



电子信息学院实验课程报告

信号与系统实验报告

姓名	： 此处填写姓名
学号	： 此处填写学号
班级	： U08P2108.0X
学院	： X X X X 学院

提交日期：2026年1月18日

目录

课程报告	3
1 常用信号的分类与观察	4
1.1 实验目的:	4
1.2 实验器材:	4
1.3 实验原理:	4
1.4 实验步骤:	5
1.5 实验实验结果及结论:	6
1.6 实验程序及运行结果:	6
2 阶跃响应和冲激响应	10
2.1 实验目的	10
2.2 实验原理	10
2.3 实验器材	10
2.4 实验步骤与数据记录	11
2.5 实验结果及结论	11
2.6 实验程序及运行结果	12
3 连续时间系统的模拟	17
3.1 实验目的	17
3.2 实验器材	17
3.3 实验原理	17
3.4 实验步骤	19
3.5 实验结果	21
3.6 实验程序及运行结果	22
4 实验 4: 滤波器特性测量	26
4.1 实验目的	26
4.2 实验器材	26
4.3 实验原理	26
4.4 实验步骤	28
4.5 实验结果	32
4.6 实验程序及运行结果	34
5 抽样定理与信号恢复	41
5.1 实验目的	41
5.2 实验器材	41
5.3 实验原理	41
5.4 实验步骤	42
5.5 实验结果	43

5.6	实验程序及运行结果	45
6	信号卷积实验	50
6.1	实验目的	50
6.2	实验器材	50
6.3	实验原理	50
6.4	实验步骤	51
6.5	实验结果	53
6.6	实验程序及运行结果	55
7	波形的分解与合成实验	60
7.1	实验目的	60
7.2	实验器材	60
7.3	实验原理	60
7.4	实验步骤	64
7.5	实验结果	65
7.6	实验程序及运行结果	71
大作业报告 (第八次实验)		77
8	噪声抑制算法设计与实现	78
8.1	设计思路	78
8.2	具体实现过程	79
8.3	程序代码及运行结果	80
9	女声转男声音频处理算法设计与实现	88
9.1	设计思路	88
9.2	具体实现过程	89
9.3	程序代码及运行结果	91
10	音频变速算法设计与实现	94
10.1	设计思路	94
10.2	具体实现过程	95
10.3	程序代码和运行结果	96
11	基于能量阈值的语音段自动分割与反转拼接算法实现	99
11.1	设计思路	99
11.2	具体实现过程	100
11.3	程序代码及运行结果	102

课程报告

1 常用信号的分类与观察

1.1 实验目的:

- 观察常用信号的波形，了解其特点及产生方法。
- 学会使用示波器测量常用波形的基本参数，了解信号及信号的特性。

1.2 实验器材:

- 数字信号处理模块 S4

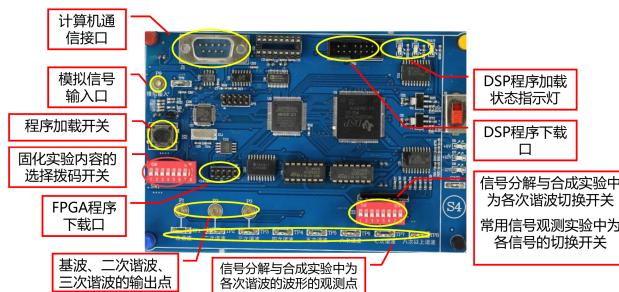


图 1.1. 数字信号处理模块 S4

- 双踪示波器

1.3 实验原理:

对于一个系统特性的研究，其中重要的一个方面是研究它的输入输出关系，即在一特定的输入信号下，系统对应的输出响应信号。因而对信号的研究是对系统研究的出发点，是对系统特性观察的基本手段与方法。在本实验中，将对常用信号和特性进行分析、研究。

信号可以表示为一个或多个变量的函数，在这里仅对一维信号进行研究，自变量为时间。常用信号有：指数信号、正弦信号、指数衰减正弦信号、抽样信号、钟形信号、脉冲信号等。

指数信号：

指数信号可以表示为：

$$x(t) = Ae^{\alpha t}$$

其中，A 为幅值， α 为指数衰减系数。

指数正弦信号：

指数正弦信号的表达式为：

$$x(t) = Ae^{\alpha t} \sin(2\pi ft + \phi)U(t)$$

其中，A 为幅值， α 为指数衰减系数，f 为频率， ϕ 为初相位。

抽样信号：

抽样信号表达式为：

$$Sa(t) = \frac{\sin t}{t}$$

$Sa(t)$ 是一个偶函数，在 t 为 π 的整数倍时，函数数值为 0，此函数在很多应用场合有独特的应用。

钟形信号:

钟形信号表达式为：

$$x(t) = Ae^{-\frac{(t-t_0)^2}{2\sigma^2}}$$

其中，A 为幅值， t_0 为 center 位置， σ 为宽度参数。

脉冲信号:

脉冲信号表达式为：

$$p(t) = U(t - \frac{\tau}{2}) - U(t + \frac{\tau}{2})$$

其中， τ 为脉冲宽度，U(t) 为单位阶跃函数。

方波信号:

方波信号表达式为：

$$x(t) = A \cdot \text{square}(2\pi ft)$$

其中，A 为幅值，f 为频率。

1.4 实验步骤:

将拨码开关 SW1 置为“0000 0001”（开关拨上为 1，拨下为 0），打开实验箱及模块电源，按下复位键 S2 加载常用信号观测功能。将拨码开关 S3 拨为“0000 0000”，点击查看设置。

1. 用示波器观测指数信号波形，并分析测量其所对应的 a 、 K 参数。
 - (a) 拨码开关 S3 第 1 位拨为“1”（从左到右），其他开关拨为“0”，用示波器在 TP1 观察输出的指数信号，并分析测量其对应的频率 a 、 K 参数。
 - (b) 拨码开关 S3 第 2 位拨为“1”（从左到右），其他开关拨为“0”，观察指数信号波形的变化，分析原因。
2. 指数正弦信号观察（正频率信号）。
 - (a) 拨码开关 S3 第 3 位拨为“1”（从左到右），其他开关拨为“0”，用示波器在 TP1 观察输出的指数增长正弦信号。
 - (b) 拨码开关 S3 第 4 位拨为“1”（从左到右），其他开关拨为“0”，注意波形变化情况，分析原因。
3. 抽样信号的观察。
 - (a) 拨码开关 S3 第 5 位拨为“1”（从左到右），其他开关拨为“0”，用示波器在 TP1 处观察输出的抽样信号。
4. 钟形信号的观察。
 - (a) 拨码开关 S3 第 6 位拨为“1”（从左到右），其他开关拨为“0”，用示波器在 TP1 观察输出的钟形信号，观测波形。

注意：该实验不要将拨码开关 S3 的第 7 位和第 8 位拨为“1”。

1.5 实验实验结果及结论:

通过示波器观察常见信号:

以下是通过示波器观察到的常见的信号。

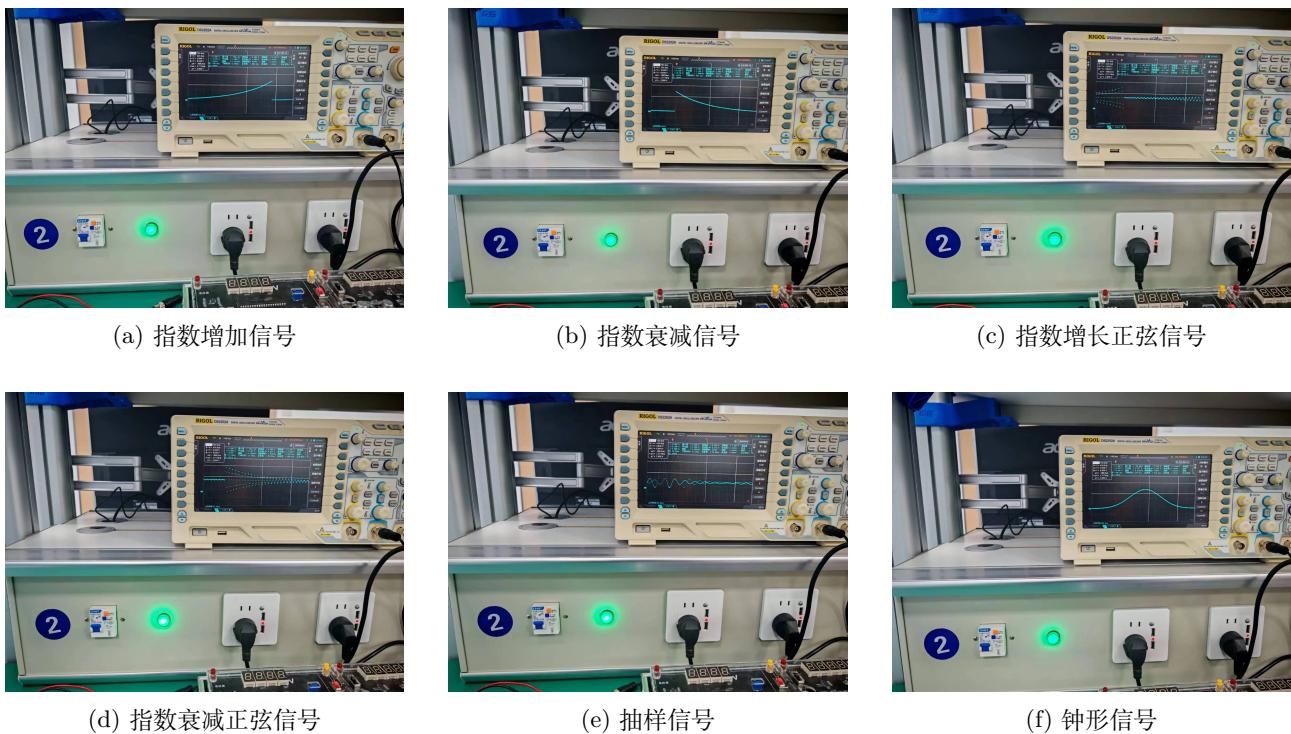


图 1.2. 常用信号观测波形图

指数信号数据记录

以下是原始数据记录, 如表1.1所示。

表 1.1. 指数信号采集数据表

x_1/ms	0.752, 2.304, 2.240, 2.148, 2.004, 1.892, 1.772, 1.628, 1.500, 1.392, 1.276, 1.180, 1.032
y_1/V	0.160, 5.040, 4.240, 3.360, 2.360, 1.800, 1.320, 0.920, 0.680, 0.520, 0.400, 0.320, 0.240
x_2/ms	1.560, 1.652, 1.728, 1.804, 1.900, 1.984, 2.068, 2.176, 2.292, 2.412, 2.500, 2.636, 2.832
y_2/V	5.840, 4.120, 3.400, 2.840, 2.200, 1.800, 1.440, 1.080, 0.840, 0.600, 0.520, 0.400, 0.280

1.6 实验程序及运行结果:

指数信号拟合

通过 MATLAB 曲线拟合器对采集指数信号数据进行指数拟合, 得到拟合曲线, 如图1.3所示。拟合参数参见表1.2。

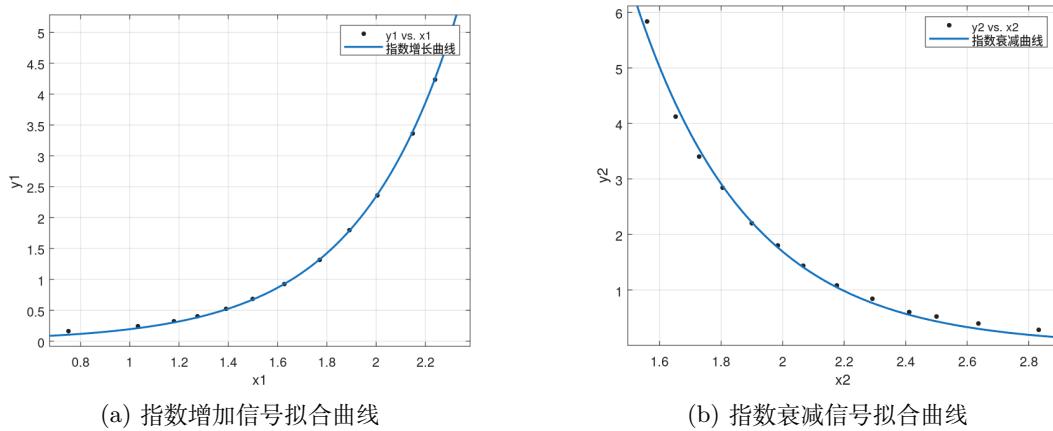


图 1.3. 指数信号拟合结果

表 1.2. 指数增长与衰减曲线拟合结果对比

(a) 指数增长曲线

指数曲线拟合 (exp1)			
$f(x) = a \cdot \exp(b \cdot x)$			
系数和 95% 置信边界			
值	下限	上限	
a	0.0161	0.0147	0.0175
b	2.4915	2.4514	2.5315
拟合优度			
值			
SSE	0.0077		
R^2	0.9998		
DFE	11		
调整 R^2	0.9997		
RMSE	0.0265		

(b) 指数衰减曲线

指数曲线拟合 (exp1)			
$f(x) = a \cdot \exp(b \cdot x)$			
系数和 95% 置信边界			
值	下限	上限	
a	387.63	266	509.27
b	-2.7171	-2.901	-2.5333
拟合优度			
值			
SSE	0.1782		
R^2	0.9948		
DFE	11		
调整 R^2	0.9943		
RMSE	0.1273		

常见信号的绘制

使用 MATLAB 绘制常见信号的代码如下：

Listing 1: 信号波形生成与绘图代码

```

x = linspace(-8,8,200); % 生成时间序列
% 信号处理
y1 = exp(0.3 .* x); % 指数增长信号
y2 = exp(-0.3 .* x); % 指数衰减信号
y3 = sin(x).*exp(0.15.*x); % 正弦指数增长信号
y4 = sin(x).*exp(-0.15*x); % 正弦指数衰减信号
y5 = sinc(1.5.*x./pi); % 采样信号
y6 = exp(-(0.3.*x).^2); % 钟形信号
% 绘图
figure;
subplot(3,2,1);
plot(x,y1);
title('指数增长信号');
subplot(3,2,2);
plot(x,y2);
title('指数衰减信号');
subplot(3,2,3);
plot(x, y3);
title('正弦指数增长信号');
subplot(3,2,4);
plot(x, y4);
title('正弦指数衰减信号');
subplot(3,2,5);
plot(x, y5);
title('采样信号');
subplot(3,2,6);
plot(x, y6);
title('钟形信号');
% 保存图像
handle = gcf; % 当前图形句柄
saveas(handle,'class1_1.jpg')

```

导出图像如图1.4所示。

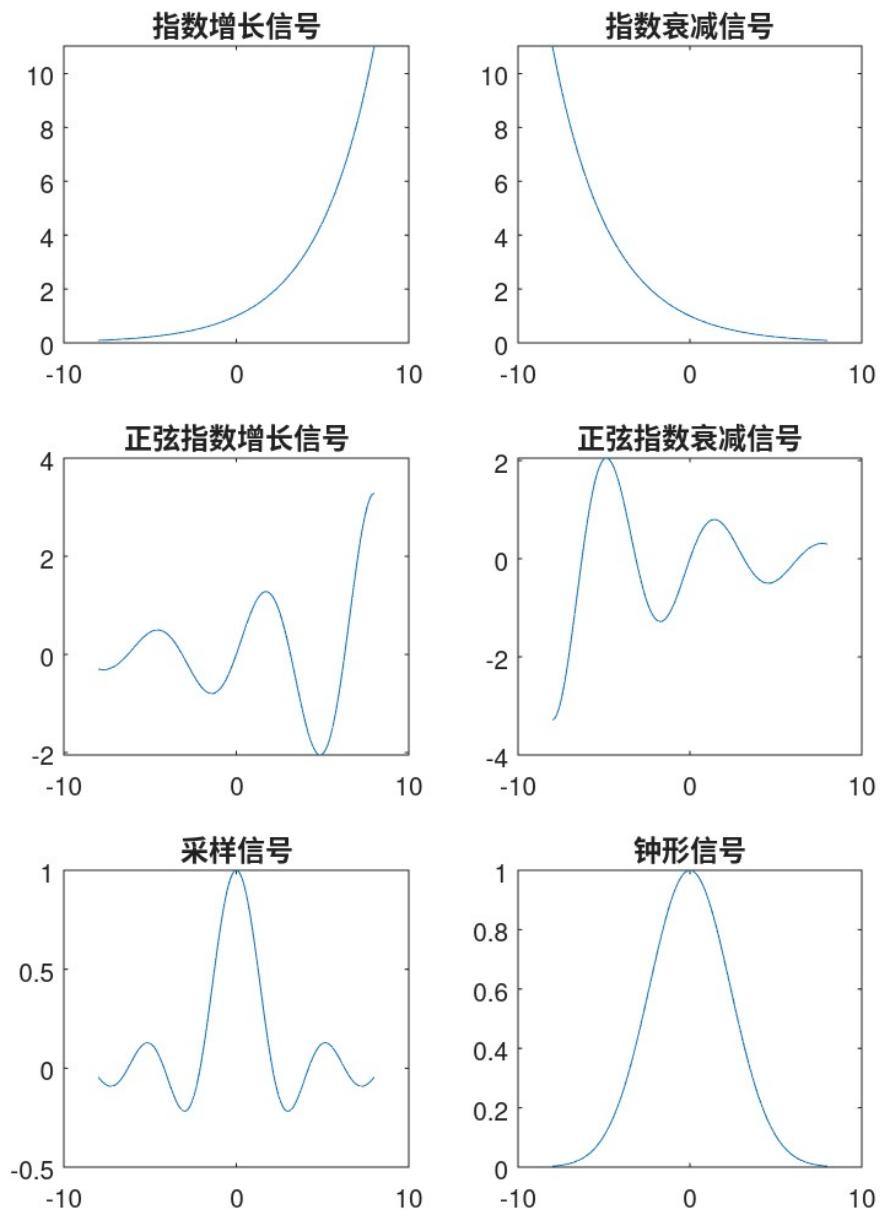


图 1.4. 常见信号波形图

2 阶跃响应和冲激响应

2.1 实验目的

- 观察和测量 RLC 串联电路的阶跃响应与冲激响应的波形和有关参数，并研究其电路元件参数变化对响应的影响。
- 掌握有关信号时域的测量分析方法。

2.2 实验原理

阶跃响应与阶跃激励的关系简单的表示为：

$$g(t) = H[u(t)] \quad u(t) \rightarrow g(t)$$

RLC 串联电路的阶跃响应与冲激响应实验其响应有以下三种状态：

- 欠阻尼状态： $R^2 < \frac{4L}{C}$ ，响应波形为振荡衰减波形。
- 临界阻尼状态： $R^2 = \frac{4L}{C}$ ，响应波形为最快的非振荡衰减波形。
- 过阻尼状态： $R^2 > \frac{4L}{C}$ ，响应波形为非振荡衰减波形。

相应的动态指标定义如下：

- 上升时间 t_r : $y(t)$ 从 0 到第一次上升到稳态值 $y(\infty)$ 的时间。
- 峰值时间 t_p : $y(t)$ 达到第一次峰值的时间。
- 调节时间 t_s : $y(t)$ 进入并保持在稳态值 $y(\infty)$ 的 5% 范围内所需的时间。
- 最大超调量 δ_p : $y(t)$ 达到的最大峰值与稳态值之差与稳态值之比，通常以百分数表示。也即

$$\delta_p = \frac{y_{max} - y(\infty)}{y(\infty)} \times 100\%$$

在满足条件的情况下，RC 电路可以对输入信号进行微分，微分电路如图??，微分电路响应如图??。

2.3 实验器材

- 信号源及频率计模块——一块
- 一阶网络模块——一块
- 数字万用表——一台
- 双踪示波器——一台

2.4 实验步骤与数据记录

任务一：阶跃响应实验波形观察与参数测量

设激励信号为方波，频率为 500Hz。模块关电，进行连线。

1. 模块开电，调整激励信号源为方波（即从模块 S2 中的 P2 端口引出方波信号）；调节频率调节旋钮 ROL1，使频率计示数 $f = 500\text{Hz}$ 。
2. 连接模块 S2 的方波信号输出端 P2 至模块 S5 中的 P12。
3. 示波器 CH1 接于 TP14，调整 W1，使电路分别工作于欠阻尼、临界和过阻尼三种状态，观察各种状态下的输出波形，用万用表测量与波形对应的 P12 和 P13 两点间的电阻值（测量时应断开电源），并将实验数据填入表2.1中。
4. TP12 为输入信号波形的测量点，可把示波器的 CH2 接于 TP12 上，便于波形比较。

表 2.1. 阶跃响应实验记录表

参数测量	状态		
	欠阻尼状态 $R < 2\sqrt{\frac{L}{C}}$	临界状态 $R = 2\sqrt{\frac{L}{C}}$	过阻尼状态 $R > 2\sqrt{\frac{L}{C}}$
$R =$			
TP12 激励波形			
TP14 响应波形			

注：描绘波形要使三种状态的 X 轴坐标（扫描时间）一致。

任务二：冲激响应的波形观察

冲激信号是由阶跃信号经过微分电路而得到。模块关电，进行连线。

1. 将方波输入信号接于 P10（输入信号频率与幅度不变）。
2. 连接 P11 与 P12。
3. 将示波器的 CH1 接于 TP11，观察经微分后响应波形（等效为冲激激励信号）。
4. 将示波器的 CH2 接于 TP14，调整 W1，使电路分别工作于欠阻尼、临界和过阻尼三种状态。
5. 观察电路处于以上三种状态时激励信号与响应信号的波形，并将观察结果填入表2.2中。

表 2.2. 冲激响应实验记录表

参数测量	状态		
	欠阻尼状态 $R < 2\sqrt{\frac{L}{C}}$	临界状态 $R = 2\sqrt{\frac{L}{C}}$	过阻尼状态 $R > 2\sqrt{\frac{L}{C}}$
$R =$			
TP11 激励波形			
TP14 响应波形			

2.5 实验结果及结论

阶跃响应的波形如图2.1所示，冲激响应的波形如图2.2所示。

根据波形可填表2.1和表2.2如下：

图 2.1. 方波输入 (阶跃响应) 的波形观察

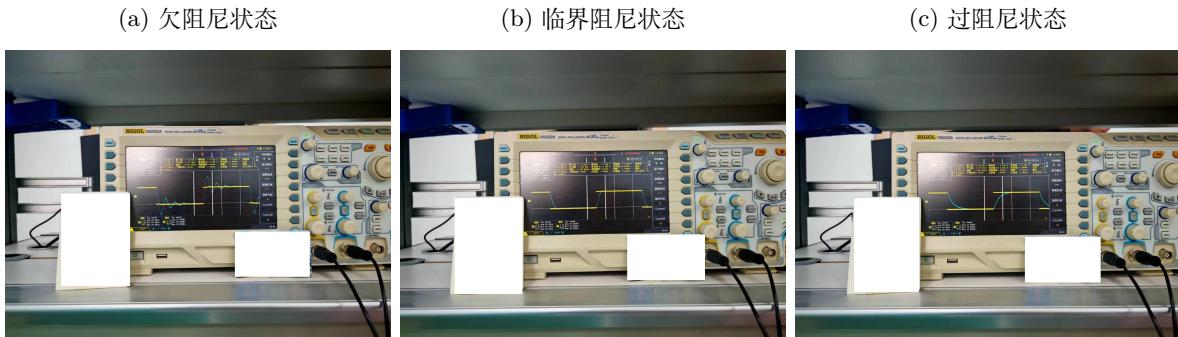


图 2.2. 脉冲输入 (冲激响应) 的波形观察

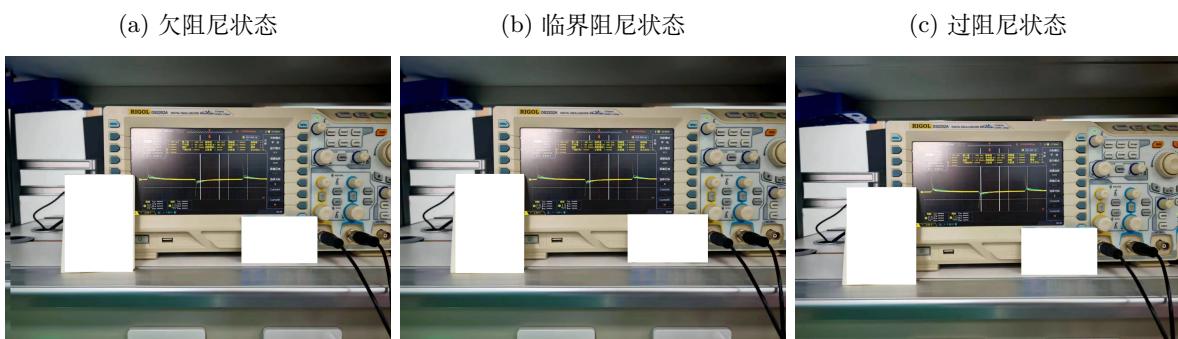


表 2.3. 阶跃响应实验记录表 (填写结果)

参数测量	状态		
	欠阻尼状态 $R < 2\sqrt{\frac{L}{C}}$	临界状态 $R = 2\sqrt{\frac{L}{C}}$	过阻尼状态 $R > 2\sqrt{\frac{L}{C}}$
$R =$			
TP12 激励波形	方波	方波	方波
TP14 响应波形	振荡衰减波形	最快非振荡衰减波形	非振荡衰减波形

表 2.4. 冲激响应实验记录表 (填写结果)

参数测量	状态		
	欠阻尼状态 $R < 2\sqrt{\frac{L}{C}}$	临界状态 $R = 2\sqrt{\frac{L}{C}}$	过阻尼状态 $R > 2\sqrt{\frac{L}{C}}$
$R =$			
TP11 激励波形	冲激波形	冲激波形	冲激波形
TP14 响应波形	振荡衰减波形	最快非振荡衰减波形	非振荡衰减波形

2.6 实验程序及运行结果

问题一

Listing 2: 信号波形生成与绘图代码

```
t = sym('t');

ut = heaviside(t);

f1 = (2 - exp(-2*t)) .* ut;
```

```

f2 = cos(pi * t /2)* (heaviside(t) - heaviside(t-4));

f3 = exp(t)* cos(t)* ut;

f4 = 2/3 *t *heaviside(t +2);

figure;

subplot(2,2,1);
fplot(f1);
title("signal 1");

subplot(2,2,2);
fplot(f2);
title("signal 2");

subplot(2,2,3);
fplot(f3);
title("signal 3");

subplot(2,2,4);
fplot(f4);
title("signal 4");

saveas(gcf, './p1_signals.png');

```

运行结果如下图2.3

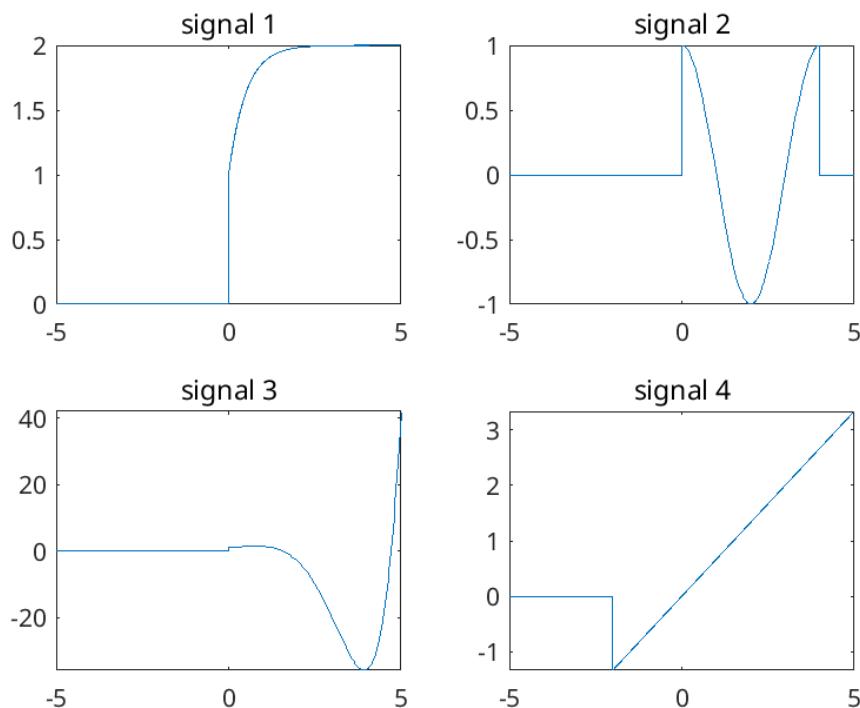


图 2.3. 问题一信号波形

问题二

Listing 3: 信号波形生成与绘图代码

```

t = sym('t');

f = 2*heaviside(t) - 2*heaviside(t-1) + heaviside (t-1) - heaviside(t-2);

f1 = subs(f, t, -t);

f2 = subs(f, t, t-2);

f3 = subs(f, t, 0.5*t);
f31 = subs(f, t, 2*t);

f4 = subs(f, t, 0.5*t +1);

figure;

subplot(2,2,1);
fplot(f1);
title("f(-t)");

subplot(2,2,2);
fplot(f2);
title("f(t-2)");

subplot(2,2,3);
fplot(f3);
hold on;
fplot(f31);
title("f(0.5t) and f(2t)");
hold off;

subplot(2,2,4);
fplot(f4);
title("f(0.5t +1)");

saveas(gcf, './p2_signals.png');

```

运行结果如下图2.4

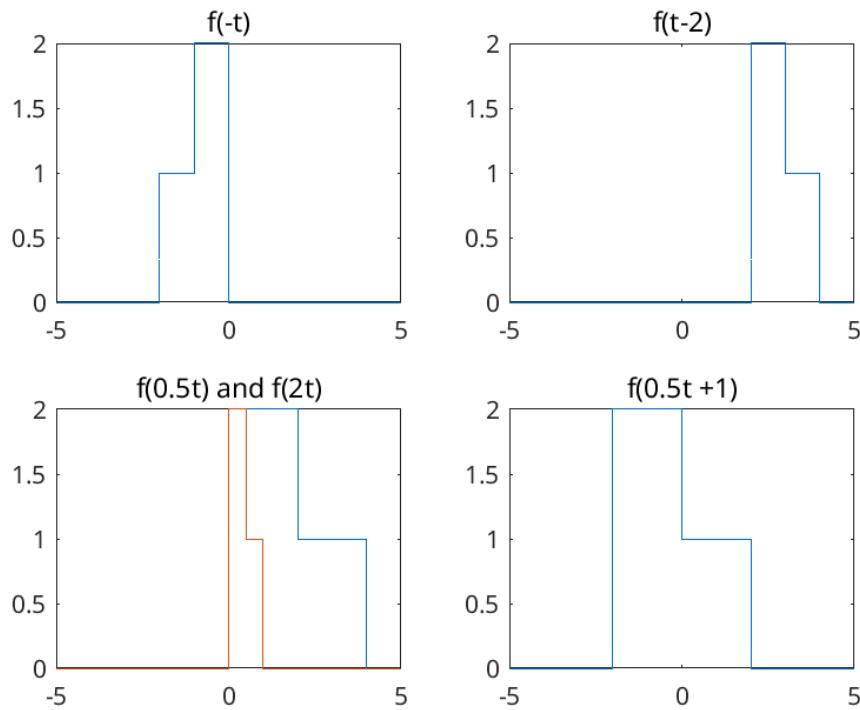


图 2.4. 问题二信号波形

问题三

Listing 4: 信号波形生成与绘图代码

```
t= sym('t');

ft = 2* exp(j*(t+ pi/4));

realf = real(ft);
imagf = imag(ft);
absf = abs(ft);
angf = angle(ft);

figure;

subplot(2,2,1);
fplot(t, realf);
title('Real Part of f(t)');

subplot(2,2,2);
fplot(t, imagf);
title('Imaginary Part of f(t)');

subplot(2,2,3);
fplot(t, absf);
title('Magnitude of f(t)');

subplot(2,2,4);
```

```
fplot(t, angf);
title('Phase of f(t)');
saveas(gcf, './p3.png');
```

运行结果如下图2.5

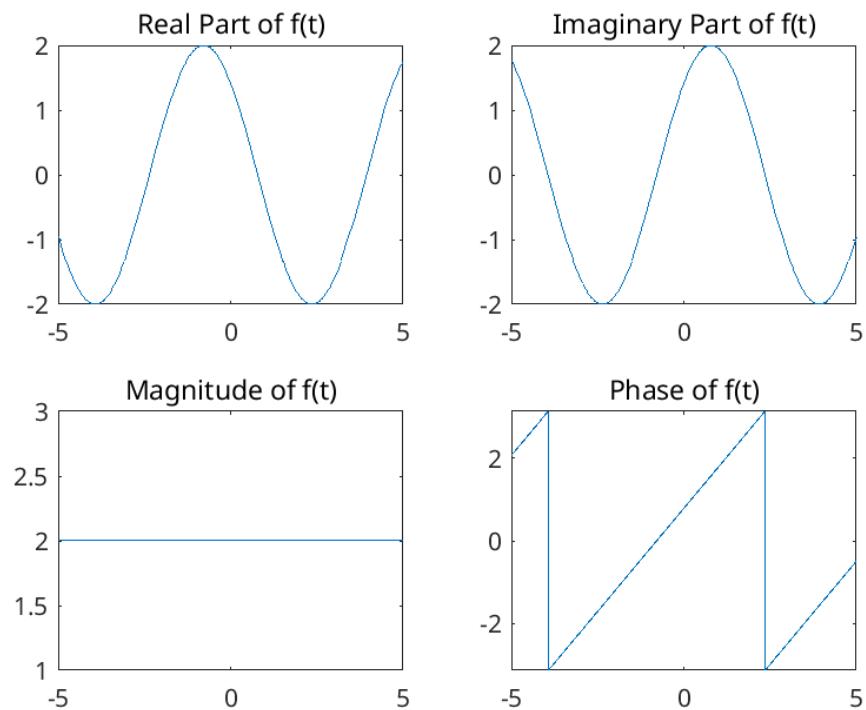


图 2.5. 问题三信号波形

3 连续时间系统的模拟

3.1 实验目的

1. 了解基本运算器——比例放大器、加法器和积分器的电路结构和运算功能。
2. 掌握用基本运算单元模拟连续时间一阶系统原理与测试方法。

3.2 实验器材

- | | |
|--------------------|-----|
| 1. 双踪示波器 | 1 台 |
| 2. 数字万用表 | 1 块 |
| 3. 电压表及直流信号源模块S1 | 1 块 |
| 4. 信号源及频率计模块S2 | 1 块 |
| 5. 基本运算单元与连续系统模块S9 | 1 块 |

3.3 实验原理

1、线性系统的模拟

利用运算放大器组成基本的运算单元（放大器、加法器、积分器等）来模拟实际系统传输特性。

2、三种基本运算电路

a、比例放大器 比例放大器的输入输出关系为：

$$u_0 = -\frac{R_2}{R_1} \cdot u_1$$

比例放大器的电路连线示意如图3.1所示。

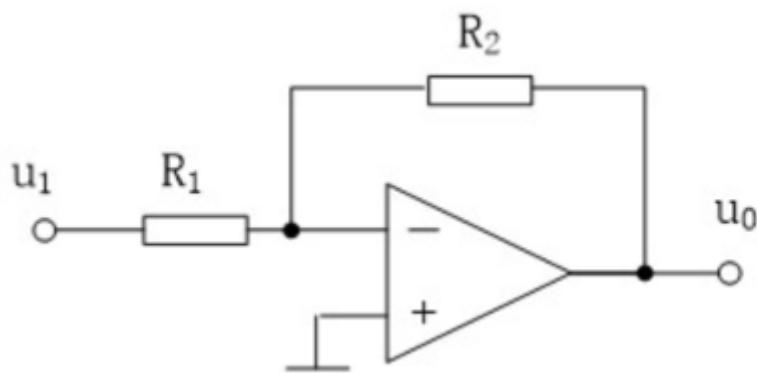


图 3.1. 比例放大电路连线示意图

b、加法器 加法器的输入输出关系为:

$$u_0 = -\frac{R_2}{R_1}(u_1 + u_2)$$

当 $R_1 = R_2$ 时, 简化为:

$$u_0 = -(u_1 + u_2)$$

加法器的电路连线示意如图3.2所示。

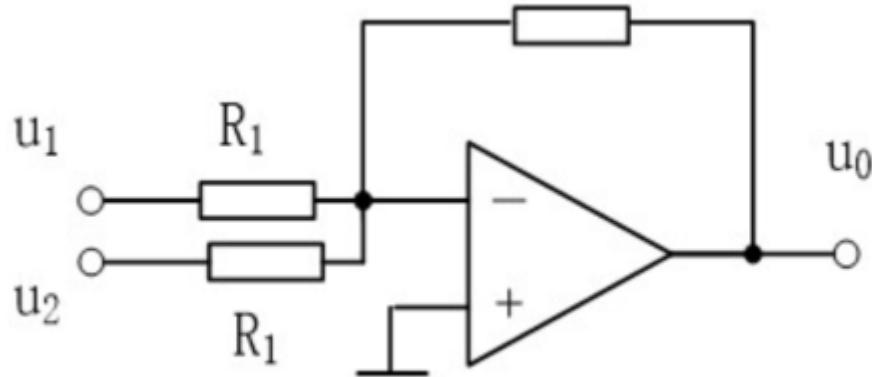


图 3.2. 加法器电路连线示意图

c、积分器 积分器的输入输出关系为:

$$u_0 = -\frac{1}{RC} \int u_1 dt$$

积分器的电路连接示意如图3.3所示。

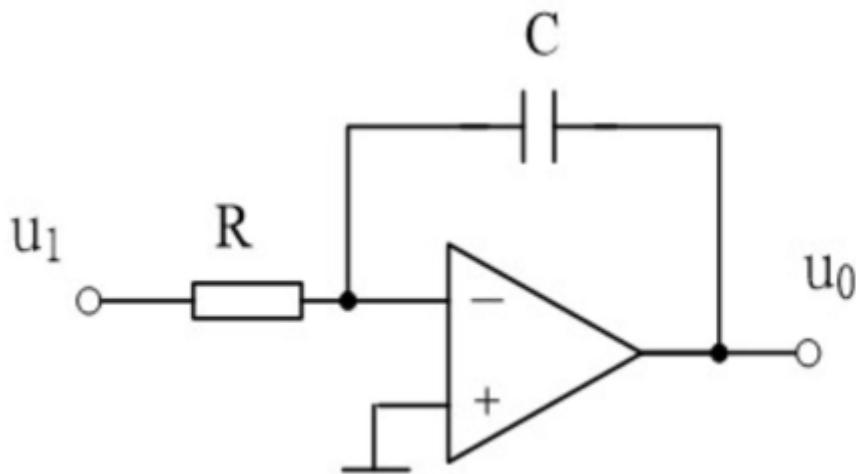


图 3.3. 积分器电路连接示意图

3、一阶系统的模拟

图3.4 (3.4a) 为一阶RC电路，可用微分方程描述：

$$\frac{dy(t)}{dt} + \frac{1}{RC}y(t) = \frac{1}{RC}x(t)$$

其模拟框图如图3.4 (3.4b) 和图3.4 (3.4c) 所示，二者数学关系等效，对应的积分关系分别为：

$$\int \left[x(t) \frac{1}{RC} - y(t) \frac{1}{RC} \right] dt = y(t)$$

$$\int [-x(t) + y(t)] \left(-\frac{1}{RC} \right) dt = y(t)$$

一阶系统模拟实验电路如图3.4 (3.4d) 所示。

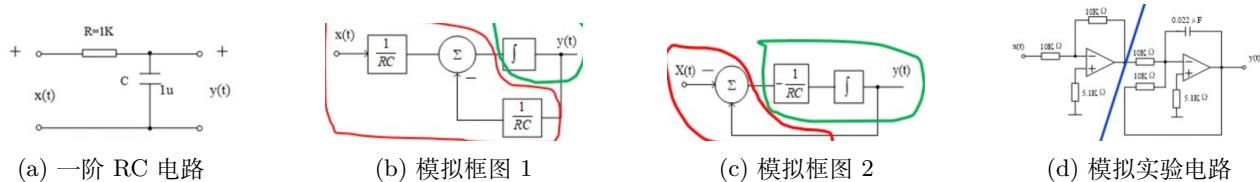


图 3.4. 一阶系统的电路与模拟框图

3.4 实验步骤

任务一基本运算器——加法器的观测

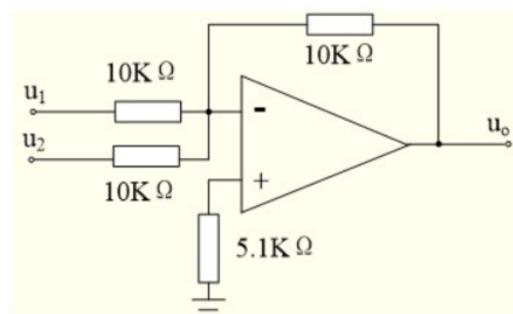


图 3.5. 加法器实验电路图

模块关电，按图3.5所示连接实验电路。

将模块S1的直流输出 1 (P1) 和直流输出 2 (P2) 分别接至加法器的 u_1 和 u_2 。模块开电，适当调节模块S1中的 W1、W2，并记录 P1 和 P2 的电压值。

用万用表或示波器测量输出 u_0 端电压，验证反相加法器“输出为两路输入之和取反”的特性，完成下表：

输入一	输入二	输出
电压 (V)	电压 (V)	电压 (V)

将输入信号改为“幅度 2V、频率 500Hz 的方波”，观察输入/输出波形并补充表格。

任务二 基本运算器——比例放大器的观测

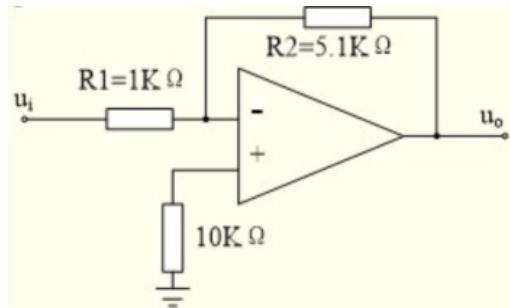


图 3.6. 比例放大器实验电路图

模块关电，连接如图3.6所示实验电路； $R_1 R_2$ 可选 2 组不同电阻值以改变放大比例。

模块开电，将“幅度 1V、频率 1KHz 的正弦波”送入输入端，用示波器观测输入/输出波形，完成下表：

电阻		输入		输出	
		电压 (V)	波形	电压 (V)	波形
$R1 = 1\text{K}\Omega$	$R2 = 5.1\text{K}\Omega$				
$R1 =$	$R2 =$				

任务三 基本运算器——积分器的观测

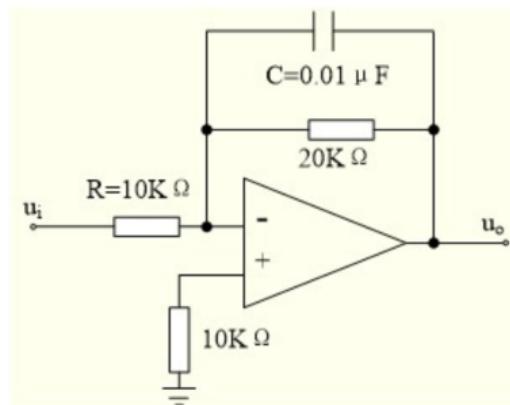


图 3.7. 积分器实验电路图

模块关电，连接如图3.7所示实验电路（ $20\text{K}\Omega$ 电阻可由两个 $10\text{K}\Omega$ 电阻串联代替）。

模块开电，用模块S2产生 $f = 1\text{KHz}$ 的方波送入输入端，用示波器观测输入/输出波形并记录数据。

任务四一阶 RC 电路的模拟

图3.4 (3.4a) 为一阶 RC 电路，模块关电后，按图3.4 (3.4d) 连接其一阶模拟电路 ($0.022\mu\text{F}$ 电容可由两个 $0.01\mu\text{F}$ 电容并联代替)。

模块开电，将“幅度 2V、频率 1KHz 的方波”送入一阶模拟电路输入端，用示波器观测输出电压波形，验证模拟效果。

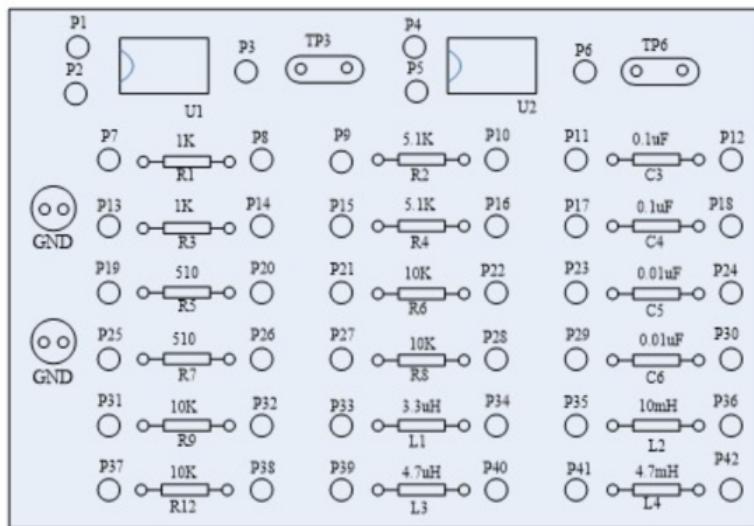


图 3.8. 模块实验电路图

3.5 实验结果

任务一：基本运算器——加法器的观测

结果如下表3.1

表 3.1. 加法器实验结果表

输入一	输入二	输出
电压 (V)	电压 (V)	电压 (V)
1	1	-2
1	-0.5	-0.5
1	0.5	-1.5

任务二：基本运算器——比例放大器的观测

结果如下表3.2

表 3.2. 比例放大器实验结果表

电阻	输入		输出	
	电压 (V)	波形	电压 (V)	波形
$R1 = 1\text{K}\Omega$ $R2 = 5.1\text{K}\Omega$	1 V	正弦波	2 V	正弦波 (反相)
$R1 = 10\text{K}\Omega$ $R2 = 5.1\text{K}\Omega$	1.92 V	正弦波	9.6 V	正弦波 (反相)

波形如图3.9所示。

(a) $R_1=1\text{K}\Omega$, $R_2=5.1\text{K}\Omega$ (b) $R_1=10\text{K}\Omega$, $R_2=5.1\text{K}\Omega$

图 3.9. 比例放大器输入输出波形

任务三：基本运算器——积分器的观测

积分电路波形如下图3.10所示。

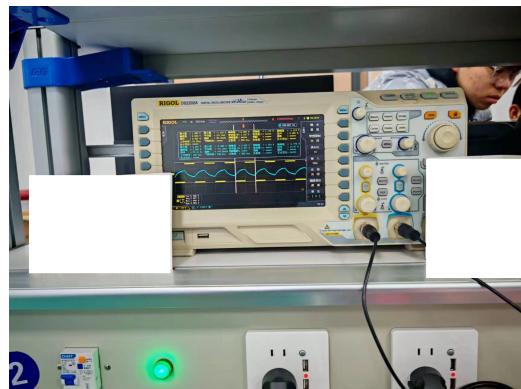


图 3.10. 积分器输入输出波形

任务四：一阶 RC 电路的模拟

电路波形如下图3.11所示。

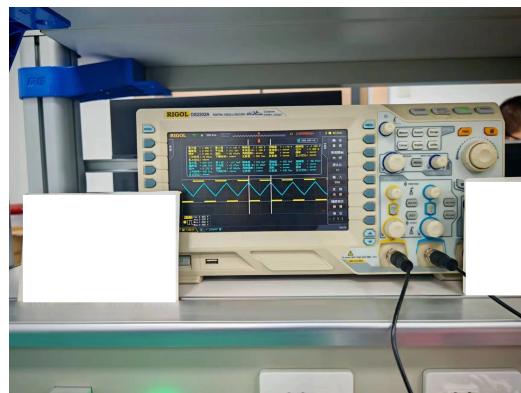


图 3.11. 一阶 RC 电路模拟输入输出波形

3.6 实验程序及运行结果**问题一运行结果**

Listing 5: 信号波形生成与绘图代码

```

sys = tf([0,3,2],[1,5,6]);
t = -3:0.01:10;
yd = impulse(sys,t);
yu = step(sys,t);

figure;
subplot(3,1,1);
plot(t,yd);
title('Impulse Response');

subplot(3,1,2);
plot(t,yu);
title('Step Response');

f= exp(-2.*t).*heaviside(t);
yf = lsim(sys,f,t);

subplot(3,1,3);
plot(t,yf);
title('Response to f(t) = e^{-2t}u(t)');

saveas(gcf,'./p1.png');

```

问题一运行结果如下图3.12所示。

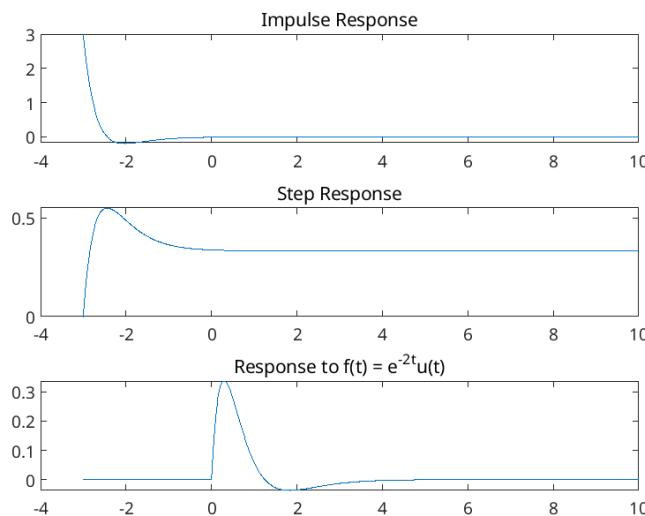


图 3.12. 问题一运行结果图

问题二运行结果

Listing 6: 信号波形生成与绘图代码

```

sys = tf([0,0,4],[0,1,12]);

```

```

t = -3:0.01:10;

yd = impulse(sys,t);
yu = step(sys,t);

f = 12 .* heaviside(t);

yf = lsim(sys,f,t);

figure;
subplot(3,1,1);
plot(t,yd);
title('Impulse Response');

subplot(3,1,2);
plot(t,yu);
title('Step Response');

subplot(3,1,3);
plot(t,yf);
title('Response to f(t) = 12u(t)');

saveas(gcf,'./p2.png');

```

问题二运行结果如下图3.13所示。

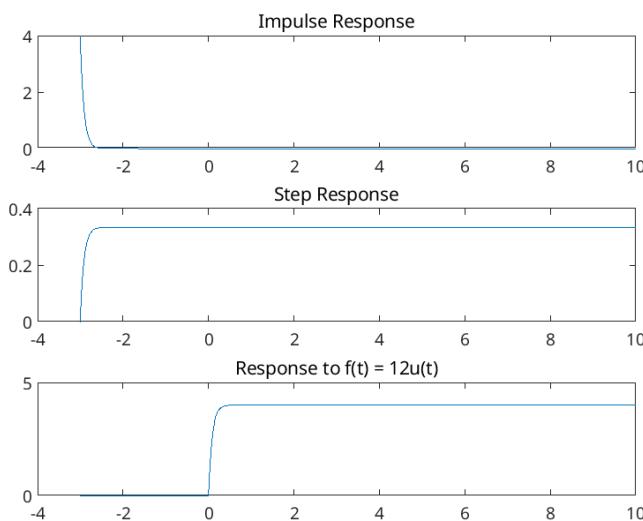


图 3.13. 问题二运行结果图

问题三运行结果

Listing 7: 信号波形生成与绘图代码

```

sys = tf([0,1,0],[1,1,100]);

t = -3:0.01:10;

yd = impulse(sys,t);

```

```
yu = step(sys,t);

figure;
subplot(2,1,1);
plot(t,yd);
title('Impulse Response');

subplot(2,1,2);
plot(t,yu);
title('Step Response');

saveas(gcf,'./p3.png');
```

问题三运行结果如下图3.14所示。

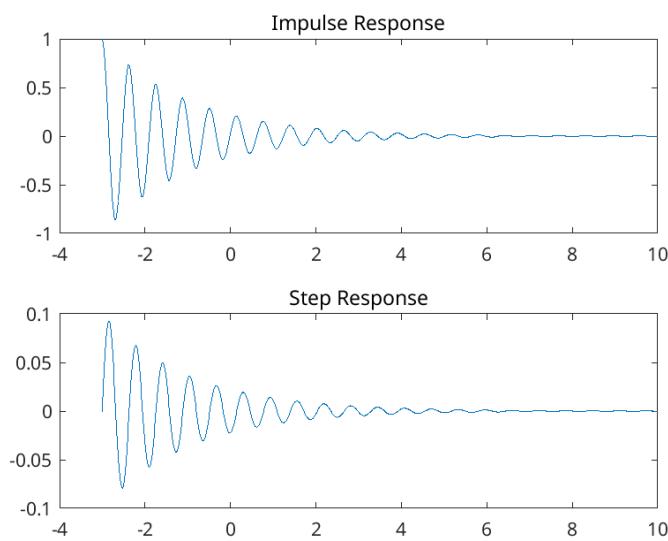


图 3.14. 问题三运行结果图

4 实验 4：滤波器特性测量

4.1 实验目的

1、熟悉滤波器构成及其特性。2、学会测量滤波器幅频特性的方法。

4.2 实验器材

1. 双踪示波器：1 台
2. 信号源及频率计模块：1 块
3. 抽样定理及滤波器模块：1 块

4.3 实验原理

滤波器是一种能使有用频率信号通过而抑制（或大为衰减）无用频率信号的电子装置，工程上常用于信号处理、数据传送和抑制干扰等，本实验主要讨论模拟滤波器。

以往滤波电路主要采用无源元件 R、L 和 C 组成；60 年代以来，集成运放迅速发展，由其与 R、C 组成的有源滤波电路，具有不用电感、体积小、重量轻等优点。此外，集成运放的开环电压增益和输入阻抗均很高，输出阻抗又低，构成的有源滤波电路还具有一定的电压放大和缓冲作用，但集成运放的带宽有限，导致有源滤波电路的工作频率难以做得很髙。

初步定义

滤波电路的一般结构如图4.1所示，图中 $v_I(t)$ 表示输入信号， $v_O(t)$ 为输出信号。假设滤波器是线性时不变网络，则在复频域内有：

$$A(s) = \frac{V_O(s)}{V_I(s)}$$

式中 $A(s)$ 是滤波电路的电压传递函数（一般为复数）。对于实际频率 ($s = j\omega$)，则有：

$$A(j\omega) = |A(j\omega)| e^{j\varphi(\omega)}$$

其中， $|A(j\omega)|$ 为传递函数的模， $\varphi(\omega)$ 为其相位角。



图 4.1. 滤波器电路的一般结构

二阶 RC 滤波器的传输函数如下表所示：

类型	传输函数
低通	$A(s) = \frac{A_0 \omega_0}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2}$
高通	$A(s) = \frac{A_0 s^2}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2}$
带通	$A(s) = \frac{A_0 \frac{\omega_0}{Q} s}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2}$
带阻	$A(s) = \frac{A_V (s^2 + \omega_0^2)}{s^2 + \frac{\omega_0}{Q} s + \omega_0^2}$

备注: 电压增益 A_V ; ω_c —低、高通滤波器的截止角频率; ω_0 —带通、带阻滤波器的中心角频率; Q —品质因数, $Q \approx \omega_0/BW$ 或 f_0/BW (当 $BW \ll \omega_0$ 时); BW —带通、带阻滤波器的带宽。

此外, 滤波电路中需关注时延 $\tau(\omega)$, 其定义为:

$$\tau(\omega) = -\frac{d\varphi(\omega)}{d\omega} \quad (4-2)$$

通常用幅频响应表征滤波电路特性, 欲使信号通过滤波器的失真很小, 需考虑相位和时延响应。当相位响应 $\varphi(\omega)$ 呈线性变化 (即时延响应 $\tau(\omega)$ 为常数) 时, 输出信号可避免失真。

滤波电路的分类

对于幅频响应, 能通过的信号频率范围定义为通带, 受阻或衰减的信号频率范围称为阻带, 通带和阻带的界限频率叫做截止频率 f_c 。

理想滤波电路在通带内应具有零衰减的幅频响应和线性的相位响应, 而在阻带内应具有无限大的幅度衰减 ($|A(j\omega)| = 0$)。根据通带和阻带的相互位置, 滤波电路通常分为以下四类, 其幅频响应如图4.2所示:

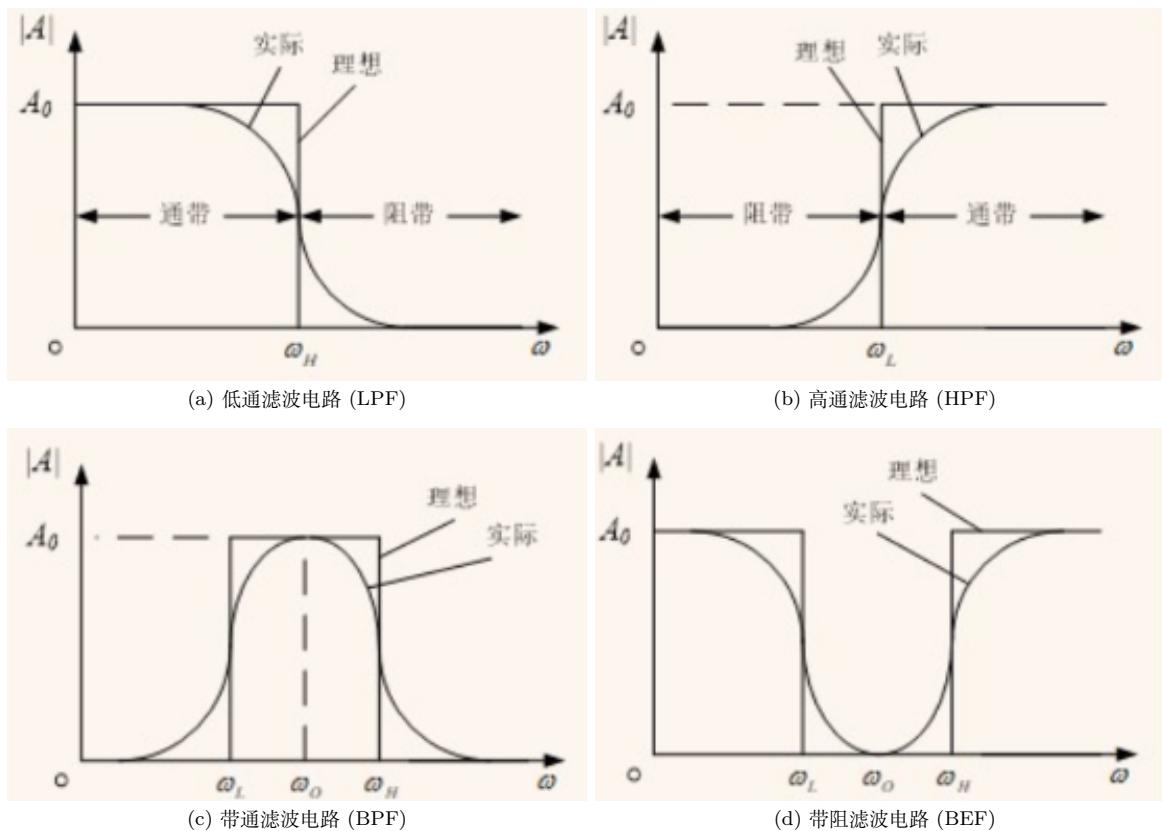


图 4.2. 各种滤波电路的幅频响应

- 低通滤波电路: 通过从零到截止角频率 ω_H 的低频信号, 对大于 ω_H 的频率完全衰减, 带宽 $BW = \omega_H$ 。
- 高通滤波电路: 阻带为 $0 < \omega < \omega_L$, 通带为 $\omega > \omega_L$, 理论带宽 $BW = \infty$, 实际受有源器件带宽限制。
- 带通滤波电路: 有两个阻带 ($0 < \omega < \omega_L$ 和 $\omega > \omega_H$), 带宽 $BW = \omega_H - \omega_L$, ω_0 为中频角频率。
- 带阻滤波电路: 有两个通带 ($0 < \omega < \omega_L$ 和 $\omega > \omega_H$) 和一个阻带 ($\omega_L < \omega < \omega_H$), 功能是衰减 ω_L 到 ω_H 间的信号, ω_0 为中频角频率, 实际通带 $\omega > \omega_H$ 受有源器件限制。

滤波器特性测量方法

研究滤波器特性需考察幅频特性曲线和相频特性曲线，常用测量方法有两种：

1. 点频法（逐点测量法）：保持输入电压不变，人工逐点改变信号发生器的频率，记录各点对应的输出电压幅度，在直角坐标平面描绘幅度-频率曲线。
 - 优点：测量准确度高，不需要特殊仪器。
 - 缺点：操作繁琐，容易漏掉某些细节，不能反应被测网络动态特性。
2. 扫频法：利用扫频信号发生器产生电压恒定、频率随时间线性变化的信号，实现输入信号频率的自动调节。
 - 优点：操作简单，速度快，频率连续变化能覆盖更多频点，可反应网络动态特性，能观察到系统脉冲干扰等冲激变化。
 - 缺点：测量准确度比点频法低。

正弦扫频信号（Sine Sweep）可作为系统激励和测取系统传递函数的有效方法，广泛应用于科研及生产中，在滤波器设计中常用其测量系统频率特性。

模块参数与扫频设置

扫频设置方法 信号源及频率计模块的扫频设置步骤如下：

1. 将模块中扫频开关 S3 拨至“ON”，按下“扫频设置”按钮 S5，此时“下限”指示灯亮，调节“ROL1”旋钮设置扫频下限频率；
2. 再次按下“扫频设置”按钮，“上限”指示灯亮，调节“ROL1”旋钮设置扫频上限频率；
3. 扫频范围设置完成后，再按一下“扫频设置”按钮，“分辨率”指示灯亮，此时扫频范围上方“上限”和“下限”指示灯亮，频率计数码管右方的“MHz”“Hz”指示灯熄灭；
4. 调节“ROL1”设置“下限频率”和“上限频率”之间的频点数（频点数越少，扫频速度越快；反之则越慢）；
5. 扫频参数设置完成后，按下“扫频设置”即可输出扫频信号。

滤波器固有参数 各滤波器的固有截止频率/中心频率参数如下：

- 低通：无源 20kHz，有源 17kHz；
- 高通：无源 14.5kHz，有源 14.5kHz；
- 带通：无源 $f_L = 1.3\text{kHz}$ 、 $f_H = 18.5\text{kHz}$ ；有源 $f_L = 2.4\text{kHz}$ 、 $f_H = 20.8\text{kHz}$ ；
- 带阻：无源 $f_L = 4.1\text{kHz}$ 、 $f_H = 65.2\text{kHz}$ ；有源 $f_L = 6.5\text{kHz}$ 、 $f_H = 38\text{kHz}$ 。

4.4 实验步骤

实验中信号源的输入信号均为 4V 左右的正弦波，设置模块：切换波形开关 S4，使“SIN”指示灯亮，调节“模拟输出幅度调节”旋钮，使信号幅度为 4V。

任务一：测量低通滤波器的频响特性

逐点测量法

1. 模块关电，连接模块 2 中模拟信号输出端 P2 与模块 3 中 P1（低通无源），保持输入信号幅度为 4V 不变（如图4.3a）；
2. 模块开电，逐渐改变输入信号频率，用示波器观测 TP2 处信号波形的峰峰值；
3. 将数据填入表4.1中；
4. 模块关电，连接模块 2 中 P2 与模块 3 中 P5（低通有源）（如图4.3b）；
5. 模块开电，逐渐改变输入信号频率，用示波器观测 TP6 处信号波形的峰峰值；
6. 将数据填入表4.2中。

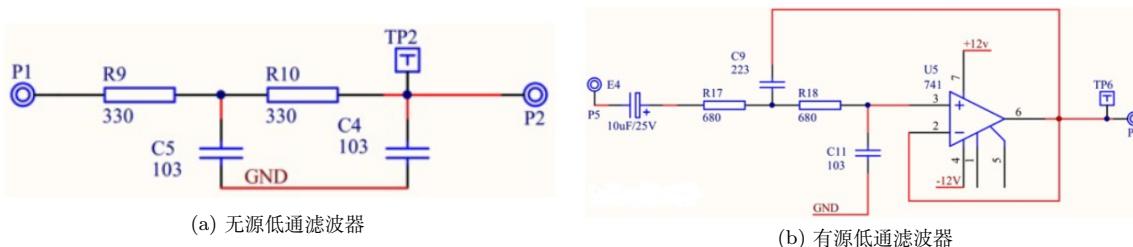


图 4.3. 低通滤波器连接示意图

表 4.1. 低通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$											
$V_O(V)$											

表 4.2. 低通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$											
$V_O(V)$											

扫频法测量

1. 将扫频范围设置为 100Hz~25kHz；
2. 把示波器 CH1 连接到信号源输出 P2 处（示波器调为直流测试档），此时 P2 输出扫频信号；
3. 分别将扫频信号输入到无源低通、有源低通滤波器的输入端；
4. 模块开电，对比观察输入输出信号，拍照记录波形。

任务二：测量高通滤波器的频响特性

逐点测量法

1. 模块关电，保持信号源输出正弦信号幅度不变，连接模块 2 中 P2 与模块 3 中 P3（高通无源）端口（如图4.4a）；

2. 模块开电，逐渐改变输入信号频率，用示波器观测 TP4 处信号波形的峰峰值；
3. 将数据填入表4.3中；
4. 模块关电，连接模块 2 中 P2 与模块 3 中 P7（高通有源）（如图4.4b）；
5. 模块开电，逐渐改变输入信号频率，用示波器观测 TP8 处信号波形的峰峰值；
6. 将数据填入表4.4中。

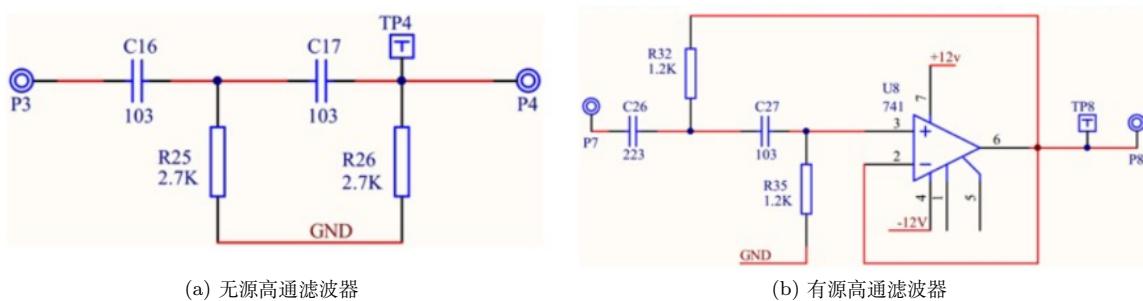


图 4.4. 高通滤波器连接示意图

表 4.3. 高通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$												
$V_O(V)$												

表 4.4. 高通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$												
$V_O(V)$												

扫频法测量

1. 将扫频范围设置为 100Hz~25kHz；
2. 把示波器连接到信号源输出 P2 处（示波器调为直流测试档），此时 P2 输出扫频信号；
3. 分别将扫频信号输入到无源高通、有源高通滤波器的输入端；
4. 模块开电，对比观察输入输出信号，拍照记录波形。

任务三：测量带通滤波器的频响特性

逐点测量法

1. 模块关电，保持信号源输出正弦波幅度为 4V 不变，连接模块 2 中 P2 与模块 3 中 P9（带通无源）（如图4.5a）；
2. 模块开电，逐渐改变输入信号频率，用示波器观测 TP10 处信号波形的峰峰值；
3. 将数据填入表4.5中；

4. 模块关电，连接模块 2 中 P2 与模块 3 中 P13（带通有源）（如图4.5b）；
5. 模块开电，逐渐改变输入信号频率，用示波器观测 TP14 处信号波形的峰峰值；
6. 将数据填入表4.6中。

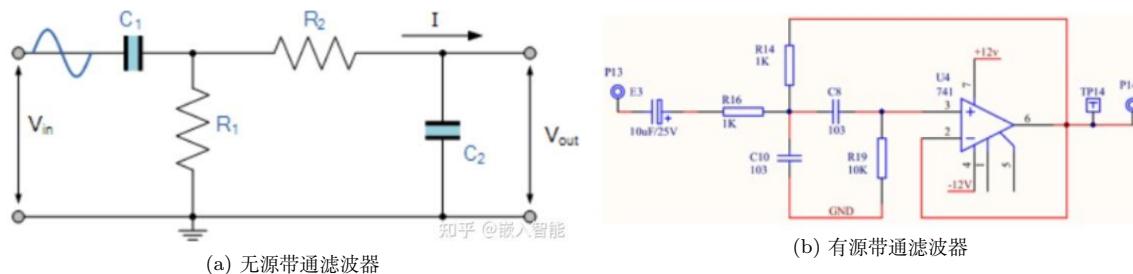


图 4.5. 带通滤波器连接示意图

表 4.5. 带通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$											
$V_O(V)$											

表 4.6. 带通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$											
$V_O(V)$											

扫频法测量

1. 将扫频范围设置为 100Hz~80kHz；
2. 把示波器连接到信号源输出 P2 处（示波器调为直流测试档），此时 P2 输出扫频信号；
3. 分别将扫频信号输入到无源带通、有源带通滤波器的输入端；
4. 模块开电，对比观察输入输出信号，拍照记录波形。

任务四：测量带阻滤波器的频响特性

逐点测量法

1. 模块关电，保持信号源输出正弦波幅度为 4V 不变，连接模块 2 中 P2 与模块 3 中 P11（带阻无源）（如图4.6a）；
2. 模块开电，逐渐改变输入信号频率，用示波器观测 TP12 处信号波形的峰峰值；
3. 将数据填入表4.7中；
4. 模块关电，连接模块 2 中 P2 与模块 3 中 P15（带阻有源）（如图4.6b）；
5. 模块开电，逐渐改变输入信号频率，用示波器观测 TP16 处信号波形的峰峰值；
6. 将数据填入表4.8中。

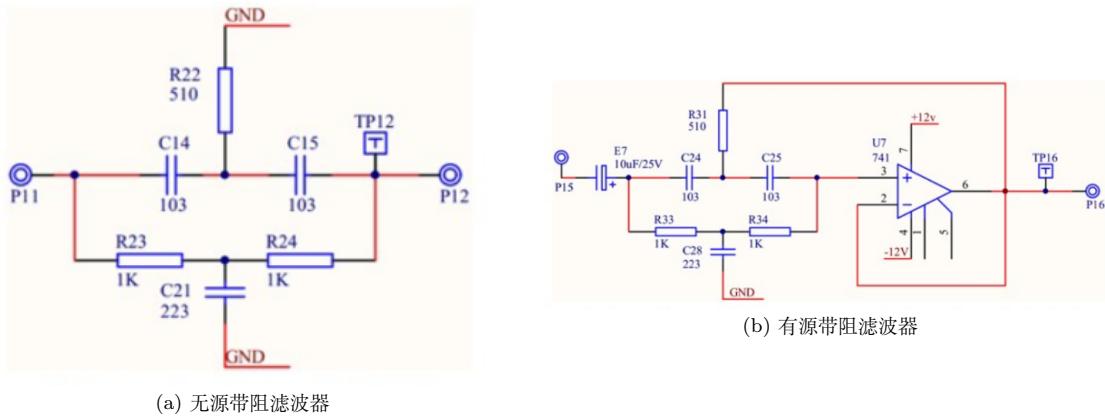


图 4.6. 带阻滤波器连接示意图

表 4.7. 带阻无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$											
$V_O(V)$											

扫频法测量

1. 将扫频范围设置为 100Hz~80kHz；
2. 把示波器连接到信号源输出 P2 处（示波器调为直流测试档），此时 P2 输出扫频信号；
3. 分别将扫频信号输入到无源带阻、有源带阻滤波器的输入端；
4. 模块开电，对比观察输入输出信号，拍照记录波形。

4.5 实验结果

任务一：低通滤波器频响特性测量结果

实验结果如下表4.9和表4.10所示。

任务二：高通滤波器频响特性测量结果

实验结果如下表4.11和表4.12所示。

任务三：带通滤波器频响特性测量结果

实验结果如下表4.13和表4.14所示。

任务四：带阻滤波器频响特性测量结果

实验结果如下表4.15和表4.16所示。

表 4.8. 带阻有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$											
$V_O(V)$											

表 4.9. 低通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	19.8k
$V_O(V)$	3.920	3.920	3.920	3.600	3.040	2.080	1.120	0.640	2.828

表 4.10. 低通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	17.1k
$V_O(V)$	4	4	4	3.920	2.960	1.120	0.480	0.240	2.828

表 4.11. 高通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	16k
$V_O(V)$	0.32	0.56	0.96	1.76	2.80	3.52	3.84	3.84	2.715

表 4.12. 高通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	13.7k
$V_O(V)$	0.24	0.56	0.96	1.76	2.80	3.52	3.84	2.24	2.715

表 4.13. 带通无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	1.21k, 24.6k
$V_O(V)$	2.08	2.8	3.2	3.2	2.72	1.92	1.2	0.72	2.624

表 4.14. 带通有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	2.31k, 24.5k
$V_O(V)$	1.28	2.08	2.96	3.2	2.8	1.92	1.12	0.64	2.624

表 4.15. 带阻无源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	4.31k, 65.3k
$V_O(V)$	3.76	3.44	2.72	1.52	0.32	1.44	2.56	3.44	2.658

表 4.16. 带阻有源滤波器逐点测量法

$V_I(V)$	4	4	4	4	4	4	4	4	截止频率 (f_L, f_H)
$f(\text{Hz})$	1k	2k	4k	8k	16k	32k	64k	128k	6.51k, 43.2k
$V_O(V)$	3.92	3.84	3.52	2.4	0.32	2.32	3.2	2.24	2.771

各个任务的扫频法测量结果

各个任务的扫频法测量结果如图4.15



图 4.7. 无源低通滤波器扫频法测量结果

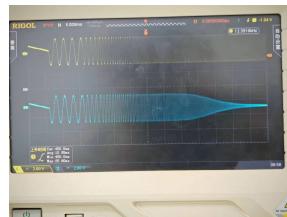


图 4.8. 有源低通滤波器扫频法测量结果

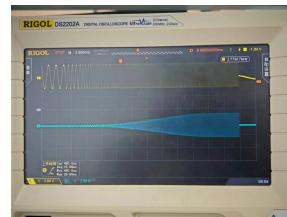


图 4.9. 无源高通滤波器扫频法测量结果



图 4.10. 有源高通滤波器扫频法测量结果

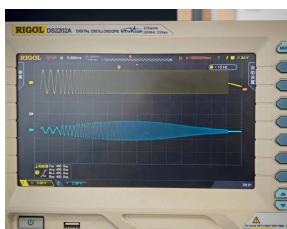


图 4.11. 无源带通滤波器扫频法测量结果



图 4.12. 有源带通滤波器扫频法测量结果



图 4.13. 无源带阻滤波器扫频法测量结果

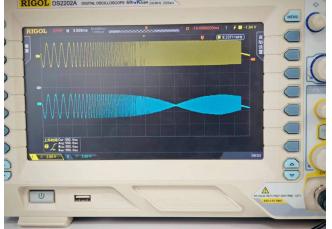


图 4.14. 有源带阻滤波器扫频法测量结果

图 4.15. 各任务扫频法测量结果

4.6 实验程序及运行结果

问题一程序及运行结果

Listing 8: 信号波形生成与绘图代码

```

dt = 0.01;
t = -5:0.01:5;
ft = sin(2.*pi.*t)./(pi.*t);
ft(isnan(ft)) = 2; % 替换t=1处的NaN为极限值2

domega = 0.01;
omega = -10:domega:10;
Fft = dt*ft*exp(-1j*t'*omega);

figure;
subplot(2,1,1);
plot(t, ft);
title('Time Domain Signal f(t)');

subplot(2,1,2);
plot(omega, abs(Fft));
title('Frequency Domain Signal |F(\omega)|');

saveas(gcf, './p1_result.png');

```

运行结果如下图4.16所示。

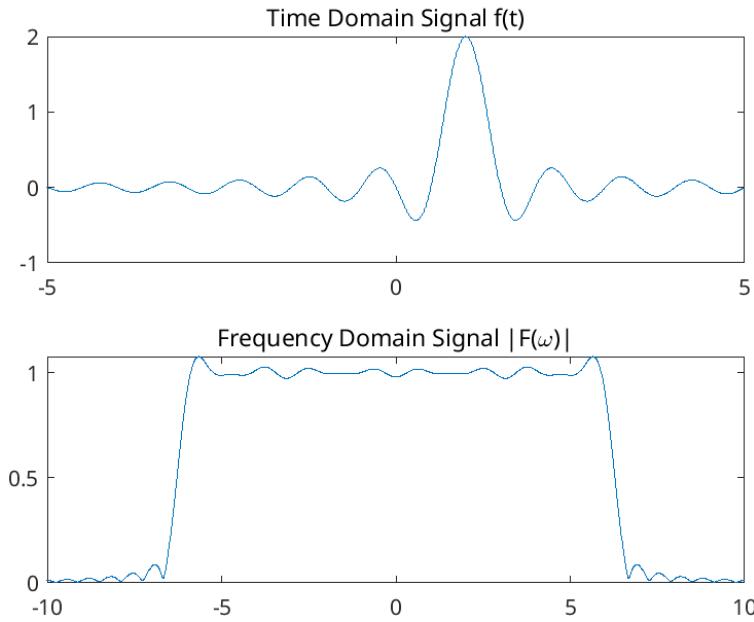


图 4.16. 问题一运行结果

问题二程序及运行结果

Listing 9: 信号波形生成与绘图代码

```
w = -10:0.01:10;
dw = 0.01;
Fw = -1j*2*w./(16+w.^2);

%inverse Fourier Transform
t = -5:0.01:5;
dt = 0.01;
ft = (1/(2*pi))*dw*Fw*exp(1j*w'*t);

figure;
subplot(2,1,1);
plot(w, abs(Fw));
title('Frequency Domain Signal |F(\omega)|');

subplot(2,1,2);
plot(t, real(ft));
title('Time Domain Signal f(t)');

saveas(gcf, './p2_result.png');
```

运行结果如下图4.17所示。

任务三程序及运行结果

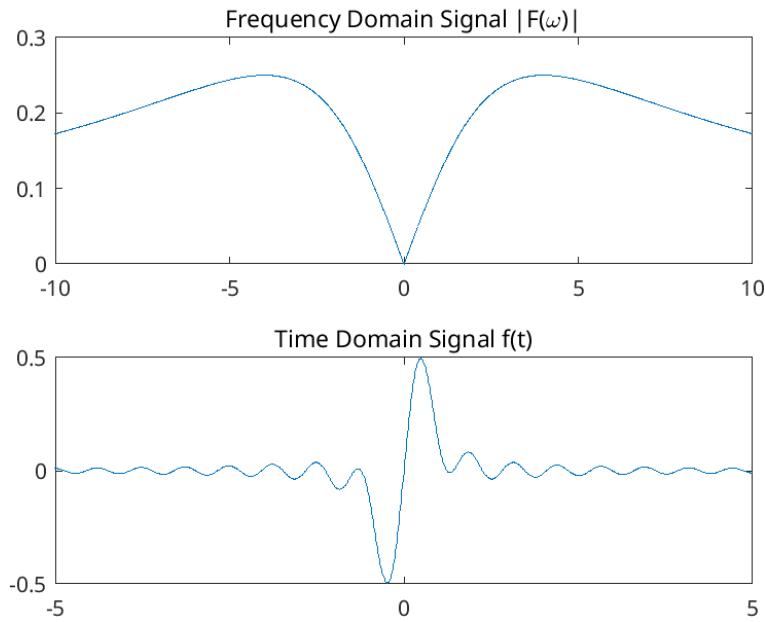


图 4.17. 问题二运行结果

Listing 10: 信号波形生成与绘图代码

```

t = -2:0.01:2;
ft = (t+2).*(t<-1) + 1*(t<=1&t>=-1) + (-(t-2)).*(t>1);

%Fourier Transform
dt = 0.01;
dw = 0.01;
w = -10:0.01:10;
Fw = dt*ft*exp(-1j*t'*w);

figure;
subplot(3,1,1);
plot(t, ft);
title('Time Domain Signal f(t)');
subplot(3,1,2);
plot(w, abs(Fw));
title('Frequency Domain Signal |F(\omega)|');

%Inverse Fourier Transform
ft_reconstructed = (1/(2*pi))*dw*Fw*exp(1j*w'*t);
subplot(3,1,3);
plot(t, real(ft_reconstructed));
title('Reconstructed Time Domain Signal f(t) from F(\omega)');
saveas(gcf, './p3_result.png');

```

运行结果如下图4.18所示。

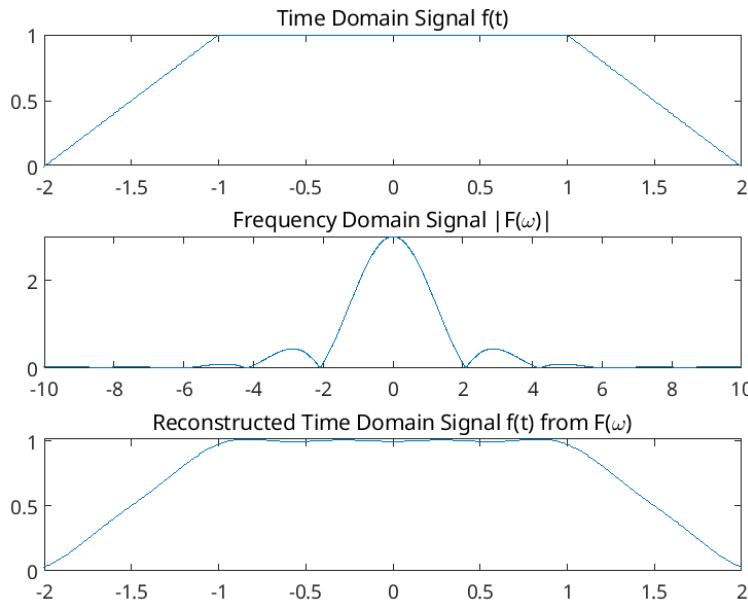


图 4.18. 任务三运行结果

任务四程序及运行结果

Listing 11: 信号波形生成与绘图代码

```
t = -5:0.01:5;
dt = 0.01;
ft = heaviside(t+0.5) - heaviside(t-0.5);
% f(t/2)
ft2 = heaviside(t/2+0.5) - heaviside(t/2-0.5);
% f(2t)
ft3 = heaviside(2*t+0.5) - heaviside(2*t-0.5);
w = -10:0.01:10;
dw = 0.01;
% Fourier Transform
% ft
Fft = dt*ft*exp(-1j*t'*w);
Fft2 = dt*ft2*exp(-1j*t'*w);
Fft3 = dt*ft3*exp(-1j*t'*w);

figure;
subplot(2,2,1);
plot(t,ft);
title('Time Domain Signal f(t)');

subplot(2,2,2);
plot(w, abs(Fft));
title('Frequency Domain Signal |F(\omega)|');

subplot(2,2,3);
plot(w, abs(Fft2));
```

```

title('Frequency Domain Signal |F_{f(t-2)}(\omega)|');

subplot(2,2,1);
plot(w, abs(Fft3));
title('Frequency Domain Signal |F_{f(2t)}(\omega)|');

saveas(gcf, './p4_result.png');

```

运行结果如下图4.19所示。

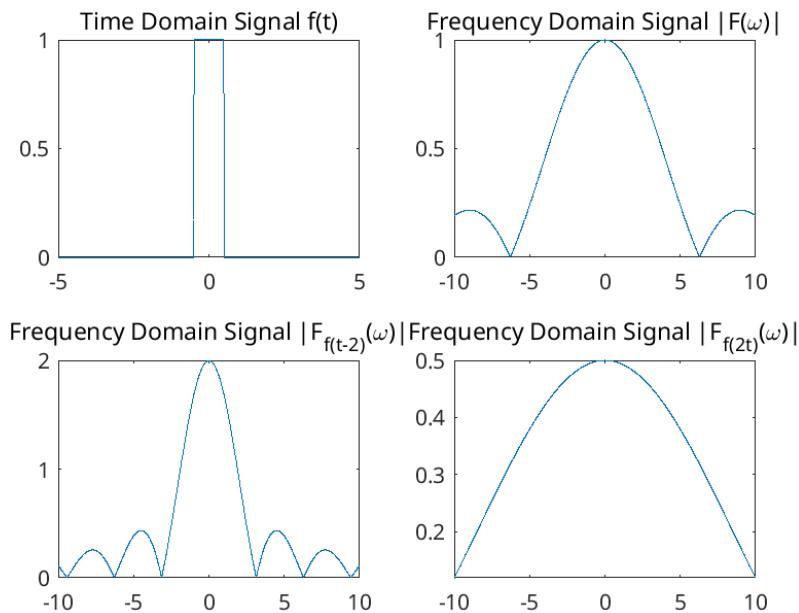


图 4.19. 任务四运行结果

实验五程序及运行结果

Listing 12: 信号波形生成与绘图代码

```

b = [0,1j,0];
a = [-1,1j,100];
w = -10:0.01:10;
res = freqs(b,a,w);

figure;
subplot(2,1,1);
plot(w, abs(res));
title('Frequency Domain Signal |F(\omega)|');

subplot(2,1,2);
plot(w, angle(res));
title('Frequency Domain Signal Phase \angle F(\omega)');

saveas(gcf, './p5_result.png');

```

运行结果如下图4.20所示。

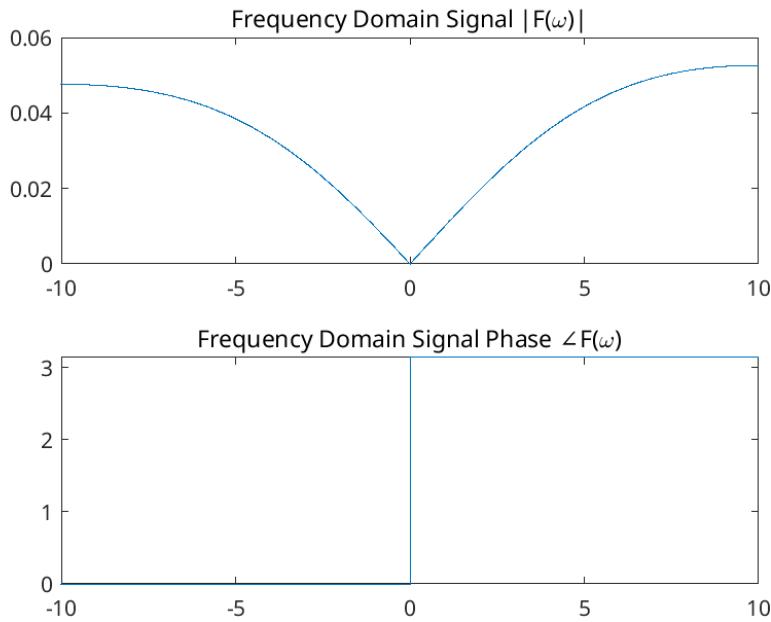


图 4.20. 任务五运行结果

实验六程序及运行结果

Listing 13: 信号波形生成与绘图代码

```
%0.2*dy/dt + y = x
%%系统传递函数H(jw) = Y(jw)/X(jw) = 1/(0.2jw+1)
x(t) = u(t) - u(t-1)
t = -5:0.01:5;
sys = tf([0,0,1],[0,0.2,1]);
xt = heaviside(t) - heaviside(t-1);

figure;
%频域分析 数值法
dt = 0.01;
w = -10:0.01:10;
dw = 0.01;
Xw = dt*xt*exp(-1j*t'*w);
Hw = freqs([0,0,1],[0,0.2,1],w);
Yw = Hw.*Xw;
yt = (1/(2*pi))*dw*Yw*exp(1j*w'*t);
%time domain output
subplot(2,1,1);
plot(t, abs(yt));
title('Time Domain Signal y(t) from Frequency Domain Analysis');

%时域分析
yt2 = lsim(sys, xt, t);
subplot(2,1,2);
plot(t, yt2);
title('Time Domain Signal y(t) from Time Domain Analysis');
```

```
saveas(gcf, './p6_result.png');
```

实验结果如下图4.21所示。

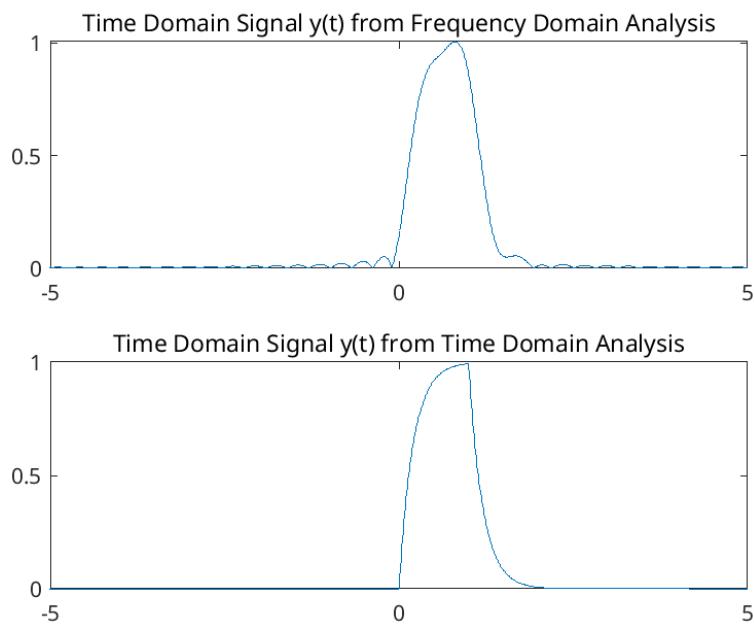


图 4.21. 任务六运行结果

5 抽样定理与信号恢复

5.1 实验目的

1. 观察抽样信号波形，了解同步与异步抽样的区别。
2. 验证抽样定理，掌握信号恢复的方法。

5.2 实验器材

1. 双踪示波器：1 台
2. 信号源及频率计模块 S2：1 块
3. 抽样定理及滤波器模块 S3：1 块

5.3 实验原理

抽样定理

抽样信号在条件 $f_s \geq 2B_f$ 时可以恢复，其中 f_s 为抽样频率， B_f 为原信号占有频带宽度。由于抽样信号频谱是原信号频谱的周期性延拓，只要通过截止频率为 f_c （满足 $f_m \leq f_c \leq f_s - f_m$ ， f_m 是原信号频谱中的最高频率）的低通滤波器就能恢复出原信号。若 $f_s < 2B_f$ ，则抽样信号频谱会出现混叠，将无法恢复。

在实际信号中，仅含有限频率成分的信号极少，大多信号的频率成分是无限的，且实际低通滤波器在截止频率附近的频率特性曲线不够理想（如图5.1所示）。若使 $f_s = 2B_f$ 、 $f_c = f_m = B_f$ ，恢复出的信号难免有失真。为减小失真，应将抽样频率 f_s 取高 ($f_s > 2B_f$)，低通滤波器需满足 $f_m < f_c < f_s - f_m$ 。

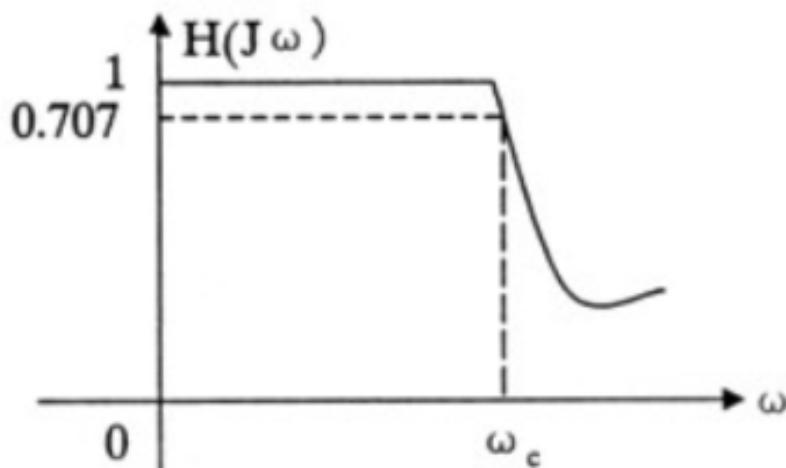


图 5.1. 实际低通滤波器在截止频率附近频率特性曲线

S3 上的自然抽样功能

抽样及恢复流程 信号自然抽样及恢复流程如图5.2所示。连续信号 $F(t)$ 经抽样器进行抽样处理，抽样输出的信号 $F_s(t)$ 再经低通滤波器，输出信号 $F'(t)$ 。当抽样时钟频率满足抽样定理时，就能从 $F_s(t)$ 中恢复出原始的连续信号 $F(t)$ 。

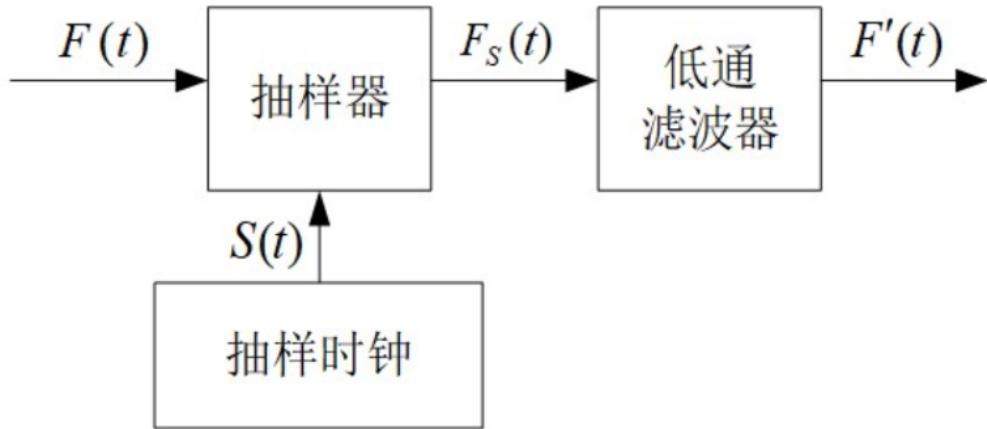


图 5.2. 信号自然抽样及恢复流程图

同步与异步抽样 S3 模块有同步和异步两种抽样方式：- 同步抽样：抽样时钟与连续信号由同一晶振源产生，便于观察到稳定的抽样信号，可对比信号抽样前后及恢复信号的波形。- 异步抽样：连续信号与抽样时钟是不同源的关系，贴近实际的信号抽样过程，抽样频率连续可调，但不利于用示波器观察到稳定的抽样信号。

有源低通滤波器 实验中，模块采用 8 阶有源低通滤波器对抽样后的信号进行滤波恢复，滤波器电路原理图如图5.3所示。异步抽样频率范围为 1.25kHz~35kHz。

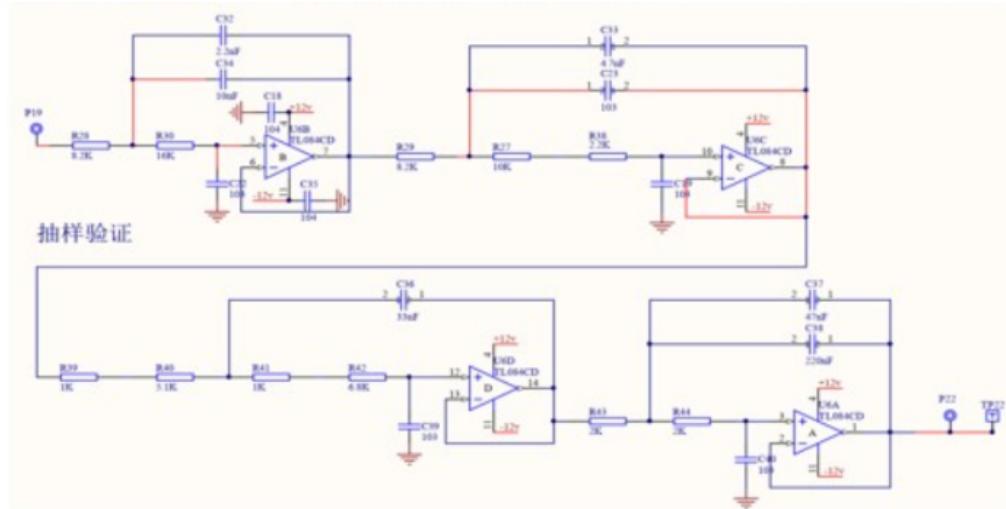


图 5.3. S3 模块有源低通滤波器电路原理图

5.4 实验步骤

任务：自然抽样及恢复

1、观察抽样信号波形 选用正弦波作为被抽样信号进行实验，步骤如下：

1. 将 S2 模块中的扫频开关 S3 置为 “OFF”，调节模拟信号源上的 “ROL1” 旋钮和 “模拟输出幅度调节” 旋钮，使 P2 处输出频率 1kHz、幅度 2V 的正弦波。
2. 连接模拟信号源输出端 P2 与抽样定理模块 S3 的连续信号输入点 P17。

3. 将开关 S2 拨至“异步”，用示波器观察 TP20 处抽样信号输出波形，调整电位器 W1 改变抽样频率，观察抽样信号的变化情况。
4. 将开关 S2 拨至“同步”，连接信号源及频率计模块 S2 中 P5 与抽样定理模块 S3 上外部开关输入点 P18。用示波器的两通道分别观察模拟信号输出端 P2、抽样信号输出波形 TP20，调整按钮 S7 改变抽样频率，观察抽样信号的变化情况，完成下表：

表 5.1. 抽样信号波形观察记录表

抽样频率	$F_s(t)$ 抽样信号 TP20 的波形
1kHz	
2kHz	
4kHz	
8kHz	

2、验证抽样定理与信号恢复

1. 连接模块 S3 的 P20 和 P19。
2. 用示波器接原始抽样信号输入点 TP17、恢复信号输出点 TP22。
3. 改变抽样时钟信号，对比观察信号恢复情况，完成下表：
4. 将开关 S2 拨至“异步”，用示波器观察 TP20 处抽样信号的波形，调整电位器 W1 改变抽样频率，观察抽样信号的变化情况。

表 5.2. 抽样定理验证与信号恢复记录表

输入信号频率	抽样频率	TP20 抽样信号输出	TP22 恢复信号输出
1kHz	1kHz		
1kHz	2kHz		
1kHz	4kHz		
1kHz	8kHz		

5.5 实验结果

1、抽样信号波形观察记录 抽样信号的波形如下图5.4所示：

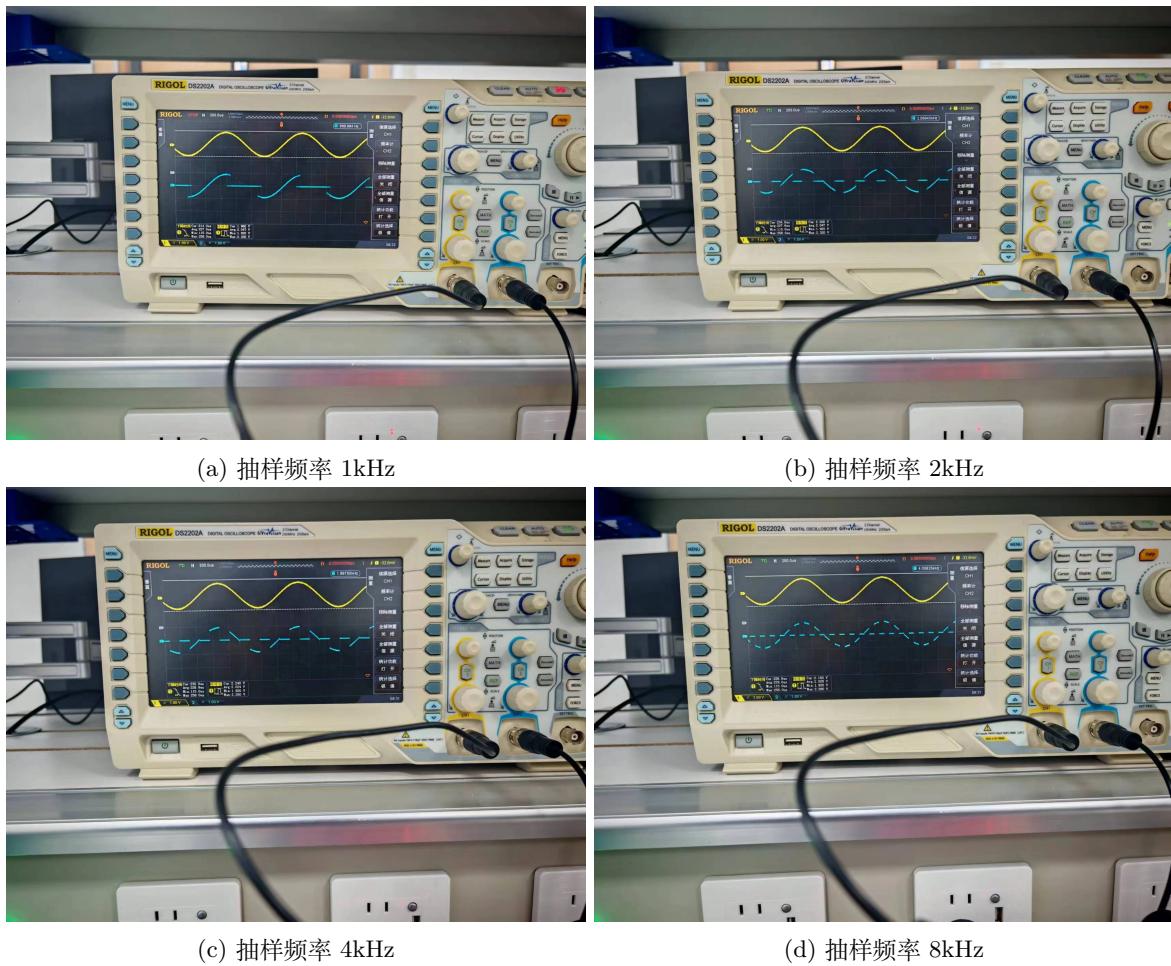


图 5.4. 抽样信号波形

2、抽样定理验证与信号恢复记录 抽样信号的恢复波形如下图5.5所示：

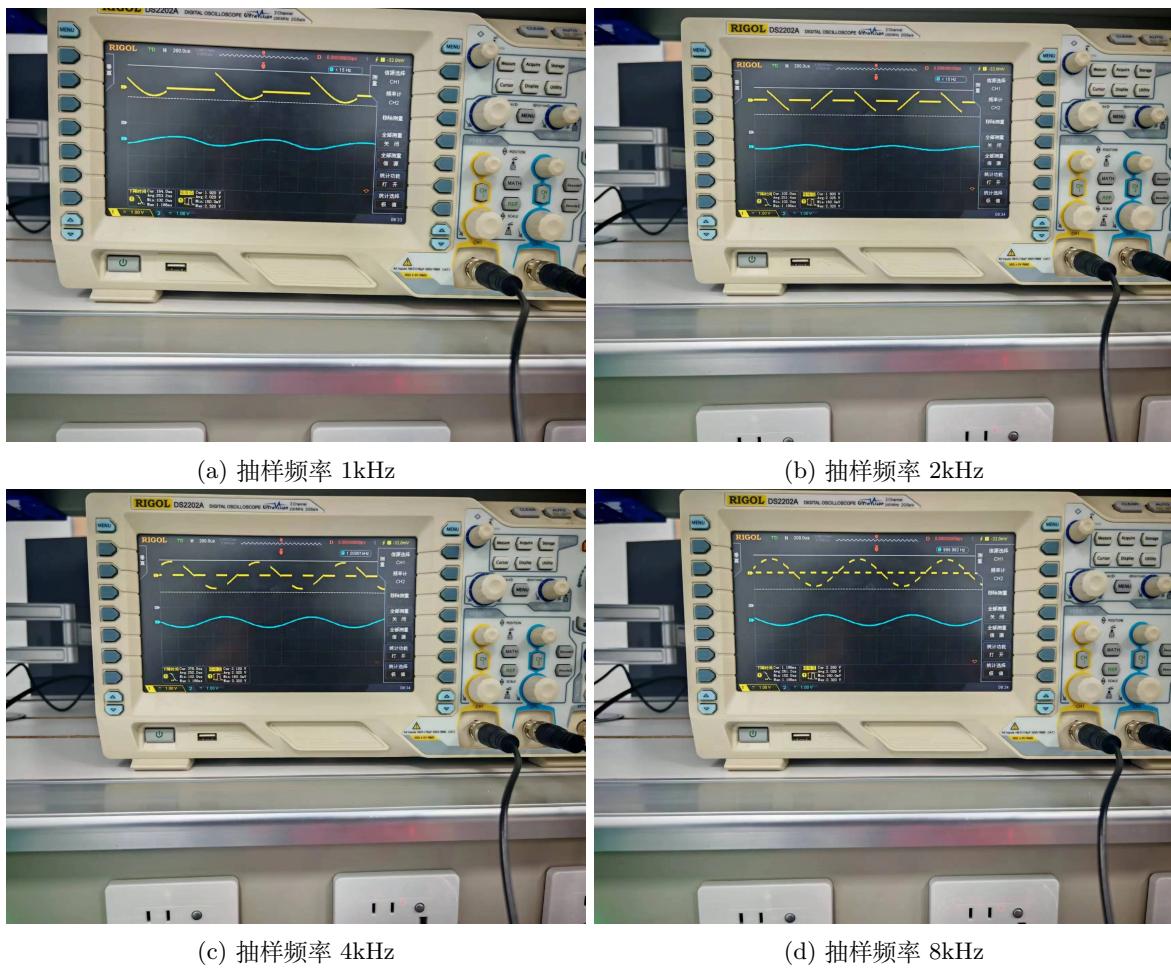


图 5.5. 抽样信号恢复波形

5.6 实验程序及运行结果

任务一程序及运行结果

Listing 14: 信号波形生成与绘图代码

```

Ts = 0.5; %采样周期
ts = -10:Ts:10;
t = -10:0.01:10;
%原始信号和采样信号

fs = sinc(ts).*(heaviside(ts+2*pi)-heaviside(ts-2*pi));
f = sinc(t).*(heaviside(t+2*pi)-heaviside(t-2*pi));
figure;
subplot(2,2,1);
plot(t,f);
title(' Original Signal f(t)');
subplot(2,2,2);
scatter(ts,fs);
title(' Sampled Signal fs(t)');

%信号恢复
wc = pi/Ts;

```

```

sinc_mat = sinc( wc/pi * (t'*ones(1,length(ts)) - ones(length(t),1)*ts) );
fr = sum( fs .* sinc_mat, 2 );
subplot(2,2,3);
plot(t,fr);
title(' Reconstructed Signal fr(t)' );

bias = fr - f';
subplot(2,2,4);
plot(t,bias);
title(' Reconstruction Error fr(t)-f(t)' );

%保存图片
handle = gcf;
saveas(handle,'./class5_1png');

```

运行结果如图5.6所示：

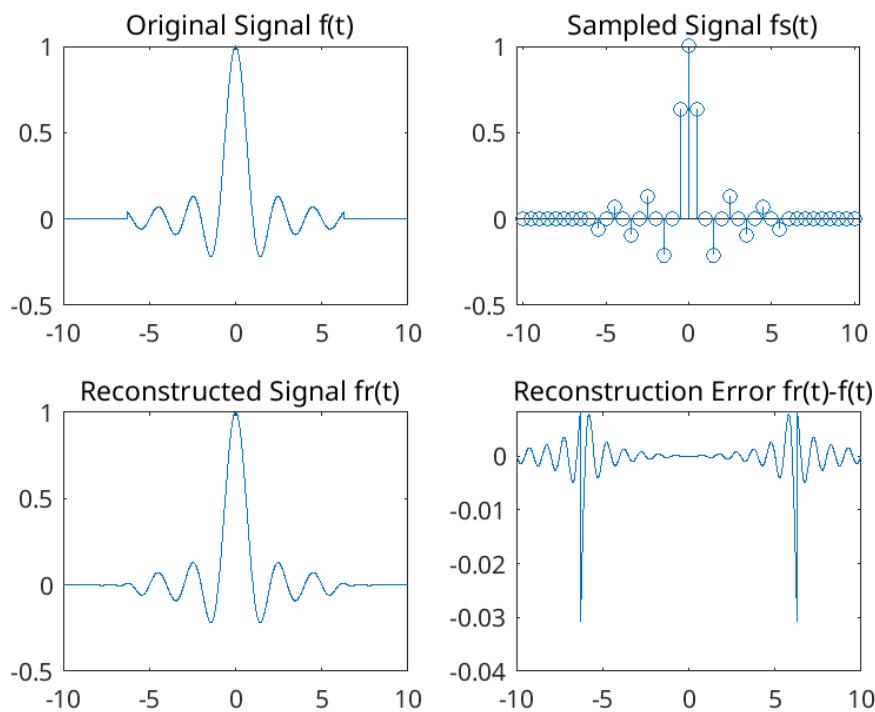


图 5.6. 任务一运行结果

Listing 15: 信号波形生成与绘图代码

任务二程序及运行结果

```

Ts = 0.3; %采样周期
ts = -4:Ts:4;
t = -4:0.01:4;
%原始信号和采样信号

```

```

fs = 1/2.* (1.+cos(ts)).*(heaviside(ts+2*pi)-heaviside(ts-2*pi));
f = 1/2.* (1.+cos(t)).*(heaviside(t+2*pi)-heaviside(t-2*pi));
figure;
subplot(2,2,1);
plot(t,f);
title(' Original Signal f(t)');
subplot(2,2,2);

stem(ts,fs);
title(' Sampled Signal fs(t)');

%信号恢复
wc = pi/Ts;
sinc_mat = sinc( wc/pi * (t'*ones(1,length(ts)) - ones(length(t),1)*ts) );
fr = sum( fs .* sinc_mat, 2 );
subplot(2,2,3);
plot(t,fr);
title(' Reconstructed Signal fr(t)');

bias = fr - f';
subplot(2,2,4);
plot(t,bias);
title(' Reconstruction Error fr(t)-f(t)');

```

%保存图片

```

handle = gcf;
saveas(handle,'class5_2.png');

```

运行结果如图5.7所示：

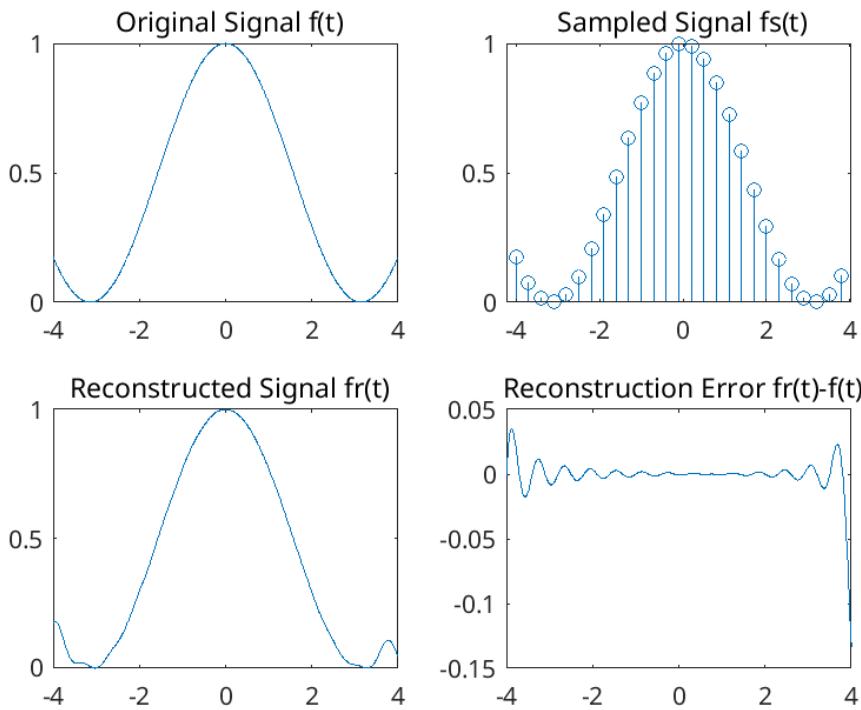


图 5.7. 任务二运行结果

Listing 16: 信号波形生成与绘图代码

任务三程序及运行结果

```

Ts = 0.1;
t = -3:0.01:3;
T = -3:Ts:3;
ft = 2.*((t<0&t>-1)|(t>1&t<2))+1*(t>=0&t<=1);

%采样信号
fTs = 2.*((T<0&T>-1)|(T>1&T<2))+1*(T>=0&T<=1);

%数值傅里叶变换
dw = 0.01;
w = -100:dw:100;
FTw = dw.*fTs*exp(-1j*T'*w);

%抽样信号的还原

%低通滤波器
Hw = heaviside(w+pi/Ts)-heaviside(w-pi/Ts);
Ftw = FTw.*Hw;

%逆傅里叶变换
f_recon = (1/(2*pi)).*Ftw*exp(1j*w'*t)*dw;

figure;

subplot(4,1,1);

```

```

plot(t,ft);
title(' Original Signal f(t)');

subplot(4,1,2);
stem(T,fTs);
title(' Sampled Signal f_{Ts}(t)');

subplot(4,1,3);
plot(w,abs(FTw));
title('Magnitude Spectrum |F(w)|');

subplot(4,1,4);
plot(t,real(f_recon));
title(' Reconstructed Signal f_{recon}(t)');

saveas(gcf,'./class5_3.png');

```

运行结果如图5.8所示：

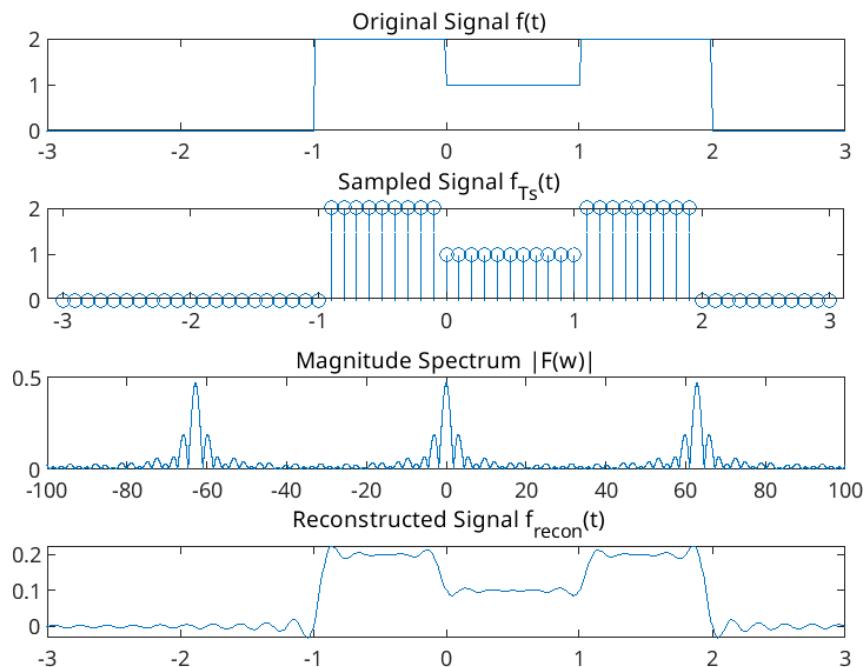


图 5.8. 任务三运行结果

6 信号卷积实验

6.1 实验目的

- 理解卷积的概念及物理意义。
- 通过实验的方法加深对卷积运算的图解方法及结果的理解。

6.2 实验器材

- 双踪示波器：1 台
- 信号源及频率计模块（S2 模块）：1 块
- 数字信号处理模块（S4 模块）：1 块

6.3 实验原理

两个矩形脉冲信号的卷积过程

两信号均为矩形脉冲信号，其卷积运算可通过图解法完成，最终结果需与实验结果进行对比。图 6-1 展示了两矩形脉冲的卷积积分运算过程与结果。

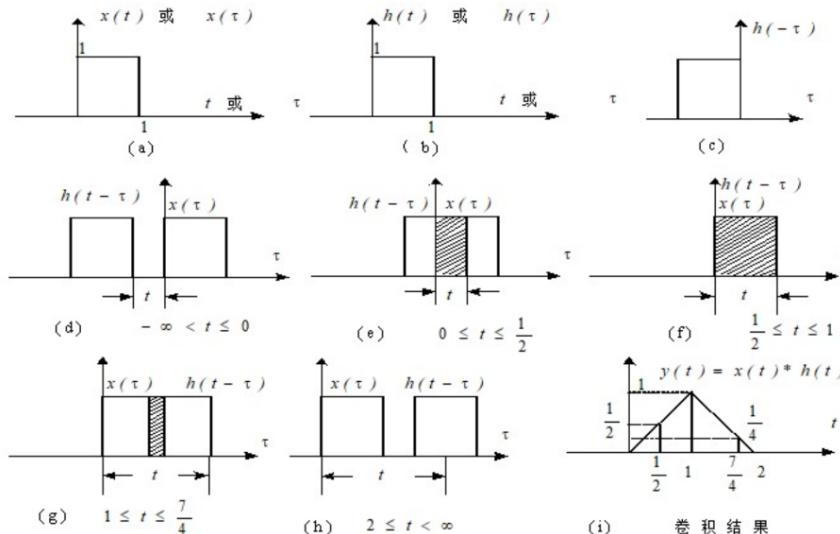


图 6.1. 两矩形脉冲的卷积积分的运算过程与结果

图解法的一般步骤 1. 置换：将变量 t 替换为 τ ，即 $f_1(t) \rightarrow f_1(\tau)$, $f_2(t) \rightarrow f_2(\tau)$ ；

- 反褶：将 τ 替换为 $-\tau$ ，即 $f_2(\tau) \rightarrow f_2(-\tau)$ ；
- 平移：将 $-\tau$ 替换为 $t - \tau$ ，即 $f_2(-\tau) \rightarrow f_2(t - \tau)$ ；
- 相乘：计算 $f_1(\tau) \cdot f_2(t - \tau)$ ；
- 积分：计算 $\int_{-\infty}^{+\infty} f_1(\tau) f_2(t - \tau) d\tau$ 。

不同占空比矩形波的自卷积过程 1. 占空比 50% 的矩形波自卷积过程：通过上述图解法步骤，最终得到的卷积结果为三角形脉冲信号；

- 占空比 25% 的矩形波自卷积过程：同样遵循图解法步骤，卷积结果的波形形态随占空比变化而改变。

矩形脉冲信号与锯齿波信号的卷积

信号 $f_1(t)$ 为锯齿波信号, $f_2(t)$ 为矩形脉冲信号, 如图 6-2 所示。根据卷积积分的运算方法, 可得到两者的卷积积分结果 $y(t)$, 具体波形如图 6-2(c) 所示。

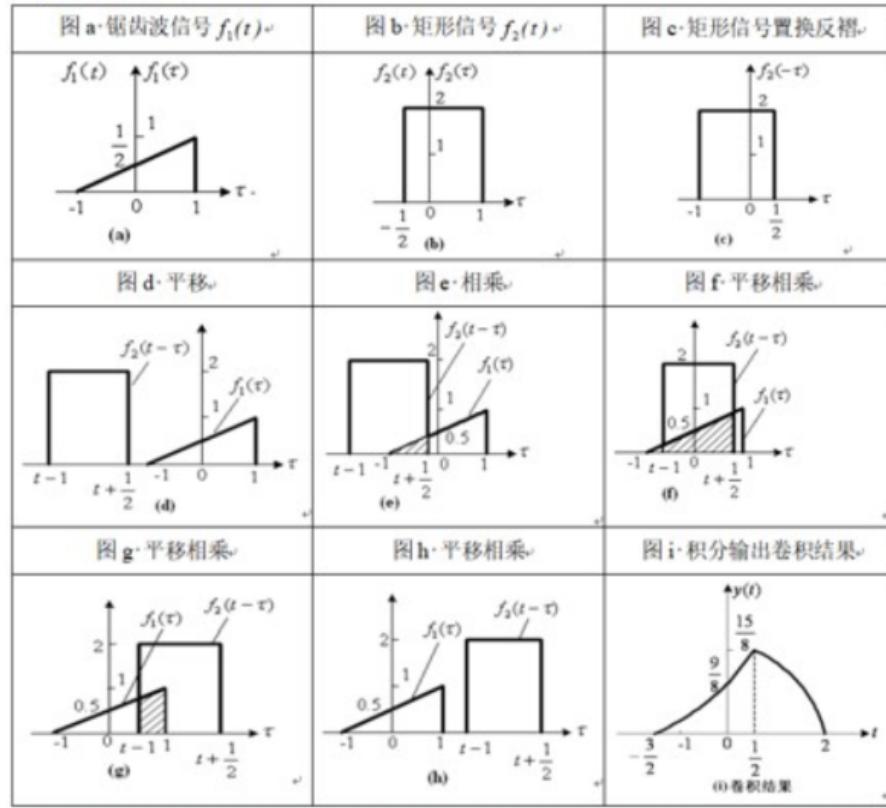


图 6.2. 矩形脉冲信号与锯齿波信号的卷积积分结果

占空比 50% 的矩形波与锯齿波的互卷积过程 按照卷积图解法的五个步骤依次运算, 最终得到的卷积结果为分段线性的连续信号, 其波形特征与矩形波和锯齿波的参数相关。

6.4 实验步骤

矩形脉冲信号的自卷积

1. 模块关电, 连接信号源及频率计模块 S2 的模拟输出 P2 和数字信号处理模块 S4 的 P9;
2. 模块开电, 调节信号源模块使 P2 输出方波信号: 扫频开关拨至 “OFF”, 调节 “ROL1” 使方波频率为 500Hz, 调节 “模拟输出幅度调节 W1” 使幅度为 1V; 然后长按 “ROL1” 旋钮约 2 秒钟, 旋转调节 “ROL1”, 使数码管上显示数据 “50” (即占空比为 50%);
3. 将拨动开关 SW1 调整为 “00000010”, 设置为自卷积功能;
4. 按下复位键 S2;
5. 将示波器的探头 CH1 接于 TP2, 探头 CH2 接于 TP1, 对比观察占空比为 50% 的输入信号与卷积后输出信号波形, 照片记录在表6.1中;
6. 改变矩形波占空比: 长按信号源模块的 “频率调节” 旋钮切换到方波占空比设置功能, 旋转 “频率调节” 旋钮, 将 P2 输出矩形波的占空比调整至 25% 和 75%; 用示波器探头 1 观测信号源模块的 P2, 确认占空比变化; 用示波器探头 2 观测数字信号处理模块的 TP1, 记录卷积信号输出的变化情况, 照片填入表6.1;
7. 改变矩形波的幅度: 调节信号源模块的 “模拟输出幅度调节 W1” 使 P9 输入幅度为 2V; 用示波器分别观测信号源模块的 P2 和数字信号处理模块的 TP1, 记录自卷积输出变化情况, 照片填入表6.1。

表 6.1. 矩形脉冲信号自卷积实验数据记录

信号条件	TP2 波形	TP1 波形
占空比 50%，幅度 1V		
占空比 25%，幅度 1V		
占空比 75%，幅度 1V		
占空比 50%，幅度 2V		

矩形信号与矩形信号的互卷积

- 激励信号为幅度 1V、频率 500Hz、占空比约为 50% 的方波信号（由信号源模块提供并输入到 S4 模块的 P9）；将 S4 模块的 S3 开关第 8 位拨为 1，此时 S4 模块内部产生频率 500Hz、占空比 50% 的方波信号（测试点为 TP2），卷积输出测试点为 TP1。具体步骤如下：
1. 模块关电，连接信号源及频率计模块 S2 的模拟输出 P2 和数字信号处理模块 S4 的 P9；
 2. 模块开电，调节信号源模块使 P2 输出方波信号：扫频开关拨至“OFF”，调节“ROL1”使方波频率为 500Hz，调节“模拟输出幅度调节 W1”使幅度为 1V；长按“ROL1”旋钮约 2 秒钟，旋转调节“ROL1”，使数码管显示“50”（占空比 50%）；
 3. 将拨动开关 SW1 调整为“00000011”（互卷积功能），将拨码开关 S3 拨为“00000001”（TP2 输出矩形波信号）；
 4. 按下复位键 S2；
 5. 将示波器探头 CH1 接于 TP2，探头 CH2 接于 TP1，对比观察两占空比 50% 矩形信号的卷积输出波形，照片记录在表6.2中；
 6. 改变矩形波占空比：长按信号源模块的“频率调节”旋钮切换到占空比设置功能，旋转旋钮将 P2 输出矩形波的占空比调整至 25% 和 75%；用示波器探头 1 观测 S2 模块的 P2，确认占空比变化；用探头 2 观测 S4 模块的 TP1，记录卷积输出变化情况，照片填入表6.2；
 7. 改变矩形波的幅度：调节信号源模块的“模拟输出幅度调节 W1”使 P9 输入幅度为 2V；用示波器分别观测 S2 模块的 P2 和 S4 模块的 TP1，记录互卷积输出变化情况，照片填入表6.2。

表 6.2. 矩形信号互卷积实验数据记录

信号条件	TP2 波形	TP1 波形
两信号占空比均为 50%，幅度 1V		
输入信号占空比 25%，内部信号占空比 50%，幅度 1V		
输入信号占空比 75%，内部信号占空比 50%，幅度 1V		
两信号占空比均为 50%，幅度 2V		

矩形信号与锯齿波信号的互卷积

- 激励信号为幅度 1V、频率 500Hz、占空比 50% 的方波信号（由 S2 模块提供并输入到 S4 模块的 P9）；将 S4 模块的 S3 开关第 8 位拨为 0，此时 S4 模块内部产生频率 500Hz、占空比 50% 的锯齿波信号（测试点为 TP2），卷积输出测试点为 TP1。具体步骤如下：
1. 模块关电，连接信号源及频率计模块 S2 的 P2 与数字信号处理模块 S4 的 P9；
 2. 模块开电，调节信号源上相应旋钮，使 P2 输出幅度 1V、频率 500Hz、占空比 50% 的矩形波；
 3. 将 S4 模块的拨动开关 SW1 调整为“00000011”（互卷积功能），将拨码开关 S3 拨为“00000000”（TP2 输出锯齿波信号），按下复位键 S2；
 4. 用示波器探头连接 S4 模块的 TP2，观测锯齿波波形；
 5. 用示波器探头连接 S4 模块的 TP1，观测卷积后输出信号的波形，照片记录在表6.3中；
 6. 改变矩形波占空比：长按信号源模块的“频率调节”旋钮切换到占空比设置功能，旋转旋钮将 P2 输出

矩形波的占空比调整至 25% 和 75%；观测 P9 输出矩形波的占空比变化，记录卷积输出变化情况，照片填入表6.3；

7. 改变锯齿波占空比：改变拨码开关 S3 的前 7 位，设置锯齿波输出为不同占空比信号；用示波器探头 1 观测 S4 模块的 TP2，确认锯齿波占空比变化；用探头 2 观测 S4 模块的 TP1，记录不同占空比下锯齿波与矩形波的互卷积输出变化情况，照片填入表6.3。

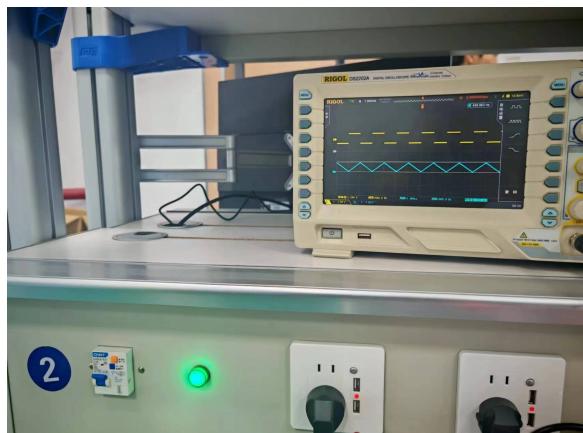
表 6.3. 矩形信号与锯齿波互卷积实验数据记录

信号条件	TP2 波形 (S4 模块)	TP1 波形
矩形波、锯齿波占空比均为 50%，幅度 1V		
矩形波占空比 25%，锯齿波占空比 50%，幅度 1V		
矩形波占空比 75%，锯齿波占空比 50%，幅度 1V		
矩形波占空比 50%，锯齿波占空比 25%，幅度 1V		

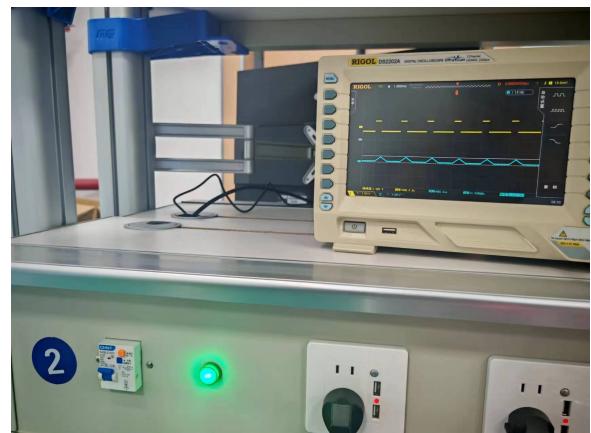
6.5 实验结果

矩形脉冲信号的自卷积

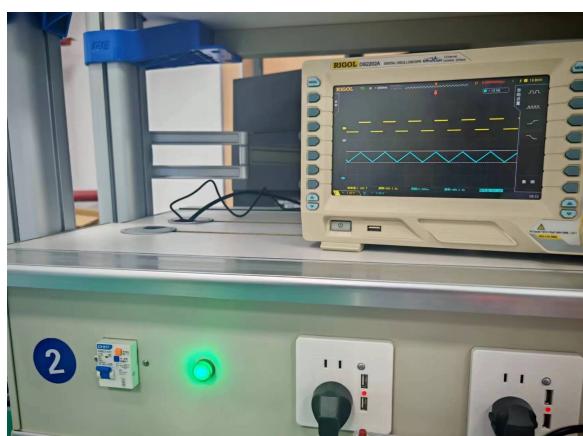
自卷积信号的波形如下图6.3所示：



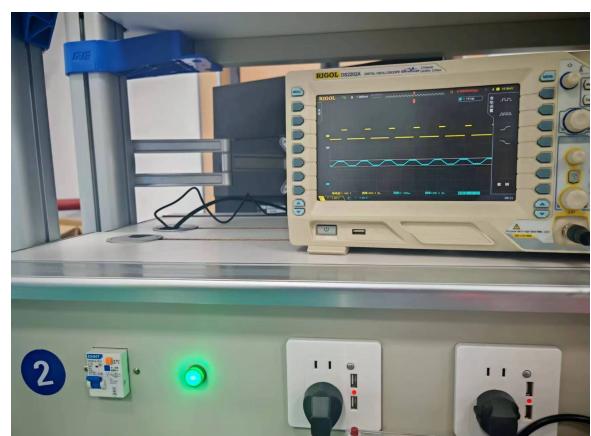
(a) 占空比 50%，幅度 1V



(b) 占空比 25%，幅度 1V



(c) 占空比 75%，幅度 1V

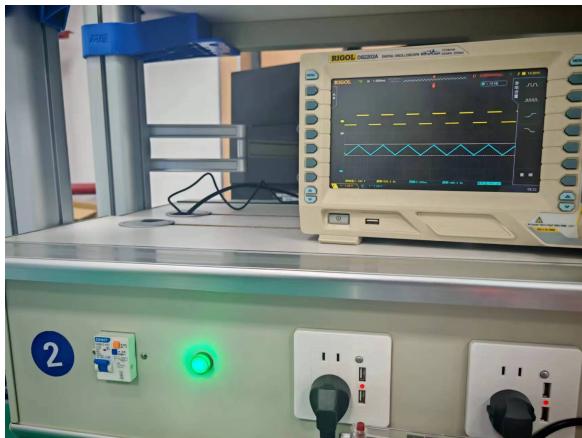


(d) 占空比 50%，幅度 2V

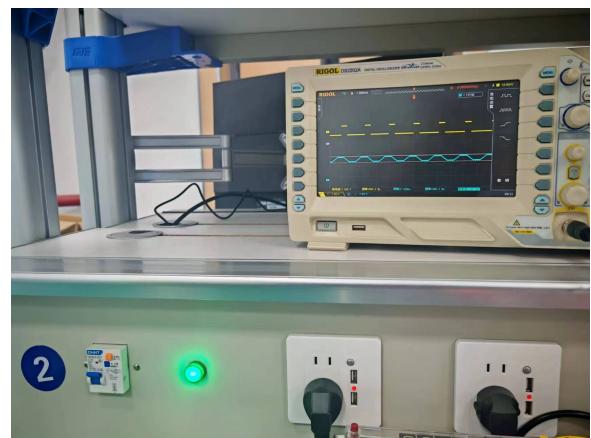
图 6.3. 矩形脉冲信号自卷积波形

矩形信号与矩形信号的互卷积

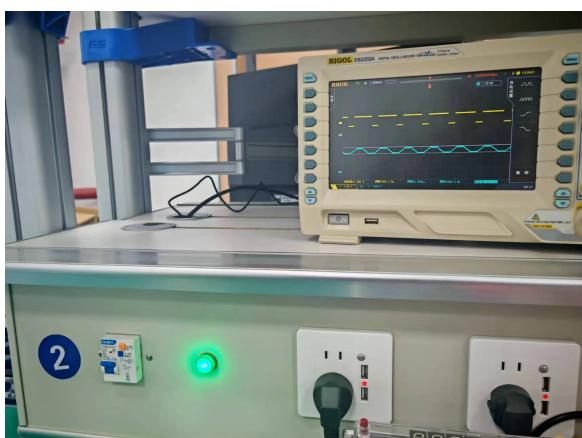
互卷积信号的波形如下图6.4所示：



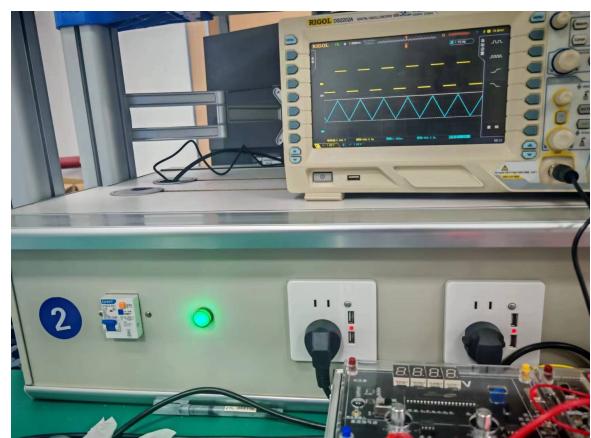
(a) 两信号占空比均为 50%，幅度 1V



(b) 输入信号占空比 25%，内部信号占空比 50%，幅度 1V



(c) 输入信号占空比 75%，内部信号占空比 50%，幅度 1V

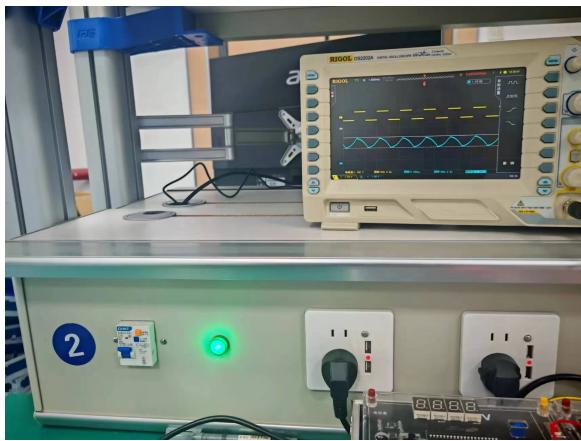


(d) 两信号占空比均为 50%，幅度 2V

图 6.4. 矩形信号互卷积波形

矩形信号与锯齿波信号的互卷积

互卷积信号的波形如下图6.5所示：



(a) 矩形波、锯齿波占空比均为 50%，幅度 1V



(b) 矩形波占空比 25%，锯齿波占空比 50%，幅度 1V



(c) 矩形波占空比 75%，锯齿波占空比 50%，幅度 1V



(d) 矩形波占空比 50%，锯齿波占空比 25%，幅度 1V

图 6.5. 矩形信号与锯齿波互卷积波形

6.6 实验程序及运行结果

任务一程序及运行结果

Listing 17: 信号波形生成与绘图代码

```

delta = 0.01;
t = -10:delta:10;
f1 = heaviside(t) - heaviside(t-2);
f2 = heaviside(t+3) - heaviside(t);

c12 = conv(f1, f2) * delta;
c11 = conv(f1, f1) * delta;
c22 = conv(f2, f2) * delta;
ctime = -20:delta:20;

figure;
subplot(2,3,1);
plot(t, f1);
title('f1(t)');

subplot(2,3,2);

```

```

plot(t, f2);
title('f2(t)');

subplot(2,3,4);
plot(ctime, c11);
title('f1(t) * f1(t)');

subplot(2,3,5);
plot(ctime, c12);
title('f1(t) * f2(t)');

subplot(2,3,6);
plot(ctime, c22);
title('f2(t) * f2(t)');

saveas(gcf, './p1.png');

```

运行结果如图6.6所示：

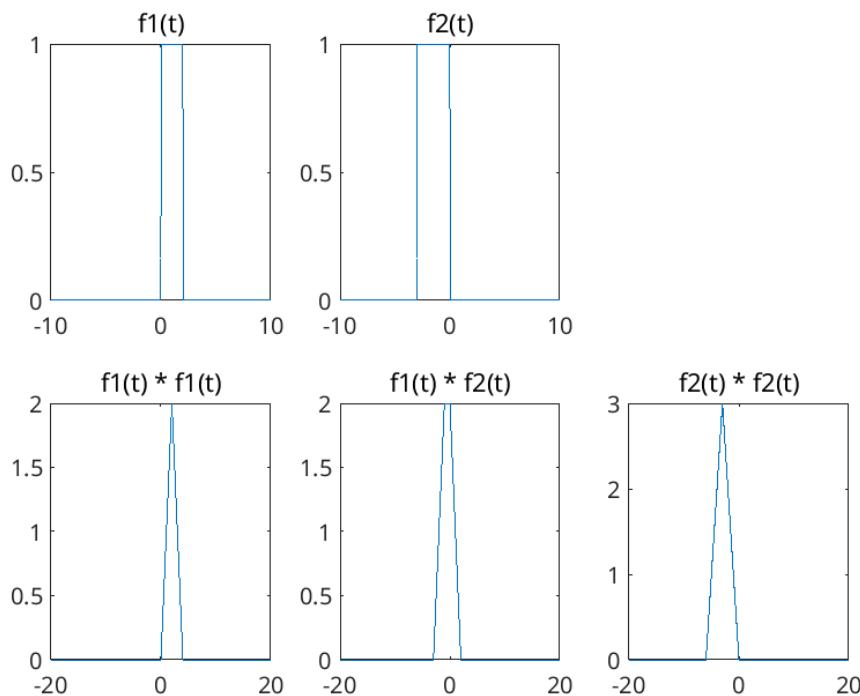


图 6.6. 任务一运行结果

任务二程序及运行结果

Listing 18: 信号波形生成与绘图代码

```

delta = 0.01;
t = -5:delta:5;
f1 = heaviside(t) - heaviside(t-2);
f2 = heaviside(t) - heaviside(t-3) + heaviside(t-1) - heaviside(t-2);

c = conv(f1, f2) * delta;

```

```

ctime = -10:delta:10;

figure;
subplot(3,1,1);
plot(t, f1);
title('f1(t)');

subplot(3,1,2);
plot(t, f2);
title('f2(t)');

subplot(3,1,3);
plot(ctime, c);
title('f1(t) * f2(t)');

saveas(gcf, './p2.png');

```

运行结果如图6.7所示：

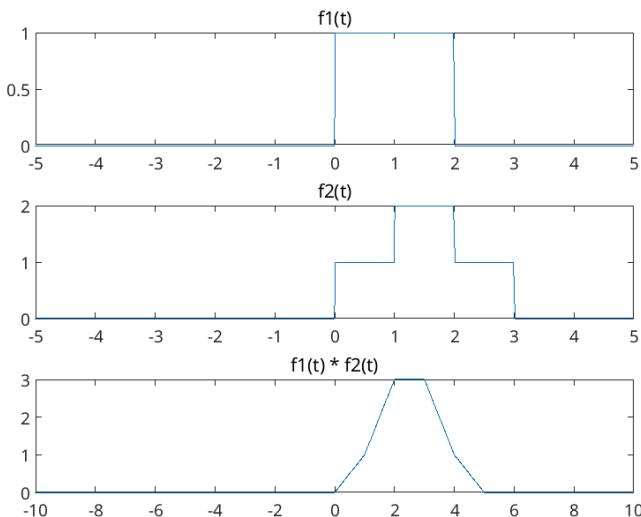


图 6.7. 任务二运行结果

任务三程序及运行结果

Listing 19: 信号波形生成与绘图代码

```

delta = 0.01;
t = -5:delta:5;
f1 = heaviside(t+1) - heaviside(t-1);
f2 = (t + 3).*(t>-3&t<-2) + 1.* (t>=-2&t<=0) + (1-t).* (t>0&t<1);

c = conv(f1, f2) * delta;
ctime = -10:delta:10;

figure;
subplot(3,1,1);
plot(t, f1);

```

```

title('f1(t)');

subplot(3,1,2);
plot(t, f2);
title('f2(t)');
title('f2(t)');

subplot(3,1,3);
plot(ctime, c);
title('f1(t) * f2(t)');

saveas(gcf, './p3.png');

```

运行结果如图6.8所示：

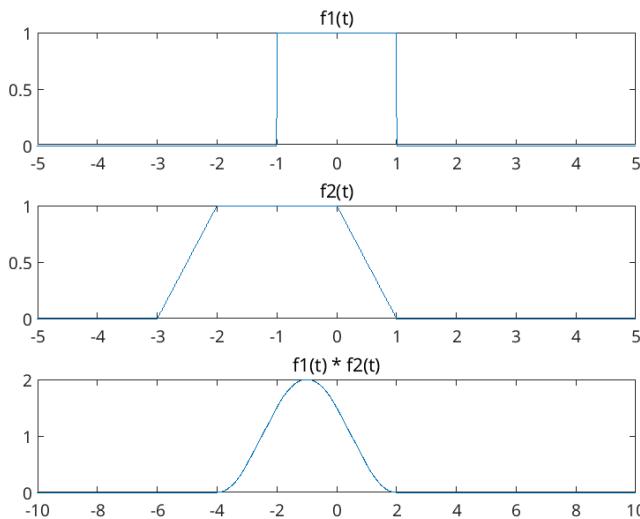


图 6.8. 任务三运行结果

任务四程序及运行结果

Listing 20: 信号波形生成与绘图代码

```

delta = 0.01;
t = -5:delta:5;
f1 = (t-1).*(heaviside(t-1)-heaviside(t-3));
f2 = heaviside(t+2) - 2*heaviside(t-2);

c = conv(f1, f2) * delta;
ctime = -10:delta:10;

figure;
subplot(3,1,1);
plot(t, f1);
title('f1(t)');

subplot(3,1,2);
plot(t, f2);

```

```
title('f2(t)');

subplot(3,1,3);
plot(ctime, c);
title('f1(t) * f2(t)');

saveas(gcf, './p4.png');
```

运行结果如图6.9所示：

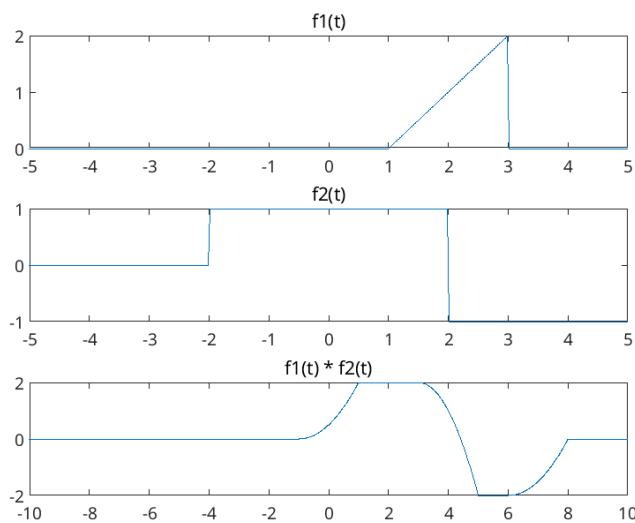


图 6.9. 任务四运行结果

7 波形的分解与合成实验

7.1 实验目的

1. 了解和熟悉波形分解与合成原理；2. 了解和掌握用傅里叶级数进行谐波分析的方法。

7.2 实验器材

- 双踪示波器：1 台
- 数字万用表：1 台
- 信号源及频率计模块（S2 模块）：1 块
- 数字信号处理模块（S4 模块）：1 块

7.3 实验原理

信号的频谱与测量

信号的时域特性和频域特性是对信号的两种不同描述方式。对于满足狄利克莱（Dirichlet）条件的周期信号 $f(t)$ ，可展开为三角形式的傅里叶级数，在区间 $[-T/2, T/2]$ 内表示为：

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\Omega t + b_n \sin n\Omega t)$$

其中 a_0 为直流分量， a_n 、 b_n 为各次谐波的系数， $\Omega = 2\pi/T$ 为基波角频率。

信号的时域与频域特性的内在联系如图 7-1 所示：图 (a) 为幅度-时间-频率三维图形，图 (b) 为时域波形图，图 (c) 为振幅频谱图。周期信号的振幅频谱具有离散性、谐波性、收敛性三大性质。

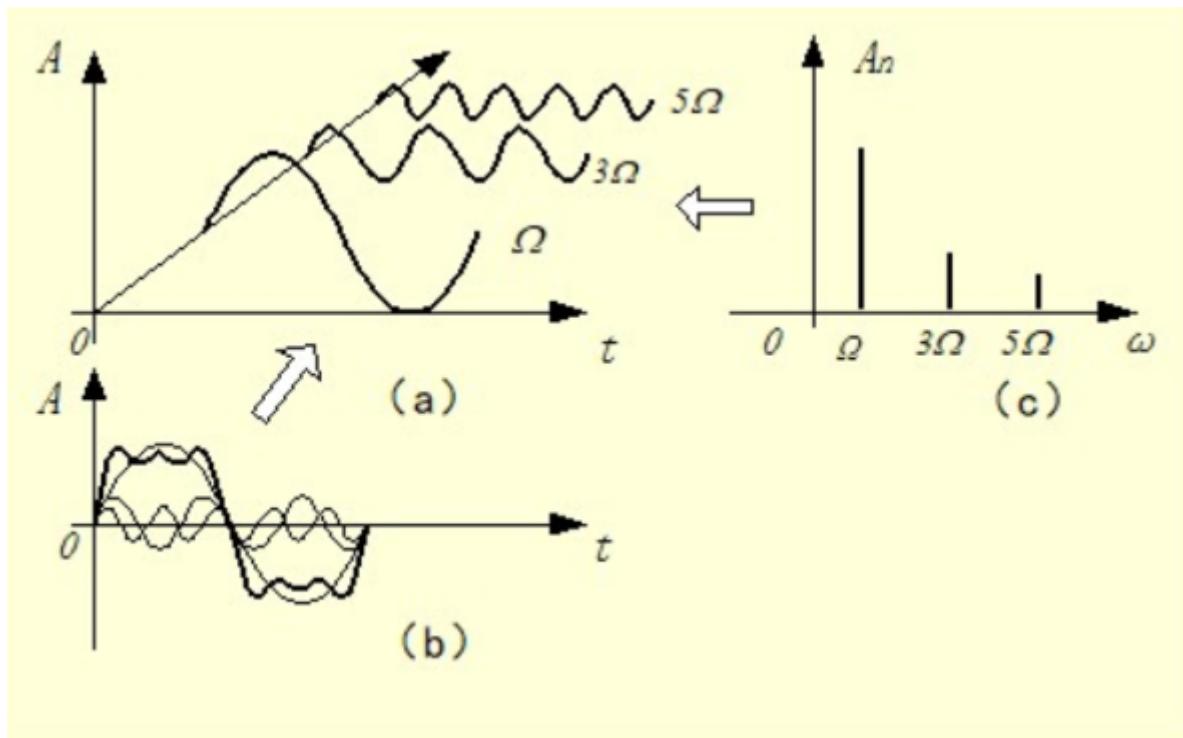


图 7.1. 信号的时域特性和频域特性

测量方法分为同时分析法和顺序分析法，本实验采用同时分析法，其原理是利用多个滤波器，将中心频率分别调至被测信号的各次谐波频率，可在信号发生的时间内同时测得所有频率分量，原理框图如图 7-2 所示。

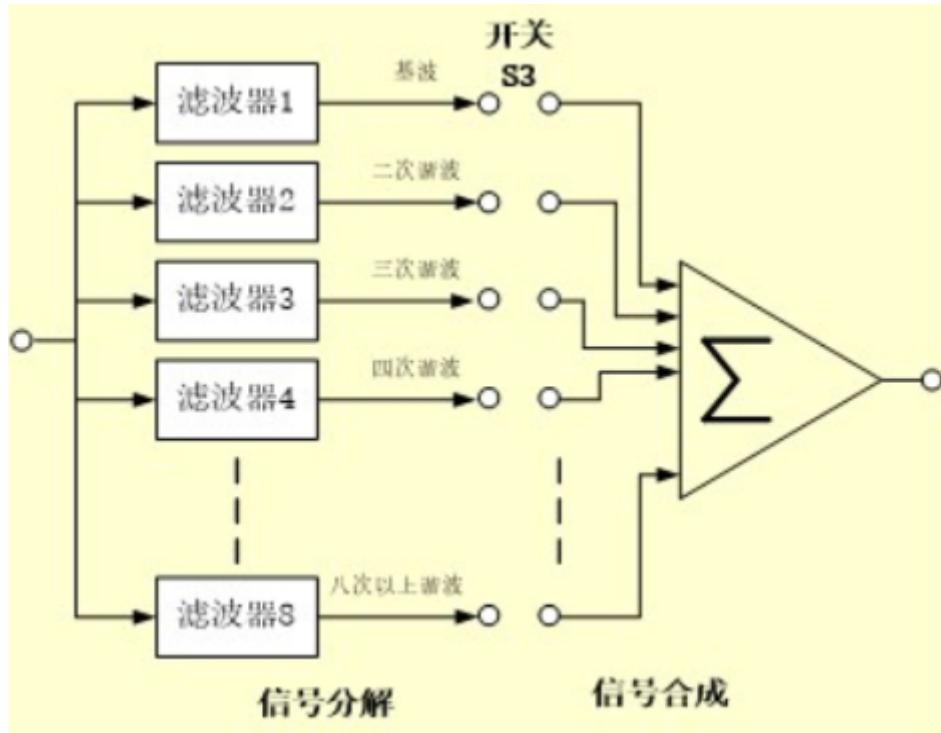


图 7.2. 用同时分析法进行频谱分析的原理框图

方波信号的分解（占空比 50%）

占空比 50% 的方波信号在一个周期内的解析式为：

$$f(t) = \begin{cases} A & 0 < t \leq \frac{T}{2} \\ -A & \frac{T}{2} < t \leq T \end{cases}$$

其傅里叶级数系数推导如下：

$$\begin{aligned} B_{k\omega} &= \frac{4}{T} \int_0^{T/2} A \sin k\omega t dt \\ &= \frac{4A}{T k \omega} [-\cos k\omega t]_0^{T/2} \\ &= \frac{4A}{k \pi} \quad (k = 1, 3, 5, 7, \dots) \end{aligned}$$

因此，傅里叶级数展开式为：

$$f(t) = \frac{4A}{\pi} \left(\sin \omega t + \frac{1}{3} \sin 3\omega t + \frac{1}{5} \sin 5\omega t + \frac{1}{7} \sin 7\omega t + \dots \right)$$

可见该信号仅含 1、3、5、7……等奇次谐波分量，偶次谐波分量为 0。当信号峰峰值为 2V（即 $A = 1$ ）时，各次谐波分量的峰值理论值如下表所示：

表 7.1. 占空比 50% 方波信号各次谐波峰值理论值 (峰峰值 2V)

谐波次数	峰值理论值 (V)
1 次 (基波)	1.2732395
3 次	0.4244131
5 次	0.2546479
7 次	0.1818914

矩形信号的分解 (占空比 40%)

矩形信号占空比为 40

傅里叶级数相关系数推导:

$$a_0 = \frac{1}{T} \int_{-\tau/2}^{\tau/2} A dt = \frac{A\tau}{T}$$

$$a_n = \frac{2}{T} \int_{-\tau/2}^{\tau/2} A \cos(n\omega t) dt = \frac{2A}{n\pi} \sin\left(\frac{n\pi\tau}{T}\right) = \frac{2A}{n\pi} \sin\left(\frac{2n\pi}{5}\right)$$

三角级数形式:

$$f(t) = \frac{2}{5}A + \frac{2AT}{5\pi} \sum_{n=1}^{\infty} Sa\left(\frac{n\omega T}{5}\right) \cos(n\omega t)$$

指数形式:

$$f(t) = \frac{2A}{5} \sum_{n=-\infty}^{\infty} Sa\left(\frac{n\omega T}{5}\right) e^{jn\omega t}$$

当信号峰峰值为 2V ($A = 1$) 时, 各次谐波分量的峰值理论值如下表所示:

表 7.2. 占空比 40% 矩形信号各次谐波峰值理论值 (峰峰值 2V)

谐波次数	峰值理论值 (V)
基波 (1 次)	1.2109
2 次	0.3742
3 次	0.2495 (相位与基波相反)
4 次	0.3027 (相位与基波相反)
5 次	0
6 次	0.2018
7 次	0.1069

三角波信号的分解

三角波信号在一个周期内的解析式为:

$$f(t) = \begin{cases} \frac{4A}{T}t & 0 \leq t \leq \frac{T}{4} \\ -\frac{4A}{T}t + 2A & \frac{T}{4} \leq t \leq \frac{T}{2} \end{cases}$$

利用积分公式 $\int x \sin ax dx = \frac{1}{a^2} \sin ax - \frac{1}{a}x \cos ax$, 推导谐波系数:

$$B_{k\omega} = \begin{cases} \frac{8A}{k^2\pi^2} & k = 1, 5, 9, \dots \\ -\frac{8A}{k^2\pi^2} & k = 3, 7, 11, \dots \end{cases}$$

傅里叶级数展开式为：

$$f(t) = \frac{8A}{\pi^2} \left(\sin \omega t - \frac{1}{3^2} \sin 3\omega t + \frac{1}{5^2} \sin 5\omega t - \frac{1}{7^2} \sin 7\omega t + \dots \right)$$

当 $A = 1$ 时，各次谐波分量的峰值理论值如下表所示：

表 7.3. 三角波信号各次谐波峰值理论值 ($A = 1$)

谐波次数	峰值理论值 (V)
1 次 (基波)	0.81056941
3 次	0.09006321
5 次	0.03242261
7 次	0.0165422

信号的分解提取与合成

信号分解提取 进行信号分解和提取是滤波系统的一项基本任务。当对信号的某些分量感兴趣时，可利用选频滤波器提取有用部分，滤除其他部分。数字信号处理模块 (S4 模块) 内置 8 个滤波器 (1 个低通、6 个带通、1 个高通)，可将复杂信号分解为各次谐波分量，分解后的 8 路信号分别从测试点 TP1 TP8 输出 (TP1 TP7 对应 1~7 次谐波，TP8 对应 8 次以上高次谐波)。

信号合成 矩形脉冲信号通过 8 路滤波器输出的各次谐波分量，DSP 将选中的谐波分量相加后从 TP8 输出。谐波的叠加组合通过 S4 模块的 8 位拨码开关 S3 控制 (闭合为参与合成)：第 1 位对应基波，第 2 位对应二次谐波，依此类推，8 位开关均闭合时各次谐波全部参与合成。分解前的原信号可通过 TP9 观测，此时合成波形应与原信号一致。

图 7-3 为 MATLAB 仿真的方波谐波合成结果，直观展示了不同谐波叠加次数对合成波形的影响：

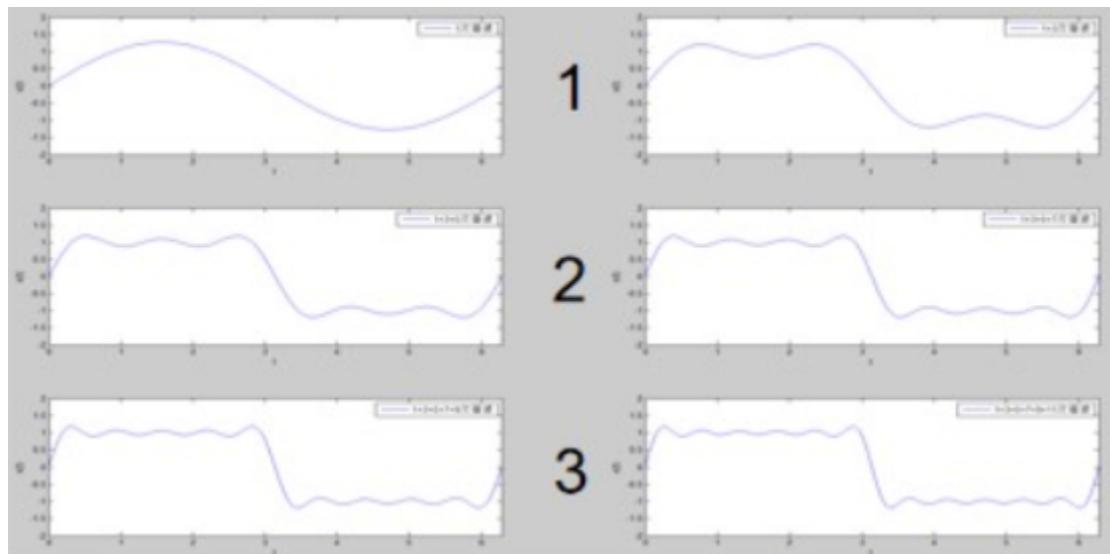


图 7.3. 方波 (500Hz、2V、50% 占空比) 谐波合成仿真结果

左 1 图：1 次谐波；右 1 图：1+3 次谐波；左 2 图：1+3+5 次谐波；右 2 图：1+3+5+7 次谐波；左 3 图：1+3+5+7+9 次谐波；右 3 图：1+3+5+7+9+11 次谐波

7.4 实验步骤

方波的分解

- 模块关电，连接信号源及频率计模块 S2 的信号输出端口 P2 与数字信号处理模块 S4 的 P9 端口；
- 模块开电，设置 S2 模块，使 P2 输出幅度 2V、频率 500Hz 的方波（占空比调为 50%）。将 S4 模块的拨动开关 SW1 调整为“00000101”，按下复位键 S2，选择矩形信号分解及合成功能；将拨码开关 S3 拨为“00000000”；
- 用示波器分别观测 S4 模块 TP1 TP7 输出的 1~7 次谐波波形及 TP8 输出的 8 次以上高次谐波波形，拍照记录；
- 保持信号幅度 2V、频率 500Hz 不变，将方波占空比调整为 40%。按表 7-4、表 7-5 要求，记录两种占空比下方波信号各次谐波的测量值（电压峰峰值）。

表 7.4. 占空比 50% ($\tau/T = 1/2$) 矩形脉冲信号的频谱 ($f = 500Hz, E = 2V$)

谐波频率	1f	2f	3f	4f	5f	6f	7f	8f 以上
理论值 (电压峰峰值, V)								
测量值 (电压峰峰值, V)								

表 7.5. 占空比 40% ($\tau/T = 2/5$) 矩形脉冲信号的频谱 ($f = 500Hz, E = 2V$)

谐波频率	1f	2f	3f	4f	5f	6f	7f	8f 以上
理论值 (电压峰峰值, V)								
测量值 (电压峰峰值, V)								

方波的合成

本任务仅做占空比 50%。保持 S4 模块 SW1 为“00000101”，将 S2 模块设置为输出幅度 2V、频率 400Hz、占空比 50%。用示波器观测 S4 模块 TP8（合成输出）和 S2 模块 P2（原信号），按表 7-6 要求设置拨码开关 S3，观察并记录各组合的合成波形；3. 每次切换 S3 状态后，对比合成波形与原信号的差异，重点观察谐波叠加次数对合成波形逼近原信号的影响。

表 7.6. 矩形脉冲信号的各次谐波合成要求

拨码开关 S3 状态	合成要求
10000000	基波单独合成
11000000	基波 + 二次谐波合成
10100000	基波 + 三次谐波合成
10010000	三次 + 五次谐波合成
10001000	基波 + 五次谐波合成
10101000	基波 + 三次 + 五次谐波合成
11111111	所有谐波合成
11011111	无三次谐波的其他谐波合成
11110111	无五次谐波的其他谐波合成
01111111	无八次以上高次谐波的其他谐波合成

三角波的分解与合成

- 模块关电，保持 P2 与 P9 的连接，S4 模块 SW1 仍设为“00000101”；
- 模块开电，设置 S2 模块，使 P2 输出幅度 2V、频率 400Hz 的三角波；
- 三角波分解：将 S3 拨为“00000000”，用示波器观测 TP1 TP7 的 1、3、5、7 次谐波波形，拍照记录，按表 7-7 记录各次谐波的测量值；
- 三角波合成：按表 7-8 要求设置拨码开关 S3，观测 TP8 的合成波形，拍照记录，对比不同合成组合与原三角波的差异。

表 7.7. 三角波信号的频谱 ($f = 400Hz$, $A = 1V$)

谐波频率	$1f$	$2f$	$3f$	$4f$	$5f$	$6f$	$7f$	$8f$ 以上
理论值 (电压峰峰值, V)								
测量值 (电压峰峰值, V)								

表 7.8. 三角波信号的各次谐波合成要求

拨码开关 S3 状态	合成要求
10000000	基波单独合成
10100000	基波 + 三次谐波合成
10001000	基波 + 五次谐波合成
10101000	基波 + 三次 + 五次谐波合成
10101010	基波 + 三次 + 五次 + 七次谐波合成
11111111	所有谐波合成
11011111	无三次谐波的其他谐波合成
11110111	无五次谐波的其他谐波合成
11111101	无七次谐波的其他谐波合成

7.5 实验结果

方波的分解

占空比 50% 方波分解结果 分解后各次谐波波形如图7.4所示：

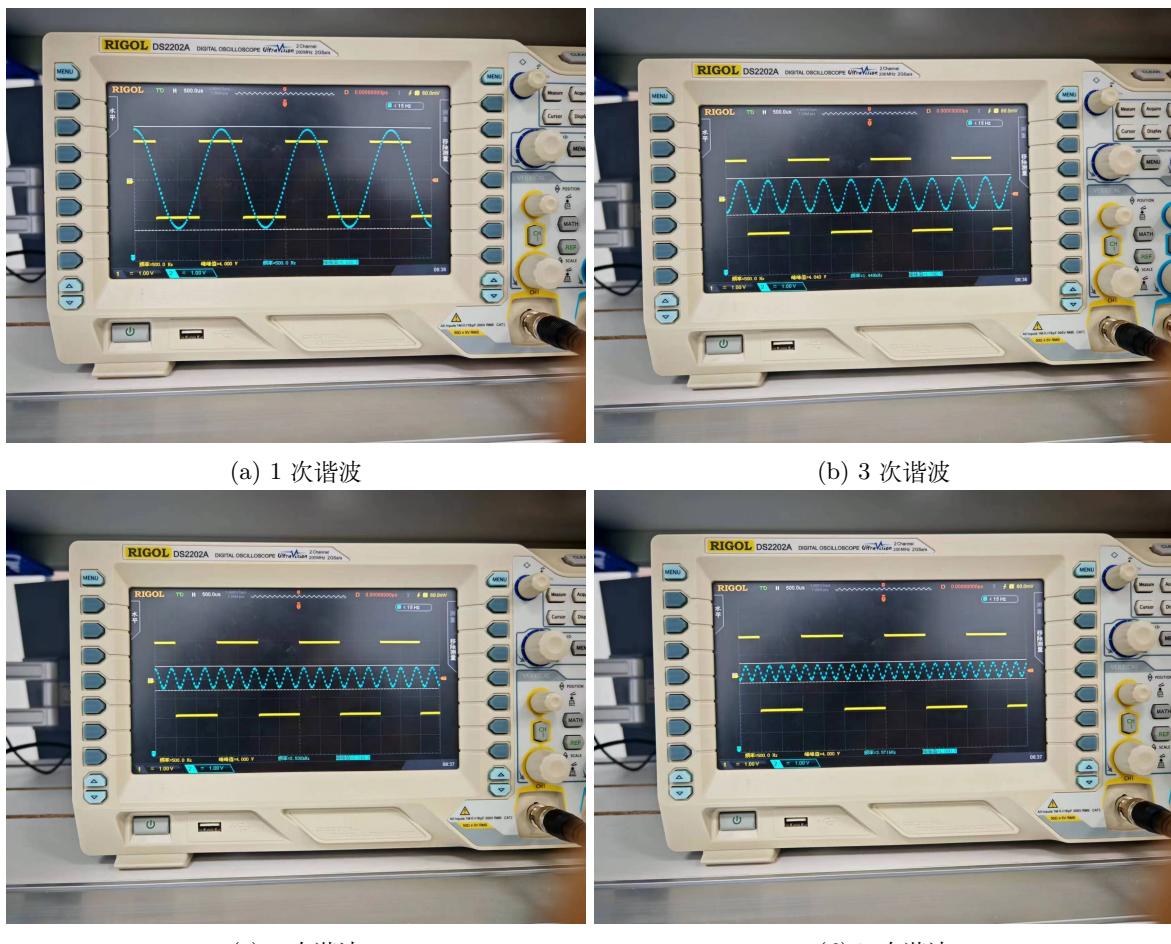


图 7.4. 占空比 50% 方波分解后各次谐波波形

各次谐波测量值记录在表7.9中。

表 7.9. 占空比 50% ($\tau/T = 1/2$) 矩形脉冲信号的频谱 ($f = 500Hz, E = 2V$)

谐波频率	$1f$	$2f$	$3f$	$4f$	$5f$	$6f$	$7f$	$8f$ 以上
理论值 (电压峰峰值, V)	2.546	0	0.849	0	0.509	0	0.364	-
测量值 (电压峰峰值, V)	2.060	0.320	0.940	0.140	0.620	0.140	0.500	-

占空比 40% 方波分解结果 分解后各次谐波波形如图7.5所示：

各次谐波测量值记录在表7.10中。

表 7.10. 占空比 40% ($\tau/T = 2/5$) 矩形脉冲信号的频谱 ($f = 500Hz, E = 2V$)

谐波频率	$1f$	$2f$	$3f$	$4f$	$5f$	$6f$	$7f$	$8f$ 以上
理论值 (电压峰峰值, V)	2.421	0.748	0.499	0.606	0.000	0.404	0.214	-
测量值 (电压峰峰值, V)	2.240	0.740	0.620	0.660	0.180	0.500	0.380	-

方波的合成

方波合成结果如图7.6所示：

三角波的分解与合成

三角波分解结果 三角波分解后各次谐波波形如图7.7所示：

各次谐波测量值记录在表7.11中。

表 7.11. 三角波信号的频谱 ($f = 400Hz, A = 2V$)

谐波频率	$1f$	$2f$	$3f$	$4f$	$5f$	$6f$	$7f$	$8f$ 以上
理论值 (电压峰峰值, V)	1.621	0	0.180	0	0.065	0	0.033	-
测量值 (电压峰峰值, V)	1.180		0.300		0.220		0.180	-

三角波合成结果 三角波合成结果如图7.8所示：

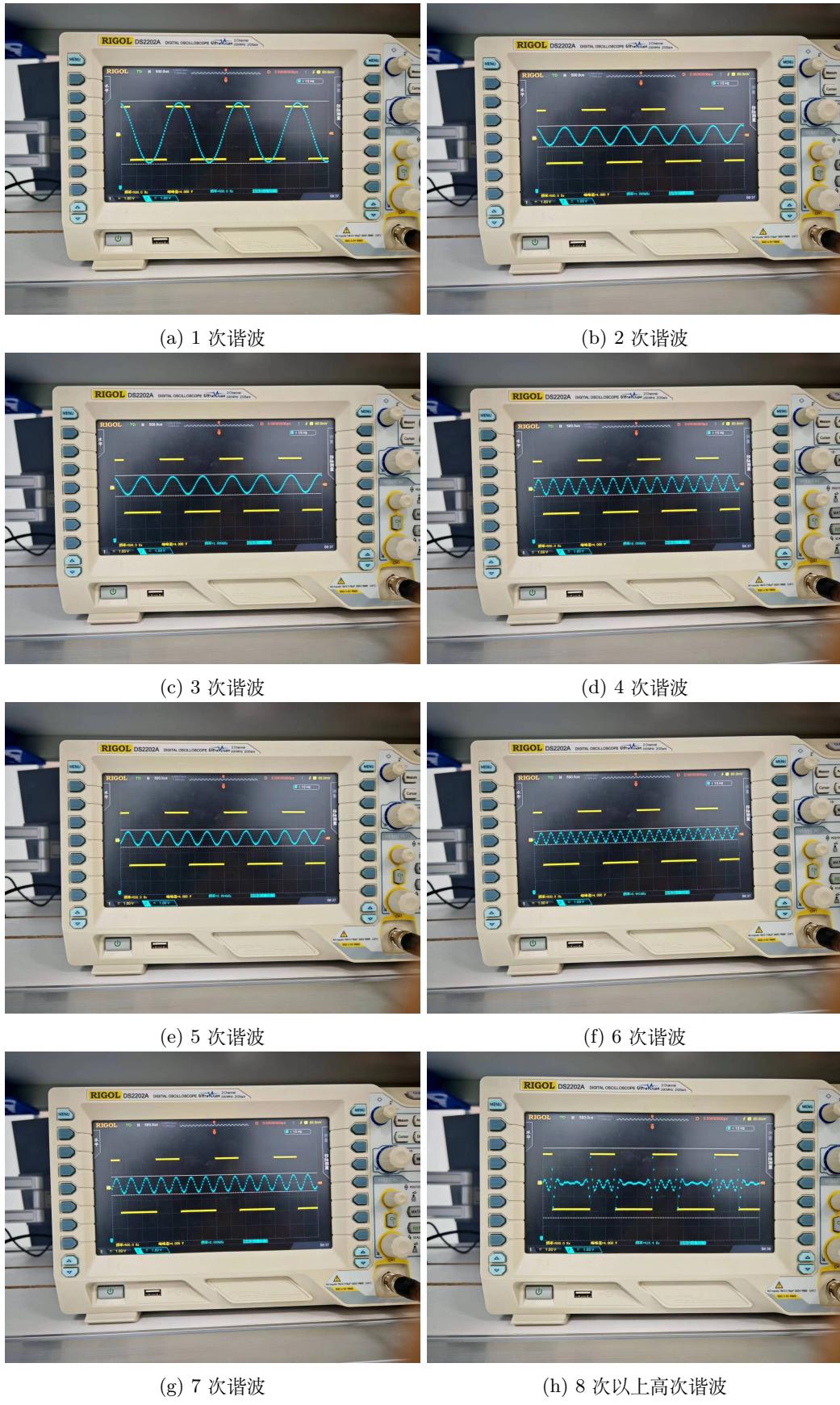


图 7.5. 占空比 40% 方波分解后各次谐波波形

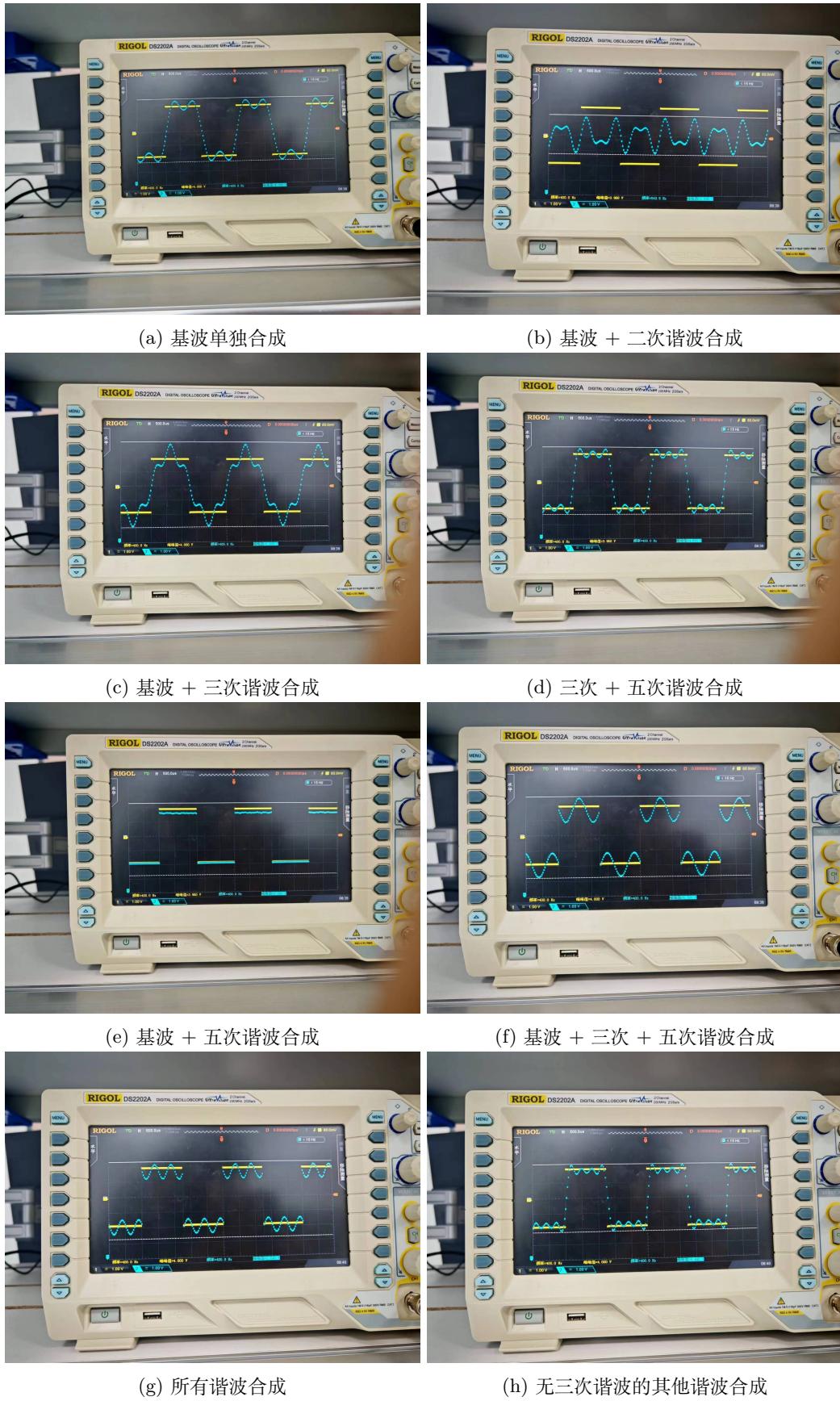


图 7.6. 方波各次谐波合成结果

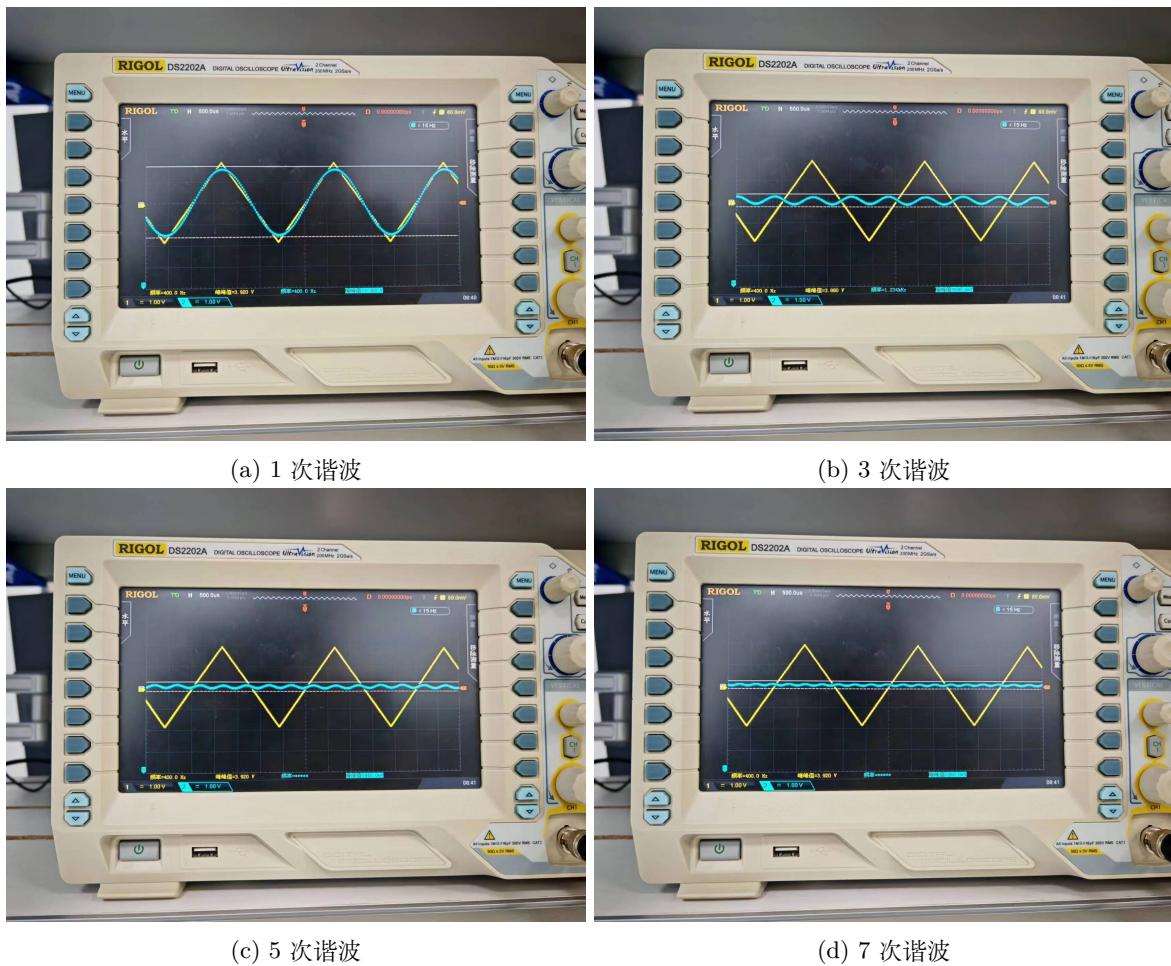


图 7.7. 三角波分解后各次谐波波形

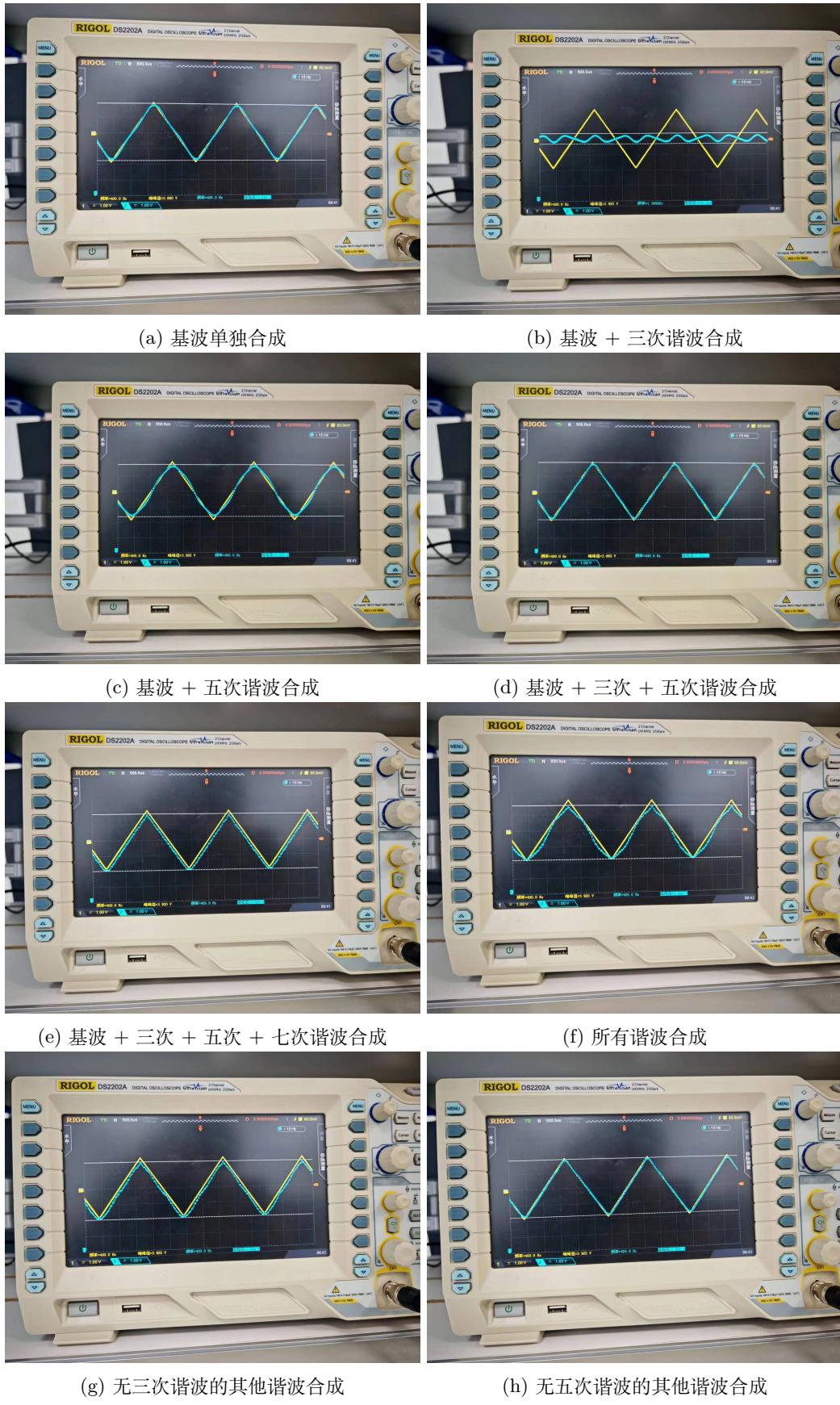


图 7.8. 三角波各次谐波合成结果

7.6 实验程序及运行结果

以下程序中，采样点数 N 均取 64，频谱图中，传统方法计算结果用红色曲线表示，FFT 计算结果用蓝色杆状图表示。

任务一

任务一代码如下所示：

Listing 21: 信号波形生成与绘图代码

```
t_length = 20;
N = 64;

T = t_length/N;
t = (0:N-1)*T - t_length/2;

x = sin(2 .* pi .* (t - 1))./(pi .* (t - 1));

w_length = 2*pi/T;
W = w_length/N;
w = (0:N-1)*W;
disp('Computing FFT...');

tic
X = T*fft(x,N);
X = fftshift(X);
toc

x_num = x;
w_num = linspace(-w_length/2, w_length/2, N);
disp('Computing Numerical FT...');

tic
W_num = x_num .* exp(-1j * w_num' * t);
toc

figure;
subplot(2,1,1);
stem(t,x);
title('Signal x(t)');

subplot(2,1,2);
stem(w-w_length/2,abs(X));
hold on;
plot(w_num, abs(sum(W_num,2))*T, 'r');
title('Magnitude Spectrum of x(t)');
hold off;

saveas(gcf, './p1.png');
```

运行结果如图7.9所示：

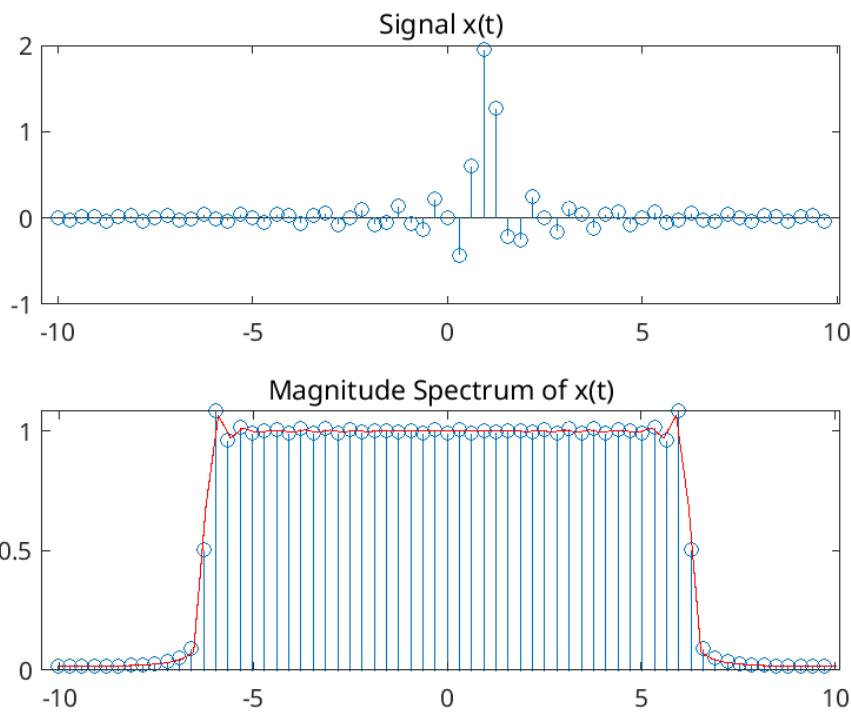


图 7.9. 任务一运行结果

任务二

任务二代码如下所示：

Listing 22: 信号波形生成与绘图代码

```
t_length = 20;
N = 64;

T = t_length/N;
t = (0:N-1)*T - t_length/2;

x = (t + 2).*(t>=-2&t<-1)+1.*((t>=-1&t<=1)+(2 - t).*(t>1&t<=2));

w_length = 2*pi/T;
W = w_length/N;
w = (0:N-1)*W;
disp('Computing FFT...');

tic
X = T*fft(x,N);
X = fftshift(X);
toc
time_of_fft = toc - tic;

x_num = x;
w_num = linspace(-w_length/2, w_length/2, N);
disp('Computing Numerical FT...');

tic
```

```

W_num = x_num .* exp(-1j * w_num' * t);
toc
toc_of_num = toc - tic;
figure;
subplot(2,1,1);
stem(t,x);
title('Signal x(t)');

subplot(2,1,2);
stem(w-w_length/2,abs(X));
hold on;
plot(w_num, abs(sum(W_num,2))*T, 'r');
title('Magnitude Spectrum of x(t)');
hold off;

saveas(gcf, './p2.png');

```

运行结果如图7.10所示：

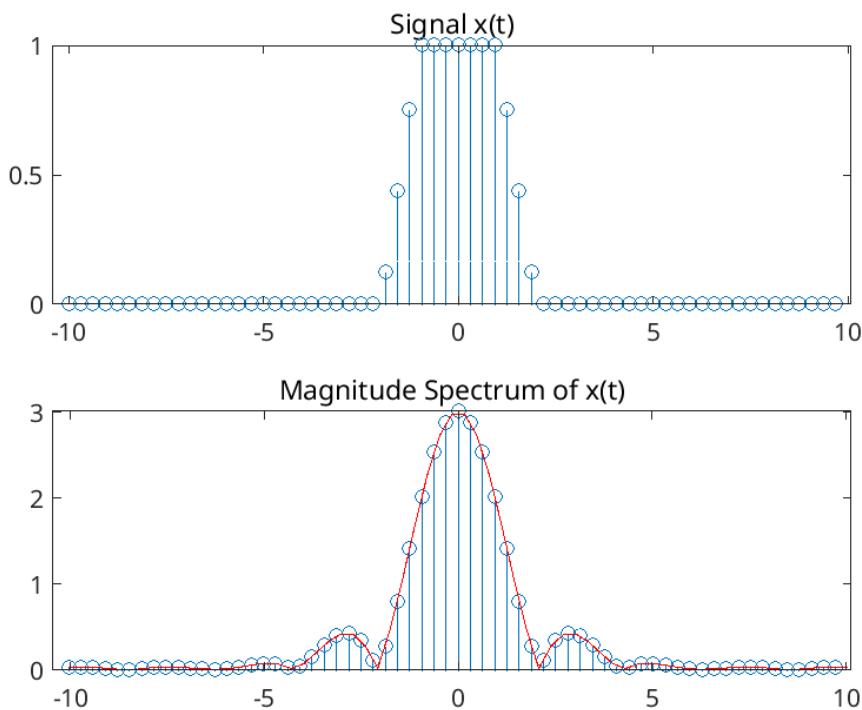


图 7.10. 任务二运行结果

任务三

任务三代码如下所示：

Listing 23: 信号波形生成与绘图代码

```

t_length = 10;
N = 64;

T = t_length/N;

```

```

t = (0:N-1)*T - t_length/2;

x = heaviside(t+0.5) - heaviside(t-0.5);
w_length = 2*pi/T;
W = w_length/N;
w = (0:N-1)*W;
disp('f(t) Computing FFT...');

tic
X = T*fft(x,N);
X = fftshift(X);
toc

x_num = x;
w_num = linspace(-w_length/2, w_length/2, N);
disp('Computing Numerical FT...');

tic
W_num = x_num .* exp(-1j * w_num' * t);
toc

figure;
subplot(3,2,1);
stem(t,x);
title('Signal x(t)');

subplot(3,2,2);
stem(w-w_length/2,abs(X));
hold on;
plot(w, abs(sum(W_num,2))*T, 'r');
title('Magnitude Spectrum of x(t)');
hold off;

x = heaviside(t/2+0.5) - heaviside(t/2-0.5);
w_length = 2*pi/T;
W = w_length/N;
w = (0:N-1)*W;
disp('f(t/2) Computing FFT...');

tic
X = T*fft(x,N);
X = fftshift(X);
toc

x_num = x;
w_num = linspace(-w_length/2, w_length/2, N);
disp('Computing Numerical FT...');

tic
W_num = x_num .* exp(-1j * w_num' * t);
toc

subplot(3,2,3);
stem(t,x);
title('Signal x(t)');

```

```

subplot(3,2,4);
stem(w-w_length/2,abs(X));
hold on;
plot(w_num, abs(sum(W_num,2))*T, 'r');
title('Magnitude Spectrum of x(t)');
hold off;

x = heaviside(2.*t+0.5) - heaviside(2.*t-0.5);
w_length = 2*pi/T;
W = w_length/N;
w = (0:N-1)*W;
disp('f(2t) Computing FFT...');

tic
X = T*fft(x,N);
X = fftshift(X);
toc

x_num = x;
w_num = linspace(-w_length/2, w_length/2, N);
disp('Computing Numerical FT...');

tic
W_num = x_num .* exp(-1j * w_num' * t);
toc

subplot(3,2,5);
stem(t,x);
title('Signal x(t)');

subplot(3,2,6);
stem(w-w_length/2,abs(X));
hold on;
plot(w_num, abs(sum(W_num,2))*T, 'r');
title('Magnitude Spectrum of x(t)');
hold off;

saveas(gcf, './p3.png');

```

运行结果如图7.11所示：

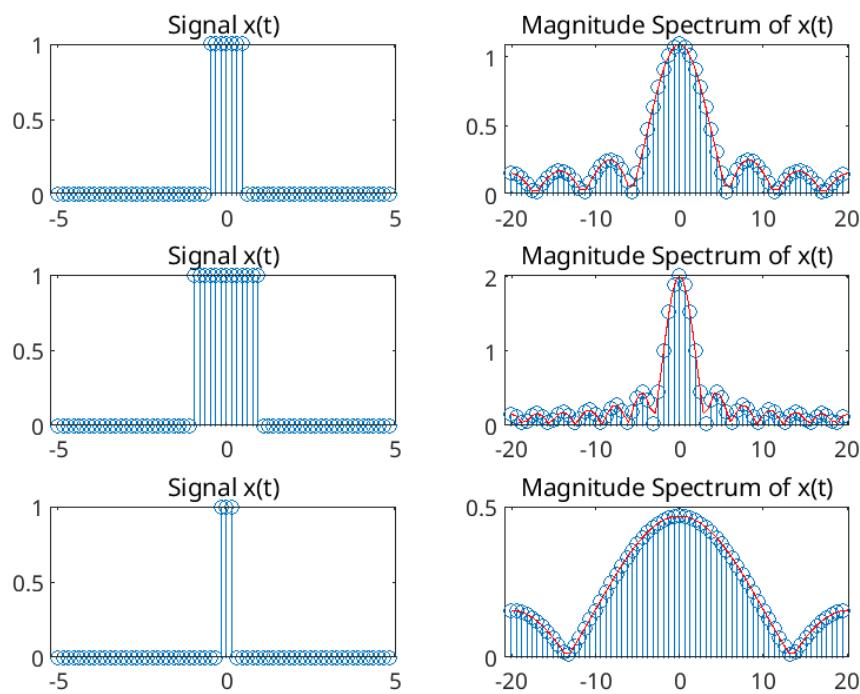


图 7.11. 任务三运行结果

终端输出结果如下所示：

Listing 24: 任务三终端输出结果

```
f(t) Computing FFT...
历时 0.000569 秒。
Computing Numerical FT...
历时 0.000585 秒。
f(t/2) Computing FFT...
历时 0.000154 秒。
Computing Numerical FT...
历时 0.000244 秒。
f(2t) Computing FFT...
历时 0.000144 秒。
Computing Numerical FT...
历时 0.000244 秒。
```

大作业报告（第八次实验）

8 噪声抑制算法设计与实现

8.1 设计思路

本算法针对 1800Hz 窄带噪声构建高斯陷波滤波器，通过分帧频域处理实现噪声精准抑制，核心逻辑为将信号分帧后转换至频域，利用高斯型带阻特性衰减目标噪声频率，最终重构时域信号。

背景知识

1. 音频信号的非平稳特性：音频/电信号属于典型的非平稳时变信号，全局处理无法精准针对局部噪声特征，需通过分帧将信号划分为若干短时帧（通常 10~30ms），每帧可近似视为短时平稳信号，为频域处理提供前提。
2. 窄带噪声抑制的通用思路：单一频率/窄带噪声（如电磁干扰、设备谐振噪声）无法通过时域滤波精准抑制，频域陷波滤波是主流方案；传统矩形陷波滤波器易引入吉布斯现象（频谱振铃），高斯型陷波滤波器凭借平滑过渡带可显著降低信号失真。
3. 分帧加窗的通用准则：分帧会导致信号截断，引入频谱泄漏，汉明窗、汉宁窗等余弦类窗函数可通过平滑帧边缘抑制泄漏；重叠率（通常 25%~75%）是平衡帧间连续性与计算效率的关键，高重叠率（如 75%）可避免重构后信号断裂。
4. 频域处理的核心逻辑：快速傅里叶变换（FFT）将时域信号转换为频域，可分离幅值谱与相位谱，仅对幅值谱进行带阻/带通处理（相位保持），能避免音色失真，逆 FFT（IFFT）可将处理后的频域信号转回时域。
5. 信号重构的鲁棒性设计：重叠相加法是分帧处理后信号重构的通用方法，需对重叠区域幅值加权平均（权重为窗函数附加值），补偿加窗导致的幅值衰减；同时需处理帧长超限、除零等异常，保证输出信号长度与输入一致。

技术原理

1. 分帧与加窗原理：将原始信号划分为若干重叠帧，帧长由采样率和 10ms 时间窗长确定，重叠率 75% 保证帧间连续性。汉明窗表达式为：

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \dots, N-1$$

其中 N 为帧长，加窗可抑制频域分析的频谱泄漏。

2. 高斯陷波滤波器设计：针对 1800Hz 中心噪声频率，构造高斯型带阻增益函数：

$$\text{bandElimination} = 1 - \exp\left(-\frac{d(f)^2}{2\sigma^2}\right)$$

式中 $d(f) = \min(|f - f_0|, |f_s - f - f_0|)$ ($f_0 = 1800\text{Hz}$ 为噪声中心频率， f_s 为采样率)，保证频率轴对称特性； $\sigma = 300\text{Hz}$ 为高斯函数标准差，决定带阻宽度。

3. 频域处理与信号重构：

- (a) 对每帧加窗信号做快速傅里叶变换（FFT），分离幅值谱 $|X(k)|$ 和相位谱 $\angle X(k)$ ；
- (b) 幅值谱乘以高斯陷波增益，相位谱保持不变，重构频域信号： $X'(k) = |X(k)| \cdot \text{bandElimination} \cdot e^{j\angle X(k)}$ ；

- (c) 对重构频域信号做逆 FFT (IFFT) 转回时域;
- (d) 采用重叠叠加法重构完整信号, 对重叠区域幅值加权平均 (权重为窗函数附加值), 补偿加窗导致的幅值衰减。

8.2 具体实现过程

算法通过 MATLAB 函数 ‘noise_suppress’ 实现, 输入为待降噪信号 y 和采样率 fs , 输出为降噪后信号 Y 和采样率 Fs , 具体步骤如下:

步骤 1: 输入校验与预处理

1. 校验输入信号 y 是否为向量, 若非向量则抛出错误并提示当前维度;
2. 将信号转换为列向量, 记录信号长度 sig_len , 将采样率赋值给输出变量 Fs 。

步骤 2: 分帧参数计算

1. 设定帧时间为 10ms, 计算帧长 $frameLength = \text{round}(0.010 \times fs)$; 若帧长大于信号长度, 则调整为 $\max(\text{floor}(sig_len/2), 2)$, 保证分帧有效性;
2. 设定重叠率 75%, 计算重叠点数 $overlapLength = \text{floor}(frameLength \times 0.75)$, 帧移 $frameStep = frameLength - overlapLength$;
3. 计算总帧数: $frameNumber = \text{ceil}((sig_len - frameLength)/frameStep) + 1$, 确保所有信号点被覆盖。

步骤 3: 信号分帧

1. 初始化帧矩阵 $frames$ (维度: 帧数 \times 帧长);
2. 遍历每帧, 计算帧起始索引 $start_idx = (i-1) \times frameStep + 1$ 和结束索引 $end_idx = start_idx + frameLength - 1$;
3. 若结束索引超出信号长度, 对该帧补零; 否则直接截取对应信号段, 完成帧矩阵填充。

步骤 4: 加窗处理

1. 生成汉明窗 $window$, 维度与帧长一致;
2. 将帧矩阵与窗函数按列相乘 ($windowedFrames = frames \cdot window'$), 完成每帧的加窗操作, 抑制频谱泄漏。

步骤 5: 频域降噪处理

1. 计算频率轴: $freq = (0 : frameLength - 1) \times (fs/frameLength)$;
2. 遍历每帧执行以下操作:
 - (a) 对加窗帧做 FFT 得到 $frameFFT$, 提取幅值 $magnitude = \text{abs}(frameFFT)$ 和相位 $phase = \text{angle}(frameFFT)$;
 - (b) 计算频率与 1800Hz 的距离 $freqDist$, 构造高斯陷波增益 $bandElimination$;
 - (c) 重构频域信号 $denoisedFrameFFT = magnitude \cdot bandElimination \cdot \exp(1j \times phase)$;
 - (d) 对重构频域信号做 IFFT 并取实部, 得到降噪后的时域帧 $denoisedFrame$ 。

步骤 6：窗补偿与信号重构

1. 初始化输出信号 Y 和权重向量 $weight$ （均为全零列向量）；
2. 遍历每帧，将降噪帧叠加到 Y 的对应位置，同时将窗函数叠加到 $weight$ 的对应位置；若帧结束索引超出信号长度，仅处理有效长度部分；
3. 对权重非零区域，将 Y 除以权重完成幅值补偿；权重为零区域赋值为 0，避免除零错误，保证输出信号长度与输入完全一致。

步骤 7：滤波器特性可视化

1. 创建图形窗口，绘制高斯陷波滤波器的频率响应曲线（横轴为频率，纵轴为增益）；
2. 设置坐标轴范围（1000 2500Hz），添加网格和边框，保存图片至指定路径后关闭窗口；
3. 实际滤波器如图8.1

算法核心特性

1. 鲁棒性：处理了帧长大于信号长度、帧索引越界、除零等异常情况，避免程序崩溃；
2. 低失真：采用高斯陷波滤波器，过渡带平滑，保持相位不变，降低信号失真；
3. 完整性：重构信号长度与输入完全一致，保证时序连续性。

8.3 程序代码及运行结果

程序代码

代码分为三部分：

1. ‘noise_suppress’：包含一个函数，输入音频，返回降噪音频
2. ‘draw_spectrogram’：包含一个函数，输入音频，绘制语谱图并返回音频语谱图的句柄
3. ‘runNoiseSuppress’：控制流，添加噪声，去噪，绘图，保存音频

下面是代码：

Listing 25: noise_suppress

```
function [Y, Fs] = noise_suppress(y, fs)
% 降噪函数：修复信号长度/维度/幅值异常问题
% 输入：y-输入信号（向量），fs-采样率
% 输出：Y-输出信号（列向量，长度与输入完全一致），Fs-采样率

if ~isvector(y)
    error('输入y必须是向量！当前是矩阵，维度: %s', mat2str(size(y)));
end
y = y(:);
sig_len = length(y);
Fs = fs;
```

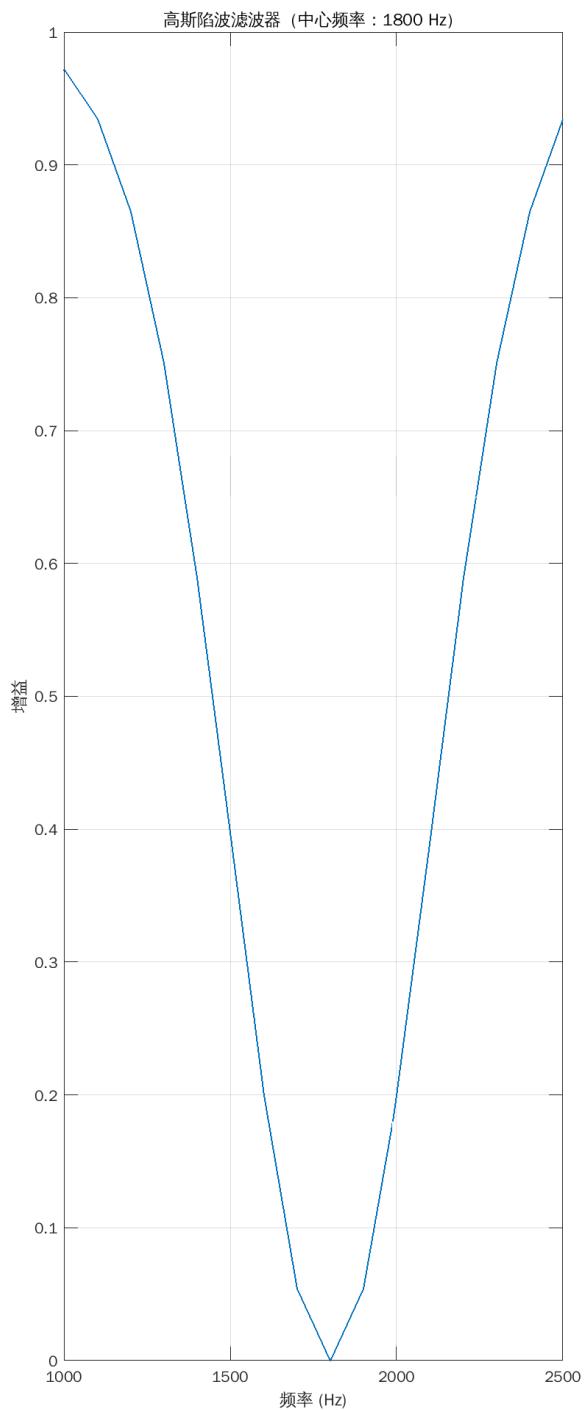


图 8.1. 高斯陷波滤波器频率响应 (中心频率 1800Hz)

```

frameTime = 0.010; % 帧长 10ms
frameLength = round(frameTime * fs); % 帧长 (点数)
if frameLength >= sig_len
    frameLength = max(floor(sig_len/2), 2);
end
overlapRate = 0.75; % 重叠率
overlapLength = floor(frameLength * overlapRate); % 重叠点数
frameStep = frameLength - overlapLength; % 帧移

frameNumber = ceil((sig_len - frameLength) / frameStep) + 1;

frames = zeros(frameNumber, frameLength); % 帧矩阵: 帧数×帧长
for i = 1:frameNumber
    start_idx = (i-1)*frameStep + 1;
    end_idx = start_idx + frameLength - 1;

    if end_idx > sig_len
        frame_data = zeros(1, frameLength);
        valid_len = sig_len - start_idx + 1;
        frame_data(1:valid_len) = y(start_idx:end);
        frames(i, :) = frame_data;
    else
        frames(i, :) = y(start_idx:end_idx)';
    end
end

window = hamming(frameLength);
windowedFrames = frames .* window';

noiseFreq = 1800; % 噪声中心频率
eliminationRate = 300; % 带阻宽度
denoisedFrames = zeros(size(windowedFrames)); % 降噪帧矩阵

freq = (0:frameLength-1) * (fs / frameLength);

for i = 1:frameNumber

    frameFFT = fft(windowedFrames(i, :));
    magnitude = abs(frameFFT); % 幅值
    phase = angle(frameFFT); % 相位

    freqDist = abs(freq - noiseFreq);
    freqDist = min(freqDist, abs(fs - freq - noiseFreq));
    bandElimination = 1 - exp(-freqDist.^2 / (2 * eliminationRate^2));
    % 频域降噪
    denoisedFrameFFT = magnitude .* bandElimination .* exp(1j * phase);

    denoisedFrame = real(ifft(denoisedFrameFFT));
    denoisedFrames(i, :) = denoisedFrame;
end

```

```

compensationWindow = hamming(frameLength); % 补偿窗

denoisedFrames = denoisedFrames ./ (compensationWindow' + eps) * 0.5;
denoisedFrames(isnan(denoisedFrames) | isnan(denoisedFrames)) = 0;

Y = zeros(sig_len, 1);
weight = zeros(sig_len, 1);
for i = 1:frameNumber
    start_idx = (i-1)*frameStep + 1;
    end_idx = start_idx + frameLength - 1;

    if end_idx > sig_len
        end_idx = sig_len;
    valid_len = end_idx - start_idx + 1;
    Y(start_idx:end_idx) = Y(start_idx:end_idx) + denoisedFrames(i, 1:valid_len)';
    weight(start_idx:end_idx) = weight(start_idx:end_idx) + window(1:valid_len);
    else
        Y(start_idx:end_idx) = Y(start_idx:end_idx) + denoisedFrames(i, :)';
        weight(start_idx:end_idx) = weight(start_idx:end_idx) + window;
    end
end
Y(weight > 0) = Y(weight > 0) ./ weight(weight > 0);
Y(weight == 0) = 0;

figure('Name', '1800Hz高斯陷波滤波器');
plot(freq, bandElimination, 'LineWidth', 1.2);
xlabel('频率 (Hz)'); ylabel('增益');
title(sprintf('高斯陷波滤波器 (中心频率: %d Hz)', noiseFreq));
xlim([1000, 2500]); grid on; box on;
saveas(gcf, './filter.png');
close(gcf);

end

```

Listing 26: draw_spectrogram

```

function fig = draw_spectrogram(signal, fs, save_path)

if ~isa(signal, 'double')
    signal = double(signal);
end
if ismatrix(signal)
    signal = signal(:, 1);
end
signal = signal(:);

% 语谱图参数
window_size = 256;

```

```

overlap = 128;
nfft = 512;

[S, F, T] = spectrogram(signal, window_size, overlap, nfft, fs);
S_abs = abs(S);
S_abs(S_abs == 0) = eps;
S_dB = 10*log10(S_abs);

fig_width = 1200;
fig_height = 800;
fig = figure(...  

    'Name', 'Spectrogram', ...  

    'Position', [100, 100, fig_width, fig_height], ... % [左偏移, 下偏移, 宽, 高]  

    'Units', 'inches', ...  

    'Color', 'white' ...  

);

ax = axes(...  

    'Parent', fig, ...  

    'Position', [0.08, 0.1, 0.75, 0.8], ... % [左, 下, 宽, 高]  

    'Units', 'normalized' ...  

);  

imagesc(ax, T, F, S_dB);
axis(ax, 'xy');
xlabel(ax, 'Time (s)', 'FontSize', 12);
ylabel(ax, 'Frequency (Hz)', 'FontSize', 12);
title(ax, 'Spectrogram (Unified Scale & Width)', 'FontSize', 14);
set(ax, 'FontSize', 10); %

cb = colorbar(ax, 'Position', [0.87, 0.1, 0.03, 0.8]);
cb.Label.String = 'Amplitude (dB)';
cb.Label.FontSize = 11;
set(cb, 'FontSize', 10);

caxis(ax, [-80 20]);

if nargin >= 3 && ~isempty(save_path)

    print(fig, save_path, '-dpng', '-r600', '-painters');
    fprintf('语谱图已保存至: %s (600dpi, 尺寸: %dx%d英寸) \n', save_path, fig_width, fig_height);
end
end

```

Listing 27: runNoiseSuppress

```

clc; close all; clear;
audioPath = '../name.wav'
outputPath = './denoised_name.wav'

```

```

outputNoisedPath = './noised_name.wav'

% 噪声参数
noiseAmplitude = 0.02; % 噪声幅度
noiseFrequency = 1800; % 噪声频率 Hz

[y, fs] = audioread(audioPath);

original_fig = draw_spectrogram(y,fs);
saveas(original_fig, 'original_spectrogram.png')

% 添加噪声
noise = noiseAmplitude * sin(2 * pi * noiseFrequency * (0:length(y)-1)' / fs);
noisedSignal = y + noise;
audiowrite(outputNoisedPath, noisedSignal, fs);

% 绘制语谱图分析
% fig = draw_spectrogram(y, fs);
noised_fig = draw_spectrogram(noisedSignal, fs);
% saveas(fig, 'original_spectrogram.png');
saveas(noised_fig, 'noised_spectrogram.png');

% 降噪处理
[Y, Fs] = noise_suppress(noisedSignal, fs);
denoised_fig = draw_spectrogram(Y, Fs);
saveas(denoised_fig, 'denoised_spectrogram.png');
audiowrite(outputPath, Y, Fs);

figure;
stem(y, 'b', 'filled');
hold on;
stem( noisedSignal, 'r');
stem( Y, 'g');
hold off;
disp('Noise suppression processing completed.');

```

运行结果

首先是原音频的语谱图8.2:

加入噪声后的音频语谱图如下图8.3

在纵轴的 1800HZ 处可以明显看到噪声对应的频谱信号。

使用滤波器去噪后的音频语谱图如下 8.4

经过去噪后，可见噪声已经有明显衰减。

所有处理后的语音都在在文件夹 ‘/matlab/noiseElimination/‘中储存。包括：

- ‘noised_name.wav’: 加入噪声后的音频
- ‘denoised_name.wav’: 去除噪声后的音频

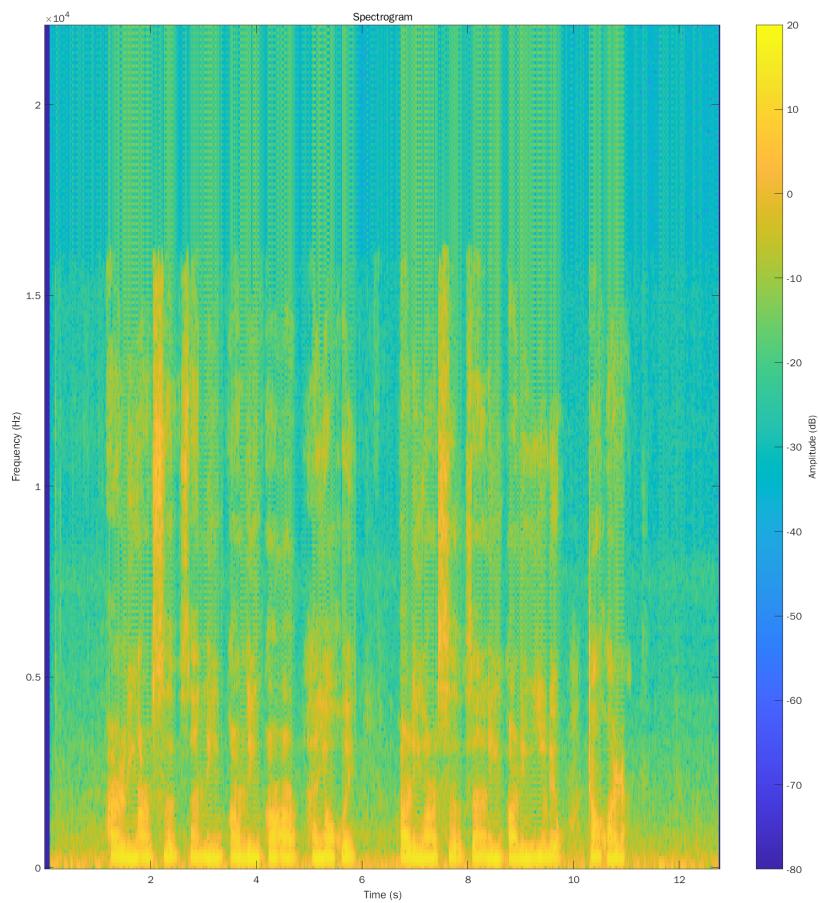


图 8.2. 原音频语谱图

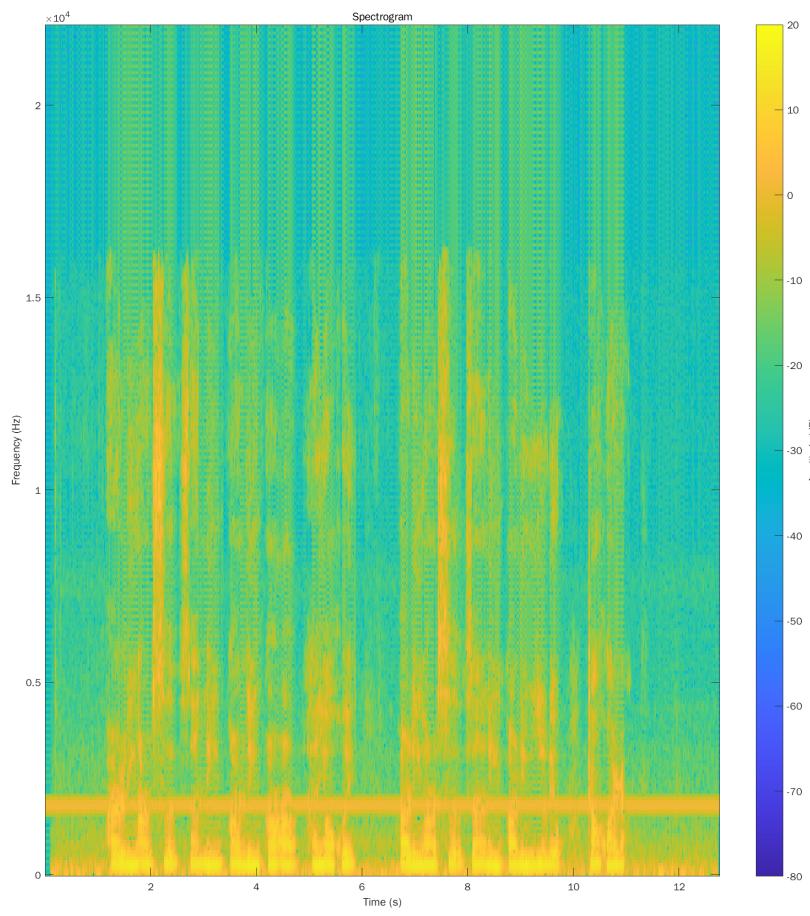


图 8.3. 加入噪声后音频语谱图

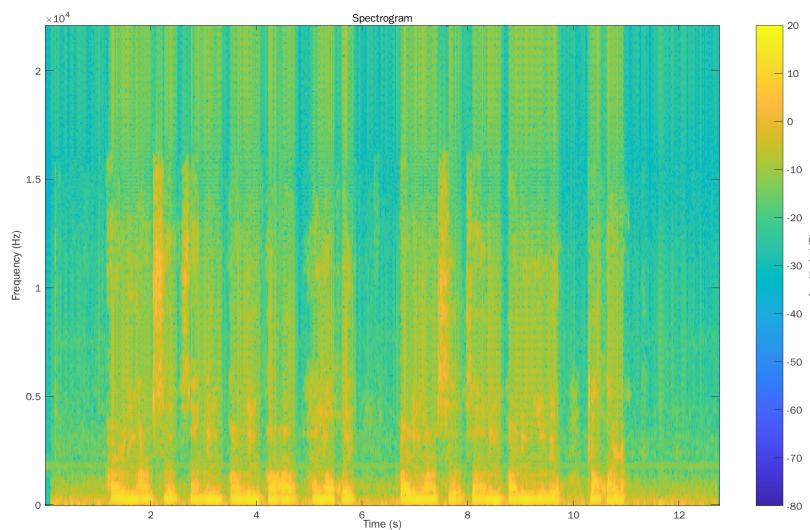


图 8.4. 去除噪声后的语谱图

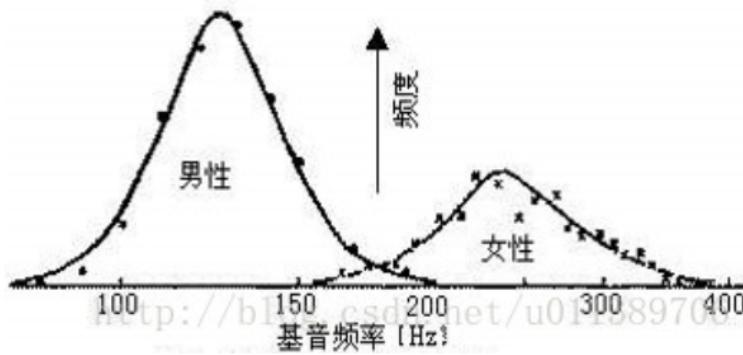


图 9.1. 不同性别音色频谱图

9 女声转男声音频处理算法设计与实现

9.1 设计思路

本算法实现音频的变调不变速，以男女声的音频的转换为例。

背景知识

- 人声音高与音色的核心特征：基频（Fundamental Frequency）决定音高，女声基频（200~500Hz）显著高于男声（85~180Hz），调整基频是女声转男声的核心手段；共振峰（Formant）决定音色，是否保留共振峰直接影响变声后音色的自然度。
- 短时傅里叶变换（STFT）的窗长适配：STFT 窗长决定频率/时间分辨率，窗长过小（如 512 点）会导致频率分辨率不足，信号处理失真；1024 点窗长是人声处理的经典取值，可平衡分辨率与处理效率。
- 高频增强的合理方式：人声清晰感依赖 1000~5000Hz 高频成分，硬高通滤波会滤除 0~3kHz 人声核心频段，导致声音丢失；“带通提取 + 温和叠加”的方式仅增强高频，不破坏核心频段，是人声增强的通用策略。
- 音频幅值归一化的必要性：音频处理后易出现幅值异常（过小/过大），将幅值归一化到 [-1,1] 可避免削波失真，增益调整（如 0.9）可保证最终音量适中；同时需检测处理后信号丢失等异常，降级使用原始音频避免程序报错。
- 频谱分析的验证价值：通过 FFT 计算单侧边频谱是验证音频处理效果的通用手段，可直观对比处理前后核心频段（如 0~3kHz）的幅值分布，确认核心信号未丢失。

技术原理

- 基频调整原理：基于 STFT 的音高偏移算法（shiftPitch 函数），通过调整每帧信号的基频实现音高降低（nsemitones 为半音数，负值表示降频）；设置汉明窗、75
- 温和高频增强原理：替代硬高通滤波，采用 2 阶巴特沃斯带通滤波器提取 1000~5000Hz 高频成分，将其以小幅增益叠加回原信号，仅增强高频清晰度，不滤除 0~3kHz 人声核心频段。

3. 频谱分析原理：通过快速傅里叶变换（FFT）计算音频单侧边频谱，公式如下：

$$\begin{aligned} P_2 &= |\text{FFT}(audio)|/L, \quad (L) \\ P_1 &= P_2(1 : L/2 + 1), \quad (\text{单侧边频谱}) \\ P_1(2 : \text{end} - 1) &= 2 \cdot P_1(2 : \text{end} - 1), \quad (\text{幅值补偿}) \\ f_p &= f_s \cdot (0 : (L/2))/L, \quad (\text{频率轴}) \end{aligned}$$

其中 P_1 为单侧边幅值谱， f_p 为频率轴，通过对比原始/处理后频谱验证核心频段信号存在。

4. 音量归一化原理：将处理后信号幅值归一化到 [-1,1] 区间，再通过增益系数调整音量，同时增加防除零处理，避免信号丢失时程序报错。

9.2 具体实现过程

算法通过 MATLAB 脚本实现女声转男声，核心解决“声音消失、音色闷/憨”问题，具体步骤如下：

步骤 1：环境初始化与音频读取

1. 执行环境清理：‘clear; clc; close all;’ 清空变量、清除命令行、关闭所有图形窗口；
2. 读取音频文件：调用 ‘audioread’ 读取 ‘..//name.wav’，返回音频数据 ‘audio_In’ 和采样率 ‘fs’；
3. 立体声转单声道：若音频为立体声（列数 >1），通过 ‘mean(audio_In, 2)’ 取列均值转为单声道；
4. 音频有效性校验：若音频最大幅值小于 1e-6，抛出错误提示“原始音频文件无信号”。

步骤 2：核心调整参数设置

1. 基频调整参数：‘nsemitones = -9.5’（降 9.5 个半音，平衡男声感和声音保留）；
2. 共振峰参数：‘preserve_formants = false’（不保留共振峰，避免音色闷/憨）；
3. 音量参数：‘gain = 0.9’（提升音量，避免声音过小）；
4. 高频增强参数：‘high_freq_boost = 1.05’（温和高频增益，仅小幅增强）。

步骤 3：基频调整（核心变声）

1. 设置窗长：‘win_len = 1024’（恢复合理窗长，避免 512 窗长导致的信号失真）；
2. 调用音高偏移函数：‘audio_Out = shiftPitch(audio_In, nsemitones, ...)’，关键参数：
 - (a) ‘PreserveFormants’：设为 ‘false’，不保留共振峰，避免音色闷/憨；
 - (b) ‘Window’：设为 ‘hamming(win_len)’，汉明窗减少频谱泄漏；
 - (c) ‘OverlapLength’：设为 ‘round(win_len * 0.75)’，75
 - (d) ‘LockPhase’：设为 ‘true’，锁相避免相位失真导致的破音。

步骤 4：温和高频增强（修复声音消失）

1. 设计带通滤波器：调用 ‘butter(2, [1000 5000]/(fs/2), ’bandpass)’‘设计 2 阶巴特沃斯带通滤波器，通带 1000-5000Hz；
2. 提取高频成分：调用 ‘filtfilt(b, a, audio_Out)’对变声后音频做零相位滤波，提取高频成分 ‘audio_high’；
3. 温和叠加高频：‘audio_Out = audio_Out + (audio_high * (high_freq_boost - 1))’，仅将高频成分以 0.05 倍增益叠加，不滤除核心频段。

步骤 5：音量归一化与异常处理

1. 信号丢失检测：若 ‘audio_Out’最大幅值小于 1e-6，弹出警告并将 ‘audio_Out’重置为原始音频 ‘audio_In’；
2. 幅值归一化：‘audio_Out = audio_Out / max(abs(audio_Out))’，将幅值归一化到 [-1,1]；
3. 音量提升：‘audio_Out = audio_Out * gain’，将归一化后的音频乘以增益系数 0.9，提升音量。

步骤 6：频谱对比绘制（验证信号）

1. 创建图形窗口：‘figure(’Color’, ’w’)’设置白色背景，开启 ‘hold on’ 和网格；
2. 绘制频谱：调用自定义子函数 ‘plotSpectrum’ 分别绘制原始音频（蓝色）和处理后音频（红色）的单侧边频谱；
3. 设置坐标轴：聚焦人声核心频段 ‘xlim([0 3000])’，设置纵轴范围 ‘ylim([0 0.03])’，添加标题、坐标轴标签和图例；

步骤 7：音频保存与播放

1. 保存音频：调用 ‘audiowrite(’to_male.wav’, audio_Out, fs)’ 将处理后音频保存为 ‘to_male.wav’；
2. 输出提示：‘disp(’ 音频已保存为 to_male.wav’)’，提示保存完成。

子函数：plotSpectrum（绘制单侧边频谱）

1. 输入参数：‘audio’（音频数据）、‘fs’（采样率）、‘label’（图例标签）、‘color’（线条颜色）；
2. 计算音频长度：‘L = length(audio)’；
3. 计算 FFT 并转换为单侧边频谱：按 1.2.2 节的频谱分析公式计算 P_1 和 f_p ；
4. 绘制频谱：‘plot(fp, P1, color, ’LineWidth’, 1.2, ’DisplayName’, label)’，设置线宽和图例。

算法核心特性

1. 问题修复：替换硬高通滤波为温和高频增强，避免核心频段丢失导致的声音消失；优化窗长和共振峰参数，避免音色闷/憋；
2. 鲁棒性：增加音频有效性校验、信号丢失异常处理，避免程序报错；
3. 可验证性：绘制频谱对比图，直观验证处理后核心频段信号存在；
4. 实用性：输出可播放的音频文件，参数可灵活调整以平衡“男声感”和“声音清晰度”。

9.3 程序代码及运行结果

程序代码

程序源码如下：

Listing 28: genderConversion

```

clear; clc; close all;

%% 1. 读取音频 (兼容立体声/单声道)
[audioIn, fs] = audioread('../name.wav');
if size(audioIn, 2) > 1
    audioIn = mean(audioIn, 2); % 转为单声道
end
% 检查原始音频是否有效
if max(abs(audioIn)) < 1e-6
    error('原始音频文件无信号, 请检查name.wav是否存在且有效! ');
end

%% 2. 核心调整参数 (平衡"男声感"和"不憋+有声音")
nsemitones = -9.5; % 基频: -8 (适度降频, 保留声音)
preserve_formants = false; % 保留共振峰 (核心: 避免闷/憋, 且不丢声音)
gain = 0.9; % 提高音量 (避免声音小)
high_freq_boost = 1.05; % 温和高频增强 (不滤掉核心频段)

%% 3. 核心: 调整基频 (恢复合理窗长, 避免信号丢失)
win_len = 1024; % 恢复1024窗长 (512大小易导致信号处理失真)
audioOut = shiftPitch(audioIn, nsemitones, ...
    'PreserveFormants', preserve_formants, ...
    'Window', hamming(win_len), ...
    'OverlapLength', round(win_len * 0.75), ...
    'LockPhase', true);

%% 4. 温和高频增强 (替代硬高通, 不丢声音)
% 问题根源: 2kHz高通滤掉了人声核心频段→声音消失!
% 修正: 用"低通+高频增益", 仅增强高频, 不滤掉低频/中频
% 设计1000Hz~5000Hz的带通增强 (人声清晰频段)
[b, a] = butter(2, [1000 5000]/(fs/2), 'bandpass'); % 2阶带通
audio_high = filtfilt(b, a, audioOut); % 提取高频
audioOut = audioOut + (audio_high * (high_freq_boost - 1)); % 温和增强高频

%% 5. 音量归一化 (避免失真+确保声音大小)
% 增加防除零: 若信号全零, 直接用原始音频 (避免报错)
if max(abs(audioOut)) < 1e-6
    warning('处理后信号丢失, 使用原始音频! ');
    audioOut = audioIn;
end
audioOut = audioOut / max(abs(audioOut)); % 归一化到[-1,1]
audioOut = audioOut * gain; % 提升音量

%% 6. 绘制频谱对比 (验证信号存在)
figure('Color','w'); hold on; grid on;
plotSpectrum(audioIn, fs, '原始女声', 'b');

```

```
plotSpectrum(audioOut, fs, '调整后男声（有声音+不愁）', 'r');

title('音频频谱对比（修复声音消失问题）');
xlabel('频率 (Hz)'); ylabel('幅值');
xlim([0 3000]); % 聚焦人声核心频段 (0-3kHz)
ylim([0 0.03]);
legend('Location','best');
hold off;

%% 7. 保存+播放音频
audiowrite('to_male.wav', audioOut, fs);
disp('音频已保存为 to_male.wav');

%% 子函数：绘制单侧边频谱
function plotSpectrum(audio, fs, label, color)
    L = length(audio);
    ffta = fft(audio);
    P2 = abs(ffta/L);
    P1 = P2(1:L/2+1);
    P1(2:end-1) = 2*P1(2:end-1);
    fp = fs*(0:(L/2))/L;
    plot(fp, P1, color, 'LineWidth', 1.2, 'DisplayName', label);
end
```

运行结果

音频变调后结果如下图9.2：

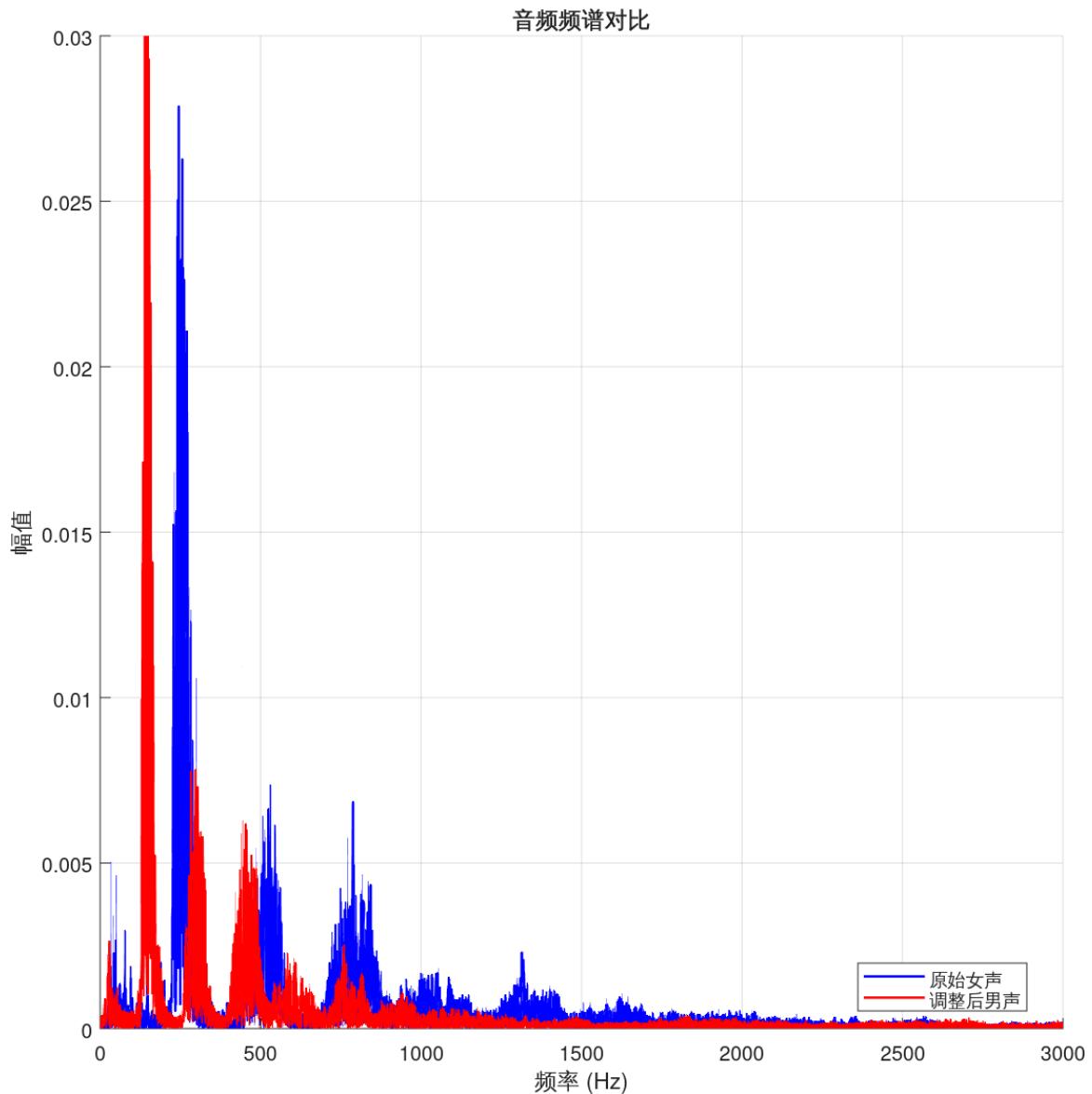


图 9.2. 女声转男声音频频谱对比 (0~3kHz 核心频段)

可见变调后频谱整体左移，低频分量增加，高频分量减小。

输出音频保存在 ‘./matlab/genderConverse/to_male.wav’

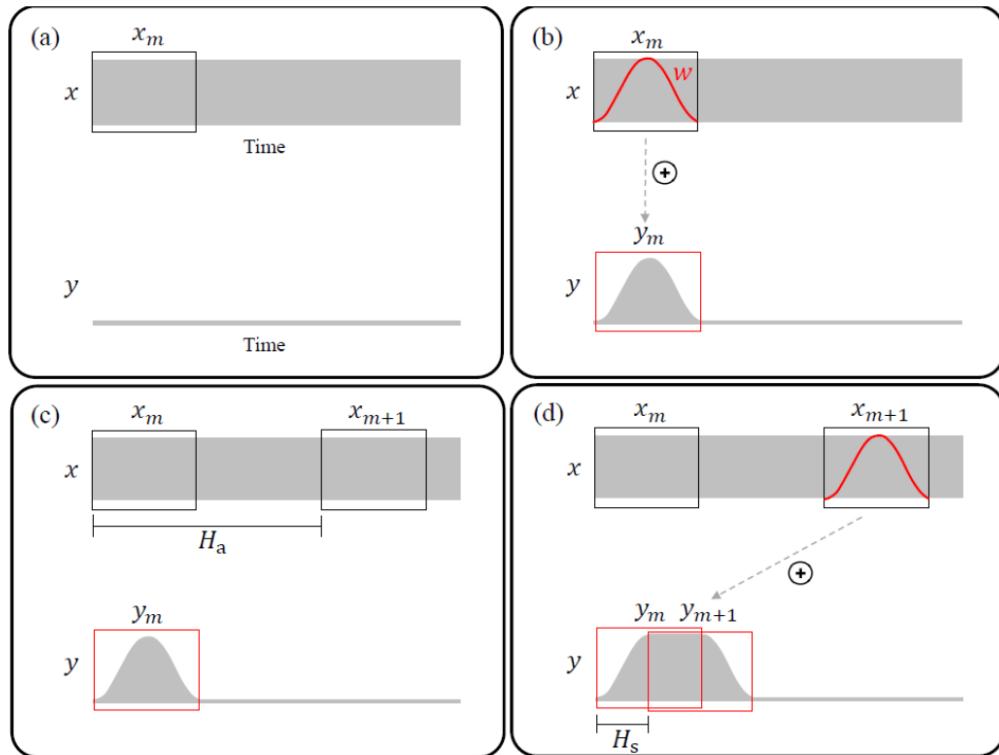


图 10.1. 分帧与复原

10 音频变速算法设计与实现

10.1 设计思路

本算法基于时域分帧重叠相加原理实现音频变速，核心逻辑是通过调整输出帧的重叠长度改变帧移，在保持音频音调不变的前提下调整播放时长，避免频域处理的复杂度与失真问题。

背景知识

背景知识

1. 音频变速的时域实现逻辑：变速的核心是改变帧移（帧间步进长度），而非修改采样率（采样率不变可保证音调不偏移）；加速时增大帧移，减速时减小帧移，是时域变速的通用思路。
2. 分帧参数的通用计算：帧长通常由固定时间窗（如 10ms）与采样率换算($frame_Length = frame_Time \times fs$)，重叠率（25% 75%）需根据场景适配——低重叠率（如 25%）提升计算效率，高重叠率提升信号连续性。
3. 重叠相加法的重构原理：输出信号长度由帧长、帧数、输出重叠长度共同决定，每帧按输出帧移叠加到对应位置，重叠区域幅值累加可保证时域连续性，避免拼接毛刺。
4. 窗函数的通用作用：汉明窗是音频分帧处理的标配窗函数，可抑制频谱泄漏，同时使帧间叠加更平滑；窗函数需与帧矩阵维度匹配（行/列向量对齐），避免维度错误。
5. 变速算法的兼容性：时域重叠相加法无需频域变换，计算复杂度低，可兼容加速（rate>1）与减速（rate<1）场景，仅需调整输出重叠率即可适配不同变速比率。

技术原理

1. 分帧参数计算原理:

- (a) 输入帧长 (点数): $frame_Length = frame_Time \times fs$ ($frame_Time=0.010s$ 为固定帧时间, fs 为采样率);
- (b) 输入重叠长度: $overlapLength = \text{floor}(frame_Length \times overlapRate)$ ($overlapRate=0.25$ 为输入重叠率);
- (c) 输入帧数: $frameNumber = \text{floor}((\text{length}(y)-frame_Length)/(frame_Length-overlapLength))+1$, 确保所有有效输入信号被分帧覆盖;
- (d) 输出重叠率自适应: $outputOverlapRate = \frac{rate+overlapRate-1}{rate}$, $rate>1$ 时输出重叠率降低, 输出帧移增大, 总时长缩短 (实现加速);
- (e) 输出重叠长度: $outputOverlapLength = \text{floor}(frame_Length \times outputOverlapRate)$.

2. 加窗原理: 采用汉明窗抑制频谱泄漏, 汉明窗公式为:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \dots, N-1$$

其中 $N = frame_Length$ 为帧长, 窗函数与帧矩阵按元素相乘实现加窗。

3. 重叠相加重构原理: 输出信号长度为 $frame_Length + (frameNumber - 1) \times (frame_Length - outputOverlapLength)$, 每帧信号按输出帧移 ($frame_Length - outputOverlapLength$) 叠加到对应位置, 重叠区域幅值累加, 保证信号连续。
4. 变速核心逻辑: $rate>1$ 时, 输出帧移 ($frame_Length - outputOverlapLength$) 大于输入帧移 ($frame_Length - overlapLength$), 相同帧数的信号在更短的时长内完成叠加, 实现音频加速; 采样率保持不变 ($Fs=fs$), 保证音调不发生偏移。

10.2 具体实现过程

算法通过 MATLAB 函数 ‘speed_change’ 实现音频变速, 具体步骤如下:

步骤 1: 基本参数计算与初始化

1. 设定固定帧时间: $frame_Time = 0.010$ (单位: 秒, 对应 10ms 帧长);
2. 计算输入帧长 (点数): $frame_Length = frame_Time \times fs$;
3. 设定输入重叠率: $overlapRate = 0.25$, 计算输入重叠长度: $overlapLength = \text{floor}(frame_Length \times overlapRate)$;
4. 计算输入总帧数: $frameNumber = \text{floor}((\text{length}(y)-frame_Length)/(frame_Length-overlapLength))+1$;
5. 计算输出重叠率: $outputOverlapRate = (rate + overlapRate - 1)/rate$, 并推导输出重叠长度: $outputOverlapLength = \text{floor}(frame_Length \times outputOverlapRate)$.

步骤 2：输入信号分帧

1. 初始化帧矩阵 $frames$ (维度: $frameNumber \times frame_Length$), 初始值全为 0;
2. 遍历每帧 ($i = 1 : frameNumber$), 计算当前帧的信号索引: 起始索引: $(i - 1) \times (frame_Length - overlapLength) + 1$, 结束索引: $(i - 1) \times (frame_Length - overlapLength) + frame_Length$;
3. 将输入信号 y 的对应索引段赋值给帧矩阵 $frames(i, :)$, 完成分帧。

步骤 3：帧加窗处理

1. 生成汉明窗 $window$: $window = hamming(frame_Length)'$ (转置为行向量, 匹配帧矩阵维度);
2. 帧矩阵与窗函数按元素相乘: $windowedFrames = frames.*window$, 完成每帧的加窗处理, 抑制频谱泄漏。

步骤 4：重叠相加重构输出信号

1. 初始化输出信号 Y : 维度为 $1 \times (frame_Length + (frameNumber - 1) \times (frame_Length - outputOverlapLength))$, 初始值全为 0;
2. 遍历每帧 ($i = 1 : frameNumber$), 计算当前帧在输出信号中的起始索引: $startIndex = (i - 1) \times (frame_Length - outputOverlapLength) + 1$;
3. 将加窗后的帧 $windowedFrames(i, :)$ 叠加到 Y 的 $startIndex : startIndex + frame_Length - 1$ 区间, 重叠区域幅值累加;
4. 遍历完成后, Y 即为重构后的变速音频信号。

步骤 5：采样率赋值

1. 将输出采样率 Fs 赋值为输入采样率 fs , 保证音频音调不随速度变化。

算法核心特性

1. 音调保持: 采样率全程不变 ($Fs=fs$), 仅通过调整帧重叠长度改变播放时长, 保证变速后音调不偏移;
2. 低复杂度: 基于时域分帧重叠相加实现, 无需频域变换, 计算效率高;
3. 低失真: 汉明窗抑制频谱泄漏, 重叠相加保证帧间连续性, 避免信号断裂/毛刺;
4. 灵活性: 变速比率 $rate$ 可灵活调整, $rate > 1$ 实现加速, $rate < 1$ 可实现减速 (算法兼容);
5. 鲁棒性: 帧数计算基于整数取整, 确保分帧覆盖有效信号, 无索引越界风险。

10.3 程序代码和运行结果

程序代码

程序代码包含两部分:

1. ‘speed_change’: 包含一个函数, 输入一个音频和一个标量 (加速倍率) 返回加减速后的音频
2. ‘runSpeedChange’: 控制流, 控制读入音频, 加减速音频和保存结果

Listing 29: speed_change

```

function [Y, Fs] = speed_change(y, fs, rate)
% Y 输出信号, 向量形式
% Fs 输出采样率
% y 输入信号, 向量形式
% fs 输入采样率
% rate 变数比率, >1 加速

% 基本参数设置

frameTime = 0.010; % 帧长 ms
frameLength = frameTime * fs; % 帧长 点数
overlapRate = 0.25; % 重叠率
overlapLength = floor(frameLength * overlapRate); % 重叠点数
frameNumber = floor((length(y)- frameLength) / (frameLength - overlapLength)) + 1; % 帧数
outputOverlapRate = (rate + overlapRate - 1) / rate; % 输出重叠率
outputOverlapLength = floor(frameLength * outputOverlapRate); % 输出重叠点数

% 分帧
frames = zeros(frameNumber, frameLength);
for i = 1:frameNumber
    frames(i, :) = y((i-1)*(frameLength - overlapLength) + 1 : (i-1)*(frameLength - overlapLength)
        + frameLength);
end

% 加窗
window = hamming(frameLength)';
windowedFrames = frames .* window;

% 重叠相加
Y = zeros(1, frameLength+ (frameNumber -1)*(frameLength - outputOverlapLength));
for i = 1:frameNumber
    startIndex = (i-1)*(frameLength - outputOverlapLength) + 1;
    Y(startIndex : startIndex + frameLength -1) = Y(startIndex : startIndex + frameLength -1) +
        windowedFrames(i, :);
end

Fs = fs;

end

```

Listing 30: runSpeedChange

```

% 基本信息
audiopath = '../name.wav';
outputpathFast = './fastname.wav'
outputpathSlow = './slowname.wav'

[y,Fs] = audioread(audiopath);

```

```
[Yf, Fsf] = speed_change(y, Fs, 2);
audiowrite(outputpathFast, Yf, Fsf);

[ys, Fss] = speed_change(y, Fs, 0.5);
audiowrite(outputpathSlow, ys, Fss);

%输出基本信息
disp('Speed change processing completed.');
```

代码运行结果

输出音频存储在 ‘./matlab/changeSpeed/‘ 包含

- ‘slowname.wav’: 减速后的音频
- ‘fastname.wav’: 加速后的音频

11 基于能量阈值的语音段自动分割与反转拼接算法实现

11.1 设计思路

本算法针对数字语音音频（110 段）实现“自动分割-段起始前移-反转拼接”全流程处理，核心逻辑是通过短时能量分析识别语音段，自适应调整能量阈值匹配目标分割段数，再对语音段进行前移修正和反转拼接，最终输出反转后的音频并可视化验证。

背景知识

背景知识

1. 语音端点检测的核心依据：语音段的短时能量远高于静音段，滑动窗口计算短时能量（均方根能量）是端点检测的经典方案，窗口时长（20ms）是语音处理的通用取值，可精准区分语音/静音。
2. 阈值自适应的鲁棒性设计：固定能量阈值易受音频信噪比、说话人音量影响，导致分割结果偏离目标；通过小幅迭代调整阈值（如 $\pm 10\%$ ），并限制最大调整次数（如 3 次），是平衡效果与收敛性的通用策略。
3. 语音段过滤的通用规则：需设置最小段长（如 100ms）过滤误分割的短段，避免无效段干扰；段起始前移（如 0.25 秒）可补偿能量上升延迟，避免丢失语音段开头的有效信息。
4. 音频拼接的时序完整性：拼接时需保留原始静音间隔，保证反转后音频的时序逻辑与原音频一致；静音间隔包括段间间隔与最后一段后的间隔，是拼接类算法的核心考量。
5. 可视化验证的通用价值：绘制原始/处理后音频波形，标记关键特征点（如段起始），可直观验证分割与拼接效果；设置中文显示适配（如 SimHei 字体）是可视化的基础要求。

技术原理

1. 短时能量计算原理：对音频每个采样点，取以其为中心的 20ms 滑动窗口，计算窗口内音频的均方根能量，公式为：

$$E(i) = \sqrt{\frac{1}{N} \sum_{n=start_idx}^{end_idx} x(n)^2}$$

其中 N 为窗口内采样点数， $start_idx = \max(1, i - half_win)$ ， $end_idx = \min(total_samples, i + half_win)$ ， $x(n)$ 为音频采样值；

2. 能量阈值分割：设定能量阈值 $energy_thresh$ ，当 $E(i) > energy_thresh$ 时标记为语音段起始，当 $E(i) \leq energy_thresh$ 时标记为语音段结束，同时过滤长度小于 min_seg_len 的短段（避免误分割）；
3. 阈值自适应调整：若分割段数少于目标值，将阈值降低 10%；若多于目标值，将阈值升高 10%，最多调整 3 次保证算法收敛；
4. 段起始前移：将每个语音段的起始索引前移 $shift_samples = fs \times shift_time$ 个采样点（ fs 为采样率， $shift_time$ 为前移时间），且保证前移后索引 1；
5. 反转拼接：先提取各语音段间的静音间隔，将语音段和间隔分别反转，再按“反转语音段 + 反转静音间隔”的顺序拼接，保证反转后音频的静音间隔与原音频一致。

11.2 具体实现过程

算法通过 MATLAB 脚本实现语音段分割与反转拼接，输入为数字语音音频文件，输出为反转拼接后的音频文件及可视化结果，具体步骤如下：

步骤 1：环境初始化与参数配置

1. 环境清理：执行 ‘clear; clc; close all;’ 清空变量、清除命令行、关闭所有图形窗口；
2. 配置核心参数：构建结构体 ‘cfg’，包含以下参数：
 - (a) ‘cfg.audio_path = ‘..//number.wav’’：输入音频文件路径；
 - (b) ‘cfg.shift_time = 0.25’：段起始前移时间（秒）；
 - (c) ‘cfg.energy_thresh = 0.05’：初始能量阈值；
 - (d) ‘cfg.win_size_ms = 20’：滑动窗口时长（毫秒）；
 - (e) ‘cfg.min_seg_len_ms = 100’：最小语音段长度（毫秒）；
 - (f) ‘cfg.target_seg_num = 10’：目标分割段数（匹配 1-10 数字语音）。

步骤 2：音频读取与预处理

1. 读取音频：调用 ‘audioread(cfg.audio_path)’ 返回音频数据 ‘audio_In’ 和采样率 ‘fs’；
2. 立体声转单声道：‘audio_In = mean(audio_In, 2)’，保证音频为列向量；
3. 幅值归一化：‘audio_In = audio_In / max(abs(audio_In))’，将幅值归一化到 [-1,1]；
4. 计算音频基本信息：总采样数 ‘total_samples = length(audio_In)’，总时长 ‘total_duration = total_samples / fs’，并打印音频信息。

步骤 3：短时能量计算

1. 窗口参数计算：滑动窗口采样数 ‘win_size = round(fs * cfg.win_size_ms / 1000)’，窗口半长 ‘half_win = floor(win_size / 2)’；
2. 初始化能量数组：‘audio_energy = zeros(total_samples, 1)’，存储每个采样点的短时能量；
3. 遍历每个采样点计算短时能量：
 - (a) 计算窗口起始索引：‘start_idx = max(1, i - half_win)’；
 - (b) 计算窗口结束索引：‘end_idx = min(total_samples, i + half_win)’；
 - (c) 截取窗口内音频：‘window_audio = audio_In(start_idx:end_idx)’；
 - (d) 计算均方根能量：‘audio_energy(i) = sqrt(mean(window_audio.^2))’。

步骤 4：语音段自动分割

1. 最小段长转换：‘min_seg_len = round(fs * cfg.min_seg_len_ms / 1000)’，将毫秒转换为采样点数；
2. 初始分割语音段：
 - (a) 初始化语音段矩阵 ‘segments’（存储起始/结束索引）、起始标记 ‘start_idx = 0’；
 - (b) 遍历采样点，当能量大于阈值且无起始标记时，标记 ‘start_idx = i’；

- (c) 当能量小于等于阈值且有起始标记时，标记结束索引 ‘end_idx = i’，若段长大于最小段长则存入 ‘segments’；
- (d) 遍历结束后，若仍有未结束的语音段且长度达标，补充存入 ‘segments’。

3. 自适应调整阈值：

- (a) 初始化调整次数 ‘adjust_count = 0’，最大调整次数 ‘max_adjust_times = 3’；
- (b) 若分割段数 目标段数且未达最大调整次数： - 段数不足： 阈值 $\times 0.9$ ，扩大识别范围； - 段数过多： 阈值 $\times 1.1$ ，缩小识别范围；
- (c) 重新分割语音段，更新段数 ‘num_seg = size(segments, 1)’；
- (d) 调整结束后，若段数仍不匹配，输出警告及调整建议；匹配则打印成功提示。

步骤 5：语音段起始前移修正

1. 计算前移采样数：‘shift_samples = round(fs * cfg.shift_time)’；
2. 初始化调整后段矩阵 ‘segments_adjusted = segments’；
3. 遍历每个语音段，将起始索引前移：‘segments_adjusted(i, 1) = max(1, segments_adjusted(i, 1) - shift_samples)’，保证索引不越界。

步骤 6：提取音频间隔（静音区）

1. 初始化间隔数组 ‘intervals_all’；
2. 提取段间间隔：遍历 1 num_seg-1 段，计算 ‘interval_between = segments_adjusted(i+1, 1) - segments_adjusted(i, 2)’，存入 ‘intervals_all’；
3. 提取最后一段后的间隔：‘interval_after_last = total_samples - segments_adjusted(end, 2)’，补充存入 ‘intervals_all’。

步骤 7：反转拼接音频

1. 反转语音段和间隔：‘reversed_segments = flipud(segments_adjusted)’，‘reversed_intervals = flipud(intervals_all)’；
2. 初始化输出音频 ‘audioOut = []’；
3. 遍历反转后的语音段：
 - (a) 截取当前段音频：‘seg_audio = audio_In(reversed_segments(i, 1):reversed_segments(i, 2))’；
 - (b) 拼接语音段：‘audioOut = [audioOut; seg_audio]’；
 - (c) 生成对应静音间隔：‘silence = zeros(reversed_intervals(i), 1)’；
 - (d) 拼接静音间隔：‘audioOut = [audioOut; silence]’。
4. 音量归一化：‘audioOut = audioOut / max(abs(audioOut)) * 0.9’，避免失真且保证音量适中。

步骤 8：音频保存与可视化

1. 保存音频：调用 `audiowrite('numberConverse.wav', audioOut, fs)`，输出保存提示；
2. 创建可视化窗口：`figure('Color','w','Position',[100 100 1000 700],'Name',' 音频反转结果可视化')`；
3. 绘制原始音频波形：
 - (a) 子图 1 (2,1,1)：绘制原始音频时域波形，标记原段起始（红色虚线）和前移后段起始（绿色实线），标注段序号；
 - (b) 设置坐标轴、网格、图例，保证中文显示正常。
4. 绘制反转后音频波形：
 - (a) 子图 2 (2,1,2)：绘制反转后音频时域波形，标记反转后段起始（蓝色虚线），标注反转后段序号；
 - (b) 设置坐标轴、网格，保证中文显示正常。
5. 保存可视化图片：`saveas(gcf, 'numberConverse_visualization.png')`；

算法核心特性

1. 自适应分割：通过能量阈值微调，自动匹配目标分割段数，降低人工调参成本；
2. 鲁棒性强：段起始前移避免丢失语音开头信息，最小段长过滤误分割短段，阈值调整次数限制避免无限循环；
3. 可视化验证：绘制原始/反转音频波形并标记语音段，直观验证分割和拼接效果；
4. 低失真：全程仅做幅值归一化，无频域处理，保证语音音色完整性；
5. 可扩展性：参数可灵活调整（如前移时间、窗口时长、目标段数），适配不同语音场景。

11.3 程序代码及运行结果

程序代码

Listing 31: 信号波形生成与绘图代码

```
clear; clc; close all;

%% 基础参数
cfg = struct();
cfg.audio_path = '../number.wav'; % 输入音频文件
cfg.shift_time = 0.25; % 段起始前移时间 (秒)
cfg.energy_thresh = 0.05; % 能量阈值，控制语音段识别
cfg.win_size_ms = 20; % 滑动窗口时长 (毫秒)
cfg.min_seg_len_ms = 100; % 最小语音段长度 (毫秒)
cfg.target_seg_num = 10; % 目标分割段数

[audioIn, fs] = audioread(cfg.audio_path);
audioIn = mean(audioIn, 2);
audioIn = audioIn / max(abs(audioIn));
total_samples = length(audioIn);
total_duration = total_samples / fs;
```

```

fprintf(' 音频信息: \n');
fprintf(' 采样率: %d Hz | 总时长: %.2f 秒 | 总样本数: %d\n', fs, total_duration, total_samples);
fprintf('-----\n');

win_size = round(fs * cfg.win_size_ms / 1000);
half_win = floor(win_size / 2);
audio_energy = zeros(total_samples, 1);

for i = 1:total_samples
    start_idx = max(1, i - half_win);
    end_idx = min(total_samples, i + half_win);
    window_audio = audioIn(start_idx:end_idx);
    audio_energy(i) = sqrt(mean(window_audio.^2));
end

%% 语音段自动分割
min_seg_len = round(fs * cfg.min_seg_len_ms / 1000);
segments = [];% 语音段存储矩阵
start_idx = 0;% 语音段起始标记

for i = 1:total_samples
    if audio_energy(i) > cfg.energy_thresh && start_idx == 0
        start_idx = i;
    elseif audio_energy(i) <= cfg.energy_thresh && start_idx ~= 0
        end_idx = i;
        if (end_idx - start_idx) > min_seg_len
            segments = [segments; start_idx, end_idx];
        end
        start_idx = 0;
    end
end

if start_idx ~= 0 && (total_samples - start_idx) > min_seg_len
    segments = [segments; start_idx, total_samples];
end

% 微调阈值以匹配目标段数
num_seg = size(segments, 1);
max_adjust_times = 3;% 最大微调次数
adjust_count = 0;

while num_seg ~= cfg.target_seg_num && adjust_count < max_adjust_times
    adjust_count = adjust_count + 1;
    if num_seg < cfg.target_seg_num
        cfg.energy_thresh = cfg.energy_thresh * 0.9;
        fprintf(' 第%d次微调: 段数不足 (%d段), 能量阈值降至%.3f\n', adjust_count, num_seg, cfg.
            energy_thresh);
    else
        cfg.energy_thresh = cfg.energy_thresh * 1.1;
        fprintf(' 第%d次微调: 段数过多 (%d段), 能量阈值升至%.3f\n', adjust_count, num_seg, cfg.
            energy_thresh);
    end
end

```

```

segments = [];
start_idx = 0;
for i = 1:total_samples
    if audio_energy(i) > cfg.energy_thresh && start_idx == 0, start_idx = i; end
    if audio_energy(i) <= cfg.energy_thresh && start_idx ~= 0
        if (i - start_idx) > min_seg_len, segments = [segments; start_idx, i]; end
        start_idx = 0;
    end
end
if start_idx ~= 0 && (total_samples - start_idx) > min_seg_len
    segments = [segments; start_idx, total_samples];
end
num_seg = size(segments, 1);
end

% 最终段数检查
if num_seg ~= cfg.target_seg_num
    if num_seg < cfg.target_seg_num
        tip = '调小energy_thresh (如0.04)';
    else
        tip = '调大energy_thresh (如0.06)';
    end
    warning('最终分割出%d段 (目标%d段) \rightarrow %s', num_seg, cfg.target_seg_num, tip);
else
    fprintf('成功分割出%d段语音 (匹配1~10) \n', cfg.target_seg_num);
end

%% 语音段起始前移
shift_samples = round(fs * cfg.shift_time);
segments_adjusted = segments;
for i = 1:num_seg
    segments_adjusted(i, 1) = max(1, segments_adjusted(i, 1) - shift_samples);
end

%% 提取原音频所有间隔
intervals_all = [];
for i = 1:num_seg-1
    interval_between = segments_adjusted(i+1, 1) - segments_adjusted(i, 2);
    intervals_all = [intervals_all; interval_between];
end
interval_after_last = total_samples - segments_adjusted(end, 2);
intervals_all = [intervals_all; interval_after_last];

%% 拼接音频
reversed_segments = flipud(segments_adjusted);
reversed_intervals = flipud(intervals_all);

audioOut = [];
for i = 1:num_seg
    seg_audio = audioIn(reversed_segments(i, 1):reversed_segments(i, 2));
    audioOut = [audioOut; seg_audio];
    silence = zeros(reversed_intervals(i), 1);
end

```

```

audioOut = [audioOut; silence];
end

audioOut = audioOut / max(abs(audioOut)) * 0.9;

%% 保存最终音频
output_path = 'numberConverse.wav';
audiowrite(output_path, audioOut, fs);
fprintf('音频已保存: %s\n', output_path);

%% 可视化绘图
figure('Color','w','Position',[100 100 1000 700],'Name','音频反转结果可视化');

% 原始音频
subplot(2,1,1);
t_raw = (0:total_samples-1)/fs;
plot(t_raw, audioIn, 'Color', [0.2 0.4 0.6], 'LineWidth', 1); hold on;
% 标记原始段和前移后的段
for i = 1:num_seg
    plot([segments(i,1)/fs, segments(i,1)/fs], ylim, 'Color', [0.8 0.2 0.2], 'LineStyle', '--', ...
        'LineWidth', 1);
    plot([segments_adjusted(i,1)/fs, segments_adjusted(i,1)/fs], ylim, 'Color', [0.2 0.8 0.2], ...
        'LineStyle', '-', 'LineWidth', 1.5);
    text(segments_adjusted(i,1)/fs, 0.8, num2str(i), 'Color', [0.2 0.8 0.2], 'FontSize', 9, ...
        'FontWeight', 'bold');
end
title('原始音频', 'FontSize', 12, 'FontWeight', 'bold');
xlabel('时间 (秒)', 'FontSize', 10); ylabel('幅值', 'FontSize', 10);
xlim([0, total_duration]); ylim([-1.1, 1.1]);
grid on;
ax1 = gca;
ax1.LineWidth = 0.5;
box on;
legend('原始音频','原段起始','前移后段起始','Location','best','FontSize',9);

% 反转音频
subplot(2,1,2);
t_rev = (0:length(audioOut)-1)/fs;
plot(t_rev, audioOut, 'Color', [0.8 0.4 0.2], 'LineWidth', 1); hold on;
rev_seg_times = [];
current_pos = 1;
for i = 1:num_seg
    seg_len = reversed_segments(i,2) - reversed_segments(i,1) + 1;
    seg_start = (current_pos - 1)/fs;
    seg_end = (current_pos + seg_len - 1)/fs;
    rev_seg_times = [rev_seg_times; seg_start, seg_end];
    plot([seg_start, seg_start], ylim, 'Color', [0.2 0.4 0.6], 'LineStyle', '--', 'LineWidth', 1);
    text((seg_start+seg_end)/2, 0.8, num2str(11-i), 'Color', [0.2 0.4 0.6], 'FontSize', 9, 'FontWeight ...
        ', 'bold');
    current_pos = current_pos + seg_len + reversed_intervals(i);
end
title('反转后音频', 'FontSize', 12, 'FontWeight', 'bold');

```

```

xlabel('时间 (秒)', 'FontSize', 10); ylabel('幅值', 'FontSize', 10);
xlim([0, t_rev(end)]); ylim([-1.1, 1.1]);
grid on;
ax2 = gca;
ax2.LineWidth = 0.5;
box on;

set(gcf, 'DefaultAxesFontName', 'SimHei');
set(gcf, 'DefaultTextFontName', 'SimHei');

saveas(gcf, 'numberConverse_visualization.png');

```

运行结果

程序运行结果如下图11.1

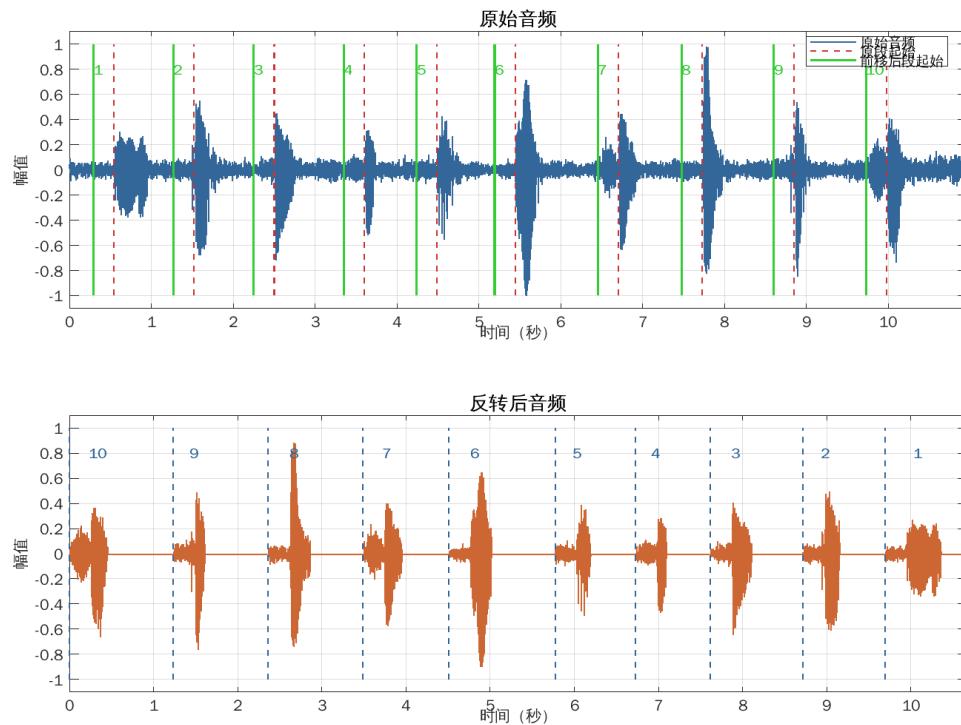


图 11.1. 语音段分割与反转拼接可视化结果

图中标注了代码语音分割后每一段音频以及对应音频段所读的数字。

输出音频保存在 ‘./matlab/countConverse/numberConverse.wav’

同时，另有一份代码和音频展示直接倒放音频的结果作为对比，保存在 ‘./matlab/countConverse/directConverse.m’ 和 ‘./matlab/countConverse/directConverse.wav’ 中。其原理为直接反转离散音频序列并输出，此处不再赘述。