PCM 信号的产生和接收仿真

【实验目的】

1. 理解脉冲编码调制（PCM）的基本原理，掌握模拟信号数字化的完整过程。
2. 掌握抽样定理，理解抽样频率与信号带宽之间的关系。
3. 理解均匀量化与非均匀量化（A 律和 μ 律压扩）的原理及其差异。
4. 掌握 PCM 编码与解码的实现方法，理解量化比特数对信号质量的影响。
5. 通过眼图分析数字通信系统的传输质量，理解码间干扰的影响。
6. 分析不同信噪比条件下 PCM 系统的误码率性能。

【实验原理】

1. PCM 编码基本原理

PCM（Pulse Code Modulation，脉冲编码调制）是将模拟信号转换为数字信号的一种重要方法。PCM 编码过程包括三个基本步骤：抽样、量化和编码。

2. 抽样

根据 Nyquist 采样定理，若模拟信号 $m(t)$ 的最高频率为 f_m，则抽样频率 f_s 必须满足：

$$f_s \geq 2f_m$$

抽样后得到的离散信号为：

$$m_s(nT_s) = m(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

其中 $T_s = 1/f_s$ 为抽样周期。

3. 量化

3.1 均匀量化

均匀量化将信号幅度范围等分为 L 个量化级，量化间隔为：

$$\Delta = \frac{V_{max} - V_{min}}{L}$$

均匀量化的量化噪声功率为：

$$N_q = \frac{\Delta^2}{12}$$

3.2 非均匀量化

为了提高小信号的信噪比，实际 PCM 系统采用非均匀量化。常用的压扩特性包括 A 律和 μ 律。

A 律压扩特性（主要用于欧洲和中国）：

$$y = \begin{cases} \frac{Ax}{1+\ln A}, & 0 \le |x| \le \frac{1}{A} \\ \frac{1+\ln(A|x|)}{1+\ln A}, & \frac{1}{A} < |x| \le 1 \end{cases}$$

其中 $A = 87.6$ 为压扩系数。

μ 律压扩特性（主要用于北美和日本）：

$$y = \frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \cdot \mathrm{sgn}(x)$$

其中 $\mu = 255$ 为压扩系数。

4. 编码

量化后的信号需要转换为二进制码字。若量化级数为 L，则所需比特数为：

$$n = \lceil \log_2 L \rceil$$

常用的编码方式包括自然二进制码和折叠二进制码。

5. PCM 解码

接收端的解码过程是编码的逆过程，包括：

1. 将接收到的二进制码字转换为量化值
2. 进行扩张（非均匀量化时）
3. 通过低通滤波器恢复模拟信号

6. 眼图

眼图是评估数字通信系统性能的重要工具。通过将接收信号以符号周期为单位叠加显示，可以观察：

- 眼开度：反映抗噪声能力
- 眼宽度：反映定时误差容限
- 交叉点散度：反映码间干扰程度

7. 误码率

PCM 系统的误码率受信噪比影响。对于 AWGN 信道，理论误码率为：

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

其中 E_b 为每比特能量，N_0 为噪声功率谱密度。

【Matlab 仿真】

主要参数说明：

- `fs`：抽样频率，设置为 8000 Hz（电话语音标准）
- `f0`：输入正弦波信号频率，设置为 200 Hz
- `n_bits`：量化比特数，分别测试 4、6、8 比特
- `A`：A 律压扩系数，取值 87.6
- `mu`：μ 律压扩系数，取值 255
- `SNR_dB`：信噪比范围，0-20 dB

仿真代码：

```
%% PCM信号的产生和接收仿真
clear; clc; close all;

%% 参数设置
fs = 8000;          % 抽样频率 (Hz)
f0 = 200;           % 信号频率 (Hz)
T = 0.05;           % 仿真时长 (s)
t = 0:1/fs:T-1/fs;  % 时间向量
N = length(t);

% 原始模拟信号（正弦波）
signal = sin(2*pi*f0*t);

%% Case 1: 抽样过程演示
figure('Name', 'Case 1: 抽样过程', 'Position', [100 100 1000 600]);

% 高分辨率时间轴用于显示原始信号
t_high = 0:1/(fs*50):T-1/(fs*50);
signal_high = sin(2*pi*f0*t_high);

subplot(2,1,1);
plot(t_high*1000, signal_high, 'b-', 'LineWidth', 1);
hold on;
stem(t*1000, signal, 'r', 'LineWidth', 1, 'MarkerSize', 4);
xlabel('时间 (ms)');
ylabel('幅度');
title('原始信号与抽样信号');
legend('原始模拟信号', '抽样信号');
grid on;

subplot(2,1,2);
```

```
% 显示抽样频谱
N_fft = 1024;
f_axis = (-N_fft/2:N_fft/2-1)*(fs/N_fft);
signal_padded = [signal, zeros(1, N_fft-N)];
spectrum = fftshift(abs(fft(signal_padded)));
plot(f_axis, spectrum/max(spectrum), 'b', 'LineWidth', 1);
xlabel('频率 (Hz)');
ylabel('归一化幅度');
title('抽样信号频谱');
grid on;
xlim([-fs/2, fs/2]);

%% Case 2: 均匀量化（不同比特数）
figure('Name', 'Case 2: 均匀量化', 'Position', [100 100 1200 800]);

n_bits_array = [4, 6, 8];
for i = 1:3
    n_bits = n_bits_array(i);
    L = 2^n_bits;  % 量化级数

    % 均匀量化
    delta = 2 / L;  % 量化间隔（信号范围-1到1）
    quantized = round(signal / delta) * delta;
    quantized = max(min(quantized, 1-delta/2), -1+delta/2);

    % 量化误差
    quant_error = signal - quantized;

    % 计算信噪比
    signal_power = mean(signal.^2);
    noise_power = mean(quant_error.^2);
    SNR = 10*log10(signal_power/noise_power);

    % 绘制波形
    subplot(3,2,2*i-1);
    plot(t*1000, signal, 'b-', 'LineWidth', 1);
    hold on;
    stairs(t*1000, quantized, 'r-', 'LineWidth', 1);
    xlabel('时间 (ms)');
    ylabel('幅度');
    title([num2str(n_bits), '比特均匀量化 (SNR=', num2str(SNR,'%.2f'), ' dB)']);
    legend('原始信号', '量化信号');
    grid on;

    % 绘制量化误差
    subplot(3,2,2*i);
    plot(t*1000, quant_error, 'g-', 'LineWidth', 1);
    xlabel('时间 (ms)');
```

```
    ylabel('幅度');
    title([num2str(n_bits), '比特量化误差']);
    grid on;
    ylim([-0.3, 0.3]);
end

%% Case 3: 非均匀量化（A律和mu律压扩）
figure('Name', 'Case 3: 非均匀量化', 'Position', [100 100 1200 600]);

n_bits = 8;
L = 2^n_bits;

% A律压扩参数
A = 87.6;

% mu律压扩参数
mu = 255;

% A律压缩
x = signal;
y_A = zeros(size(x));
for k = 1:length(x)
    if abs(x(k)) <= 1/A
        y_A(k) = (A*abs(x(k)))/(1+log(A)) * sign(x(k));
    else
        y_A(k) = (1+log(A*abs(x(k))))/(1+log(A)) * sign(x(k));
    end
end

% mu律压缩
y_mu = log(1+mu*abs(x))/log(1+mu) .* sign(x);

% 均匀量化压缩后的信号
delta = 2/L;
quant_A = round(y_A/delta)*delta;
quant_mu = round(y_mu/delta)*delta;

% A律扩张
expanded_A = zeros(size(quant_A));
for k = 1:length(quant_A)
    if abs(quant_A(k)) <= 1/(1+log(A))
        expanded_A(k) = abs(quant_A(k))*(1+log(A))/A * sign(quant_A(k));
    else
        expanded_A(k) = exp(abs(quant_A(k))*(1+log(A))-1)/A * sign(quant_A(k));
    end
end

% mu律扩张
```

```
expanded_mu = ((1+mu).^abs(quant_mu)-1)/mu .* sign(quant_mu);

% 计算SNR
error_A = signal - expanded_A;
error_mu = signal - expanded_mu;
SNR_A = 10*log10(mean(signal.^2)/mean(error_A.^2));
SNR_mu = 10*log10(mean(signal.^2)/mean(error_mu.^2));

subplot(2,2,1);
plot(t*1000, signal, 'b-', 'LineWidth', 1);
hold on;
plot(t*1000, expanded_A, 'r--', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title(['A律压扩 (SNR=', num2str(SNR_A,'%.2f'), ' dB)']);
legend('原始信号', '解码信号');
grid on;

subplot(2,2,2);
plot(t*1000, signal, 'b-', 'LineWidth', 1);
hold on;
plot(t*1000, expanded_mu, 'r--', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title(['\mu律压扩 (SNR=', num2str(SNR_mu,'%.2f'), ' dB)']);
legend('原始信号', '解码信号');
grid on;

subplot(2,2,3);
plot(t*1000, error_A, 'g-', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title('A律量化误差');
grid on;

subplot(2,2,4);
plot(t*1000, error_mu, 'g-', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title('\mu律量化误差');
grid on;

%% Case 4: PCM编码与解码
figure('Name', 'Case 4: PCM编码解码', 'Position', [100 100 1200 600]);

n_bits = 8;
L = 2^n_bits;

```

```
% 量化
delta = 2/L;
quantized_index = round((signal + 1) / delta);
quantized_index = max(min(quantized_index, L-1), 0);

% 编码（转换为二进制）- 使用内置函数替代de2bi
binary_str = dec2bin(quantized_index, n_bits);
binary_code = zeros(length(quantized_index), n_bits);
for idx = 1:length(quantized_index)
    binary_code(idx, :) = binary_str(idx, :) - '0';
end

% 生成PCM码流
pcm_stream = reshape(binary_code', 1, []);

% 解码 - 使用内置函数替代bi2de
received_index = zeros(1, size(binary_code, 1));
for idx = 1:size(binary_code, 1)
    received_index(idx) = bin2dec(num2str(binary_code(idx, :), '%d'));
end
decoded_signal = received_index * delta - 1 + delta/2;

subplot(3,1,1);
plot(t*1000, signal, 'b-', 'LineWidth', 1);
hold on;
stairs(t*1000, decoded_signal, 'r-', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title('原始信号与PCM解码信号');
legend('原始信号', '解码信号');
grid on;

subplot(3,1,2);
% 显示部分PCM码流
t_bits = (0:min(200, length(pcm_stream)-1)) / (n_bits*fs) * 1000;
stairs(t_bits, pcm_stream(1:min(201, length(pcm_stream))), 'k-', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('比特值');
title('PCM码流（部分）');
ylim([-0.2, 1.2]);
grid on;

subplot(3,1,3);
error = signal - decoded_signal;
plot(t*1000, error, 'g-', 'LineWidth', 1);
xlabel('时间 (ms)');
ylabel('幅度');
title('解码误差');
```

```
grid on;

%% Case 5: 眼图分析
figure('Name', 'Case 5: 眼图', 'Position', [100 100 1200 600]);

% 生成随机PCM数据
num_symbols = 1000;
data_bits = randi([0 1], 1, num_symbols);

% 脉冲成形（矩形脉冲）
samples_per_symbol = 10;
pulse = ones(1, samples_per_symbol);
tx_signal = kron(data_bits, pulse);

% 添加不同程度的噪声
SNR_values = [20, 10];

for i = 1:2
    SNR_dB = SNR_values(i);

    % 添加高斯白噪声
    signal_power = mean(tx_signal.^2);
    noise_power = signal_power / (10^(SNR_dB/10));
    noise = sqrt(noise_power) * randn(size(tx_signal));
    rx_signal = tx_signal + noise;

    % 简单低通滤波
    [b, a] = butter(4, 0.5);
    rx_filtered = filter(b, a, rx_signal);

    % 绘制眼图
    subplot(1,2,i);
    eye_length = 2 * samples_per_symbol;
    num_traces = floor(length(rx_filtered) / eye_length) - 1;

    hold on;
    for k = 1:min(num_traces, 100)
        start_idx = (k-1) * eye_length + 1;
        end_idx = start_idx + eye_length - 1;
        if end_idx <= length(rx_filtered)
            t_eye = (0:eye_length-1) / samples_per_symbol;
            plot(t_eye, rx_filtered(start_idx:end_idx), 'b-', 'LineWidth', 0.5);
        end
    end
    hold off;

    xlabel('符号周期');
    ylabel('幅度');
```

```
272     title(['眼图 (SNR = ', num2str(SNR_dB), ' dB)']);
273     grid on;
274 end
275 
276 %% Case 6: 误码率分析
277 figure('Name', 'Case 6: 误码率', 'Position', [100 100 800 600]);
278 
279 SNR_dB_range = 0:2:20;
280 BER_simulated = zeros(size(SNR_dB_range));
281 num_bits = 100000;
282 
283 for i = 1:length(SNR_dB_range)
284     SNR_dB = SNR_dB_range(i);
285 
286     % 生成随机比特
287     tx_bits = randi([0 1], 1, num_bits);
288 
289     % BPSK调制
290     tx_symbols = 2*tx_bits - 1;
291 
292     % 添加高斯白噪声
293     EbN0 = 10^(SNR_dB/10);
294     noise_std = sqrt(1/(2*EbN0));
295     noise = noise_std * randn(size(tx_symbols));
296     rx_symbols = tx_symbols + noise;
297 
298     % 判决
299     rx_bits = rx_symbols > 0;
300 
301     % 计算误码率
302     BER_simulated(i) = sum(tx_bits ~= rx_bits) / num_bits;
303 end
304 
305 % 理论误码率
306 EbN0_linear = 10.^(SNR_dB_range/10);
307 BER_theoretical = 0.5 * erfc(sqrt(EbN0_linear));
308 
309 semilogy(SNR_dB_range, BER_theoretical, 'b-', 'LineWidth', 2);
310 hold on;
311 semilogy(SNR_dB_range, BER_simulated, 'ro', 'MarkerSize', 8, 'LineWidth', 1.5);
312 xlabel('E_b/N_0 (dB)');
313 ylabel('误码率 (BER)');
314 title('PCM系统误码率性能');
315 legend('理论值', '仿真值');
316 grid on;
317 ylim([1e-6, 1]);
318 
319 disp('仿真完成! ');
```

仿真结果截图

Case 1：抽样过程

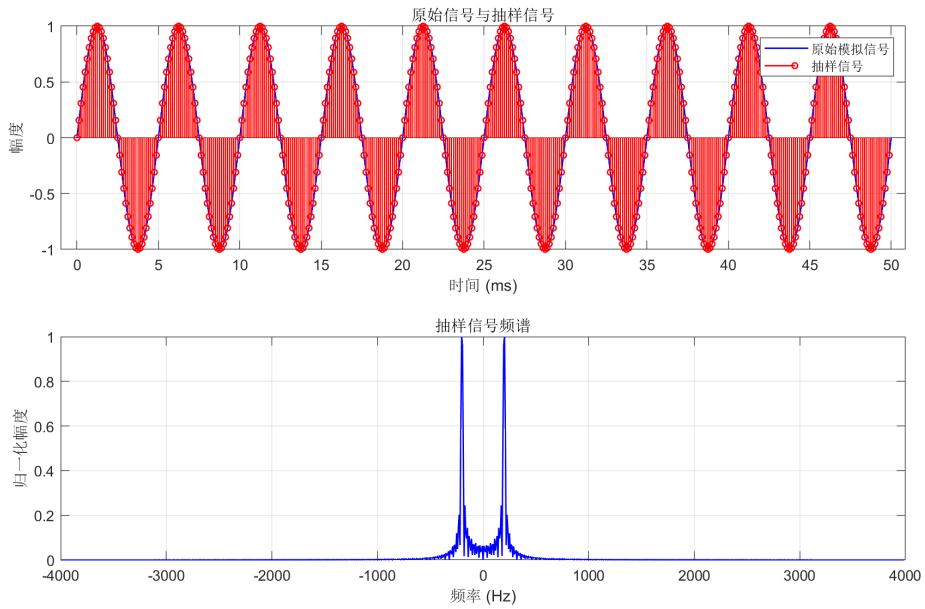

图 1: 抽样过程

Case 2：均匀量化（不同比特数）

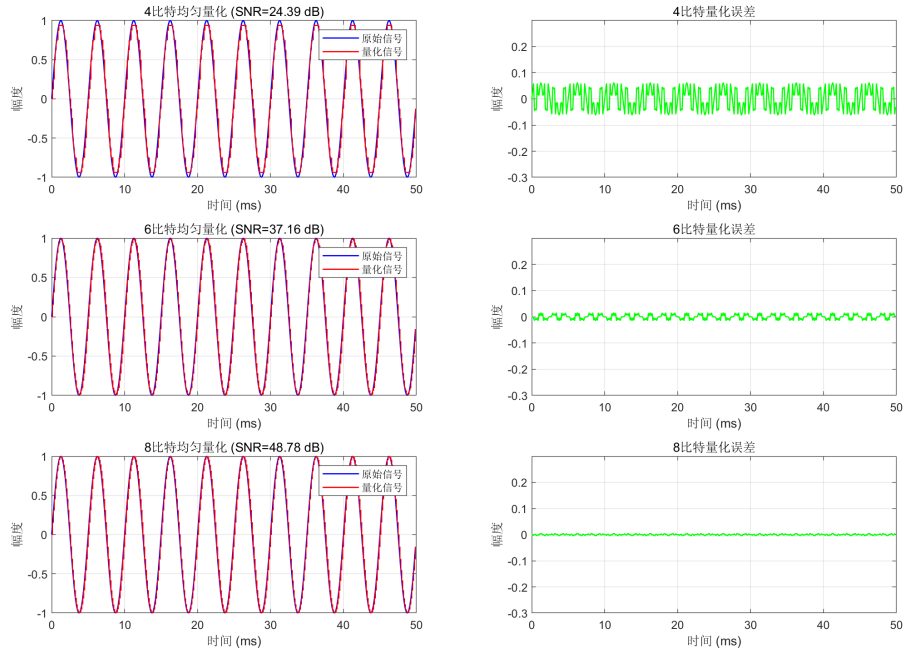

图 2: 均匀量化结果

Case 3：非均匀量化（A 律和 μ 律压扩）

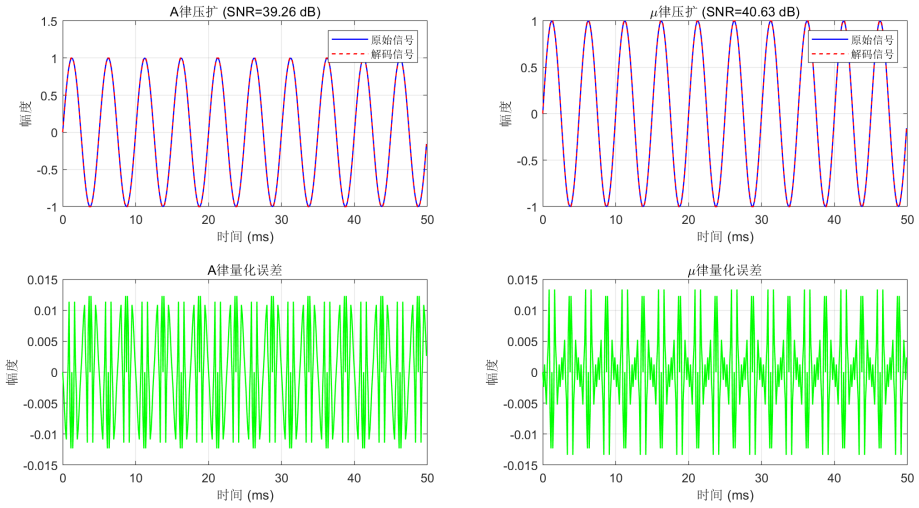

图 3: 非均匀量化结果

Case 4：PCM 编码与解码

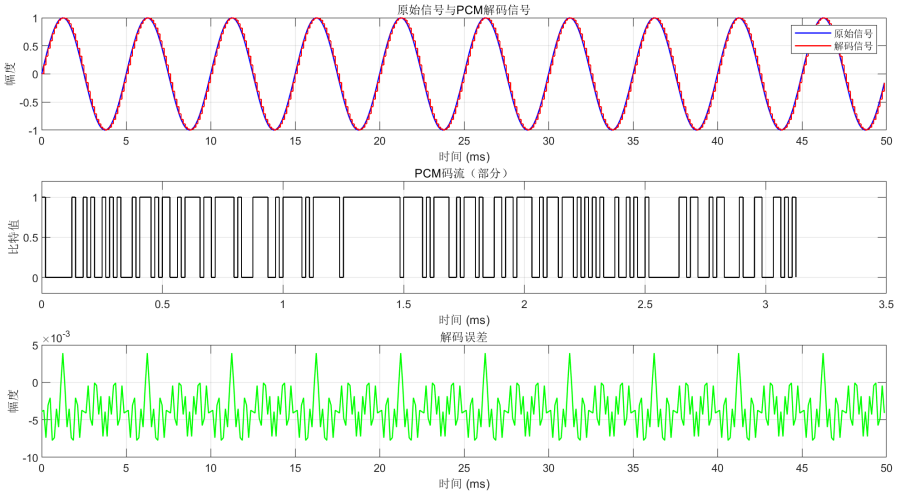

图 4: PCM 编码解码结果

Case 5：眼图分析

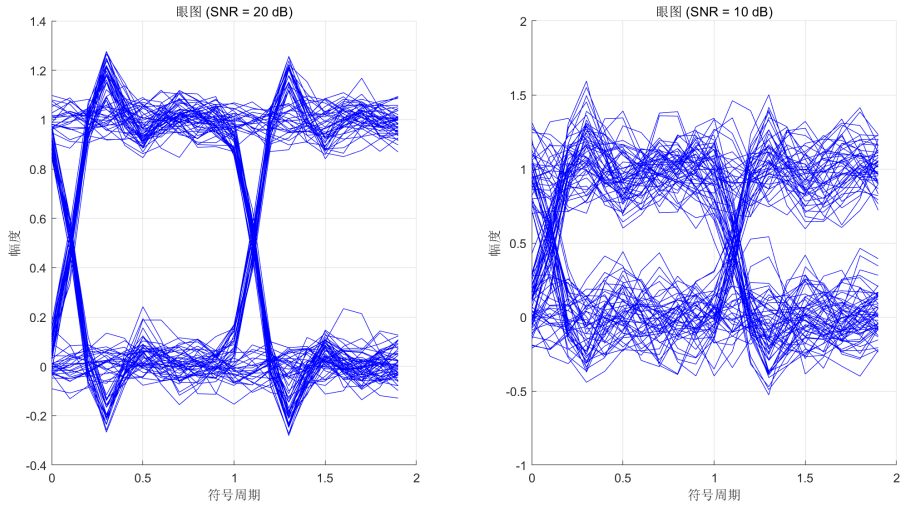

图 5: 眼图

Case 6：误码率分析

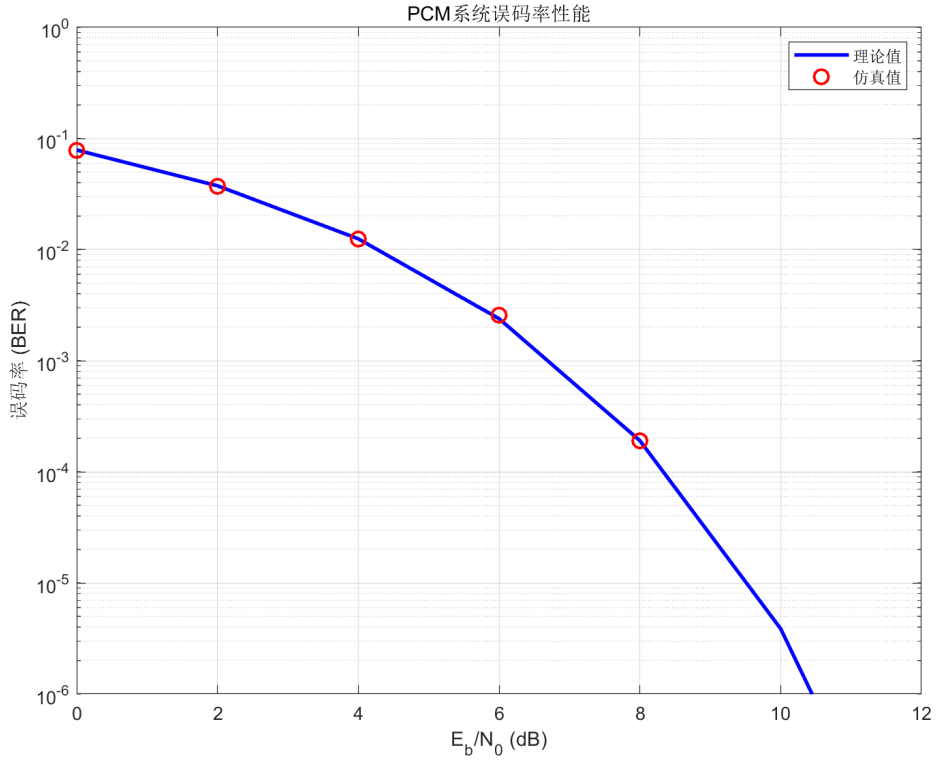

图 6: 误码率曲线

【仿真结果分析】

1. 抽样过程分析

通过 Case 1 的仿真结果可以观察到，当抽样频率 $f_s = 8000$ Hz 时，对于频率为 200 Hz 的正弦信号，满足奈奎斯特抽样定理的要求（$f_s > 2f_0$）。抽样后的离散信号能够完整保留原始信号的信息，频谱分析显示信号能量集中在 200 Hz 附近，验证了抽样定理的正确性。

2. 均匀量化分析

从 Case 2 的仿真结果可以看出，量化比特数对信号质量有显著影响。4 比特量化时，量化级数仅为 16 级，量化误差明显，信噪比较低；6 比特量化时，量化级数增加到 64 级，信号质量有所改善；8 比特量化时，量化级数达到 256 级，量化误差显著减小，信噪比明显提高。理论上，每增加 1 比特，信噪比约提高 6 dB，仿真结果与理论分析吻合。

3. 非均匀量化分析

Case 3 展示了 A 律和 μ 律压扩的效果。相比均匀量化，非均匀量化对小信号的量化精度更高，因为压扩过程将小信号的动态范围扩展，使其占用更多的量化级。A 律压扩（$A = 87.6$）和 μ 律压扩（$\mu = 255$）在相同比特数下都能获得比均匀量化更好的小信号信噪比性能，这在语音信号处理中尤为重要，因为语音信号大部分时间处于小幅度状态。

4. PCM 编码解码分析

Case 4 演示了完整的 PCM 编码解码过程。通过将量化值转换为 8 比特二进制码，生成了数字 PCM 码流。解码过程能够准确恢复量化信号，解码误差即为量化误差。PCM 编码的优点是抗干扰能力强，便于数字信号处理和存储传输。

5. 眼图分析

Case 5 的眼图分析揭示了信噪比对数字传输质量的影响。在高信噪比（20 dB）条件下，眼图开度大、眼睛张开明显，表明系统具有良好的抗噪声能力和较大的定时裕量；在低信噪比（10 dB）条件下，眼图开度减小、交叉点散度增加，反映出码间干扰和噪声影响加剧。眼图的" 眼睛" 越大越清晰，表示系统性能越好。

6. 误码率分析

Case 6 展示了 PCM 系统在 AWGN 信道下的误码率性能。仿真结果与理论曲线高度吻合，验证了误码率公式的正确性。当 E_b/N_0 增加时，误码率显著下降。在 $E_b/N_0 = 10$ dB 时，误码率约为 10^{-4} 量级；在 $E_b/N_0 = 15$ dB 时，误码率降至 10^{-6} 量级以下。这表明增加发射功率或降低噪声水平可以有效提高系统的传输可靠性。

7. 总结

通过本次仿真实验，我们系统地研究了 PCM 信号的产生和接收过程。实验结果表明：

1. 抽样频率必须满足奈奎斯特准则才能避免混叠
2. 量化比特数直接影响信号质量，每增加 1 比特，信噪比约提高 6 dB
3. 非均匀量化能够改善小信号的量化性能，适用于语音信号处理
4. 眼图是评估数字传输系统性能的有效工具
5. 误码率性能与信噪比密切相关，符合理论预测

实验成员：张韫译萱、杨宸源、张凯铭

日期：2025 年 12 月 2 日