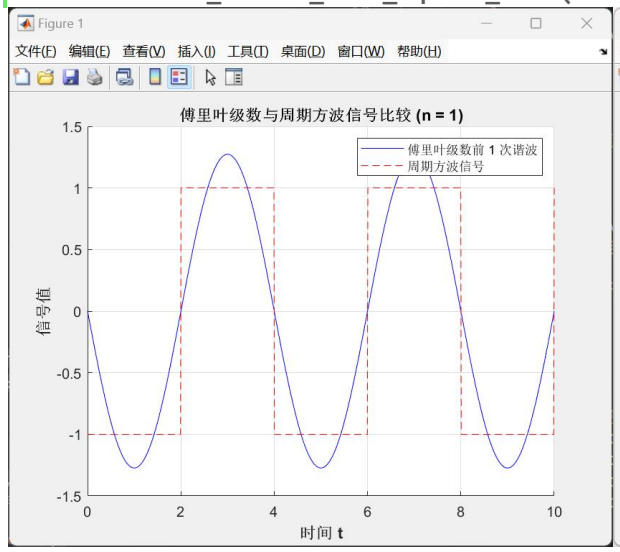
实验一 周期信号的分解与合成

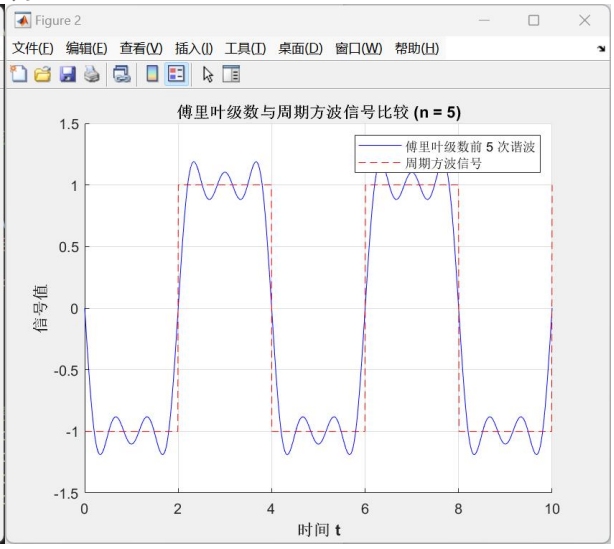
一、实验代码及运行结果

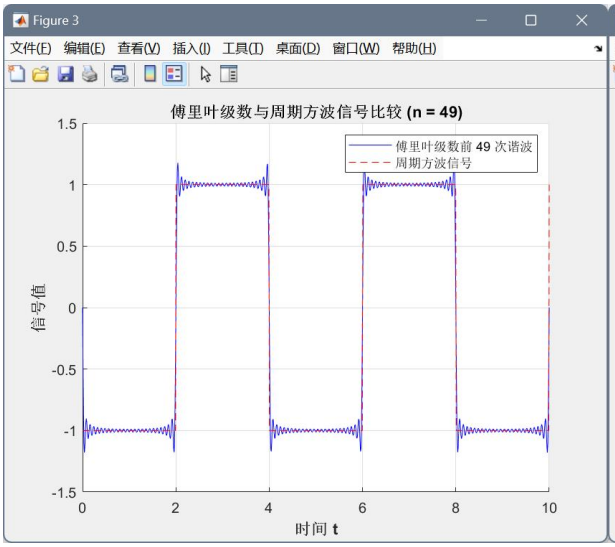
1. 观察连续周期方波信号的分解与合成

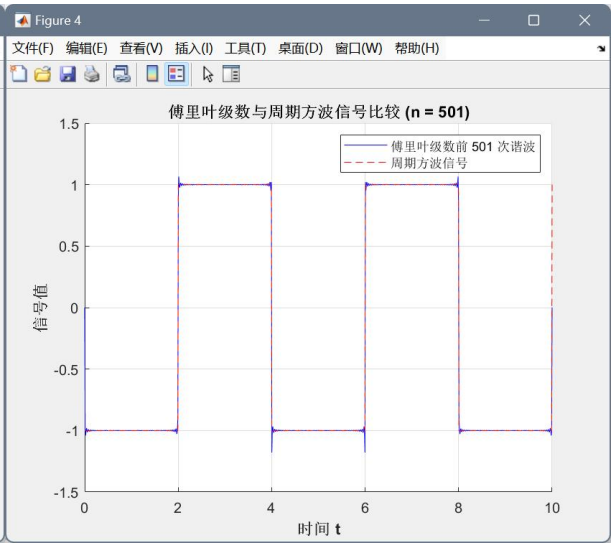
```
% 1. 观察连续周期方波信号的分解与合成

function output = square_wave(t)
    output = zeros(size(t));

    for i = 1:length(t)

        if mod(t(i), 4) >= 2 && mod(t(i), 4) <= 4
            output(i) = 1;
        else
            output(i) = -1;
        end

    end

end

function f_t = fourier_series_with_square_wave(n)
    % n: 傅里叶级数的最高阶数
    t = linspace(0, 10, 1000);
    f_t = zeros(size(t)); % 初始化傅里叶展开信号
    swave = square_wave(t); % 周期方波信号

    % 计算傅里叶级数的前 n 项（只考虑奇数阶次）
    for k = 1:2:n
        f_t = f_t - (4 / (pi * k)) * sin(k * pi * t / 2);
    end

    figure;
    hold on;
    plot(t, f_t, 'b', 'DisplayName', ['傅里叶级数前 ', num2str(n), ' 次谐波']);
    plot(t, swave, 'r--', 'DisplayName', '周期方波信号');
    title(['傅里叶级数与周期方波信号比
较 (n = ', num2str(n), ')'], 'FontWeight', 'bold');
    xlabel('时间 t', 'FontWeight', 'bold');
    ylabel('信号值', 'FontWeight', 'bold');
    legend;
```

```
        grid on;
        hold off;
    end

    fourier_series_with_square_wave(1);
    fourier_series_with_square_wave(5);
    fourier_series_with_square_wave(49);
    fourier_series_with_square_wave(501);
```





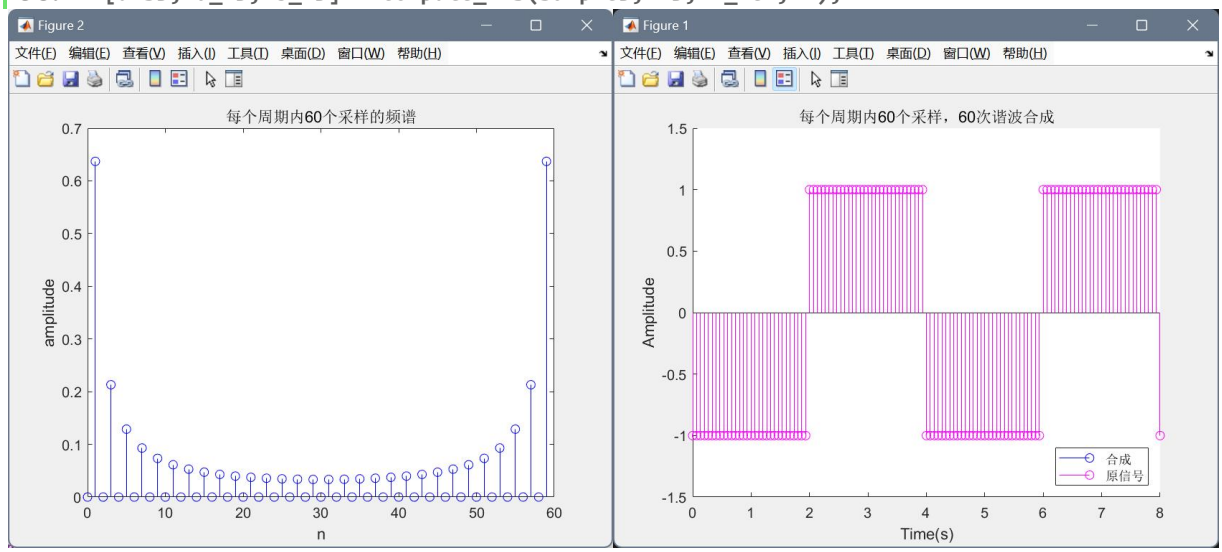




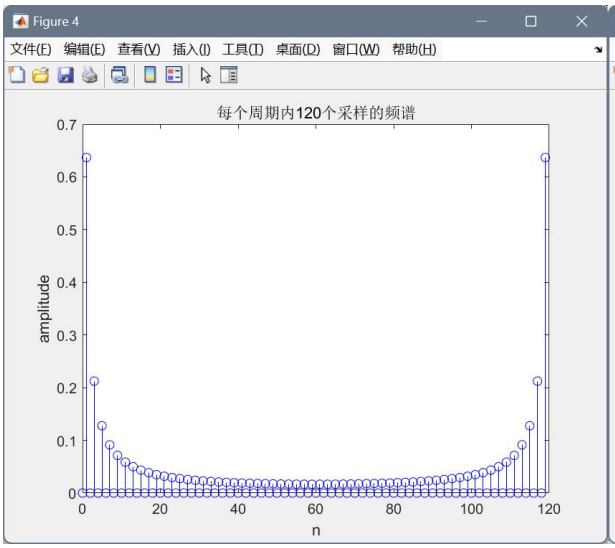
2、观察离散方波信号的分解和合成

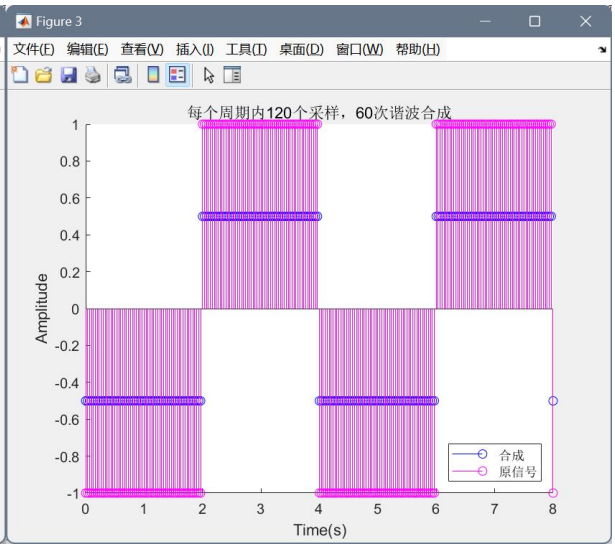
```
% 2. 观察离散周期方波信号的分解与合成
function [dfs, a_k, b_k] = compute_DFS(sample, N, T_len, T)
    % 从给定的样本数据中计算出离散傅里叶级数的系数和对应的信号
    % sample: 采样信号
    % N: 采样点数
    % T_len: 采样信号的长度
    % T: 采样信号的周期
    dfs = zeros(N, (T_len / T) * N);
```

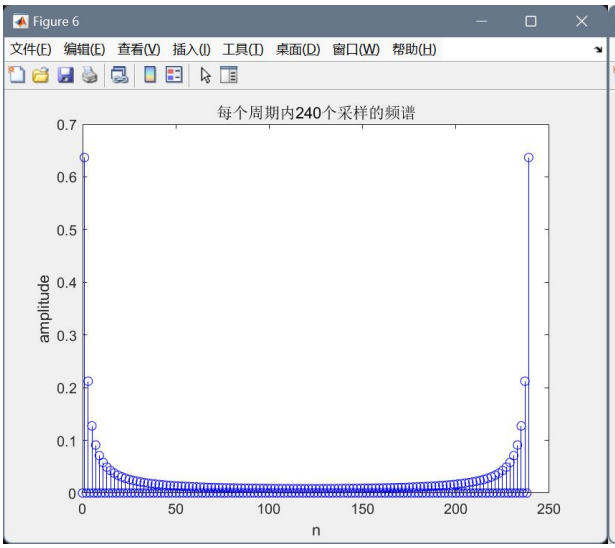
```
        a_k = zeros(1, N);
        b_k = zeros(1, N);

        % 计算 DFS 系数
        for i = 0:N - 1
            a_k(i + 1) = sample(1:N) * cos(2 * pi * i / N * (0:N - 1))' ./ N;
            b_k(i + 1) = -sample(1:N) * sin(2 * pi * i / N * (0:N - 1))' ./ N;
        end

        % 计算 DFS
        for k = 0:N - 1

            for n = 0:(T_len / T) * N
                dfs(k + 1, n + 1) = a_k(k + 1) * cos(2 * pi * k * n / N) - b_k(k + 1) * sin(2 * pi * k * n / N);
            end

        end

    end

    T = 4;
    T_len = 8;
    N1 = 60; N2 = 120; N3 = 240;
    max_order = 60;
    sample1 = square_wave(0:(T / N1):T_len);
    sample2 = square_wave(0:(T / N2):T_len);
    sample3 = square_wave(0:(T / N3):T_len);
    [dfs1, a_k1, b_k1] = compute_DFS(sample1, N1, T_len, T);
    [dfs2, a_k2, b_k2] = compute_DFS(sample2, N2, T_len, T);
    [dfs3, a_k3, b_k3] = compute_DFS(sample3, N3, T_len, T);
```

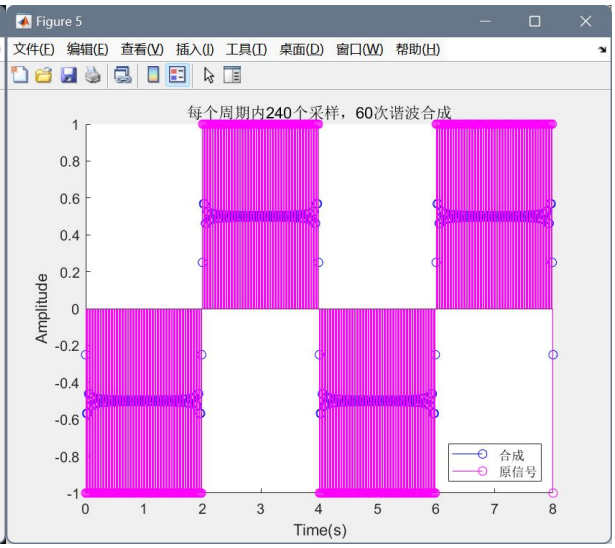


改变每个周期采样的次数，固定谐波合成次数为 60：

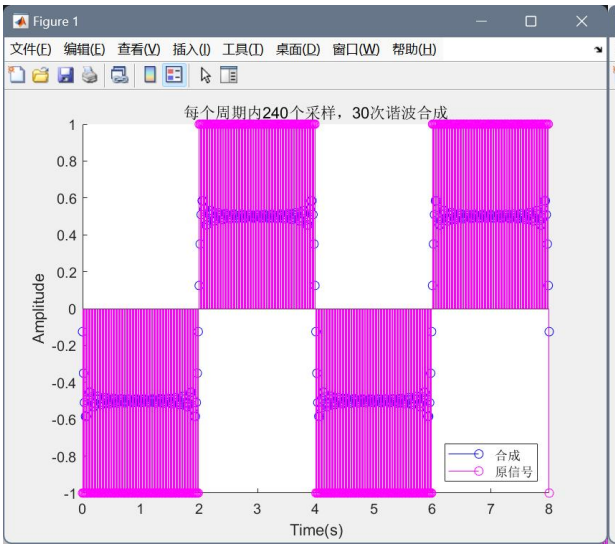


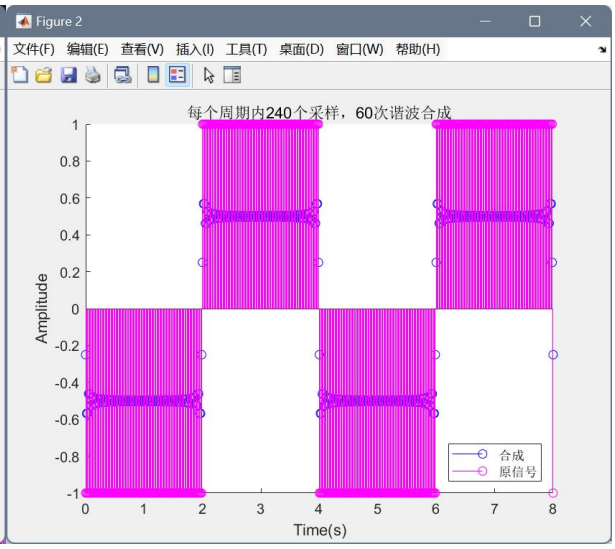


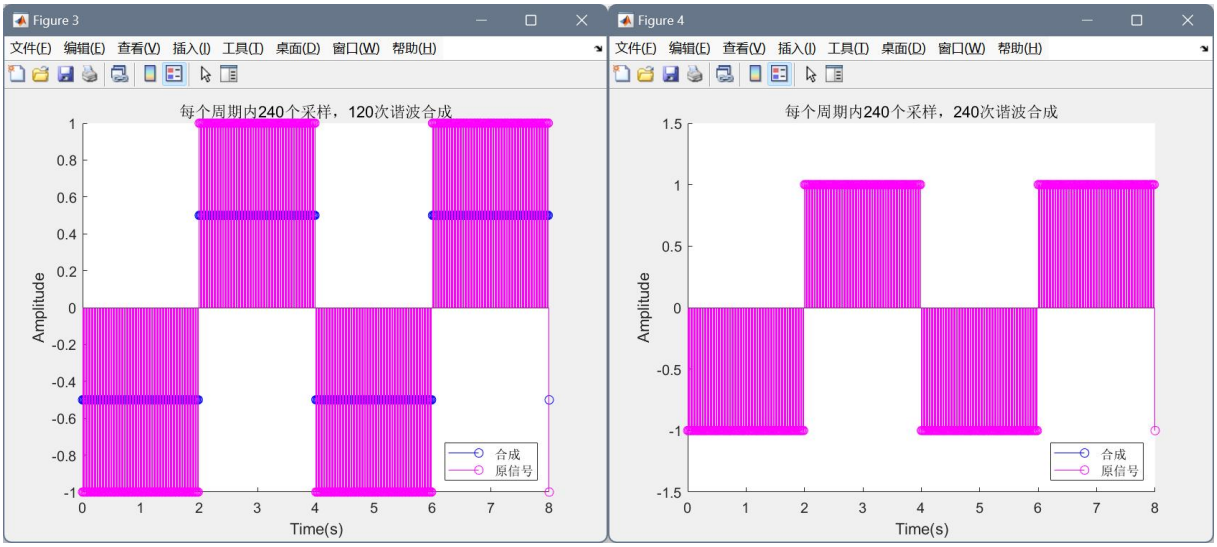




固定每个周期采样的次数为 240，改变谐波合成次数：





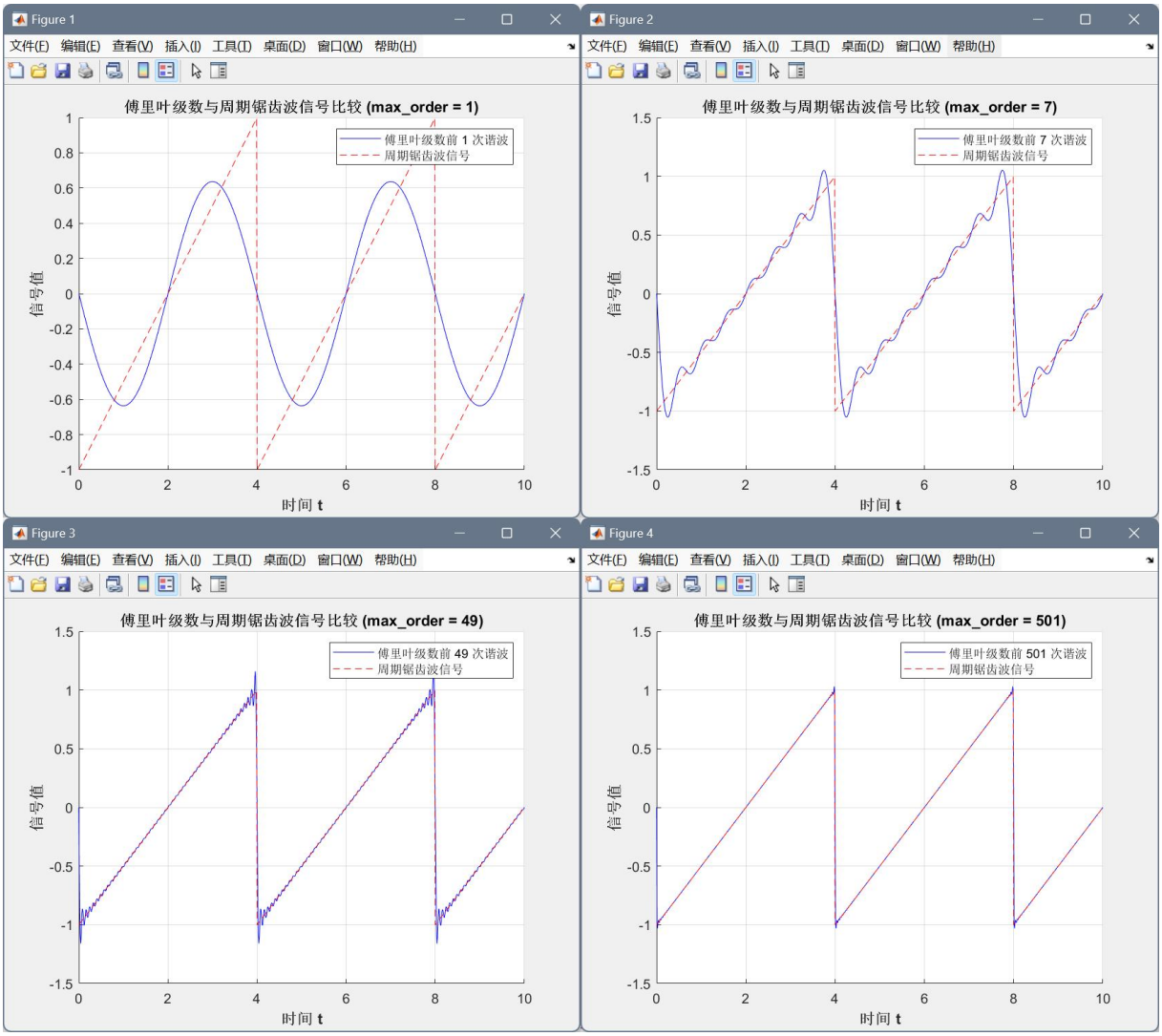


由图像可以看出，当谐波合成次数小于每周期的采样次数时，会有能量损失，当谐波合成次数等于每周期的采样次数时，没有能量损失。

3、观察周期锯齿波信号的分解和合成

```
% 3. 观察连续周期锯齿波信号的分解与合成
function output = jagged_wave(t)
    output = mod(t, 4) / 2 - 1; % 计算锯齿波信号
end

function f_t = fourier_series_with_jagged_wave(max_order)
    % max_order: 傅里叶级数的最高阶数
    T = 4; % 周期
    t = 0:0.01:10; % 时间范围
    jwave = jagged_wave(t); % 周期锯齿波信号

    a_k = zeros(1, max_order); % 傅里叶级数的余弦系数
    b_k = zeros(1, max_order); % 傅里叶级数的正弦系数
    jagged_fourier = zeros(max_order, length(t)); % 傅里叶级数信号

    % 计算 a0
    a0 = integral(@jagged_wave, 0, T) / T;

    % 计算系数 a_k 和 b_k
    for k = 1:max_order
        a_func = @(x) (jagged_wave(x) .* cos(k * 2 * pi / T * x));
        b_func = @(x) (jagged_wave(x) .* sin(k * 2 * pi / T * x));
        a_k(k) = (2 / T) * integral(a_func, 0, T);
        b_k(k) = (2 / T) * integral(b_func, 0, T);
    end
```

```
26.
27.         % 计算傅里叶级数
28.         for i = 1:length(t)
29.
30.             for j = 1:max_order
31.                 jagged_fourier(j, i) = a_k(j) * cos(j * 2 * pi / T * t(i)) + b_k(j) *
    sin(j * 2 * pi / T * t(i));
32.             end
33.
34.         end
35.
36.         % 计算总的傅里叶级数信号
37.         f_t = a0 + sum(jagged_fourier, 1); % 计算总信号，包括常数项 a0
38.
39.         figure;
40.         hold on;
41.         plot(t, f_t, 'b', 'DisplayName', ['傅里叶级数前 ', num2str(max_order), ' 次谐波
    ']);
42.         plot(t, jwave, 'r--', 'DisplayName', '周期锯齿波信号');
43.         title(['傅里叶级数与周期锯齿波信号比
    较 (max\_order = ', num2str(max_order), ')'], 'FontWeight', 'bold');
44.         xlabel('时间 t', 'FontWeight', 'bold');
45.         ylabel('信号值', 'FontWeight', 'bold');
46.         legend;
47.         grid on;
48.         hold off;
49.     end
50.
51.     fourier_series_with_jagged_wave(1);
52.     fourier_series_with_jagged_wave(7);
53.     fourier_series_with_jagged_wave(49);
54.     fourier_series_with_jagged_wave(501);
```



二、实验分析

1. 观察连续周期方波信号的分解与合成

通过图像可以清晰地观察到周期性方波信号的傅里叶级数分解过程，以及利用不同阶次的谐波来逐步重建原始方波信号的具体情况。随着所使用谐波数量的增加，合成信号与理想状态下的方波信号之间的相似度提高，能够更加精细地再现方波的形态特征。

在进行这种重建过程中会出现“吉布斯现象”。具体表现为在原函数存在不连续性的位置附近出现过冲或下冲的现象，形成一个固定的超调量。这个超调量大约为原始跳跃幅度的 9% 左右，并不会随着谐波次数的无限增加而消失，而是会在不连续点两侧形成一个小范围内的振荡区域。

这种现象揭示了使用有限项傅里叶级数逼近非光滑函数时的一个固有限制。

2、观察离散方波信号的分解和合成

由图可以观察到不同采样间隔得到的 60 次谐波合成图像和原图像的对比。其中 60 次采样

时，合成信号的图像与原图像重合导致合成信号图像无法看到。

当每个周期内进行 120 次采样且使用 60 次谐波合成时，合成的信号幅度值约为原信号的一半，从频谱中可以看到，高次谐波仍然具有很大的幅度值，当只使用 60 次谐波时，高次谐波的幅度值无法体现，因此造成了幅度的变化。

由图像可以看出，当谐波合成次数小于每周期的采样次数时，会有能量损失，当谐波合成次数等于每周期的采样次数时，没有能量损失。

3、观察周期锯齿波信号的分解和合成

可以观察到周期锯齿信号的分解与合成情况，观察到其规律与连续周期方波信号的分解与合成的规律一致。

三、思考题

1、结合理论与实验结果，简述连续周期信号频谱的特点。

答：连续周期信号的频谱为离散非周期信号。

2、结合理论与实验结果，简述离散周期信号频谱的特点。

答：离散周期信号的频谱特点为离散周期信号。

总结：

时域	频域
周期性	离散性
离散性	周期性
非周期性	连续性
连续性	非周期性

3、以周期矩形脉冲信号为例分析：当信号的周期 T 和脉冲宽度 τ 发生变化的时候，信号的频谱将如何变化？离散时间矩形周期信号当采样间隔发生变化时，信号的频谱会如何变化？

答：设周期脉冲信号的周期T为脉冲宽度为τ，则它的复傅里叶系数为：

$$F_n = \frac{E\tau}{T_1} Sa(\frac{nw_1\tau}{2})$$

傅里叶级数为：

$$f(t)=\frac{E\tau}{T_1}\sum_{n=-\infty}^{\infty}Sa(\frac{nw_1\tau}{2})e^{jnw_1t}=\frac{E\tau}{T_1}+\frac{2E\tau}{T_1}\sum_{n=1}^{\infty}[Sa(\frac{n\pi\tau}{T_1})\cos(n\omega_1t)]$$

由此可知，频谱的幅度随T增大而减小，随τ增大而增大。

由实验结果中图片可知，离散矩形周期信号在采样间隔变大时一个频谱的周期长度增加，而幅度值基本不变。

实验二 时域采样与频域采样

一、实验代码及运行结果

定义信号：

```
% 给定模拟信号 x(t) = A * exp(-alpha * t) * sin(omega1 * t) * u(t)
function output = signal1(input)
    A = 444.128;
    alpha = 50 * sqrt(2) * pi;
    omega1 = 50 * sqrt(2) * pi;
    output = zeros(size(input));
    output(input >= 0) = A .* exp(-alpha .* input(input >= 0)) .* sin(omega1 .* input(input >= 0));
end

% 给定离散信号 x(n) = n + 1, 0 <= n <= 13; 27 - n, 14 <= n <= 26, 0, 其他
function output = signal2(n)
    output = zeros(size(n));

    idx1 = n >= 0 & n <= 13;
    output(idx1) = n(idx1) + 1;

    idx2 = n >= 14 & n <= 26;
    output(idx2) = 27 - n(idx2);
end
```

1、时域采样

绘制原连续信号的频谱：

```
syms t w

A = 444.128; alpha = 50 * sqrt(2) * pi; omega1 = 50 * sqrt(2) * pi;

x_t = A * exp(-alpha * t) * sin(omega1 * t) * heaviside(t);

X_w = simplify(fourier(x_t, t, w));

f = linspace(-200, 200, 1000);
X_val = double(subs(X_w, w, f));
figure;
plot(f, abs(X_val), 'color', 'blue');
title('原连续信号频谱');
xlabel('频率 (rad/s)');
ylabel('|X(w)|');
```

```
    grid on;
    xlim([-200 200]);
```

时域采样：

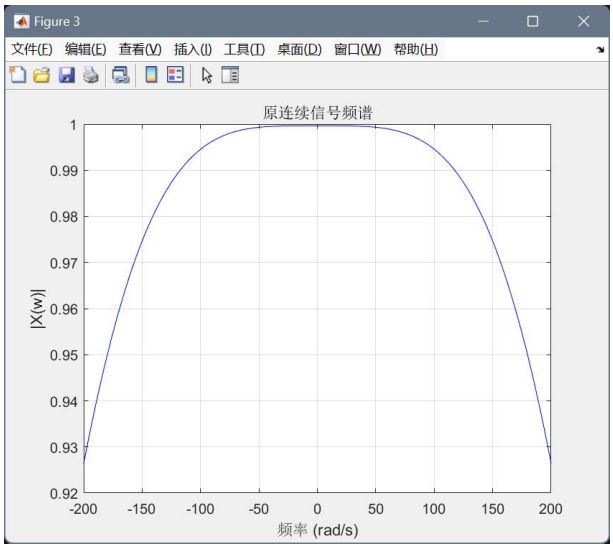
```
    % 时域采样
    fs1 = 1000; fs2 = 300; fs3 = 200; % 采样频率
    Tp = 64e-3; % 观察时间，0.064s
    t1 = 0:(1 / fs1):Tp; t2 = 0:(1 / fs2):Tp; t3 = 0:(1 / fs3):Tp; % 采样点

    xn1 = signal1(t1); xn2 = signal1(t2); xn3 = signal1(t3); % 离散时域信号
    Xk1 = fft(xn1); Xk2 = fft(xn2); Xk3 = fft(xn3); % 离散频域信号

    % 离散时域信号的长度
    w1 = 0:(2 * pi / length(Xk1)):2 * pi;
    w2 = 0:(2 * pi / length(Xk2)):2 * pi;
    w3 = 0:(2 * pi / length(Xk3)):2 * pi;
    % 在使用（0：ts：T）表示离散时间点时，会多出一个点，需要去掉最后一个点
    subplot(3, 2, 1); stem(t1, xn1, "Color", "blue"); title("f_s=1000Hz 采样点图"); xlabel("时间/s");
    subplot(3, 2, 2); plot(w1(1:end - 1), abs(Xk1), "Color", "blue"); title("f_s=1000Hz 幅频特性
"); xlabel("频率/rad");
    subplot(3, 2, 3); stem(t2, xn2, "Color", "blue"); title("f_s=300Hz 采样点图"); xlabel("时间/s");
    subplot(3, 2, 4); plot(w2(1:end - 1), abs(Xk2), "Color", "blue"); title("f_s=300Hz 幅频特性"); xlabel("
频率/rad");
    subplot(3, 2, 5); stem(t3, xn3, "Color", "blue"); title("f_s=200Hz 采样点图"); xlabel("时间/s");
    subplot(3, 2, 6); plot(w3(1:end - 1), abs(Xk3), "Color", "blue"); title("f_s=200Hz 幅频特性"); xlabel("
频率/rad");
```

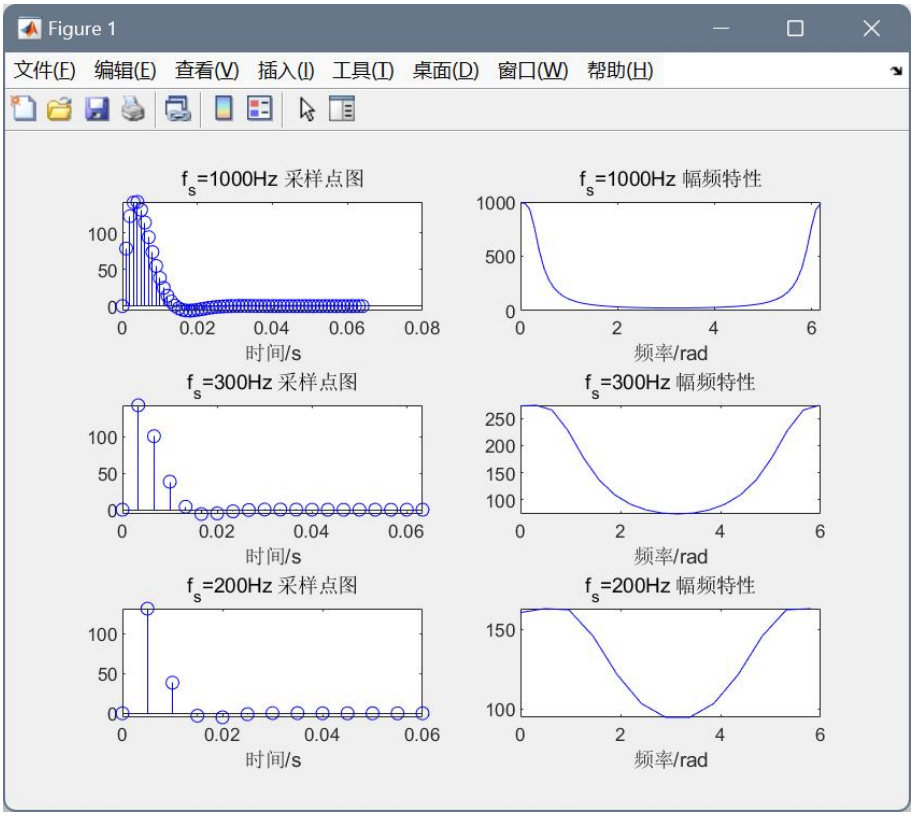
2、频域采样

```
    % 频域采样
    figure;
    x = signal2(0:10000);
    X = fft(x);
    Xk64 = X(round(linspace(1, length(X), 64)));
    Xk32 = Xk64(1:2:end);
    Xk16 = Xk32(1:2:end);
    xn64 = ifft(Xk64); xn32 = ifft(Xk32); xn16 = ifft(Xk16);

    w64 = 0:(2 * pi / length(Xk64)):2 * pi;
    w32 = 0:(2 * pi / length(Xk32)):2 * pi;
    w16 = 0:(2 * pi / length(Xk16)):2 * pi;

    subplot(4, 2, 1); stem(0:32, abs(x(1:33)), "Color", "blue"); title("信号二：三角波序列，长度为 32");
    subplot(4, 2, 2); plot(1:length(X), abs(X), "Color", "blue"); title("三角波序列的频谱");
    subplot(4, 2, 3); stem(w64(1:end - 1), abs(Xk64), "Color", "blue"); title("信号频域 64 点采样");
    subplot(4, 2, 4); stem(1:length(xn64), abs(xn64), "Color", "blue"); title("信号频域 64 点采样 -> 时域");
```

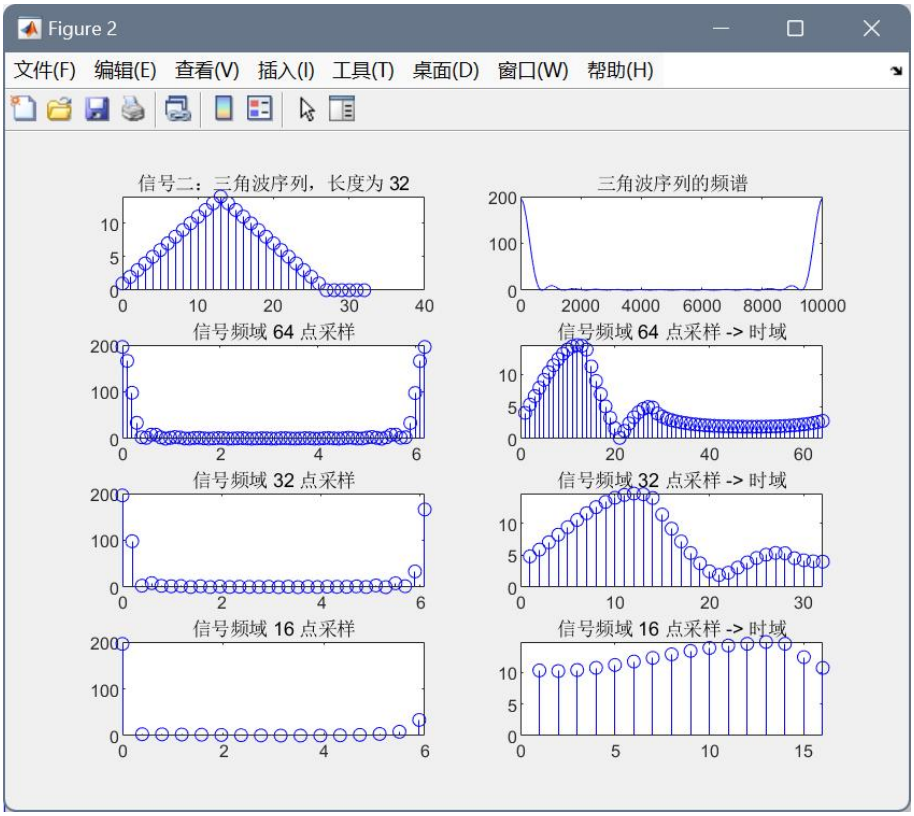
```
18.  subplot(4, 2, 5); stem(w32(1:end - 1), abs(Xk32), "Color", "blue"); title("信号频域 32 点采样");
19.  subplot(4, 2, 6); stem(1:length(xn32), abs(xn32), "Color", "blue"); title("信号频域 32 点采样 -> 时域");
20.  subplot(4, 2, 7); stem(w16(1:end - 1), abs(Xk16), "Color", "blue"); title("信号频域 16 点采样");
21.  subplot(4, 2, 8); stem(1:length(xn16), abs(xn16), "Color", "blue"); title("信号频域 16 点采样 -> 时域");
```

二、实验分析





上图分别是给定的模拟信号在 1000Hz，300Hz，200Hz 三种采样频率下的采样点图和其对应的幅度谱图（使用 FFT 产生），可见当采样频率较低时产生了混叠。



上图是信号二：三角波序列以及利用 FFT 得到的频谱和分别使用 32 点、16 点在频域采样后对应的时域图像。可见当频域的采样间隔较大、采样频率较低时，对应的时域信号发生了混叠。

三、思考题

1. 如果序列 $x(n)$ 的长度为 M ，希望得到其频谱 $X(e^{j\omega})$ 在 $[0,2\pi]$ 上的 N 点等间隔采样，当 $N<M$ 时，如何用一次最少点数的 DFT 得到该频谱采样。

答：应将原来长度为 M 的序列 $x(n)$ 以 N 为周期进行延拓，然后做 DFT。因为若周期序列时域周期为 N 则频域两相邻谱线之间的间隔为 $\Omega=\frac{2\pi}{N}$，所以此时 DFT 的结果在 $[0,2\pi]$ 间有 N 个点，这 N 个点即为在原序列频谱上的采样点。

2. 在时域采样的验证过程中，为什么采用 DFT 或者 FFT 求该模拟信号的幅频特性？

答：DTFT 的计算涉及到无限的求和，通常在实际应用中难以直接实现。相比之下，DFT 和 FFT 是针对有限长信号的离散变换，计算上更加可行和高效。计算机无法处理连续信号，并且 DFT 求出的结果是在连续的频谱上的采样并截取周期，因此可以反映信号的幅频特性，而 FFT 是 DFT 的快速算法，功能与 DFT 相同。

实验三 使用快速傅里叶变换进行频谱分析

一、实验代码及运行结果

1. 通过 FFT 分析以下离散时间序列，并在绘图窗口绘制幅度谱和相位谱。采样频率为 fs=1000Hz,采样点数 N=10000。

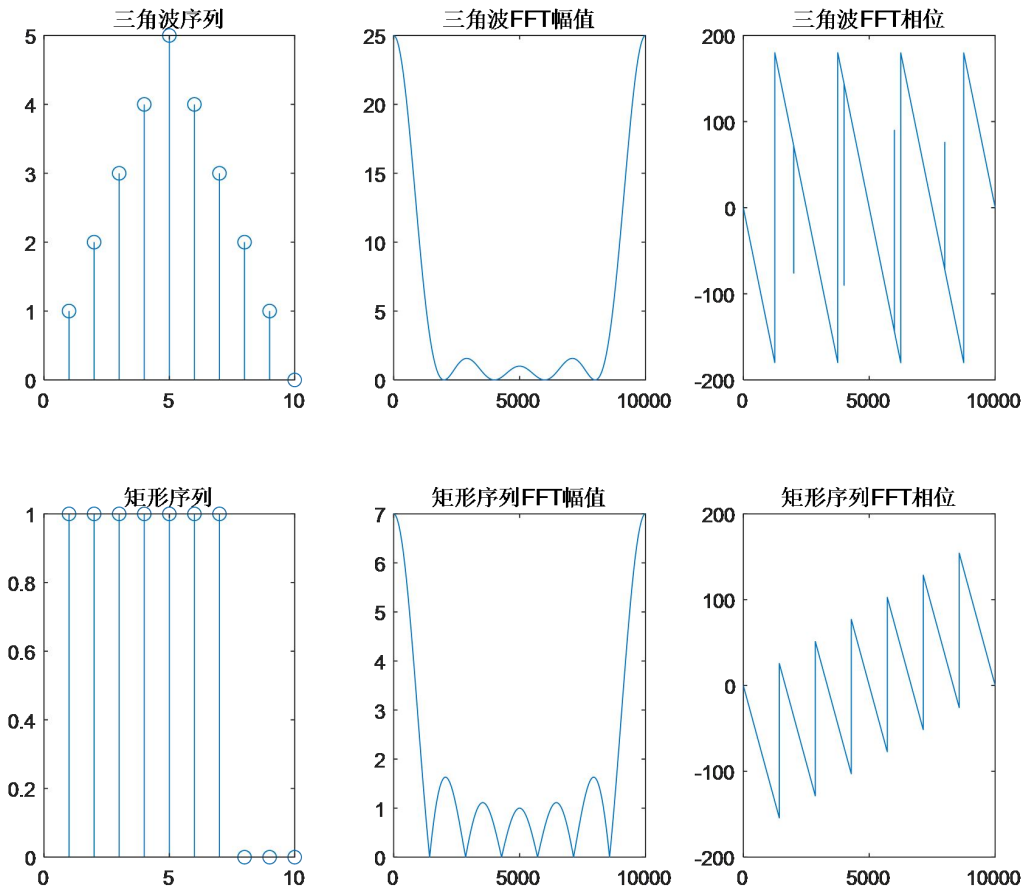
（1）三角波序列：$x_1[n]=\begin{cases} n+1, 0\le n\le 4, \\ 9-n, 5\le n\le 8 \\ 0, 其它 \end{cases}$

（2）矩形序列：$x_2[n]=\begin{cases} 1, 0\le n\le 6, \\ 0, 其它 \end{cases}$

```
%% 通过 FFT 分析以下离散时间序列，并在绘图窗口绘制幅度谱和相位谱。
clc; clear; close all;
x = zeros(1, 10);
y = zeros(1, 10);
n = 1:10000;
% 三角波序列
x(1:5) = 1:5;
x(6:9) = 4:-1:1;

% 绘制三角波序列及其 FFT
subplot(2, 3, 1);
stem(x);
title('三角波序列');

x1 = fft(x, 10000);
subplot(2, 3, 2);
plot(n, abs(x1));
title('三角波 FFT 幅值');
subplot(2, 3, 3);
plot(n, 180 / pi * angle(x1));
title('三角波 FFT 相位');

% 生成矩形序列
y(1:7) = 1;

% 绘制矩形序列及其 FFT
subplot(2, 3, 4);
stem(y);
```

```
29.   title('矩形序列');
30.
31.   y1 = fft(y, 10000);
32.   subplot(2, 3, 5);
33.   plot(n, abs(y1));
34.   title('矩形序列 FFT 幅值');
35.   subplot(2, 3, 6);
36.   plot(n, 180 / pi * angle(y1));
37.   title('矩形序列 FFT 相位');
```

运行结果图：



2. 取采样频率 fs=1000Hz，样点数 N=20000，用 FFT 计算下列两个离散序列的线性卷积，在绘图窗口绘制这两个离散序列及其卷积后序列的幅度谱，分析幅度谱之间的关系。

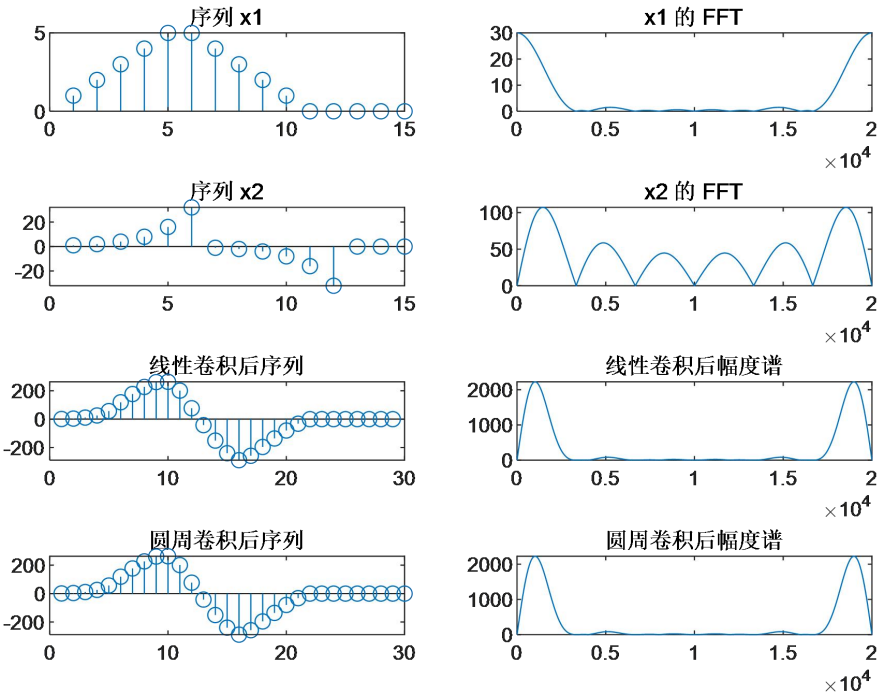
```
1.    clc; clear; close all;
2.    %% 用 FFT 计算下列两个离散序列的线性卷积，在绘图窗口绘制这两个离散序列及其卷积后序列的幅度谱
3.    x1 = zeros(1, 15);
```

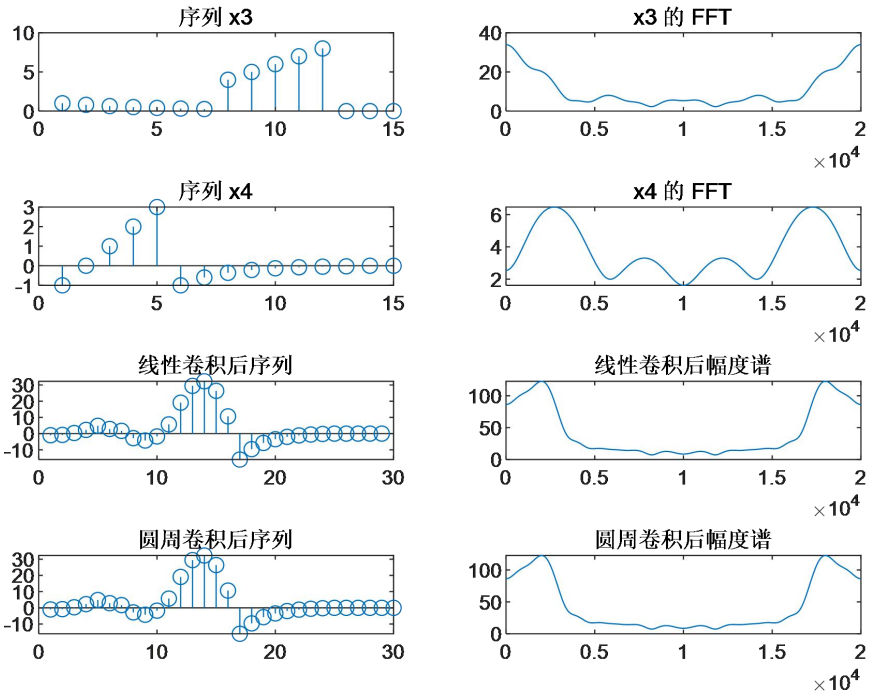
```
x2 = zeros(1, 15);
x3 = zeros(1, 15);
x4 = zeros(1, 15);
n = 1:20000;

% 生成序列 x1
x1(1:5) = 1:5;
x1(6:10) = 5:-1:1;

% 绘制序列 x1 及其 FFT
subplot(4, 2, 1);
stem(x1);
title('序列 x1');

x1_fft = fft(x1, 20000);
subplot(4, 2, 2);
plot(n, abs(x1_fft));
title('x1 的 FFT');

% 生成序列 x2
x2(1:6) = 2 .^ (0:5);
x2(7:12) = -2 .^ (0:5);

% 绘制序列 x2 及其 FFT
subplot(4, 2, 3);
stem(x2);
title('序列 x2');

x2_fft = fft(x2, 20000);
subplot(4, 2, 4);
plot(n, abs(x2_fft));
title('x2 的 FFT');

% 计算卷积后幅度谱及序列

z2 = conv(x1, x2);
z2_fft = fft(z2, 20000);
subplot(4, 2, 5);
stem(1:length(z2), z2);
title('线性卷积后序列');
subplot(4, 2, 6);
plot(n, abs(z2_fft));
title('线性卷积后幅度谱');

```

```
z = x1_fft .* x2_fft;
z1 = ifft(z);
subplot(4, 2, 7);
stem(1:30, z1(1:30));
title('圆周卷积后序列');
subplot(4, 2, 8);
plot(n, abs(z));
title('圆周卷积后幅度谱');

figure;

% 生成序列 x3
x3(1:7) = 0.8 .^ (0:6);
x3(8:12) = 4:8;

% 绘制序列 x3 及其 FFT
subplot(4, 2, 1);
stem(x3);
title('序列 x3');

x3_fft = fft(x3, 20000);
subplot(4, 2, 2);
plot(n, abs(x3_fft));
title('x3 的 FFT');

% 生成序列 x4
x4(1:5) = -1:3;
x4(6:13) = -0.6 .^ (0:7);

% 绘制序列 x4 及其 FFT
subplot(4, 2, 3);
stem(x4);
title('序列 x4');

x4_fft = fft(x4, 20000);
subplot(4, 2, 4);
plot(n, abs(x4_fft));
title('x4 的 FFT');

% 计算卷积后幅度谱及序列
o_2 = conv(x3, x4);
o_2_fft = fft(o_2, 20000);
subplot(4, 2, 5);
stem(1:length(o_2), o_2);
```

```
92.  title('线性卷积后序列');
93.  subplot(4, 2, 6);
94.  plot(n, abs(o_2_fft));
95.  title('线性卷积后幅度谱');
96.
97.  o = x3_fft .* x4_fft;
98.  o1 = ifft(o);
99.  subplot(4, 2, 7);
100. stem(1:30, o1(1:30));
101. title('圆周卷积后序列');
102. subplot(4, 2, 8);
103. plot(n, abs(o));
104. title('圆周卷积后幅度谱');
```

运行结果图：





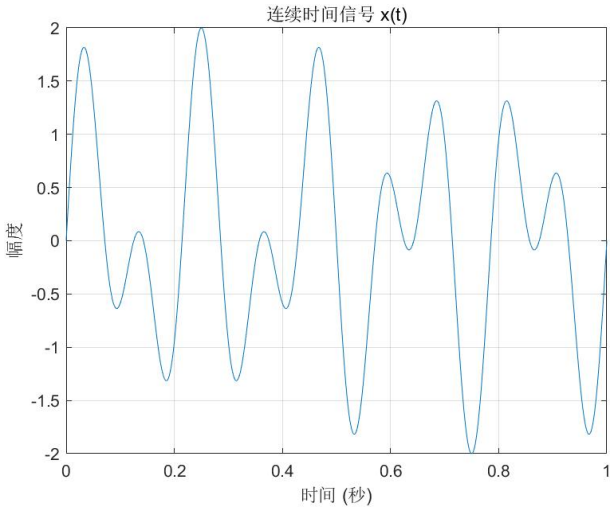
3. 假设一个连续时间信号 $x(t)$ 由两个正弦信号叠加而成，两个正弦信号的初始相位为零，频率分别是 5Hz 和 9Hz

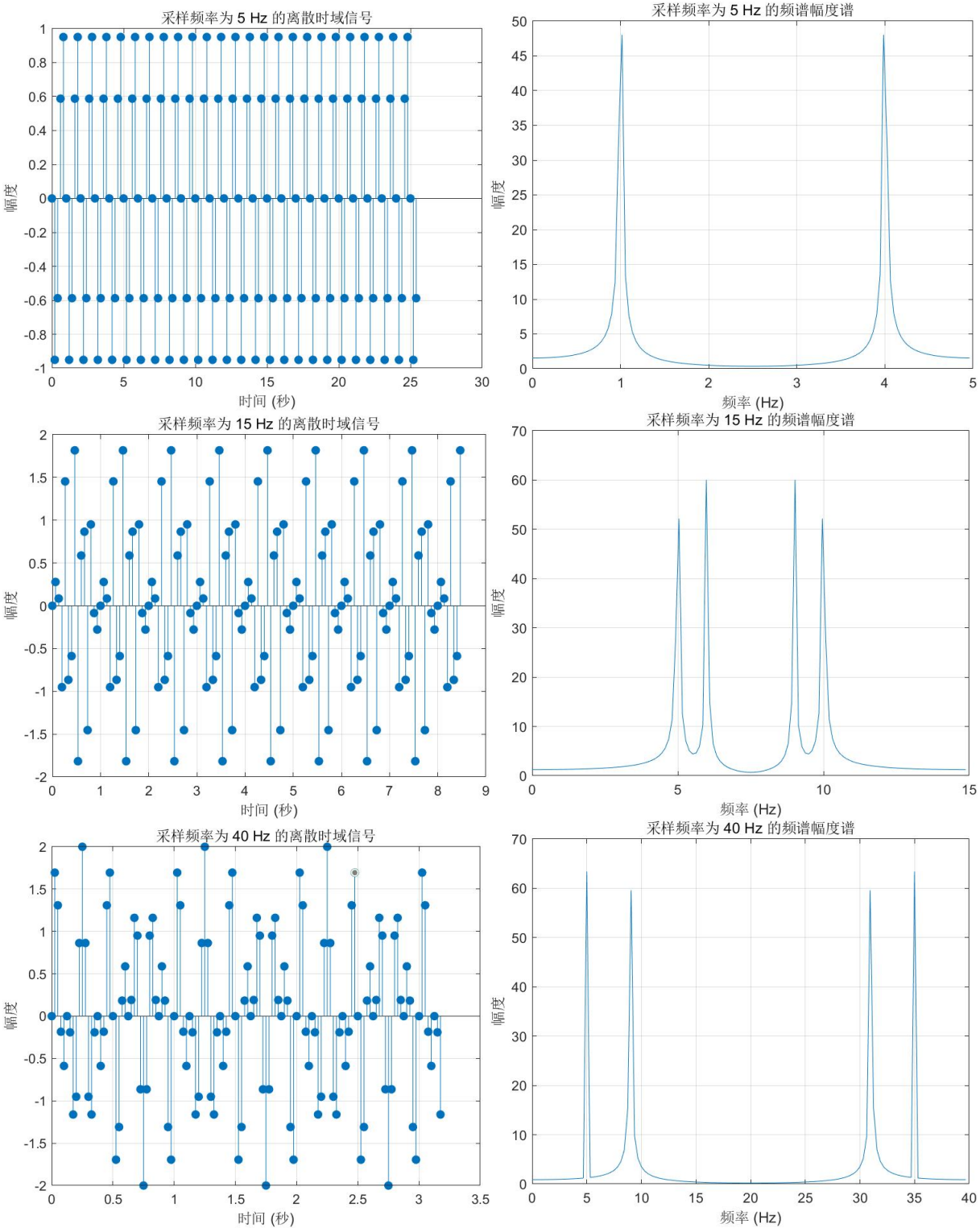
3.1. 当采样频率分别为 $f_1 = 5Hz$， $f_2 = 15Hz$ 和 $f_3 = 40Hz$ 时，信号采样后取 128 点离散序列，画出离散序列时域波形和频谱幅度谱

```
%% 3.1 当采样频率分别为 5、15、40Hz 时，信号采样后取 128 点离散序列，画出离散序列时域波形和频谱幅度谱
% 定义时间变量
fs = 1000; % 采样频率
t = 0:1 / fs:1; % 时间向量，从 0 到 1 秒，步长为 1/fs
f1 = 5; f2 = 9;

% 生成连续时间信号 x(t)
x_t = sin(2 * pi * f1 * t) + sin(2 * pi * f2 * t);

% 绘制信号
figure;
plot(t, x_t);
title('连续时间信号 x(t)');
xlabel('时间 (秒)');
ylabel('幅度');
grid on;

sampling_rates = [5, 15, 40];
```

```
19.
20.   % 进行采样并绘制结果
21.   for i = 1:length(sampling_rates)
22.       fs_sample = sampling_rates(i);
23.       n = 0:127; % 采样点，128 个点
24.       t_sample = n / fs_sample; % 离散时间向量
25.
26.       % 对信号进行采样
27.       x_sampled = sin(2 * pi * f1 * t_sample) + sin(2 * pi * f2 * t_sample);
28.
29.       % 绘制时域图
30.       figure;
31.       stem(t_sample, x_sampled, 'filled');
32.       title(['采样频率为 ', num2str(fs_sample), ' Hz 的离散时域信号']);
33.       xlabel('时间 (秒)');
34.       ylabel('幅度');
35.       grid on;
36.
37.       % 计算并绘制频谱幅度谱
38.       X_f = fft(x_sampled);
39.       X_magnitude = abs(X_f);
40.       f_axis = (0:127) * (fs_sample / 128); % 对应的频率轴，覆盖 0 到 fs_sample
41.
42.       figure;
43.       plot(f_axis, X_magnitude);
44.       title(['采样频率为 ', num2str(fs_sample), ' Hz 的频谱幅度谱']);
45.       xlabel('频率 (Hz)');
46.       ylabel('幅度');
47.       grid on;
48.   end
```

实验结果图：





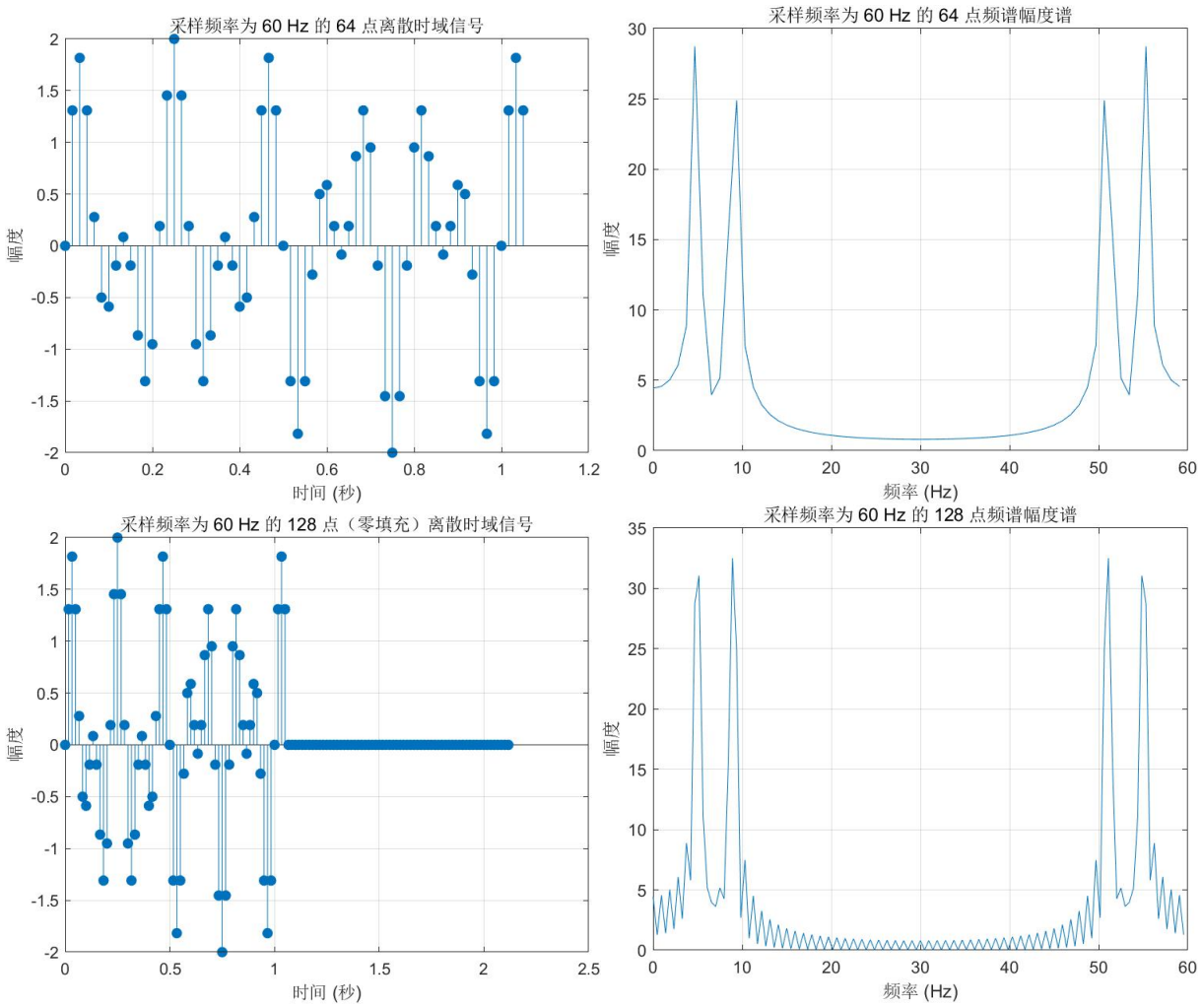
3.2. 取采样频率为 60Hz 时 x(t)的 64 点离散序列，再将其结尾补零加长到 128 点，画出上述两种情况下的离散序列时域波形和频域幅度谱。

```
1.  %% 3.2 取采样频率为 60Hz 时 x(t)的 64 点离散序列，结尾补零到 128 点，画出两种情况下的离散序列时域波形和频域幅度谱。
2.  clc; clear; close all;
3.
4.  f1 = 5; f2 = 9;
```

```
5.
6.  % 采样频率
7.  fs_sample = 60;
8.  n = 0:63; % 64 点采样
9.  t_sample = n / fs_sample; % 离散时间向量
10.
11. % 对信号进行 64 点采样
12. x_sampled = sin(2 * pi * f1 * t_sample) + sin(2 * pi * f2 * t_sample);
13.
14. % 绘制 64 点采样的时域波形
15. figure;
16. stem(t_sample, x_sampled, 'filled');
17. title('采样频率为 60 Hz 的 64 点离散时域信号');
18. xlabel('时间 (秒)');
19. ylabel('幅度');
20. grid on;
21.
22. % 计算并绘制 64 点频谱幅度谱
23. X_f_64 = fft(x_sampled);
24. X_magnitude_64 = abs(X_f_64);
25. f_axis_64 = (0:63) * (fs_sample / 64); % 频率轴
26.
27. figure;
28. plot(f_axis_64, X_magnitude_64);
29. title('采样频率为 60 Hz 的 64 点频谱幅度谱');
30. xlabel('频率 (Hz)');
31. ylabel('幅度');
32. grid on;
33.
34. % 零填充至 128 点
35. x_sampled_padded = [x_sampled, zeros(1, 64)];
36.
37. % 绘制 128 点补零后的时域波形
38. t_padded = (0:127) / fs_sample;
39. figure;
40. stem(t_padded, x_sampled_padded, 'filled');
41. title('采样频率为 60 Hz 的 128 点（零填充）离散时域信号');
42. xlabel('时间 (秒)');
43. ylabel('幅度');
44. grid on;
45.
46. % 计算并绘制 128 点频谱幅度谱
47. X_f_128 = fft(x_sampled_padded);
48. X_magnitude_128 = abs(X_f_128);
```

```
f_axis_128 = (0:127) * (fs_sample / 128); % 频率轴

figure;
plot(f_axis_128, X_magnitude_128);
title('采样频率为 60 Hz 的 128 点频谱幅度谱');
xlabel('频率 (Hz)');
ylabel('幅度');
grid on;
```

实验结果图：

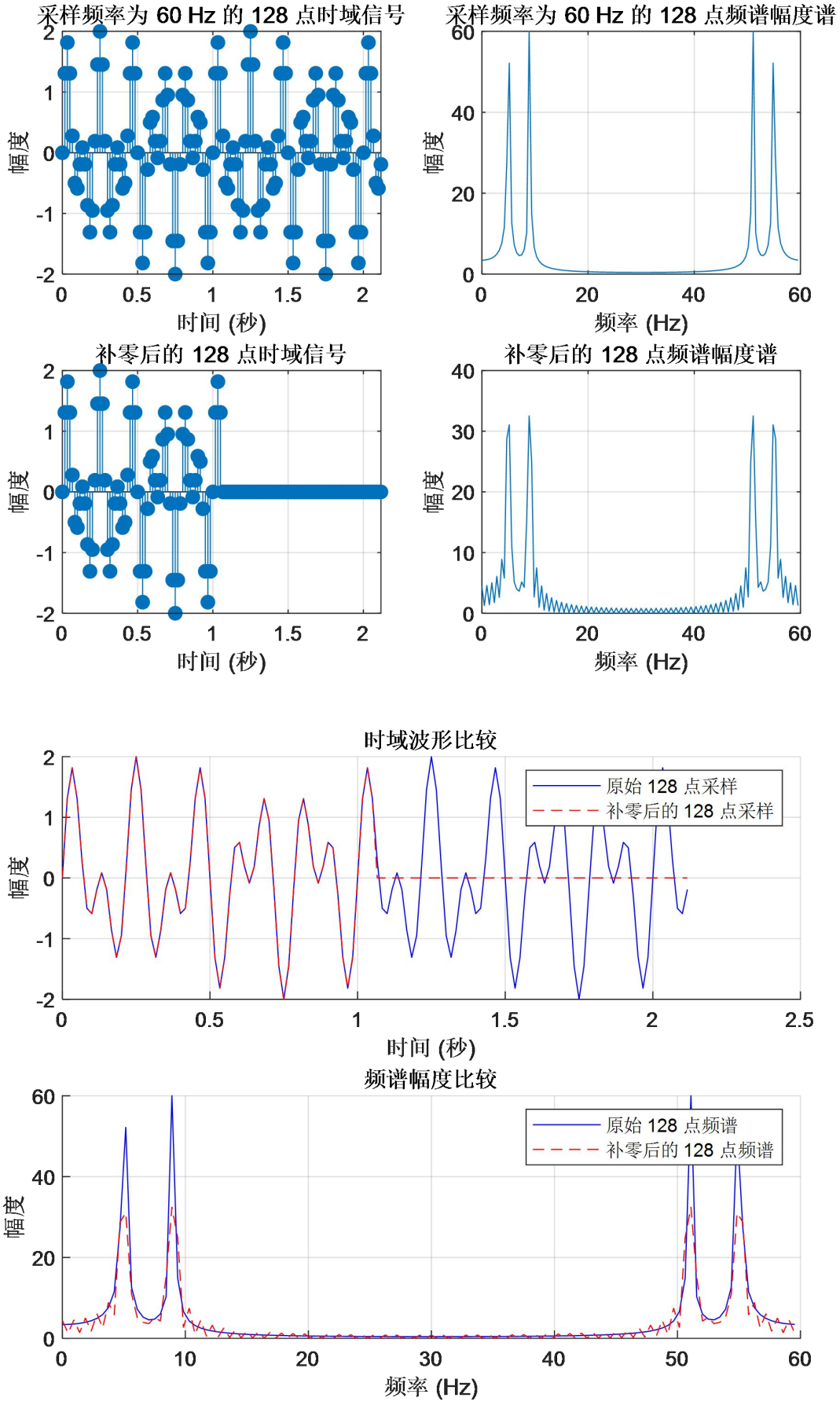


3.3. 取采样频率为 60Hz 时 x(t)的 128 点离散序列，画出离散序列时域波形和频域幅度谱，并与（2）中结尾补零、加长到 128 点的序列波形和频谱相比较

```
%% 3.3 取采样频率为 60Hz 时 x(t)的 128 点离散序列，画出离散序列时域波形和频域幅度谱
clc; clear; close all;

% 定义时间变量
fs = 1000; % 原始采样频率
t = 0:1 / fs:1; % 时间向量，从 0 到 1 秒，步长为 1/fs
```

```
f1 = 5; f2 = 9;

% 生成连续时间信号 x(t)
x_t = sin(2 * pi * f1 * t) + sin(2 * pi * f2 * t);

% 采样频率
fs_sample = 60;
n = 0:127; % 128 点采样
t_sample = n / fs_sample; % 离散时间向量

% 对信号进行 128 点采样
x_sampled_128 = sin(2 * pi * f1 * t_sample) + sin(2 * pi * f2 * t_sample);

% 绘制 128 点采样的时域波形
figure;
subplot(2, 2, 1);
stem(t_sample, x_sampled_128, 'filled');
title('采样频率为 60 Hz 的 128 点时域信号');
xlabel('时间 (秒)');
ylabel('幅度');
grid on;

% 计算并绘制 128 点频谱幅度谱
X_f_128_sampled = fft(x_sampled_128);
X_magnitude_128_sampled = abs(X_f_128_sampled);
f_axis_128 = (0:127) * (fs_sample / 128); % 频率轴

subplot(2, 2, 2);
plot(f_axis_128, X_magnitude_128_sampled);
title('采样频率为 60 Hz 的 128 点频谱幅度谱');
xlabel('频率 (Hz)');
ylabel('幅度');
grid on;

% 补零后的 128 点序列
x_sampled_64 = x_sampled_128(1:64);
x_sampled_padded = [x_sampled_64, zeros(1, 64)];

% 绘制补零后的 128 点时域波形
subplot(2, 2, 3);
t_padded = (0:127) / fs_sample;
stem(t_padded, x_sampled_padded, 'filled');
title('补零后的 128 点时域信号');
xlabel('时间 (秒)');
```

```
ylabel('幅度');
grid on;

% 计算并绘制补零后的 128 点频谱幅度谱
X_f_128_padded = fft(x_sampled_padded);
X_magnitude_128_padded = abs(X_f_128_padded);

subplot(2, 2, 4);
plot(f_axis_128, X_magnitude_128_padded);
title('补零后的 128 点频谱幅度谱');
xlabel('频率 (Hz)');
ylabel('幅度');
grid on;

% 绘制时域波形比较
figure;
subplot(2, 1, 1);
hold on;
plot(t_sample, x_sampled_128, '-b', 'DisplayName', '原始 128 点采样');
plot(t_sample, x_sampled_padded, '--r', 'DisplayName', '补零后的 128 点采样');
title('时域波形比较');
xlabel('时间 (秒)');
ylabel('幅度');
legend;
grid on;
hold off;

% 绘制频谱幅度比较
subplot(2, 1, 2);
hold on;
plot(f_axis_128, X_magnitude_128_sampled, '-b', 'DisplayName', '原始 128 点频谱');
plot(f_axis_128, X_magnitude_128_padded, '--r', 'DisplayName', '补零后的 128 点频谱');
title('频谱幅度比较');
xlabel('频率 (Hz)');
ylabel('幅度');
legend;
grid on;
hold off;
```


采样频率为 60 Hz 的 128 点时域信号
幅度
时间 (秒)
采样频率为 60 Hz 的 128 点频谱幅度谱
幅度
频率 (Hz)
补零后的 128 点时域信号
幅度
时间 (秒)
补零后的 128 点频谱幅度谱
幅度
频率 (Hz)
时域波形比较
原始 128 点采样
补零后的 128 点采样
幅度
时间 (秒)
频谱幅度比较
原始 128 点频谱
补零后的 128 点频谱
幅度
频率 (Hz)

二、实验分析

1. 由实验结果图可知，对于非周期信号进行 FFT，在截断处会造成主要频率的能量泄露。

2. 由实验可知计算的点数足够多的前提下，计算线性卷积和圆周卷积的结果一致。

3. （1）原函数的周期为 1s，为三种采样频率对应周期的整倍数，三种序列均为周期序列。但是由于采样的频率不一样，采样序列在频谱上的频率分布不一样。（2）（3）对比完整采样的 64 和 128 点，可以看到由于 128 点采样后的频谱分布更加细致，更加突出主要频率的成分。而补 0 后的 128 点采样信号由于时域截断造成了频率泄露，在频谱上多了很多小尖峰，且主要频率成分的幅度低于完整 128 点采样后的频谱分布。

三、思考题

阐述线性卷积、圆周卷积和周期卷积的区别和（或）联系。用 FFT 计算线性卷积时，FFT 的长度 N 应满足什么条件？

答：

1、线性卷积

线性卷积是对两个有限长度信号进行卷积的一种方法。假设有两个序列 $x[n]$ 和 $h[n]$，它们的线性卷积 $y[n]$ 定义为：$y[n]=x[n]*h[n]=\sum_{m=-\infty}^{\infty}x[m]\cdot h[n-m]$。线性卷积结果的长度是两个输入信号长度之和减去 1，即 $N_y=\mathrm{N}_x+N_h-1$

2、周期卷积

周期卷积是指对周期性信号进行的卷积运算，假设信号 $x[n]$ 和 $h[n]$ 的周期均为 N，则其周期卷积的结果 $y[n]$ 的周期也为 N，定义为：$y[n]=x[n]\circledast h[n]=\sum_{m=0}^{N-1}x[m]\cdot h[n-m]$

3、圆周卷积

圆周卷积是对两个长度相同的有限长序列进行卷积。假设有长度均为N的两个信号$x[n]$和$h[n]$，它们的圆周卷积$y[n]$定义为：$y[n]=x[n]\circledast h[n]=[\sum_{m=0}^{N-1}x[m]\cdot h((n-m))_N]\cdot R_N(n)$

圆周卷积结果的长度与原信号长度相同。圆周卷积常用于离散傅里叶变换（DFT）的计算，因为DFT中的时域卷积定理表明，在频域中两个信号相乘等价于时域中进行圆周卷积。

4、联系

线性卷积用于非周期信号，计算出的长度一般大于输入信号的长度，而周期卷积用于周期信号，计算结果与输入信号长度相同

圆周卷积是对给定长度的信号进行的操作，强调处理有限长度的循环信号。而周期卷积更强调信号本身具有周期性，即信号被假设在时域中无限重复。

在使用FFT计算线性卷积时，FFT的长度N应满足$N\geq L+M-1$，其中L和M分别为两个序列的长度。

2. 实数序列的频域幅度谱和相位谱有什么规律？虚数序列的频域幅度谱和相位谱有什么规律？

答:幅度谱是各个分量的幅度随信号频率的变化,相位谱是各个分量的相位随信号频率的变化。实数序列的幅度谱是偶对称的，相位谱是奇对称的；而虚数序列则相反。

3. 同一连续信号离散化后有两种情况，第一种是取较长的离散序列求 FFT；第二种是取较短的离散序列，结尾补零扩展成与第一种中的长度相等，再求 FFT。上述两种情况下，信号的频谱有何异同点？

答：第一种情况和第二种情况产生的频谱，峰值的频率相同，但第一种峰值更加明显且曲线更平滑。

实验四 滤波器的设计

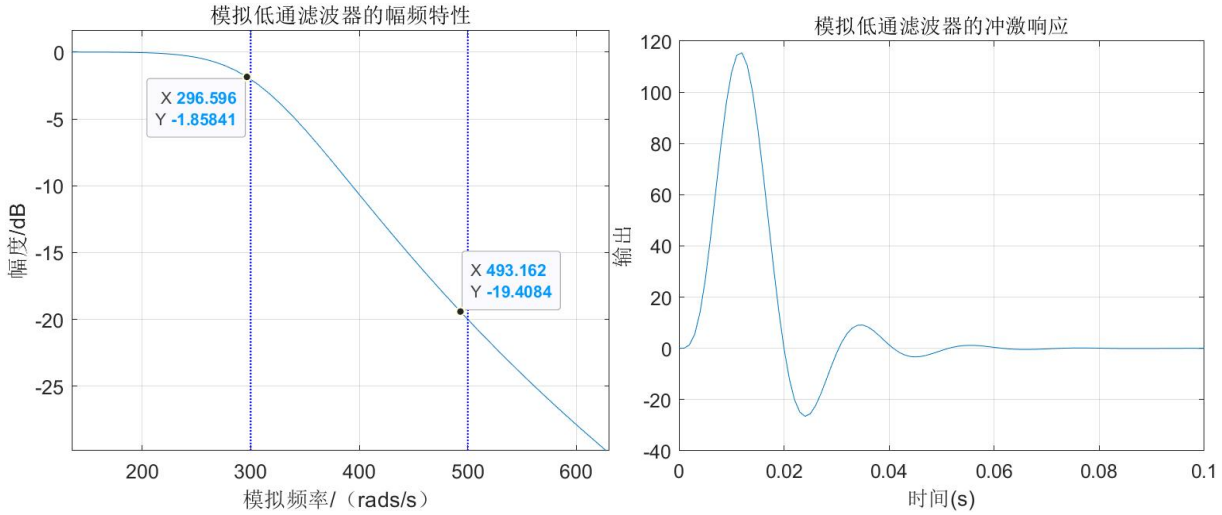
一、实验代码及运行结果

1、模拟低通滤波器设计

实验内容：通带截止频率为 300rad/s，阻带截止频率为 500rad/s，通带内波动为 3dB（即通带最大衰减对应 3dB），阻带内最小衰减为 20dB，仿真计算模拟滤波器的系统函数，使用 freqs 函数求解滤波器的频率特性，在绘图窗口绘制滤波器的冲激响应（横轴时间取 0~0.1s）、幅频特性（坐标轴的显示范围设置为 axis([0 600 -30 8])以便观察），绘图时显示网格（grid on），显示 x 轴、y 轴坐标含义文本。（依次用到函数：buttord, butter, tf, impulse, freqs）

```
close all; clear; clc;
%% 1. 模拟低通滤波器设计
set(0, 'DefaultAxesFontSize', 12);
set(0, 'DefaultTextFontSize', 12);
Wp = 300; % 通带截止频率
Ws = 500; % 阻带截止频率
Rp = 3; % 通带最大衰减
Rs = 20; % 阻带最小衰减
[n, Wn] = buttord(Wp, Ws, Rp, Rs, 's'); % 计算巴特沃斯滤波器的阶数和截止频率
[b, a] = butter(n, Wn, 'low', 's'); % 低通滤波器设计
[H, W] = freqs(b, a);
plot(W, 20 * log10(abs(H)));
xlabel('模拟频率/（rads/s）');
ylabel('幅度/dB');
title('模拟低通滤波器的幅频特性');
axis([0 600 -30 8]);
line([300 300], ylim, 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
line([500 500], ylim, 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
grid on;

[y, t] = impulse(tf(b, a), 0:0.001:0.1); % 低通滤波器的冲激响应
figure;
plot(t, y);
xlabel('时间(s)');
ylabel('输出');
title('模拟低通滤波器的冲激响应');
grid on;
```

实验结果图：



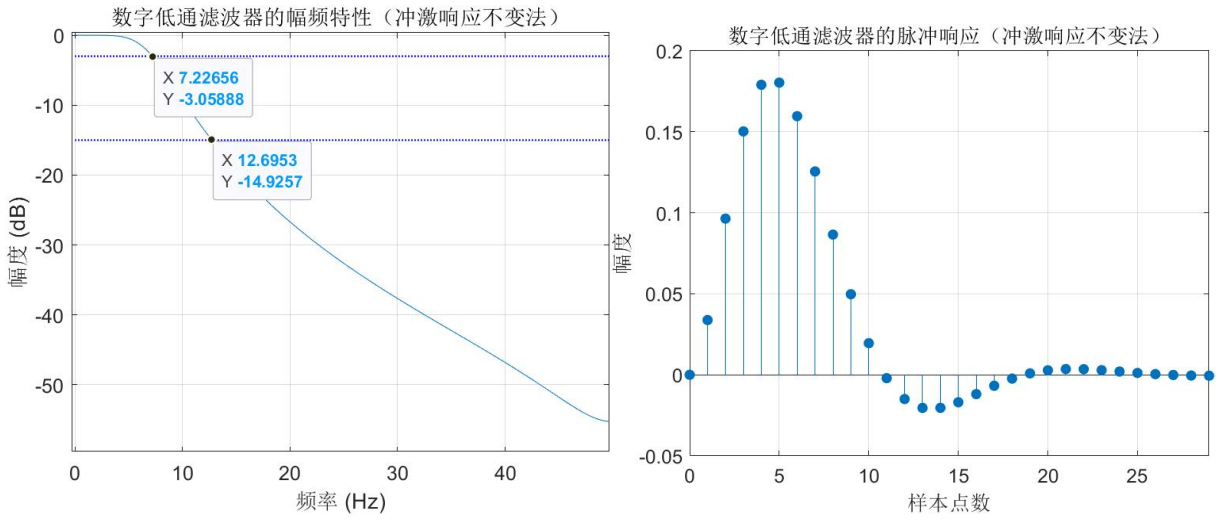
2、数字低通滤波器设计：冲激响应不变法

实验内容：采用冲激响应不变法设计一个巴特沃思低通数字滤波器，其通带截止频率为 0.4rad，阻带截止频率为 0.8rad，采样频率为 Fs=100Hz（根据采样定理，最高截止频率为 50Hz），通带内的最大衰减为 3dB，阻带内的最小衰减为 15dB，仿真计算数字滤波器系统函数，在绘图窗口绘制出滤波器的脉冲响应、幅频特性。

```
close all; clear; clc;
%% 2. 数字低通滤波器设计
set(0, 'DefaultAxesFontSize', 12);
set(0, 'DefaultTextFontSize', 12);
Wp = 0.4; % 数字通带截止频率（rad）
Ws = 0.8; % 数字阻带截止频率（rad）
Rp = 3; % 通带最大衰减（dB）
Rs = 15; % 阻带最小衰减（dB）
fs = 100; % 采样频率（Hz）

% 使用 buttord 设计模拟低通滤波器的阶数和截止频率
% 将数字频率转换为模拟频率（rad/s）
Wp_analog = Wp *  fs;
Ws_analog = Ws * fs;
[N, Wn] = buttord(Wp_analog, Ws_analog, Rp, Rs, 's');
[b_analog, a_analog] = butter(N, Wn, 's'); % 设计模拟 Butterworth 滤波器

% 将模拟滤波器转换为数字滤波器，采用冲激响应不变法
[b_digital, a_digital] = impinvar(b_analog, a_analog, fs);

% 计算幅频特性
figure;
[H, f] = freqz(b_digital, a_digital); % 计算频率响应
f_Hz = f * fs / (2 * pi); % 将频率转换为 Hz
plot(f_Hz, 20 * log10(abs(H))); % 绘制幅频特性，将幅度转换为分贝
```

```
line(xlim, [-3 -3], 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
line(xlim, [-15 -15], 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
xlabel('频率 (Hz)');
ylabel('幅度 (dB)');
title('数字低通滤波器的幅频特性（冲激响应不变法）');
grid on;
% 计算脉冲响应
figure;
impz(b_digital, a_digital, 30); % 绘制脉冲响应
xlabel('样本点数');
ylabel('幅度');
title('数字低通滤波器的脉冲响应（冲激响应不变法）');
grid on;
```

实验结果图：

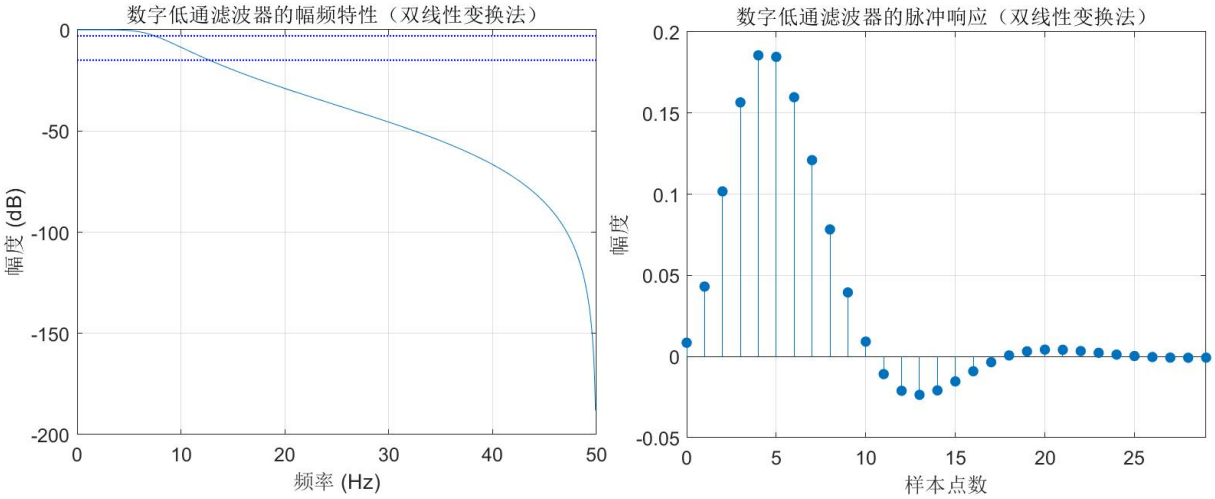


3、数字低通滤波器设计：双线性变换法

```
set(0, 'DefaultAxesFontSize', 12);
set(0, 'DefaultTextFontSize', 12);
Wp = 0.4; % 数字通带截止频率（rad）
Ws = 0.8; % 数字阻带截止频率（rad）
Rp = 3; % 通带最大衰减（dB）
Rs = 15; % 阻带最小衰减（dB）
fs = 100; % 采样频率（Hz）
% 使用 buttord 设计模拟低通滤波器的阶数和截止频率
% 将数字频率转换为模拟频率（rad/s）
Wp_analog = 2 * fs * tan(Wp / 2);
Ws_analog = 2 * fs * tan(Ws / 2);
[N, Wn] = buttord(Wp_analog, Ws_analog, Rp, Rs, 's');
[b_analog, a_analog] = butter(N, Wn, 's'); % 设计模拟 Butterworth 滤波器

% 将模拟滤波器转换为数字滤波器，采用双线性变换法
[b_digital, a_digital] = bilinear(b_analog, a_analog, fs);
% 计算幅频特性
```

```
figure;
[H, f] = freqz(b_digital, a_digital); % 计算频率响应
f_Hz = f * fs / (2 * pi); % 将频率转换为 Hz
plot(f_Hz, 20 * log10(abs(H))); % 绘制幅频特性，将幅度转换为分贝
line(xlim, [-3 -3], 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
line(xlim, [-15 -15], 'Color', 'b', 'LineStyle', ':', 'LineWidth', 1);
xlabel('频率 (Hz)');
ylabel('幅度 (dB)');
title('数字低通滤波器的幅频特性（双线性变换法）');
grid on;
% 计算脉冲响应
figure;
impz(b_digital, a_digital, 30); % 绘制脉冲响应
xlabel('样本点数');
ylabel('幅度');
title('数字低通滤波器的脉冲响应（双线性变换法）');
grid on;
```

实验结果图：



二、实验分析

1、由图可见设计的模拟滤波器的幅频特性满足要求

2、3、由图可见冲激响应不变法得到的频谱在最后变化趋势趋于平稳，而双线性变换法在频率增大时频谱变化更加剧烈。

模拟滤波器，处理连续信号，无频谱混叠，处理范围有限，频谱带内波动小，频谱更加平滑。数字滤波器处理离散信号，遵循奈奎斯特原则，可能出现频谱混叠，处理信号可能有时延，处理范围广泛，频谱波动较大，设计灵活但可能引入畸变。

三、思考题

1. 总结巴特沃思低通滤波器幅频特性的特点。

答：巴特沃斯低通滤波器在通带内的幅频特性较为平坦且没有起伏，尤其是在频率为 0 的点处非常平直；其幅频曲线单调下降，阶数越高通带内越平坦，且过渡带越窄。

2. 双线性变换法中模拟频率与数字频率之间的关系是非线性的，在设计数字滤波器时，应如何处理这种非线性关系？

答：需要别注意频率预畸变问题。先对模拟滤波器的临界频率加以畸变，使其通过双线性变换后正好映射为需要的频率：

$$\omega = \frac{2}{T}\tan\frac{\Omega}{2}$$