Simulating the Izhikevich spiking neuron model using the Brian2 software

Alex Beltran and Jorge Orbegozo

Abstract

Spiking Neural Networks have been an interesting subject and open topic on Neural Network research since 1952, when the Hodgkin–Huxley neuron model was first presented, a biologically inspired simple neuron. From then on (and more when the Neural Network boom happened thanks to a more advanced and powerful computing units that allowed to bypass the high computing costs of Neural Nets) researchers have been active trying to find good uses for these biologically - accurate models.

But these models that try to, to some extent, simulate the brain, have run into hard problems, due to the more ”exotic nature” (the use of actual electrical spikes instead of, for example, encoded features), where researchers have not been able to find good uses of spiking neural networks for machine learning tasks (also due to how hard they are to train), even if they have been able to prove the fact that they are really computationally powerful.

But one good use that has been found is to replicate and test different stimuli on biologically fair neuron models that work as extra tools on diverse neuroscience problems. One of the models usually used for that is the Izhikevich spiking neuron model, the one we are going to discuss in this paper.

Our particular intent with this project is to try and create an understandable model in python 3 using Brian 2 software that can replicate the mayor features of the model.

Contents

1 Description of the problem

in this project, we have been given the task of modeling the Izhikevich spiking neuron model [1] [2] using python 3 software, specifically, the Brian 2 python 3 library (See: `https://brian2.readthedocs.io/en/stable/`, that tries to provide a work-space for creating-modelling fast and extremely compact Spiking Networks and Spiking Neurons. Using this new model, we are asked to try and reproduce the several spiking neuron patterns mentioned in the Izhikevich papers [1] [2].

In order make things more clear, the spiking neuron patterns are comprehended on these few sub-classes, according to what we are trying to simulate:

- 1 Type.
  - 1.1 Excitatory neurons.
  - 1.2 Inhibitory neurons.
  - 1.3 Thalamo-Cortical neurons.
- 2. Dynamics.
- 3. Neuro-Computational Features.

2 Description of our approach

2.1 Research

Before getting our hands on the model, we decided it would be a better idea to learn to use the Brian 2 software prior to investigate the model. Brian 2 has extensive documentation that can be easily accessed in `https://brian2.readthedocs.io/en/stable/`, but luckily it also incorporates several well constructed and easy to follow tutorials in here created in jupyter notebooks, so they were perfect to familiarize quicker with the library, that can bee quite complex otherwise.

Next step was to get more familiar with spiking neural networks and the Izhikevich model in general. The paper [3], thought it was harder to read due to the good amount of technical details gave us a good idea of general intuition behind the model, trying to reproduce and model to a high scale the mammalian brain systems.

The most important papers overall were [1] [2], that mathematically described the model and showed several graphics / figures with the patterns to obtain. But more important than that was that, for each neuron type/feature they provided an intuition behind the parameter choice for that specific neuron type/feature, like how fast spiking inhibitory neurons fire periodic hits of action-potentials with extremely high frequency practically without any adaptation (they do not slow down to recover), and that's why the a parameter that models the recovery is really low, to model a fast recovery value. Also, for showing the parameters of the several neuron feature figures, and also as help, the Izhikevich page provides a matlab (.m) implementation of the simple neuron model, that was great help for this project.

Yet with all of this, implementing the model ended up being a quite hard task, since python3, using Brian2, ended up being much different than simply using matlab. Thankfully we found some help on an old neural modeling tutorial [4] that showed us things that weren't really clear on the more theoretical papers, like the changes Izhikevich makes on the function constants and many more. It also gave us the idea of making the parameter I able to be introduced as a constant or as an expression. The page was really useful with lost of good explanation. Sadly, the source code the page offers is long lost, but we managed to find it on The Way back Machine, and that was really useful overall for the parameters and for general python tricks we used on modeling, even if we used the Brian2 framework. The link for that original source in the way-back machine can be found here.

2.2 The Izhikevich's model.

The Izhikevich's model is an specific model for spiking neurons, designed by Eugene M. Izhikevich. The goal of the model is to be able to represent several neuron specific patters without incurring into high costs. The first instance of this model was presented in 2003.

The original model uses two-dimensional system of ordinary differential equations that go as follows:

$$v' = 0.04v^2 + 5v + 140 - u + I$$
$$u' = a(bv - u)$$
$$v' = \frac{dv}{dt}\ , u' = \frac{du}{dt}$$

A second system defines what happens when v reaches its threshold value (30). When that happens, the following equation system is used as a reset system:

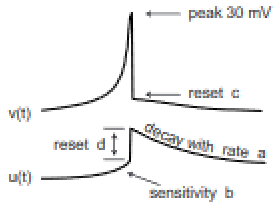
$$v = c$$
$$u = u + d$$

Where the parameters and variables are the following:

- Variables:
  - v: variable that represents the membrane potential of the neuron.
  - u: variable that represents a membrane recovery variable which provides negative feedback to v.
  - I: variable that represents the input current.
- Parameters:
  - a: This parameter defines the time scale of the recovery variable u.
  - b: This parameter defines the sensitivity of the recovery variable u to the sub-threshold fluctuations of the membrane potential v.
  - c: This parameter defines the after-spike reset value of the membrane potential v.
  - d: This parameter defines the after-spike reset of the recovery variable u.

In Figure 1 we can see a simple recreation of Izhikevich's spiking neuron model.

Figure 1: Izhikevich's spiking neuron model.



2.3 Our python3 Brian2 implementation.

Our code consists of 3 functions used to model the original Izhikevich's model in python3 using the Brian2 framework **. The functions are the following:

- *printParameters(pParameters*)*
- *printOutcome(pParameters*)*
- *Izh_spiking_model(pParameters*)*

** pParameters refers to the original parameters of the function. This will be used as a placeholder to make the paper look clearer, but the parameters will be explained on each function.*

*** Its heavily recommended to understand the following sections to have a look at the jupyter notebook and the explanations on it, as well to the code level comments. All needed to know to understand the code is here, but to keep the paper clear we decided not to add code snippets here.*

2.3.1 printOutcome(pParameters*)

2.3.1.1 ***Use:*** This function is an auxiliary function to $Izh\_spiking\_model(pParameters*)$ that uses 2 Brian 2 monitors. A monitor is a "recorder" of what is happening to the neuron (or net, if there is more than one). One of the monitors just records the spikes (the moment when the spike happens) and the other one records the values of I and V.

2.3.1.2 ***Parameters:*** 2 Brian2 monitors.

2.3.1.3 ***Pre-condition:*** A Brian2 spike monitor and a Brian2 normal Monitor.

2.3.1.4 ***Post-condition / Outcome:*** Prints / Shows the graph showing the results of the model.

2.3.2 printParameters(pParameters)

2.3.2.1 ***Use:*** This function is an auxiliary function to $Izh\_spiking\_model(pParameters*)$, that prints the parameters that are feed into the model. We decided to make it to make the function of the model cleaner and smaller.

2.3.2.2 ***Parameters:*** This function uses the same parameters as *Izh_spiking_model(pParameters*).*

2.3.2.3 ***Pre-condition:*** The parameters of the model.

2.3.2.4 ***Post-condition / Outcome:*** Prints the parameters of the model.

2.3.3 Izh_spiking_model(pParameters*)

2.3.3.1 *Use:* This is the main function we use to model the spiking neuron model.

2.3.3.2 *Parameters:*

- n_neurons: Default value = 1. This parameter defines the number of neurons we are gonna use. For every pattern on this paper, this number should be 1.
- the a,b,c,d variables from Izhikevich's model's equations.
- v0: Default value = -65. This parameter defines the initial membrane potential.
- tau: Default value = 10. This parameter defines the time step.
- time: Default value = 2500. This parameter defines the time the simulation will run.
- cInjection: Default value = "10". It represents the input current. This value can be a normal number (a constant) or it can be a string expression. This is useful to gain precision with controlled input currents. An example of an expression that gives the parameter the value 20 when t (time unit) is above 10ms could be: $"20 * (t >= 10 * ms)"$.
- u0: Default value = "empty". This parameter defines the initial value of u in the function. If left "empty", it will automatically assign the value $b * v0$, if introduced a constant number, it will assign that number. This is used since on a particular example, Izhikevich uses a different starting value and a different definition of $\frac{du}{dt}$.
- differential_constants: Default value = "empty". This parameter defines the constants of the $\frac{dv}{dt}$ function. If left "empty", it will introduce the default values of the constants (0.04, 5, 140). If introduced a numpy array of size 3, it will introduce the values of the array as constants. Example: [p1,p2,p3] will assign constant 1 p1, constant 2 p2 and constant 3 p3. This is useful since it turns out that in a few of Izhikevich's examples he changes up the values of some of constants in the $\frac{dv}{dt}$ equation.
- injectionMode: Default value = "once". This parameter defines if the injection is done once of it it repeats every millisecond. This is useful and mostly used when an expression is introduced as a parameter in cInjection, since we need to check if the expression is true or false on every step of the run. To change to this mode, the parameter should be introduces as: $injectionMode = "dynamic"$

2.3.3.3 *Pre-condition:* The parameters mentioned (if none, it results to the default values).

2.3.3.4 *Post-condition / Outcome:* The outcome of the $printOutcome(pParameters)$ function.

2.3.4 A more in depth view of our function.

A quick eye looking at our model will have noticed that in our model the equations are a bit different. The general equations don't take into account the time step, we needed to add it.

We also define the main equation $\frac{dv}{dt}$ differently, with the constants as variables. As mentioned in the parameter explanation, this is because it turns out that in a few of Izhikevich's examples he changes up the values of some of constants in it. So we just declared the constants as 3 variables that can be changed thorough the parameters of the function. Our functions end up being:

$$v' = (k1v^2 + k2v + k3 - u + I)/tau$$
$$u' = a((bv) - u)/tau$$
$$v' = \frac{dv}{vt} \text{ , } u' = \frac{du}{dt}$$

2.3.5 A strange case: The (R) Accommodation pattern.

The (R) accommodation pattern supposes a problem to our model, since to generate that exact pattern, Izhikevich overall changes the entire $\frac{du}{dt}$ equation. Since there is no easy way to get that equation also a parameter, and making another model just for a small change seems excessive, we decided that when that pattern wants to be created, the $\frac{du}{dt}$ function must be changed to $\frac{du}{dt} = a(b(v + 65))/tau$ manually. For this strange case there is also a parameter that must be used, and that's giving the u0 the following value: $u0 = -16$

3 Results

3.1 By type

3.1.1 Excitatory cortical cell classes:

3.1.1.1 *(RS) Regular Spiking*

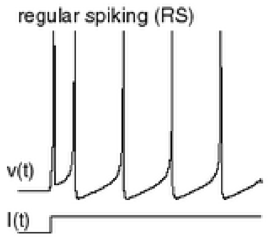


Figure 2: Izhikevich original model result.

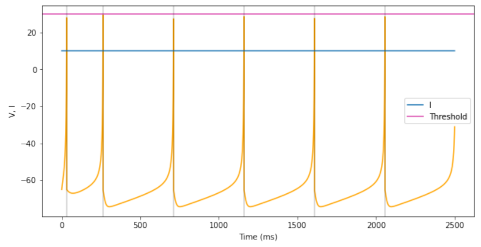


Figure 3: Our model's result.

Parameters used: Default configuration and $d = 8$.

3.1.1.2 *(IB) Intrinsically bursting*

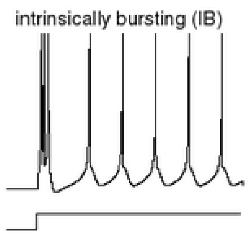


Figure 4: Izhikevich original model result.

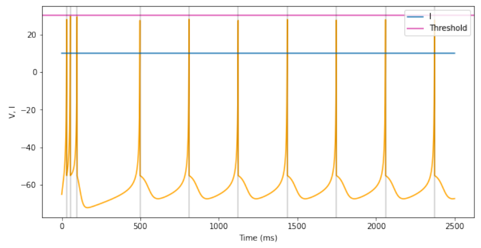


Figure 5: Our model's result.

Parameters used: Default configuration and $c = -55 + d = 4$.

3.1.1.3 *(CH) Chattering*

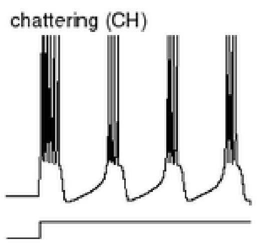


Figure 6: Izhikevich original model result.

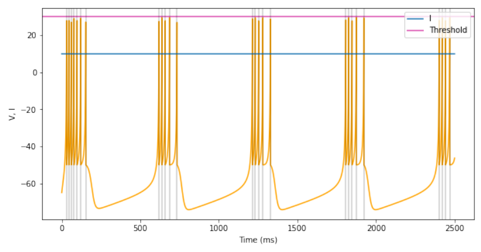


Figure 7: Our model's result.

Parameters used: Default configuration and $c = -50$.

3.1.2 Inhibitory cortical cell classes:

3.1.2.1 *(FS) Fast spiking*

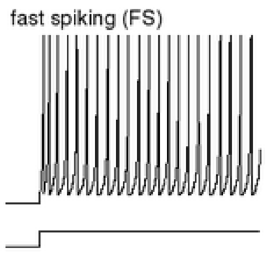


Figure 8: Izhikevich original model result.

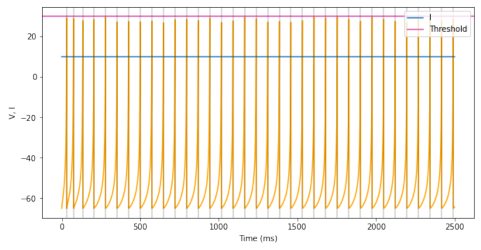


Figure 9: Our model's result.

Parameters used: Default configuration and $a = 0.1$.

3.1.2.2 *(LTS) Low-threshold spiking*

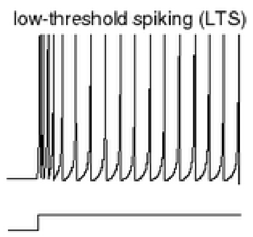


Figure 10: Izhikevich original model result.

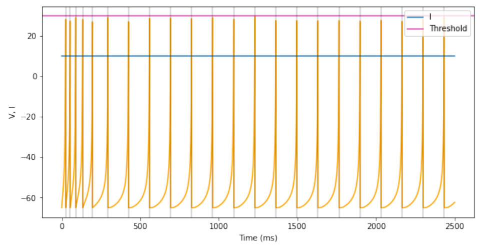


Figure 11: Our model's result.

Parameters used: Default configuration and $b = 0.25$.

3.1.3 Thalamo-cortical neurons:

3.1.3.1 *(TC) Thalamo-cortical 1*

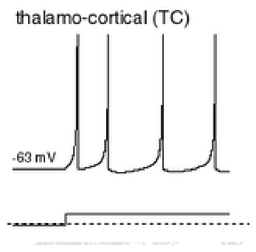


Figure 12: Izhikevich original model result.

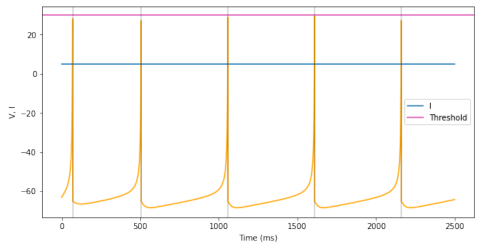


Figure 13: Our model's result.

Parameters used: Default configuration and $cInjection = 5, v0 = -63$.

3.1.3.2 *(TC) Thalamo-cortical 2*

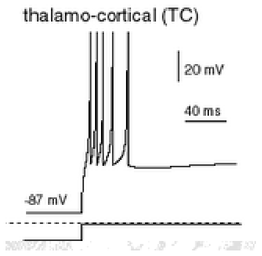


Figure 14: Izhikevich original model result.

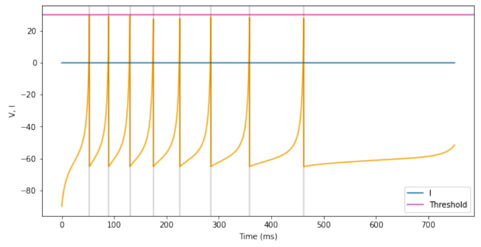


Figure 15: Our model's result.

Parameters used: Default configuration and
$b = 0.25, d = 0.05, cInjection = "-10 * int(t <= 300 * ms)", v0 = -90, time = 750.$

3.2 Neuro-Computational Features.

In order to keep the report from breaking the maximum report length, we will show our results without comparing them to the original Izhikevich model. We would like to remind the reader that the originals can be found here.

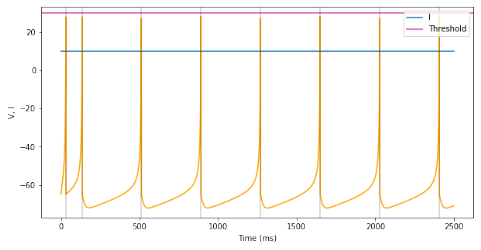


Figure 16: (A) Tonic Spiking.

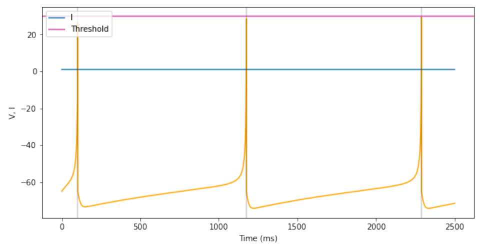


Figure 17: (B) Phasic Spiking.

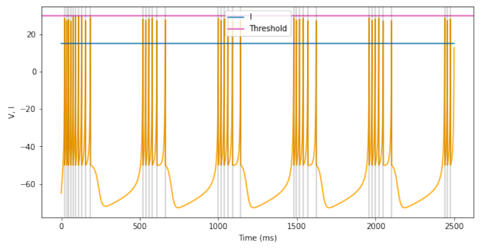


Figure 18: (C) Tonic Bursting.

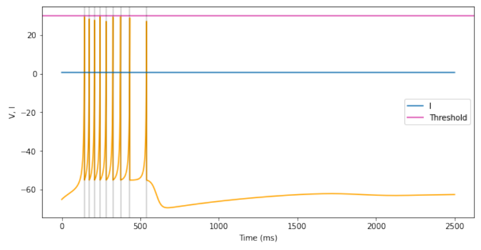


Figure 19: (D) Phasic Bursting.

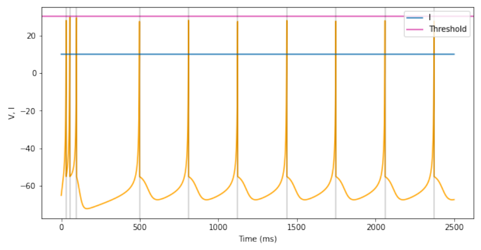


Figure 20: (E) Mixed Mode.

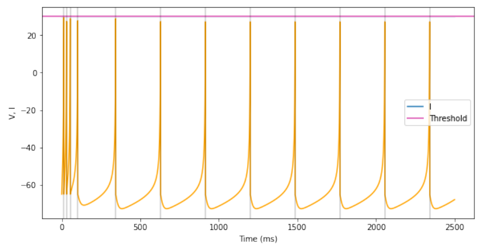


Figure 21: (F) Spike Frequency Adaptation.

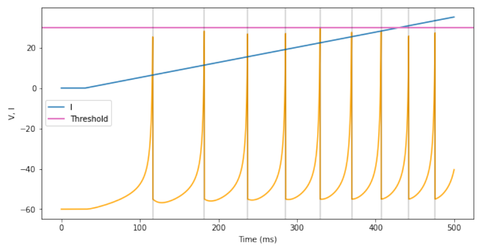


Figure 22: (G) Class 1 Excitable.

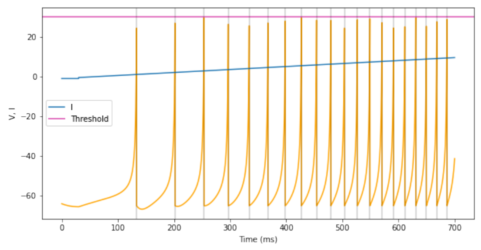


Figure 23: (H) Class 2 Excitable.

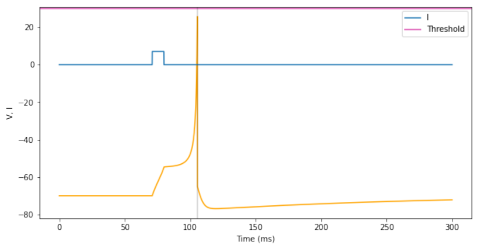


Figure 24: (I) Spike Latency.

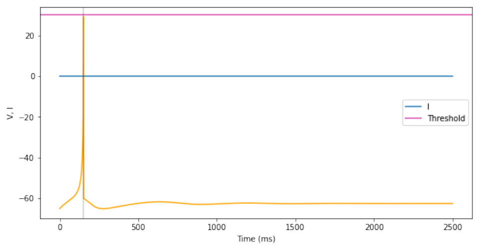


Figure 25: (J) Subthreshold Oscillations.

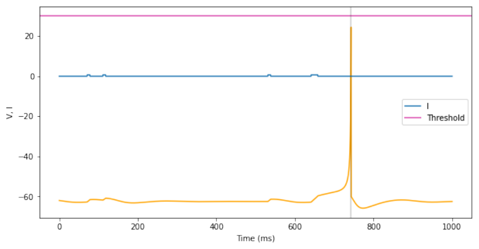


Figure 26: (K) Resonator.

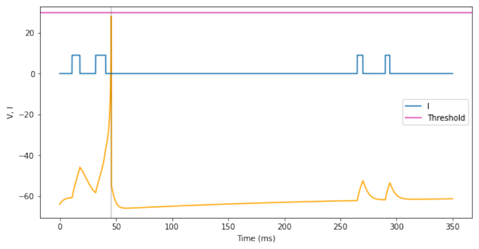


Figure 27: (L) Integrator.

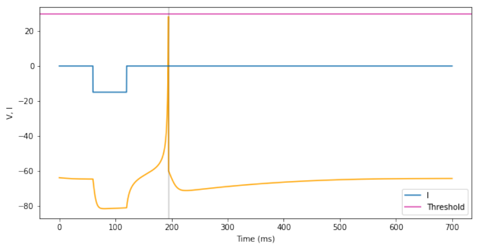


Figure 28: (M) Rebound Spike.

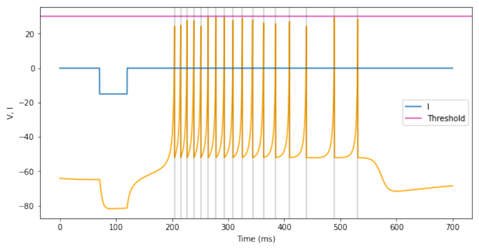


Figure 29: (N) Rebound Burst.

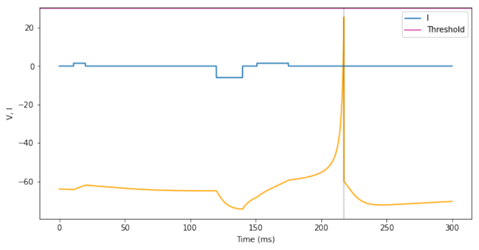


Figure 30: (O) Threshold variability.

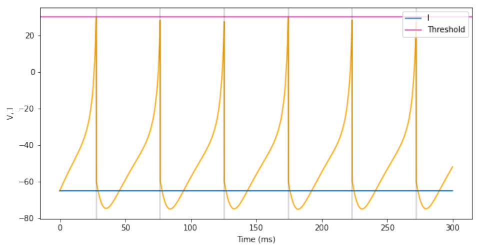


Figure 31: (P) PBistability.

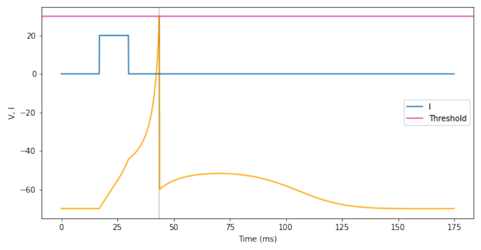


Figure 32: (Q) Depolarizing After-Potential.

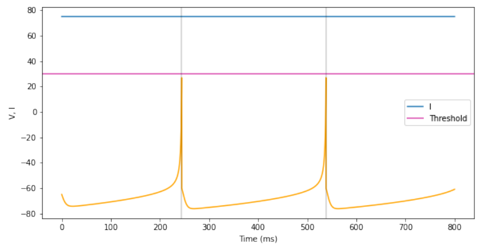


Figure 33: (S) Inhibition Induced Spiking.

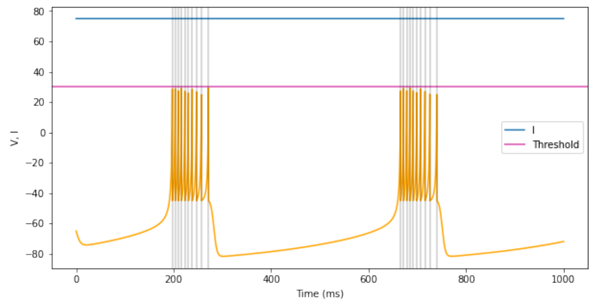


Figure 34: (T) Inhibition Induced Bursting.

3.3 Some bad results.

From the 26 patterns we could correctly replicate every single one but 2 of them. These two patterns are the (R) Accommodation pattern and the only Dynamic pattern, the (RZ) Resonator.

For the correctly replicated patterns, some of them use the default parameters entirely while some others required adjustments to the I variable. This is noticeable on most of the patterns that use expressions for the I variable, since we had to time the controlled stimulus differently in order to get the patterns with Brian 2. The rest of the parameters are always the original parameters declared by Izhikevich.

On the other hand, the bad results. Our model found impossible to replicate the (R) and the (RZ) patterns. We have different hypothesis on why this could have happened. For the RZ Resonator pattern, our hypothesis is that we were not able to find the correctly timed input of I with a logical expression. This would explain why we could replicate the other patterns of the paper just fine. Going back to the (R) accommodation we have 2 hypothesis: First, the same one as for the (RZ) pattern, the input of I was not correctly timed. This is the most unlikely scenario taking into account the nature if the pattern. Our second hypothesis is that the specifics of the pattern just break our model. Lets remember what needed to be done to replicate this pattern: Redefine the $\frac{du}{dt}$ entirely to a new equation and also forces us to introduce a strange value as u0 (-16). We think all of these changes makes the correct timing of the I input overall impossible to achieve correctly.

4 Conclusions

From this work we can infer a couple of thoughts / conclusions:

- First of all, mention how powerful and reliable the Brian 2 framework is. It allowed us to work fast and clean on the work at hand, and executions never took longer than expected. Also, debugging and error solving ended out being pretty easy thanks to a really well done error back-tracing system that allowed us to solve code errors extremely fast.
- Focusing on spiking neural networks, it was really interesting to finally be able to work with a more ”biologically” fair type of neural network, and point out how easy to interpret the spike system ended up being.

Now with the Izhikevich model, we ended up having mixed feelings about it.

- On the one hand, the model turned out to be extremely simple to implement (if we ignore the (R) pattern of course) equation-wise. It also turned out to be a really fast model that could replicate lots and lots of neuron patterns (given the correct parameters).
- On the other hand, we found it sometimes to be extremely abstract. Our main complain about the model and the work in general is how it seems to be full of what its known as ”magic numbers” (constants and/or parameter values that appear without justification or intuitive basis that end up being really confusing to the person who didn’t code them). Sometimes is not really obvious what the a,b,c,d values need to be to achieve desired patterns, and Izhikevich doesn’t clearly state when he decides to change the constant on the $\frac{dv}{dt}$ equation. This problems aggravated when we had to change the timings of the input currents for our Brian 2 models, that turned out to be an unnecessary amount of trial and error.

  Nevertheless we still think its a brilliant work outside the few criticism we have since overall the point of the Izhikevich neurons is to be able to to be replicate a wide range of experimentally observed neural spiking behavior cheaply for use in large-scale simulations, not to gain an understanding of how the neuron works.

On new advancements that could be made, it would be interesting to add synapses and try to generate a general mammalian brain activity simulation. The work on SNN is still on a more ”simulation” side, helping neuroscientific research in all kinds of tasks. We cannot wait for actual ML tasks being solved efficiently with SNNs, since they are the most similar types of networks if we look at the human brain. And by now is common acceptance that the human brain is overall the best ML task solver.

References

[1] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[2] E. M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–1070, 2004.

[3] E. M. Izhikevich and G. M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences (PNAS)*, 105(9):35931–3598, 2008.

[4] Byron Galbraith. Neural modeling with python (part 3). *Neurdon*, 2011.