EECE 5136/6036: Intelligent Systems
Homework 1: Given 9/8/20; Due 9/20/20

One of the most widely used current models for neurons is the *Izhikevich model*, proposed by Eugene Izhikevich (Izhikevich, 2003, 2004). It models the membrane potential of a neuron by two coupled differential equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I(t) \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

with the condition that:

$$if \;\; v \geq 30, \quad v = c, \;\; u = u + d \tag{3}$$

Thus, the model is defined completely by four parameters: a, b, c and d. By setting the parameters to different values, neurons with many different natural behaviors can be obtained, as described in Izhikevich (2003, 2004). Whenever v crosses 30, a spike is said to have been generated, but in fact, the model generates the full shape of the action potential. Thus, it is more than an integrate-and-fire model but far less complicated than a full biophysical model.

In this homework, you will implement one of the simplest types of neurons that Izhikevich (2004) calls a *Regular Spiking Neuron*. A MATLAB program used to generate its plot is given in the file *neuronRS.m* in the attached materials (Izhikevich, 2004). You can modify this or write your own code in any language of your choice.

Problem 1. (100 points) Implement the program to simulate a single neuron and drive it with different levels of external input I. In the file *neuronRS.m*, this is implemented in lines 24-28, which produces a step-function I input set by default to $I = 1.0$. The step occurs at time $T1 = 50$. Note that the membrane potential is represented by V in the code, while the array VV is used to accumulate the time-series of V for plotting. The array *spike_ts* is used to mark spikes (0 = no spike, 1 = spike).

You will run several simulations with different I values, starting from $I = 0$ (which should produce no response) and increase in each trial in steps of 0.5 to a level of $I = 20$. Thus, you will run a total of 41 trials for $I = 0,\ 0.5,\ 1.0,\ ...,\ 20.0$. In each case, you should simulate the neuron for $1,000$ steps, discard the output time-series VV for the first 200 steps, and for the last 800 steps, calculate the *mean spike rate*

$$R = \frac{number\ of\ spikes\ in\ the\ last\ 800\ steps}{800} \tag{4}$$

Again, keep in mind that you will actually be counting spikes over 3,200 data=points, but will divide by 800, not 3,200 because there are 4 data-points for each time-step.

Discuss your results briefly. What kind of pattern do you see, and what does it say about the behavior of the neuron? In what sense is the neuron performing any useful function here?

Required Plots:

1. The full 1,000-step time-series of the membrane potential v for the $I = 1$; 5; 10; 15 and 20 cases as five separate graphs in a single figure (i.e., five plots stacked vertically). Note that you will do this by

plotting VV, which is 4,001 data-points with 4 data-points for each time step. Thus, the 4,001 data-points represent time from 0 to 1,000. The MATLAB code plots both VV and *spike_ts* for illustration - you will only plot VV.

2. A graph plotting R vs. I.

Make sure all your graphs are titled properly, and all axes are labeled with appropriate values on all ticks, etc. When you discuss a graph in your text, it should be cited by its title or label (e.g., 'Figure 1.2(a)'), not as "the figure below" or "the figure on the last page". You will need to add this labeling – it is not in the code.

Keep in mind that you are simulating nonlinear differential equations. The given code uses Euler's method to do this, which requires integrating the state variables in small steps. Thus, each actual time step is simulated as 4 micro-steps of update. The step-size $tau = 0.25$ indicates this. Due to this, the time-series generated over 1,000 time-steps has 4,001 data-points – one initial value for $t = 0$ and then 4 micro-steps for each actual step. However, time should be labeled on the graphs in actual units, not the micro-steps used by the simulator. Thus, the point marked t=100 in your graph will actually be the 401st point of the generated data, and your time axis should go from 0 to 1,000.

Also, in the given code, I is set to remain 0 for the first 50 steps. You should change this so I starts at the desired value for that trial from step 1 and then remains fixed.

Problem 2. (100 points) Now you will simulate a very simple network of two neurons, A and B of the same type as in Problem 1, where A gets steady external input I_A (as in Problem 1), and B gets a steady external input I_B and also an input from the output of A:

$$I_B(t) = I_B + w_{BA} y_A(t) \tag{5}$$

$$y_i(t) = 1 \quad if \;\; v_i(t) > 30, \quad else \;\; y_i(t) = 0 \tag{6}$$

where v_A and v_B are the membrane potentials of neurons A and B at time t, $y_A(t)$ and $y_B(t)$ are the (binary) outputs of neurons A and B at time t, and w_{BA} is the synaptic weight from A to B. Note that $y_A(t)$ is just *spike_ts* for neuron A. Thus, neuron B gets an input of $I_B + w_{BA}$ whenever neuron A produces a spike, and I_B at all other times. There is no other input to B. The system is shown in Figure 1.

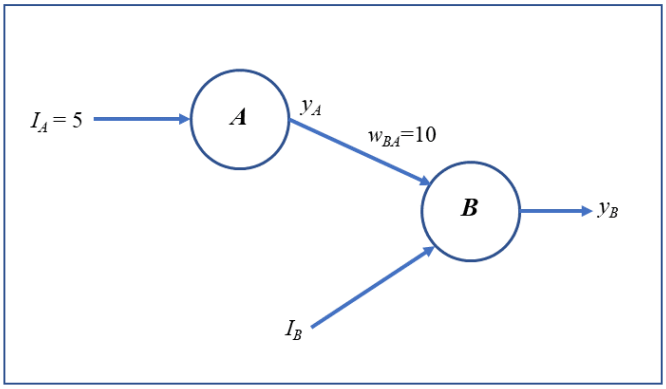

Figure 1: Two-neuron system for Problem 2.

For this problem, assume that $w_{BA} = 10$, and fix $I_A = 5$ for all cases. Run a series of simulations with increasing values of I_B from 0 to 20 in steps of 0.5. Thus, you will run a total of 41 trials for I_B = 0, 0.5, 1.0, ..., 20.0 (just like you did for I in Problem 1). As before, each simulation should be run for 1,000 time-steps.

Required Plots:

1. A graph plotting the mean spike rate, R_B, of neuron B vs. I_B. As in Problem 1, use the *spike_ts* for the last 800 time-steps (3,200 data-points) to calculate the rate. On the same graph, using the x-axis as I, re-plot the mean spike rate, R, of the single neuron from Problem 1. Thus, on this graph, you will have two curves: R_B vs. I_B from Problem 2 and R vs. I from Problem 1.
2. The full 1,000 steps of the time-series of v_B for the I_B = 1; 5; 10; 15 and 20 cases as five separate graphs in a single figure (i.e., five plots stacked vertically). Again, you will do this by plotting the VV variable for your simulation of neuron B.

Discuss your results briefly. In particular, discuss the differences (if any) in the mean spike rate plots of R vs. I from Problem 1 and R_B vs. I_B from Problem 2.

You can implement each run of this simulation either by simultaneously simulating both neurons, or by first simulating neuron A separately to obtain its output time-series and using that as input for a separate simulation of neuron B.

Reference Material:

The accompanying *HW1_Material.zip* file has four items:

1. Izhikevich paper from 2003. This describes the model and you should read it.
2. Izhikevich paper from 2004. This longer paper gets more into the underlying theory. You can read as much of the paper as you want. It calls Regular Spiking Neuron a Tonic Spiking Neuron.
3. MATLAB program to implement a single neuron. You can use this, or re-implement in a language of your choice. It is a very short, simple program.
4. A tutorial/technical report by Yoshua Bengio entitled "Learning Deep Architectures for AI". It is meant as a sample on how to write your reports, but it is also an excellent introduction to deep learning.

If you cannot access the material, please send me mail at. Ali.Minai@uc.edu.

Instructions for the Report:

Submit a report of your work on both problems written like a publication in *a single column format* in PDF. Please look at the technical report by Bengio included with this homework, and use a similar format, though some small variations are all right. The report should meet the following guidelines:

1. **The report file should be named HW1_report.pdf**.
2. Answers to both problems should be in a single report, but begin a new page for each problem. For example, if the answer to Problem 1 takes 1.5 pages, leave the remaining part of the second page blank and begin the answer to Problem 2 on page 3. *Number all pages consecutively.*
3. Use *11 point text, single spaced* for the report. Figure captions can be in a smaller font size. Use Times Roman, Calibri, or some other standard font suitable for a professional document. No weird calligraphic fonts please!

4. The answer to each problem should have a heading with the problem number.

5. There should be four separate sections in the answer to each problem: **Problem Summary**, **Results**, **Discussion**, and **Conclusions**. Each section heading should be in bold letters as in the sample by Bengio. The **Problem Summary** section should not simply copy the problem statement in the homework, but should summarize what you did in about 10 lines of text. The results should refer to the figures by number (see below).

6. *The figures must include all the plots required for the problem.* You may also include additional plots that you think will help explain your comments, but the required plots must be included and identified explicitly as required plots. Make sure the plots are connected properly with the text.

7. All figures should be integrated into the text rather than being put on separate pages (refer to the Bengio document or the way Figure 1 is embedded in this document). Each figure must be given a number, e.g., the first figure in the answer to Problem 1 could be Figure 1.1. Each figure should have a caption underneath the figure, e.g.,

   Figure 1.1: *Plot of mean firing rate as a function of input I. The firing rate is calculated by averaging over the last 800 time steps of a 1,000 time step simulation.*

   All axes in each graph must be labeled. Font sizes for any symbols or tick labels must be large enough to be legible. If plotting multiple plots on the same graph, each plot must be clearly distinguished by different colors, line types or symbols. Lines should be thick enough to be clearly visible.

8. . The answers to the problems should be no more that 2 pages each *including all figures.* You should figure out how best to fit the figures on the page so they are legible but fit within the limits.

9. *All of the above should be in the report document, and NOT in the code. You should not expect anyone to read your code to see your results, comments, discussion, etc.*

10. **Code:** *The entire code used to generate your results must be submitted with your homework.* It should be commented to identify every significant part of the algorithm by which you generated your results. You can use parts of the Izhikevich code, but your program must include a statement at the top acknowledging that part of the code came from this source.

Submission Instructions:

The homework is due at 11:59 PM US Eastern Time on the due date.

You should submit your report on-line through Canvas. Your submission will have three documents: 1) A report **HW1_report.pdf** as described above; 2) A file (or .zip or .rar file) with all your source code; and 3) A brief README file with instructions on how to compile (if needed) and run the code. If Canvas does not let you submit a file with the type postfix (.py, .m, etc.) of a source code file, try changing the postfix manually to .docx or .txt and submit it. State in the README file what the postfix should be changed to in order to run the code.

Grading:

Points will be awarded for correctness of the results, proper plots, and the clarity of description and conclusions.

You may consult your colleagues for ideas, but *please write your own programs and your own text.* Both the text of the reports and the code will be checked randomly for similarity with that of other students. *Any text or code found to be copied will lead to an automatic zero on the entire homework for all students involved.* Repeat offenses in the future will incur more severe consequences.