate_bit_error_rate(bitBanDau, bitNhanDuoc)
print("Xac xuat loi bit: {:.2f}%".format(ber * 100))
```

Phổ biến nhất, kênh không được sử dụng trực tiếp cho việc truyền tải tín hiệu dải gốc. Tín hiệu dải gốc signal được sử dụng để điều khiển một số thông số của sóng nạp, khiến cho các thông số thay đổi theo sự thay đổi của tín hiệu dải gốc, đây là gọi là tín hiệu tần số. Theo nguyên lý, sóng nạp được chỉnh sửa có thể là bất kỳ dạng sóng nào, miễn là nó phù hợp cho việc truyền tải trung gian và có thể phân biệt được giữa các tín hiệu khác nhau. Thực tế, trong hầu hết các hệ thống giao tiếp số, tín hiệu *sin* là lựa chọn làm sóng nạp, vì nó đơn giản và dễ tạo và nhận. So với tín hiệu tương tự và tín hiệu số, nguyên lý cơ bản là giống nhau, khác biệt chỉ là trong quá trình tín hiệu tương tự, thông số của sóng nạp được chỉnh sửa liên tục, tại đầu nhận, các thông số của sóng nạp được đánh giá liên tục.

5 Cơ sở lí thuyết

Khi tín hiệu dữ liệu số được truyền bằng phương pháp FM, thì kỹ thuật này được gọi là kỹ thuật dời tần (FSK: Frequency- Shift Keying). Nó được sử dụng rộng rãi vì khả năng chống lại sự mất tầm rất mạnh. FSK được sử dụng để truyền tín hiệu kỹ thuật số với các tần số khác nhau, và tần số của tín hiệu nạp được điều khiển bởi tín hiệu kỹ thuật số cơ sở. Tín hiệu nạp được nhận tại đầu nhận được chuyển đổi thành tín hiệu kỹ thuật số để hoàn thành quá trình truyền thông thông tin. 2FSK là hình thức đơn giản nhất của chuyển dấu chuyển tần số, có thể biểu diễn như sau:

$$S(t) = m_1(t)A\cos(2\pi f_1 t + \varphi_1) + m_2(t)A\cos(2\pi f_2 t + \varphi_2)$$

Mô hình điều chế đa tần số có thể được hình thành bằng cách thêm sóng hình sin với các số khác nhau,tần số và biên độ khác nhau. Ví dụ:

$$\begin{aligned} S(t) = \quad & m_1(t)A\cos(2\pi f_1 t + \varphi_1) \quad + \\ & m_2(t)A\cos(2\pi f_2 t + \varphi_2) \quad + \\ & m_3(t)A\cos(2\pi f_3 t + \varphi_3) \quad + \\ & \qquad\qquad\qquad\qquad\quad \vdots \\ & m_m(t)A\cos(2\pi f_m t + \varphi_m) \quad + \end{aligned}$$

5.1 Điều chế

Phương pháp điều chế 2-FSK sử dụng các tần số khác nhau của sóng mang để biểu diễn các bit khác nhau (0 hoặc 1) của tín hiệu nhị phân

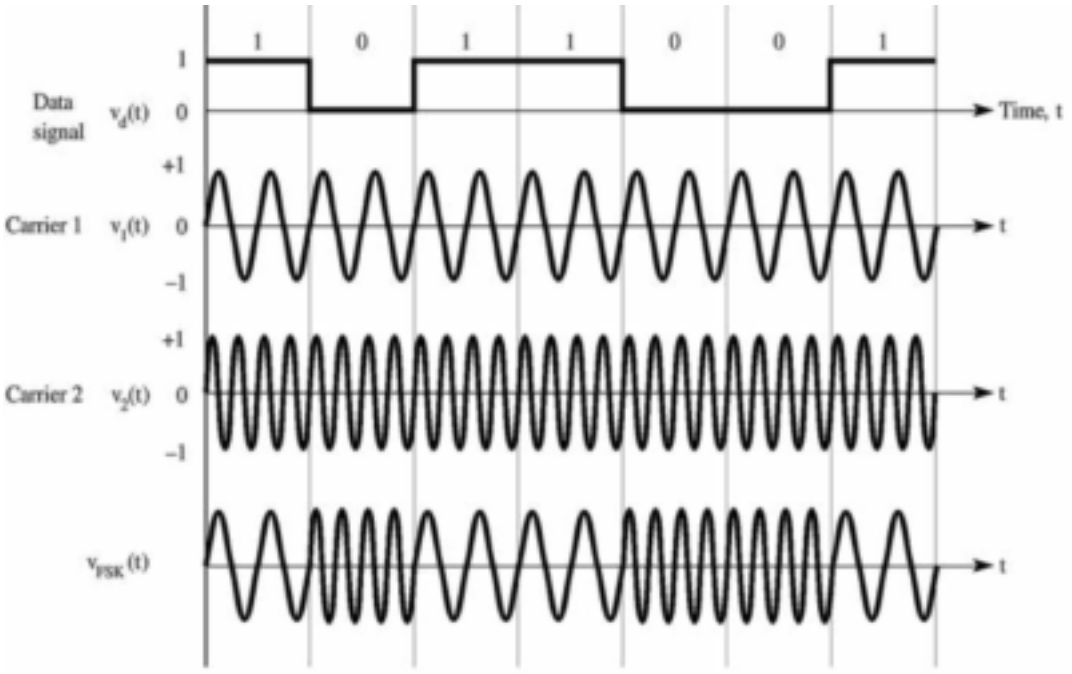
cần mã hóa, nói theo cách khác, sóng tín hiệu truyền đi được tạo ra bằng cách thay đổi tần số của sóng mang.
Ta có không gian tín hiệu M:

$$s_1 = \sqrt{\frac{2}{Tb}} cos(2\pi f_1 t)$$

$$s_2 = \sqrt{\frac{2}{Tb}} cos(2\pi f_2 t)$$

Ta có $f_0 = 1/R_b$. Giải sử sóng mang có cần tần số f_1, $f_2 = kf_0$, $(k \in Z)$.
Ta gán:

- Với giá trị bit 1 : $\sqrt{\frac{2}{Tb}} cos(2\pi f_1 t)$
- Với giá trị bit 0 : $\sqrt{\frac{2}{Tb}} cos(2\pi f_2 t)$



Hình 1: Sơ đồ điều chế FSK

5.2 Bộ lọc phối hợp Matched Filter

Mỗi bộ lọc được đặc trưng bởi đáp ứng xung: $h(t)$
Tín hiệu đầu ra $y(t)$ được xác định bởi tín hiệu đầu vào $x(t)$ và đáp ứng xung như sau:

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

Theo bài ra ta có:

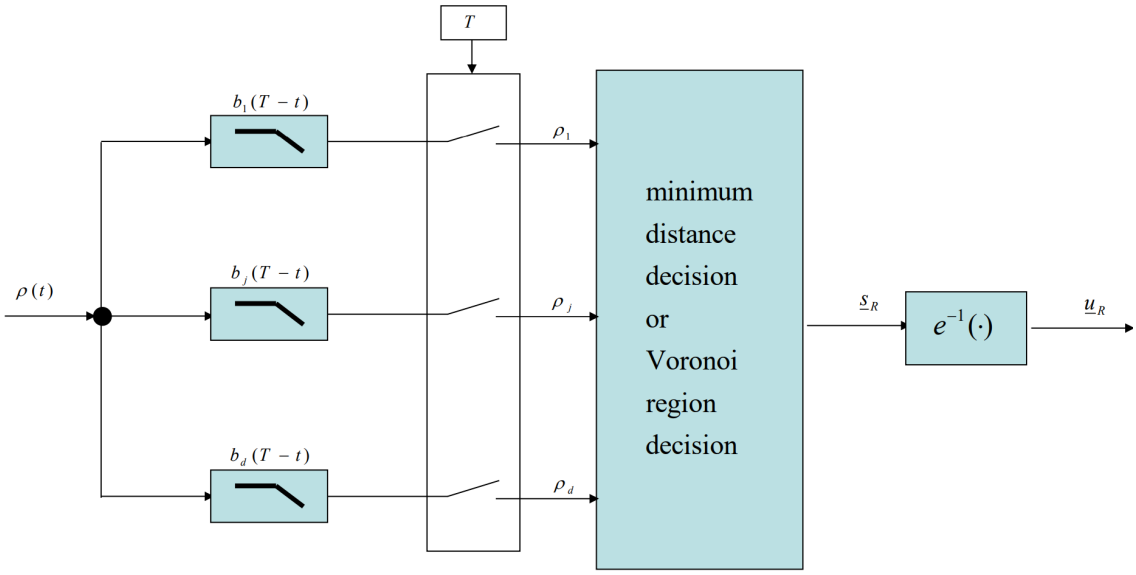
- Tín hiệu đầu vào là tín hiệu nhận được sau khi điều chế: $p(t)$
- Đáp ứng xung là $h(t) = b_j(T - t)$

Từ đó ta thu được đầu ra của bộ lọc là:

$$y(t) = \int_{-\infty}^{\infty} p(\tau)h(t-\tau)d\tau = \int_{-\infty}^{\infty} p(\tau)b_j(T-t+\tau)d\tau$$

Giả thiết lấy mẫu tín hiệu đầu ra tại thời điểm $t = T$

$$y(t=T) = \int_{-\infty}^{\infty} p(\tau)b_j(\tau)d\tau = \int_{0}^{T} p(\tau)b_j(\tau)d\tau = p_j$$



5.3 Nguyên lí giải điều chế

Các bước giải điều chế FSK:

1. Từ M xây dựng cơ sở trực chuẩn:

$$b_1(t) = \sqrt{\frac{2}{T}} P_T(t) cos(2\pi f_1 t)$$

$$b_2(t) = \sqrt{\frac{2}{T}} P_T(t) cos(2\pi f_2 t)$$

2. Xây dựng bộ lọc phối hợp Matched Filter (MF)
3. Xác định giá trị đầu ra của bộ lọc tại các thời điểm $t = nT$
4. Tính khoảng cách đến các điểm s_1, s_2 được biểu diễn trong hệ trục tọa độ trực chuẩn. Nếu khoảng cách nào nhỏ nhất thì quyết định sóng mang này biểu diễn bit nào.

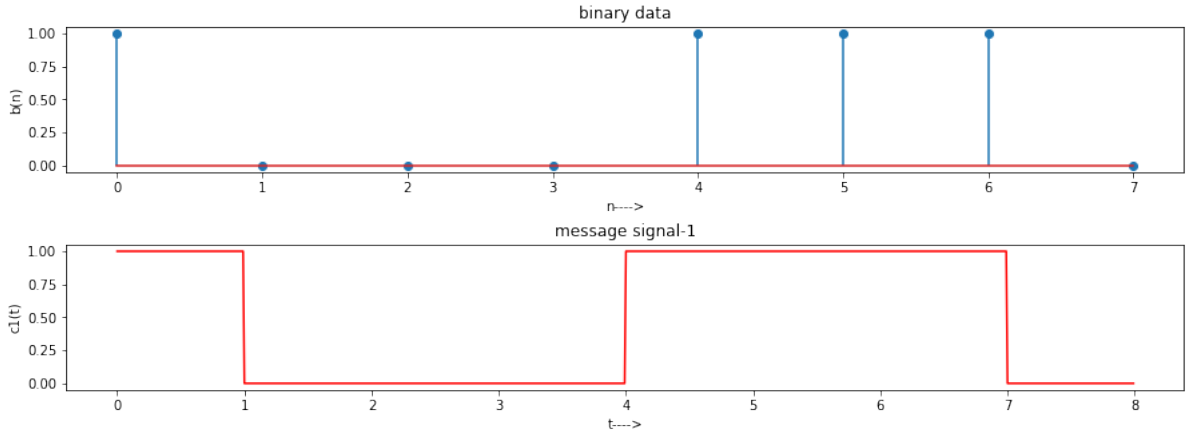
6 Lập trình với python

6.1 Tạo chuỗi bit ngẫu nhiên và tín hiệu xung

Tạo chuỗi N bit ngẫu nhiên bằng hàm *rantint* trong python và biểu diễn dưới dạng xung vuông:

```
import random
import numpy as np
import matplotlib.pyplot as plt

def random_bits_array(n):
    return [random.randint(0, 1) for _ in range(n)]
def message_signal(m,T):
    t = np.arange(0, T, T/100)
    message = []
    not_message = []
    for i in range(len(m)):

        if m[i] == 1:
            m_s = np.ones(len(t))
            invm_s = np.zeros(len(t))
        else:
            m_s = np.zeros(len(t))
            invm_s = np.ones(len(t))
        message.extend(m_s)
        not_message.extend(invm_s)
    return message , not_message

N = 8
Tb = 1
T = 1*N
t = np.arange(0, T, T/800)

m = random_bits_array(N)

fig , ax = plt.subplots(2, 1, figsize=(15, 5))
fig.subplots_adjust(hspace=0.5)
ax[0].stem(m)
ax[0].set_title('binary data')
ax[0].set_xlabel('n------>')
ax[0].set_ylabel('b(n)')
message , not_message = message_signal(m,T)
print(len(message), len(not_message))
```

```
ax[1].plot(t,message, 'r')
ax[1].set_title('message signal-1')
ax[1].set_xlabel('t------>')
ax[1].set_ylabel('c1(t)')
```



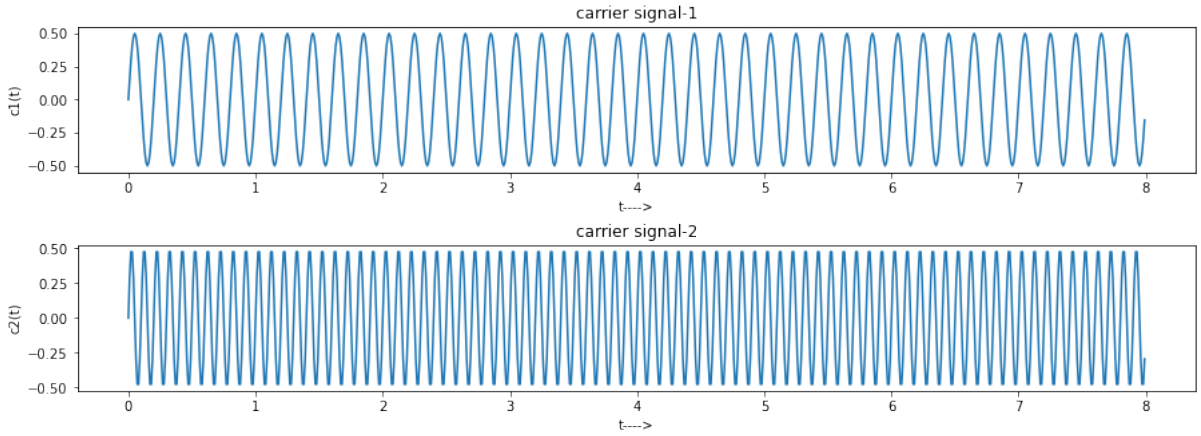
Hình 2: Tín hiệu bit và tín hiệu xung

6.2 Tạo tín hiệu sóng mang

Tạo hai sóng mang cho các tín hiệu tương ứng

- Với giá trị bit 1 : $\sqrt{\frac{2}{Tb}}cos(2\pi f_1 t)$
- Với giá trị bit 0 : $\sqrt{\frac{2}{Tb}}cos(2\pi f_2 t)$

```
fc1 = 5
fc2 = 10
c1 = np.sqrt(2/Tb) * np.sin(2 * np.pi * fc1 * t)
c2 = np.sqrt(2/Tb) * np.sin(2 * np.pi * fc2 * t)
fig, ax = plt.subplots(2, 1, figsize=(15, 5))
fig.subplots_adjust(hspace=0.5)
ax[0].plot(t, c1)
ax[0].set_title('carrier signal-1')
ax[0].set_xlabel('t------>')
ax[0].set_ylabel('c1(t)')

ax[1].plot(t, c2)
ax[1].set_title('carrier signal-2')
ax[1].set_xlabel('t------>')
ax[1].set_ylabel('c2(t)')
```

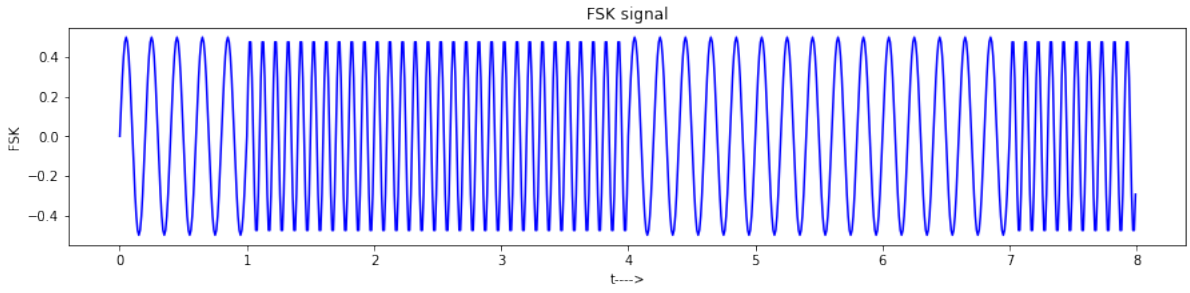


Hình 3: Tín hiệu sóng mang

6.3 Tín kiệu điều chế FSK

Gán tín hiệu với bit tương ứng. Bit 1 có tần số f_1, bit 0 có tần số f_2. Gán fsk là mảng chứa giá trị của tín hiệu sau khi điều chế

```
fig, ax = plt.subplots(1, 1, figsize=(15, 3))
def modulation(message_signal,not_message_signal):
    return message_signal*c1+ not_message*c2
fsk = modulation(message,not_message)
ax.figsize=(15, 5)
ax.plot(t,fsk, 'b')
ax.set_title('FSK signal')
ax.set_xlabel('t------>')
ax.set_ylabel('FSK')
```

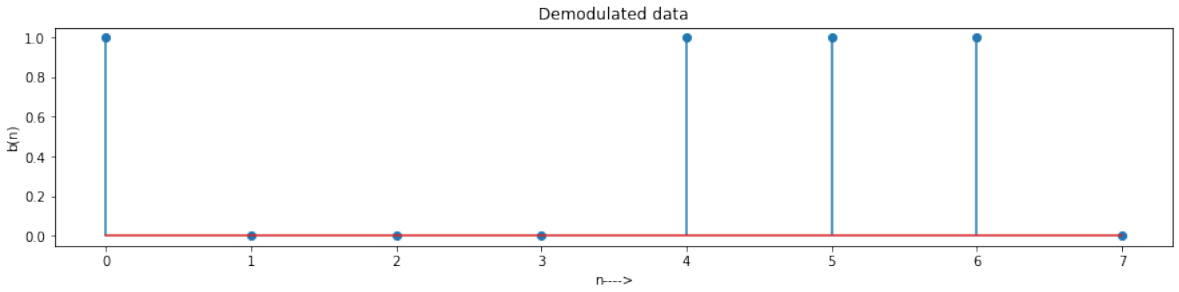


Hình 4: Tín hiệu điều chế

6.4 Giải điều chế

```
fig, ax = plt.subplots(1, 1, figsize=(15, 3))
def demodulation(fsk,N):
```

```
    start = 0
    end = 99
    demod = np.zeros(N)
    for i in range(N):
        x1 = np.sum(c1[start:end] * fsk[start:end])
        x2 = np.sum(c2[start:end] * fsk[start:end])
        x = x1 - x2
        if x > 0:
            demod[i] = 1
        else:
            demod[i] = 0
        start += 100
        end += 100
    return demod
demod = demodulation(fsk,N)
ax.stem(demod)
ax.set_title("Demodulated data")
ax.set_xlabel("n------>")
ax.set_ylabel("b(n)")
```



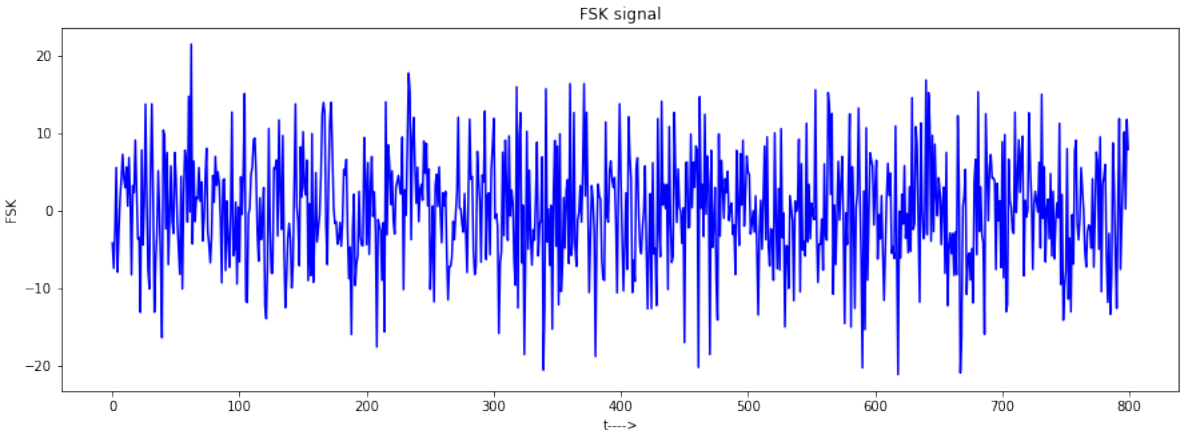
Hình 5: Giải điều chế

6.5 Điều chế và giải điều chế FSK dưới tác động của nhiễu Gausian

Dùng hàm random.random trong python để tạo phân bố chuẩn Gauss

```
def white_noise(rho, sr, n, mu=0):
    sigma = rho * np.sqrt(sr/2)
    noise = np.random.normal(mu, sigma, n)
    return noise
gauss = fsk + white_noise(1,1,800)
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
ax.plot(gauss, 'b')
ax.set_title('FSK signal')
```

```
ax.set_xlabel('t------>')
ax.set_ylabel('FSK')
```



Hình 6: Tín hiệu thêm nhiễu Gauss

7 Tính xác suất lỗi bit

Chiếu hai tín hiệu nên không gian trực chuẩn ta có:

$$M = \{s_1(1,0), s_2(0,1)\}$$

Từ đó ta có vùng Voronoi được định nghĩa như sau:

$$V(s_1) = \{p = (p_1, p_2), p_1 \geq p_2\}$$

$$V(s_2) = \{p = (p_1, p_2), p_2 > p_1\}$$

Ta có:

$$P_s(e) = \frac{1}{2}(P_s(e|s_T = s1) + P_s(e|s_T = s_2) \quad (1)$$

Ta cần tính:

$$P_s(e|s_T = s_1) = P(p \in s_2|s_T = s_1) = P(p_2 > p_1|s_T = s_1)$$

Mà $p_1 = s_1 + n_1$, $p_2 = s_2 + n_2$ và $s_1 = s_2 = 1$ suy ra:

$$P_s(e|s_T = s_1) = P(n_2 > n_1) = P(n_2 - n_1 > 0)$$

$$\rightarrow P_s(e|s_T = s_1) = \frac{1}{2}erfc(0)$$

Tương tụ ta có $P_s(e|s_T = s_2) = \frac{1}{2}erfc(0)$
Thay vào (1) ta có $P_s(e) = \frac{1}{2}erfc(0)$
Vậy xác suất lỗi bit trong AWGN là: $P_s(e) = \frac{1}{2}erfc(0)$

8 Trích Nguồn

1) Tạ Hải Tùng, Slide Nhập môn Kỹ thuật Truyền thông. 2018.
2) Mau-Chung Frank Chang's Lab, "Analysis of Noncoherent ASK ModulationBased RF-Interconnect for Memory Interface" in IEEE Journal on Emerging and Selected Topics in Circuits and Systems.
3) "Digital Communications: Fundamentals and Applications" của Bernard Sklar
4) "Introduction to Digital Communication" của Rodger E. Ziemer và Roger L. Peterson
5) "Principles of Digital Communication" của Robert G. Gallager
6) "Digital Communication" của Simon Haykin
7) "Communication Systems Engineering" của John G. Proakis và Masoud Salehi
3)