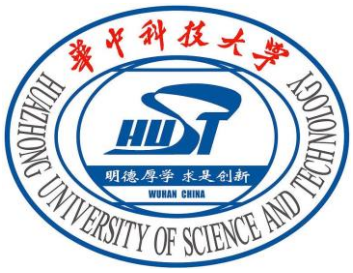


华中科技大学
生物医学数字信号处理实验报告

学院	工程科学学院
班级	工程科学学院（生医）1701 班
姓名	汪能志
学号	U201713082
指导老师	全廷伟

2020 年 9 月 1 日

目录

1. 卷积与稀疏重建

1.1. 实验任务

信号的产生：利用一连串的脉冲序列模拟动作电位，和指数下降函数进行卷积后得到钙信号，并对得到的钙信号添加高斯白噪音。

信号的重建：利用稀疏逆卷积，从钙信号中重建出动作电位。

1.2. 实验原理

1.2.1. 一维卷积

一维卷积的公式为：

$$y[s]=x[s]*h[s]=\sum_{k=-\infty}^{+\infty}x[k]\cdot h[s-k] \tag{1-1}$$

其也可以写成线性方程组（矩阵乘法形式）如下：

$$y=\begin{pmatrix}y_1\\y_2\\\vdots\\y_{m+n-1}\end{pmatrix}_{(m+n-1,1)}=A_h\cdot x=\begin{pmatrix}h_1&0&\cdots&0&0\\h_2&h_1&\cdots&0&0\\h_3&h_2&\cdots&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots\\h_n&h_{n-1}&\cdots&\cdots&\cdots\\0&h_n&\cdots&\cdots&\cdots\\\vdots&\vdots&\vdots&\vdots&\vdots\\0&0&\cdots&h_n&h_{n-1}\\0&0&\cdots&0&h_n\end{pmatrix}_{(m+n-1,m)}\cdot\begin{pmatrix}x_1\\x_2\\\vdots\\x_m\end{pmatrix}_{(m,1)} \tag{1-2}$$

1.2.2. 稀疏逆卷积（稀疏重建）

稀疏重建即为求解如下的优化问题：

$$X^*=\arg\min_X(\|Y-A\cdot X\|_2^2+\lambda\|X\|_1) \tag{1-3}$$

λ越大，则稀疏性的权重越大；λ越小，则准确性的权重越大。

记式 3 的最优解为：

$$X^*=[x_1,x_2,\cdots]^T \tag{1-4}$$

以式 1-4 为基础，对式 1-3 进行一定的修改：

$$\begin{cases}\Lambda=\lambda\cdot[\dfrac{1}{|x_1|+\iota},\dfrac{1}{|x_2|+\iota},\cdots]^T=[\lambda_1^{'},\lambda_2^{'},\cdots]^T\\X^*=\arg\min_X(\|Y-A\cdot X\|_2^2+\|\Lambda^T\cdot X\|_1)\end{cases} \tag{1-5}$$

其中ι为一个非常小的常数，其目的是避免出现分母为 0 的情况。

由此可知，X^*中的元素越小，则其在计算稀疏性时的权重越大。因此式 1-5 的解比式 1-4 的解更具有稀疏性。

则有迭代公式：

$$X^{[k+1]}=G(X^{[k]})\Leftrightarrow\begin{cases}\Lambda^{[k]}=\lambda\cdot[\dfrac{\alpha}{|x_1^{[k]}|+\iota},\dfrac{\alpha}{|x_2^{[k]}|+\iota},\cdots]^T=[\lambda_1^{[k]},\lambda_2^{[k]},\cdots]^T\\ X^{[k+1]}=\arg\min_X(\|Y-A\cdot X\|_2^2+\|(\Lambda^{[k]})^T\cdot X\|_1)\end{cases}\tag{1-6}$$

同时，在Λ中所有元素的值相同时，式 1-6 退化为式 1-3。

式 1-6 的解为：

$$\begin{aligned}&X^{[k+1]*}=X^{[k]}-\delta\cdot[A^T(A\cdot X^{[k]}-Y)]=X^{[k]}-\delta\cdot M[\tilde{h}*(X^{[k]}*h-Y)]\\&x_i^{[k+1]}=\begin{cases}\text{sign}(x_i^{[k+1]*})\cdot(|x_i^{[k+1]*}|-\lambda_i^{'}) & |x_i^{[k+1]*}|\geq\lambda_i^{'}\\0 & |x_i^{[k+1]*}|<\lambda_i^{'}\end{cases}\\&\Lambda=\lambda\cdot[\frac{1}{|x_1|+\iota},\frac{1}{|x_2|+\iota},\cdots]^T=[\lambda_1^{'},\lambda_2^{'},\cdots]^T\end{aligned}\tag{1-7}$$

可知，从初值$X^{[0]}$开始，通过式 1-7 进行迭代，$X^{[k]}$的稀疏性会逐渐增加，最终得到一个足够稀疏的解。在刚开始迭代时，可以使用等权重的Λ计算范数。初值$X^{[0]}$可以使用全零进行初始化。

1.3. MATLAB 程序实现

1.3.1. 稀疏重建

```
function x = deconv_L1(y, h, lambda)
% 初始化
% x, y & h should be column vector
lambda = 1e-3 * lambda;
len_y = size(y, 1);
len_h = size(h, 1);
len_x = len_y - len_h + 1;
x = zeros(len_x, 1);
% 反转卷积模板
h_tilde = flipud(h);
% 初始化权重和迭代参数
weight = lambda * ones(len_x, 1);
lr = 0.01;
epoch_1 = 10;
epoch_2 = 100;

for i_1 = 1:epoch_1
    for i_2 = 1:epoch_2
        % 梯度下降和软阈值迭代

        % 快速逆卷积，计算梯度
        grad = conv(h_tilde, (conv(x, h, 'full') - y), 'full');
        grad = grad(len_h:len_y);
        descent = lr .* grad;
        x = x - descent;
```

```
        % 软阈值操作并引入非负性
        x = (abs(x) > weight) .* (abs(x) - weight) .* sign(x);
        x = max(x, 0);
    end
    % 根据梯度下降结果，更新权重
    weight = lambda .* (1 ./ (x + 1e-6));
end
end
```

1.3.2. 高斯噪声

MATLAB 中的 randn 函数可以生成符合标准正态分布（高斯分布）的矩阵。输入参数 SNR 代表信号的信噪比，单位为分贝（dB）。

```
function Signal_Noise = Add_Noise(Signal, SNR)
% 计算信号功率
Signal_Power = sum(abs(Signal(:)).^2) / numel(Signal);
Signal_dB = 10 * log10(Signal_Power);

% 计算噪声功率
Noise_dB = Signal_dB - SNR;
Noise_Power = 10^(Noise_dB / 10);

% 生成噪声信号
Rand_Noise = randn(size(Signal));
Noise = sqrt(Noise_Power) * Rand_Noise;

Signal_Noise = Signal + Noise;
end
```

1.3.3. 实验

```
%%
% 信号的产生
% 动作电位脉冲
x = zeros(1024, 1);
ap_train = [200, 220, 250, 270, 300, 400, 500, 720, 600, 800, 900];
x(ap_train, 1) = 1;
% 指数下降模板
t = 15;
h = exp(-(0:1:ceil(10 * t)) / t)';
% 无噪声的钙信号
y = conv(x, h, 'full');
% 信噪比 dB
SNR = 15;
% 有噪声的钙信号
y_noise = Add_Noise(y, SNR);
```

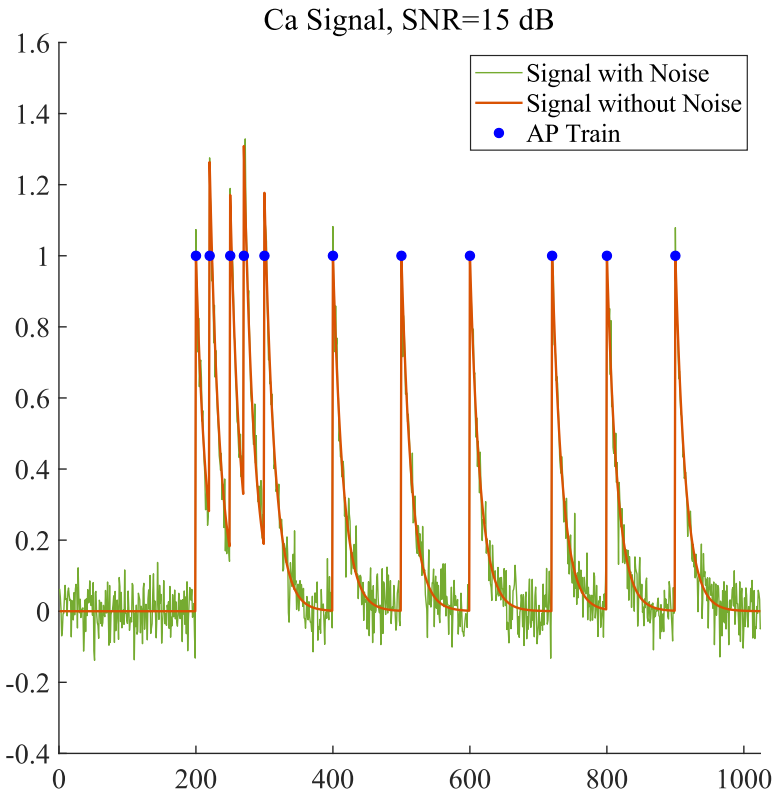
```
% 稀疏重建
% 正则化系数（*1e-3)
L = 0.5;
x_deconv_L1 = deconv_L1(y_noise, h, L);

%%
% 绘图
figure(1)
plot(y_noise, '-', 'color', [0.46, 0.67, 0.19])
hold on
plot(y, '-', 'LineWidth', 1.5, 'color', [0.85, 0.32, 0.01]);
plot(ap_train, 1, '.b', 'MarkerSize', 20);
hold off
title(sprintf('Ca Signal, SNR=%d dB', SNR))
legend('Signal with Noise', 'Signal without Noise', 'AP Train')
xlim([0, 1024])
ylim([-0.4, 1.6])
box off

figure(2)
plot(x_deconv_L1, '-r');
hold on
plot(ap_train, 1, '.b', 'MarkerSize', 20);
legend('Sparse Reconstruction', 'AP Train')
hold off
title(sprintf('Sparse Reconstruction, SNR=%d dB, λ=%.5f', SNR, L * 1e-3))
xlim([0, 1024])
ylim([-0.02, 1.3])
box off
```

1.4. 实验结果

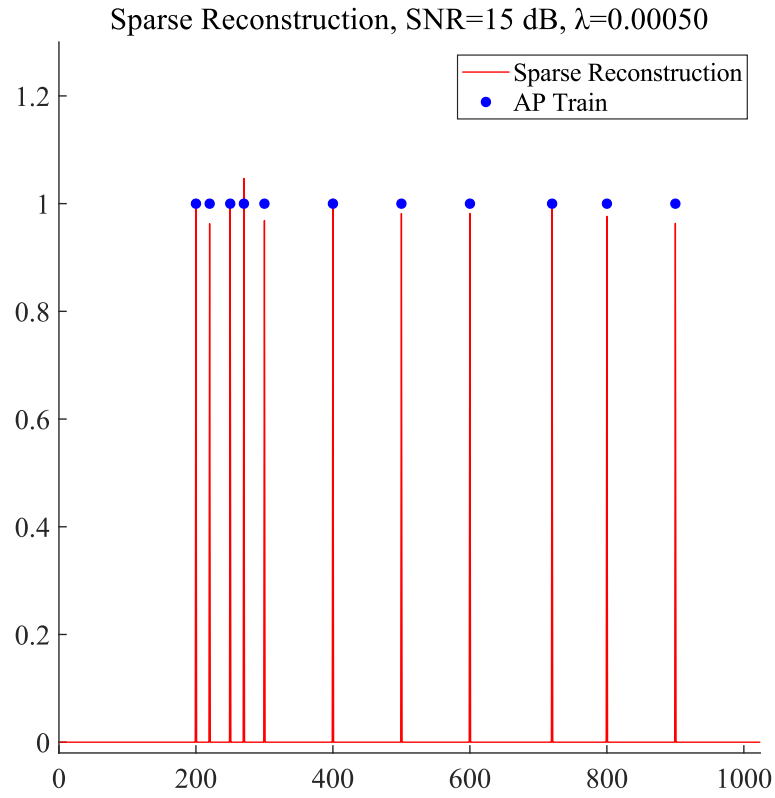
1.4.1. 钙信号生成和稀疏重建

在 200，220，250，270，300，400，500，600，720，800 和 900 处共有 11 次动作电位。指数下降模板的时间常数为 15。和动作电位信号卷积后得到钙信号，并添加信噪比为 15dB 的高斯噪声。

随后对带有噪声的钙信号进行稀疏重建。

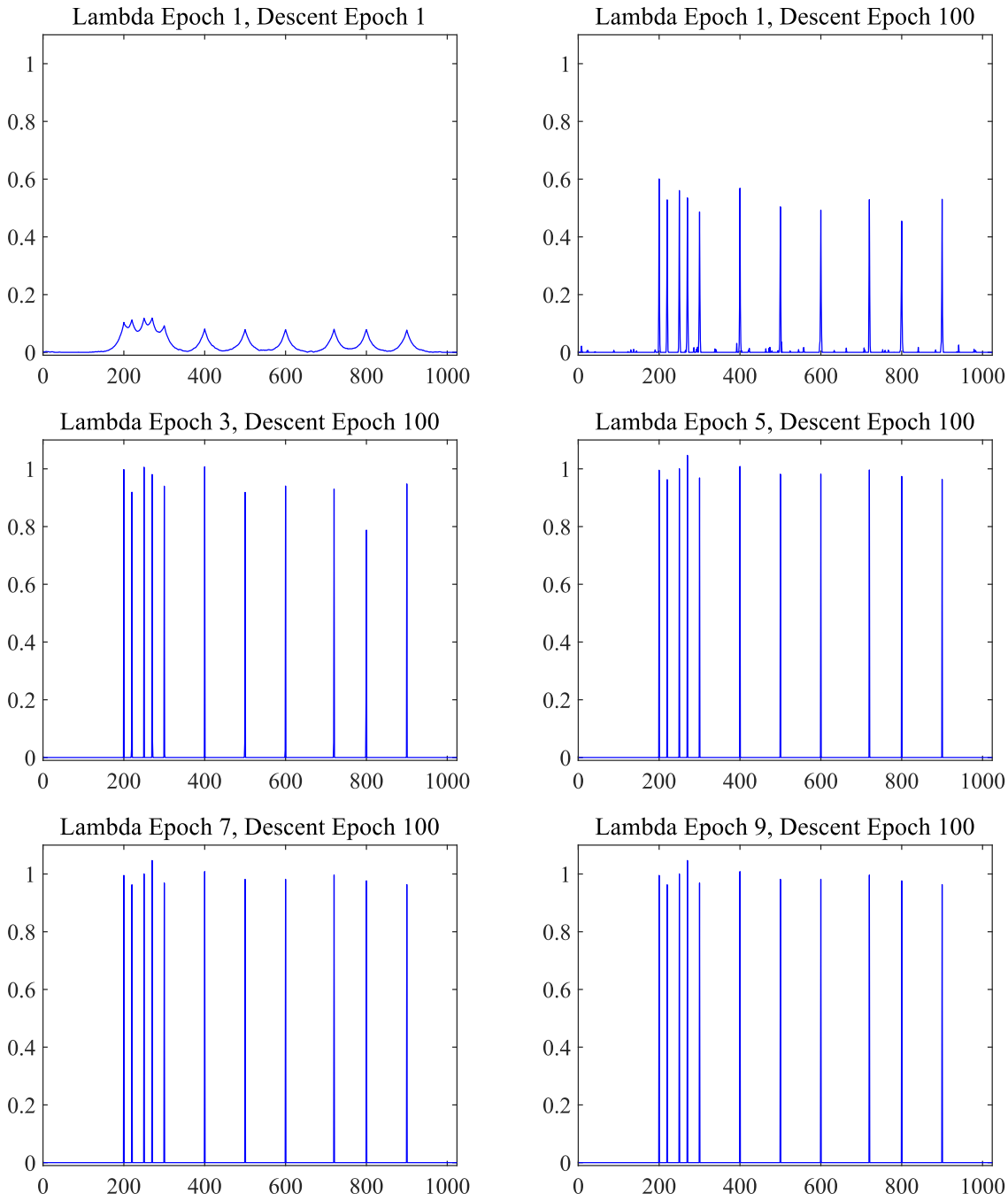


信噪比为 15dB 的钙信号



信噪比为 15dB，正则化系数为5×10^{-4}时的稀疏重建结果

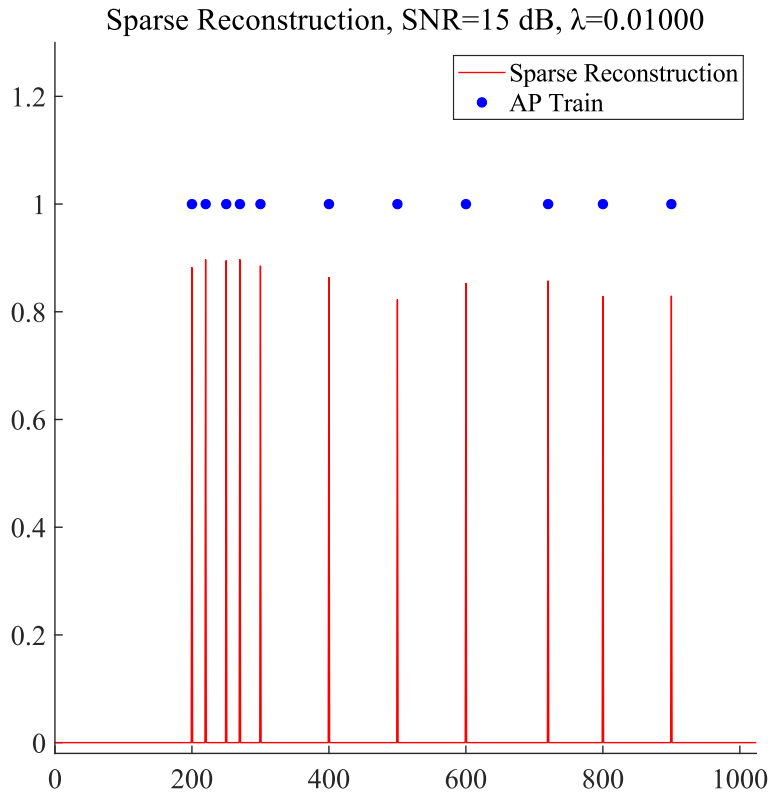
1.4.2. 迭代过程中重建信号的变化



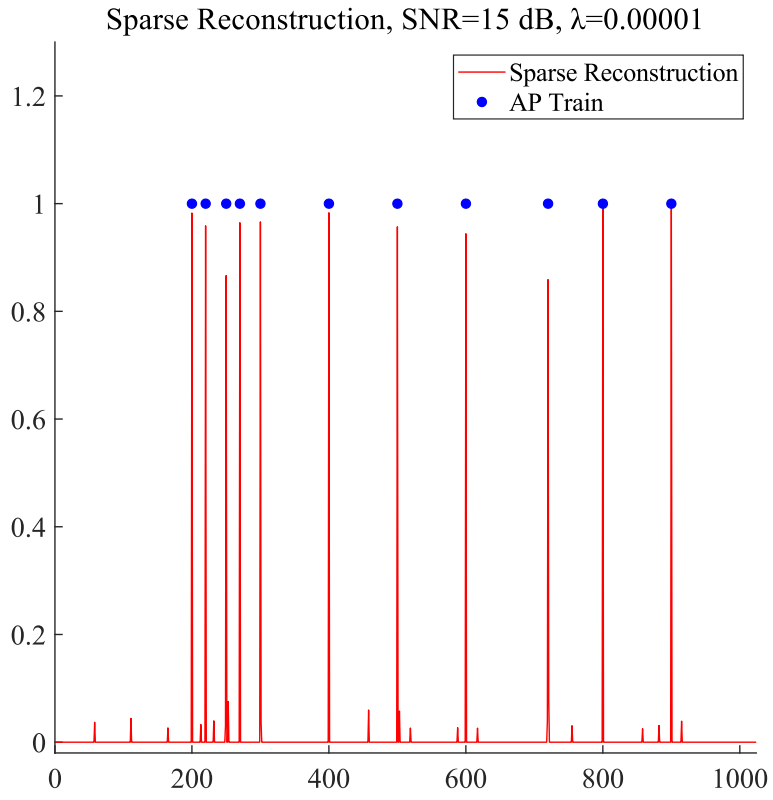
可见，在迭代刚开始时，动作电位序列就已经可以较为明显地被分辨出来。在根据梯度下降的结果调整范数权重后，动作电位序列会更明显地凸显出来，而因为噪声导致的信号会被抑制。

1.4.3. 正则化系数对重建结果的影响

保持钙信号的信噪比为 15dB，迭代次数和梯度下降率均保持不变，调整重建时的正则化系数。



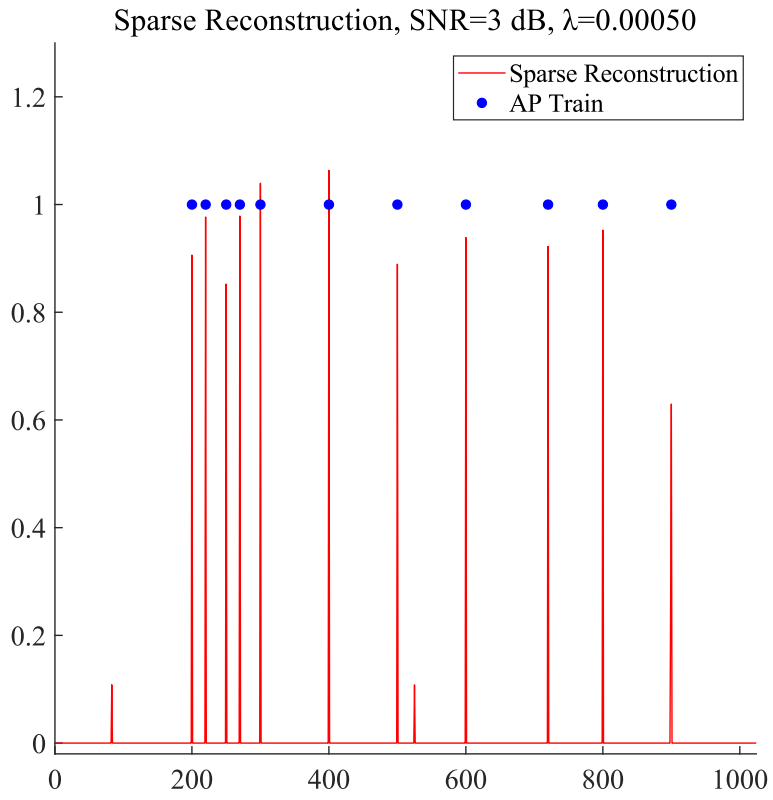
信噪比为 15dB，正则化系数为1×10^{-2}时的稀疏重建结果



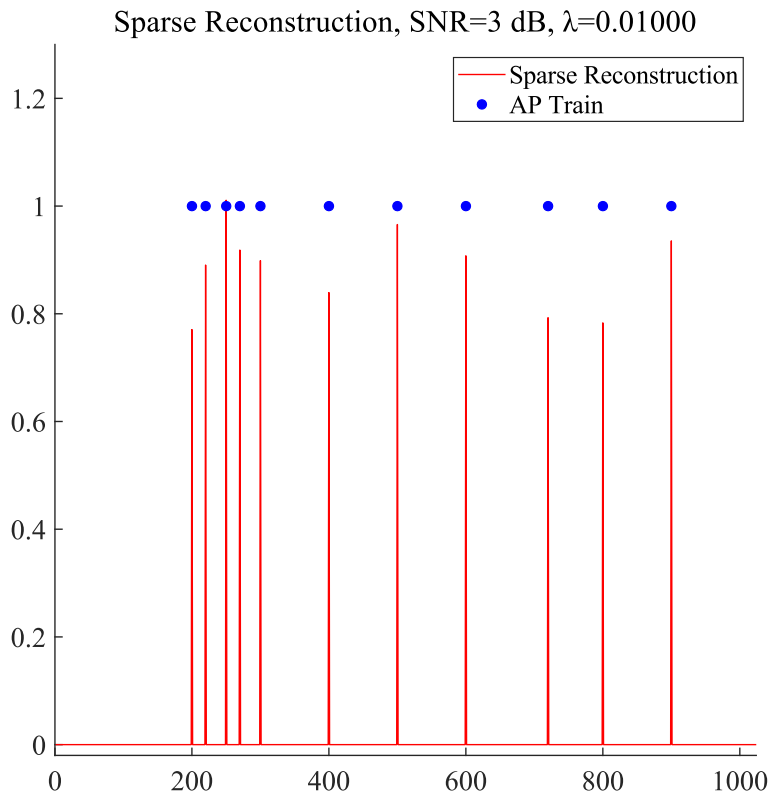
信噪比为 15dB，正则化系数为1×10^{-5}时的稀疏重建结果

可见，在提高正则化系数（即提高稀疏性时），计算得到的动作电位值会下降。在继续提高正则化系数时，甚至会重建出全零的信号。而在降低正则化系数时，会导致噪声信号不能被很好地抑制。

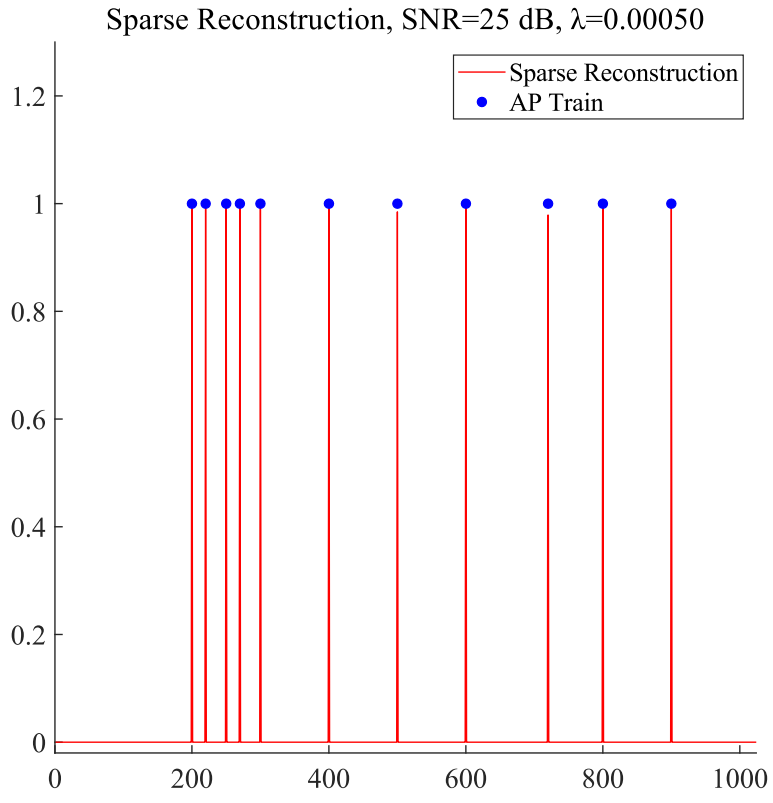
1.4.4. 信噪比对重建结果影响



信噪比为 3dB，正则化系数为5×10^{-4}时的稀疏重建结果



信噪比为 3dB，正则化系数为1×10^{-2}时的稀疏重建结果



信噪比为 25dB，正则化系数为5×10^{-4}时的稀疏重建结果

将钙信号的信噪比降低到 3dB，仍然能较好地重建动作电位，但是动作电位的幅度波动较大，而且噪声信号并没有被很好地抑制。在提高正则化系数后，噪声信号得到了较好地抑制，但是动作电位幅度仍然波动较大，由于已知原始信号是二值信号，可以考虑对重建结果进行二值化来获得较好的重建结果。

将信噪比提高到 25dB 时，此时噪声已经非常小了，重建效果已经非常接近原始数据了。

1.5. 实验总结

本次实验的主要内容是利用卷积模板生成钙信号，并对带有噪声的钙信号进行稀疏重建。生成信号的任务较为简单，

由于在之前的实验阶段，已经对二维图像的稀疏逆卷积操作有所了解，这次的稀疏重建算法也在一定程度上参考了之前实验的代码。同时对于计算过程进行了一定的修改，使其更简洁。较为复杂的部分是稀疏重建的超参数选择，如梯度下降率和正则化系数。需要多次寻找，才能找到一组合适的超参数，获得较好的重建结果。

2. 傅里叶级数与傅里叶变换

2.1. 实验目的

用傅里叶级数逼近方波信号；
利用傅里叶变换从钙信号中重建出动作电位序列；
理解并实现 fft，ifft，fft2 和 ifft2 这四个函数的功能。

2.2. 方波及其傅里叶级数逼近

2.2.1. 实验原理

对于离散信号，可以使用累加的方法来近似计算积分：

$$\int_a^b f(x)\mathrm{d}x = \lim_{n\to\infty}\sum_{i=0}^{n}\frac{b-a}{n}f(a+\frac{b-a}{n}i) \tag{2-1}$$

则，对于傅里叶级数：

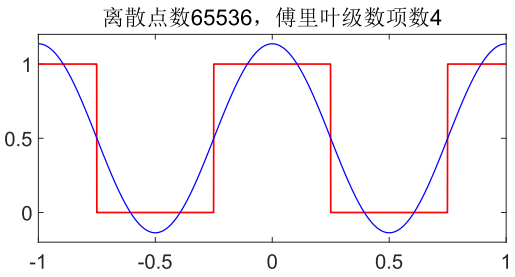
$$\hat{f}(k) = \frac{1}{T}\int_T f(t)e^{-jk\omega}\mathrm{d}t = \lim_{n\to\infty}\sum_{i=0}^{n}\frac{1}{n}f(\frac{T}{N}i)e^{-jk\omega\frac{T}{n}i} \tag{2-2}$$

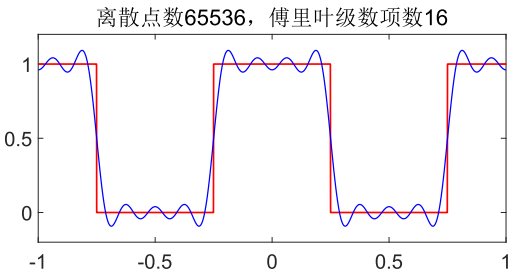
2.2.2. MATLAB 实现

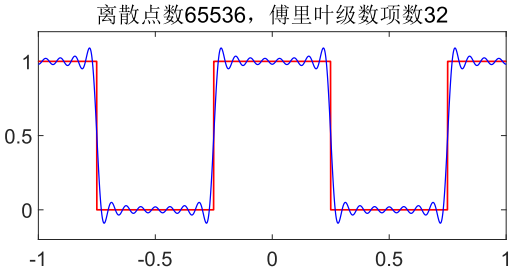
```
T = 1;                                  % 周期
w = (2 * pi) / T;
num_sample = 65536;                      % 采样点数
t = linspace(-1, 1, num_sample);
x = double(cos(w * t) >= 0);

num_k = 128;
% 计算傅里叶级数的正交基
base = Get_Base(num_k, t, T);
% 计算傅里叶级数的系数
spec = FS(x, base, T);
% 计算重建信号
recover_x = IFS(spec, base, T);
figure(1)
plot(t, x, '-r', 'LineWidth', 2)
hold on
plot(t, recover_x, '-b', 'LineWidth', 1.5)
hold off
ylim([-0.2 1.2])
title_str = sprintf('离散点数%d，傅里叶级数项数%d', num_sample, num_k);
title(title_str)

function fourier_series = FS(f, base, T)
% 计算傅里叶级数
```

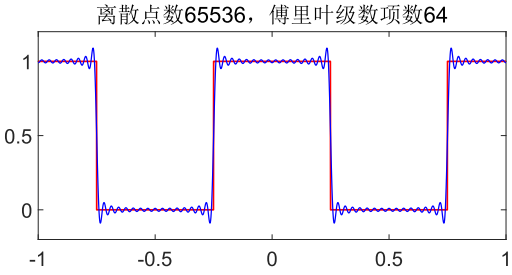
```
fourier_series = Approx_Inner(f, base, T) ./ sqrt(T);
end

function recover_signal = IFS(fourier_series, base, T)
% 根据傅里叶级数逼近原信号
recover_signal = real(fourier_series * base .* sqrt(T));
end

function base = Get_Base(num_k, t, T)
% 计算正交基
K = -ceil(num_k / 2):1:ceil(num_k / 2);
base = exp(1j .* 2 .* pi .* (1 / T) .* Outer(K, t)) ./ sqrt(T);
end

function z = Approx_Inner(f, g, T)
% 对于给定的函数（值）f 和 g，以及采样点序列 T，近似计算 f 和 g 的内积
num_sample = numel(f);
z = (f * conj(g')) .* (T ./ num_sample);
end

function z = Outer(a, b)
% a = [a1, …, am] and b = [b1, …, bn]
% result = [
% a1 * b1, a1 * b2, …, a1 * bn;
% a2 * b1, a2 * b2 ,…, a2 * bn;
% …
% am * b1, am * b2, …, am * bn;
% ]
% 效果和 numpy.outer 相同
M = numel(a);
N = numel(b);
z = zeros(M, N);
for i = 1:M
    for j = 1:N
        z(i, j) = a(i) .* b(j);
    end
end
end
```

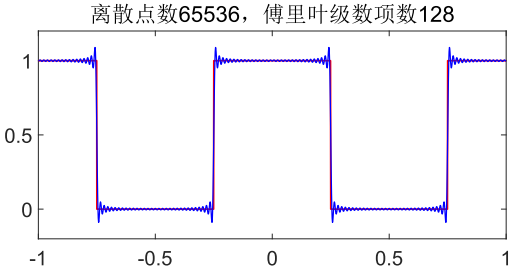
2.2.3. 实验结果

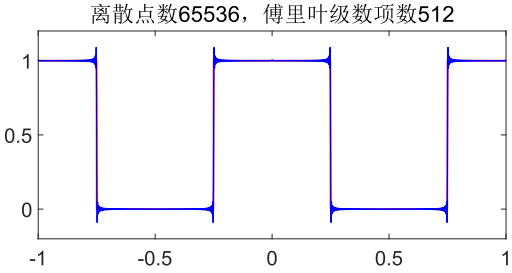

离散点数65536，傅里叶级数项数4

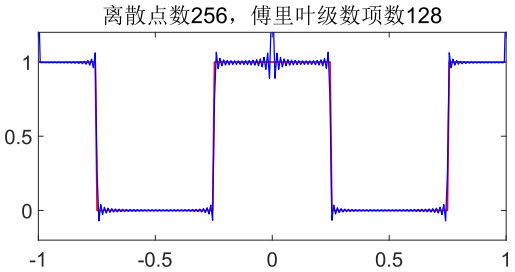

离散点数65536，傅里叶级数项数16

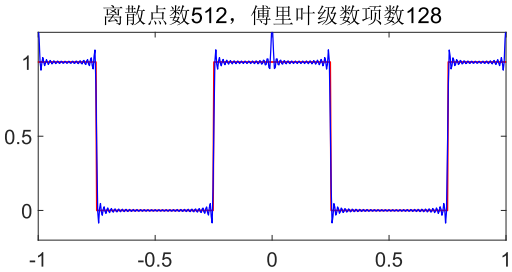

离散点数65536，傅里叶级数项数32

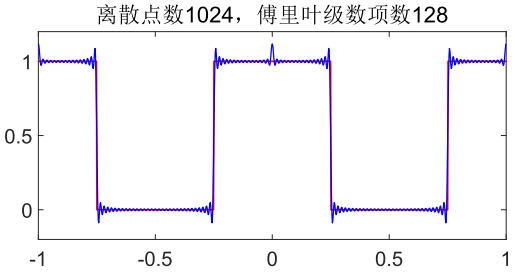

离散点数65536，傅里叶级数项数64

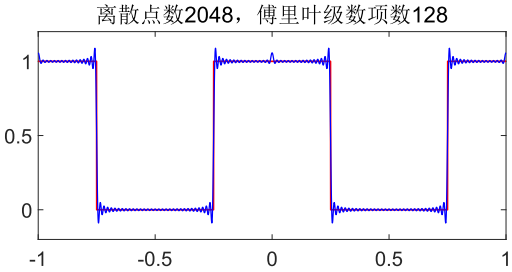

离散点数65536，傅里叶级数项数128

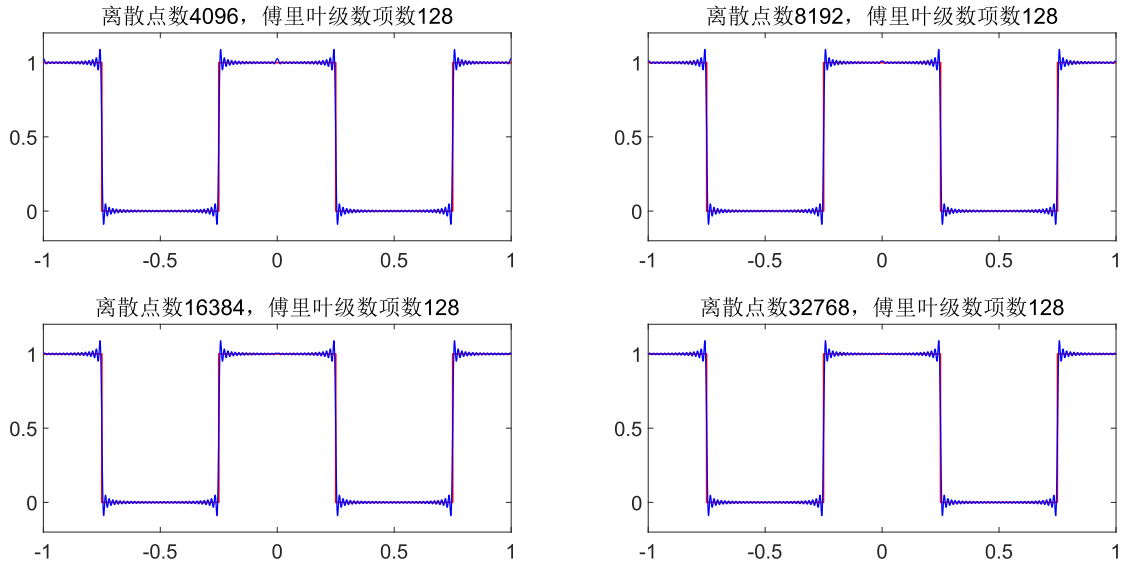

离散点数65536，傅里叶级数项数512


离散点数256，傅里叶级数项数128


离散点数512，傅里叶级数项数128


离散点数1024，傅里叶级数项数128


离散点数2048，傅里叶级数项数128



很容易发现，在保持离散点数不变的情况下，随着傅里叶级数的项数的增加，级数逼近的结果越来越接近方波。但是，由于需要拟合的信号中存在阶跃，因此吉布斯现象永远存在。

由于实验中是使用累加求和的方式近似计算积分，因此离散点越多，则级数逼近的信号越接近方波。在离散点数较少时，在非信号的非阶跃处也出现了类似吉布斯现象的情况。

2.3. 傅里叶变换与卷积

2.3.1. 实验原理

定义$\otimes$为循环卷积，$\hat{x}$、$\hat{h}$和$\hat{y}$为补零到相同长度后的信号。则循环卷积有以下性质：

$$\hat{y} = \hat{x} \otimes \hat{h} \Leftrightarrow \mathrm{DFT}(\hat{y}) = \mathrm{DFT}(\hat{x}) \times \mathrm{DFT}(\hat{h}) \tag{2-3}$$

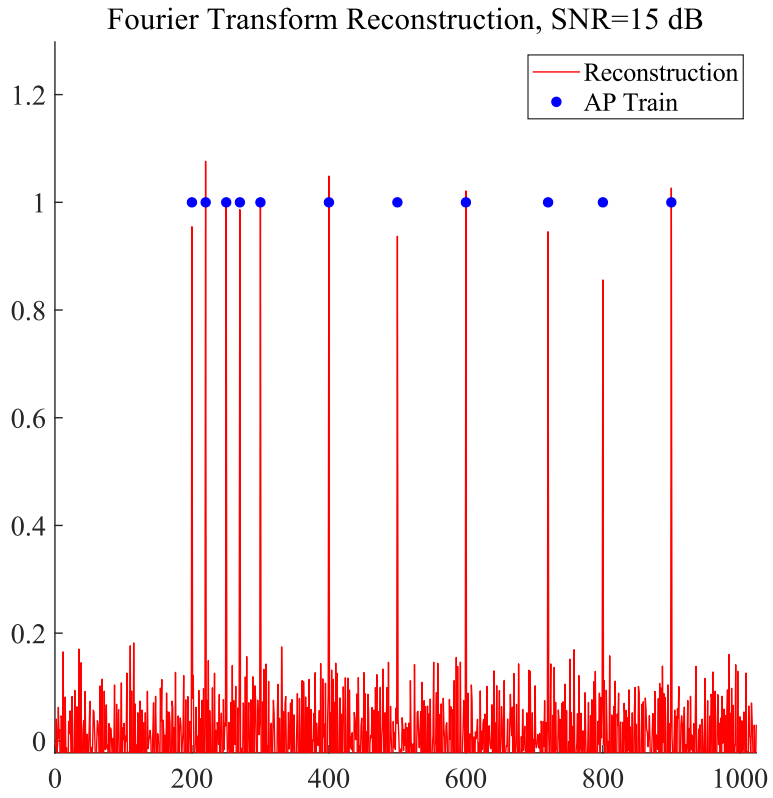
该性质也可用于计算反卷积。

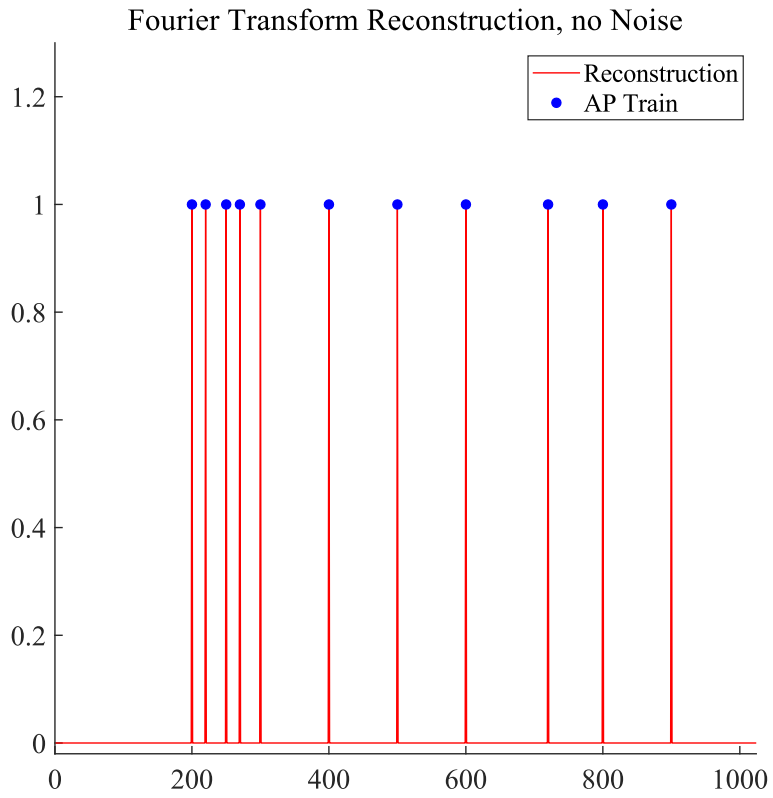
2.3.2. MATLAB 实现

```
% 信号的产生
% 动作电位脉冲
x = zeros(1024, 1);
ap_train = [200, 220, 250, 270, 300, 400, 500, 720, 600, 800, 900];
x(ap_train, 1) = 1;
% 指数下降模板
t = 15;
h = exp(-(0:1:ceil(10 * t)) / t)';
% 无噪声的钙信号
y = conv_ft(x, h);
% 信噪比 dB
SNR = 15;
% 有噪声的钙信号
y_noise = awgn(y, SNR, 'measured');
%%
x_deconv = deconv_dft(y_noise, h);
psnr(y_noise, y)
```

```
psnr(x_deconv, x)

figure(2)
plot(x_deconv, '-r');
hold on
plot(ap_train, 1, '.b', 'MarkerSize', 20);
legend('Reconstruction', 'AP Train')
hold off
title('Fourier Transform Reconstruction, no Noise')
xlim([0, 1024])
ylim([-0.02, 1.3])
box off

function y = conv_ft(x, h)
% 傅里叶变换卷积
len_x = size(x, 1);
len_h = size(h, 1);
len_y = len_x + len_h - 1;

% 对卷积模板和信号进行补零
x_circle = zeros(len_y, 1);
x_circle(1:len_x) = x;
h_circle = zeros(len_y, 1);
h_circle(1:len_h) = h;

Fy = fft(x_circle) .* fft(h_circle);
y = real(ifft(Fy));
end

function x = deconv_dft(y, h)
% 傅里叶变换反卷积
len_y = size(y, 1);
len_h = size(h, 1);
len_x = len_y - len_h + 1;

% 对卷积模板进行补零
h_circle = zeros(len_y, 1);
h_circle(1:len_h) = h;

Fx = fft(y) ./ fft(h_circle);
x = real(ifft(Fx));
x = x(1:len_x);
end
```

2.3.3. 实验结果


Fourier Transform Reconstruction, SNR=15 dB
Reconstruction
AP Train
1.2
1
0.8
0.6
0.4
0.2
0
0
200
400
600
800
1000


Fourier Transform Reconstruction, no Noise
Reconstruction
AP Train
1.2
1
0.8
0.6
0.4
0.2
0
0
200
400
600
800
1000

对于无噪声的信号，通过傅里叶变换和循环卷积可以重建出误差很小的信号。

但是，对于带有噪声的钙信号，由于叠加的噪声并不是周期性信号，傅里叶变换无法对其进行有效的降噪。由于叠加的是白噪声，因此直接对频域信号进行计算时，甚至有可能增大高频部分的噪声。

2.4. 一维离散傅里叶变换及其逆变换（DFT & IDFT）

2.4.1. 实验原理

长度为N的数字信号序列的离散傅里叶变换可以定义为：

$$\begin{cases} X(k)=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi}{N}}=\sum_{n=0}^{N-1}x(n)W_N^{nk} \\ x(n)=\mathrm{IDFT}[X(k)]=\frac{1}{N}\sum_{n=0}^{N-1}X(k)e^{j\frac{2\pi}{N}nk}=\frac{1}{N}\sum_{n=0}^{N-1}X(k)W_N^{-nk} \end{cases} \quad k=0,1,2,\cdots,N-1 \tag{2-4}$$

$\vec{X}$和$\vec{x}$可以用列向量的形式表示：

$$\begin{cases} \vec{X}=[X(0),X(1),\cdots,X(N-1)]^T \\ \vec{x}=[x(0),x(1),\cdots,x(N-1)]^T \end{cases} \tag{2-5}$$

由式 2-4 定义的离散傅里叶变换及其逆变换也可以用矩阵表示为：

$$\vec{X}=W\cdot\vec{x}=\begin{bmatrix} e^{-j2\pi\frac{0\times0}{N}} & e^{-j2\pi\frac{0\times1}{N}} & e^{-j2\pi\frac{0\times2}{N}} & \cdots & e^{-j2\pi\frac{0\times(N-1)}{N}} \\ e^{-j2\pi\frac{1\times0}{N}} & e^{-j2\pi\frac{1\times1}{N}} & e^{-j2\pi\frac{1\times2}{N}} & \cdots & e^{-j2\pi\frac{1\times(N-1)}{N}} \\ e^{-j2\pi\frac{2\times0}{N}} & e^{-j2\pi\frac{2\times1}{N}} & e^{-j2\pi\frac{2\times2}{N}} & \cdots & e^{-j2\pi\frac{2\times(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{-j2\pi\frac{(N-1)\times0}{N}} & e^{-j2\pi\frac{(N-1)\times1}{N}} & e^{-j2\pi\frac{(N-1)\times2}{N}} & \cdots & e^{-j2\pi\frac{(N-1)\times(N-1)}{N}} \end{bmatrix}\cdot\vec{x} \tag{2-6}$$

$$\vec{x}=W^{-1}\cdot\vec{X}=\frac{1}{N}\begin{bmatrix} e^{j2\pi\frac{0\times0}{N}} & e^{j2\pi\frac{0\times1}{N}} & e^{j2\pi\frac{0\times2}{N}} & \cdots & e^{j2\pi\frac{0\times(N-1)}{N}} \\ e^{j2\pi\frac{1\times0}{N}} & e^{j2\pi\frac{1\times1}{N}} & e^{j2\pi\frac{1\times2}{N}} & \cdots & e^{j2\pi\frac{1\times(N-1)}{N}} \\ e^{j2\pi\frac{2\times0}{N}} & e^{j2\pi\frac{2\times1}{N}} & e^{j2\pi\frac{2\times2}{N}} & \cdots & e^{j2\pi\frac{2\times(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{j2\pi\frac{(N-1)\times0}{N}} & e^{j2\pi\frac{(N-1)\times1}{N}} & e^{j2\pi\frac{(N-1)\times2}{N}} & \cdots & e^{j2\pi\frac{(N-1)\times(N-1)}{N}} \end{bmatrix}\cdot\vec{X} \tag{2-7}$$

2.4.2. MATLAB 实现

DFT

```
function F = dft(x)
len_x = size(x, 1);
w = (0:1:len_x - 1);
```

```
W = exp(-2 .* 1i .* pi .* (w' * w) / len_x);
F = W * x;
end
```

IDFT

```
function x = idft(F)
len_x = size(F, 1);
w = (0:1:len_x - 1);
G = exp(2 .* 1i .* pi .* (w' * w) / len_x) ./ len_x;
x = G * F;
end
```

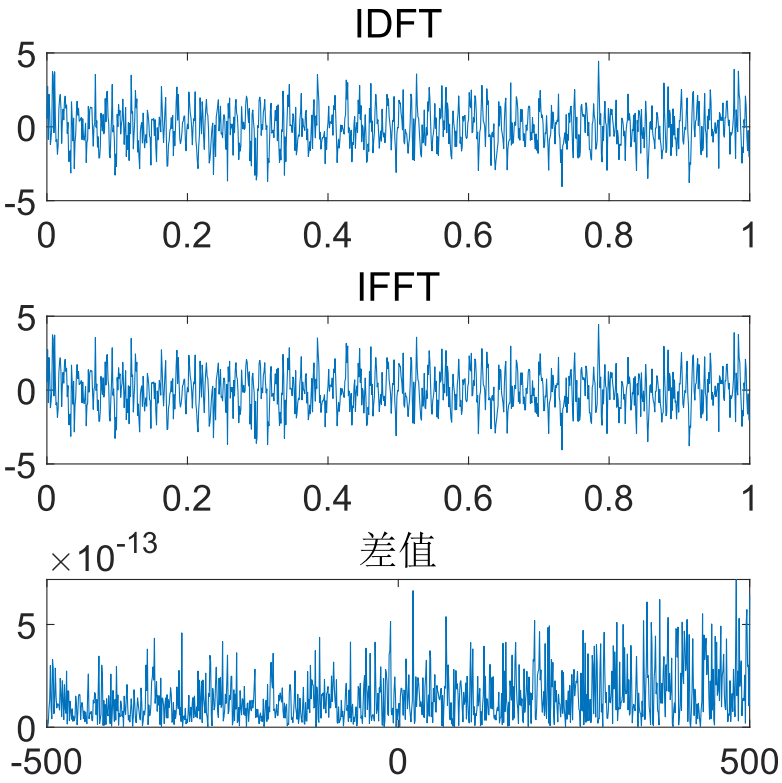
实验内容

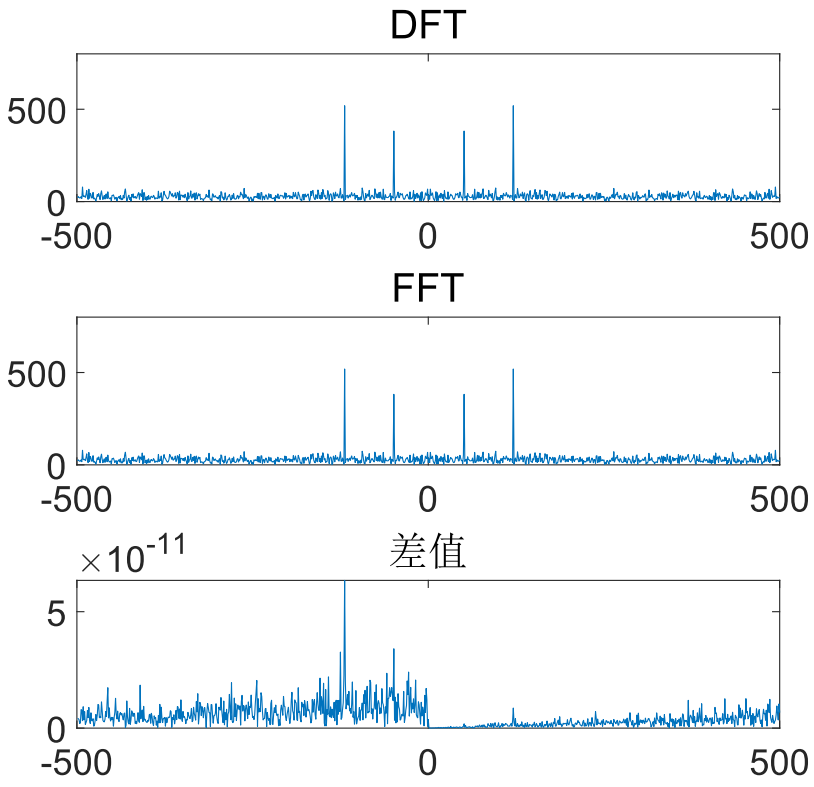
```
Fs = 1000;
T = 1 / Fs;
L = 1000;
t = (0:L - 1)' * T;

S = 0.7 * sin(2 * pi * 50 * t) + sin(2 * pi * 120 * t);

X = S + 1 * randn(size(t));

Fx_dft = dft(X);
Fx_fft = fft(X);

Fx_diff = Fx_dft - Fx_fft;

X_ifft = real(ifft(Fx_fft));
X_idft = real(idft(Fx_dft));

X_diff = (X_idft - X);

f = Fs * (1:1:L) / L - 500;

figure()
subplot(3, 1, 1)
plot(f, abs(fftshift(Fx_dft)))
ylim([0, 800])
title('DFT')
subplot(3, 1, 2)
plot(f, abs(fftshift(Fx_fft)))
ylim([0, 800])
title('FFT')
subplot(3, 1, 3)
```

```
plot(f, abs(fftshift(Fx_diff)))
title('差值')

figure()
subplot(3, 1, 1)
plot(t, X_ifft)
title('IDFT')
subplot(3, 1, 2)
plot(t, X_idft)
title('IFFT')
subplot(3, 1, 3)
plot(f, abs(X_ifft - X_idft))
title('差值')
```

2.4.3. 实验结果

测试数据为采样率 1000Hz，时间 1s（即 1000 个离散点）的数据。其中包括频率为 50Hz 和 120Hz 的正弦波和高斯噪声。对数据分别使用库函数（fft 和 ifft）以及自定义的离散傅里叶变换函数进行傅里叶变换和逆变换。





在进行离散傅里叶变换后，可以在±50Hz 和±120Hz 处看到明显的峰。库函数和自定义函数的结果的差值小于10^{-11}，属于浮点数运算误差。

2.5. 二维离散傅里叶变换及其逆变换（DFT2 & IDFT2）

2.5.1. 实验原理

类似一维离散傅里叶变换，对于$M \times N$的二维离散信号，傅里叶变换和逆变换的定义为：

$$\begin{cases} F[u,v] = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j\cdot 2\pi(\frac{ux}{M}+\frac{vy}{N})} \\ f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u,v)e^{j\cdot 2\pi(\frac{ux}{M}+\frac{vy}{N})} \end{cases} \tag{2-8}$$

和一维傅里叶变换类似，傅里叶变换及其逆变换也可以用矩阵乘法表示：

$$G_M = \begin{bmatrix} e^{-j2\pi\frac{0\times 0}{M}} & e^{-j2\pi\frac{0\times 1}{M}} & e^{-j2\pi\frac{0\times 2}{M}} & \cdots & e^{-j2\pi\frac{0\times (M-1)}{M}} \\ e^{-j2\pi\frac{1\times 0}{M}} & e^{-j2\pi\frac{1\times 1}{M}} & e^{-j2\pi\frac{1\times 2}{M}} & \cdots & e^{-j2\pi\frac{1\times (M-1)}{M}} \\ e^{-j2\pi\frac{2\times 0}{M}} & e^{-j2\pi\frac{2\times 1}{M}} & e^{-j2\pi\frac{2\times 2}{M}} & \cdots & e^{-j2\pi\frac{2\times (M-1)}{M}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{-j2\pi\frac{(M-1)\times 0}{M}} & e^{-j2\pi\frac{(M-1)\times 1}{M}} & e^{-j2\pi\frac{(M-1)\times 2}{M}} & \cdots & e^{-j2\pi\frac{(M-1)\times (M-1)}{M}} \end{bmatrix} \tag{2-9}$$

$$G_N = \begin{bmatrix} e^{-j2\pi\frac{0\times0}{N}} & e^{-j2\pi\frac{0\times1}{N}} & e^{-j2\pi\frac{0\times2}{N}} & \cdots & e^{-j2\pi\frac{0\times(N-1)}{N}} \\ e^{-j2\pi\frac{1\times0}{N}} & e^{-j2\pi\frac{1\times1}{N}} & e^{-j2\pi\frac{1\times2}{N}} & \cdots & e^{-j2\pi\frac{1\times(N-1)}{N}} \\ e^{-j2\pi\frac{2\times0}{N}} & e^{-j2\pi\frac{2\times1}{N}} & e^{-j2\pi\frac{2\times2}{N}} & \cdots & e^{-j2\pi\frac{2\times(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{-j2\pi\frac{(N-1)\times0}{N}} & e^{-j2\pi\frac{(N-1)\times1}{N}} & e^{-j2\pi\frac{(N-1)\times2}{N}} & \cdots & e^{-j2\pi\frac{(N-1)\times(N-1)}{N}} \end{bmatrix} \tag{2-10}$$

$$F = G_M \cdot f \cdot G_N \tag{2-11}$$

$$G_M^{-1} = \frac{1}{M}\begin{bmatrix} e^{j2\pi\frac{0\times0}{M}} & e^{j2\pi\frac{0\times1}{M}} & e^{j2\pi\frac{0\times2}{M}} & \cdots & e^{j2\pi\frac{0\times(M-1)}{M}} \\ e^{j2\pi\frac{1\times0}{M}} & e^{j2\pi\frac{1\times1}{M}} & e^{j2\pi\frac{1\times2}{M}} & \cdots & e^{j2\pi\frac{1\times(M-1)}{M}} \\ e^{j2\pi\frac{2\times0}{M}} & e^{j2\pi\frac{2\times1}{M}} & e^{j2\pi\frac{2\times2}{M}} & \cdots & e^{j2\pi\frac{2\times(M-1)}{M}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{j2\pi\frac{(M-1)\times0}{M}} & e^{j2\pi\frac{(M-1)\times1}{M}} & e^{j2\pi\frac{(M-1)\times2}{M}} & \cdots & e^{j2\pi\frac{(M-1)\times(M-1)}{M}} \end{bmatrix} \tag{2-12}$$

$$G_N^{-1} = \frac{1}{N}\begin{bmatrix} e^{j2\pi\frac{0\times0}{N}} & e^{j2\pi\frac{0\times1}{N}} & e^{j2\pi\frac{0\times2}{N}} & \cdots & e^{j2\pi\frac{0\times(N-1)}{N}} \\ e^{j2\pi\frac{1\times0}{N}} & e^{j2\pi\frac{1\times1}{N}} & e^{j2\pi\frac{1\times2}{N}} & \cdots & e^{j2\pi\frac{1\times(N-1)}{N}} \\ e^{j2\pi\frac{2\times0}{N}} & e^{j2\pi\frac{2\times1}{N}} & e^{j2\pi\frac{2\times2}{N}} & \cdots & e^{j2\pi\frac{2\times(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{j2\pi\frac{(N-1)\times0}{N}} & e^{j2\pi\frac{(N-1)\times1}{N}} & e^{j2\pi\frac{(N-1)\times2}{N}} & \cdots & e^{j2\pi\frac{(N-1)\times(N-1)}{N}} \end{bmatrix} \tag{2-13}$$

$$f = G_M^{-1} \cdot F \cdot G_N^{-1} \tag{2-14}$$

2.5.2. MATLAB 实现

DFT2

```
function F = dft2(x)
[row, col] = size(x);
w_row = (0:1:row - 1);
w_col = (0:1:col - 1);
Gm = exp(-2 .* 1i .* pi .* (w_row' * w_row) / row);
Gn = exp(-2 .* 1i .* pi .* (w_col' * w_col) / col);
F = Gm * x * Gn;
```

IDFT2

```
function x = idft2(F)
```

```
[row, col] = size(F);
w_row = (0:1:row - 1);
w_col = (0:1:col - 1);
Gm = exp(2 .* 1i .* pi .* (w_row' * w_row) / row) ./ row;
Gn = exp(2 .* 1i .* pi .* (w_col' * w_col) / col) ./ col;
x = Gm * F * Gn;
```

实验内容

```
img = imread('cameraman.tif');
img = im2double(img);
img = imresize(img, [512 512]);

img_dft = dft2(img);
img_idft = real(idft2(img_dft));
img_freq_log = log10(abs(fftshift(img_dft)));

img_dft_2 = fft2(img);
img_idft_2 = real(ifft2(img_dft_2));
img_freq_log_2 = log10(abs(fftshift(img_dft_2)));
img_dft_diff = (img_idft - img_idft_2);

figure(1)
subplot(1, 2, 1)
imshow(img_idft, [])
colorbar
title('IDFT')
subplot(1, 2, 2)
imshow(img_freq_log, [])
colorbar
title('DFT / log')

figure(2)
subplot(1, 2, 1)
imshow(img_idft_2, []);
colorbar
title('IFFT')
subplot(1, 2, 2)
imshow(img_freq_log_2, [])
colorbar
title('FFT / log')

figure(3)
subplot(1, 2, 1)
imshow(img_dft_diff, [])
```

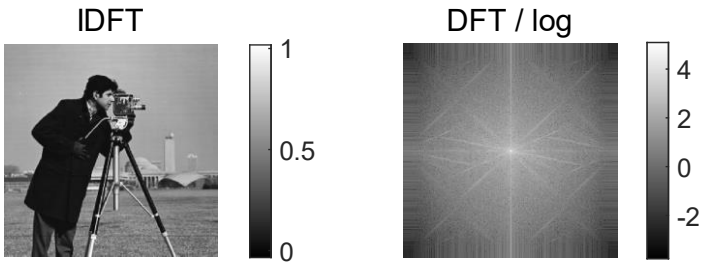
```
colorbar
title('重建图像差值')
subplot(1, 2, 2)
imshow(abs(fftshift(img_dft - img_dft_2)), [])
colorbar
title('频谱差值')
```

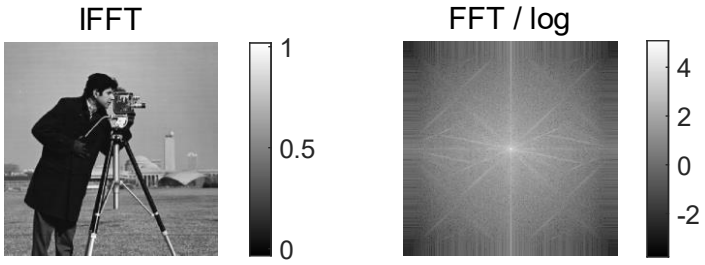
2.5.3. 实验结果

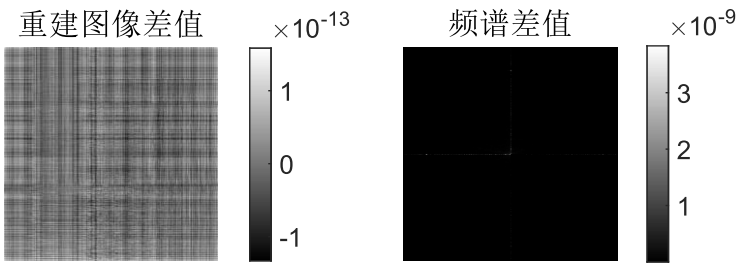
使用 MATLAB 中的图片 cameraman.tif，并将其尺寸放大到512×512，动态范围线性缩放到 0-1 进行实验。

使用 MATLAB 中的函数 immse 衡量图像之间的误差（均方误差），对于尺寸为$M \times N$的两张图片，其均方误差为：

$$\mathrm{IMMSE}(X,Y) = \frac{\| X - Y \|_2^2}{MN} \tag{2-15}$$







通过库函数（fft2 和 ifft2）计算得到的结果与通过自定义函数得到的结果之间的均方误差为 1.01×10^{-27}，可以认为是浮点数运算过程中引入的误差。

2.6. 实验总结

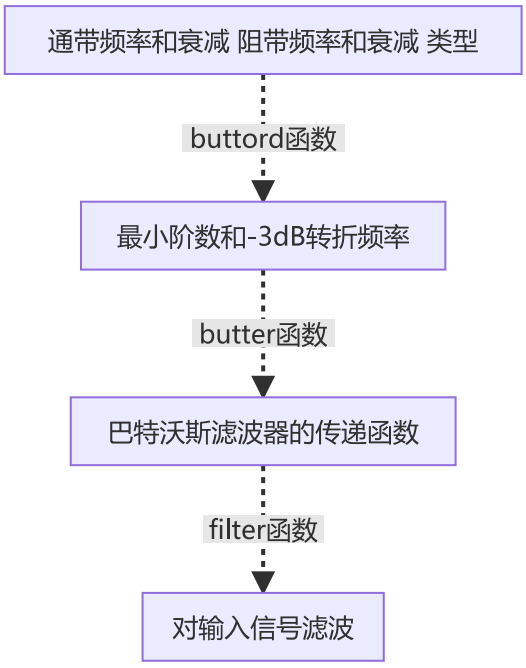
本次实验的主要内容是傅里叶级数和傅里叶变换。在之前的《信号与系统》课程中，我也接触过和离散傅里叶变换相关的内容，但是对于离散傅里叶变换的实现之前并没有过多的了解。

3. 数字滤波器

3.1. 实验原理

3.1.1. 数字滤波器的设计流程

由于模拟滤波器的设计较为成熟，因此本次实验中设计数字滤波器的方法为：先设计模拟滤波器，再将模拟滤波器的系统函数映射为数字滤波器的系统函数。



3.1.2. 模拟滤波器到数字滤波器的映射

常用的将模拟滤波器转换为数字滤波器的方法有冲激响应不变法和双线性变换法。

冲激响应不变法从单位冲激响应出发，目标是使数字滤波器的单位冲激响$h[n]$尽可能逼近模拟滤波器的单位冲激响应$h(t)$即$h[n]=h_a(nT)$。在拉普拉斯变换域和 z 变换域上，满足变换关系：

$$\begin{cases} z=e^{sT} \\ H_a(s)=\sum_{k=1}^{N}\frac{A_k}{s-s_k} \Rightarrow H(z)=\sum_{k=1}^{N}\frac{A_k}{1-e^{s_kT}z^{-1}} \end{cases} \tag{3-1}$$

在变换前后，s 平面和 z 平面不算单值对应的。在 s 平面上相距$2j\pi$的两个点，在变换后的 z 平面上是重合的。因此，冲激响应不变法会出现频率混叠现象。

从时域的角度来看，可以认为冲激响应不变法的思路是：对模拟滤波器的单位冲激响应$h(t)$以间隔T进行采样，将采样信号作为数字滤波器的单位冲激响应。此时对应的采样率为$\Omega_s=\frac{2\pi}{T}$。根据奈奎斯特采样定理，模拟滤波器的单位冲激响应必须是截止频率不大于$\Omega_h=\frac{\pi}{T}$的低通信号，否则就会出现频谱混叠的情况。

为此，需要引入双线性变换，双线性变换法得到的数字滤波器的频率响应和模拟滤波器的频率响应相似，其模拟频率Ω和数字频率ω之间是单值变换。

在 MATLAB 中，数字频率定义为归一化频率，对于以fs采样的模拟信号，有$\omega = \frac{\Omega}{\pi \cdot fs}$。

为了方便和 MATLAB 中提供的函数进行对比测试，以下讨论内容和 MATLAB 中的定义保持一致。

双线性变换法的映射关系为：

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{3-2}$$

其中T为数字信号的采样周期，即采样率的倒数。在实际计算中，T的取值不影响最终的计算结果（因为数字序列对应的频率会随着采样率的变化同步变化）。为简化计算，一般取$T = 2$。

由于其模拟频率和数字频率之间是单值映射，因此不会和冲激响应不变法一样出现频率混叠。但是，由于其变换不是线性变换，因此会出现频率畸变现象。在设计模拟滤波器式，需要引入预畸变进行补偿。

$$\Omega = \frac{2}{T} \tan \frac{\pi \omega}{2} \tag{3-3}$$

在 MATLAB 的库函数 butter.m 中，进行频率预畸变的代码如下：

```
71  % step 1: get analog, pre-warped frequencies
72  if ~analog
73      fs = 2;
74      u = 2*fs*tan(pi*Wn/fs);
75  else
76      u = Wn;
77  end
```

在将 73 行的 fs 修改为其他值后，会得到错误的结果。这与式 3-2 的结论：数字信号采样周期（采样率）不影响最终系统函数矛盾。这里应该是第 74 行代码出现错误，其正确形式应为：

```
74      u = 2*fs*tan(pi*Wn/2);
```

其将常量 2 错误写为变量 fs，因此只有在 fs=2 时，才能得到正确的结果。

3.1.3. 双线性映射

模拟巴特沃斯滤波器和数字巴特沃斯滤波器的系统函数都可以表示为零极点增益形式：

$$\begin{aligned} H(s) &= k_s \cdot \frac{\prod (s - s_{zi})}{\prod (s - s_{pi})} \\ H(z) &= k_z \cdot \frac{\prod (z - z_{zi})}{\prod (z - z_{pi})} \end{aligned} \tag{3-4}$$

在变换前后，s 平面的极点s_{pi}和 z 平面的极点z_{pi}相互对应；s 平面的零点s_{zi}和 z 平面的零点z_{zi}相互对应。因此只需要计算 s 平面的极零点和增益变换到 z 平面后的值，就可以得到数字滤波器的系统函数。

代入式 3-2，极点和零点的变换公式为：

$$z=\frac{1+\frac{T}{2}\cdot s}{1-\frac{T}{2}\cdot s}\quad z=z_{zi},z_{pi}\quad s=s_{zi},s_{pi} \tag{3-5}$$

增益的变换公式：

$$k_z=k_s\cdot\frac{\prod(\frac{2}{T}-z_{si})}{\prod(\frac{2}{T}-p_{si})} \tag{3-6}$$

式 3-6 中不包括位于∞处的零点。

3.1.4. 归一化巴特沃斯低通滤波器

模拟巴特沃斯低通滤波器是全极点系统，其幅度平方函数为：

$$|H_a(j\Omega)|^2=\frac{1}{1+(\frac{\Omega}{\Omega_c})^{2N}} \tag{3-7}$$

将其拓展到拉普拉斯变换域，有：

$$|H_a(s)|^2=H_a(s)H_a(-s)=\frac{1}{1+(\frac{s}{j\Omega_c})^{2N}} \tag{3-8}$$

式 3-8 的极点为：

$$s_k=\Omega_c e^{j(\frac{1}{2}+\frac{2k-1}{2N})\pi}\quad k=1,2,\cdots,2N \tag{3-9}$$

这些极点均匀分布在 s 平面上，半径为Ω_c的巴特沃斯圆上。且其中$H_a(s)$和$H_a(-s)$的极点两两一组，关于虚轴对称。为了使系统稳定，$H_a(s)$的极点需要全部在虚轴左侧，共有 N 个。

3.1.5. 滤波器阶数和截止频率计算

计算流程为：

1. 将数字频率预畸变为模拟频率；
2. 计算模拟滤波器的阶数和 3dB 转折频率；
3. 将模拟频率反畸变为数字频率。

对于阻带衰减R_s，通带衰减R_p，阻带频率ω_s和通带频率ω_p的数字滤波器，将数字频率畸变为对应的模拟频率Ω_s和Ω_p后，以下将讨论在低通、高通和带通的情况下的计算方法。

低通滤波器

$$\begin{cases} g=\dfrac{10^{0.1\cdot R_s-1}}{10^{0.1\cdot R_p-1}} \\ \lambda_s=\dfrac{\Omega_s}{\Omega_p} \\ N\geq\dfrac{\log(g)}{2\cdot\log(\lambda_s)} \end{cases} \tag{3-10}$$

以恰好满足通带条件和恰好满足阻带条件进行计算，可以得到两个截止频率的表达式：

$$\Omega_{c0} \geq \frac{\Omega_p}{\sqrt[2N]{10^{0.1\cdot R_p}-1}} = \Omega_{cp} \tag{3-11}$$

$$\Omega_{c0} \leq \frac{\Omega_s}{\sqrt[2N]{10^{0.1\cdot R_s}-1}} = \Omega_{cs} \tag{3-12}$$

为了获得更大的阻带衰减，一般使用$\Omega_{c0} = \Omega_{cs}$。
对于低通滤波器，有：

$$\Omega_c = \Omega_p \cdot \Omega_{c0} \tag{3-13}$$

高通滤波器

$$\begin{cases} g = \dfrac{10^{0.1\cdot R_s}-1}{10^{0.1\cdot R_p}-1} \\ \lambda_s = \dfrac{\Omega_p}{\Omega_s} \\ N \geq \dfrac{\log(g)}{2\cdot\log(\lambda_s)} \end{cases} \tag{3-14}$$

类似低通滤波器计算截止频率，有：

$$\Omega_c = \frac{\Omega_p}{\Omega_{c0}} \tag{3-15}$$

带通滤波器

$$\begin{cases} B_p = \Omega_{p2} - \Omega_{p1} \quad \Omega_{p0}^2 = \Omega_{p2}\Omega_{p1} \\ \overline{\Omega_{st1}} = \dfrac{\Omega_{s1}^2 - \Omega_{p_1}^2}{\Omega_{s1}B_p} \quad \overline{\Omega_{st2}} = \dfrac{\Omega_{s2}^2 - \Omega_{p_2}^2}{\Omega_{s2}B_p} \quad \lambda_s = \overline{\Omega_s} = \min(|\overline{\Omega_{st1}}|, |\overline{\Omega_{st2}}|) \\ g = \dfrac{10^{0.1\cdot R_s}-1}{10^{0.1\cdot R_p}-1} \\ N \geq \dfrac{\log(g)}{2\cdot\log(\lambda_s)} \end{cases} \tag{3-16}$$

$$\Omega_c = \pm\frac{\Omega_{c0}\cdot(\Omega_2 - \Omega_1)}{2} + \sqrt{\frac{\Omega_{c0}^2}{4}\cdot(\Omega_2 - \Omega_1)^2 + (\Omega_2 \cdot \Omega_1)} \tag{3-17}$$

3.1.6. 模拟滤波器去归一化

模拟巴特沃斯低通滤波器是全极点滤波器，其系统函数可以表示为如下的零极点增益形式：

$$H(s) = k \cdot \frac{1}{\prod_{i=1}^{N}(s - p_i)} \tag{3-18}$$

如果Ω_p为去归一化后的 3dB 转折频率，则对于低通、高通和带通滤波器，去归一化前后的系统函数自变量之间的变换公式为：

$$\begin{aligned}&\text{Low Pass to Low Pass:}\quad \overline{s}=\frac{s}{\Omega_p}\\&\text{Low Pass to High Pass:}\quad \overline{s}=\frac{\Omega_p}{s}\\&\text{Low Pass to Band Pass:}\quad \overline{s}=\frac{\Omega_0}{\Delta\Omega}\left(\frac{s}{\Omega_0}+\frac{\Omega_0}{s}\right)\quad \Omega_0=\sqrt{\Omega_{p1}\Omega_{p2}},\Delta\Omega=|\Omega_{p1}-\Omega_{p2}|\end{aligned}\tag{3-19}$$

对于零点、极点和增益分别为$\bar{z}$、$\bar{p}$和k的归一化巴特沃斯低通滤波器$H(\bar{s})$，其去归一化后的结果如下。

以下的变换公式仅适用于对截止频率为 1 的低通巴特沃斯滤波器进行去归一化。

去归一化到低通滤波器

$$\begin{cases}z_i=\overline{z_i}\\p_i=\Omega_p\cdot\overline{p_i}\quad i=1,2,\cdots,N\\k=\Omega_p^N\cdot\overline{k}\end{cases}\tag{3-20}$$

去归一化到高通滤波器

$$\begin{cases}z_i=\dfrac{1}{\overline{z_i}}\\p_i=\dfrac{\Omega_p}{\overline{p_i}}\quad i=1,2,\cdots,N\\k=\overline{k}\end{cases}\tag{3-21}$$

去归一化到带通滤波器

$$\begin{cases}z_i=\dfrac{1}{\overline{z_i}}\qquad i=1,2,\cdots,N\\k=\Delta\Omega^N\end{cases}\tag{3-22}$$

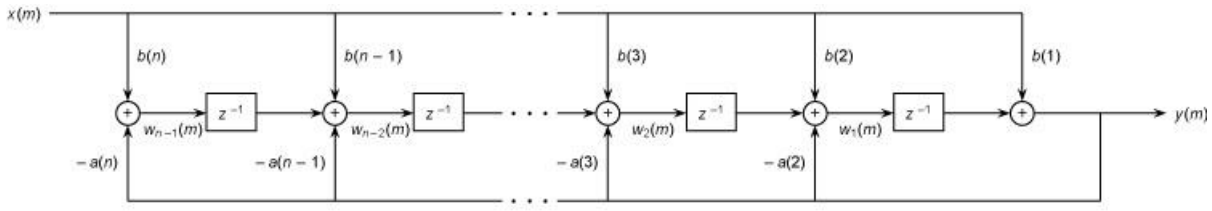
需要注意，从低通滤波器归一化到带通滤波器时，极点的数量会增加一倍。即 N 阶带通滤波器会有 2N 个极点。每一对极点都是以下方程的一组共轭解：

$$s^2-\Delta\Omega\cdot\overline{p_i}\cdot s+\Omega_{p1}\Omega_{p2}=0\tag{3-23}$$

即：

$$p_{2i-1},p_{2i}=\frac{\Delta\Omega\cdot\overline{p_i}\pm\sqrt{(\Delta\Omega\cdot\overline{p_i})^2-4\Omega_{p1}\Omega_{p2}}}{2}\tag{3-24}$$

3.1.7. 滤波器函数的实现



通过传递函数和输入信号计算输出信号的实现参考了数字信号处理教程和 MATLAB 的帮助文档中说明的方法，使用直接 II 型 IIR 滤波器的转置结构实现。

3.2. MATLAB 实现

3.2.1. 计算巴特沃斯滤波器的阶数和转折频率（buttord）

```
function [order, wc] = my_buttord(wp, ws, Rp, Rs)
% in MATLAB $ w = W / (pi * Fs) $ not $ w = W / Fs $ in textbook
% 滤波器类型
wp = sort(abs(wp));
ws = sort(abs(ws));
if numel(wp) == 1
    if wp < ws
        % 低通
        ftype = 1;
    elseif wp > ws
        % 高通
        ftype = 2;
    end
elseif numel(wp) == 2
    if wp(1) < ws(1) && wp(2) > ws(2)
        % 带阻，未完成
        ftype = 3;
    elseif wp(1) > ws(1) && wp(2) < ws(2)
        % 带通
        ftype = 4;
    end
end

% 根据数字频率计算模拟频率，进行预畸变
T = 2;
Wp = (2 / T) .* tan(pi .* wp ./ 2);
Ws = (2 / T) .* tan(pi .* ws ./ 2);
if ftype == 1
    % 低通
    Wst = Ws / Wp;
elseif ftype == 2
```

```
    % 高通
    Wst = Wp / Ws;
elseif ftype == 4
    % 带通
    Wst = (Ws.^2 - Wp(1) * Wp(2)) ./ (Ws * (Wp(1) - Wp(2)));
    Wst = min(abs(Wst));
end

% 计算最小阶数
G = (10^(0.1 * Rs) - 1) / (10^(0.1 * Rp) - 1);
order = ceil(log(G) / (2 * log(Wst)));
% 计算截止频率
W0 = Wst / ((10^(0.1 * Rs) - 1)^(1 / (2 * order)));
if ftype == 1
    Wc = Wp * W0;
elseif ftype == 2
    Wc = Wp / W0;
elseif ftype == 4
    W0 = [-W0, W0];
    W2 = Wp(2);
    W1 = Wp(1);
    Wc = W0 * (W2 - W1) / 2 + sqrt((W0.^2) / 4 * (W2 - W1)^2 + W1 * W2);
    Wc = sort(abs(Wc));
end

% 根据模拟频率计算数字频率，反畸变
wc = (2 / pi) .* atan(T .* Wc ./ 2);
end
```

3.2.2. 计算数字滤波器的传递函数（butter）

```
function [bz, az] = my_butter(N, wc, type)
%%
% 频率预畸变
T = 2;
Wc = (2 / T) .* tan(pi .* wc ./ 2);
%%
% 计算截止频率为 1 rad/s 的归一化模拟巴特沃斯滤波器的系统函数
% [zn, pn, kn]  零极点增益形式
% [bn, an]      传递函数形式
% H(s)不存在零点（在无穷大处）
zn = inf .* ones(N, 1);
% 计算归一化巴特沃斯滤波器的极点
% P 为 H(s)*H(-s)的所有极点
if mod(N, 2) == 0
```

```
    P = exp(1j .* pi .* ((1:2 * N) - 0.5) ./ N);
elseif mod(N, 2) == 1
    P = exp(1j .* pi .* (1:2 * N) ./ N);
end

% H(s)的极点都需要在左半平面
pn = P(real(P) < 0);
kn = 1;

bn = kn .* poly(zn);
an = poly(pn);
%%
% 去归一化，将归一化低通滤波器映射到低通/高通/带通滤波器
% [zs, ps, ks]  零极点增益形式
% [bs, as]      传递函数形式
if type == 'lp'
    % 模拟低通到模拟低通
    zs = zn;
    ps = Wc .* pn;
    ks = kn .* (Wc^N);
elseif type == 'hp'
    % 模拟低通到模拟高通
    zs = 1 ./ zn;
    ps = Wc ./ pn;
    ks = kn;
elseif type == 'bp'
    % 模拟低通到模拟带通
    zs = 1 ./ zn;
    W0 = sqrt(Wc(1) * Wc(2)); % 中心频率
    Bw = abs(Wc(1) - Wc(2)); % 带宽
    for i = 1:N
        ps(2 * i - 1:2 * i) = roots([1, -Bw * pn(i), W0^2]);
    end
    ks = Bw^N;
end

bs = ks .* poly(zs);
as = poly(ps);
%%
% 双线性变换，将模拟系统映射到数字系统
% [zz, pz, kz]  零极点增益形式
% [bz, az]      传递函数形式
zs = zs(isfinite(zs));
Z = (1 + zs * (T / 2)) ./ (1 - zs * (T / 2));
```

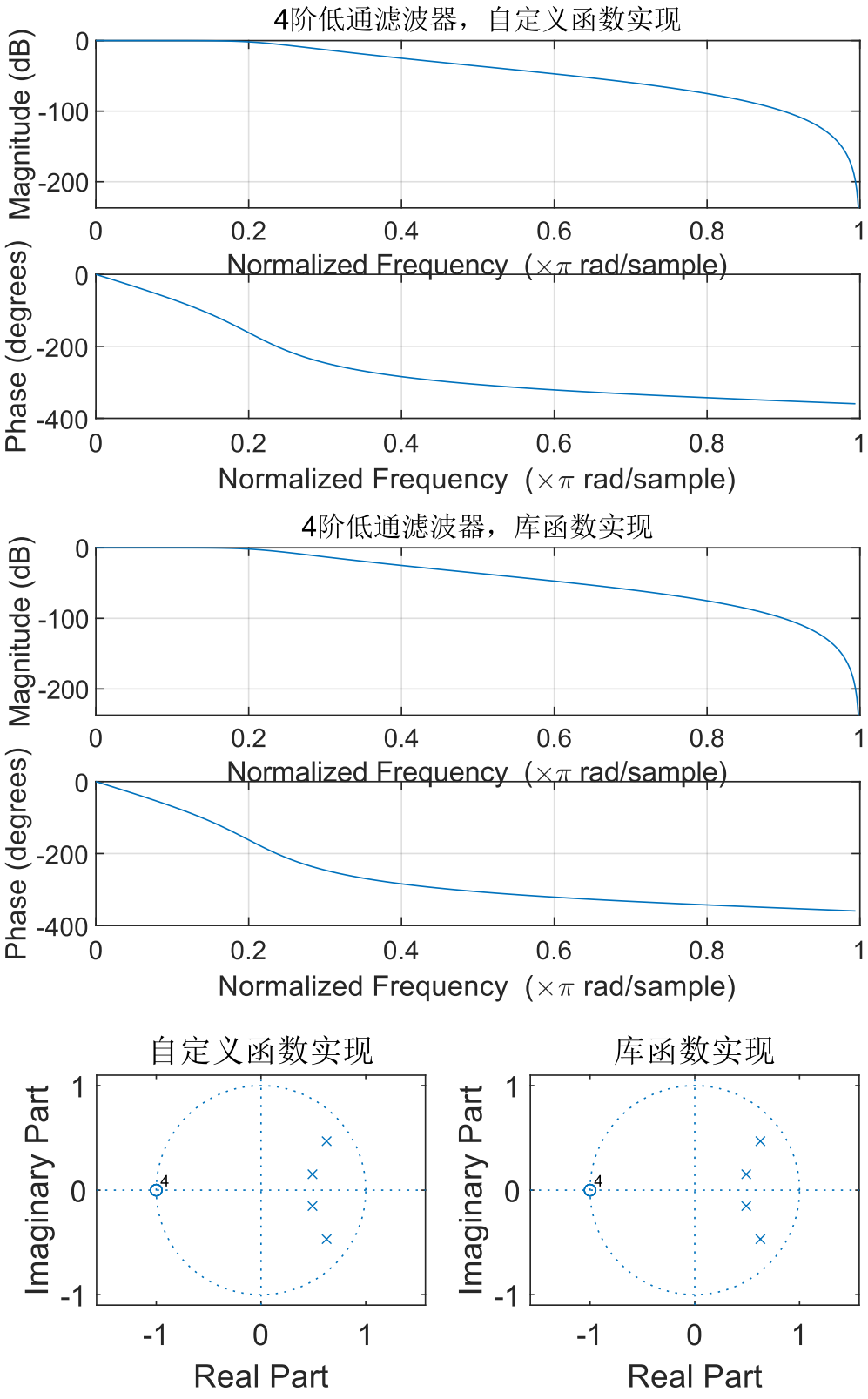
```
pz = (1 + ps * (T / 2)) ./ (1 - ps * (T / 2));
kz = ks .* real((prod((2 / T) - zs)) ./ prod((2 / T) - ps));
zz = -ones(size(pz));

for i = 1:numel(Z)
    zz(i) = Z(i);
end

az = real(poly(pz));
bz = real(kz .* poly(zz));
end
```

3.2.3. 滤波器函数（filter）

```
function y = my_filter(b, a, x)
% 直接 II 型 IIR 滤波器的转置结构
% x     输入信号
% y     输出信号
% u     缓存
order = max(numel(a), numel(b)) - 1;

a = a ./ a(1);
b = b ./ a(1);
a(1) = [];
y = zeros(size(x));
u = zeros(1, order);

for i = 1:length(x)
    y(i) = b(1) * x(i) + u(1);
    u = [u(2:order), 0] + b(2:end) * x(i) - a * y(i);
end
end
```

3.3. 实验结果

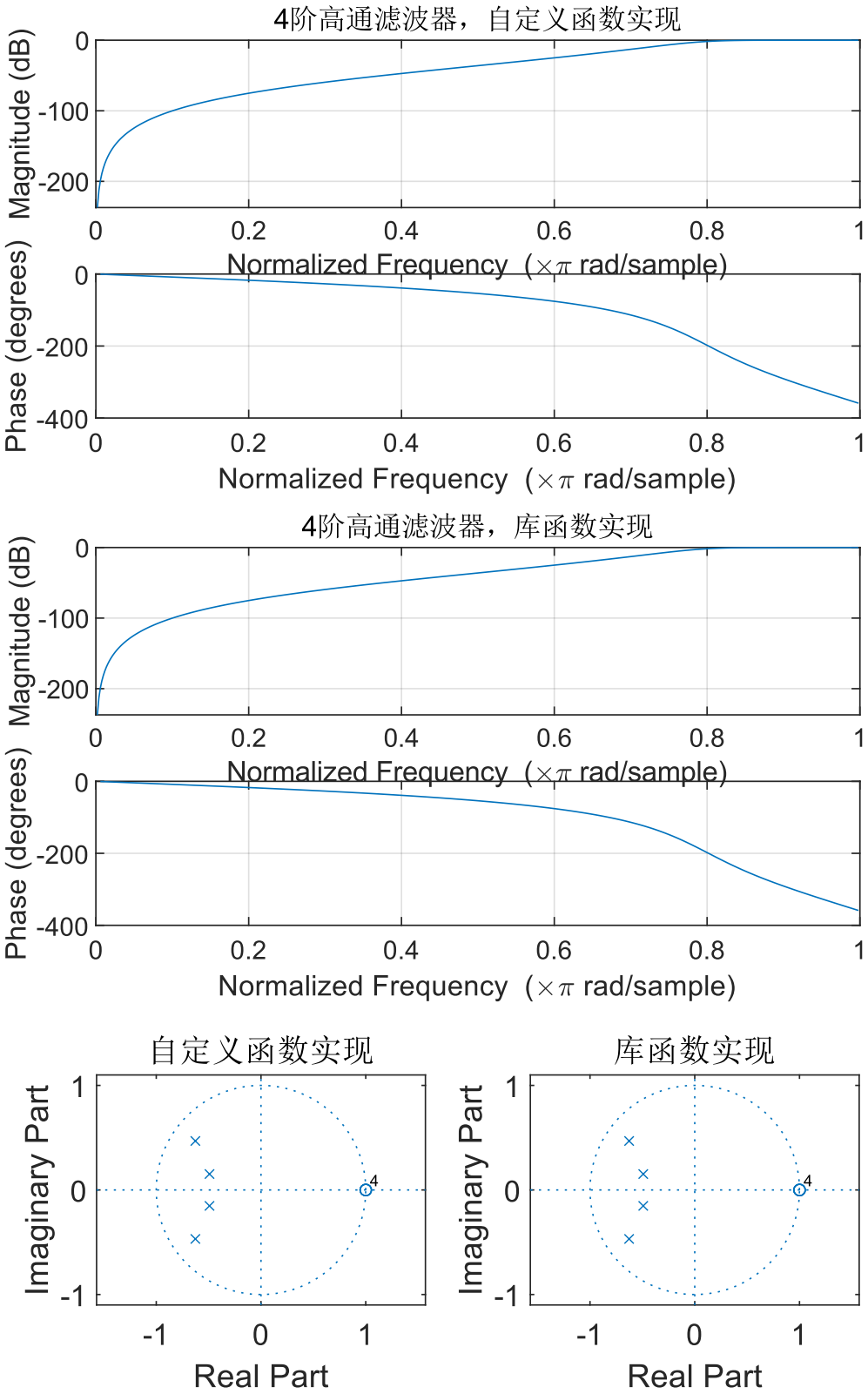
3.3.1. 低通滤波器



低通滤波器的频率响应曲线和零极点分布

通带频率0.2π，衰减小于 2dB；阻带频率0.4π，衰减大于 25dB

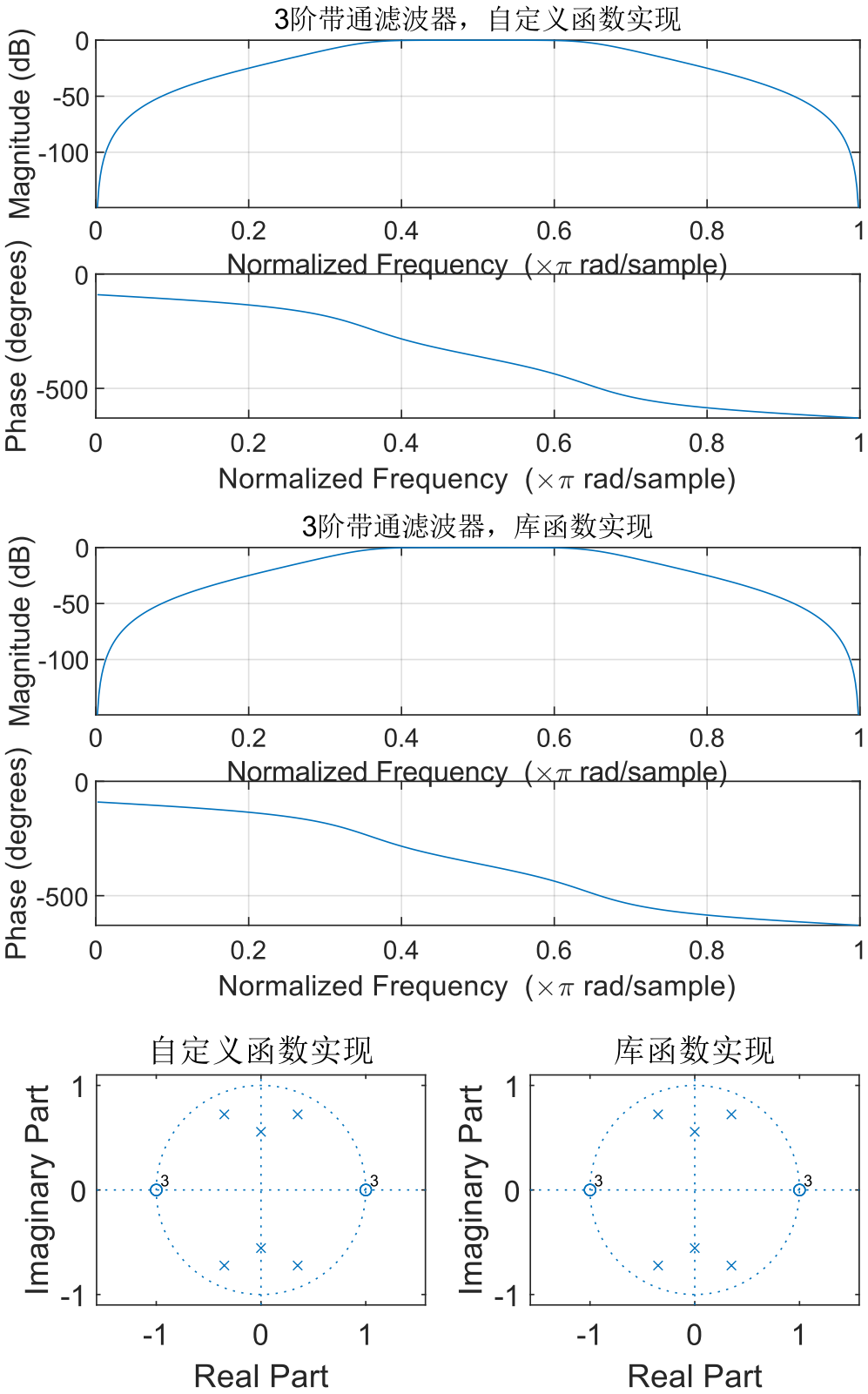
3.3.2. 高通滤波器



高通滤波器的频率响应曲线和零极点分布

通带频率0.8π，衰减小于 2dB；阻带频率0.6π，衰减大于 25dB

3.3.3. 带通滤波器

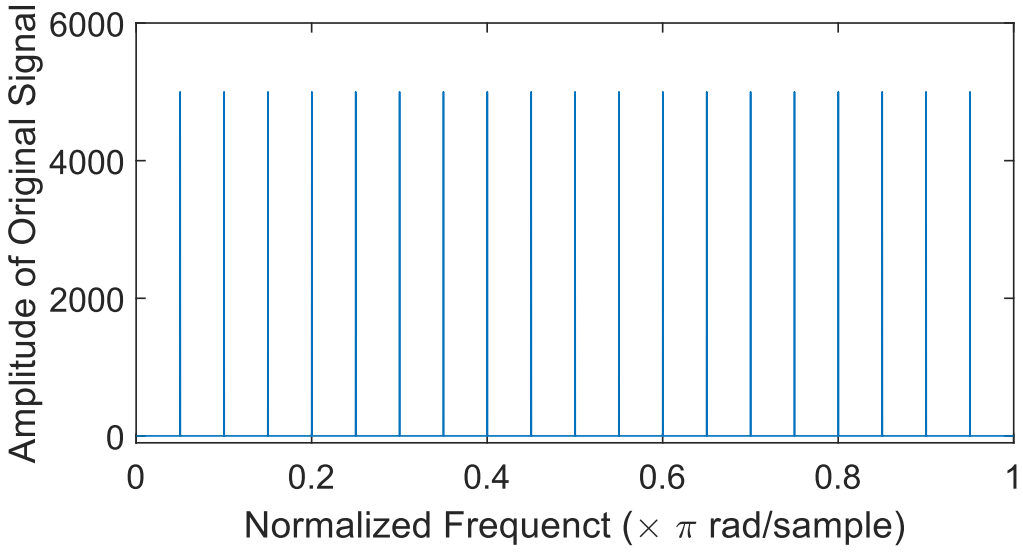


带通滤波器的频率响应曲线和零极点分布

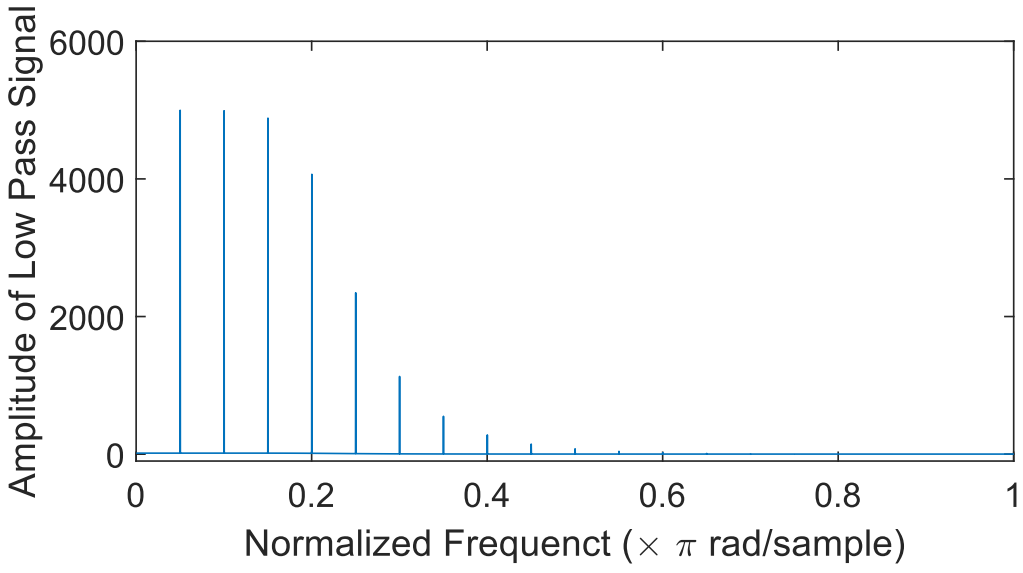
通带频率0.4π和0.6π，衰减小于 2dB；阻带频率0.2π和0.8π，衰减大于 25dB

3.3.4. 根据传递函数计算输出信号

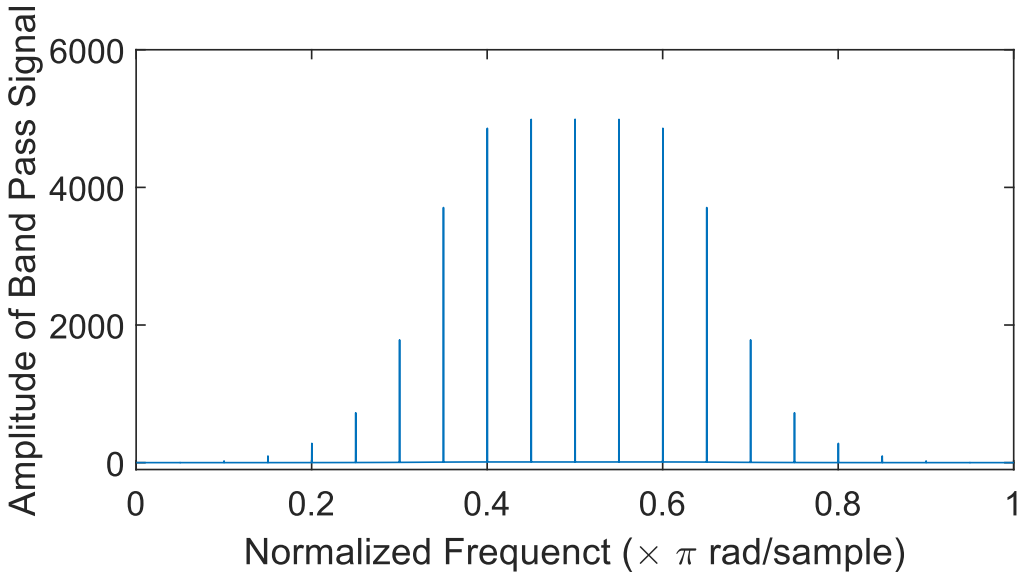
输入信号采样率为 10kHz，时长 1s。信号为频率从 250Hz 开始，以 250Hz 为步进增加到 4750Hz 的 19 个等振幅的正弦信号叠加而成。计算该信号分别通过前文中设计的低通、带通和高通滤波器后的频谱。



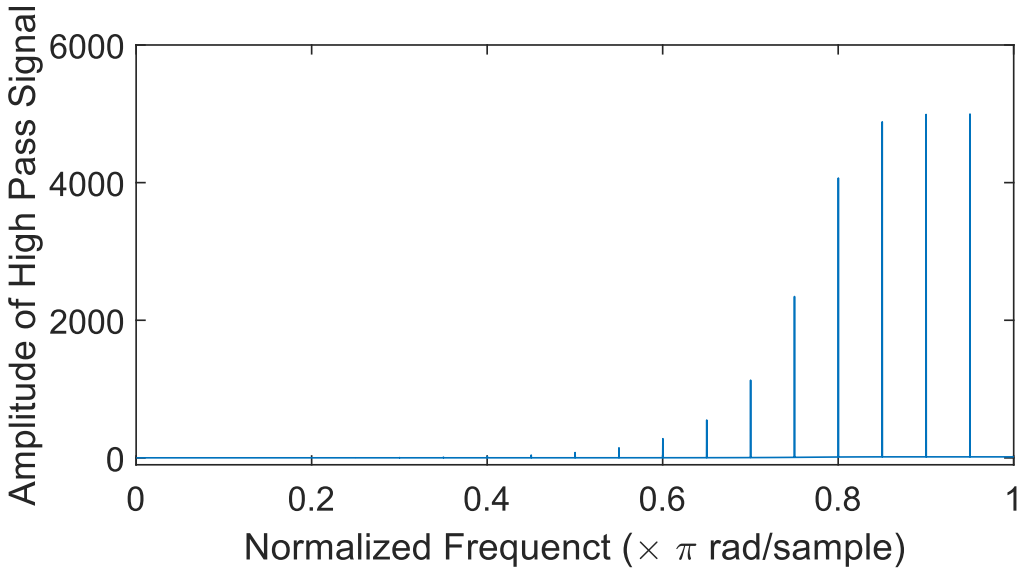
输入信号的频谱



信号通过低通滤波器后的频谱

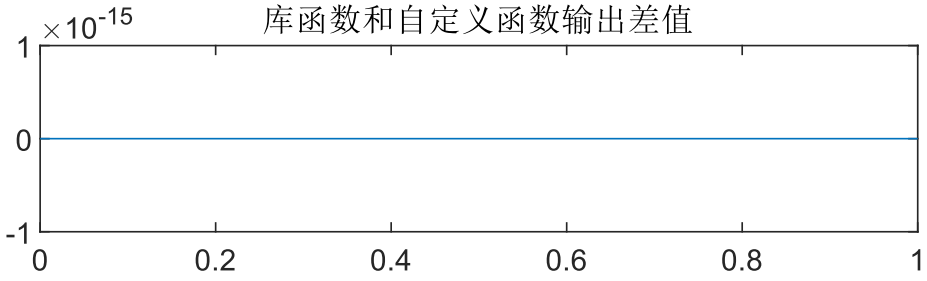


信号通过带通滤波器后的频谱



信号通过高通滤波器后的频谱

使用自定义函数和库函数分别计算高通滤波器的输出，并求差值，结果如下。



3.4. 实验总结

本次实验中，我实现了数字低通、高通和带通巴特沃斯滤波器的系统函数计算，以及根据传递函数计算输出结果。较前一次课程作业，我在其中增加了带通滤波器的内容，并完善了对滤波器去归一化过程的计算。

推导并完成计算滤波器的系统函数的 MATLAB 计算函数时，我参考了 MATLAB 中的库函数和数字信号处理教材，根据其中的内容，逐步推导计算公式并验算结果。在此基础上，逐步把库函数中调用的其他函数替换为自己的计算内容，最后得到结果。除此之外，我还参考了 python 的 scipy.signal 中和滤波器设计相关的函数源代码和 Stack Overflow 论坛中关于 filter 函数的实现方法，最后完成了相关的计算功能。