A Computational Study of the Role of Synchrony in Neural Input-Output Relationship

Yangluo Jim Wang
jim.wang@rice.edu

August 25, 2008
Department of Computational and Applied Mathematics
Rice University, Houston, Texas 77005

This report summarizes my project under the mentorship of Dr. Kresimir Josic and Dr. Peter Saggau through the Rice/Texas Medical Center Computational Neuroscience Research Experience for Undergrad (REU) program. This REU is partially supported by the National Science Foundation. This work was done at Baylor College of Medicine, with weekly visits to the University of Houston.

1 Introduction

With the exception of motor neurons, all neurons fire a single output as an action potential only after receiving sufficient inputs from other neurons. For most neurons, the output depends on the spatiotemporal patterns of the inputs. However, the details of this pattern remain poorly understood, largely because it is experimentally expensive to blindly stimulate at a large number of dendritic sites. In Dr. Peter Saggau's lab, we have instruments capable of stimulating individual spines on a dendrite at fast speeds (it only takes 0.02 ms to go from one location to another), but a computation simulation is needed to make the experimental time more worthwhile.

The goal of this project is to computationally generate inputs with statistically quantifiable spatiotemporal patterns, and analyze the output produced by a simple neuron model driven by those inputs. With all the simulations done in MATLAB R2008a, two statistical populations of inputs were studied. These two populations had the same rate and pairwise correlations but differed in their higher-ordered statistics. By jittering the inputs, which disrupted synchrony, we have studied how jitter transformed the inputs into predictable spatiotemporal patterns and analyzed how those patterns affected the output of the model. For example, it was found that increasing jitter decreased firing rate exponentially. The result of these simulations would eventually help guide a similar experiment and provide an expectation of the outcome. However, while the tools for producing the inputs and outputs have been created, more work is still needed to fully analyze the results.

2 Two Population Models for Correlated Neural Inputs

Under the general assumption that only the timing, not the form, of the presynaptic spikes are important, we can model neuronal inputs as Poisson point processes, where if r is the rate of the input, then $r \times dt$ is the probability of a stimulus occurring in a time interval $(t, t + dt)$ of the point process.

First let us define a few terminologies. A *train* is another term that we will use for point process. A *spike* denotes a point in a train where a stimulus occurred. *Interspike interval* (ISI) defines the average time between spikes in a given train. It is easy to check that ISI is inversely related to rate: ISI = 1/rate. In this report, we will use ISI and rate interchangeably since they both define the same concept.

In this project, we concerned ourselves with two different models to generate a population of Poisson spike trains. These models were introduced by Kuhn et al [1].

2.1 Single Interaction Process

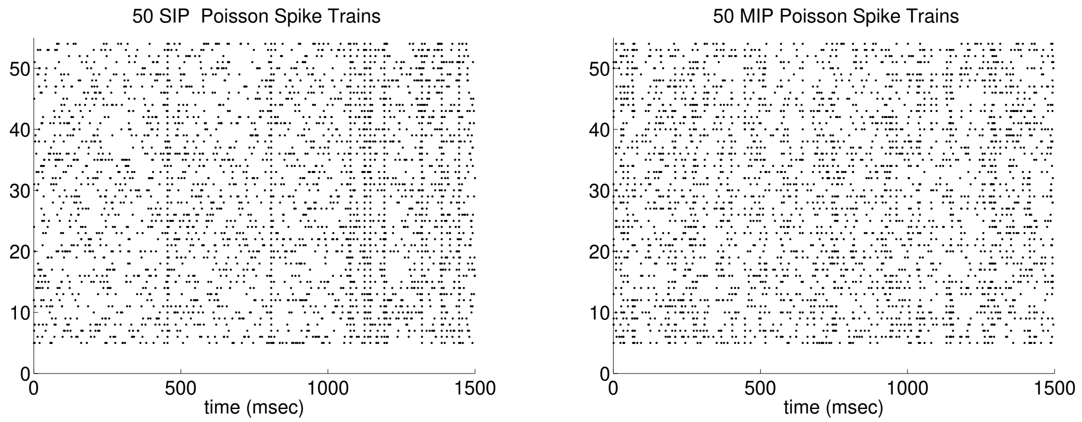
The Single Interaction Process (SIP) represents a population that displays asynchronous activity at some times but bursts synchronously, with all neurons firing, at other times. It is defined as follows:

Let w_u be a Poisson process with rate α. Then each spike train x_i of in a set of N SIP processes is defined by

$$x_i = w_u + w_i, \qquad 1 \leq i \leq N, \tag{1}$$

where each w_i is a Poisson process with rate β. Then the rate of x_i is $\alpha + \beta$, and the pairwise correlation coefficient between two trains in the population is $\alpha/(\alpha + \beta)$ [1].

Figure 1(a) shows a rasterplot of a population of SIP processes. Note that the mother process, w_u, is realized in each individual SIP spike train, introducing a pairwise statistical dependence defined by the correlation coefficient and a total synchrony in the higher orders, involving more than two trains.



(a) Single Interaction Process (b) Multiple Interaction Process

Figure 1: Fifty SIP (a) and MIP (b) spike train populations with identical rates (.04Hz) and pairwise correlations (.15) but different higher-order statistics.

2.2 Multiple Interaction Process

In the second model of spiking population, the Multiple Interaction Process (MIP), only a subset of the spike trains fire synchronously. It is defined as follows:

Let w_u be a Poisson process with rate γ. Each spike train y_i in a set of N MIP processes is defined by random thinning of w_u, where ϵ is the probability that a spike in w_u will also exist in y_i. The rate of y_i is thus $\gamma\epsilon$ and the pairwise correlation coefficient between two trains in the population is ϵ [1].

Figure 1(b) shows a rasterplot of a population of MIP processes. Note that the trains that fire synchronously at a particular time are only a subset of the entire population.

It is easy to check that one can generate SIP and MIP populations with the same ISI and pairwise correlation coefficient. In fact, given a pair of spike trains, either from the SIP or MIP populations (provided that both populations have the same ISI and pairwise correlation coefficient), one cannot determine whether that pair of trains came from the SIP or MIP model. This is because the only difference that distinguishes the SIP and the MIP processes is their higher-order structures. Indeed, on the pairwise level, the SIP and the MIP are exactly the same, but on the higher-order level, the two populations are the opposite extremes of each other, the SIP being highly ordered and its synchrony predictable while the MIP being the "worst case scenario," its higher-order structures entirely unpredictable. Since synchrony involves many trains, we maintained the pairwise statistics the same between the two population models to focus on synchrony in the higher-order structures.

2.3 Disrupting the Synchrony Through Jittering

To study the role of synchrony in neural input/output relationship, we must first disrupt synchrony, but in a statistically quantifiable manner. We do so by jittering the SIP and MIP processes. The algorithm for jitter is as follows:

For each spike at time t in a spike train, pick a random number r from a normal distribution with standard deviation j. Shift that spike from time t to time $t + r$.

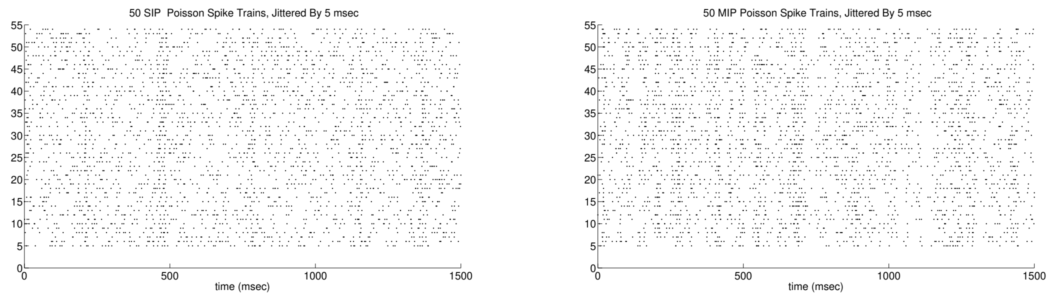
Now we introduce another notation. To *jitter by j* means to jitter a process with a normal distribution of standard deviation j. We will frequently use this notation in the subsequent sections. Figure 2 shows populations of SIP and MIP trains jittered by 5 msec. While synchrony is disrupted in both cases, the order in the SIP is still visible. We chose to jitter with a normal distribution because gaussians are statistically simple and easy to quantify. Furthermore, the cross correlation of two processes, obtained independently by twice jittering a Poisson process with gaussians, is the convolution of the individual normal jittering distributions, which is another gaussian with mean equal to the sum of the individual means and likewise for variance [6], as shown in Figure 3. Figure 4 shows the the role of jitter on the pairwise correlation coefficient between two jittered SIP or MIP processes. Indeed, as jitter increases, the pairwise correlation coefficient decreases exponentially (by the same factor) in both cases; however, the correlation coefficient never reaches zero because statistically, there is still a random chance for there to be a correlation. Since it is difficult to distinguish between the SIP and the MIP processes on a pairwise level, the similarity between the SIP and MIP curves in figure 4 as are expected.

Therefore, the gaussian jittering distribution (mean and standard deviation), cross correlation, and correlation coefficient all help in statistically quantifying the degree of synchrony in these SIP and MIP neural inputs.

3 The Izhikevich Firing Model

3.1 The Equations

A single compartmental neuron model introduced by Eugene Izhikevich [2] was used in this project. The equations are derived by using bifurcation methods to reduce biophysically accurate neural models like the Hodgkin-Huxley type equations [4]. The end result is a



(a) jittered SIP (b) jittered MIP

Figure 2: Fifty SIP (a) and MIP (b) spike train populations jittered by 5 msec. ISI = 25 msec, time step = 0.02 msec, unjittered correlation coefficient = 0.15. Note that the higher-order structures in the SIP, although disrupted, can still be observed.

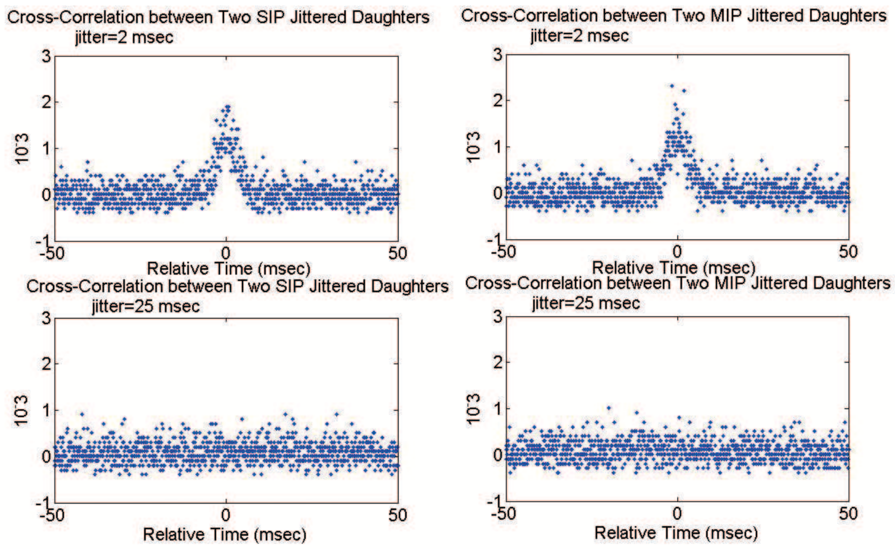


Figure 3: Cross correlation between two trains from the SIP population jittered by 2 msec (A) and 25 msec (C), and the MIP population jittered by 2 msec (B) and 25 msec (D). Pairwise cross-correlation is the same between SIP and MIP. Increasing jitter flattens the cross correlation function. Although flattened, the functions in (C) and (D) are still gaussians.

two-dimensional set of ordinary differential equations of the form

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{2}$$
$$u' = a(bv - u) \tag{3}$$

with the after-spike resetting

$$\text{if } v \geq 30 \text{ mV, then set } \begin{cases} v = c, \\ u = u + d. \end{cases} \tag{4}$$

The parameters a, b, c, and d are dimensionless, and $' = d/dt$, where t is time. v represents membrane potential, and u represents a membrane recovery variable, accounting for the K^+ current activation and Na^+ current inactivation. The neuron is considered to have fired if v reaches 30 mV, at which time v and u are reset according to (4).

Equation (3) was obtained so that v has dimensions in mV and t in ms. This model was

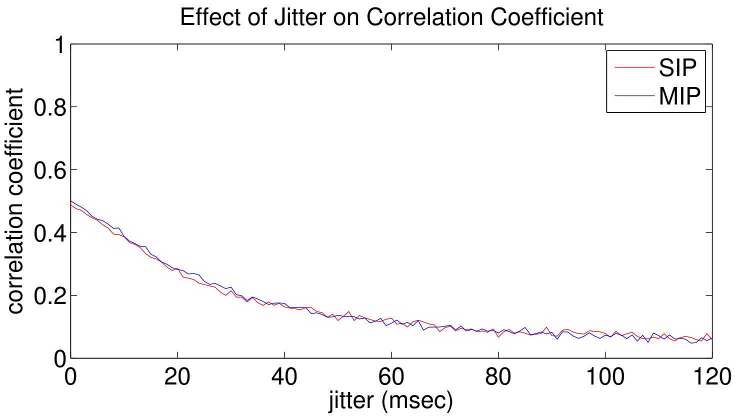


Figure 4: Effect of jitter on pairwise correlation coefficient between two SIP (red) or MIP (blue) spike trains. In both cases, correlation coefficient decreases exponentially with jitter. ISI = 25 msec, time step = 0.1 msec, unjittered correlation coefficient = 0.5, window radius = 50 msec.

chosen because it is computationally simple and can produce a rich range of the spiking and bursting patterns displayed by real neurons simply by changing the parameters a,b,c, and d. The model, fails, however, at capturing subthreshold dynamics. Since we were interested in output firing patterns, the advantages of the model clearly outweighed the disadvantage.

In this project, we studied the output (i.e. firings) of **regular spiking** (RS) neurons, which are presented in later sections. Figure 5 shows the regular spiking (a) and chattering neurons, two kinds of spiking behavior that the Izhikevich model is capable of. More spiking patterns can be found in [2] and [5].

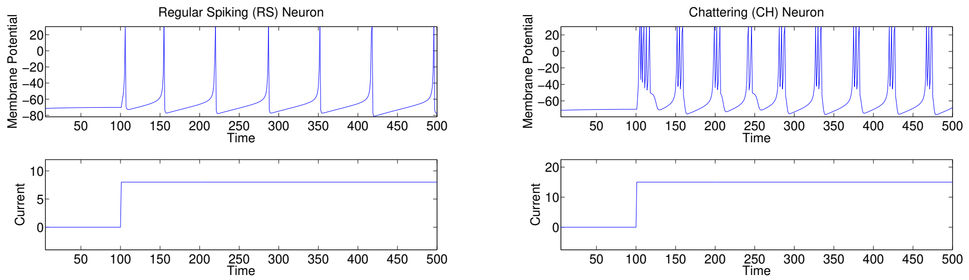


Figure 5: Different firing patterns correspond to different values of the parameters a, b, c, and d in the model. (left) Regular spiking neurons. In this model, $a = 0.02, b = 0.2, c = -65\text{mV}, d = 8$. (right) Chattering neurons, $a = 0.02, b = 0.2, c = -50\text{mV}, d = 2$. The values for c and d means that RS neurons have a deeper voltage reset and larger after-spike jump in u than CH neurons.

3.2 Modifications

Some modifications to the Izhikevich model were made to be more biophysically accurate. Since a single stimulus changes the membrane conductance, g, in an alpha-like function

$$g = g_0(e^{t/\tau_2} - e^{t/\tau_1}), \tag{5}$$

where τ_1 is the rise time and τ_2 the decay time, we modified the model to be conductance based, adding

$$I = g(V - E) \tag{6}$$

as an additional equation to (2) and (3), where E is the reversal potential. We matched our simulations parameters close to the biological parameters for a hippocampal CA1 neuron synapse with AMPA receptors, setting $E = 0$, $\tau_1 = 0.37$, and $\tau_2 = 5$. Figure 6 shows the membrane conductance, current, and voltage change due to a single stimulus. The amplitude is adjusted so that the maximum conductance is 2 μS, which is agreeable with [7].

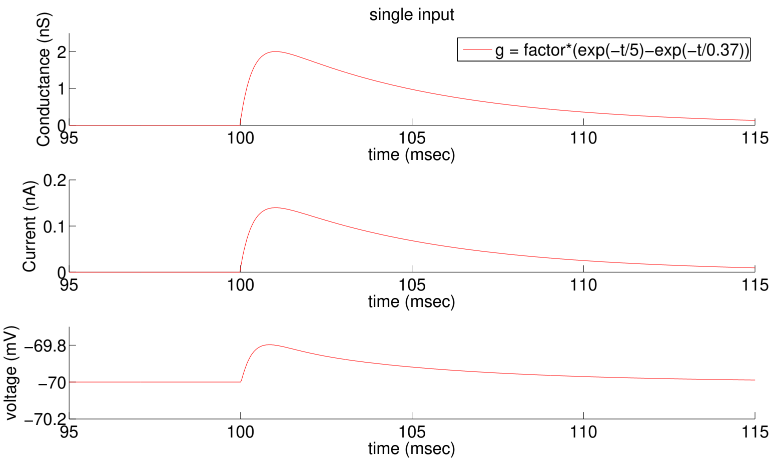


Figure 6: Response of a regular spiking Izhikevich model to a single input. The factor is the number necessary to make the conductance peak at 2 μS.

Notice that the Izhikevich model is a deterministic model, that is, a set of inputs will always produce the same output no matter how many times the simulation is run. This is due to the deterministic nature of (2) and (3) and the hard reset in (4).

4 Results

The effect of jittering both population models were tested on the Izhikevich regular spiking neuron model. We stimulated the cell three times at its one compartment with 50 SIP inputs, jittered by 0, 5, and 10 msec, respectively. Since each stimulus generated a conductance change in the form of an alpha-like function and a subsequent excitatory post synaptic potential (EPSP), recorded by the membrane potential v, we summed up those changes at the membrane. If the summed EPSP was greater than 30 mV, as according to equation (4), we considered the cell to have fired. The same simulation was also run for 50 MIP inputs. Figure 7 shows plots of the membrane conductance, current, and voltage changes during the first 1500 msec for the SIP and MIP simulations.

4.1 Mean Firing Rate

We investigated the effect of jittering SIP and MIP populations on the firing rate of the regular spiking Izhikevich neuron model (Figure 8). The firing rates of both SIP and MIP populations were approximately equal at jitter = 0, and approached the asymptote at zero exponentially as jitter increases. However, the firing rate of MIP approached the asymptote faster than that of the SIP. Indeed, the second derivative of the MIP firing rate was greater than that of the SIP. The result was expected because the synchrony in the SIP involve the entire population, as opposed to a subset of the population in the MIP case, so jittering the

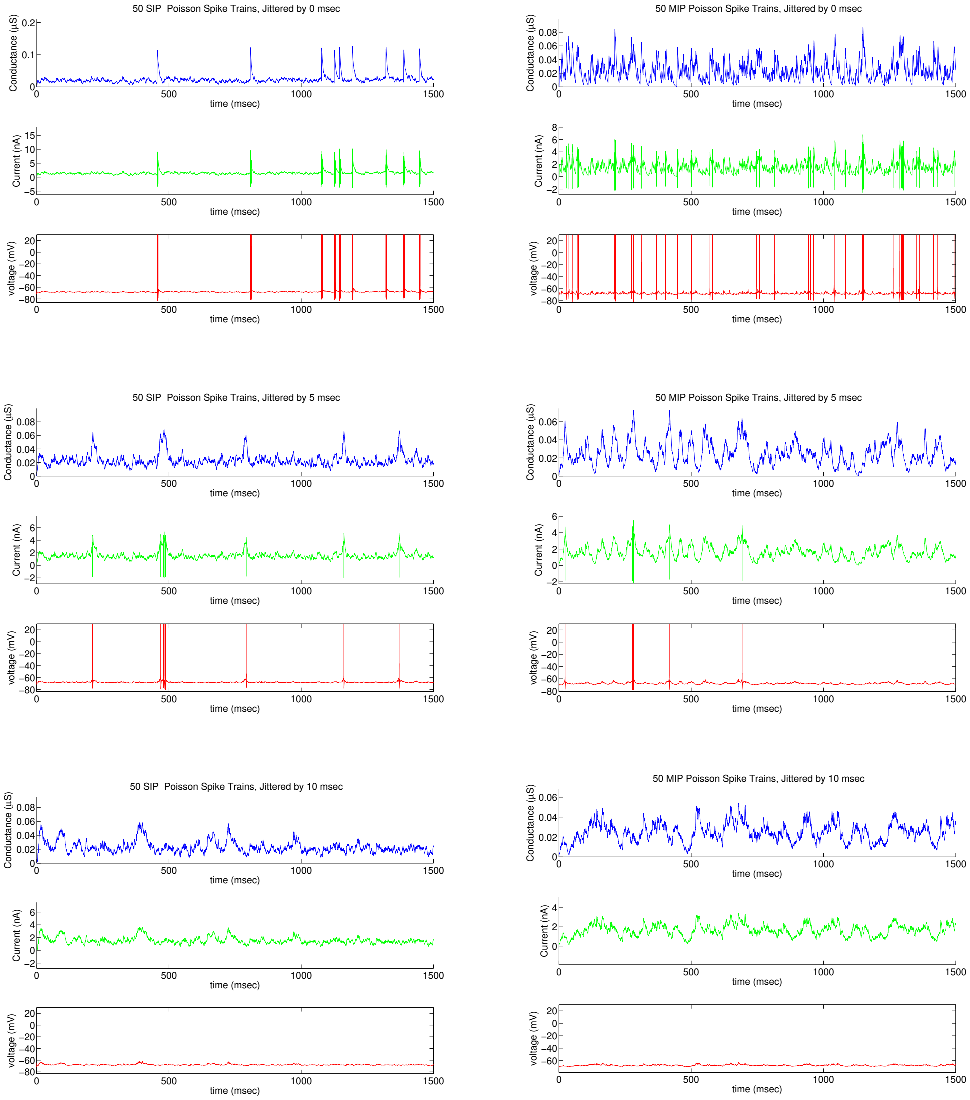


Figure 7: Response of a regular spiking Izhikevich model to 50 input spike trains, jittered by various times. In both the SIP and MIP, as jitter increased from 0 to 5 to 10 msec, the number of action potentials decreased. Total time = 15000 msec, time step = 0.02 msec, unjittered pairwise correlation coefficient = 0.15, for both the SIP and MIP models.

SIP synchrony will have a less profound effect on the population, the summed membrane conductance, EPSP, and ultimately the output.

5 Summary and Discussion of Future Work

The work this summer provided the tools (MATLAB programs) to computationally study the role of synchrony in neuronal input-output relationship. Indeed, we generated two types of neural inputs with well-defined spatiotemporal patterns, which have the same pairwise correlation but different higher-ordered correlations. The degrees of synchrony in these spatiotemporal patterns are statistically quantified via the mean and standard deviation of the normal jittering distribution, the cross correlation, and the correlation coefficient. By changing a computationally simple yet diverse neuron model from current-based to conductance-based, we added biophysical accuracy to our simulations. Finally, we linked the inputs with

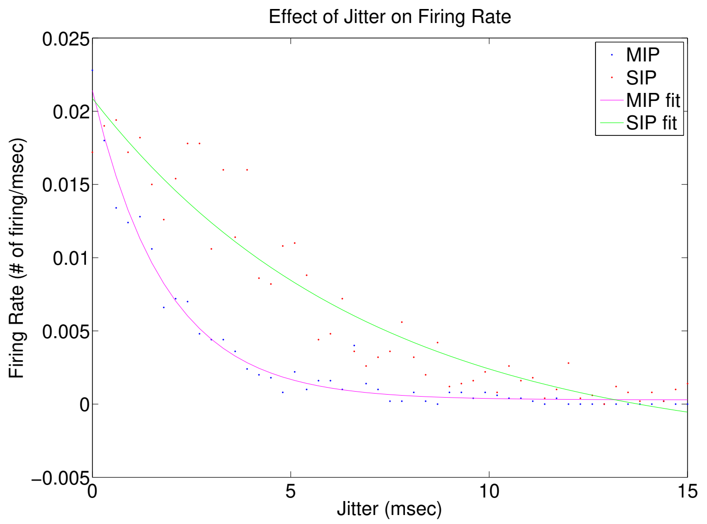


Figure 8: Effect of jittering SIP and MIP populations on firing rate. Fifty spike trains, Total time = 5000, time step = 0.02 msec, unjittered pairwise correlation coefficient = 0.15, regular spiking Izhikevich neuron model.

the neuron model to produce outputs. This work left a foundation where different kinds of experimental setups can be simulated. One experimental setup was investigated this summer, and it was found that as jitter increased, the output firing rate decreased exponentially. Furthermore, the firing rate of the MIP process decreased faster than that of the SIP. In an effort to understand synchrony, more work is required in the future to study other setups and mathematically analyze the results. For example, since the Izhikevich model is a deterministic model, one might add some noise (independent Poisson processes) to the SIP (MIP) inputs and ask the question: "How much jittering is necessary to make the output not reliable? That is, at which point of jitter will the output deviate so much from the unjittered output so as to make the two outputs unrelated?" Of course, one must first define a lower bound for the amount of correlation between the jittered and unjittered outputs that would classify the former as being "reliable."

6 Special Thanks

I would like to thank Brandon Weislak for his collaboration in writing the code that produced the inputs, Bradley Losavio, Keith Kelleher, Dr. Kresimir Josic, and Dr. Peter Saggau for their dedicated mentorship, Robert Rosenbaum for writing some of the programs involving spike trains, Dr. Steve Cox for getting me interested in Computational Neuroscience and organizing this program, Baylor College of Medicine and University of Houston for an enjoyable workplace, and the National Science Foundation for partially supporting this work with NSF REU Grant DMS-0755294.

Appendix

Below are two manuals briefly describing the .m programs used to generate the results of this report.

```
This file briefly describes what each program does.  For more information on what a
program does and how to use it, just enter "help program_name" in the MATLAB command
```

line.

Please refer to my report for a detailed description of the project. Feel free to contact me at chesswonk@gmail.com if you have any questions - Jim Wang.

A small portion of the program descriptions and programs were written by Robert Rosenbaum.

For description of programs specific to the Izhikevich model, see the other readme file below this one.

Note: many of the .m programs will have "_st" as part of the name. This means the Poisson processes used in these programs are in spike times (a vector of spike occurrence times). These programs require less memory than their counterparts without the "_st" ending, which express Poisson processes as spike trains (a binary vector detailing whether a spike occurred or not at every time step).

SCRIPTS:

Bauerle_n_3.m
Generates 3 Bauerle daughters, jittered by multivariate normal dist. plots cross correlation between 2 such daughters, and fits the cross correlation with a Gaussian function, whose mean is sum of the individual means and variance is sum of the individual variance minus the covariance

Bauerle_n_ind_exponential.m
Generates 3 Bauerle daughters, jittered by independent exponential dist. Plots cross correlation between 2 such daughters

checkpoisson.m
checks to make sure Poisson.m returns a Poisson distribution

convtest.m
Written by Keith. Tests convolution?

Example1.m
Generate a pair of Poisson spike trains with a Gaussian cross-correlation

Example2.m
Create a renewal process and plot its calculated isi distribution. Also calculate its autocorrelation function.

run_cc_versus_jitter.m
graphs the effect of jitter on correlation coefficient of input

run_cc_versus_winrad.m
script that graphs the effect of window radius on correlation coefficient of input

run_find_correlation_between_jittered_daughters.m
plotting the cross correlations between two jittered sip or mip daughters

run_find_correlation_between_mother_and_jittered_daughter.m
plotting the cross correlations between mother process and sip or mip jittered daughter. This result is not that interesting.

run_find_correlation_unjittered_daughter_and_jittered_version.m
plotting the correlations between the sip or mip unjittered daughter and its jittered version. Again, this result is not that interesting.

run_make_inputs_data.m
Makes input data files for mip and sip processes, data stored in Data folder. This is THE script file that makes all the input data for the Izhikevich model.

run_rasterplot_SIP_MIP.m
Generates raster plots of mother process and n SIP/MIP daughters and displays sum.

run_rasterplot_SIP_MIP_with_jitter_and_cross_corr.m
Generates a raster plot between a mother spike train and SIP/MIP daughters with small and large jitters, and cross correlagrams between the two daughter processes.

test_Bauerle.m
tests the Bauerle algorithm for case $n = 3$ by fitting cross correlation of two daughter processes with a gaussian function

test_cc_versus_jitter_for_input.m
graphing correlation coefficient of input files versus jitter; tests to make sure the input files are as expected

test_cc_versus_winrad_for_input.m
graphing correlation coefficient of input files versus window radius; tests to make sure the input files are as expected

test_cross_corr2.m
testing cross_corr2.m versus cross_corr.m to make sure they produce the same result

test_input_files.m
more tests on input data files: computing average isi, cross correlation and correlation coefficient between 2 daughters, and visualizing data

test_Poisson_st.m
test Poisson_st.m by computing average isi and checking the boundary

-----

FUNCTIONS:

convert_spike.m
Convert spike time vector into spike train vector

corr_coeff.m
Computes the correlation coefficient between to spike trains.

corr_coeff2.m
computes the correlation coefficient between two vectors

cov_coeff.m
Computes the covariance coefficient between to spike trains (i.e., the covariance of the spike counts divided by the window size).

cross_corr.m
Compute the cross correlation function of trains p1 and p2

cross_corr2.m
Compute the cross correlation function of spike time vectors p1 and p2

CV.m
Computes the coefficient of variation of a spike train.

discrete_rv.m
Takes a vector of weights (non-negative numbers) and returns a realization of a discrete rv with pmf given by the normalized weights.

easy_rasterplot.m
Creates a raster plot of any number of spike trains.

Fano_factor.m
Calculates the fano factor of a spike train over a given window size.

findcorr.m
Generates SIP or MIP daughters from a mother Poisson process and computes the correlation among the daughters and between daughters and mother

fitgauss.m
fit a gaussian curve "f" to experimental data

gen_SIP_MIP.m
Generates SIP or MIP daughter spike trains from a mother Poisson train process

gen_SIP_MIP_st.m
Generates SIP or MIP daughter spike times from a mother Poisson spike time process

isi_density.m
Estimates the interspike interval density of a spike train.

jitter.m
Returns a 'jittered' version of a spike train.

jitter_st.m
Return a 'jittered' version of the spike time process

make_identical_with_jitter.m
write identical spike trains/times with various jitter into the Data folder, in .sts format

make_SIP_MIP_with_jitter.m
write sip or mip spike trains/times with various jitter into the Data folder, in .sts format

makejitter.m
make jittered version of a Poisson train and store them in directory

makejitter_SIP_MIP.m
jitters the MIP or SIP daughters made from gen_SIP_MIP.m

makejitter_SIP_MIP.m
jitters the spike time MIP or SIP daughters made from gen_SIP_MIP_st.m

mip.m
takes in a mother poisson train and generates MIP daughters

mip_st.m
Takes in a mother Poisson spike time vector and generates spike time vectors of MIP daughters

num_spike_winrad.m
where each element represents the number of spikes in the corresponding spiking train within that window size as the window slews across the trains

num_spike_winrad_st.m
reads two vectors of spike times with given window radius and produce two vectors where each element represents the number of spikes in the corresponding spiking time vector within that window size as the window slews across the spike time vectors

plot_processes.m
plots spike times for train input x into current axes

plot_processes_for_rasterplot_with_different_colors.m
plots spike times for train input x into current axes in DIFFERENT COLORS

plot_processes_st.m
plots spike times for input spike time x into current axes

plot_processes_with_mother.m
plots spike times for input x (the mother train is the last row of x) into current axes

plot_processes_with_sum.m
plots spike times for input x into current axes; also plots the sum in a subplot

Poisson.m
Creates a poisson spike train. This function is faster than using renewal.m with exponentially distributed interspike intervals.

Poisson_st.m
Creates a poisson spike time vector.

randnum.m
Generates a random number from a specified (discrete) distribution.

randraw.m
Random variates generator

rasterplot.m
Creates a raster plot of any number of spike trains. The input format for this program is non-intuitive. Use easy_rasterplot.m instead.

reflected_exp.m
Exponential Distributed Random Numbers reflected across y-axis with probability $\frac{1}{2}$

renewal.m
Generates a renewal process with a given ISI density.

sip.m
Creates a set of spike trains where some proportion of the spikes are synchronous (throughout all spike trains).

sip_st.m
generate a set spike time vectors where some proportion of the spikes are synchronous (throughout all spike time vectors)

thin.m
Thins a spike train by deleting some proportion of its spikes.

thin_st.m
Thins a spike time vector by deleting some proportion of its spikes.

u_corr_coeff.m
Calculates the uncorrected correlation coefficient between two spike trains. Uses the areas of the correlation functions without correcting. This is fine when looking at

correlations over large windows.

u_corr_coeff.m
Compute the uncorrected correlation coefficient between spike trains p1 and p2

u_cov_coeff.m
Calculates the uncorrected covariance coefficient between two spike trains.

u_var_coeff.m
Calculates the uncorrected variance coefficient of a spike train.

var_coeff.m
Calculates the variance coefficient of a spike train.

This file briefly describes the programs SPECIFIC to the Izhikevich model. For more information on what a program does and how to use it, just enter "help program_name" in the MATLAB command line.

Please refer to my report for a detailed description of the project. Feel free to contact me at chesswonk@gmail.com if you have any questions - Jim Wang.

The Izhikevich model can be described by

```
v' = .04v^2+5v+140-u+I
u' = a(bv-u)
if v >= 30, set v = c; u = u+d;
a,b,c,d are constants, v is membrane potential, u is membrane recovery
variable, I is current
```

You can visit www.izhikevich.com for more information on the Izhikevich model.

For description of programs NOT specific to the Izhikevich model, see the readme file in the yw folder.

SCRIPTS:

bifurk.m
An eigenvalue test for bifurcation. By Keith Kelleher

Izi.m
Runs the Izhikevich model and plots the voltage

Izi_3D_freq_vs_b_vs_I.m
Plots a 3D graph of Izhikevich firing rate vs b vs current

Izi_conductance.m
test #1 for conductance based Izhikevich model

Izi_constant_current.m
injects constant current to find input resistance and membrane time constant for two neurons modeled by the Izhikevich model

Izi_simple_net.m
Written by Eugene Izhikevich. See www.izhikevich.com

run_cc_of_output_vs_jitter.m

drives the model twice for two populations of the same process with the same jitter and compare correlation coefficient of the two outputs versus jitter

run_izi_conductance.m
runs the Izhikevich model for a population of processes and plots the resulting conductance, current, and output voltage as a function of time

run_izi_crosscorr_input_output.m
runs the Izhikevich model for a population of processes and plots cross correlation between the summed inputs and the output

run_izi_num_fired_vs_jitter.m
runs the Izhikevich model for a population of processes and plots the number of firing versus jitter; also varying the time constant to see how that effects the plot

run_izi_two_cell_analysis.m
drives the model twice for two populations of the same process with the same jitter and plots cross correlation between the two summed inputs and the cross correlation between the two outputs

test_Izi_potential_for_single_input.m
Runs Izhikevich conductance model for a single input. Plots the changes in conductance, current, and voltage. Use this script to determine the values for tau1, tau2, and pulsepeak (variables for the double exponential function - lines 15-17) that fit your experiment.

test_Izi_simple_model.m
verifying Izhikevich's paper on "Simple Model of Spiking Neurons;" generates different firing patterns by changing the parameters a, b, c, d, and I.

test_num_fire_vs_current.m
used to verify firing models as Class 1 or Class 2, as defined by Izhikevich; Class 1 neurons will fire at low frequencies, while Class 2 will not.

FUNCTIONS:

alpha.m
creates an alpha function

alpha_over_spike_train.m
convolves an alpha function with a spike train

convconductance.m
convolves double exponential function with input spike train. The spike train is typically the sum of a population of spike trains. Convolving yields a conductance vector that is fed into the Izhikevich model, with the additional equation $I = g(V - E)$. This is program is more efficient than sumconductance.m

double_exp.m
creates a double exponential function of the form $e^{-t/\tau_2} - e^{-t/\tau_1}$

izhikevich_which_model.m
Written by Eugene Izhikevich. See www.izhikevich.com

izi_conductance_sptrain.m
Given a conductance vector g, runs the Izhikevich model equipped with the equation $I = g(V - E)$

```
izi_conductance_sptrain_conv.m
```
Convolves input train and runs the Izhikevich model equipped with $I = g(V - E)$. DO NOT USE THIS FUNCTION because it is more computationally costly than a combination of convconductance.m and izi_conductance_sptrain.m

```
sumconductance.m
```
given a set of input times, convolve those inputs with conductance alpha functions, then sum them up to produce the final conductance. This method is the same, but not as efficient, as summing up the inputs first and then convolving the sum using convconductance.m

References

[1] A. Kuhn, A. Aertsen and S. Rotter, "Higher-order Statistics of Input Ensembles and the Response of Simple Model Neurons," *Neural Computation*, vol. 14, pp. 1569-1572, Nov. 2003.

[2] E. M. Izhikevich, "Simple Model of Spiking Networks," *IEEE Trans Neural Networks*, vol. 14, pp. 1569-1572, Nov. 2003.

[3] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons," *IEEE Trans Neural Networks*, vol. 15, pp. 1063-1070, Sept. 2004.

[4] E. M. Izhikevich, *Dynamical Systems in Neuronscience: The Geometry of Excitability and Bursting*, The MIT Press, 2007.

[5] E. M. Izhikevich, www.izhikevich.com. 9 May, 2003.

[6] D. R. Cox, V. Isham, *Point Processes*, Chapman & Hall, 1980.

[7] T. A. Benke, A. Luthi, J. T. R. Isaac, "Modulation of AMPA receptor unitary conductance by synaptic activity," *Nature*, vol. 393, pp. 793-797, Jun. 1998.