Computational Neuroscience

Final Examination

# Exercise 1) Izhikevich Model Neurons

## A) For I(t)=10mA, find a set of parameters (a, b, c, d) such that the neuron fires at f~50Hz in “fast spiking” mode as shown. Plot V(t) and ISI (inter-spike-interval) histogram for 500ms.

I had no idea about the approximate scale of each parameter, so I first searched for the paper explaining the Izhikevich model. In the paper written by Izhikevich himself[[1]](#footnote-1), parameter values that are used to explain some neurons are introduced as below:

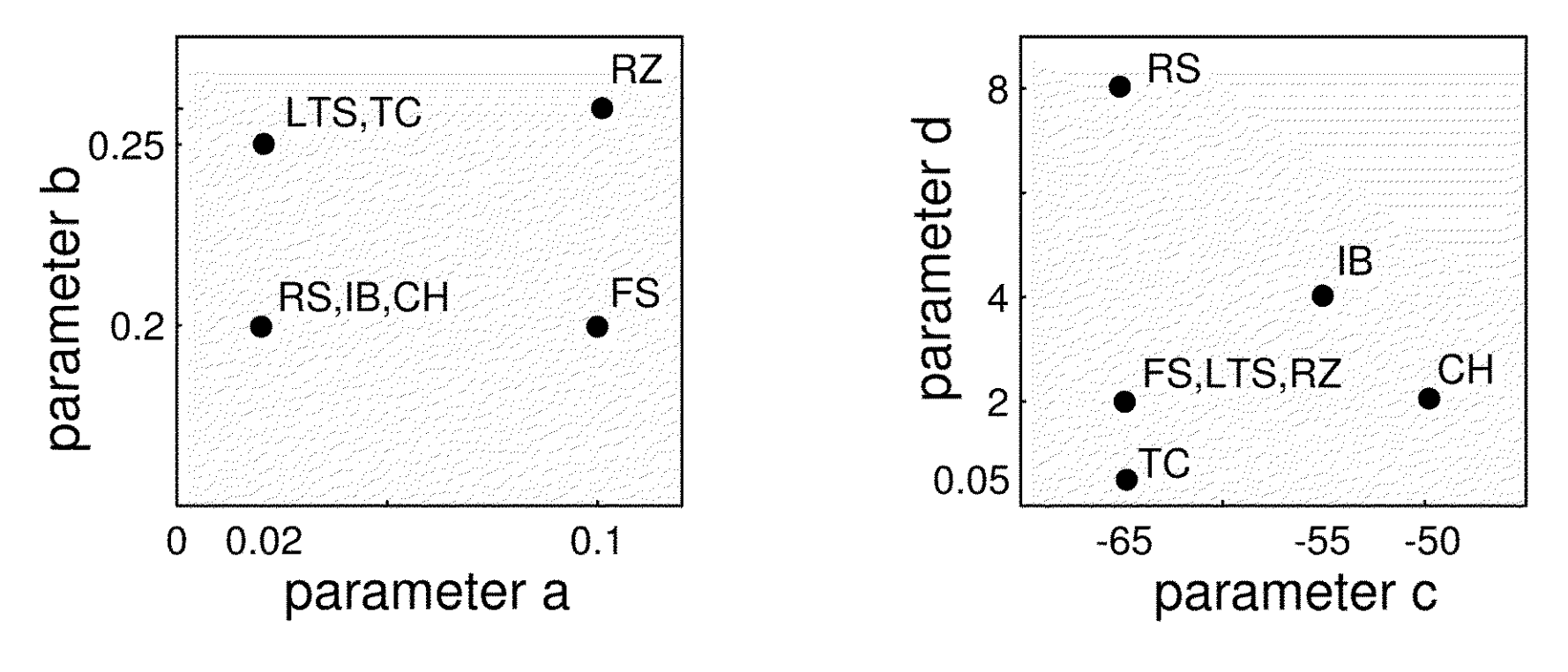


Figure . The examples of parameters a, b, c, d for many different type of neurons.

Then what exactly do these parameters mean? The Izhikevich model follows two equations:

Since the resting potential of the neuron is about -65mV, the initial value of is also negative. Thus, the voltage would increase monotonically. However, since the voltage value is negative, until some moment, would be a negative value. (i.e. -65\*0.2-(-4)=-13+4=-9) Thus, decreases monotonically until . When the voltage exceeds this condition, starts to increase. However, there are additional rule in the Izhikevich model:

The parameter is added to , enabling to increase with a big leap. Usually, c is a voltage value near the resting membrane potential. value thus becomes negative again, and starts to decay again. From this, we can predict that would have a form of periodic form, where decides the decaying rate. If is large, that means starts to increase when is large enough. (i.e. when =0.2, =-4, starts to increase when =-20mV, whereas it starts to increase when =-16mV if =0.25) Thus, b can be considered as sensitivity. Parameter , in the other hand, is the reset value of membrane voltage when spike is generated, and is the parameter enabling to increase.

First, I used the parameter for ‘FS’ (Fast Spiking) which was suggested in Izhikevich’s paper. And obtained the result as below:

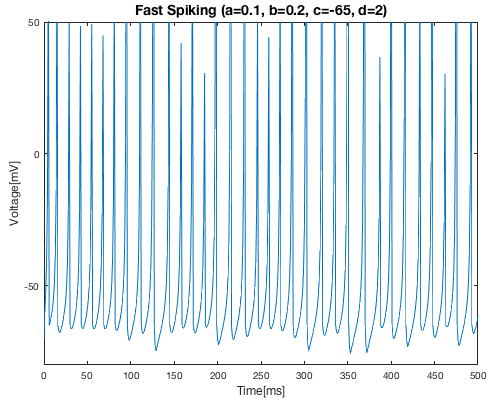


Figure . Voltage plot of the neuron based on Izhikevich’s model, where a=0.1, b=0.2, c=-65, and d=2. The shape is quite similar to what we want to obtain, but the firing rate seems to be too high.

Figure/%5BProb1a%5D%20a=0.1%20Spiking%20Rate.png

As we expected, the firing pattern showed a very similar pattern with the figure that was given in the problem. However, the firing rate was 68Hz, which was too high. Then what value should we change? Based on my interpretation about the parameters, I decided to change parameter . If is set to a smaller value, would decrease in a slower rate, and thus the time for to generate a spike would increase. When I changed value to 0.04, the result was as below:

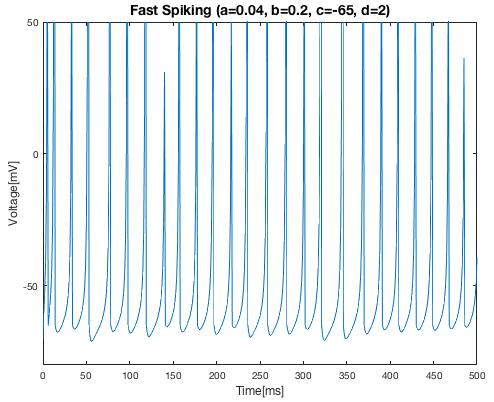


Figure . Voltage plot of the neuron based on Izhikevich’s model, where a=0.04, b=0.2, c=-65, and d=2. The shape is quite similar to what we wanted.

Figure/%5BProb1a%5D%20a=0.04%20Spiking%20Rate.png

As I expected, the firing rate decreased, and it became exactly 50Hz. The firing pattern maintained its shape. Now, I obtained the spike timing by finding moments where .

Then, inter-spike-interval (ISI) histogram was obtained as below:

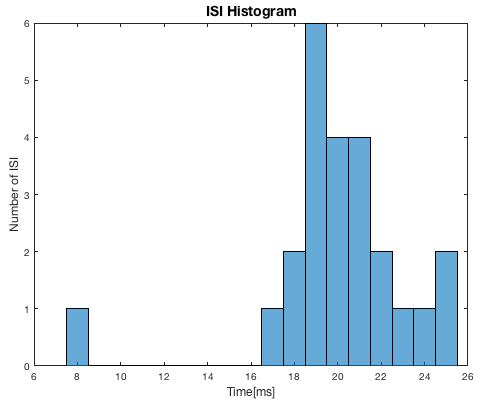


Figure . The ISI histogram of fast spiking neuron model. Most of the inter-spike-interval was around 20ms, which is the expected inter-spike-interval for a constant firing rate neuron with 50Hz.

As shown in Figure 4, most of the inter-spike-interval was around 20ms. This is the expected inter-spike-interval of a neuron with a constant firing rate, 50Hz.

## B) For I(t)=10mA, find a set of parameters (a, b, c, d) such that the neuron fires at f~50Hz in “bursting” mode as shown. Plot V(t) and ISI histogram for 500ms.

Previously, I made a neuron model with ‘fast spiking’ pattern. Then what should I do to obtain a ‘bursting’ pattern? Let’s think about the meaning of the parameters. What would happen if the initializing value of voltage is little bit higher than normal? (such as -50mV) Assume that is around -4, and =0.2. Then, would be -6, thus would decrease, and thus obtains larger force to increase. However, increasing increases , thus leading to smaller force to work on to increase its value.

Now imagine the case where the initializing value of is larger. Larger leads to larger , thus giving force to decrease. But what we also have to consider is the fact that it became much easier for the voltage to reach the spike generating threshold. Thus, much larger would be needed for to be ‘pulled down’ to a value where the neuron stops to make spikes. After the neuron comes to the resting state, would start to decay, and at some moment, would decrease to a value enough to force back the neuron to a spiking state. Until becomes large enough, the neuron would generate spikes consistently, and then drop back to the resting state when reaches some value. What would you call this spiking pattern? Yes, a ‘bursting’!

To confirm whether my prediction was right, I simply changed the parameter ’s value from -65 to -50. Then, I got a result as below:

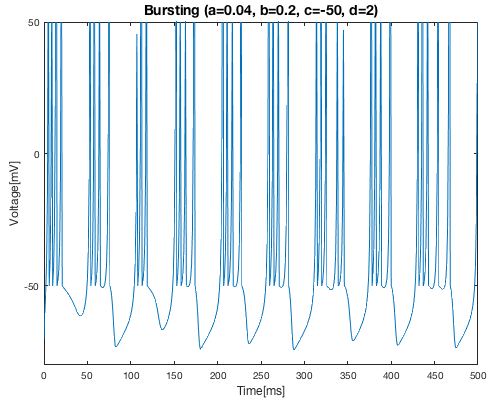


Figure . Voltage plot of the neuron based on Izhikevich’s model, where a=0.04, b=0.2, c=-50, and d=2. The shape is quite similar to what we wanted. (the bursting pattern) However, the number of spikes are larger than what we want.

As we can see in Figure 5, I was right! Simply changing the parameter changed the firing pattern to a bursting pattern. However, the number of spikes in this case was too large. The number of ‘burst clusters’ are nine in this case, but what we want is eight of them. (Also we have to have a firing rate near 50Hz) Using the same logic as in Prob1a, reducing the parameter value might increase the interval between burst clusters, and thus enabling us to obtain only eight of them.

Thus, I simply changed from 0.04 to 0.02, and obtained a result as below:

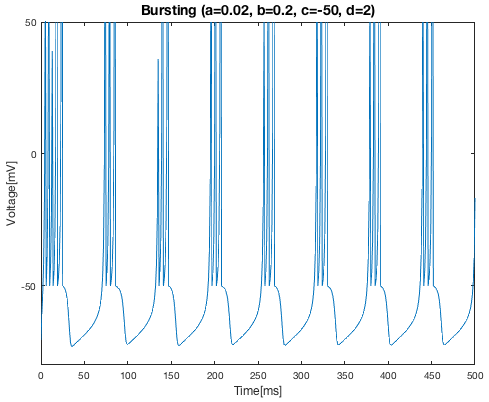


Figure . Voltage plot of the neuron based on Izhikevich’s model, where a=0.02, b=0.2, c=-50, and d=2. The shape is quite similar to what we wanted. (the bursting pattern) However, the firing rate was 52Hz, which was little bit more than I expected.

Now the result seems much more similar with the one that is given in the problem. However, the firing rate for this model was,

Figure/%5BProb1b%5D%20a=0.02%20d=2%20Bur%20Firing%20Rate.png

I want the firing rate to be little bit smaller, and also want to stretch the voltage graph. (If you see carefully, the time interval between each burst cluster should be little bit longer.) This time, I increased the parameter instead of . If the parameter increases, would increase faster, thus reaching the time for forcing the neuron to transit to resting state more quickly (reducing number of spikes per burst) and also lengthen the time for the neuron to transit back to the bursting state. (lengthen the interval between burst cluster) To achieve this, was changed from 2 to 2.5, and the result was as below:

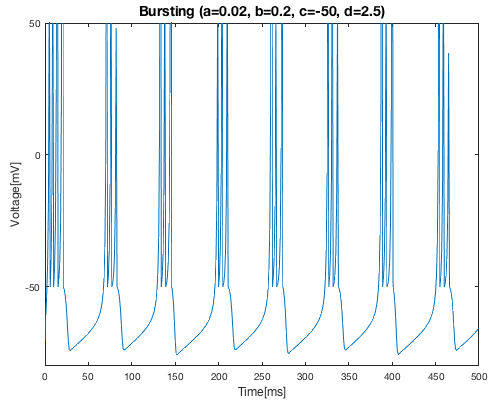


Figure . Voltage plot of the neuron based on Izhikevich’s model, where a=0.02, b=0.2, c=-50, and d=2.5. The shape is quite similar to what we wanted. (the bursting pattern)

Figure/%5BProb1b%5D%20a=0.02%20d=2.5%20Bur%20Firing%20Rate.png

Hurrah! We obtained the exact firing rate that we wanted, and the bursting pattern became more similar to what we wanted. Now, I obtained the ISI histogram, as same as for Prob1a.

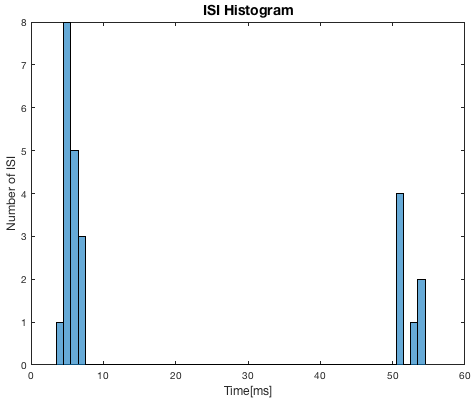


Figure . The ISI histogram of bursting neuron model. Most of the ISI were less than 10ms (ISI between spikes in one burst) whereas seven of them were larger than 50ms. (ISI between spikes in different burst)

As shown in Figure 8, the ISI histogram was very different from the one obtained from fast spiking model: the ISI in fast spiking model were mostly values near 20ms, whereas the ISI in bursting model had two different types of ISI. Short ISI, which is ISI between spikes in same burst, and long ISI, which is ISI between spikes in different burst.

## C) In one figure, make a raster plots of spikes in a and b, with different colors. Plot the accumulated histogram of spike number for both cases.

After obtaining the spike timings of fast spiking neuron and bursting neuron, we can obtain a raster plot as below:

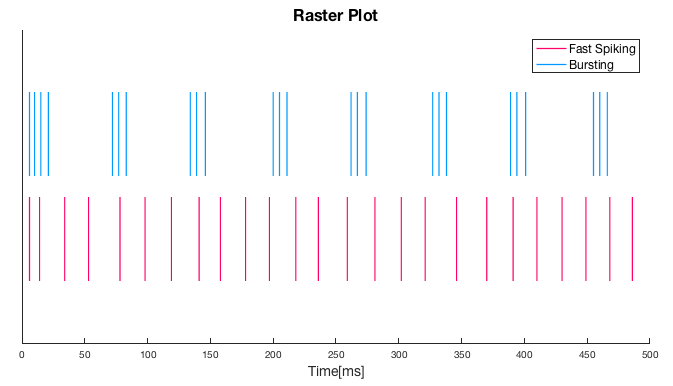


Figure . Raster plot of fast spiking (bottom) and bursting (top) neurons. Both neurons have 25 spikes in 500ms time length, thus the average firing rate of both neurons are 50Hz. However, the spike pattern is significantly different.

Then, I selected the bin width as 10ms, and counted the number of spikes in each bin. (The reason why I chose the bin width as 10ms is because making the bin width too small would show too much information – thus making it harder to understand difference between neurons – whereas too large bin width wouldn’t show any difference between the two neurons.) The number of spikes for each bin was plotted in an accumulated bar graph.

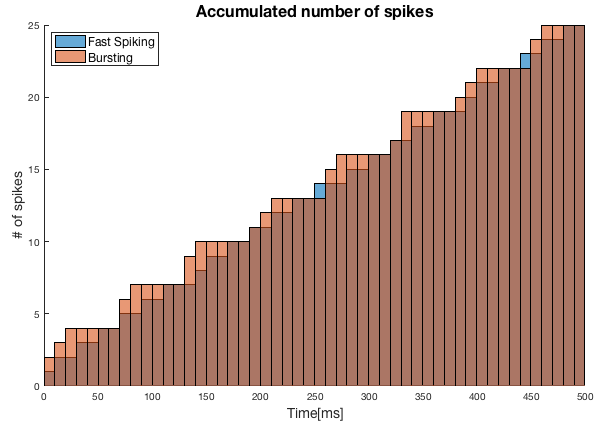


Figure . The accumulated bar graph of spikes for both fast spiking (blue) and bursting. (orange)

For bursting neurons, the number of spikes increased relatively quickly than the fast spiking neuron in some range, and became constant for a while. Fast spiking neurons, on the other hand, tended to increase monotonously. Thus, as shown in Figure 10, the accumulated number of spikes for bursting neurons was larger in certain range, (i.e. time=0~40ms) but since it stops to increase for a while, the fast spiking neuron’s accumulated number of spikes catches up. (i.e. time=50~60ms) This pattern is shown for the whole time range, and this is because bursting neurons show many spikes for a short time, and then have no spikes for a while, whereas fast spiking neurons have relatively consistent interval between spikes. (consistent ISI)

## D) In a and b, mean firing rates are the same, but instantaneous firing rates are different. Estimate the instantaneous firing rate of two cases.

Using the convolution function, we can apply a Gaussian filter to the firing rate. However, with a limited amount of neurons, the exact firing rate of the neuron is impossible to be accurately determined. Thus, using a ‘kernel’ (in our problem, the Gaussian filter), we can estimate the firing rate of the neuron. However, even after deciding the function that we will use as the Kernel window, a question remains: what should be the optimal width of the filter? Let’s say that the real firing rate of the neuron is , and the estimated neuron is . Since we are using the Kernel to estimate the firing rate, what we would expect is for the estimated neuron’s firing rate to be similar with the real firing rate as much as possible.

This means that the mean integrated square error between the real firing rate and the estimated firing rate,

should be minimum. This can be re-written as

The first component does not depend on the choice of the Kernel (since it is the ‘real’ firing rate). So, it can be considered as a constant. To minimize the mean integrated square error,

should be minimized. When we set the Kernel window to be,

then the estimated firing rate is

where is the firing function: , where is the spike generation time. Using the fact that , the first component of the cost function can be re-written as

Using the general decomposition of the covariance of two random variables,

Then, the second component of this can be re-written as

The third equality was derived by using the fact that spikes are independent in Poisson point process. Thus,

Now, using these facts, the cost function can be written as

now, using the approximation that , the cost function can be estimated as

where

and is the total number of spikes.[[2]](#footnote-2) Based on this theory, the cost function was generated in MATLAB. Then, for fast spiking neuron and bursting neuron, optimal kernel width was obtained and the instantaneous firing rate was obtained as below:

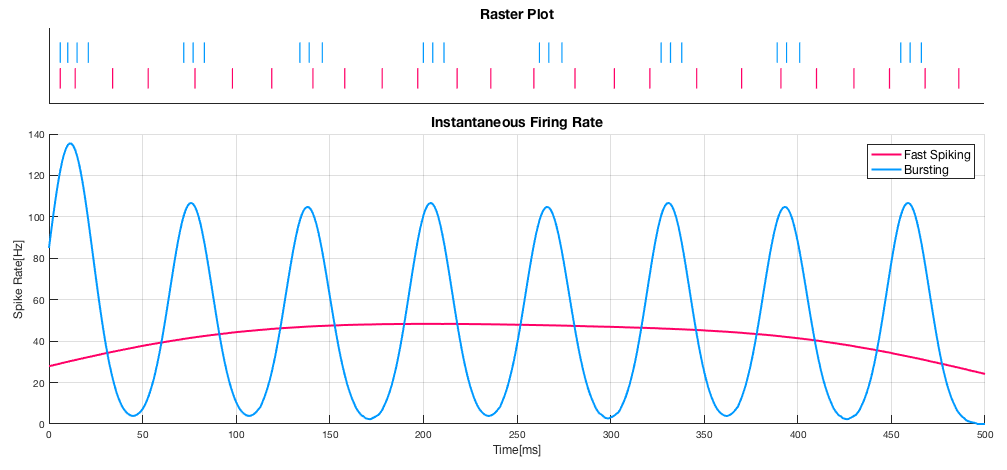


Figure . Instantaneous firing rate estimated using Gaussian kernel with optimal width for fast spiking (left) and bursting (right) model. The firing rate is well estimated in the middle of the fast spiking model (approximately 50Hz). The bursting model also was well estimated.

As shown in Figure 11, for both fast spiking and bursting, the instantaneous firing rate was well estimated. For fast spiking model, both end had some error because we used a Gaussian kernel, but it was well estimated in the middle region as 50Hz. Bursting model, on the other hand, showed an oscillating form of firing rate: The firing rate is approximately 100Hz in the bursting state, whereas it is nearly 0Hz in regions between bursting. Thus, although fast spiking and bursting model has same mean firing rate, using an appropriate kernel, we can obtain the instantaneous firing rate for both models, which are different. (fast spiking model seems to have a more constant firing rate, whereas the bursting model has an oscillating form of firing rate.)

## E) Now suppose this neuron provides presynaptic input to another (target) model neuron. Find a condition that this target neuron fires with the input in b, but not fire with the input in a. Implement your simulation to realize this, and plot the response of the target neuron for the two input conditions.

According to the paper written by Ruben Guerrero-Rivera in 2006[[3]](#footnote-3), the simplest way to model dendritic currents generated by a synapse in response to a spike train would be to use exponential decay synapse model. When we define as the spike occurring time of spike inputs. Then, the dendritic current would be

where is the Heaviside function, is the synaptic efficacy, and is the time constant of the exponential decay. As we confirmed in problem 1a and 1b, fast spiking has a relatively consistent ISI, whereas the bursting had two different type of ISI: ISI between spikes in same bursting cluster, and ISI between spikes in different bursting clusters. Let’s say the average ISI of fast spiking neuron is , whereas the ISI of bursting neuron is and . Note that

Thus, the effect of the current in voltage change for would be

, although this is not the only effect that the current causes. (The current effects the voltage value every moment, and thus also changes depending on the current. However, let’s just make the problem simple only by considering the direct effect of the current.)

We know that constant current with 10mA was able to generate a spike in the previous case. To estimate the condition where the somatic current can generate a spike, maybe

should hold. (Thus the effect of the current for a certain time should have the same amount of effect as with the constant current.) The ISI of the fast spiking was approximately 20ms, whereas the short ISI of the bursting neuron was approximately 8ms. Thus, we can expect to have two bursting neuron spikes between the fast spike interval. Thus, for a neuron to have different response to a presynaptic input from bursting neuron and fast spiking neuron, the below condition might have to hold:

and

I want the current to decay near to 0 before another spike comes in, for the case of fast spiking thus, the time constant should be a value where

I thought would be enough for that approximation, so I decided to select as .Thus, the first condition about would be

which can be re-written as

Like-wise, the second condition would be

which can be re-written as

Thus, any holding

would be an appropriate value, so I chose as my coefficient. Current generated using this rule were as below:

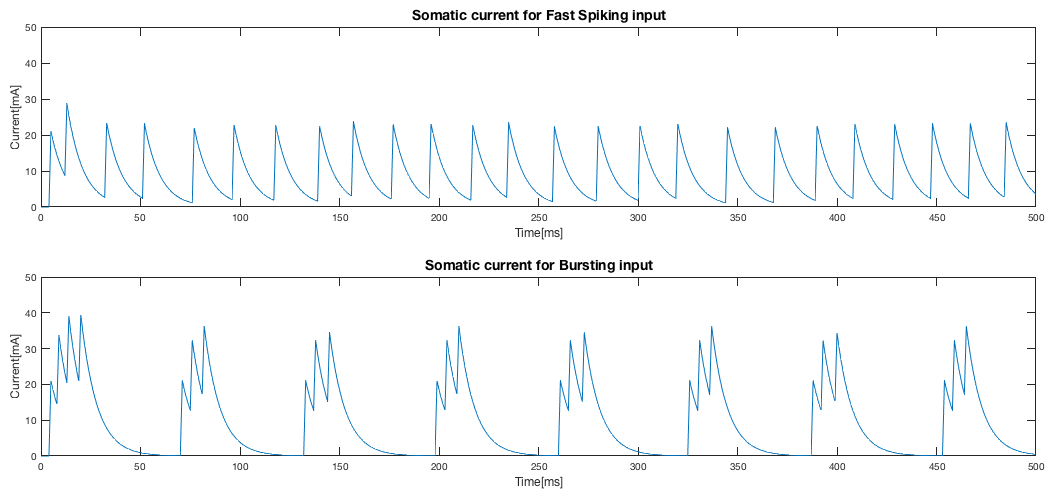


Figure . Somatic current for both fast spiking input (top) and bursting input (bottom). The current value does not exceed 30mA in the case of fast spiking input, whereas it exceeded 30mV in the bursting input.

Then, this current was simply added to the Izhikevich fast spiking model, constructed in problem 1a. The result was as below:

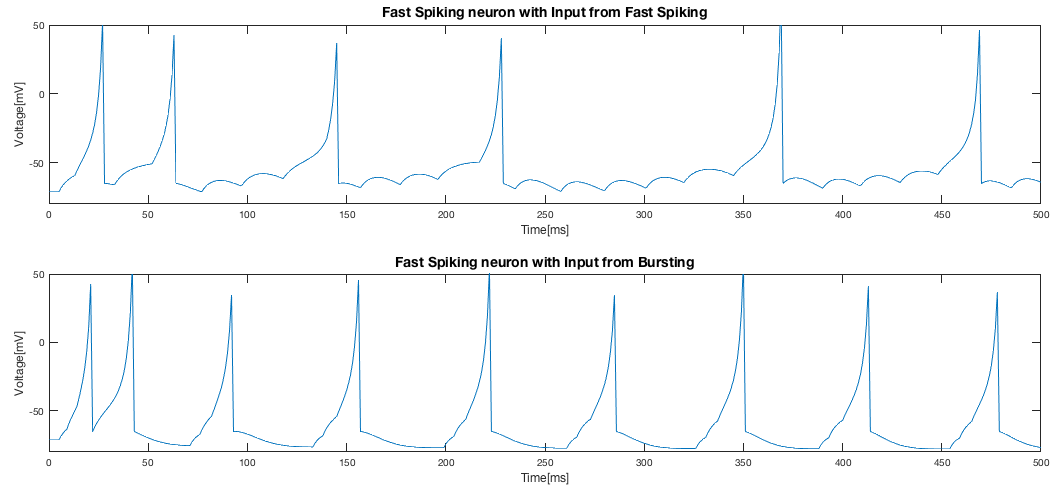


Figure . The voltage of the fast spiking neuron with input from fast spiking (top) and bursting (top). Spikes were generated properly for bursting inputs, but failed to have no response to fast spiking: train of inputs succeeded to generate a spike even for the fast spiking input.

Unfortunately, spikes were also generated in case of fast spiking, unlike what we expected. Why would this happen? Now, let’s think carefully about what each parameter meant. Since and are parameters which are only meaningful when a spike is generated, what we should rather focus on is parameter and . Imagine the case where is smaller (i.e. 0.08). Smaller means the threshold value where becomes increase comes earlier. In other words, smaller means the membrane voltage where stops to help the membrane voltage to rise comes earlier: Thus, to increase the membrane voltage to the threshold value, more current would be needed. The case where we used the fast spiking parameter showed to be too generous about spike trains. I expected that reducing the parameter would make the neuron to sufficiently generate a spike for bursting inputs (which has a higher current) whereas it cannot generate a spike for fast spiking inputs. However, I’ve notice that changing the parameter to 0.08 led to a result like this:

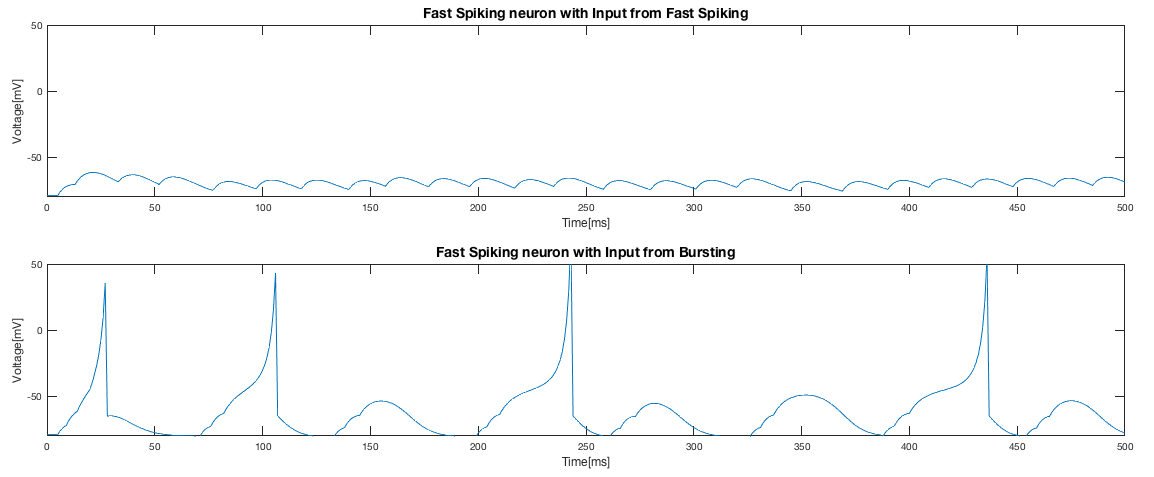


Figure . Response of the neuron (a=0.02, b=0.08, c=-65, d=2) to input from fast spiking (top) and bursting (bottom) neuron. Although the output response has no spike for fast spiking input, the problem is that no spike was generated for some of the bursting signals.

Now the problem was that there was no response for some of the bursting inputs. Now, I moved on my sight to the parameter . Parameter represents the decaying rate of .[[4]](#footnote-4)1 Too small might have caused the to decay too slowly, reducing its force to generate a spike even when it had to. Thus, I changed from 0.02 to 1, and obtained the result as below:

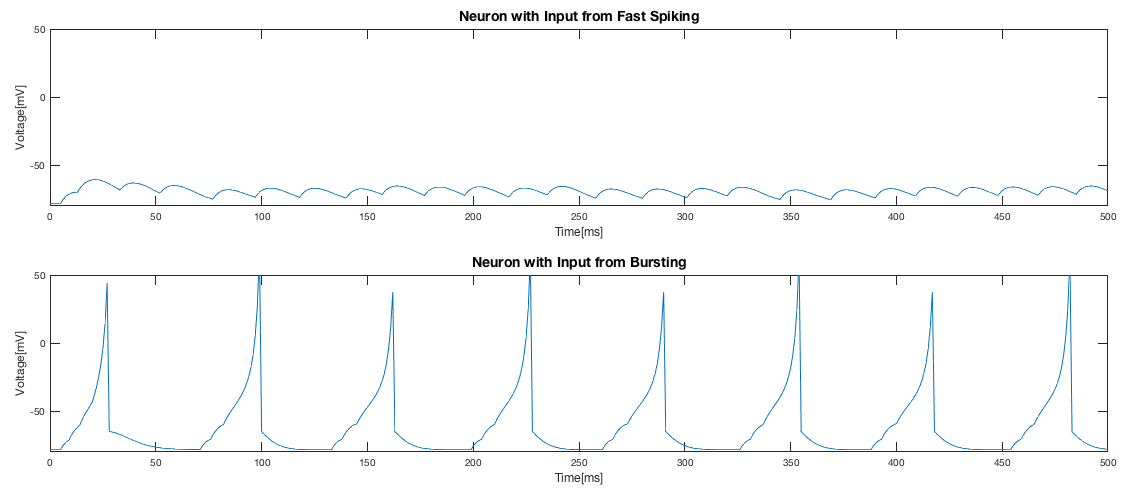


Figure . Response of the neuron (a=1, b=0.08, c=-65, d=2) to input from fast spiking (top) and bursting (bottom) neuron. Surprisingly, it showed response to bursting input, but not to fast spiking input.

Yes! That’s exactly what we wanted! As a conclusion, by simply changing the parameter and so that it requires a strong current (which is only observed in the case of bursting input) for generating a spike, and has a faster decay rate in , we obtained a neuron that generates spike only to a bursting input, and not to a fast spiking input.

# Exercise 2) Principal component analysis

## A set of data () (N=1000) was sampled for 2D Gaussian as shown, which can be described with three parameters () = (SD of long axis, SD of short axis, Orientation angle of long axis) Use the file “DATA.mat” posted to read this data and find the three parameters above.

Using the Oja’s rule, I simply obtained the main axis of the data set:

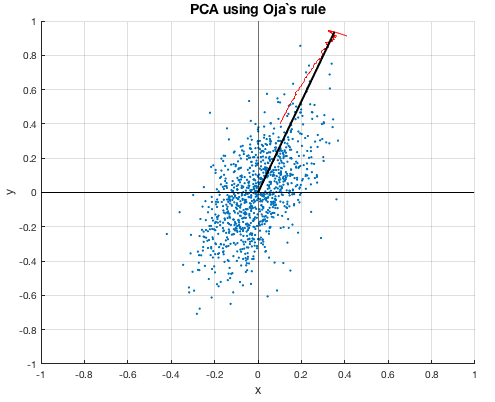


Figure . PCA analysis based on Oja’s rule. Blue dots are the data set, and the black line is the main axis obtained through the code. Red line is the trajectory of the weight.

Using the knowledge about the main axis vector, , the rotated angle of the 2D Gaussian distribution can be calculated as

Using this fact, I rotated all the data points with rotation angle in clock-wise. The below is raw data plotted with the rotated data:

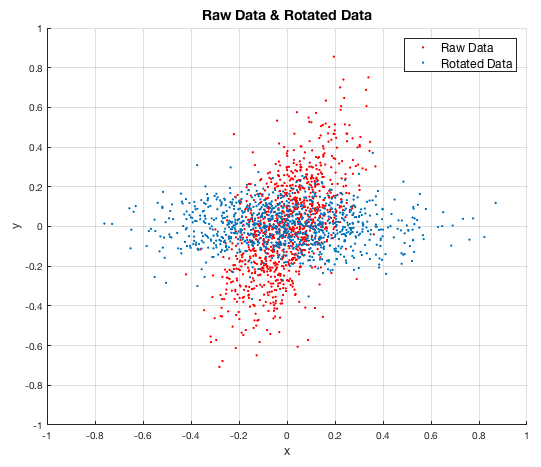


Figure . The raw data (red dot) plotted with the rotated data. (blue dot)

Now, by simply calculating the standard deviation of the rotated data in axis and axis direction, we can obtain and .

The obtained , , values were as below:

Figure/%5BProb2%5D%20Result.png

A 2-D Gaussian distribution with center in (0, 0), rotated in angle counter-clockwise can be expressed as

where

and is a constant.

I wanted to confirm whether the parameter , , that I obtained was correct, so I plotted the raw data with the 2-D Gaussian distribution contour as below:

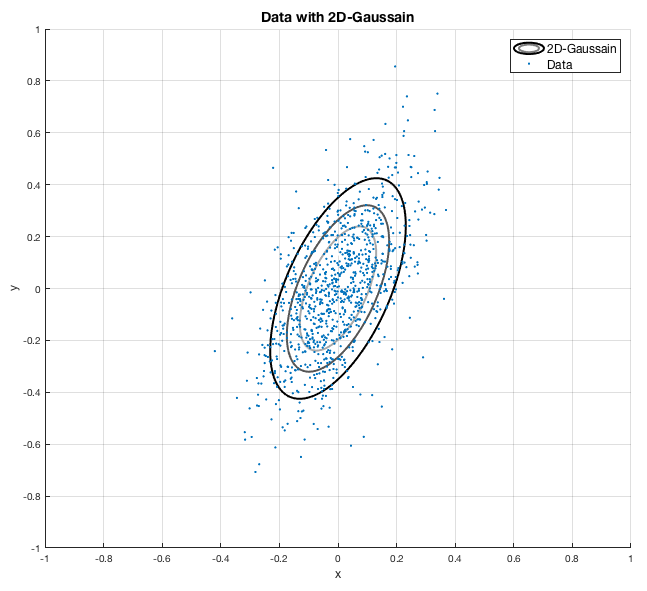


Figure . The data points plotted (blue dot) with the 2-D Gaussian distribution that is predicted to be used in generating data points. The rotation angle, standard deviation (long axis and short axis) of obtained using Oja’s rule seems to be quite accurate.

As confirmed in Figure 18, the 2-D Gaussian distribution well explained the distribution of the data points, so I concluded that the 2-D Gaussian distribution the data points were generated is quite well predicted using Oja’s rule.

# Exercise 3) Information Process in a Simple Associative Network

## A) Assume the value of each input activity is given randomly. Generate sample input patterns (e.g.) for many trials and estimate the entropy of input. Compare this with the analytically calculated value of the maximum entropy of input.

According to Shannon’s entropy theory, the entropy of the input can be written as

have 32 different states, (i.e. [1 0 0 1 0], [0 1 0 1 1], …) and each state has same probability, which would be 1/32. Thus, the total amount of entropy of the input would be

I wanted to confirm whether the numerical entropy is also similar with the analytically calculated value, so I first needed to find the probability of each input patterns. First, I generated 10000 input values. (since each input has five neurons, 50000 in total) I assumed that the probability for the input value to be 0 or 1 is same for all neurons, being 1/2. Thus, 5\*10000 random number between 0 and 1 were generated using ‘rand’ function, and values over 0.5 were transformed to 1, and those with lower value were transformed to 0.

As a result, I generated 10000 random input trains, each input having input values of five neurons. Now, the problem was to find unique set of input data, and this required to distinguish patterns such as [0 0 0 0 1] and [1 0 0 0 0]. (The previous one is when the fifth neuron fires, whereas the latter one is when the first neuron fires) My strategy was to multiply [1 2 4 8 16] matrix with the input train, so that I can obtain a decimal number distinguishing all 32 cases. (0~31) Then, I simply used the ‘histogram’ function to obtain the probability of each input patterns. The below is the result of that histogram:

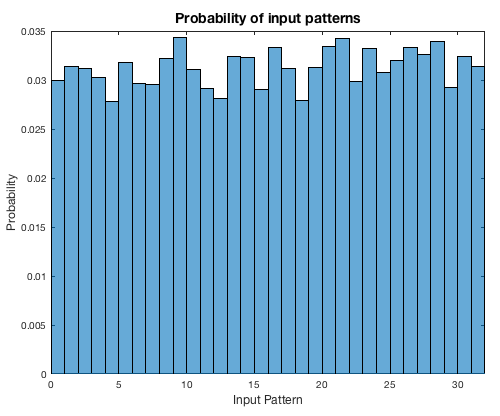


Figure . The probability of each input pattern. All patterns have similar probability, as expected.

Using the same entropy equation, I obtained the numerically calculated entropy of the input, and the value obtained was as below:

Figure/%5BProb3a%5D%20Entropy.png

So yes, the numerically calculated entropy of the input was very similar with the analytically calculated maximum entropy. The difference between the two of them are due to the bias in estimating the probability of each input, which is because the number of input samples was not big enough. (Maybe 100000 samples would have a better estimation.)

## B) When all synaptic weights are the same as =0.5, calculate for all input patterns you got in a. Estimate the mutual information of this system numerically.

Now, let’s look up for the mutual information of this system. Mutual information, which is defined as “quantity representing the amount of information obtained about one random variable, through other random variable”, can be calculated by using the below equation, derived from the Dayan’s book[[5]](#footnote-5):

is calculated using

and using the function ‘unique’, I obtained the unique values of , using them to calculate . Then, using the fact that

was calculated. As a result, the mutual information was obtained as below:

Figure/%5BProb3b%5D%20I_m.png

## C & D) Now each synaptic weight are chosen from random numbers uniformly distributed in [0, 1]. Again calculate for all input patterns and estimate the mutual information as in b. Discuss the difference between two results. Discuss the condition of where the mutual information is maximized. Try repeat above process using this condition and estimate the mutual information.

Now using the same methodology, but different weights – thus different , the mutual information was again obtained:

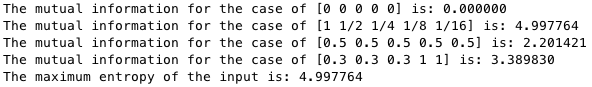
Figure/%5BProb3c%5D%20I_m.png

This time, the mutual information was bigger than the case where all the weights were 0.5. It became so large that it reached the maximum entropy of the input! How should we interpret this? Let’s look at the definition of the mutual information again: mutual information represents the amount of information you can obtain about a random variable (that would be , in this case) when you observe another different random variable. (that would be in this case) Imagine the case when all synaptic weight is same, as in problem 3b. In this case, doesn’t really favor an input from a specific neuron. It treats all neurons equally, and doesn’t really care about which neuron fired. As a result, would have the same response for both [1 0 1 0 0] and [0 1 0 0 1] input: which would be 1. Thus, when you observe that is 1, you know that two out of five neurons fired, but you can’t know which neuron fired. Then what if, the neuron weight is [1 1/2 1/4 1/8 1/16] for each neuron? In this case, each would have their unique value, thus making an one-to-one correspondence. Thus, simply by looking at the ’s value, we can confidently say what the was. Thus, we obtain larger amount of mutual information when the and has an one-to-one correspondence. In problem b, and did not have an one-to-one correspondence, in problem c, we usually have an one-to-one correspondence. This is why we had a larger mutual information in prob3c: if the weight values are selected from an uniform distribution, it hardly meets a condition where and does not have an one-to-one correspondence: in other words, it usually has an one-to-one correspondence.

Then, what would be the exact condition where mutual information becomes maximum? The mutual information becomes maximum when and has on-to-one correspondence relationship, in other words,

holds for all and , when is any input train (i.e. [1 0 0 1 0]) and . One example of such weight would be [1 1/2 1/4 1/8 1/16] as mentioned above. Then what would be the case where we obtain no mutual information? This case would be when the weight is [0 0 0 0 0] since all would be mapped to =0. Thus, we would obtain no information when we observe : we don’t even have a clue about what the would have been!

According to my interpretation, the case where weight is [0.5 0.5 0.5 0.5 0.5] had lower mutual information than the random weight case because neurons were not able to be distinguished. Then, what about the case where weight is [0.3 0.3 0.3 1 1]? Would it have an intermediate mutual information value? (Three sets of neurons (=0.3) and two sets of neurons (=1) are indistinguishable in their own set, but can be distinguished between the set) In ‘Prob2d.m’ code, I confirmed whether 1) case where the weight is [1 1/2 1/4 1/8 1/16] has the maximum mutual information 2) case where the weight is [0 0 0 0 0] has the smallest mutual information 3) case where weight is [0.3 0.3 0.3 1 1] has intermediate mutual information between case of [0.5 0.5 0.5 0.5 0.5] and [1 1/2 1/4 1/8 1/16].



As predicted, [0 0 0 0 0] had the smallest mutual information, whereas case where input and output had one-to-one correspondence (weight is [1 1/2 1/4 1/8 1/16]) had maximum mutual information. (same value as with the maximum entropy of the input) Case of [0.3 0.3 0.3 1 1] had a mutual information between case of problem 3b and maximum value. Thus, I confirmed that all my predictions were correct and thus my interpretation can be considered to be correct.

1. EM Izhikevich, *Transactions on Neural Networks* (2003), 14(6) [↑](#footnote-ref-1)
2. H Shimazaki, *et al*., *J Comput Neurosci* (2010) 29: 171-182., H Shimazaki, *et al*., *Neural Coding* (2007), 120-123. [↑](#footnote-ref-2)
3. R Guerrero-Rivera, *et al*., *Neural Comput.* (2006) 18(11): 2651-2679. [↑](#footnote-ref-3)
4. 1 EM Izhikevich, *Transactions on Neural Networks* (2003), 14(6) [↑](#footnote-ref-4)
5. P Dayan, *et al*., *MIT Press*, Theoretical Neuroscience, 2005 [↑](#footnote-ref-5)