MMCS Project2

522031910213 朱涵

May 20, 2024

1 TASK1–PA MODELING

1.1 问题定义

PA（功率放大器）是一种接受输入信号并产生输出信号的电路元件，其输入信号的序列和输出信号的序列是有一定关系的，在本部分中，我们尝试建立模型来模拟这种关系，并测试模型的准确性。 根据建议，PA的输入输出信号关系可以这样定义：第i个输出信号y_i由前m个输入信号x_{i-m}~x_i来决定，即

$$y_i = f(x_i, x_{i-1}, ..., x_{i-m})$$

在接下来的部分我们将根据上述模型尝试使用MATLAB对给出的数据集进行训练和测试。

1.2 数据的预处理

由于数据是复数类型的，在训练数据之前需要先进行分离，即把数据分为实数部分和虚数部分，分别进行相同方法的训练和测试。 另外，数据在训练前还需要先进行标准化和归一化。过程见下图。

```
% 计算训练数据集的均值和标准差
train_input_real_mean = mean(train_input_real);
train_input_real_std = std(train_input_real);
train_input_imag_mean = mean(train_input_imag);
train_input_imag_std = std(train_input_imag);
train_output_real_mean = mean(train_output_real);
train_output_real_std = std(train_output_real);
train_output_imag_mean = mean(train_output_imag);
train_output_imag_std = std(train_output_imag);

% 对训练数据集进行归一化
train_input_real = (train_input_real - train_input_real_mean) / train_input_real_std;
train_input_imag = (train_input_imag - train_input_imag_mean) / train_input_imag_std;
train_output_real = (train_output_real - train_output_real_mean) / train_output_real_std;
train_output_imag = (train_output_imag - train_output_imag_mean) / train_output_imag_std;

% 使用训练数据集的均值和标准差对测试数据集进行归一化
test_input_real = (test_input_real - train_input_real_mean) / train_input_real_std;
test_input_imag = (test_input_imag - train_input_imag_mean) / train_input_imag_std;
test_output_real = (test_output_real - train_output_real_mean) / train_output_real_std;
test_output_imag = (test_output_imag - train_output_imag_mean) / train_output_imag_std;
```

Figure 1: 数据的预处理1：标准归一化

因为模型的输入变量是m维的，输出变量是1维的，因此需要进行一些处理使得输入输出序列符合模型，其中包括用滑动窗口把序列划分成m维和1维的，以及切除前m项数据（因为输出没有对应的输入数据），如下图。

```
% 定义记忆深度 m
m = 5;

% 构建m维的输入序列和1维的输出序列进行训练
train_input_real_split = [];
train_input_imag_split = [];
train_output_real_split = [];
train_output_imag_split = [];

for i = m+1:length(train_input_real)
    window_input_real = train_input_real(i-m:i-1);
    window_input_imag = train_input_imag(i-m:i-1);
    train_input_real_split = [train_input_real_split; window_input_real];
    train_input_imag_split = [train_input_imag_split; window_input_imag];
    train_output_real_split = [train_output_real_split; train_output_real(i)];
    train_output_imag_split = [train_output_imag_split; train_output_imag(i)];
end

test_input_real_split = [];
test_input_imag_split = [];
test_output_real_split = [];
test_output_imag_split = [];

for i = m+1:length(test_input_real)
    window_input_real = test_input_real(i-m:i-1);
    window_input_imag = test_input_imag(i-m:i-1);
    test_input_real_split = [test_input_real_split; window_input_real];
    test_input_imag_split = [test_input_imag_split; window_input_imag];
    test_output_real_split = [test_output_real_split; test_output_real(i)];
    test_output_imag_split = [test_output_imag_split; test_output_imag(i)];
end
```

Figure 2: 数据的预处理2：划分变量

1.3 模型的训练

根据建议，我使用了MATLAB中的fitnlm函数来进行非线性模型的拟合。我尝试使用了三次多项式的模型函数，并且初始化所有系数为1进行猜测。然后进行回归，得到模型。

```
% 定义模型函数为多项式函数（三次）
fun = @(beta, x) beta(1) + sum(beta(2:m+1)' .* x, 2) + sum(beta(m+2:2*m+1)' .* x.^2, 2) + sum(beta(2*m+2:3*m+1)' .* x.^3, 2);

% 初始化参数
beta0 = zeros(3*m + 1, 1);

% 使用非线性回归进行建模
rng default % 设置随机数种子，以确保结果可重复
nlm_regressor_real = fitnlm(train_input_real_split, train_output_real_split,fun, beta0);
nlm_regressor_imag = fitnlm(train_input_imag_split, train_output_imag_split,fun, beta0);
```

Figure 3: 训练模型

模型的训练过程中，我尝试了多种模型函数以及多个记忆深度值。模型函数包括指数函数、对数函数等，但是效果都不尽人意。最终无奈之下采用了三次多项式的模型函数，但得到的模型仍不完美。 如下图是测试的主要过程，包括使用模型进行预测，计算对应的NMSE，计算结果数据的归一化幅度以及绘制原数据和预测数据的曲线图象。

```
% 训练集预测
train_predicted_real = predict(nlm_regressor_real, train_input_real_split);
train_predicted_imag = predict(nlm_regressor_imag, train_input_imag_split);

% 测试集预测
test_predicted_real = predict(nlm_regressor_real, test_input_real_split);
test_predicted_imag = predict(nlm_regressor_imag, test_input_imag_split);

% 计算 NMSE
train_nmse =log10( (sum((train_predicted_real - train_output_real_split).^2)+sum((train_pre
test_nmse =log10( (sum((test_predicted_real - test_output_real_split).^2)+sum((test_predict

disp(['训练集 NMSE: ', num2str(train_nmse)]);
disp(['测试集 NMSE: ', num2str(test_nmse)]);

% 计算归一化幅度
train_input_amp = abs(train_input_real + 1j * train_input_imag);
train_output_amp = abs(train_output_real + 1j * train_output_imag);
train_predicted_amp = abs(train_predicted_real + 1j * train_predicted_imag);

test_input_amp = abs(test_input_real + 1j * test_input_imag);
test_output_amp = abs(test_output_real + 1j * test_output_imag);
test_predicted_amp = abs(test_predicted_real + 1j * test_predicted_imag);


% 绘制训练集实数部分和归一化幅度曲线
figure;
scatter(test_input_amp, test_output_amp, 'b','bo', 'filled', 'SizeData', 20);
hold on;
scatter(test_input_amp(m+1:end), test_predicted_amp', 'r','bo', 'filled', 'SizeData', 20);
title('测试集归一化幅度','FontName','楷体');
xlabel('输入归一化幅度','FontName','楷体');
ylabel('输出归一化幅度','FontName','楷体');
```

Figure 4: 测试结果的处理过程

1.4 模型的测试

使用三次多项式模型对测试集的数据进行预测并计算NMSE。不同记忆深度m下的结果如下表。可以看到NMSE在m=5时比较小，在大于5后差别不大。

记忆深度m	1	2	3	4	5	6	7	8	9	10
NMSE	-0.12737	-0.14669	-0.1479	-0.14792	-0.14805	-0.14794	-0.14786	-0.14779	-0.14767	-0.14757

Table 1: 不同记忆深度下得到的模型测试结果

如下图是m=5时的预测曲线以及原测试数据曲线。显然模型预测的结果和原数据相差甚大。介于本人能力有限，实在是不知道如何用非线性回归的方法得出一个较接近原数据的模型，最终只能展示如下的结果。

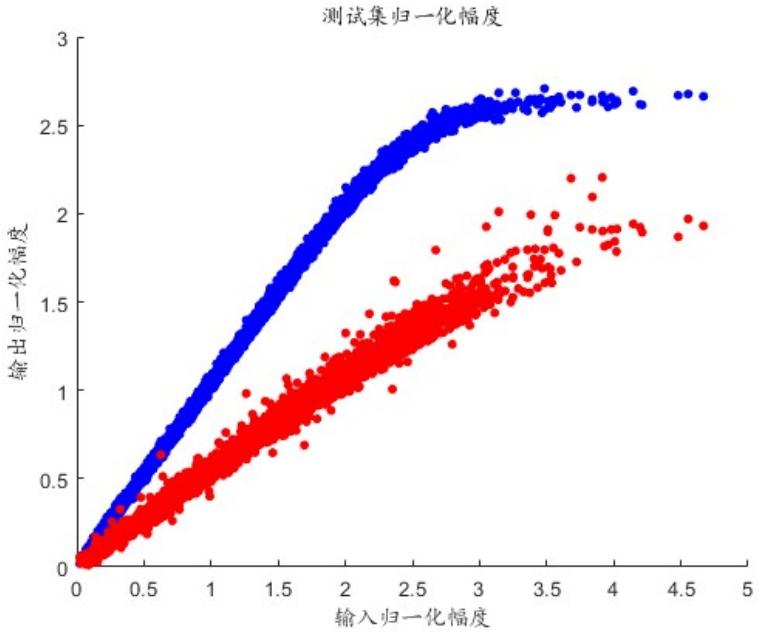

Figure 5: m=5时的测试结果曲线

1.5 总结

本次实验我尝试使用了MATLAB的非线性回归方法对与功率放大器（PA）的输入信号与输出信号间的关系进行了建模、拟合以及测试，初步学会了使用MATLAB进行建模以及回归分析，但模型结果缺陷比较明显，希望后续能够在老师的指导下进行改进。