中国海洋大学

信号与系统
扩展项目报告

小组成员 ： 李祖乐 李延涛 李帅

任课教师 ： 冯晨

日期 ： 2023/12/10

目录

拓展项目报告

一、项目分工

李祖乐：组长、分析需求、代码编写、汇总排版与修改润色

李延涛：报告编写（问题描述、项目要求部分、拓展部分的了解 FFT）

李　帅：报告编写（拓展部分中分段平均周期法对功率谱估计进行改进、总结部分）

二、问题描述

2.1 数据描述

现有一组从 AUV（Autonomous Underwater Vehicle，自主式水下航行器）推进系统模型中采集的数据，模拟了桨叶受到不同程度的缠绕干扰时，以 1000Hz 采样率采集的电机电压变化情况。

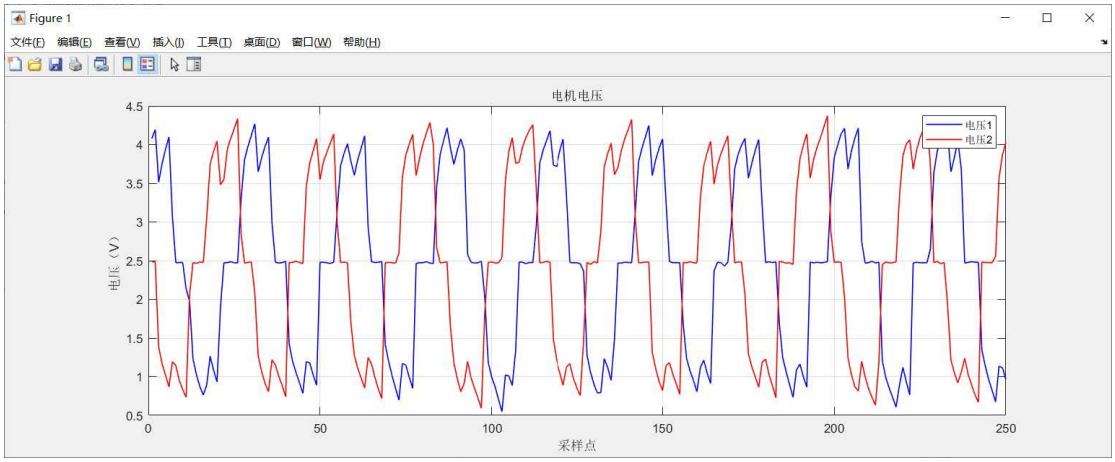

图 1 三相电机中的两相

2.2 项目要求

①编写程序，绘制不同程度缠绕干扰下电机电压的功率谱，并观察区别；

②使用不同的信号长度，绘制功率谱，并观察区别；

③向信号中添加噪声，观察功率谱的区别

2.3 拓展

① 了解 FFT（Fast Fourier Transform，快速傅里叶变换）算法的工作原理，以及相对于离散傅里叶变换具有的计算速度优势；

② 用有限长样本序列的傅里叶变换来表示随机序列的功率谱，只是一种估计或近似，不可避免存在误差。为了减少误差，使功率谱估计更加平滑，可采用分段平均周期图法（Bartlett 法）、加窗平均周期图法（Welch 法）等方法加以改进。

三、项目要求实现

3.1 问题一：编写程序，绘制不同程度缠绕干扰下，电机电压的功率谱，并观察区别

3.1.1 Matlab 程序

主程序

```
clear;clc
load data.mat
%% global config
sam_f = 1000; % 采样频率
sam_tim = 1/sam_f; % 采样时间间隔
%% local config
N = 500; % 计算的随机序列长度,
%% req_1：编写程序，绘制不同程度缠绕干扰下，电机电压的功率谱，并观察区别；
figure(1)
for i=20:10:100
    plot_i = (i-10)/10;
    data_name = strcat('data', num2str(i));
    data = eval(data_name);  % 获取变量数据
    [f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
    S = (X_m.^2)/N; % 计算功率谱 S-f
```

```
        subplot(3, 3, plot_i)
        plot(f, S)
        xlabel('Hz')
        ylabel('功率谱密度 V^2/Hz')
        title('功率谱密度')
        legend(strcat('未缠绕程度—', num2str(i)))
    end
```

其中函数“DFT”代码见附录

3.1.2 实验结果图

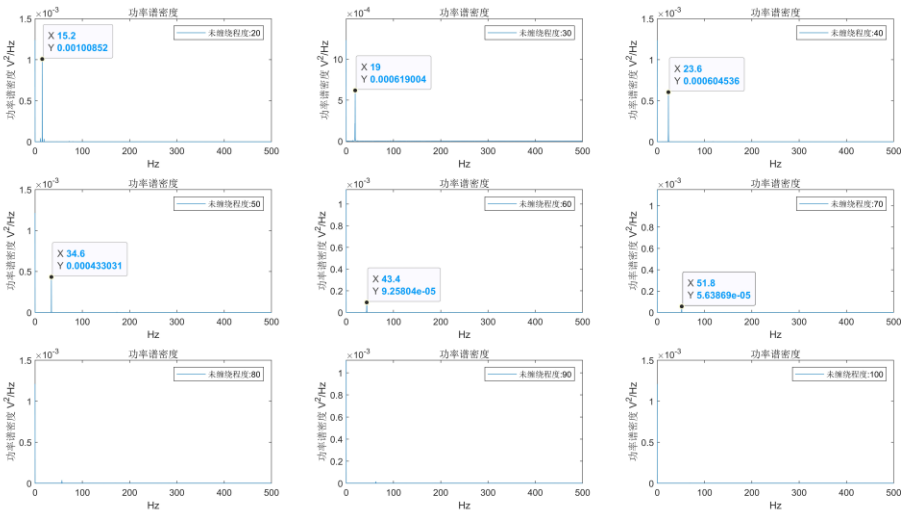

图 1 不同缠绕干扰程度下的电机功率谱密度

3.1.3 结论

对 3.1.2 实验结果图分析观察得如下规律：

1、缠绕程度越高，桨叶所受阻尼越大，功率谱密度主波峰越靠近低频值，即更低频的信号分量占比更多。

2、功率谱密度最大值随桨叶受到的未缠绕程度增加而降低。

3.2 使用不同的信号长度，绘制功率谱，并观察区别

此处**信号长度 N** 代表的是选取**有限长随机信号序列为$x(n)$的长度**，此小节取**未缠绕程度为固定值 60**，同时**选取不同的信号长度 N** 进行实验。

3.2.1 Matlab 代码

```
%% req_2：使用不同的信号长度，绘制功率谱，并观察区别；
for i=20:10:100
    figure_i = (i)/10;
    data_name = strcat('data', num2str(i));
    data = eval(data_name);  % 获取变量数据
    figure(figure_i)
    for N=1000:1000:9000 %信号长度从 1000 取 9000，间隔为 1000
        plot_i = N/1000;
        [f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
        S = (X_m.^2)/N; % 计算功率谱 S-f
        subplot(3, 3, plot_i)
        plot(f, S);
        title(strcat('功率谱密度 N=', num2str(N)))
        xlabel('Hz')
        ylabel('V^2/Hz')
        legend(strcat('未缠绕程度:', num2str(i)))
    end
end
```

3.2.2 实验结果图

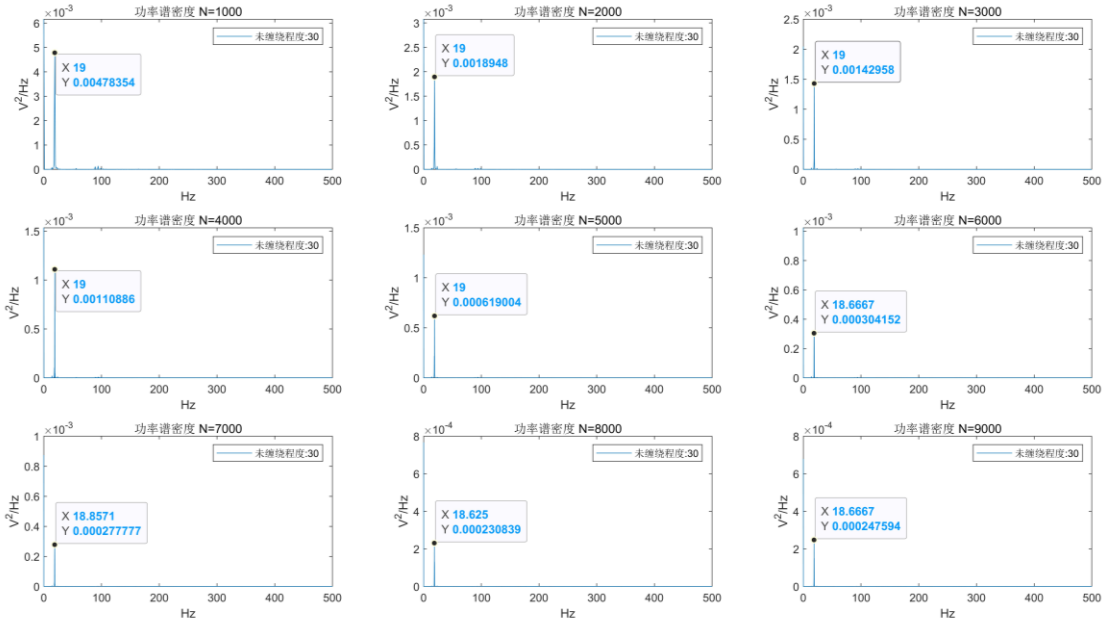

图 2 未缠绕程度为 30 时不同信号长度下电机功率谱密度

3.2.3 结论

对 3.2.2 实验结果图分析观察得如下规律：

1、功率谱密度表明了信号在各个频率上的功率分布情况，绘制信号功率谱密度所取的信号序列长度越长，频率分辨率越高。

2、随着信号序列长度的不断增大，频率分辨率不断提高，功率谱密度峰值处的信号频率也更加精确，同时功率谱密度峰值的大小也在不断变化，但可以发现，当信号序列长度达到一定值后，其功率谱密度峰值大小趋于一定值，表明了信号序列长度的选取对功率谱密度计算有着较大的影响，且信号长度越大，所得到的结果也更加稳定，同时更具有代表性。

3.3 向信号中添加噪声，观察功率谱的区别

本节分别向原信号加入了两种噪声，分别为**高斯白噪声**和**高斯随机噪声**，并分别在**取定未缠绕程度为 20** 的情况下，分别通过改变**信噪比**和**均值期望**来观察噪声在信号功率谱密度中的影响。

3.3.1 Matlab 程序

```
%% req_3：向信号中添加噪声，观察功率谱的区别
%local config
N = 500; % 计算的随机序列长度
level = 20;
data_name = strcat('data', num2str(level)); % data20
data = eval(data_name);  % 获取变量数据
figure(figure_i+1)
% 绘制无噪声情况
plot_i = 1;
[f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
S = (X_m.^2)/N; % 计算功率谱 S-f
subplot(2, 3, plot_i)
plot(f, S); % 平均功率谱
title(strcat('平均功率谱 无噪声'))
legend(strcat('缠绕程度—', num2str(level)))
% 加入不同信噪比的高斯白噪声信号
for nsr=0.05:0.2:0.85
    plot_i = fix(nsr/0.2) + 2;
    data_with_noise = awgn(data, nsr);
    [f,X_m,X_phi] = DFT(data_with_noise(1:N), sam_tim, N, 0); % 进行离散傅
里叶单边变换
    S = (X_m.^2)/N; % 计算功率谱 S-f
    subplot(2, 3, plot_i)
    plot(f, S); % 平均功率谱
    title(strcat('平均功率谱 white gauss nsr=', num2str(nsr)))
    legend(strcat('缠绕程度—', num2str(level)))
end
figure(figure_i+2)
% 绘制无噪声情况
plot_i = 1;
[f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
S = (X_m.^2)/N; % 计算功率谱 S-f
subplot(2, 3, plot_i)
plot(f, S); % 平均功率谱
title(strcat('平均功率谱 无噪声'))
legend(strcat('缠绕程度—', num2str(level)))
% 加入不同分布的高斯随机噪声
for mes=0.1:3:15
    plot_i = fix(mes/3)+2;
    data_with_noise = data + randn(size(data));
    [f,X_m,X_phi] = DFT(data_with_noise(1:N), sam_tim, N, 0); % 进行离散傅
```

```
里叶单边变换
    S = (X_m.^2)/N; % 计算功率谱 S-f
    subplot(2, 3, plot_i)
    plot(f, S); % 平均功率谱
    title(strcat('平均功率谱 random gauss mes=', num2str(mes)))
    legend(strcat('缠绕程度—', num2str(level)))
    end
```

3.3.2 实验图片

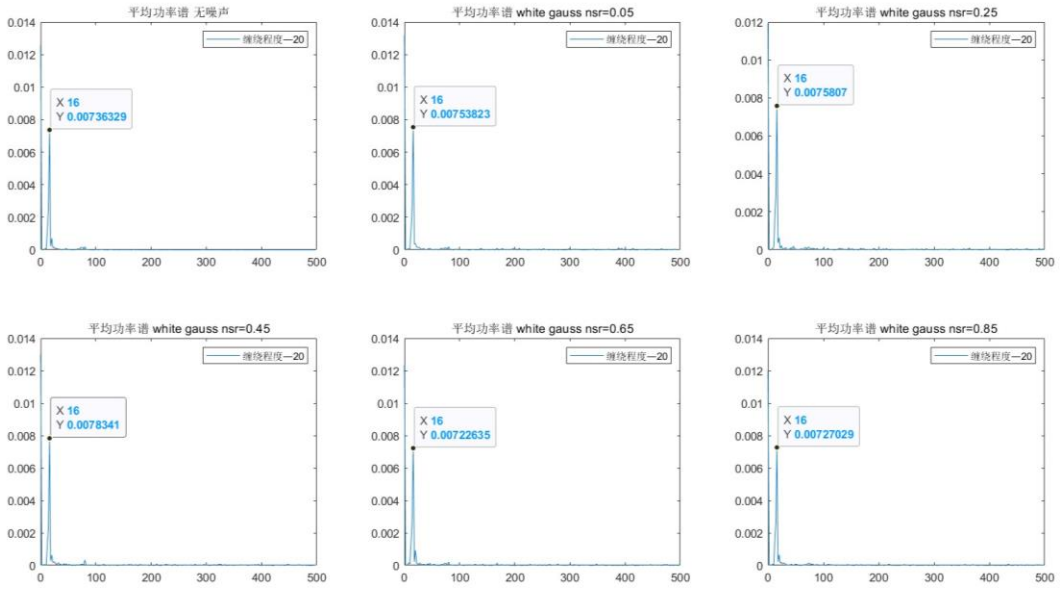

图 3 加入高斯白噪声并取不同信噪比

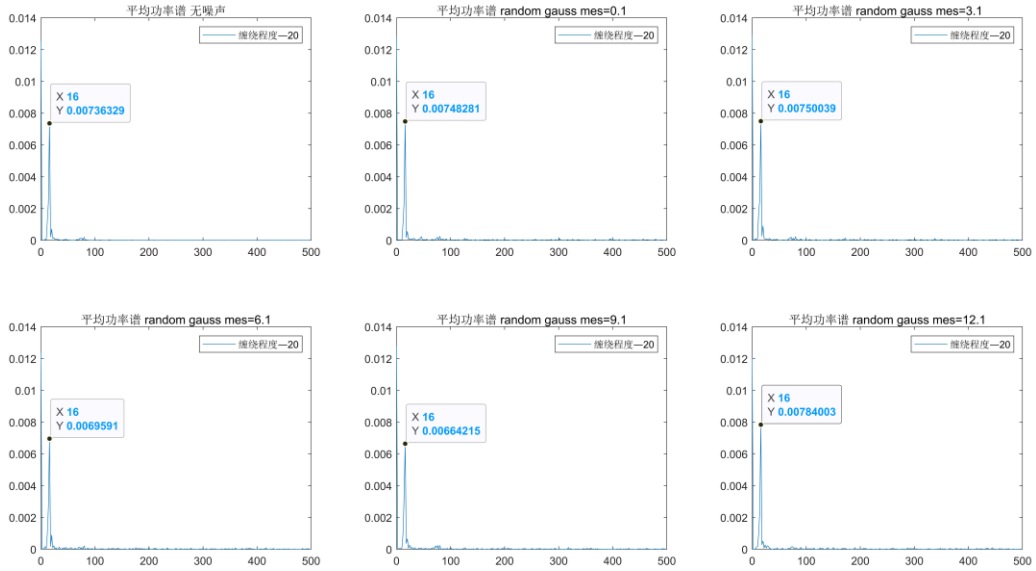

图 4 加入高斯随机噪声并取不同期望

3.3.3 结论

对 2.3.2 实验结果图分析观察得如下规律：

1. 从图中可知相比于无噪声时的情况，在加入高斯噪声后功率谱密度在其他频率成分上有了更多的起伏，表明噪声信号在这些频率分段上占有的功率大于原信号本身在此频率上的功率占比。
2. 同时也可观察到噪声信号的加入使得功率谱密度的峰值产生一些变化，推测可能为高斯噪声加入后在原有峰值对应频率上的功率叠加所致。

四、拓展内容实现

4.1 了解 FFT

4.1.1 FFT 简介

FFT 是快速傅里叶变换（Fast Fourier Transformation）的缩写。它是一种用于对非周期函数进行频谱密度函数逼近的离散傅里叶变换算法。FFT 算法实质上就是 DFT 算法的改良版，而 DFT 算法则是傅里叶变换的离散版。

4.1.2 FFT 计算流程

1. 首先，我们把输入的时域信号$x[n]$ （n = 0 , 1 , . . . , N – 1）根据索引分为奇偶两部分：

$$f_{odd}[n] = x[2n] \ , \ f_{even}[n] = x[2n+1] \ n = 0,1,2,\ldots,\frac{N}{2}-1$$

2.此时对 DFT 公式进行化简：

$$\mathrm{X[k]} = \sum_{n=0}^{N-1} x[n]W_N^{k_n} , k = 0,1,\ldots,N-1 \ \ W_N^{k_n} = e^{-jk\frac{2\pi}{N}n}$$

化简后结果：

$$\mathrm{X[k]} = \sum_{m=0}^{\frac{N}{2}-1} x[2m]W_N^{k_{2m}} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]W_N^{k_{2m+1}} , k = 0,1,\ldots,N-1$$

3. 根据旋转因子的缩放性，进一步换算得：

缩放性：$W_N^{k_n} = W_{N/2}^{\frac{k_n}{2}}$

$$\mathrm{X[k]} = \sum_{m=0}^{\frac{N}{2}-1} f_{even}[m]W_{N/2}^{k_m} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_{even}[m]W_{N/2}^{k_m} , k = 0,1,\ldots,N-1$$

即：$\mathrm{X[k]} = F_{even}[k] + W_N^k F_{odd}[k] , k = 0,1,\ldots,\frac{N}{2}-1$

其中$F_{even}[k]$和$F_{odd}[k]$分别是$f_{even}[n]$和$f_{odd}[n]$的 DFT 结果。

4. 分析$F_{even}[k]$和$F_{odd}[k]$性质并进行简化

$$F_{even}[k+\frac{N}{2}] = F_{even}[k]$$

$$F_{odd}[k+\frac{N}{2}] = F_{odd}[k]$$

由上可得：$\mathrm{X[k]} = F_{even}[k] + W_N^k F_{odd}[k] , k = 0,1,\ldots,\frac{N}{2}-1$

即有：$\mathrm{X[k+\frac{N}{2}]} = F_{even}[k] + W_N^k F_{odd}[k] , k = 0,1,\ldots,\frac{N}{2}-1$

可以看出：一个 N 点的 DFT 结果，可以被两个奇偶输入的 DFT 结果计算得到。举个例子也即，8 点的 DFT，可以被偶 4 点 DFT 结果和奇 4 点 DFT 结果计算得到，同理奇/偶 4 点 DFT 又可以被 2 点 DFT 结果计算得到，以此类推，分治求解。

4.1.3 FFT 优缺点

优点：

1 速度快： FFT 算法的时间复杂度为 O(nlogn)，相比于暴力计算 DFT 的 O(n^2)时间复杂度算法的速度更快。因此 FFT 算法适用于大规模数据处理，如数字信号处理、图像处理等领域。

2 计算复杂度低：FFT 算法的计算复杂度低，可以在较短的时间内完成大量数据的处理。这使得 FFT 算法成为了许多科学计算和工程应用中的重要工具。

3.适用性广：FFT 算法不仅适用于数字信号处理和图像处理等领域，还可以应用于其他领域，如计算机视觉、机器学习、量子计算等。

缺点：

1.精度问题：FFT 算法在计算过程中可能会出现精度问题。这是因为 FFT 算法需要对浮点数进行离散化处理，从而可能导致精度损失。为了解决这个问题，可以采用高精度计算或者使用其他算法。

2.实现复杂度高：FFT 算法的实现复杂度较高，需要一定的数学基础和编程技能。此外，FFT 算法的实现还需要考虑数据的存储方式、计算精度等问题，这增加了实现的难度。

3.数据长度限制：FFT 算法的计算速度和精度都与数据长度有关。当数据长度过大时，FFT 算法的计算速度会变慢，精度也会受到影响。因此，FFT 算法的应用受到了数据长度的限制。

4.2 分段平均周期法（Bartlett 法）对功率谱估计进行改进

本节采用**周期为 500 进行**绘制 Bartlett 图

4.2.1 Matlab 代码

```
clear;clc
load data.mat
%% global config
sam_f = 1000; % 采样频率
sam_tim = 1/sam_f; % 采样时间间隔
%% local config
N = 500; % 计算的随机序列长度,
%% req_1：编写程序，绘制不同程度缠绕干扰下，电机电压的功率谱，并观察区别；
figure(1)
```

```
for i=20:10:100
    plot_i = (i-10)/10;
    data_name = strcat('data', num2str(i));
    data = eval(data_name);  % 获取变量数据
    subplot(9, 3, (plot_i-1)*3+1)
    plot(1:N, data(1:N))
    xlabel('采样点')
    ylabel('V')
    title('电机电压')
    S_y = zeros([fix(length(data)/N)*N, 1]); % 存储总的功率谱
    S_y_mean = zeros([fix(N/2), 1]);         % 存储总的功率谱平均值
    S_x_mean = (0:(length(S_y_mean)-1)).*2;  % S_y_mean 的横坐标
    S_x = zeros([fix(length(data)/N)*N, 1]); % 存储对应 S_y 的横坐标
    for j=1:fix(length(data)/N)
        [f,X_m,X_phi] = DFT(data(((j-1)*N+1):j*N), sam_tim, N, 0); % 进行离
散傅里叶单边变换
        S = (X_m.^2)/N; % 计算功率谱 S-f
        S_y((j-1)*N/2+1:j*N/2) = S;
        S_x((j-1)*N/2+1:j*N/2) = f;
        S_y_mean = S_y_mean + S;
    end
    subplot(9, 3, (plot_i-1)*3+2)
    scatter(S_x, S_y, '.'); % 平均功率谱
    xlabel('Hz')
    ylabel('V^2/Hz')
    title('功率谱-频率')
    subplot(9, 3, (plot_i-1)*3+3)
    S_x_mean = f;
    plot(S_x_mean, S_y_mean/i); % 平均功率谱
    xlabel('Hz')
    ylabel('V^2/Hz')
    title('平均功率谱-频率')
end
%% req_2：使用不同的信号长度，绘制功率谱，并观察区别；
for i=20:10:100
    figure_i = (i)/10;
    data_name = strcat('data', num2str(i));
    data = eval(data_name);  % 获取变量数据
    figure(figure_i)
    for N=1000:1000:5000
        plot_i = N/1000;
        S_y = zeros([fix(length(data)/N)*N, 1]); % 存储总的功率谱
        S_y_mean = zeros([fix(N/2), 1]);         % 存储总的功率谱平均值
        S_x_mean = (0:(length(S_y_mean)-1)).*2;  % S_y_mean 的横坐标
```

```
        S_x = zeros([fix(length(data)/N)*N, 1]); % 存储对应 S_y 的横坐标
        for j=1:fix(length(data)/N)
            [f,X_m,X_phi] = DFT(data(((j-1)*N+1):j*N), sam_tim, N, 0); % 进
行离散傅里叶单边变换
            S = (X_m.^2)/N; % 计算功率谱 S-f
            S_y((j-1)*N/2+1:j*N/2) = S;
            S_x((j-1)*N/2+1:j*N/2) = f;
            S_y_mean = S_y_mean + S;
        end
        subplot(2, 5, plot_i)
        scatter(S_x, S_y, '.'); % 功率谱
        title(strcat('N=', num2str(N)))
        subplot(2, 5, 5 + plot_i)
        S_x_mean = f;
        plot(S_x_mean, S_y_mean/i); % 平均功率谱
        title(strcat('N=', num2str(N)))
    end
end
%% req_3：向信号中添加噪声，观察功率谱的区别
%local config
N = 500; % 计算的随机序列长度
data_name = strcat('data', num2str(20)); % data20
data = eval(data_name);  % 获取变量数据
figure(figure_i+1)
% 绘制无噪声情况
plot_i = 1;
[f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
S = (X_m.^2)/N; % 计算功率谱 S-f
subplot(2, 5, plot_i)
scatter(S_x, S_y, '.'); % 功率谱
title(strcat('平均功率谱 无噪声'))
subplot(2, 5, 5 + plot_i)
plot(S_x_mean, S_y_mean/i); % 平均功率谱
title(strcat('平均功率谱 无噪声'))
% 加入不同信噪比的高斯白噪声信号
for nsr=0.05:0.2:0.8
    plot_i = fix(nsr/0.2) + 2;
    data_with_noise = awgn(data, nsr);
    S_y = zeros([fix(length(data_with_noise)/N)*N, 1]); % 存储总的功率谱
    S_y_mean = zeros([fix(N/2), 1]);         % 存储总的功率谱平均值
    S_x_mean = (0:(length(S_y_mean)-1)).*2;  % S_y_mean 的横坐标
    S_x = zeros([fix(length(data_with_noise)/N)*N, 1]); % 存储对应 S_y 的横
坐标
    for j=1:fix(length(data_with_noise)/N)
```

```
        [f,X_m,X_phi] = DFT(data_with_noise(((j-1)*N+1):j*N), sam_tim, N,
0); % 进行离散傅里叶单边变换
        S = (X_m.^2)/N; % 计算功率谱 S-f
        S_y((j-1)*N/2+1:j*N/2) = S;
        S_x((j-1)*N/2+1:j*N/2) = f;
        S_y_mean = S_y_mean + S;
    end
    subplot(2, 5, plot_i)
    scatter(S_x, S_y, '.'); % 功率谱
    title(strcat('white gauss nsr=', num2str(nsr)))
    subplot(2, 5, 5 + plot_i)
    plot(S_x_mean, S_y_mean/i); % 平均功率谱
    title(strcat('white gauss nsr=', num2str(nsr)))
end
% 绘制无噪声情况
figure(figure_i+2)
plot_i = 1;
[f,X_m,X_phi] = DFT(data(1:N), sam_tim, N, 0); % 进行离散傅里叶单边变换
S = (X_m.^2)/N; % 计算功率谱 S-f
subplot(2, 5, plot_i)
scatter(S_x, S_y, '.'); % 功率谱
title(strcat('平均功率谱 无噪声'))
subplot(2, 5, 5 + plot_i)
plot(S_x_mean, S_y_mean/i); % 平均功率谱
title(strcat('平均功率谱 无噪声'))
% 加入不同分布的高斯随机噪声
for mes=0.1:3:12
    plot_i = fix(mes/3)+2;
    data_with_noise = data + randn(size(data));
    S_y = zeros([fix(length(data_with_noise)/N)*N, 1]); % 存储总的功率谱
    S_y_mean = zeros([fix(N/2), 1]);         % 存储总的功率谱平均值
    S_x_mean = (0:(length(S_y_mean)-1)).*2;  % S_y_mean 的横坐标
    S_x = zeros([fix(length(data_with_noise)/N)*N, 1]); % 存储对应 S_y 的横
坐标
    for j=1:fix(length(data_with_noise)/N)
        [f,X_m,X_phi] = DFT(data_with_noise(((j-1)*N+1):j*N), sam_tim, N,
0); % 进行离散傅里叶单边变换
        S = (X_m.^2)/N; % 计算功率谱 S-f
        S_y((j-1)*N/2+1:j*N/2) = S;
        S_x((j-1)*N/2+1:j*N/2) = f;
        S_y_mean = S_y_mean + S;
    end
    subplot(2, 5, plot_i)
    scatter(S_x, S_y, '.'); % 功率谱
```

```
    title(strcat('random gauss mes=', num2str(mes)))
    subplot(2, 5, 5 + plot_i)
    S_x_mean = f;
    plot(S_x_mean, S_y_mean/i); % 平均功率谱
    title(strcat('平均功率谱 random gauss mes=', num2str(mes)))
    end
```

4.2.2 实验图片

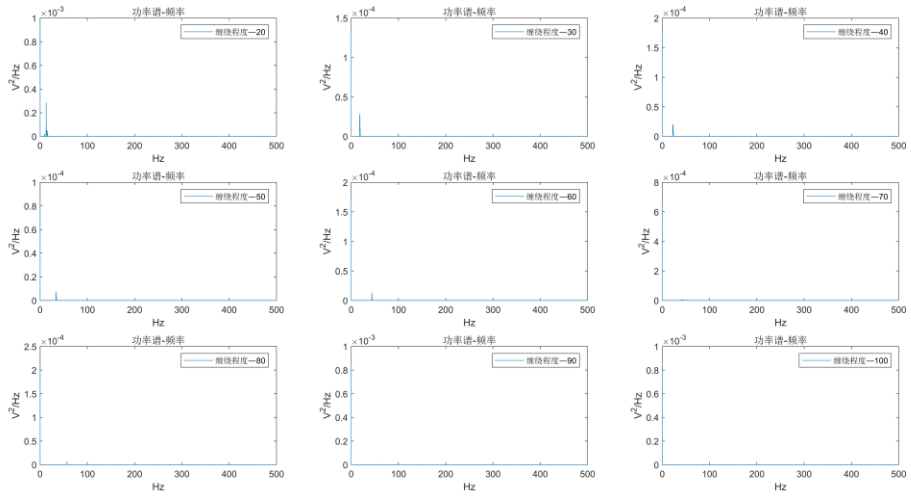

图 5 不同未缠绕程度下分段平均周期折线图

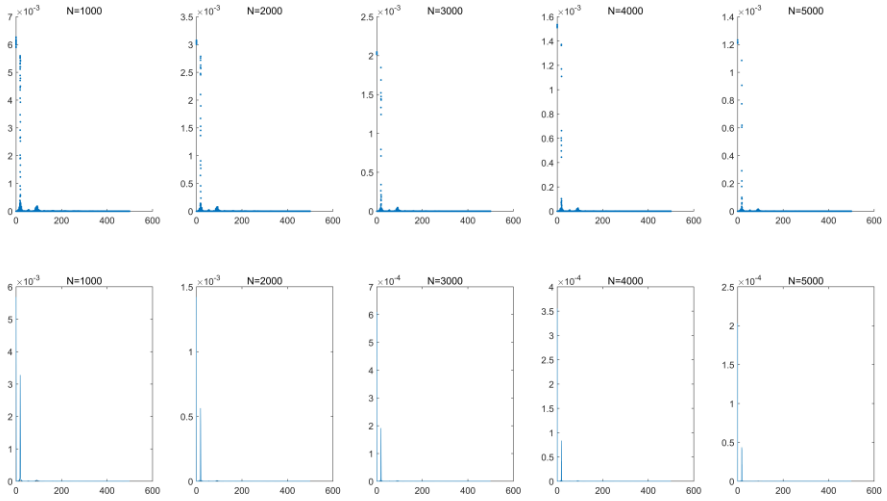

图 6 未缠绕程度为 30 时不同分段平均周期下功率谱密度

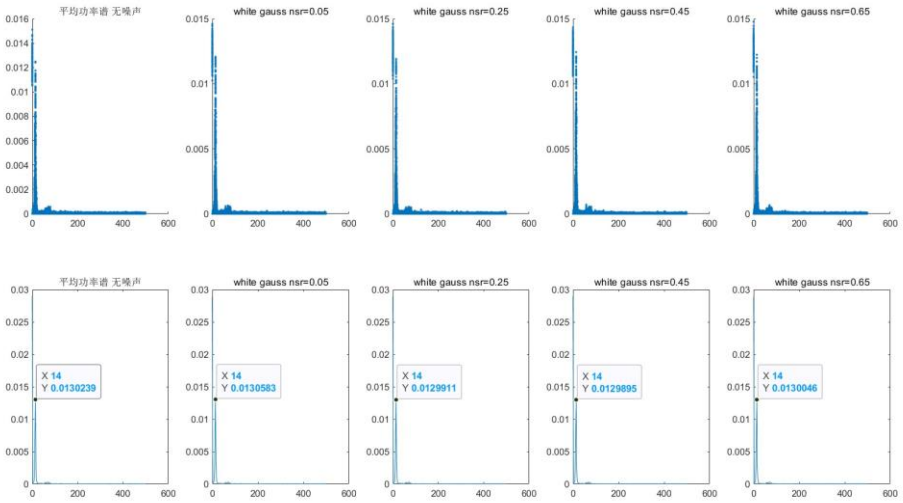

图 7 不同信噪比高斯白噪声作用下分段平均周期散点图和折线图

4.2.3 结论

1. 与全段长度下的功率谱密度相比，使用分段平均周期法绘制的功率谱密度曲线更加平滑。
2. 与全段功率谱密度相比，通过分段平均周期法绘制得到的高斯白噪声作用下的功率谱密度变化更小，这表明使用分段平均周期法求取信号的功率谱密度的抗干扰能力更强，同时数值结果也更加具有代表性。

五、实验总结

5.1 实现过程概述：

小组成员分工明确，较好地完成了项目中的三个基本需求。并在此基础上，小组成员较深入地了解了 FFT 算法原理后，经过线下细致讨论利用分段平均周期法（Bartlett 法）完成了拓展内容的要求。报告的两个部分都实现了通过分别改变未缠绕程度、信号长度以及加入不同高斯噪声得到不同的功率谱密度结果后进行相互对比得出结论的过程。

5.2 心得感受

通过本次项目，本小组成员成功实现了使用有限长序列离散傅里叶变换来求取信号功率谱密度，在此过程中收获颇丰，在此作如下总结：1.进一步加深了对傅里叶变换尤其是离散序列傅里叶变换的理解，在一定程度上也加深了对本课程内容的掌握程度。2.增强了对 Matlab 等数学分析工具的掌握程度。3.进一步增强了团队部分成员的代码编写或阅读能力。4.增强了团队成员之间的团队协作能力。

综上，此次拓展项目小组汇报是一次成果显著且极具意义的活动。

附录

DFT 函数

```
    function [f,X_m,X_phi] = DFT(xn,ts,N,drawflag)
    % [f,X_m,X_phi] = DFT(xn,ts,N,drawflag) 离散序列的快速傅里叶变换，时域转换为频域
    % 输入  xn 为离散序列 为向量
    %       ts 为序列的采样时间/s
    %       N 为 FFT 变换的点数，默认为 xn 的长度
    %       drawflag 为绘图标识位，取 0 时不绘图，其余非 0 值时绘图，默认为绘图
    % 输出 f 为频率向量
    %      X_m 为幅值向量
    %      X_phi 为相位向量，单位为°
    % 注意计算出来的 0 频分量(直流分量应该除以 2)  直流分量的符号应结合相位图来确定



    if nargin == 2
        N = length(xn);
        drawflag = 1;
    elseif  nargin == 3
        drawflag = 1;
    end


    if  isempty(N)
        N = length(xn);
```

```
end



Xk = fft(xn,N);          % FFT 变换
fs = 1/ts;               % 采样频率 HZ
X_m = abs(Xk)*2/N;       % 幅值量化变换
X_phi = angle(Xk);       % 相位
Nn = floor((N-1)/2);     % 变换后有用的点数-1
f = (0:Nn)*fs/N ;        % 横坐标 频率 HZ
X_m = X_m(1:Nn+1);       % 幅值(仅取有用点 Nn+1 个点)
X_phi = X_phi(1:Nn+1);   % 相位(仅取有用点 Nn+1 个点)
% X_phi = unwrap(X_phi);  % 去除相位的间断点(仅在出图时作用)
X_phi = X_phi*180/pi;    % 化成°单位

% 直流分量处理
X_m(1) = X_m(1)/2;       % 注意计算出来的 0 频分量(直流分量应该除以 2)

if drawflag ~= 0
    figure
    subplot(211)
    plot(f,X_m)
    title('DFT 的频率-幅值图');
    xlabel('频率/HZ');ylabel('幅值');
    grid on

    subplot(212)
    plot(f,X_phi)
    title('DFT 的频率-相位图');
    xlabel('频率/HZ');ylabel('相位/°');
    grid on
end
```