# 西北工业大学

# 《信号与系统》实验报告

学　　院：软件学院

学　　号：2021302853

姓　　名：张苏宇

专　　业：软件工程

实验时间：2023 年 12 月 14 日

实验地点：启翔楼 264

指导教师：柳艾飞、汪彦婷

西北工业大学

**2023　年 12　月**

# 一、实验目的

Ø 掌握抽样定理，验证抽样定理；

Ø 掌握利用Matlab完成信号抽样的方法，并对抽样信号的频谱进行分析；

Ø 了解运用Matlab对抽样信信号进行恢复的方法。

# 二、实验报告要求

1. 提交：实验报告一份，PDF 格式，其他格式拒收。

实验报告中需要包括：

a) 若题目要求理论结果，报告中需要给出理论结果。

b) 结果图；图中需要有适当的标识。

c) 源代码。源代码中要有合适的注释。

d) 实验体会和感悟。

2. 提交实验报告规则：

2023年12月17日21点之前将实验报告提交到坚果云

一班实验三提交链接：https://send2me.cn/fMgZAygK/TpSFwww7gnHHeA

二班实验三提交链接：https://send2me.cn/9btDbArW/QJC_S42AQ_JXng

文件名命名规则：课堂号-学号-姓名-第几次实验。

# 三、实验设备（环境）

操作系统： Windows 10

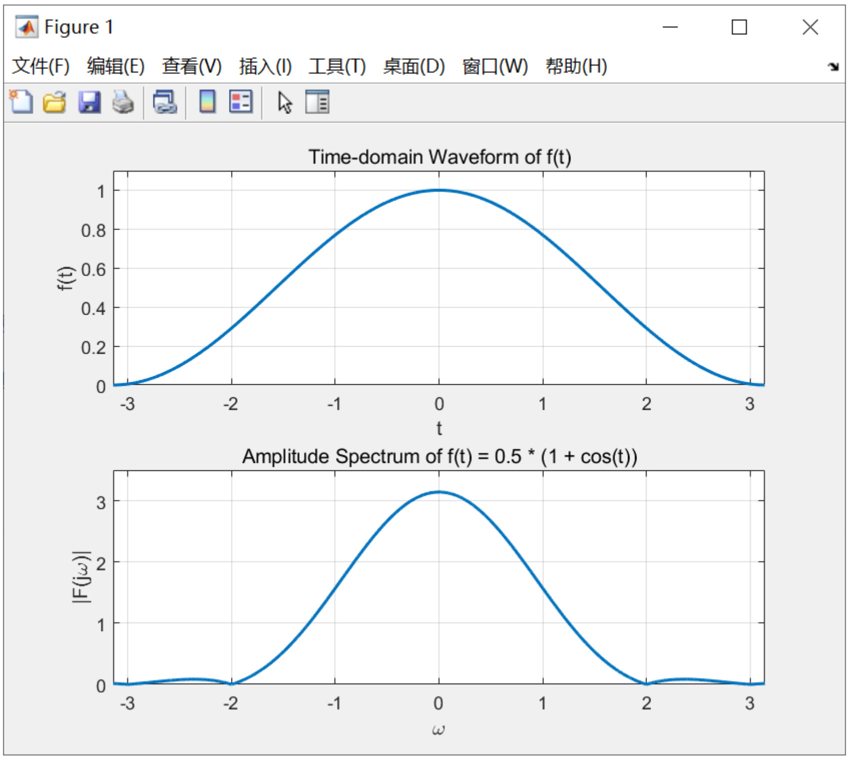
编程软件： Matlab R2023b

# 四、实验内容

## Ø 实验1：抽样定理验证实验

已知连续信号为 $f(t) = 0.5(1 + cost), -\pi \le t \le \pi$

（1）绘制$f(t)$时域波形和频谱；

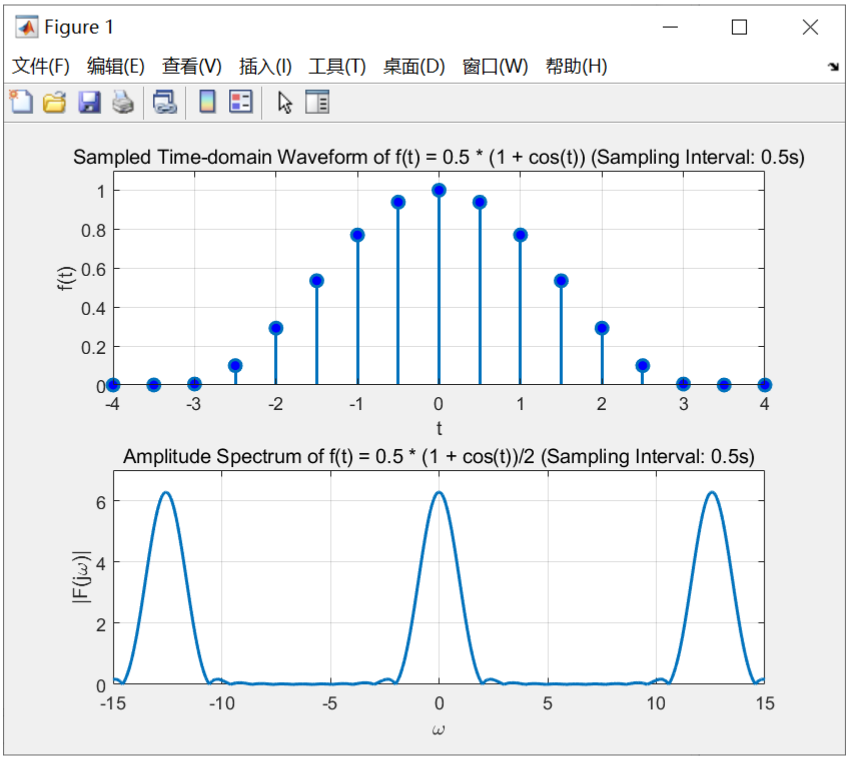



```
function plotSignalAndSpectrum()
    % Define time vector
    t = -pi:0.01:pi;

    % Define the signal function
    f = ((1 + cos(t)) / 2) .* (stepfun(t, -pi) - stepfun(t, pi));

    % Plot time-domain waveform
    figure;
    subplot(2, 1, 1);
    plot(t, f, 'LineWidth', 1.5);
    title('Time-domain Waveform of f(t)');
    grid on;
    xlabel('t');
    xlim([-pi, pi]);
    ylabel('f(t)');
    ylim([0, 1.1]);

    % Compute and plot the amplitude spectrum
    omega1 = -pi;
    omega2 = pi;
    K = 4000;
    OMEGA = omega2 - omega1;
    delta_omega = OMEGA / K;
    omega = omega1:delta_omega:omega2;

    F = 0.01 * (f * exp(-1i * t' * omega));

    subplot(2, 1, 2);
    plot(omega, abs(F), 'LineWidth', 1.5);
    title('Amplitude Spectrum of f(t) = 0.5 * (1 + cos(t))');
    grid on;
    xlabel('\omega');
    xlim([-pi, pi]);
```

```
    ylabel('|F(j\omega)|');
    ylim([0, 3.5]);
end

```

（2）分别绘制抽样间隔为0.5s、1s、2s时的抽样信号的时域波形和频谱；

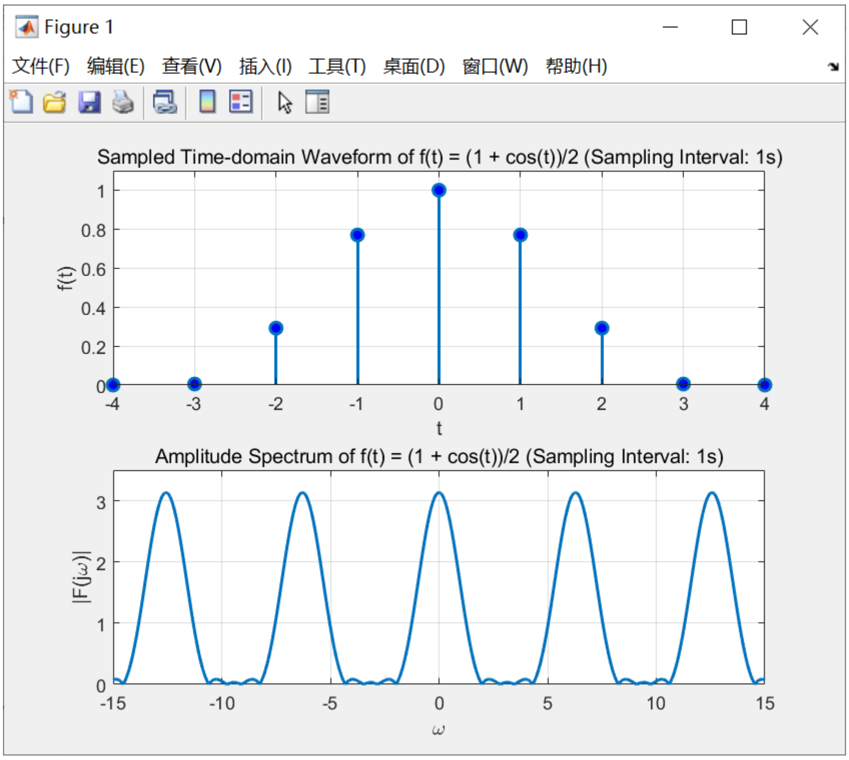
①抽样间隔为 0.5s：




```
function plotSampledSignalAndSpectrum()
    % Define sampling interval
    Ts = 0.5;

    % Define time vector
    ts = -20:Ts:20;

    % Define the sampled signal
    fs = ((1 + cos(ts)) / 2) .* (heaviside(ts + pi) - heaviside(ts - pi));

    % Plot time-domain waveform
    figure;
    subplot(2, 1, 1);
    stem(ts, fs, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    title('Sampled Time-domain Waveform of f(t) = 0.5 * (1 + cos(t)) (Sampling Interval: 0.5s)');
    grid on;
    xlabel('t');
    xlim([-4, 4]);
    ylabel('f(t)');
    ylim([0, 1.1]);

    % Compute and plot the amplitude spectrum
    omega1 = -8 * pi;
    omega2 = 8 * pi;
    K = 4000;
```

```
    OMEGA = omega2 - omega1;
    delta_omega = OMEGA / K;
    omega = omega1:delta_omega:omega2;

    % Calculate Fourier series coefficients
    Fs = Ts * (fs * exp(-1i * ts' * omega));

    subplot(2, 1, 2);
    plot(omega, abs(2 * Fs), 'LineWidth', 1.5);
    title('Amplitude Spectrum of f(t) = 0.5 * (1 + cos(t))/2 (Sampling Interval: 0.5s)');
    grid on;
    xlabel('\omega');
    xlim([-15, 15]);
    ylabel('|F(j\omega)|');
    ylim([0, 7]);
end

```

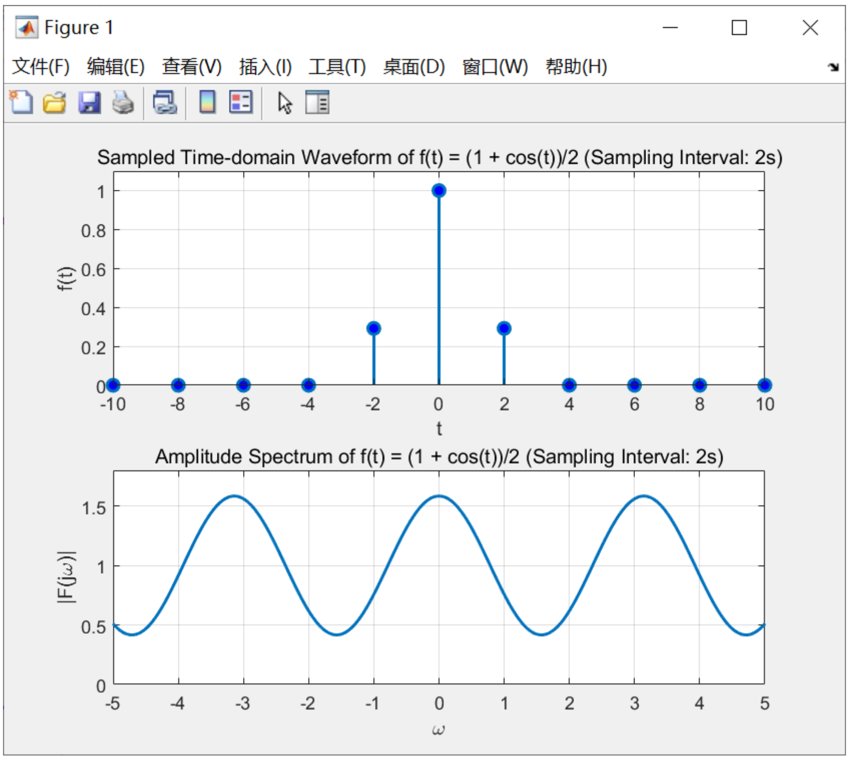
②抽样间隔为 1s：




```
function plotSampledSignalAndSpectrum()
    % Parameters
    Ts = 1;
    tRange = -20:Ts:20;
    omegaRange = -8*pi:0.001:8*pi;

    % Define the sampled signal
    fs = ((1 + cos(tRange)) / 2) .* (heaviside(tRange + pi) - heaviside(tRange - pi));

    % Plot time-domain waveform
    figure;
    subplot(2, 1, 1);
```

```
    stem(tRange, fs, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    title('Sampled Time-domain Waveform of f(t) = (1 + cos(t))/2 (Sampling Interval: 1s)');
    grid on;
    xlabel('t');
    xlim([-4, 4]);
    ylabel('f(t)');
    ylim([0, 1.1]);

    % Compute and plot the amplitude spectrum
    omega1 = -8 * pi;
    omega2 = 8 * pi;
    K = 4000;
    delta_omega = (omega2 - omega1) / K;
    omega = omega1:delta_omega:omega2;

    % Calculate Fourier series coefficients
    Fs = Ts * (fs * exp(-1i * tRange' * omega));

    subplot(2, 1, 2);
    plot(omega, abs(Fs), 'LineWidth', 1.5);
    title('Amplitude Spectrum of f(t) = (1 + cos(t))/2 (Sampling Interval: 1s)');
    grid on;
    xlabel('\omega');
    xlim([-15, 15]);
    ylabel('|F(j\omega)|');
    ylim([0, 3.5]);
end

```

③抽样间隔为 2s：




```
function plotSampledSignalAndSpectrum()
```

```matlab
    % Parameters
    Ts = 2;
    tRange = -20:Ts:20;
    omegaRange = -10*pi:0.001:10*pi;

    % Define the sampled signal
    fs = ((1 + cos(tRange)) / 2) .* (heaviside(tRange + pi) - heaviside(tRange - pi));

    % Plot time-domain waveform
    figure;
    subplot(2, 1, 1);
    stem(tRange, fs, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    title('Sampled Time-domain Waveform of f(t) = (1 + cos(t))/2 (Sampling Interval: 2s)');
    grid on;
    xlabel('t');
    xlim([-10, 10]);
    ylabel('f(t)');
    ylim([0, 1.1]);

    % Compute and plot the amplitude spectrum
    omega1 = -10 * pi;
    omega2 = 10 * pi;
    K = 4000;
    delta_omega = (omega2 - omega1) / K;
    omega = omega1:delta_omega:omega2;

    % Calculate Fourier series coefficients
    Fs = Ts * (fs * exp(-1i * tRange' * omega));

    subplot(2, 1, 2);
    plot(omega, abs(Fs / 2), 'LineWidth', 1.5);
    title('Amplitude Spectrum of f(t) = (1 + cos(t))/2 (Sampling Interval: 2s)');
    grid on;
    xlabel('\omega');
    xlim([-5, 5]);
    ylabel('|F(j\omega)|');
    ylim([0, 1.8]);
end

```

(3)观察抽样信号的频谱混叠程度，验证抽样定理。**注：抽样信号的幅度谱绘制三个周期即可。**

抽样定理（Nyquist定理）是一个基本的信号处理原理，它指导着在进行信号抽样时应该选择多大的抽样率。抽样定理的核心观点是，为了避免频谱混叠（即采样导致的信号信息丢失），信号的抽样率必须至少是信号中最高频率的两倍。

原信号的 $\omega_m = 2, \omega_s = \frac{2\pi}{T}$

当抽样间隔抽样间隔为 0.5s 时，$\omega_s = 4\pi$，抽样间隔为 1s 时，$\omega_s = 2\pi$，这两种情况时$\omega_s \geq 2\omega_m$，大于奈奎斯特抽样频率，相邻信号之间没发生混叠。

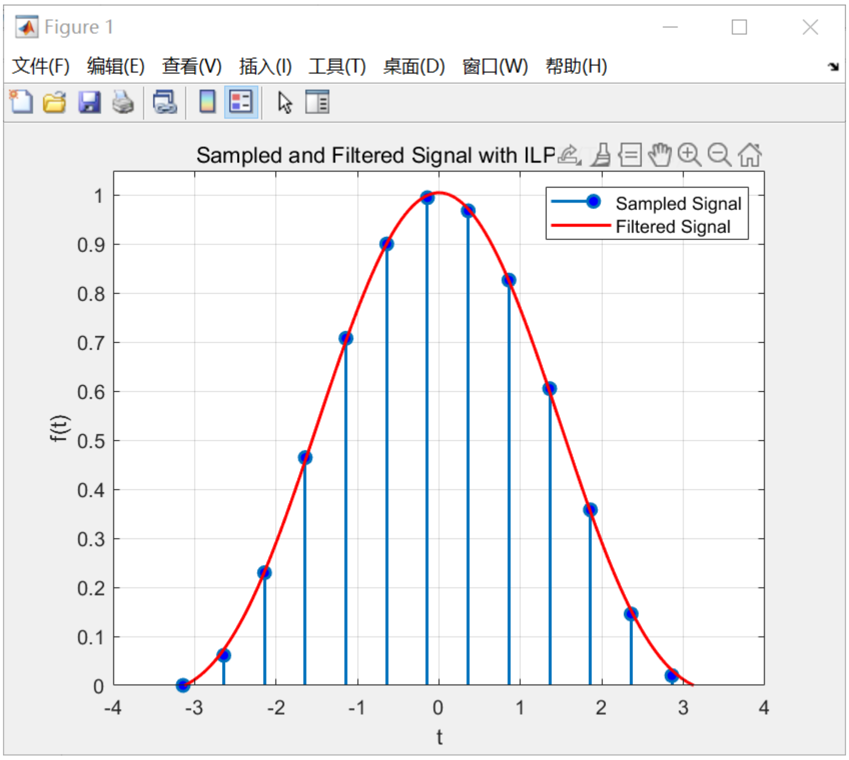
而当抽样间隔为 2s 时，$\omega_s \leq 2\omega_m$, 小于奈奎斯特抽样频率，相邻信号之间发生混叠，间隔信号失真，满足抽样定理。

## Ø 实验2：信号恢复实验

2.1 对实验1中的信号，观察到 $\omega_m = 2$。对于抽样之后的信号，采用截止频率为 $\omega_c = 1.2\omega_m$的ILPF进行信号恢复。

（1）画出三种抽样间隔下抽样信号通过ILPF后的信号时域波形图；

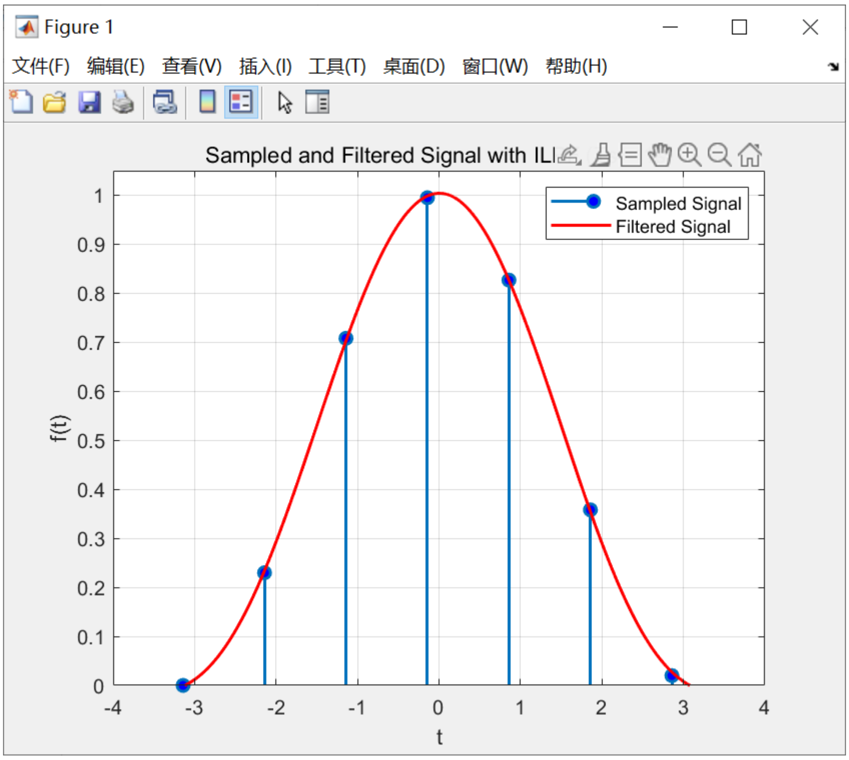
①抽样间隔为 0.5s：




```
function plotSampledAndFilteredSignal()
    % Sampling parameters
    Ts = 0.5;
    t_sampling = -pi:Ts:pi;

    % Continuous time vector
    t_continuous = -pi:0.01:pi;

    % Signal definition
    f_continuous = 0.5 * (1 + cos(t_sampling));

    % Ideal Low Pass Filter
    Wc = 2.4;
    ILPF = sinc(Wc/pi * (t_continuous - t_sampling'));

    % Sampled and filtered signal
    F_filtered = Ts * (Wc/pi) * f_continuous * ILPF;

    % Plotting
    figure;
    stem(t_sampling, f_continuous, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    hold on;
    plot(t_continuous, F_filtered, 'r', 'LineWidth', 1.5);
```

```
    hold off;

    xlabel('t');
    ylabel('f(t)');
    ylim([0, 1.05]);
    title('Sampled and Filtered Signal with ILPF (Ts = 0.5s)');
    legend('Sampled Signal', 'Filtered Signal');
    grid on;
end

```

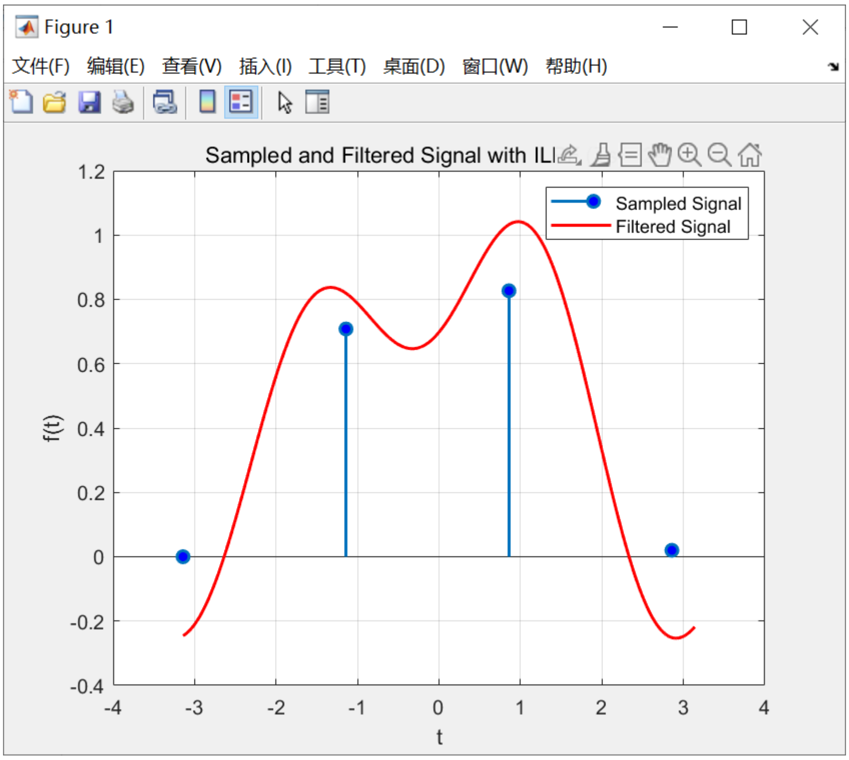
②抽样间隔为 1s:




```
function plotSampledAndFilteredSignal()
    % Sampling parameters
    Ts = 1.0;
    t_sampling = -pi:Ts:pi;

    % Continuous time vector
    t_continuous = -pi:0.01:pi;

    % Signal definition
    f_continuous = 0.5 * (1 + cos(t_sampling));

    % Ideal Low Pass Filter
    Wc = 2.4;
    ILPF = sinc(Wc/pi * (t_continuous - t_sampling'));

    % Sampled and filtered signal
    F_filtered = Ts * (Wc/pi) * f_continuous * ILPF;

    % Plotting
    figure;
```

```
    stem(t_sampling, f_continuous, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    hold on;
    plot(t_continuous, F_filtered, 'r', 'LineWidth', 1.5);
    hold off;

    xlabel('t');
    ylabel('f(t)');
    ylim([0, 1.05]);
    title('Sampled and Filtered Signal with ILPF (Ts = 1s)');
    legend('Sampled Signal', 'Filtered Signal');
    grid on;
end

```

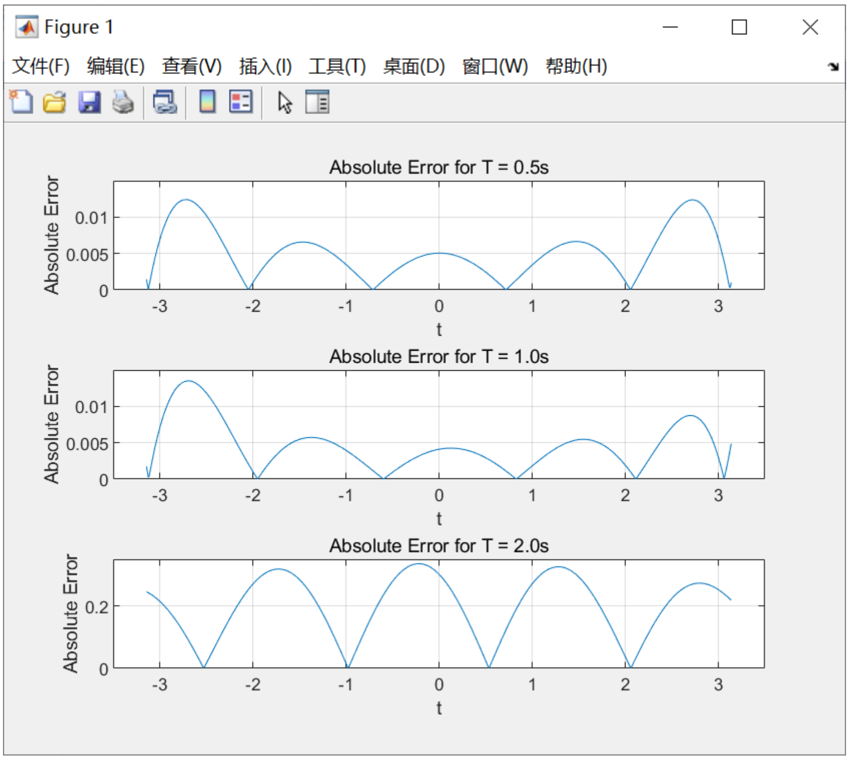
③抽样间隔为 2s:




```
function plotSampledAndFilteredSignal()
    % Sampling parameters
    Ts = 2.0;
    t_sampling = -pi:Ts:pi;

    % Continuous time vector
    t_continuous = -pi:0.01:pi;

    % Signal definition
    f_continuous = 0.5 * (1 + cos(t_sampling));

    % Ideal Low Pass Filter
    Wc = 2.4;
    ILPF = sinc(Wc/pi * (t_continuous - t_sampling'));

    % Sampled and filtered signal
    F_filtered = Ts * (Wc/pi) * f_continuous * ILPF;
```

```
    % Plotting
    figure;
    stem(t_sampling, f_continuous, 'MarkerFaceColor', 'b', 'LineWidth', 1.5);
    hold on;
    plot(t_continuous, F_filtered, 'r', 'LineWidth', 1.5);
    hold off;

    xlabel('t');
    ylabel('f(t)');
    title('Sampled and Filtered Signal with ILPF (Ts = 2s)');
    legend('Sampled Signal', 'Filtered Signal');
    grid on;
end

```

(2)绘制三种抽样间隔下的恢复信号与原信号的绝对误差图，观察并总结抽样间隔对于信号恢复过程的影响。




由以上三张绝对误差图可知：

①当抽样间隔为 0.5s 或为 1s 时，$\omega_s \geq 2\omega_m$，还原出来的在理想情况下不失真，恢复信号与原信号相比，绝对误差较小；

②当抽样间隔为 2s 时， $\omega_s \leq 2\omega_m$，还原出来的信号明显发生失真，采样以后的恢复信号与原信号相比，绝对误差较大。

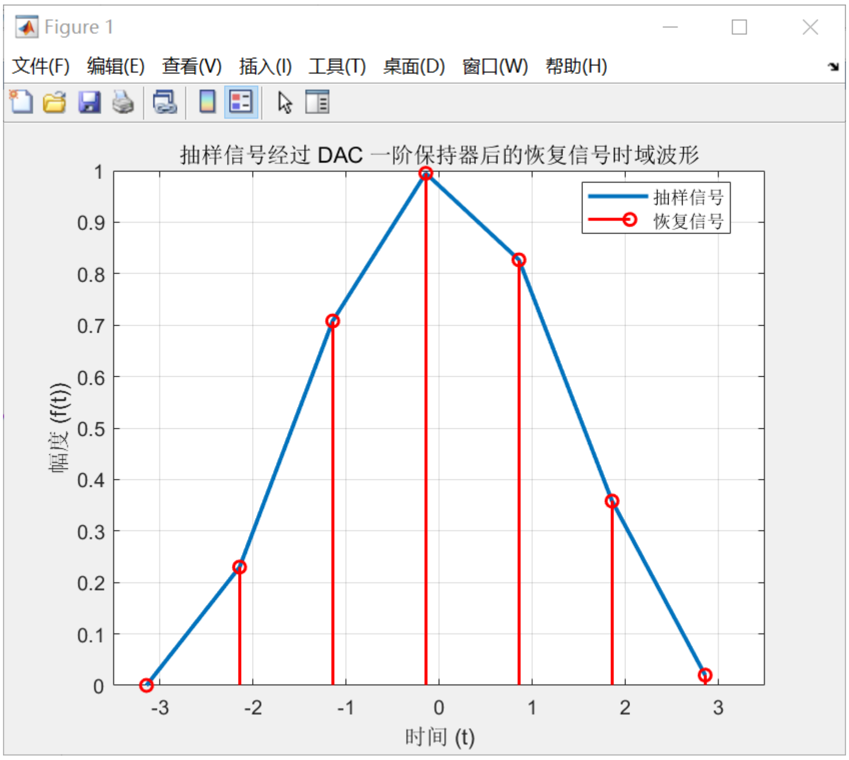
在信号恢复过程中选择的抽样间隔越小，$\omega_s$越大，信号恢复的效果越好，反之效果越差。选择恰当的抽样间隔才能既不必过于频繁的采样，又保证信号的失真在可容忍的范围内。

```
function plotAbsoluteErrorComparison()
    % Sampling intervals
    T1 = 0.5;
    T2 = 1.0;
```

```
    T3 = 2.0;

    % Sample ranges
    t1 = -pi:T1:pi;
    t2 = -pi:T2:pi;
    t3 = -pi:T3:pi;
    t = -pi:0.01:pi;
    Wc = 2.4;

    % Original signal
    f = 0.5 * (1 + cos(t));

    % Sampled and reconstructed signals
    f1 = 0.5 * (1 + cos(t1));
    F1 = T1 * (Wc/pi) * f1 * sinc(Wc/pi * (ones(length(t1),1) * t - t1' *
ones(1,length(t))));

    f2 = 0.5 * (1 + cos(t2));
    F2 = T2 * (Wc/pi) * f2 * sinc(Wc/pi * (ones(length(t2),1) * t - t2' *
ones(1,length(t))));

    f3 = 0.5 * (1 + cos(t3));
    F3 = T3 * (Wc/pi) * f3 * sinc(Wc/pi * (ones(length(t3),1) * t - t3' *
ones(1,length(t))));

    % Absolute errors
    error1 = abs(F1 - f);
    error2 = abs(F2 - f);
    error3 = abs(F3 - f);

    % Plotting
    figure;

    subplot(3,1,1);
    plot(t, error1);
    title('Absolute Error for T = 0.5s');
    xlabel('t');
    xlim([-3.5, 3.5]);
    ylabel('Absolute Error');
    ylim([0, 0.015]);
    grid on;

    subplot(3,1,2);
    plot(t, error2);
    title('Absolute Error for T = 1.0s');
    xlabel('t');
    xlim([-3.5, 3.5]);
    ylabel('Absolute Error');
    ylim([0, 0.015]);
    grid on;

    subplot(3,1,3);
    plot(t, error3);
    title('Absolute Error for T = 2.0s');
    xlabel('t');
    xlim([-3.5, 3.5]);
```

```
    ylabel('Absolute Error');
    ylim([0, 0.35]);
    grid on;
end

```

### 2.2 对实验1中的信号，绘制抽样间隔为1s下的抽样信号经过DAC一阶保持器后的恢复信号时域波形，体会一阶保持器的基本原理和作用。




```
function ex3_2_2()
    % 抽样间隔为 1s
    T = 1;
    n = -pi:pi;
    t = n * T;
    f = 0.5 * (1 + cos(t));

    % 绘制抽样信号和恢复信号的时域波形
    figure;
    plot(t, f, 'LineWidth', 2, 'DisplayName', '抽样信号');
    hold on;
    stem(t, f, 'Marker', 'o', 'LineWidth', 1.5, 'Color', 'r', 'DisplayName', '恢复信号');
    hold off;

    % 添加标题和标签
    title('抽样信号经过 DAC 一阶保持器后的恢复信号时域波形');
    xlabel('时间 (t)');
    xlim([-3.5, 3.5]);
    ylabel('幅度 (f(t))');

    % 打开网格和添加图例
    grid on;
    legend('Location', 'Best');
end
```

# 五、实验总结

在验证抽样定理的过程中，抽样间隔的选择对信号混叠的影响十分关键。当抽样间隔为0.5s时，大于奈奎斯特抽样频率，相邻信号之间没有发生混叠。而当抽样间隔为2s时，小于奈奎斯特抽样频率，相邻信号之间发生混叠，导致信号失真。这验证了抽样定理的正确性。

另外，对信号的恢复重建过程也进行了探讨。选择合适的抽样间隔是保证信号恢复效果的关键。较小的抽样间隔（对应较大的抽样频率）通常会导致更好的信号恢复效果，而较大的抽样间隔则可能导致信号失真。因此，在实践中需要平衡采样频率和信号失真的需求，以确保在可容忍范围内取得最佳效果。

通过这个实验，不仅巩固了信号的抽样定理和恢复原理，还通过 Matlab 编程锻炼了实际应用的能力。这样的实践有助于深入理解理论知识并提高解决实际问题的能力。

教师评语：

签名：

日期：

成绩：