EE 416 Autumn 2018
Final Project: Generalized Linear Model
Professor: Jim Ritcey

December 11th. 2018

He Feng 1427841
Huihao Chen 1560466

Abstract

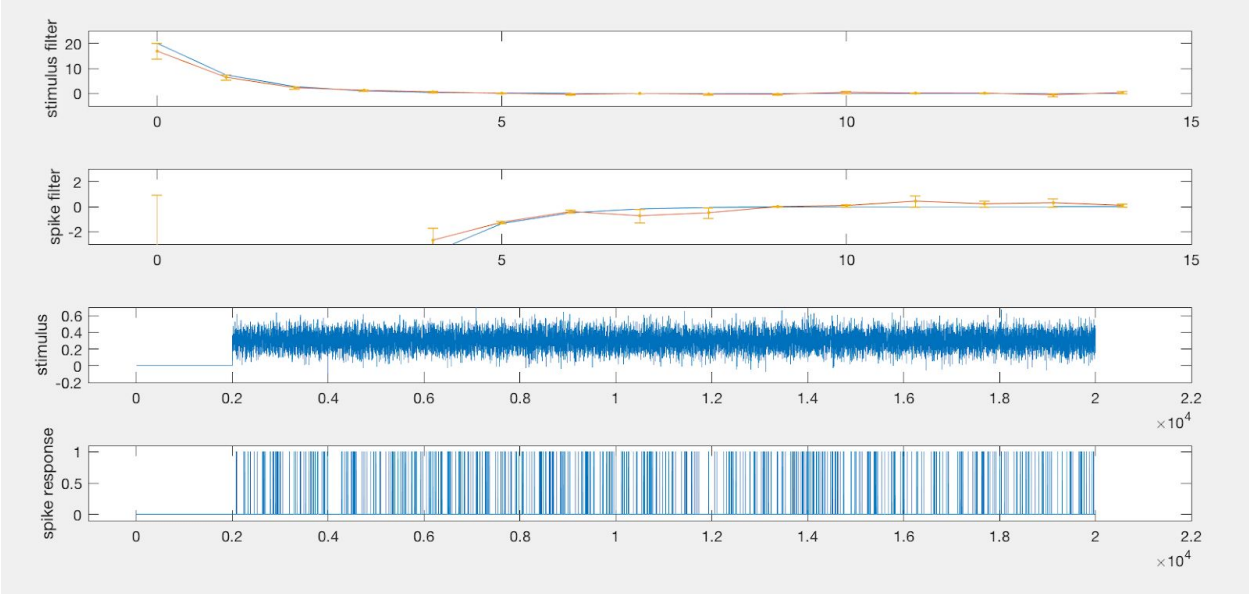
This is the final project of EE 416, and we used Matlab to generate the input and figure out the output. The whole project mainly focuses on the generalized linear model, which is a model has the dependent variables that are drawn from some distribution who has the linear transformation of a linear function of the independent variables as parameters. There are three parts in this project, and the first part has three sections.

At the beginning section of the first part, we initially used the given filters and a random variable to display the corresponding stimulus-response and spike response. We call this progress as GLM simulation. In the second section, we wrote a function to generate the simulated filters by using the stimulus-response and spike response as parameters. After creating the simulated filters, we put them into the plots from the last section to compare them with the given filters. After analyzing, we marked the error bars on the graph to see the difference more obvious. In the third section, we focus on the offset error, root means square error, and mean standard error. Initially, we generated three plots which introduce the relationship between the number of samples and each error, by adding the number of samples, we can see the difference while the number of samples is changing. Secondly, we built the functions of the spike counts and each error, and we increased the number of spike counts to find the relation between the number of samples and each error. Finally, we still want to see the relationship between the different type of error and the spike counts, but in this case, we will change the spike counts based on changing offset instead of changing the amount of data.

For the second part of the project, we changed the length of the filters, and we load real dataset as the stimulus-response and spike response, then we used the similar function in the second section of the first part to simulate new filters. After we display the filters on the screen, we marked the error bars on the filters such that we can see where did the data or we do wrong. In the last part of the project, we learned a new concept called inter-spike interval, which also called inter-arrival time. The corresponding event time is the cumulative time or time interval of the interval spike interval from the beginning to the current timestamp. The purpose for this part is to connect up the event times and counting process. In this part, we need to analyze the data and histogram and make a fit test with the Gamma distribution. The overall project is challenging but understandable, and we are excited to demonstrate our result to introduce these interesting topics.

Summary of Results

Section 4.1 and 4.2



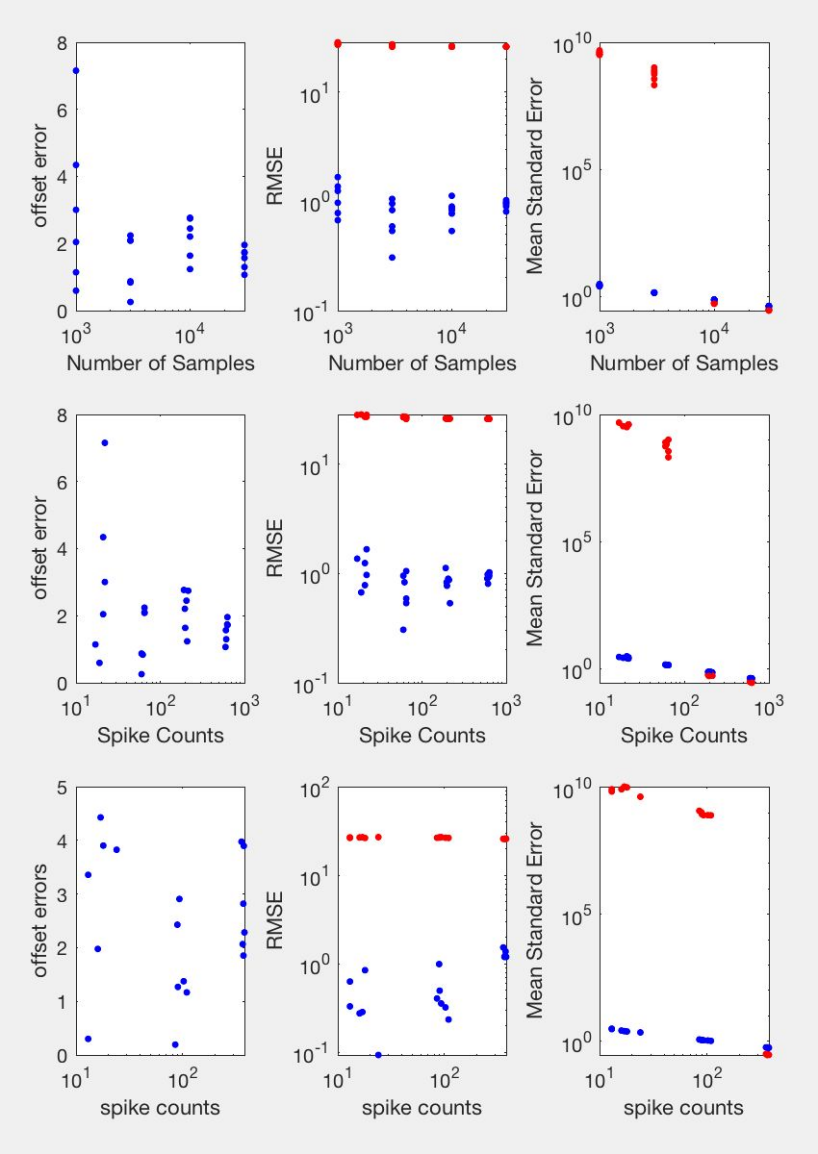
Both of the estimated parameters seem good, but the estimate stimulus filter is more accurate than the estimated spike filter because of the small error bars marked on the curve. The estimated spike filter is not that accurate because of the input spike response parameter. The function, fit_GLM, depends on the spike history from a finite number of previous time bins, and as the input value is small, the function cannot or hard to generate an accurate value While the real neurons will never pass two signals successively, the function can hardly tell what the filter is in this case. Hence if the spikes which have the value of one locate next to each other, when we pass the specific filter response as parameter to pass through the glmfit function, there will have some significant difference at the beginning of the estimate spike filter because of the instability caused by the nearly impossible input parameter. We should not consider the estimated result as a failure because most part of this plot is almost accurate, and the inaccurate beginning section is reasonable.

To determine the root squared mean error and mean standard error for the estimated spike filter and the given spike filter, we need to ignore the terms with coefficient at the beginning of the filters. In this lab, after generating the simulated spike filter and the given spike filter, we opened the workspace and figured out that the error bar is significantly large for the first five terms. Hence to make the RMSE and mean standard error more accurate, we should not count the terms

which has the significant large difference between them. If we ignore the beginning part, and start calculating the errors only for the rest of the piece. We can get a more accurate estimated spike filter with smaller error bar. The stimulus filter will not meet this problem in this case.

Section 4.3

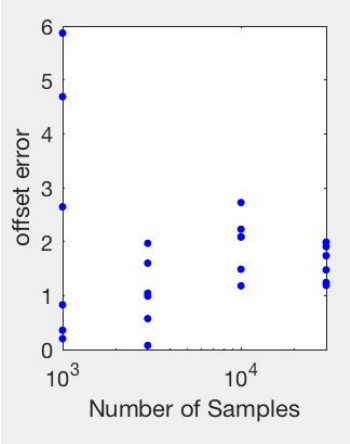
Overall output of this section:



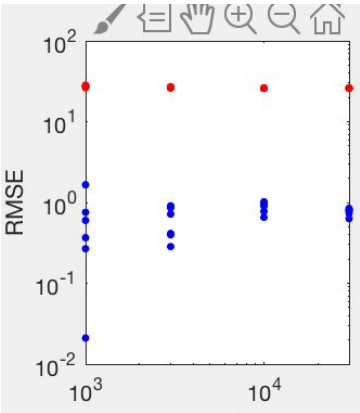
There are so many details in each separate plot that cannot be observed clearly, hence here are some graphs which have some details from the overall graph above:

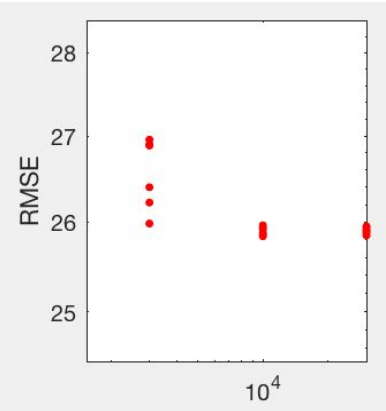
First row:

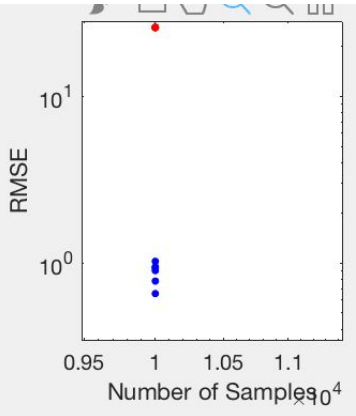
Offset Errors:



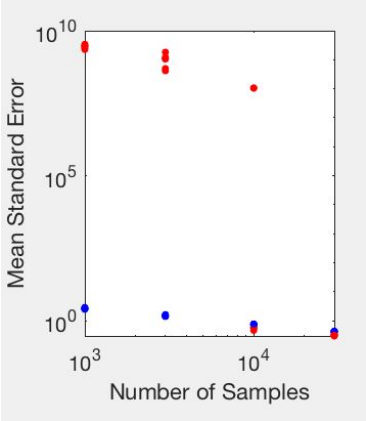
Root Squared Mean Errors:

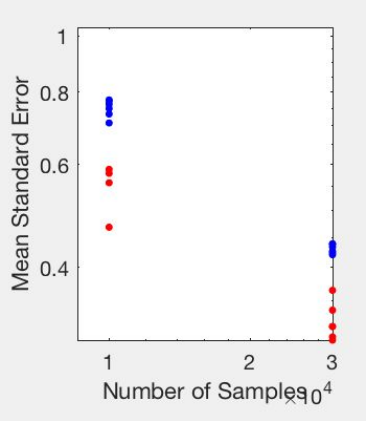


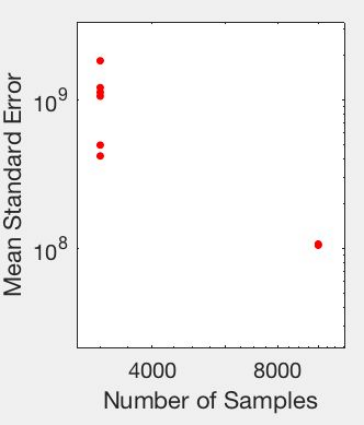




Mean Standard Errors:

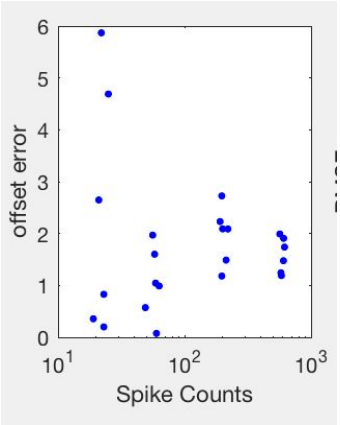




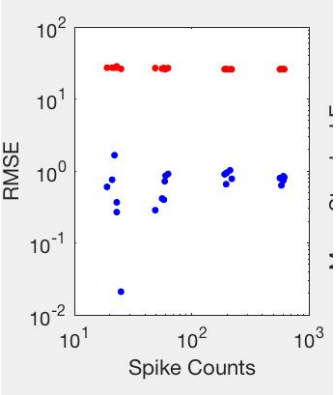


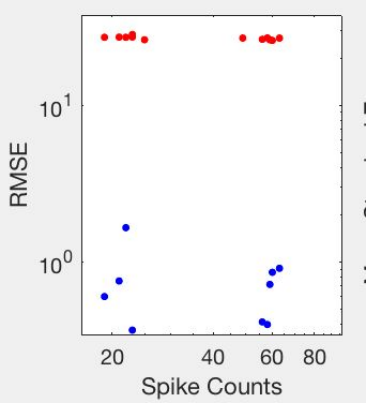
Second Row:

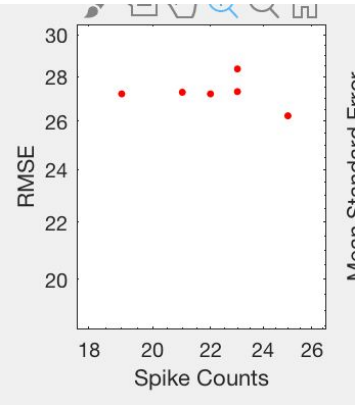
Offset Errors:



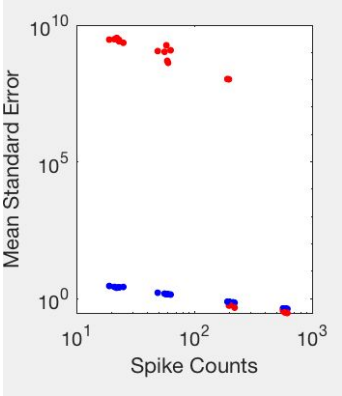
Root Square Standard Errors:

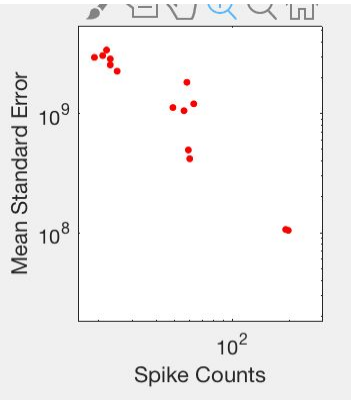


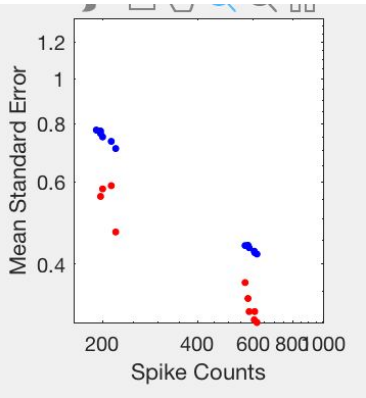




Mean Standard Errors:

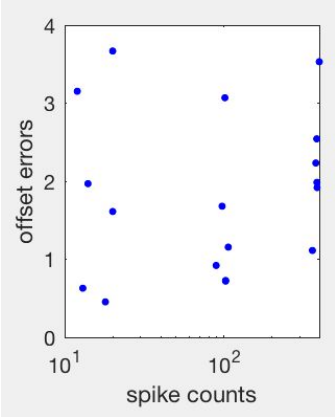




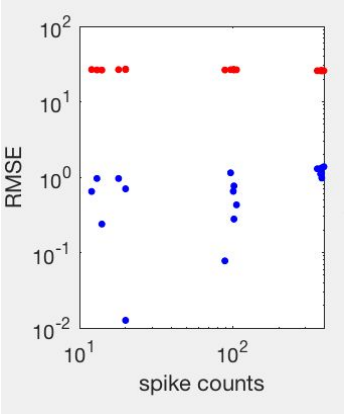


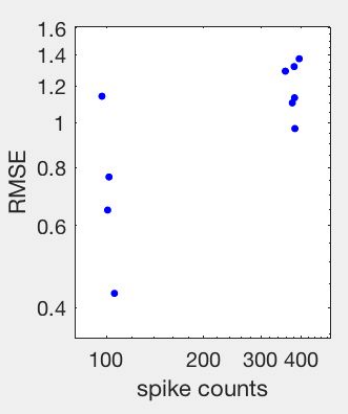
Third Row:

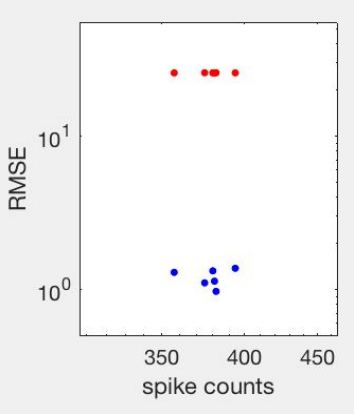
Offset Errors:



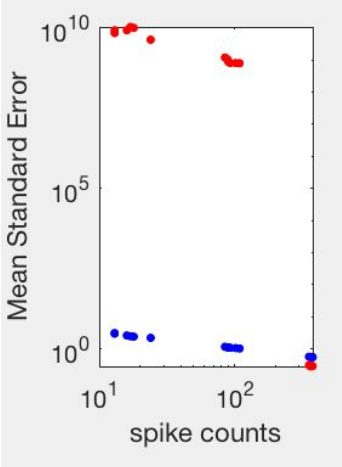
Root Square Standard Errors:

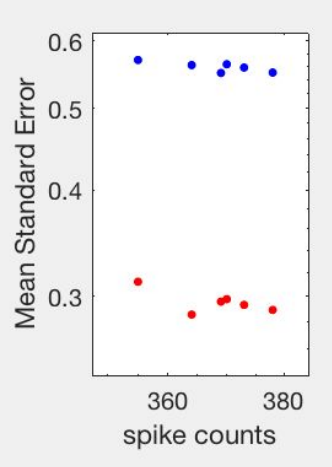


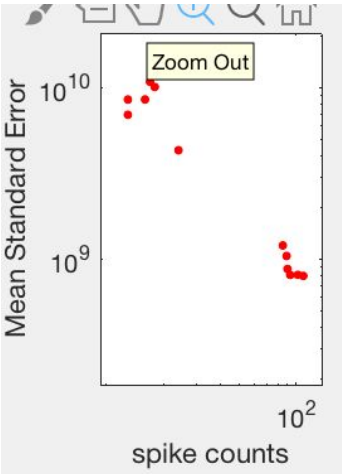




Mean Standard Errors:







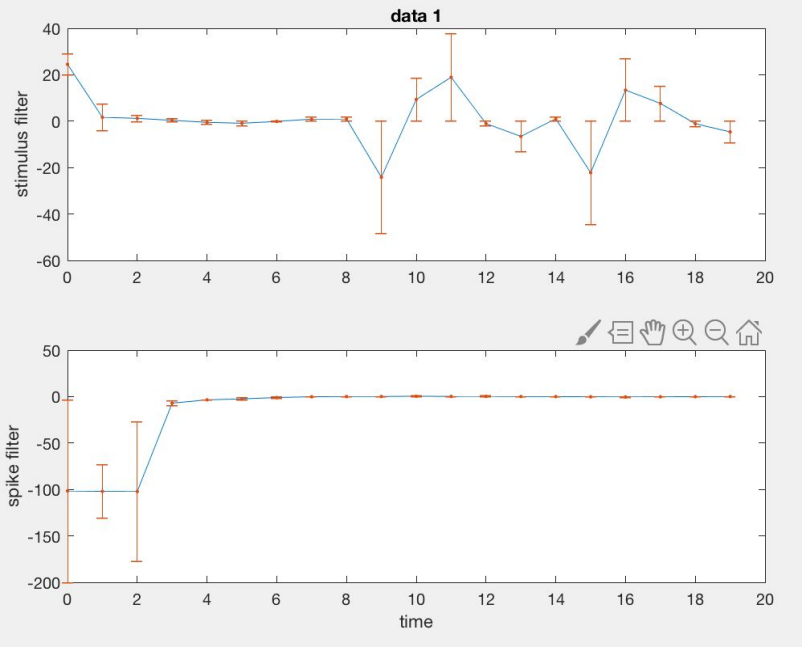
As the number of samples increased, the output will be closer to each other, and the value of the errors are also decreased, which means the accuracy of parameter fitting has been increased. The reason for this part is due to rule in statistics. While the more amount of data we have, we can get more choices to test the function, and the amount of generated output will have more authority and accuracy. The large amount of data may cover more situations, such that we can consider many different cases or situation while generating the output. Therefore as we increase the amount of samples the parameter fitting will be more accurate.

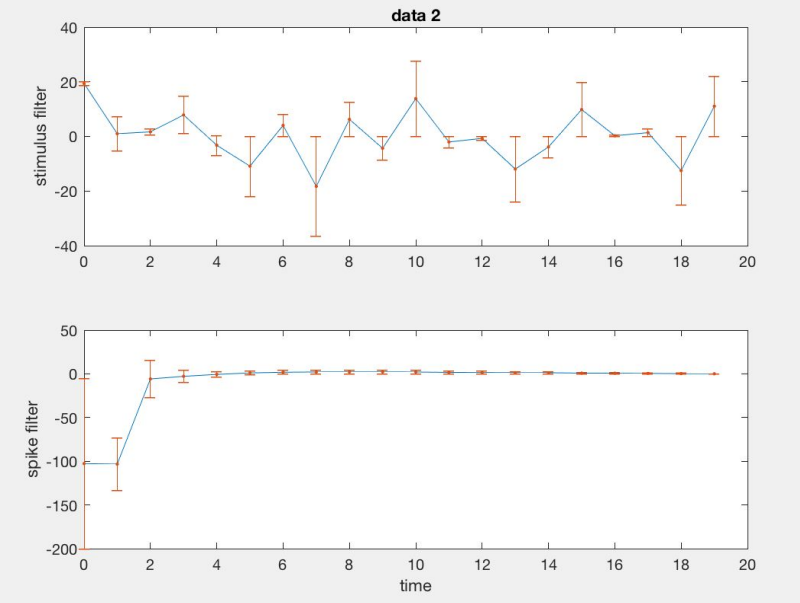
To compare the RMSE and mean standard errors. For the stimulus, the values between them are similar. They are have small values, and the difference is as the number of samples increasing, the mean standard error will decrease obviously. However, the result in RMSE is not that obvious. For the spiking, the RMSE values do not have significant changes while increasing the number of samples, and the mean standard errors are much larger than the RMSE value while the number of samples is not that big. However, as the number of samples increasing to a big enough amount, the mean standard errors decreased significantly, and its values are smaller than the corresponding RMSE values.

Then we are going to discover the relationship between the accuracy of the parameter fitting and the spike counts. The spike counts are related to the amount of samples and the offset values. As the number of spike counts increasing, the errors for different trails come closer, and the value of the errors are also decreased even though the decreasing tendency is not obvious from the plots above. The accuracy of the parameter fitting will be improved as we increase the number of spike count. This is consistent with the effect of more data. Because if we keep the offset value unchanged, the number of spike counts are relevant to the amount of data. The larger amount of data has greater amount of spike, which will influence the parameter fitting accuracy. If we keep the amount of the samples the same, the smaller offset (or the larger value of the absolute value of the offset) will cause the more accurate parameter fitting.

Even though the output result look similar, we prefer to use the spike counts in the future generation. Because the effect of using less than 1000 spike counts can cause the same effect of using more than 10000 data samples. Therefore by using spike counts we can generate the function easier and get the similar output. Before doing this project, we only know there are relationship between the error values and amount of samples, and we never thought about the changes that may be cause by the spike counts. In the future study, we need to make more brainstorming and deep thinking to figure about the other factors which may influence the lab.

Section 5



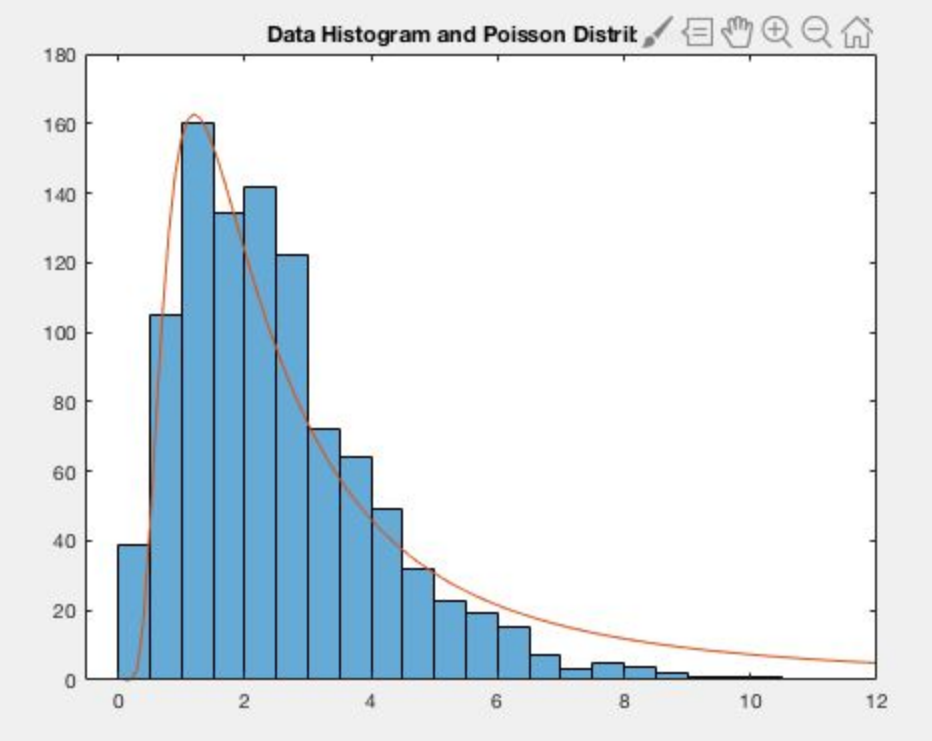


By looking at the graphics, we can tell that the filter in the first graphic is more smooth. Datasheet 1 contains about one hundred thousand data and datasheet 2 only contains half of the former one. From the Section 4.3, we can tell that the standard error will become more and more accurate when the number of samples increases. So in this case we can tell that the filters from datasheet 1 is more concise and reliable than the filters come from datasheet 2.

stimMSE1	4.1018	spikeMSE1	0.1693
stimMSE2	5.1019	spikeMSE2	0.1267

We can conform our theory from the mean standard errors. The MSE of the stimulus filter from the datasheet 1 is smaller than that from the second datasheet, while the MSE of the spike filters are both quite small.

Section 6



h	0

After plotting the data, we can tell from the graphic that the distribution of the data is very close to a Gamma distribution. We also plotted a standard Gamma distribution on the same graphic to confirm our guesses. Finally the chi-square test gave us a 0, which means the data passes the test at a significance level of 0.05

Analysis

Section 4.1

We started this project with the building of two filters. First we set the range of those two filters from 0 to 14, with a length of 15, and set the functions for them. Then we generated a white noise with the 'random' function, and this white noise signal will be the stimulated signal to generate spikes. We plot those three graphics and pass them as parameters into a new function called 'sim_GLM', which can help us get the stimulated spikes.

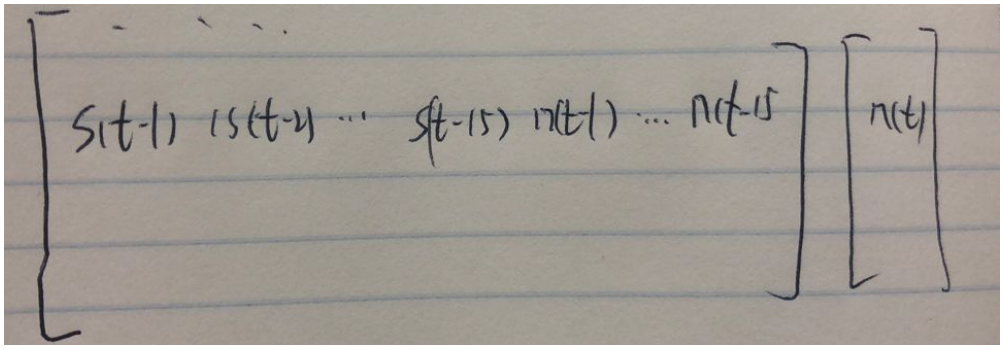
In the sim_GLM function, we used the equation that was mentioned in Section 3:

$$n(t) \sim Poisson(e^{f(0)s(t-dt)+..+f(d-1)s(t-d*dt)+h(0)n(t-dt)+..+h(d-1)n(t-d*dt)+b}),$$

To get this equation, we used a for loop to get the power of the exponential part. Then the function 'poissrnd' helped us simulate a poisson distribution. Finally we take the smaller one between 1 and the value, and then this function could return a stimulated spike which contains 1s and 0s, as show in the previous figures.

Section 4.2

In this section, we created another function called 'fit_GLM'. This function takes two set of data -- stimulations and spikes -- as the parameters, and it will help generate the offset 'b', the statistics 'stats', the stimulus filter 'f_fit' and the self-interaction filter 'h_fit'. To get it calculated, we first have to create a large matrix A that contains 30 the data in the simulations and spikes--15 from stimulations and 15 from spikes. Then we get another matrix B, which comes from the transposed spike data. The picture below shows what the two matrices look like. After getting those two matrices, we could apply the glmfit function provided by matlab to get all the statistics we need. We stored them into two data sets, 'coe' and 'stats'. The first value in 'coe' is the offset value, the second to the sixteenth value gives us the stimulus filter, and the seventeenth to the last value gives us the self-interaction filter.



Section 4.3

We aimed to figure out the relationship between different errors and amount of samples or amount of spike counts in this section. For the amount of spikes, we also discover the changes while we keep the offset unchanged and change the amount of samples, or we can keep the amount of samples unchanged and change the offset values.

To plot the offset errors plots, we used the fit_GLM function built before to generate the standard errors. Then we used the given offset value and the generated offset to figure out the difference and determine the offset error. To determine the offset errors plot as the function of the samples numbers and the spike counts are the same, the only difference is we need to change the variable on x-axis. To plot the RMSE errors plot, we used the following equation:

$$\mathrm{RMSE}_{fo} = [\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N]^{1/2}$$

By using this equation and plug in the data of stimulus and spike filters, we can get the amount of the root mean square errors. Then we need to change the x-axis variables to draw different equations to determine the relationship between the errors and the number of samples or spike counts.

To plot the standard mean errors, we just need to sum all of the standard errors in one case together and divided by the number of standard errors such that we can figure out the standard mean errors. After finishing the plotting procedures, we put all of the nine plots in one figure, and observe them to make several conclusion which has been introduced in the previous part of the project report. One of the main point in this section is to use for loop such that we can generate each sample with multiple trails easier.

Section 5

The purpose of this section is to change the fit_GLM function and apply it to a real-life situation. We started with modifying the function and named it as 'fit_GLM20', since the filter length is 20 in this case. This function is basically the same as the one we used before, we just have to change some values such as the size of the matrices and switch the indices of f_fit and h_fit.

With the new fitting function, we can read the data, put them into matrices f and h, then plug them into the function. We also plotted the graphics for those filters and found the mean standard error.

Section 6

In this section, we first loaded the data and used a for loop to get the value for each stimulus from the cumulative function. The relationship between the cumulative function and the inter-spike intervals is shown below.

$$S_n = \sum_{\ell=1}^{n} X_\ell$$

Then we used histogram function and generated the graphic. On the other hand, we created a standard Gamma distribution on the same graphic and compared them with our eye. We can tell that the data fits the Gamma distribution well.

The last thing we do was to use chi-square test to check whether statistically the data fits Gamma distribution. We used the function 'chi2gof' to help us. The defalt Alpha is 5 percent and at that significance level, the data passes the test.

Conclusion

The overall project is to introduce the generalized linear model, and we figured out the relationship between three different errors and the amount of samples or spike counts. As the amount of samples or spike counts increasing, we can generate more accurate result. The spike counts is related to the offset and the number of samples. After analyzing the errors, we load several files from daily life into the fitting GLM model to test our previous assumption, which has been proved that they are correct. The last section is to analysis the given data and make a histogram for the inter-spike intervals. The intervals actually fit a Gamma distribution. We have

learned a lot through this project, and we the things we learned about the generalized linear model will be useful in our future research.

Reference

1. RMSE: Root Mean Square Error. (2017, November 14). Retrieved from https://www.statisticshowto.datasciencecentral.com/rmse/
2. Y. (n.d.). Retrieved from https://www.mathworks.com/help/matlab/ref/errorbar.html

Appendices

main_script.m

```
% He Feng & Huihao Chen
clc;
close all;

% Define the variables.
m=0.3;
v=0.1;
s=m+v*randn(1,18000);
t = 0:1:14;
% Given the stim filter and spike filter.
f = 20*exp(-t);
h = -200*exp(-t);
b = -15;

% Generate the discretized stimulus.
stim = zeros(1, 2000);
stim = [stim s];

% Plot the stimulus filter.
figure
subplot(4,1,1);
plot(t,f);
hold on;
xlim([-1,15]);
ylim([-5,25]);
ylabel('stimulus filter');


% Plot the spike filter.
subplot(4,1,2);
plot(t,h);
xlim([-1,15]);
ylim([-3,3]);
ylabel('spike filter');
hold on;

% Plot the stimulus response.
subplot(4,1,3);
plot(stim);
xlim([-1000,22000]);
ylim([-0.2 0.7]);
ylabel('stimulus');

% Plot the spike response.
subplot(4,1,4);
spike = sim_GLM(f,h,b,stim);
plot(spike);
xlim([-1000,22000]);
ylim([-0.1 1.1]);
ylabel('spike response');
```

```
50
51      % Generate a new stim filter and spike filter by using the fitting function
52      % Plot the new stimulus filter on the same plot of the original stimulus
53      % filter.
54 -    [f_fit, h_fit, offset, stats] = fit_GLM(stim, spike);
55 -    subplot(4,1,1);
56
57 -    plot(t,f_fit);
58 -    hold on
59 -    errorbar(t,f_fit,f_fit-f,'.');
60
61
62
63      % Plot the new spike filter on the same plot of the original spike filter.
64 -    subplot(4,1,2);
65 -    plot(t,h_fit);
66 -    hold on
67 -    errorbar(t,h_fit,h_fit-h,'.');
68
```

sim_GLM.m

```
1       % He Feng & Huihao Chen
2       % Building a function which simulate the GLM. It will output the spike
3       % response.
4     function n = sim_GLM(f,h,b,s)
5
6       % Define the variables
7 -     T = length(s);
8 -     d = 15;
9 -     n = zeros(1,T);
10
11 -  for i = (d+1):T
12          % Use a for loop to generate the power which introduced in the specd, and
13          % it will be useful in the following procedure
14 -        power = 0;
15 -        for j = 1:d
16 -           power = power + s(i-j)*f(j) + n(i-j)*h(j);
17 -        end
18 -        power = power+b;
19
20          % Define each element in the spike response one by one.
21 -        n(i) = poissrnd(exp(power));
22 -        n(i) = min(n(i),1);
23 -   end
24
25
26
27 -  end
```

fit_GLM.m

```
% He Feng & Cuihao Chen
% Generating a function to fit the GLM.
function [f,h,offset,stats] = fit_GLM(s,n)
T = length(s);
d = 15;

% Building the input matrix.
A = zeros(T,30);
for i = (d+1):T
    array_s = [s(i-1),s(i-2),s(i-3),s(i-4),s(i-5),s(i-6),s(i-7),s(i-8),...
        s(i-9),s(i-10),s(i-11),s(i-12),s(i-13),s(i-14),s(i-15)];
    array_n = [n(i-1),n(i-2),n(i-3),n(i-4),n(i-5),n(i-6),n(i-7),n(i-8),...
        n(i-9),n(i-10),n(i-11),n(i-12),n(i-13),n(i-14),n(i-15)];
    A(i,:) = [array_s array_n];
end

% Building the output matrix.
B = zeros(T,1);
for j = 1:T
   B(j,:) = n(j);
end

% Generate the coefficients and standard errors by using glmfit function.
[coe,~,stats] = glmfit(A,B,'poisson','link','log');
% Define the offset and the output filters.
offset = coe(1);
f = zeros(1,15);
h = zeros(1,15);
for k = 1:15
   f(k) = coe(k+1);
   h(k) = coe(k+16);
end


end
```

fir_GLM20.m

```
% He Feng & Huihao Chen
% this function is based on the former fit_GLM
function [f,h,offset,stats] = fit_GLM20(s,n)
    T = length(s);
    d = 20;
    A = zeros(T,40);
    % build the matrix A
    for i = (d+1):T
        array_s = [s(i-1),s(i-2),s(i-3),s(i-4),s(i-5),s(i-6),s(i-7),s(i-8),...
                   s(i-9),s(i-10),s(i-11),s(i-12),s(i-13),s(i-14),s(i-15),...
                   s(i-16),s(i-17),s(i-18),s(i-19),s(i-20)];
        array_n = [n(i-1),n(i-2),n(i-3),n(i-4),n(i-5),n(i-6),n(i-7),n(i-8),...
                   n(i-9),n(i-10),n(i-11),n(i-12),n(i-13),n(i-14),n(i-15),...
                   n(i-16),n(i-17),n(i-18),n(i-19),n(i-20)];
        A(i,:) = [array_s array_n];
    end
    % interpose n to get B
    B = n';
    %B = squeeze(B);
    %input = [1:1:T; [array_s array_n];
    %output = n;
    [coe,~,stats] = glmfit(A,B,'poisson','link','log');

    offset = coe(1);
    f = zeros(1,20);
    h = zeros(1,20);
    % assign the values in coe to f_fit and h_fit
    for k = 1:20
        f(k) = coe(k+1);
        h(k) = coe(k+21);
    end

end
```

GLM_perform.m

```
% He Feng & Huihao Chen
% Section 4.3 of the final project
clc;
close all;

m=0.3;
v=0.1;

% Define the ideal filters and other variables.
t = 0:1:14;
f = 20*exp(-t);
h = -200*exp(-t);
b = -15;
d = 15;

% List the time samples.
num_sample = [1000,3000,10000,30000];

figure(1)
%% Display each errors as the function of stimulus length.
% Generate four groups of samples with diverse sample length.
for i = 1:4
    number = num_sample(i);
    offset_error = zeros(1,6);
    new_offset = zeros(1,6);
    mean_standard_stim = zeros(1,6);
    % Generate each length for six trails.
    for j = 1:6

        %% Plot the offset errors as the function of numbers of samples.
        subplot(3,3,1);
        s=m+v*randn(1,number*0.9);
        stim = [zeros(1,number*0.1) s];
        spike = sim_GLM(f,h,b,stim);
        % count the spikes number
        s_counts = size(find(spike == 1));
        [f_fit, h_fit, offset, stats] = fit_GLM(stim, spike);
        % find the absolute value of the difference between the ideal
        % offset and new generated offset.
        new_offset(j) = abs(b-offset);
        semilogx(number,new_offset(j),'b.','MarkerSize',10);
        hold on
        xlabel('Number of Samples');
        ylabel('offset error');

        %% Plot the offset errors as the function of spike counts.
        subplot(3,3,4);
        semilogx(s_counts(2),new_offset(j),'b.','MarkerSize',10);
        hold on
        xlabel('Spike Counts');
        ylabel('offset error');
```

```
52
53          %% Plot the RMSE as the function of stimulus length.
54 -        subplot(3,3,2);
55 -        total_stim = 0;
56 -        sum_stim = 0;
57 -        sum_spike = 0;
58          % Generate the RMSE for stimulus filter.
59 -        for k1 = 1:size(f_fit)
60 -            sum_stim = sum_stim + (f(k1)-f_fit(k1))^2;
61 -        end
62 -        output_stim = sqrt((1/d)*sum_stim);
63 -        loglog(number,output_stim,'b.','MarkerSize',10);
64 🔴        hold on
65
66          % Generate the RMSE for the spike filter.
67 -        for k2 = 1:size(h_fit)
68 -            sum_spike = sum_spike + (h(k2)-h_fit(k2))^2;
69 -        end
70 -        output_spike = sqrt((1/d)*sum_spike);
71 -        loglog(number,output_spike,'r.','MarkerSize',10);
72 -        hold on
73 -        xlabel('Number of Samples');
74 -        ylabel('RMSE');
75
76          %% Plot the RMSE as the function of spike counts.
77 -        subplot(3,3,5);
78 -        loglog(s_counts(2),output_stim,'b.','MarkerSize',10);
79 -        hold on
80 -        loglog(s_counts(2),output_spike,'r.','MarkerSize',10);
81 -        hold on
82 -        xlabel('Spike Counts');
83 -        ylabel('RMSE');
84
```

```
5           %% Plot the mean standard error as the function of stimulus length.
6 -         subplot(3,3,3);
7 -         sum_sderr_stim = 0;
8 -         sum_sderr_spike = 0;
9 -         std_error = stats.se;
0           % Generate the mean standard error for the stimulus filter.
1 -         for g1 = 2:16
2 -             sum_sderr_stim = sum_sderr_stim + std_error(g1);
3 -         end
4 -         output_stim = sum_sderr_stim/d;
5 -         loglog(number,output_stim,'b.','MarkerSize',10);
6 -         hold on
7
8           % Generate the mean standard error for the spike filter.
9 -         for g2 = 22:31
0 -             sum_sderr_spike = sum_sderr_spike + std_error(g2);
1 -         end
2 -         output_spike = sum_sderr_spike/(d-5);
3 -         loglog(number,output_spike,'r.','MarkerSize',10);
4 -         hold on
5 -         xlabel('Number of Samples');
6 -         ylabel('Mean Standard Error');
7
8           %% Plot the mean standard error as the function of spike counts.
9 -         subplot(3,3,6);
0 -         loglog(s_counts(2),output_stim,'b.','MarkerSize',10);
1 -         hold on
2 -         loglog(s_counts(2),output_spike,'r.','MarkerSize',10);
3 -         hold on
4 -         xlabel('Spike Counts');
5 -         ylabel('Mean Standard Error');
6
```

```
    end
end

%% Generating the function by changing offset.
b = [-14, -16, -18];
length = 10000;
for i = 1:3
    new_offset = zeros(1,6);

    for j = 1:6
        s=m+v*randn(1,length*0.9);
        stim = [zeros(1,length*0.1) s];
        spike = sim_GLM(f,h,b(i),stim);
        [f_fit, h_fit, offset, stats] = fit_GLM(stim, spike);
        s_counts = size(find(spike == 1));

        % Generate the offset errors as the function of spike number.
        new_offset(j) = abs(b(i)-offset);

        %% Generate the offset errors as the function of spike counts.
        subplot(3,3,7)
        semilogx(s_counts(2),new_offset(j),'b.','MarkerSize',10)
        hold on
        xlabel('spike counts');
        ylabel('offset errors');

        %% Generate the RMSE as the function of spike counts.
        subplot(3,3,8)
        total_stim = 0;
        sum_stim = 0;
        sum_spike = 0;
        % Generate the RMSE for stimulus filter.
        for k1 = 1:size(f_fit)
            sum_stim = sum_stim + (f(k1)-f_fit(k1))^2;
        end
        output_stim = sqrt((1/d)*sum_stim);
        loglog(s_counts(2),output_stim,'b.','MarkerSize',10);
        hold on
        % Generate the RMSE for the spike filter.
        for k2 = 1:size(h_fit)
            sum_spike = sum_spike + (h(k2)-h_fit(k2))^2;
        end
        output_spike = sqrt((1/d)*sum_spike);
        loglog(s_counts(2),output_spike,'r.','MarkerSize',10);
        hold on
        xlabel('spike counts');
        ylabel('RMSE');

        %% Generate the mean standard error as the function of spike counts.
        subplot(3,3,9);
        sum_sderr_stim = 0;
        sum_sderr_spike = 0;
        std_error = stats.se;
        for g1 = 2:16
            sum_sderr_stim = sum_sderr_stim + std_error(g1);
        end
        output_stim = sum_sderr_stim/d;
        loglog(s_counts(2),output_stim,'b.','MarkerSize',10);
        hold on

        for g2 = 22:31
            sum_sderr_spike = sum_sderr_spike + std_error(g2);
        end
        output_spike = sum_sderr_spike/(d-5);
        loglog(s_counts(2),output_spike,'r.','MarkerSize',10);
        hold on
        xlabel('spike counts');
        ylabel('Mean Standard Error');

    end
end
```

data_exam.m

```
% He Feng & Huihao Chen
% this is the pogram to find filters for the given data
clc;
close all;

t = 0:1:19;
% Given the stim filter and spike filter.
f = 20*exp(-t);
h = -200*exp(-t);

% import the given data
stim1 = importdata('binned_stim_cell_1.txt');
spike1 = importdata('binned_spikes_cell_1.txt');

stim2 = importdata('binned_stim_cell_2.txt');
spike2 = importdata('binned_spikes_cell_2.txt');

% plot and label the graphics for the first data
[f_fit1,h_fit1,offset1,stats1] = fit_GLM20(stim1,spike1);
% calculate the mean square standard error
stimMSE1 = mean(stats1.se(2:21));
spikeMSE1 = mean(stats1.se(26:41));
figure;
subplot(2,1,1);
plot(t,f_fit1);
hold on
errorbar(t,f_fit1,f_fit1-f,'.');
ylabel('stimulus filter');
title('data 1');
subplot(2,1,2);
plot(t,h_fit1);
hold on
errorbar(t,h_fit1,h_fit1-h,'.');
ylabel('spike filter');
xlabel('time');
```

```
% plot and label the graphics for the second data
figure;
[f_fit2,h_fit2,offset2,stats2] = fit_GLM20(stim2,spike2);
% calculate the mean square standard error
stimMSE2 = mean(stats2.se(2:21));
spikeMSE2 = mean(stats2.se(26:41));
t = 0:1:19;
subplot(2,1,1);
plot(t,f_fit2);
hold on
errorbar(t,f_fit2,f_fit2-f,'.');
ylabel('stimulus filter');
title('data 2');
subplot(2,1,2);
plot(t,h_fit2);
hold on
errorbar(t,h_fit2,h_fit2-h,'.');
ylabel('spike filter');
xlabel('time');

```

task6.m

```
% He Feng & Huihao Chen
% this program is to find a fitting distribution
% for the inter-spike interval
clc;
close all;

% load the data
ISIsimulated = importdata('ISIsimulated.txt');
[l,k] = size(ISIsimulated);
% find the inter-spike interval
isi = zeros(l,1);
for i = 1:1:(l-1)
    isi(i) = ISIsimulated(i+1) - ISIsimulated(i);
end
figure;
histogram(isi);

% plot a standard gamma distribution
x = 0:0.1:12;
gam_standard = 800*gampdf(3,2.5,x);
hold on;
plot(x,gam_standard);
title('Data Histogram and Poisson Distribution');

% chi-square test on the isi
pd = fitdist(isi,'Gamma');
h = chi2gof(isi,'CDF',pd);
```