Computational Neuroscience

Project No. 1 Report

Name: Mahroo Hajimehdi Student No: 610399198

Introduction

In this project, our objective is to simulate the functioning of a single neuron using the Python programming language. We have utilized several libraries, including PymoNNtorch for its neural network capabilities, Matplotlib for plotting, and the Math and Random libraries for numerical operations and stochastic processes, respectively. The simulation framework has been structured into four main components:

1- **Time Resolution**: Here, we outline the time resolution to be used within the simulation, establishing the temporal granularity at which neuron behavior is observed.

2- **Current:** We introduce a variety of current input functions to thoroughly analyze the performance and response patterns of neuron analogs under different stimuli in a controlled environment.

3- **Single Neuron Models:**

   a. Three neuron model variants, namely, Leaky Integrate-and-Fire (LIF), Exponential Leaky Integrate-and-Fire (ELIF), and Adaptive Exponential Leaky Integrate-and-Fire (AELIF) are constructed.

   b. The refinement of these prototypes includes incorporating a refractory period parameter to augment their biological plausibility and realism.

4- **Plotting and Parameter Tuning:** We employ a systematic approach to parameter tuning and plotting, ensuring that the resulting visualizations effectively elucidate the models' behaviors and performance metrics.

In the subsequent sections of this report, we examine each component in more detail and present a series of plots and illustrations to aid in the visualization and comprehension of the neuron models' functionalities.

Time Resolution

Time resolution is a critical factor in the fidelity and accuracy of neural network models, particularly for those that simulate the intricate dynamical systems seen in neuroscience. It determines the smallest timestep for which the model can integrate information, directly influencing the model's ability to replicate the precise timing of biological processes.

To study the dynamics of neural circuits and understand their responses to various stimuli, a time resolution that aligns with the real-world temporal dynamics of neural activity is imperative.

This ensures that simulations are not just abstract representations but are biologically informed and can be correlated with empirical data. Neurons can fire spikes on the order of milliseconds, and as such, a model must have a sufficiently fine time resolution to capture these events faithfully.

High time resolution allows for a more accurate representation of the rapid fluctuations in neuronal activity. While finer time resolution can yield more precise results, it also requires more computation, which can become a trade-off between accuracy and computational resources. Finer time resolutions result in more data points being stored during the simulation, leading to increased memory consumption. Also, Decreasing the time resolution typically leads to slower simulation speeds, as each time step requires additional computational effort. An optimal time resolution needs to be selected to balance these trade-offs.

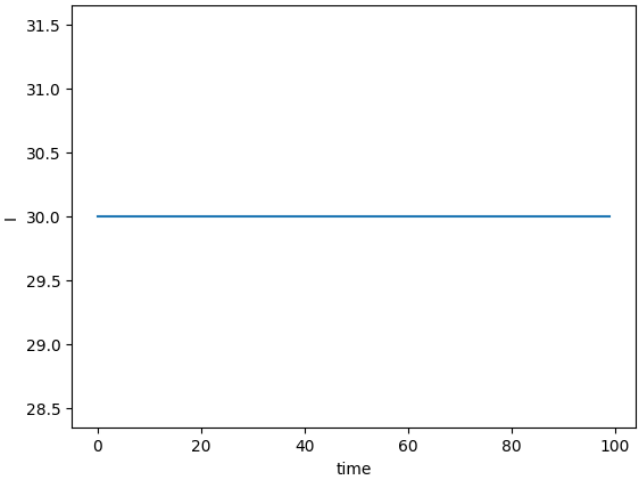
In our implementation, time resolution is encapsulated within a class structure with the same name `TimeResolution` where `dt`, representing the time step, is defined for a specific network instance. This class-based approach allows for the addition of further parameters, attributes, and methods to accommodate the intricacies of more complex models and simulations. This design choice not only ensures modularity and scalability but also provides a clear framework within which time resolution can be tailored and refined to match the detailed requirements of advanced neural network simulations.

Current

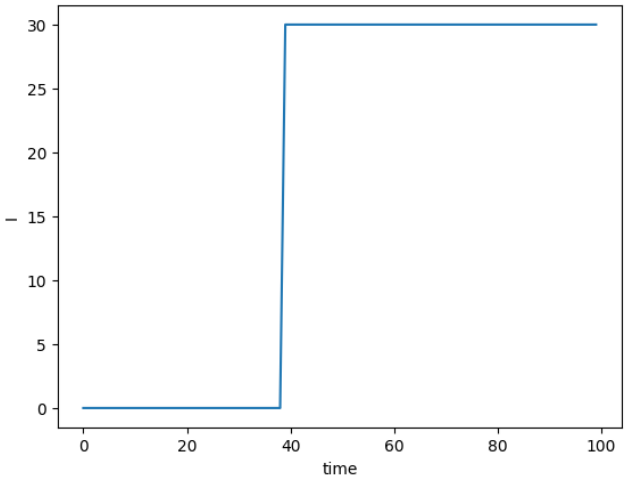
To enhance the assessment of our neural network models, 7 distinct input current patterns have been implemented. Each type is represented by a class with two main methods: `initialize()` and `forward()`. The `initialize` method is responsible for setting up the parameters, which can either be supplied to the class by the user or drawn from predetermined defaults for the specific type of current. Conversely, the `forward` method plays a crucial role in the iterative process—invoked at each step, it dynamically modulates the current input to shape the desired temporal pattern.

1. **Constant Current**: Our first current pattern entails applying a steady, unchanging current to each neuron within a specific neuron group (`ng`). In this configuration, the neurons' current tensor s reflect identical values, as determined by the input, effectively setting the current `I` for each neuron to this constant input value throughout the entire simulation period. This ensures a stable and continuous excitation for the neurons involved.

2. **Unit Step Current**: For the Unit Step Current pattern, the neurons in `ng` begin with their current values at zero. The current is altered by defining a time point (`tp`), which specifies when we want the current to shift from 0 to a predetermined value. Once this point in time is reached during the simulation, all neurons in `ng` receive and maintain this new current value, implemented within the `forward` function of the current object.

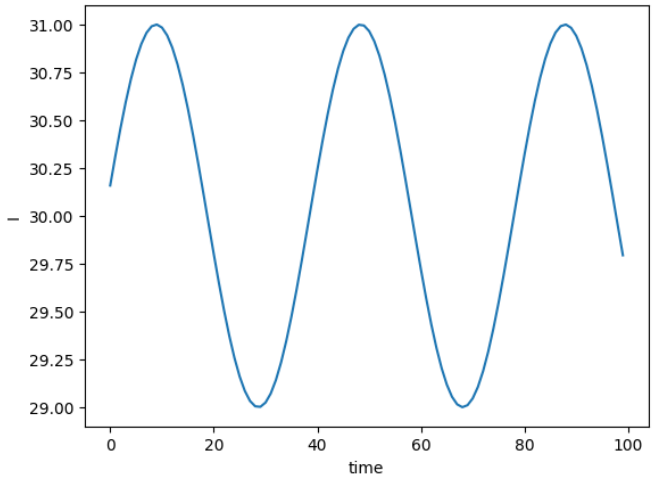


Constant Current

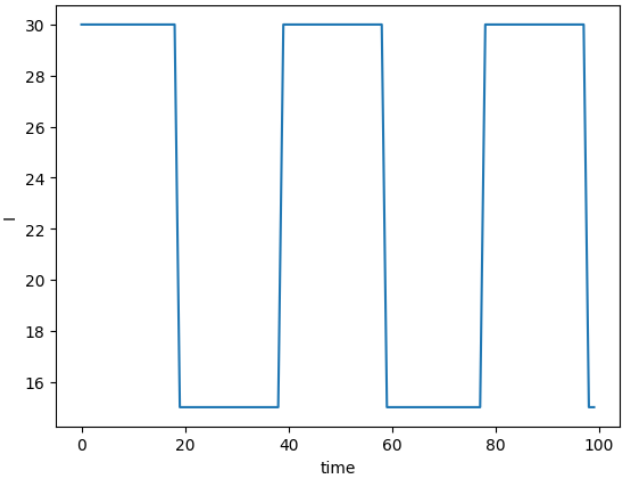


Unit Step Current

3. **Sine Wave Current**: The Sine Wave Current pattern creates a fluctuating current calculated through the sine function, with the amplitude, frequency, and initial value provided as inputs. The `forward` function calculates and sets `ng.I` iteratively, resulting in a sinusoidal pattern of current across the neurons.
4. **Square Wave Current**: For the Square Wave Current, the logic is same as the Sine Wave, but instead aims to generate a square wave pattern within the `forward` function. The decision between the wave peak value and half of that value is determined by comparing the calculation of a sine function with the base value, generating the square wave pattern.

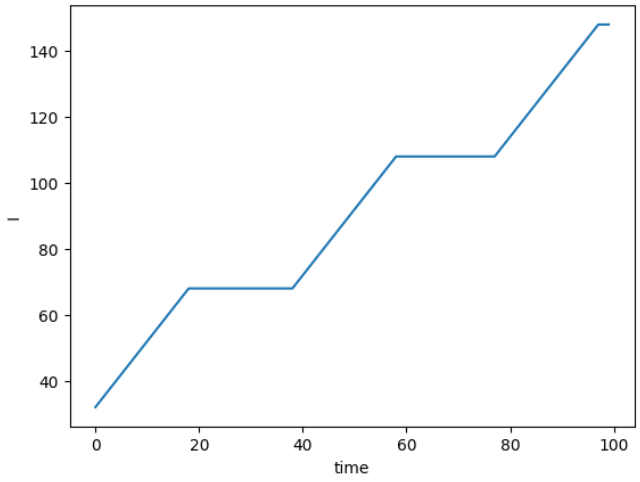


Sine Wave Current

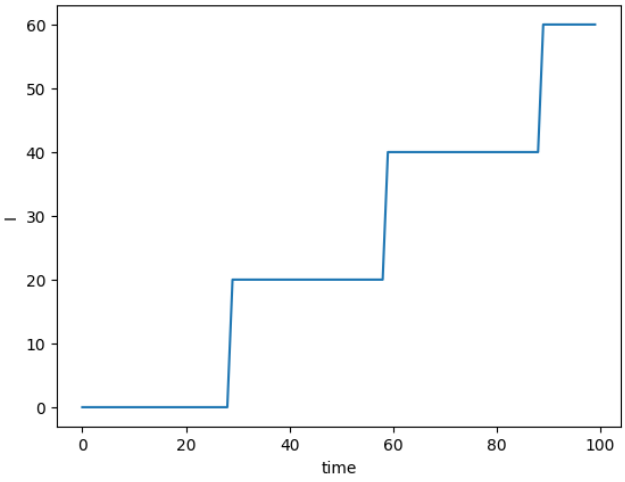


Square Wave Current

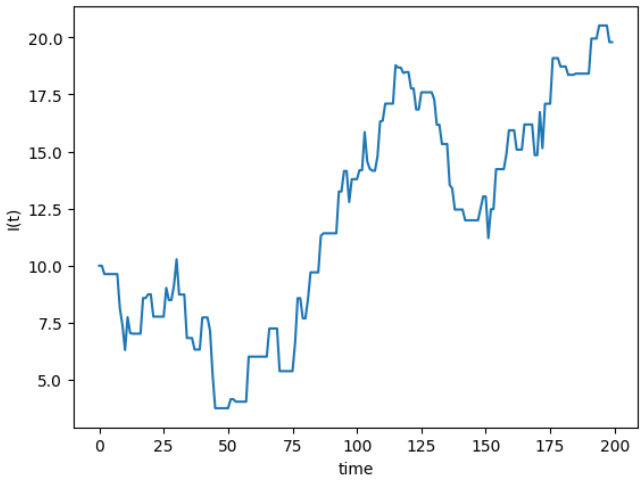
5. **Step Current with Slope**: This pattern is designed to initiate a step current with a progressive slope, avoiding sudden shifts in value. In the `forward` function, the transition between two states is softened, resulting in a stepped yet sloped progression of the current.
6. **Step Current:** Similar to the other step function, this pattern creates steps in current based on a regular interval. Every 30 model time units, if the condition is met, the `forward` function increases the current by a predetermined step value.
7. **Current with Noise**: Finally, the Current with Noise introduces random fluctuations to simulate realistic variations in neuronal currents. The rate and magnitude of noise injections are determined by the `rate` and `limit` inputs, with occurrences of noise dictated by random chance within the `forward` function.



Step current with Slope



Step Current



Current with Noise

Single Neuron Model

1) Leaky Integrate-and-Fire (LIF) Model

The Leaky Integrate-and-Fire (LIF) model is a simplistic representation of a neuron, designed to capture the essence of neuronal spiking behavior. As a reductionist model, the LIF distills a neuron's function to its core components: a membrane potential that integrates incoming signals and an action potential, or spike, that occurs when a certain threshold is reached. The LIF model assumes a linear integration of input regardless of the neuron's state, followed by a reset of the membrane potential once a spike is emitted.

Despite its simplicity and the exclusion of complex biological phenomena, such as adaptation, bursting, and inhibitory rebound, the LIF model excels in certain scenarios. Its primary strength lies in its ability to simulate precisely-timed spikes, which is a critical aspect of understanding neural coding.

1-1) *Implementation of the LIF Model*

In implementing the LIF model, an `initialization` function takes several key parameters from the input, including resistance (`R`), membrane time constant (`tau`), threshold potential (`threshold`), resting potential (`u_rest`), and reset potential (`u_reset`). Neurons in the group are then assigned an initial potential uniformly distributed between the threshold and the reset potential.

The `forward` function of the neuron group is tasked with encapsulating the dynamics of the model, which include membrane potential evolution, spike firing, and the reset process. The membrane potential (`ng.u`) is updated following the Euler method, after which a check is conducted to ascertain if any neuron has surpassed the preset threshold. Neurons reaching this mark have their potentials reset to `u_reset`.

The dynamics of the membrane potential can be updated in each iteration as follows:

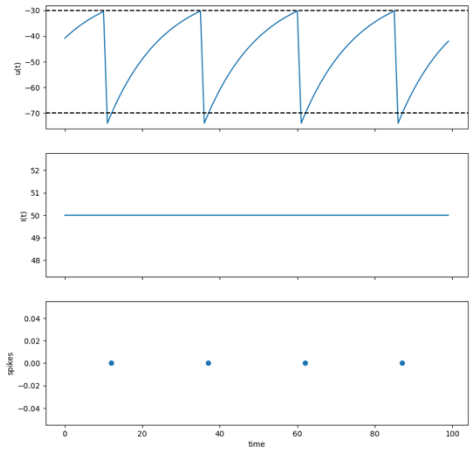
Using the Euler method to update the membrane dynamics in the LIF model, we apply a discrete approximation to the continuous dynamics of the neuron's membrane potential. The Euler update rule for the membrane potential `u` can be expressed as:

$$u(t+dt) = u(t) + \frac{-(u(t) - u_{rest}) + RI(t)}{\tau} dt$$

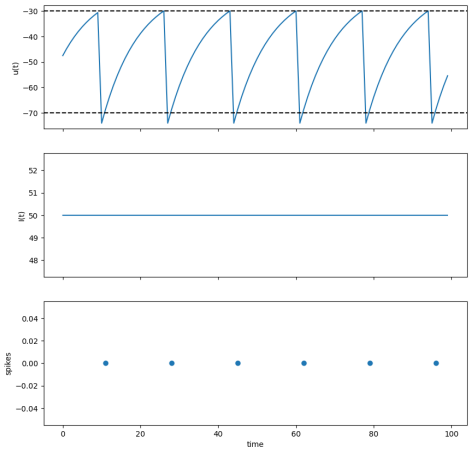
1-2) *Simulation (LIF)*

For simulating the LIF model under varying current stimulation, we begin by defining the network structure. This involves creating the network, setting its time resolution parameters, and defining the neuron groups. The behavior of the neuron groups is configured to follow the LIF dynamics, and the necessary parameters for this behavior are specified. Additionally, we set up recording tools such as a membrane potential recorder and an event recorder to monitor and plot

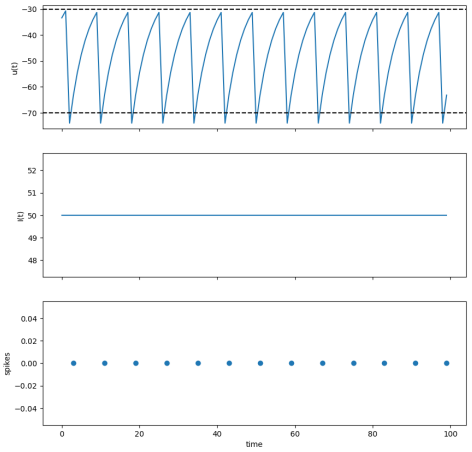
the membrane potential, the input current, and the spikes over time, serving as the output of the simulation.



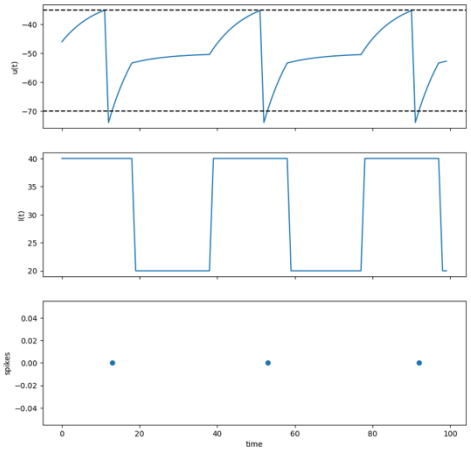
LIF Constant Current with tau = 15



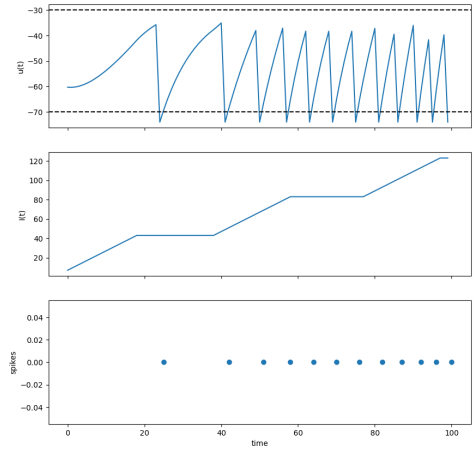
LIF Constant Current with tau = 10



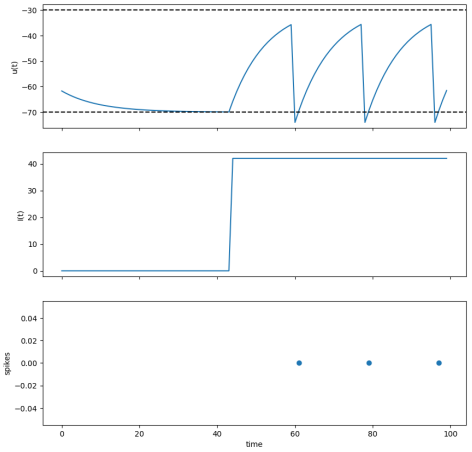
LIF Constant Current with tau = 5



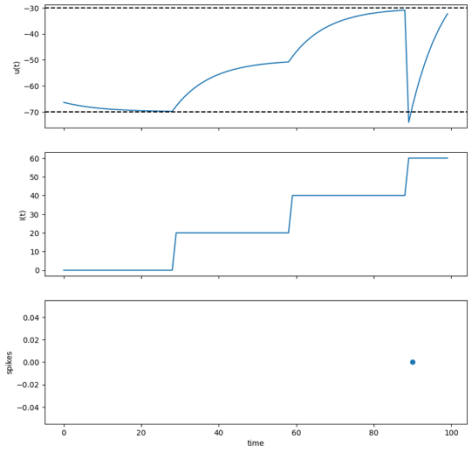
Square Wave Current LIF



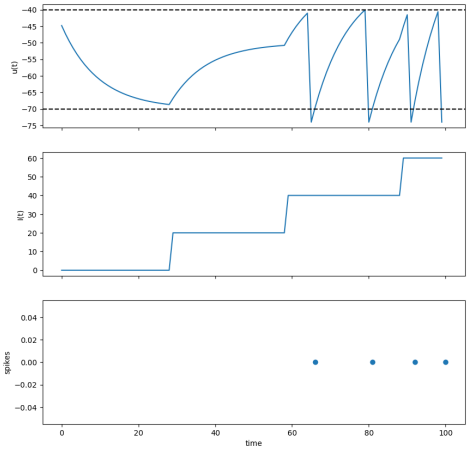
Step Current with Slope LIF



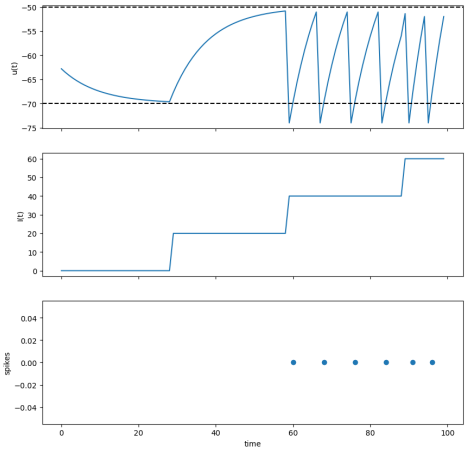
Unit Step Current LIF



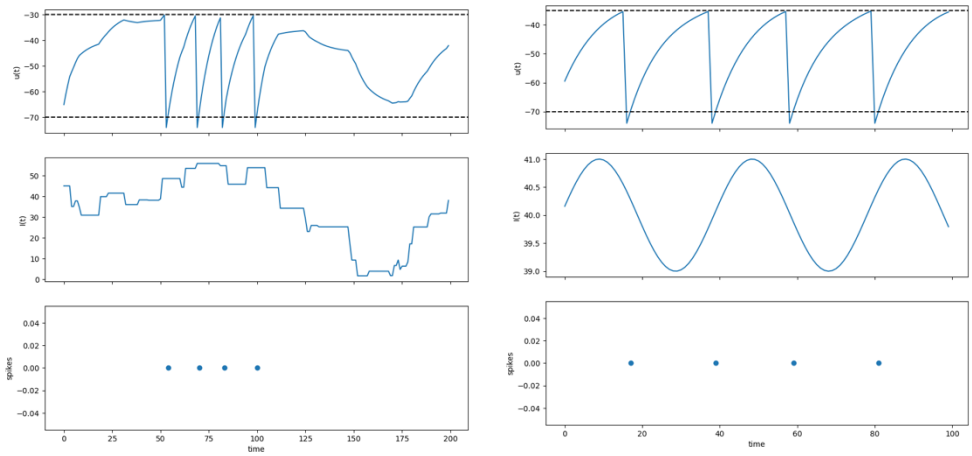
LIF Step Current with threshold=-30



LIF Step Current with threshold = -40



LIF Step Current with threshold = -50



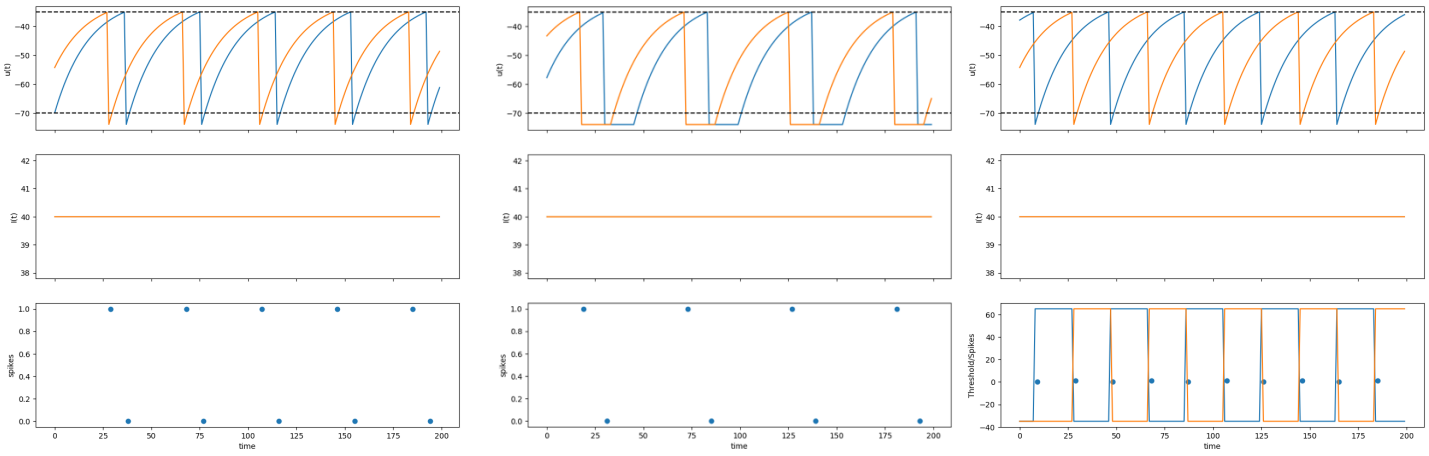
Current with Noise LIF *Sinusoidal Current LIF*

1-3) *Incorporating a Refractory Period*

To augment the LIF model with a refractory period, two implementations are possible:

1. **Time-Step Refractory**: Here, an additional parameter `rp` (refractory period) is established to dictate the number of time steps a neuron remains inactive following a spike. A tensor, `ng.rp`, is initialized to record refractory states, starting at 0. Upon firing, a neuron's `rp` count is set, decrementing with each time step. A neuron's potential is held constant at `u_rest` while `ng.rp` is above zero, signifying its refractory status. Upon reaching 0, the neuron becomes responsive once more.
2. **Threshold Modulation**: Alternatively, the adaptation could involve adjusting the neuron's activation threshold post-firing. Keeping the `rp` value, we also introduce a tensor to track individual neuron thresholds. When a neuron fires, its threshold is temporarily increased, effectively inhibiting subsequent spikes during the refractory period. Once `ng.rp` drops back to 0, the neuron's threshold is reverted to its original level, reinstating its ability to fire.

Both implementations serve the purpose of introducing a realistic pause in neural activity post-firing, which is a characteristic feature of actual neuronal behavior. Each method offers different computational implications and may be chosen based on the specific requirements of the simulation or computational efficiency considerations.



LIF without RP *LIF with RP method 1* *LIF with RP method 2*

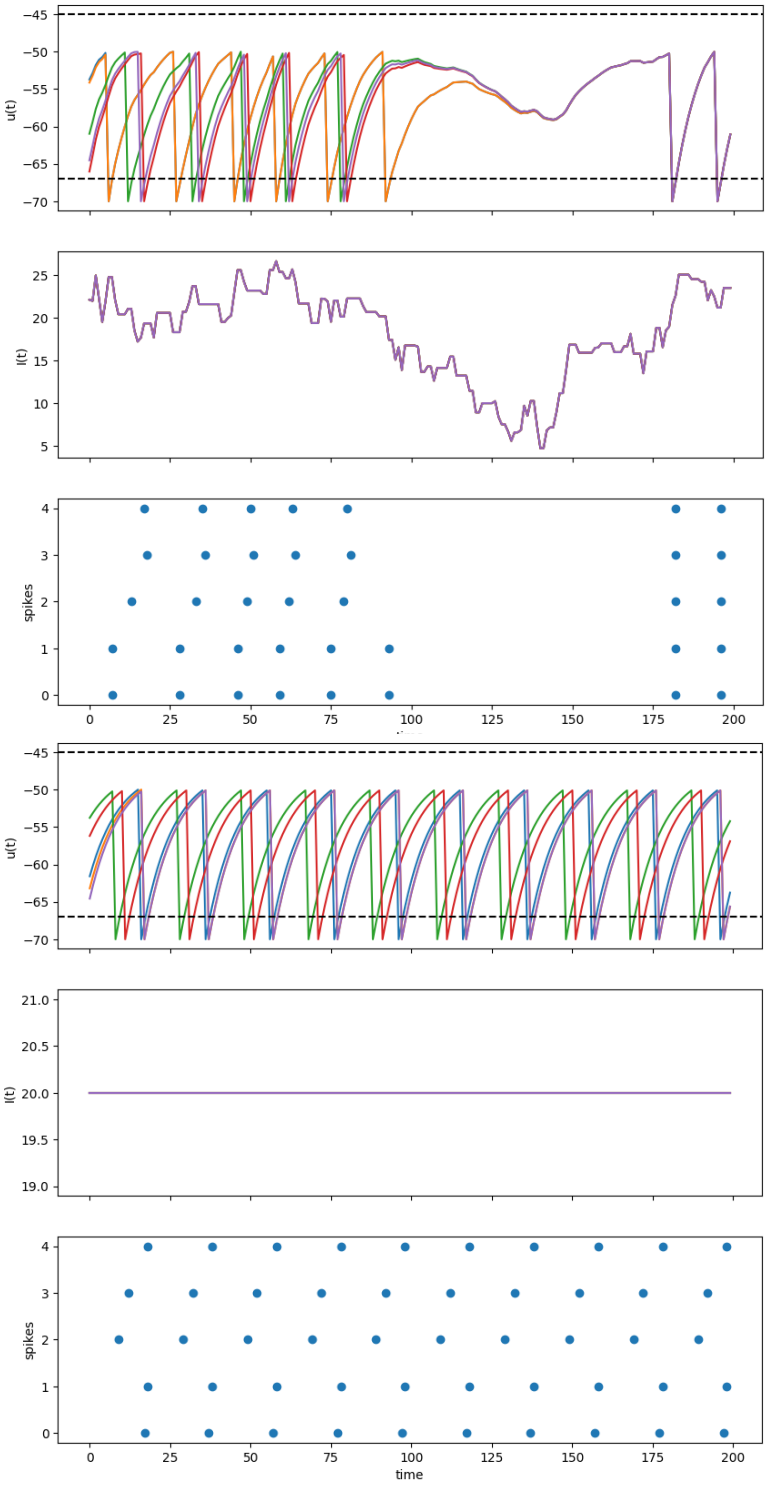
1-4) *F-I curve*

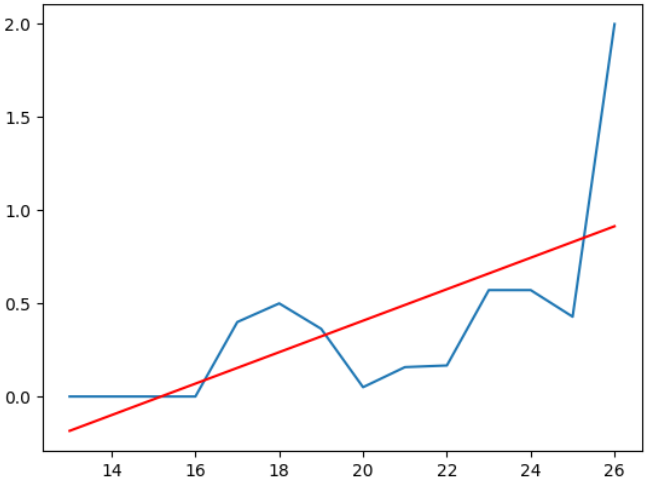
The Frequency Current (F-I) or gain function plot in neural models illustrates the relationship between the frequency of action potentials generated by a neuron and the amplitude of the input current. This plot is a fundamental tool used to characterize the input-output function of neurons and serves several purposes:

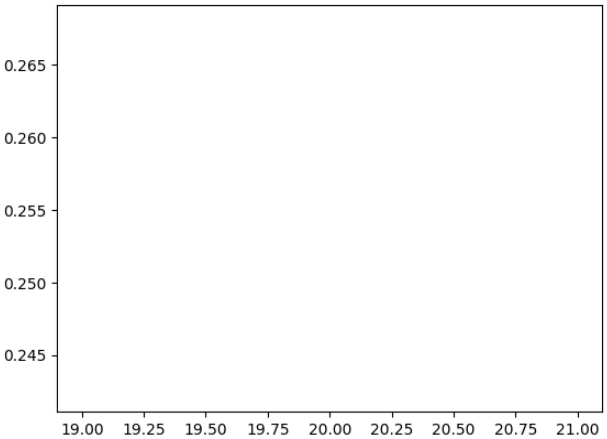
- Neural Responsiveness: It shows how sensitive a neuron is to varying levels of input current. Neurons can exhibit different thresholds for firing action potentials, and this plot helps to visualize that threshold.
- Operating Range: The F-I curve defines the range of input currents over which the neuron responds and demonstrates whether the response is linear, sub-linear, or supra-linear.
- Adaptation and Dynamics: Changes in the slope of the F-I curve indicate how a neuron's response adapts over time to constant or changing currents, which can reveal information about neuronal adaptation, accommodation, or other dynamic properties.
- Comparison Between Neurons: It allows for a comparative analysis of different neurons or neural models. By examining their F-I curves, one can compare their responsiveness and dynamical properties.
- Predicting Neural Behavior: It can predict how a neuron might respond to complex input patterns beyond constant currents, as the responsiveness to simple stimuli often correlates with responses to more complex ones.

To construct the Frequency-Current (F-I) plot, our initial step involves tallying the number of action potentials or spikes produced by the neuron for varying input currents. For each level of input current, we ascertain the number of spikes generated within a set period. Subsequently, we calculate the average firing frequency by dividing the total spike count by the time interval. This provides us with a frequency associated with each specific input current. Once the data points are

plotted, a trend line can be overlaid onto the graph to succinctly illustrate the general relationship between the input current and the neuron's firing rate. Below the plot for LIF model can be seen:







2) Exponential Leaky Integrate-and-Fire (ELIF) Model

The Exponential Leaky Integrate-and-Fire (ELIF) model is an extension of the traditional Leaky Integrate-and-Fire (LIF) model, adding an exponential term to the membrane potential equation to capture the rapid upswing of action potentials, which is observed in real neurons but not represented in the original LIF model. In the ELIF model, the membrane potential increases linearly as a current is injected, just like in LIF, but as it approaches a certain threshold, the exponential term becomes significant, causing a rapid rise in voltage that culminates in a spike.

One advantage of the ELIF model over the LIF model is its ability to better mimic the real neural firing patterns, specifically the sharp spike upswing and the variety of firing patterns observed in biological neurons. This makes the ELIF model a more realistic and powerful tool for simulating neuronal behavior, especially when the precise dynamics of spike initiation are important for the simulation's goals.

However, the increased complexity of the ELIF model is also its disadvantage; it requires the tuning of more parameters, such as the steepness of the exponential term, making it more computationally intensive than the LIF model. Furthermore, although more accurate, the benefits of this additional accuracy must be weighed against the added complexity, as for many applications, the LIF model's simplicity and lower computational cost may be sufficiently accurate and more desirable.

2-1) *Implementation of the ELIF Model*

In implementing the ELIF model, an `initialization` function takes several key parameters from the input, including resistance (`R`), membrane time constant (`tau`), threshold potential (`threshold`), firing threshold (`theta_th`), sharpness parameter (`sharpness`), resting potential (`u_rest`), and reset potential (`u_reset`). Neurons in the group are then assigned an initial potential uniformly distributed between the threshold and the reset potential.

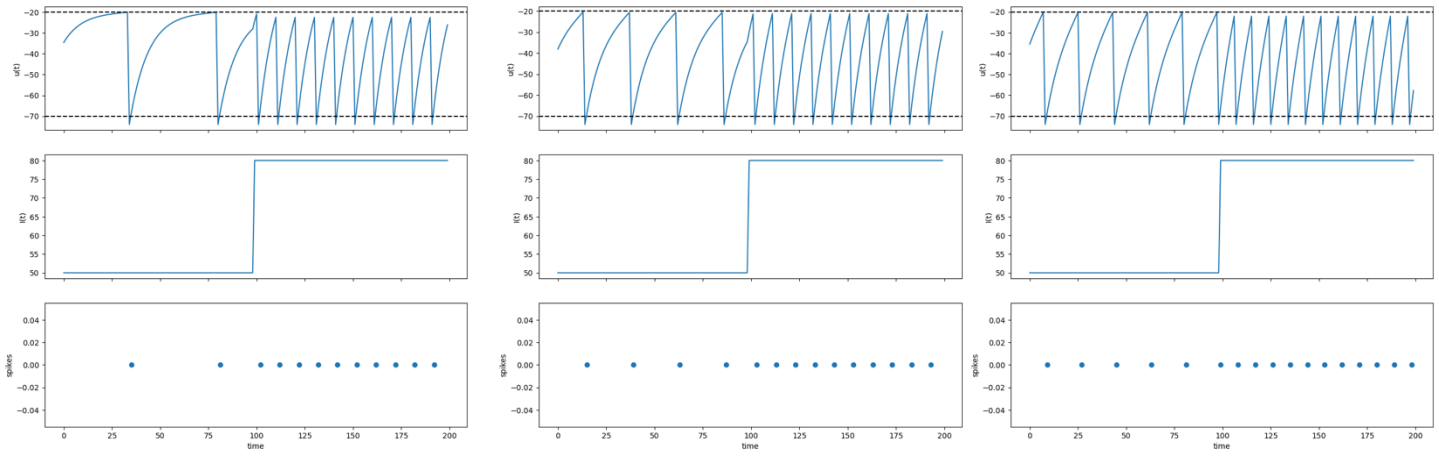
The `forward` function of the neuron group is tasked with encapsulating the dynamics of the model, which include membrane potential evolution, spike firing, and the reset process. The membrane potential (`ng.u`) is updated following the Euler method, after which a check is conducted to ascertain if any neuron has surpassed the preset threshold. Neurons reaching this mark have their potentials reset to `u_reset`.

Using the Euler method to update the membrane dynamics in the ELIF model, we apply a discrete approximation to the continuous dynamics of the neuron's membrane potential. The Euler update rule for the membrane potential `u` can be expressed as:

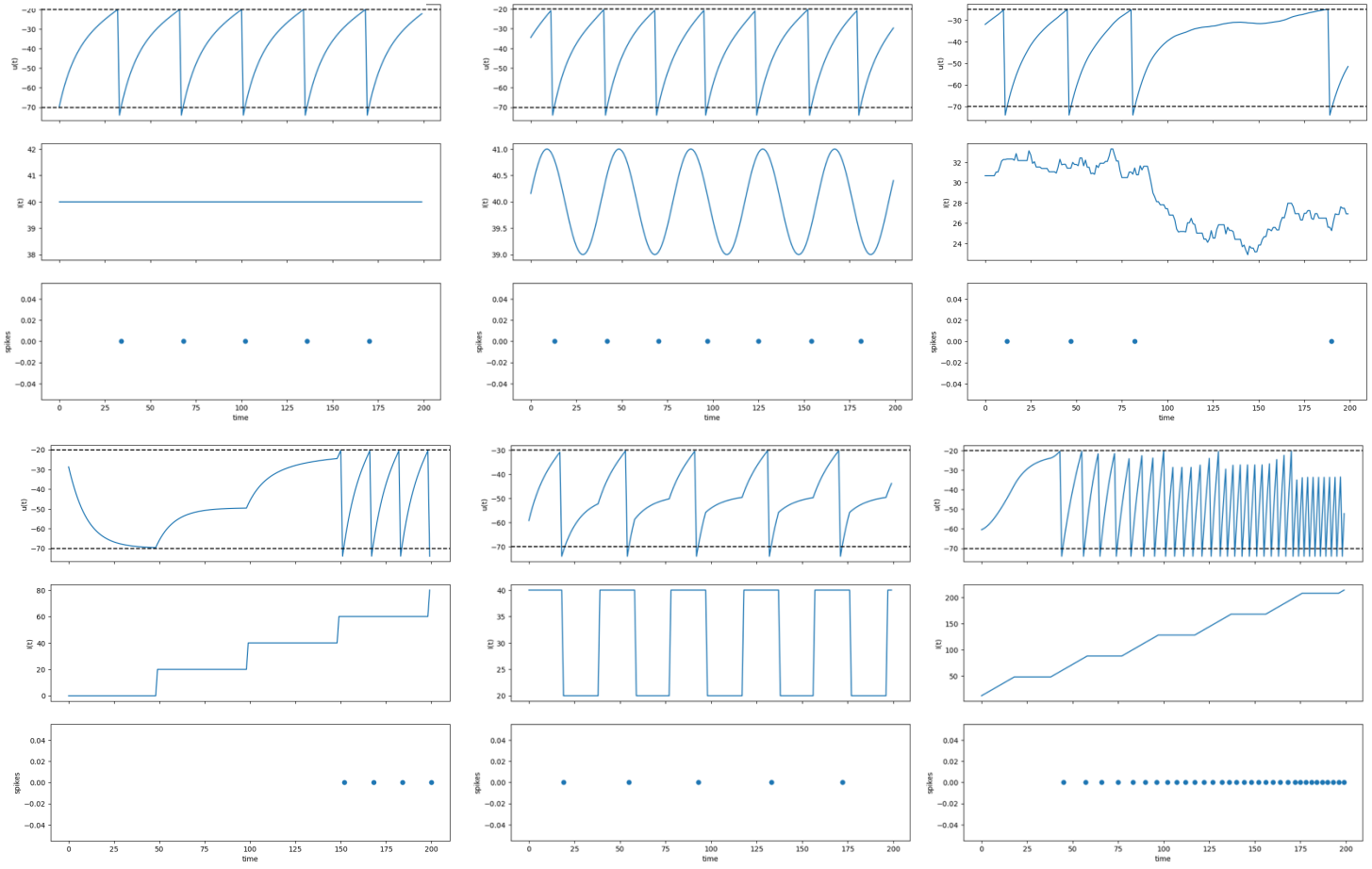
$$u(t+dt) = u(t) + \frac{-(u(t) - u_{rest}) + RI(t) + \Delta_T e^{\frac{u(t)-\theta_{rh}}{\Delta_T}}}{\tau_m} dt$$

τ_m is the membrane time constant, V_{rest} is the resting membrane potential. R is the membrane resi stance. I is the input current. ϑ_{rh} is the rheobase threshold, representing the membrane potential at which adaptation begins. Δ is a parameter that controls the sharpness.

2-2) *Simulation (ELIF)*



ELIF with sharpness = 1 *ELIF with sharpness = 10* *ELIF with sharpness = 20*

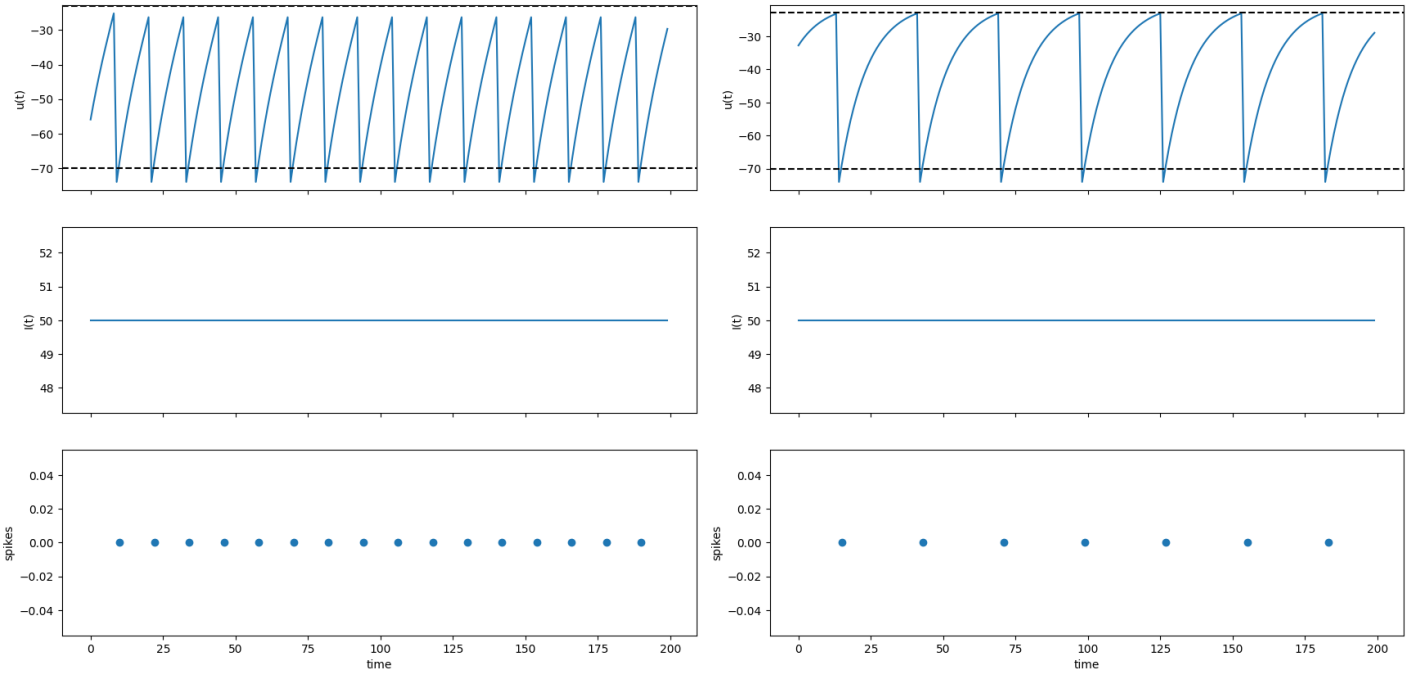


For simulating the ELIF model under varying current stimulation, we begin by defining the network structure. This involves creating the network, setting its time resolution parameters, and defining the neuron groups. The behavior of the neuron groups is configured to follow the LIF

dynamics, and the necessary parameters for this behavior are specified. Additionally, we set up recording tools such as a membrane potential recorder and an event recorder to monitor and plot.

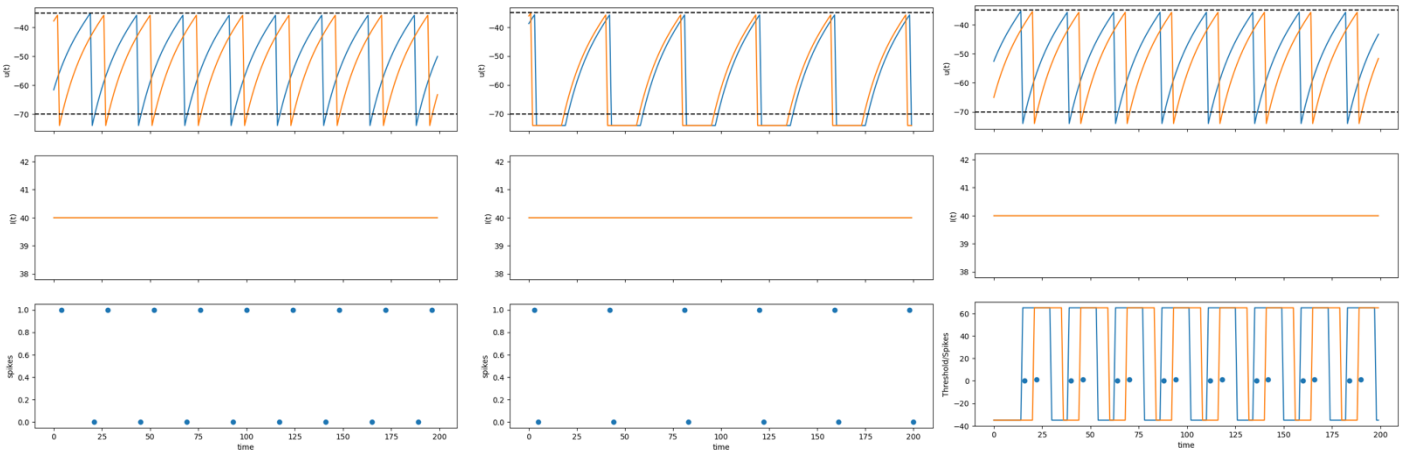
When the sharpness parameter is increased, the exponential component of the membrane potential equation becomes steeper. This means that for a given input current, it takes a longer time for the membrane potential to reach the threshold for firing. The increased steepness makes the membrane less sensitive to small changes in input, requiring either a stronger input or a longer time to accumulate enough voltage to trigger the action potential.

Therefore, a higher sharpness parameter leads to the neuron requiring a longer time to integrate inputs before reaching the threshold, increasing the Inter-Spike Interval (ISI). ISI is the time between consecutive spikes, and a longer ISI implies a lower firing rate. The first row of plots clearly demonstrates that an increase in the sharpness parameter of the exponential LIF model is correlated with an increase in the Inter-Spike Interval (ISI).



The images above illustrate that, when identical parameters in both LIF and ELIF models are held constant, the inclusion of the exponential term and associated sharpness parameter in the ELIF model results in a higher firing frequency compared to LIF. This is because the exponential component enables the neuron to reach the firing threshold more rapidly.

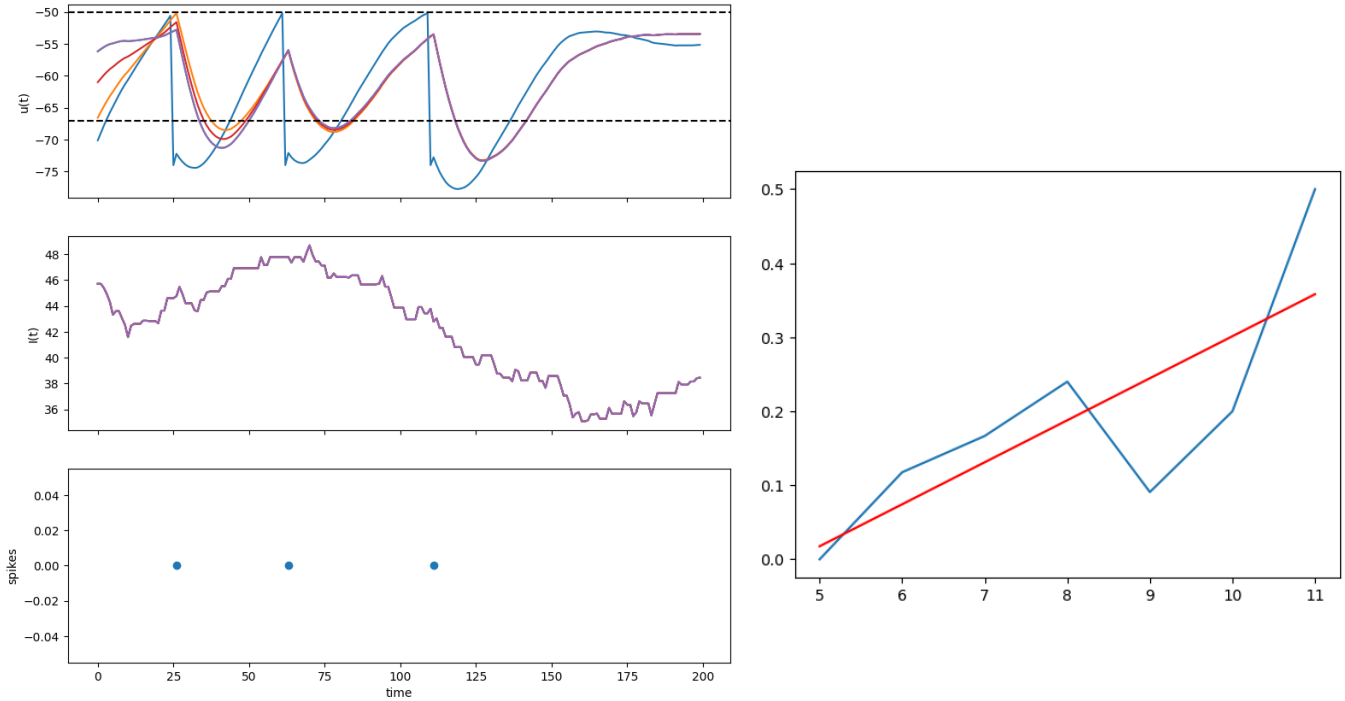
2-3) *Incorporating a Refractory Period*



ELIF without RP | *ELIF with RP method 1* | *ELIF with RP method 2*

2-4) *F-I Curve*

As explained in the previous part, to construct the Frequency-Current (F-I) plot, our initial step involves tallying the number of action potentials or spikes produced by the neuron for varying input currents. For each level of input current, we ascertain the number of spikes generated within a set period. Subsequently, we calculate the average firing frequency by dividing the total spike count by the time interval. This provides us with a frequency associated with each specific input current. Once the data points are plotted, a trend line can be overlaid onto the graph to succinctly illustrate the general relationship between the input current and the neuron's firing rate. Below the plot for ELIF model can be seen:



3) Adaptive Exponential Leaky Integrate-and-Fire (AELIF) Model

The Adaptive Exponential Leaky Integrate-and-Fire (AELIF) model is an advanced neuron model that incorporates an adaptation mechanism into the Exponential Leaky Integrate-and-Fire (ELIF) framework. The AELIF model adds a crucial feature: a spike-triggered adaptation current that increases with each action potential, making subsequent firing less likely unless the stimulus is strong enough to overcome this current. This mimics the observed behavior of real neurons that show spike frequency adaptation over time.

One significant advantage of the AELIF model over the classic LIF and ELIF models is its ability to reproduce a wider range of neuronal firing patterns observed in the biological neurons, such as adaptation and bursting. This makes the AELIF model more biologically realistic and versatile in simulating neural dynamics.

However, the inclusion of additional parameters for adaptation also introduces disadvantages. The AELIF model is computationally more complex and expensive to simulate than LIF and ELIF models. This complexity can be a drawback when modeling large networks of neurons, as it requires more computational resources. Additionally, the increased number of parameters can make the model more challenging to fit to experimental data, as it adds to the complexity of the parameter space that must be explored to accurately capture neuron behavior.

3-1) Implementation of the AELIF Model

In implementing the AELIF model, an `initialization` function takes several key parameters from the input, including resistance (`R`), membrane time constant (`tau_m`), threshold potential (`threshold`), firing threshold (`theta_th`), sharpness parameter (`sharpness`), resting potential (`u_rest`), and reset potential (`u_reset`). The adaptation variable (w) is updated based on the neuron's activity and the adaptation parameters (`a`, `b`, and `tau_w`).

The `forward` function of the neuron group is tasked with encapsulating the dynamics of the model, which include membrane potential evolution, spike firing, and the reset process. The membrane potential (`ng.u`) is updated following the Euler method, after which a check is conducted to ascertain if any neuron has surpassed the preset threshold. Neurons reaching this mark have their potentials reset to `u_reset`.

Using the Euler method to update the membrane dynamics in the ELIF model, we apply a discrete approximation to the continuous dynamics of the neuron's membrane potential. The Euler update rule for the membrane potential `u` can be expressed as:

$$u(t+dt) = u(t) + \frac{-(u(t)-u_{rest}) + RI(t) - Rw + \Delta_T e^{\frac{u(t)-\theta_{rh}}{\Delta_T}}}{\tau_m} dt$$

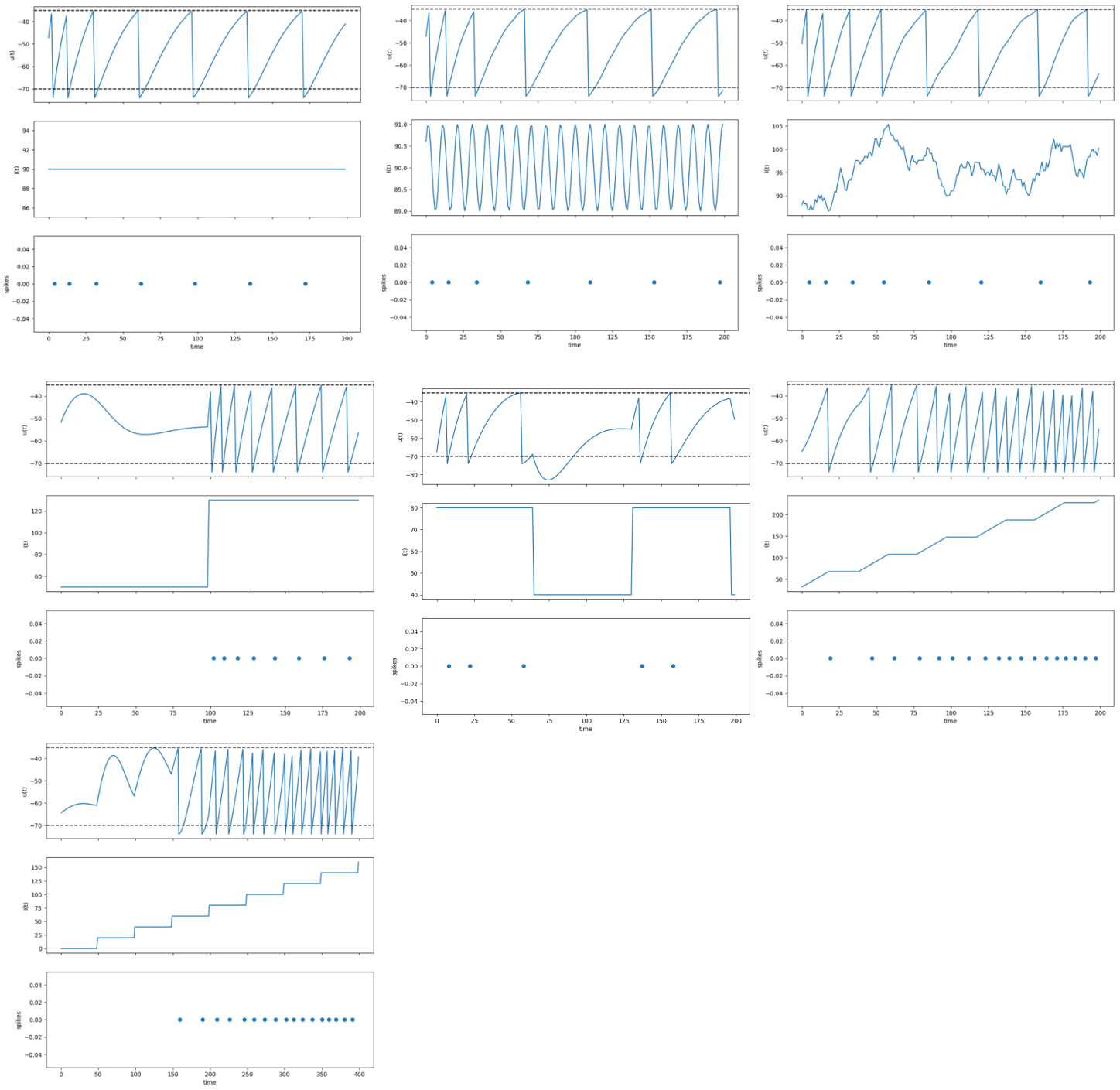
$$w(t+dt) = u + \frac{a(u-u_{rest}) - w(t) + b\tau_w \sum \delta(t-t^f)}{\tau_w} dt$$

In the Adaptive Exponential Leaky Integrate-and-Fire (AELIF) model, the following parameters are used:

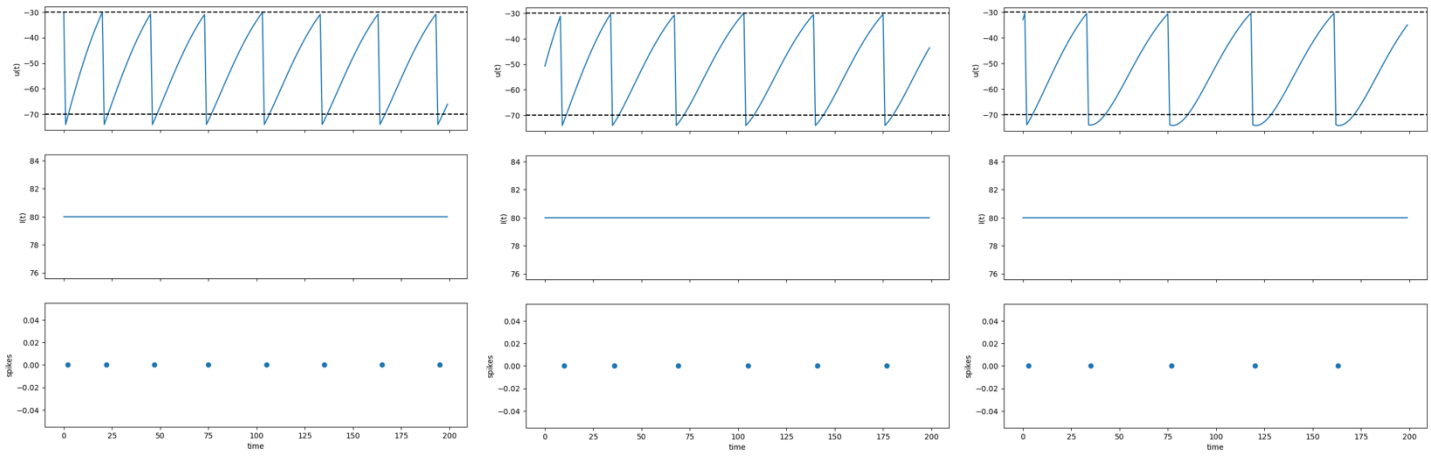
- `tau_m` (Membrane Time Constant): This parameter represents the time it takes for the membrane potential to either charge or discharge through the leaky component of the model. It is a measure of the membrane's resistance to change in voltage and is crucial in determining the rate of decay of the membrane potential towards the resting potential.
- `tau_w` (Adaptation Time Constant): The adaptation time constant dictates how quickly the adaptation variable, which controls spike-frequency adaptation, decays back to its resting state after a spike. This parameter influences the adaptation dynamics and recovery rate post-spike.
- `threshold`: This is the membrane potential value at which the neuron fires an action potential. When the membrane potential exceeds this threshold, the neuron generates a spike.
- `theta_rh` (Rheobase Threshold): This is the critical threshold near the resting potential where the exponential increase in membrane potential begins. It is a parameter unique to ELIF and AELIF models, illustrating the exponential sensitivity of the neuron to depolarization.
- `u_reset` (Reset Potential): After a neuron fires an action potential, the membrane potential is reset to this value. It is an essential parameter for controlling the refractory behavior of the neuron post-spike.
- `u_rest` (Resting Potential): The resting potential is the membrane potential value of the neuron in a stable, unstimulated state. It serves as a baseline from which depolarization or hyperpolarization occurs.
- `R` (Membrane Resistance): This parameter is the resistance of the neuron's membrane which, when combined with the membrane capacitance, contributes to the membrane time constant and affects the charging and discharging rate of the membrane potential.
- `I` (Injected Current): It represents external current injected into the neuron. This current is a key factor in driving the membrane potential towards the firing threshold.
- `Sharpness` Parameter: In the AELIF model, the sharpness parameter determines the steepness of the exponential component of the membrane potential equation, directly affecting the model's capacity to respond to rapid changes in the input current.
- `a` (Subthreshold Adaptation): This parameter controls the strength of the coupling between the membrane potential and adaptation mechanisms. A higher value represents stronger subthreshold adaptation and can lead to accommodation where the neuron becomes less responsive to input.
- `b` (Spike-Triggered Adaptation): The spike-triggered adaptation parameter b influences the increment of the adaptation variable with each spike. It modulates how much adaptation is induced by firing an action potential, influencing after-spike behavior like spike-frequency adaptation and refractoriness.

Each of these parameters interplays to define the diverse firing patterns possible within the AELIF model, contributing to its capability to emulate complex behaviors seen in biological neurons.

3-2) Simulation (AELIF)

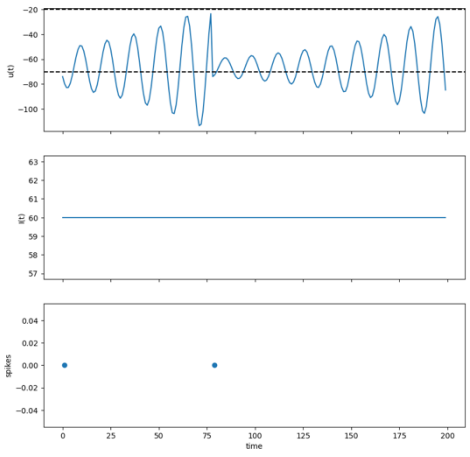


In nearly all the presented graphs, it is observable that the inter-spike interval (ISI) tends to rise when the applied current remains constant or changes only slightly. However, in cases where there is a significant increase in the current value, the spike frequency correspondingly escalates with the augmentation of the current.



The above figures display plots with the adaptation parameter b set to 10, 20, and 40, respectively, from left to right. It is apparent that an increase in the b parameter is correlated with a reduction in the number of spikes generated.

This phenomenon occurs because in the AELIF model, the `b` parameter determines the level of adaptation: it is a conductance that governs the subthreshold adaptation dynamics of the membrane. When b is increased, the neuron's membrane potential is subjected to a greater adaptation current after each spike. This adaptation acts as a negative feedback mechanism, making it progressively harder for the neuron to reach the threshold necessary for firing subsequent spikes. Therefore, as b increases, the neuron's excitability decreases, leading to fewer spikes over time for the same input current.

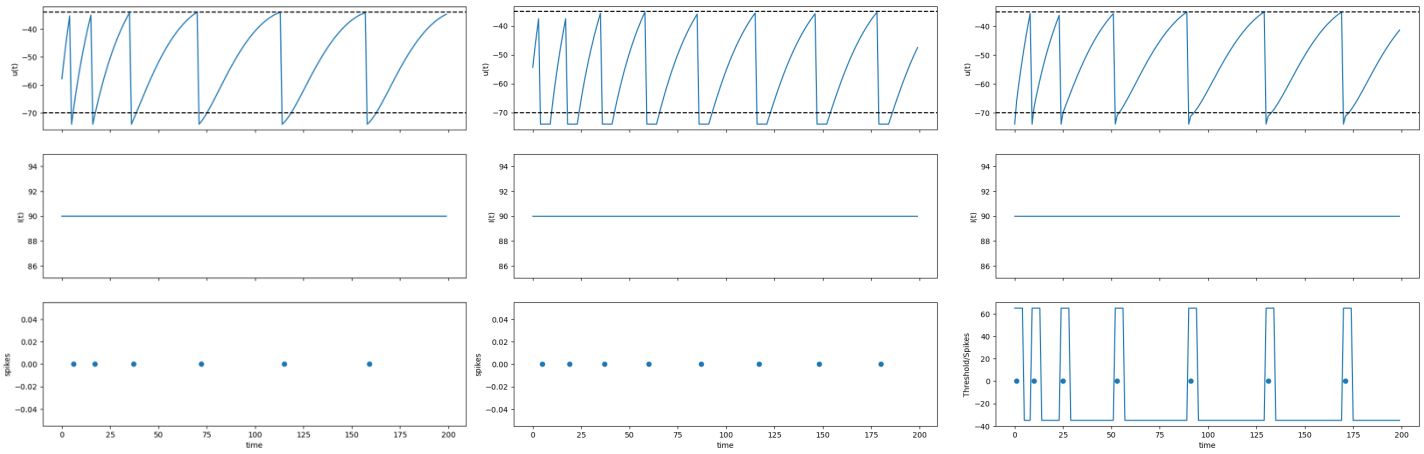


As the above plot suggests, upon altering the parameter `a` in the presence of a constant input current, the resulting potential-time plot reveals that oscillations are induced in the membrane potential. The amplitude of these oscillations progressively increases over time until it reaches the

spiking threshold. Following the spike, the membrane potential abruptly drops to its lowest value before starting to rise again, potentially leading to another spike.

This behavior is attributable to the role of the parameter `a`, which in the AELIF model governs the subthreshold resonance of the neuron. Specifically, `a` represents the coupling between the membrane potential and the adaptation current. When `a` is modified, it alters the interaction between the rate of change of the membrane potential and the adaptation mechanism. As a result, higher values of `a` facilitate the generation of membrane potential oscillations. These oscillations grow in amplitude with time as the neuron integrates the continuous input current, eventually reaching the threshold for spiking. After a spike occurs, the membrane potential is reset, leading to a drop in potential, but because the input current remains constant, the cycle repeats, and the oscillatory behavior continues, which may result in subsequent spikes. This feedback between the membrane potential and the adaptation current under constant input creates a sort of resonance effect that leads to growing oscillations until the neuron fires.

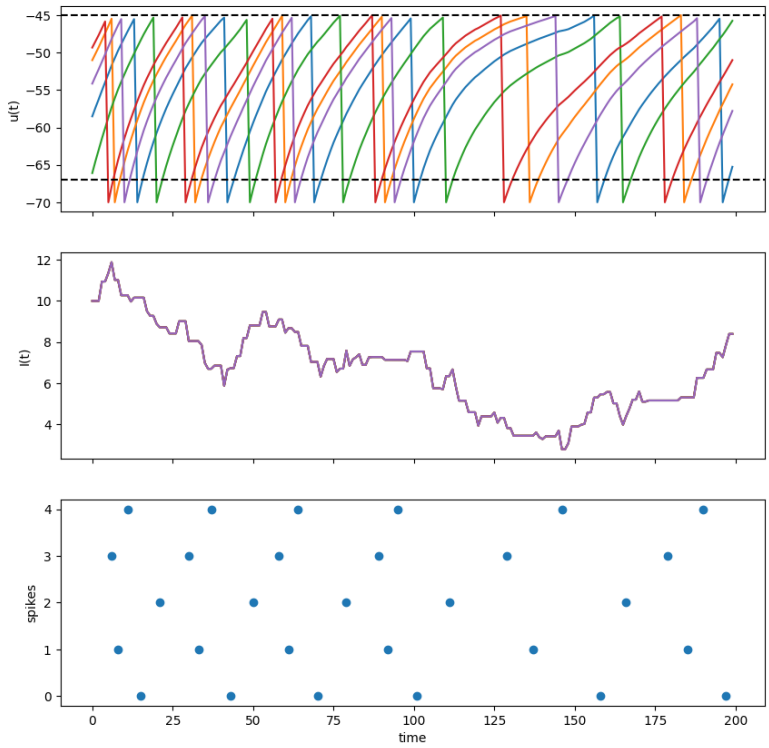
3-3) Incorporating a Refractory Period

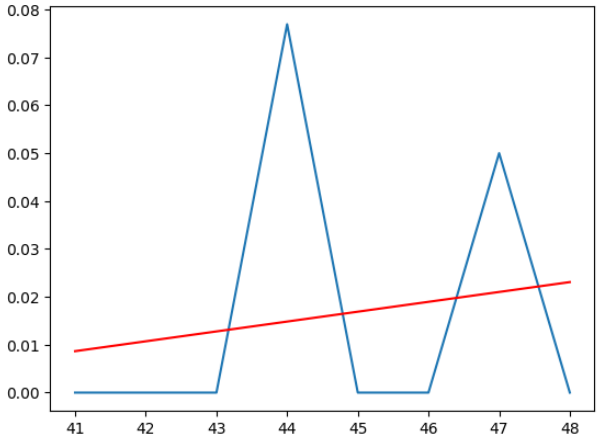


AELIF without RP *AELIF with RP method 1* *AELIF with RP method 2*

3-4) F-I Curve

As explained in the previous part, we construct the Frequency-Current (F-I) plot. Below the plot for AELIF model can be seen:





Aside from the frequency-current (F-I) curve, there are several other metrics that can provide important indices of neuronal function when analyzing neuronal models. For example:

1. **Phase Response Curve (PRC):** The PRC quantifies how a brief input shifts the phase of a neuron's firing cycle. A PRC plot usually has the cycle phase along the x-axis and the phase shift caused by the stimulus along the y-axis. It helps to predict the degree of neural synchrony and phase locking in a network.
2. **Burst Analysis**: Key metrics here include the burst duration (time span of a burst), the number of spikes within a burst, the average frequency within a burst, and the interburst interval, providing a clear view of the neuron's bursting dynamics.
3. **Spike Latency**: Measured as the time from stimulus onset to first spike, latency variability can signal changes in synaptic efficacy or membrane excitability. It measures the time delay between the application of a current and the neuron's first response, often recorded as the first spike. It indicates how quickly a neuron can respond to stimuli.